Interactive Advertising Bureau
www.iabspain.net

Interfaz de Video Player-Ad

Video Player-Ad Interfaz Definition; VPAID (Versión 2.0)

Comisión de Vídeo publicitario - IAB Spain
Marzo de 2011

Este documento ha sido desarrollado por la Comisión de Vídeo publicitario de IAB Spain. Se trata de una adaptación del documento de IAB USA en su versión 1.1.

Acerca de la Comisión de Vídeo publicitario de IAB Spain:

La Comisión de Vídeo de IAB Spain se formó en noviembre de 2010 con el fin de crear un conjunto de guías, medidas y opciones creativas para la publicidad interactiva de vídeo. La Comisión trabaja para educar a los profesionales del marketing y agencias en las fortalezas del vídeo digital, al tiempo que tiene como misión la estandarización de las prácticas del sector.
Las empresas integrantes de la subcomisión técnica responsable del presente documento han sido: Atres Advertising, Eyewonder, Google, Kewego, MediaMind, Publimedia Gestión, Publipress Media, Smartclip, Telefónica, Vivaki y Weborama.

ÍNDICE

ÍNDICE

ÍNDICE

1. RESUMEN EJECUTIVO

1. RESUMEN EJECUTIVO

El presente documento establece una definición de Vídeo Player-Ad API (VPAID) que estandariza la comunicación entre reproductores de vídeo y publicidad en vídeo in-stream, de acuerdo con el diseño del Comité de Vídeo Digital de Interactive Advertising Bureau USA (IAB) y aprobado por la Comisión de Vídeo publicitario de IAB Spain.
Este estándar pretende dar a conocer las necesidades de los formatos emergentes de publicidad *in-steam*, como por ejemplo:

- Publicidad no lineal en vídeo.
- Publicidad interactiva en vídeo.

El objetivo del estándar VPAID es solucionar los problemas de interoperabilidad conocidos entre los reproductores de vídeo de los soportes y los diferentes *ad servers* existentes. Hoy en día, la publicidad en vídeo de una plataforma publicitaria específica solo puede funcionar en reproductores de vídeo del soporte que están preintegrados en la misma plataforma de contenido. El problema es aún más grave cuando la publicidad en vídeo se expresa en formatos innovadores (como la publicidad no lineal e interactiva) que requiere un alto nivel de comunicación e interacción entre la publicidad y el reproductor de vídeo.

Este estándar no es el único existente. Los soportes y las redes publicitarias que deseen participar en un entorno más transparente de publicidad en vídeo deben adoptar también los siguientes estándares:

- Estándares de formatos publicitarios interactivos; Formatos de Vídeo

- Estándar para la entrega de publicidad en vídeo digital. VAST 2.0

Un soporte que adopte VPAID y los estándares mencionados más arriba será capaz de reproducir cualquier tipo de publicidad en vídeo desde cualquier *ad server* que esté adherido a los mismos estándares.

Mientras el propósito de este documento es promover la interoperabilidad en las áreas más comunes del panorama del vídeo digital, IAB continúa promoviendo la creatividad y la innovación en los formatos de publicidad en vídeo. Como con todas las directrices de IAB, este documento será actualizado tan pronto como el dinámico panorama de la publicidad en vídeo digital progrese y se adopten nuevos formatos publicitarios.

2. MOTIVACIÓN

2. MOTIVACIÓN

La ausencia de estándares tecnológicos en la publicidad en vídeo ha sido una preocupación creciente para IAB. Debido a la falta de marcos tecnológicos comunes, la implementación de la publicidad en vídeo es a menudo un asunto de difícil coordinación que redunda en un perjuicio para la publicidad de vídeo en su conjunto. Los anunciantes que desean utilizar los formatos de publicidad más recientes pueden verse frustrados porque sus inversiones en creatividad para una campaña a menudo solo funcionan en un reproductor de vídeo en combinación con un *ad server* específico. Esto dificulta la capacidad general de la industria de adoptar formatos de publicidad innovadores que provean mejor ROI a los anunciantes y posibiliten una experiencia menos intrusiva hacia los espectadores del contenido del vídeo. El resultado final es que los anunciantes vuelven a los formatos tradicionales (como el *pre-roll*) que en la actualidad pueden ser superados por formatos más avanzados. Los soportes también se muestran reacios a aceptar integraciones de los diferentes *ad servers* ya que cada proyecto de integración es específico para cada plataforma y no es escalable. Con la estandarización de un API, IAB espera afrontar las siguientes ineficiencias del mercado para los soportes, anunciantes y redes publicitarias:

- Falta de tecnología común de *ad servers* que emitan vídeo, provocando que los soportes no acepten formatos publicitarios de terceros.
- Falta de estándares tecnológicos comunes que incrementan el coste de la producción.
- Falta de agilidad en el suministro de publicidad en formato vídeo, aumentando los costes de integración con cada editor.

3. INTRODUCCIÓN

3. INTRODUCCIÓN

La publicidad en video *In-stream*, a diferencia de otras formas de visualización o *banners*, se ejecuta en el entorno del reproductor. El reproductor es responsable de hacer la llamada del anuncio e interpretar la respuesta desde el *ad server*. El reproductor también proporciona el entorno de tiempo de ejecución para el anuncio y controla su ciclo de vida.

Con la llegada de publicidad en vídeo avanzada, las relaciones entre el reproductor y la publicidad se hacen más complejas. Los formatos de publicidad avanzados proporcionan una experiencia de publicidad que intercala partes de anuncio con el contenido, de forma que se requiere una orquestación cuidadosa del contenido y de la reproducción del anuncio. El anuncio en sí puede ejecutar *scripts* y tiene parámetros variables en función de la interactividad del usuario y otros contextos de tiempo de ejecución. La interacción del reproductor, el contenido del vídeo y la publicidad requiere ciertas aptitudes a través de un API desde el momento en que se ejecuta un vídeo en el reproductor.

Permitir la interoperabilidad de la publicidad avanzada de vídeo a través de diferentes entornos de reproducción requiere, como mínimo, una **estandarización en los dos niveles siguientes**:

1. La respuesta del *ad server* debe ser entendida por reproductores compatibles (esto queda garantizado aplicando el estándar VAST)
2. El anuncio avanzado (anuncio interactivo) requiere un tiempo de ejecución del reproductor estándar, expresado como un API estándar (detallado en este documento y denominado VPAID).

El diagrama siguiente representa el flujo básico de la publicidad en vídeo y el punto de interfaz donde el VPAID es aplicable.

3. INTRODUCCIÓN

Figura 1: Flujo básico de la publicidad en vídeo

El flujo de la publicidad en vídeo requiere de los pasos siguientes (nótese que hay muchos otros pormenores y variaciones, pero esta descripción se ha configurado de forma sencilla de manera intencionada):

1. El reproductor hace una llamada de anuncio al ad server. IAB no ha publicado un estándar alrededor de la llamada del anuncio (*aunque el estándar **VAST** tiene un apéndice que recomienda parámetros para la publicidad en vídeo*).

2. El ad server responde con un ad XML. El estándar IAB VAST cubre el formato de vídeo de la respuesta del servidor de anuncio.

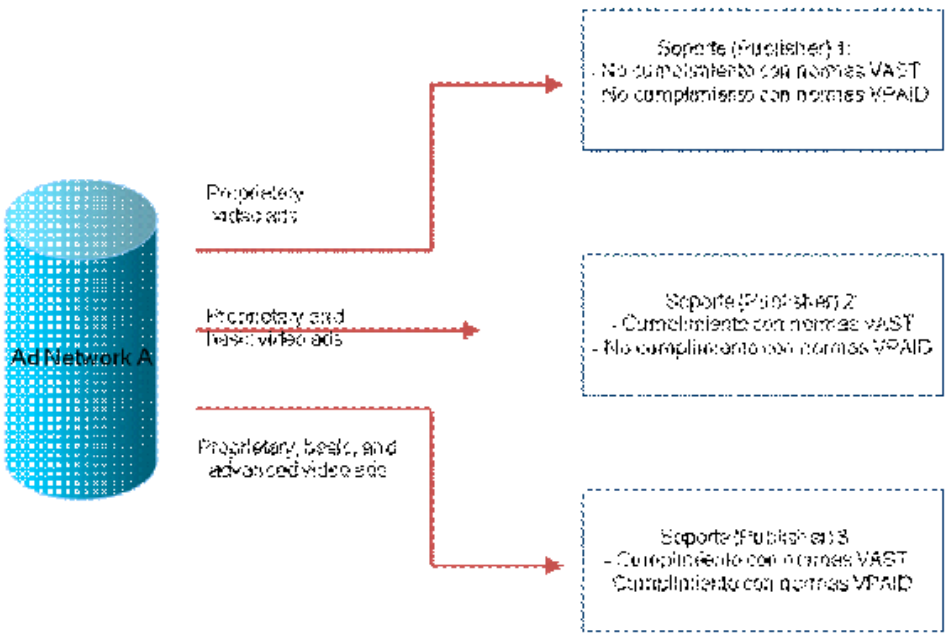
3. El reproductor recupera los parámetros de la publicidad mencionada en el ad XML y los traduce. Esta publicidad requiere un conjunto de parámetros del reproductor, y puede interactuar con el reproductor usando un API. Este estándar cubre el reproductor y la interacción de anuncio.

4. El reproductor o anuncio emite una impresión y registra la actividad (*IAB ha publicado Directrices sobre las Métricas Publicitarias en Vídeo In-Stream que cubren los aspectos de medición de la publicidad en vídeo*).

5. La renderización de anuncios, comportamiento, y las especificaciones de formatos se cubre en los Estándares de formatos publicitarios interactivos: Formatos de Vídeo.

Este estándar define un entorno de tiempo de ejecución uniforme para la publicidad de vídeo avanzada, de modo que un reproductor compatible pueda reconocer cualquier anuncio compatible de cualquier *ad server*, posibilitando así la interoperabilidad de publicidad de vídeo entre los soportes. El diagrama siguiente representa el nivel de interoperabilidad de publicidad en vídeo a través de soportes con diferente nivel de conformidad de estándares.

Figura 2: Interoperabilidad de publicidad en video a través de soportes con diferente nivel de cumplimiento de las normas



(Los 3 soportes mostrados están preintegrados con tecnología de renderización de publicidad en vídeo de la Red Publicitaria A)

Los soportes compatibles con los estándares VAST y VPAID serán capaces de aceptar mayor variedad de publicidad en vídeo de terceros *ad servers* y redes publicitarias que sean también compatibles.

VPAID es más útil (y sobre todo necesario) para anuncios que son mostrados simultáneamente con el contenido del vídeo, como en la publicidad *overlay*, o para anuncios que pueden interrumpir la reproducción del contenido del vídeo como resultado de la interacción del usuario con el anuncio, como con los anuncios interactivos. Mientras que la publicidad en vídeo básica *pre-roll*, *mid-roll* o *post-roll* no necesita usar un API, la presencia de este marco de trabajo común permite que se puedan desarrollar nuevos formatos publicitarios. Además, los formatos *pre-roll* y *post-roll* pueden ser extendidos para tener una sección que sea presentada durante la reproducción del contenido del vídeo sin cambiar el *framework* de Player-Ad.

API no requerido	API requerido
Vídeo *pre-roll*	*Pre-roll* que ejecuta el *script*
Vídeo *pre-roll* con *banner* acompañante	Clic a *overlay* interactivo
Imagen u *overlay* simple creativo	

3.1. ALCANCE Y LIMITACIONES

Este estándar sólo cubre la interacción entre el tiempo de ejecución del reproductor y una publicidad en vídeo que ejecute el *script*, como por ejemplo una construida usando la tecnología Adobe Flash. Es más, este estándar sólo cubre la interoperabilidad entre los anuncios y el reproductor escrito usando la misma oferta tecnológica. Por ejemplo, un reproductor compatible escrito usando la tecnología Flash ActionScript 3 será capaz de aceptar cualquier anuncio compatible escrito en Flash ActionScript 3, pero puede que no sea capaz de aceptar anuncios compatibles escritos usando la tecnología Silverlight y viceversa.

El mecanismo que empareja la tecnología de un anuncio compatible con la tecnología de un reproductor compatible, operacionalmente o durante el período de publicación del anuncio, está fuera del alcance de este documento.
Este documento está diseñado para cualquier reproductor de vídeo bajo demanda. Se puede utilizar esta especificación para entregas de publicidad en aplicaciones distintas que las vistas bajo demanda como vídeo *streaming* en vivo, reproductores de vídeo descargables, decodificadores, etc., pero esas aplicaciones están más allá del alcance del actual documento.

Aunque la especificación permite precargar el contenido del anuncio, separando la carga del anuncio de la renderización del mismo, la lógica real para precargar la publicidad y el tratamiento de su ramificación de impresión o el recuento de inventario está fuera del alcance de este documento.
El momento y método para contabilizar la impresión y otras especificaciones sobre las métricas relacionadas con anuncios están fuera del alcance de este documento. Se asume que los Reproductores de vídeo intentarán alinear la monitorización de impresiones con los Estándares de formatos publicitarios interactivos; Formatos de Vídeo y las Métricas Publicitarias en Vídeo In-Stream.

La especificación proporciona previsiones para secuenciar en el tiempo partes múltiples de una única pieza publicitaria. Aun así, cualquier lógica de negocios para secuenciar anuncios múltiples, bien de la misma campaña o de diferentes, está explícitamente fuera del alcance de esta especificación.

La gestión de normas de anuncio, incluyendo cualesquiera normas de negocios, que decide cuándo hacer una llamada de anuncio y qué tipos de anuncios se requieren está fuera del alcance de esta especificación.

Nótese que mientras en todo este documento se usa el término “reproductor”, el estándar VPAID no está limitado a un interfaz específico entre un reproductor de vídeo y un anuncio - puede haber otras capas de interfaces de aplicaciones siempre que, una vez combinadas, las especificaciones VPAID y la interfaz sean respetadas.

4. REFERENCIA API

4.1. PRINCIPIOS RECTORES

El estándar VPAID fue diseñado usando los siguientes principios rectores:

1. Generalidad: El API necesita ser lo suficientemente generalizado como para ser aplicable a las generaciones de formatos que están por llegar en el futuro inmediato.

a. Permite al reproductor integrar la publicidad en tiempo y espacio siempre y cuando estén disponibles todos los parámetros necesarios.

2. Sencillez: El API necesita ser tan sencillo como sea posible mientras se cumpla el punto 1.

3. Mover la carga de API al reproductor: Por ejemplo, formatos de anuncios sencillos no requieren ninguna implementación API.

4. Control del reproductor: El API está diseñado de forma que el anuncio podría requerir ciertos servicios del reproductor, pero el reproductor permanece en última instancia como centro de control de todo el entorno de tiempo de ejecución. Por ejemplo, el reproductor tiene la capacidad de descargar un anuncio que no se comporta adecuadamente.

5. Publicidad "multiplataforma": La especificación del API persigue mantener la independencia de la publicidad en cuanto a parámetros de solicitud del anuncio, esquema XML, especificaciones de las impresiones, registros o métricas, etc... Esto permite que la publicidad sea aplicable a múltiples plataformas.

6. Consistencia de las implementaciones tecnológicas: Se trata del mayor esfuerzo para especificar un API sencillo a través de todas las implementaciones tecnológicas (AS2, AS3, Silverlight, JS bridges, etc...).

4.2. NOMENCLATURA

name : Type – Especifica una variable, propiedad, o nombre de método seguido de tipo de dato, o tipo de dato devuelto.

{} – Especifica un objeto con nombre: Propiedades tipo.

Array<Type> – Especifica un *array* de un tipo particular de datos.

[] – Especifica parámetros opcionales o propiedades de estructura de datos.

| – Especifica “o” para propiedades de estructura de datos.

* – Especifica repetir 0 o más veces.

+ – Especifica repetir 1 o más veces.

4.3. REQUISITOS DEL REPRODUCTOR

El reproductor debe implementar la carga de anuncio, comprobar la presencia de <VPAID> y, si está presente, implementar la versión correcta de VPAID. El reproductor debe imponer mecanismos de recuperación si la publicidad no implementa el VPAID correctamente, p.ej. si el evento AdStopped no se recibe dentro de algún periodo de gracia siguiente a una llamada a StopAd. El reproductor no debe nunca permitir que excepciones no capturadas sean lanzadas mientras se manipulan eventos VPAID desde el anuncio. Mirar secciones de **Implementación** para requerimientos más específicos.

4.4. REQUISITOS DEL ANUNCIO

Si el anuncio implementa VPAID, debe indicar la versión correcta VPAID. El anuncio debe implementar todos los métodos y propiedades listadas, pero puede elegir no hacer nada o devolver valores indicando que el método o propiedad no está implementado. El anuncio no debe nunca permitir que excepciones no capturadas sean lanzadas durante las llamadas al VPAID desde el reproductor. Mirar secciones de **Implementación** para requerimientos más específicos.

4.5. DEPENDENCIA DEL ANUNCIO EN FORMATO XML

La especificación API está diseñada para aislar el anuncio, en la extensión posible, desde el esquema del XML usado para entregar la publicidad. Esto permite reutilizar un anuncio compatible VPAID sin ninguna modificación a través de una variedad de soportes y redes, incluso si se usan diferentes formatos XML para transmitir el anuncio. Este aislamiento disminuido para recodificar el anuncio como versiones del esquema XML (p.ej., VAST [4]) evoluciona. Por ejemplo, la especificación API explícitamente evita la necesidad de pasar todo el XML recibido desde el *ad server* al anuncio, que hubiera requerido el esquema XML específico analizando la lógica en cada anuncio.

Este enfoque, al mismo tiempo, no impide exteriorizar alguno de los parámetros de anuncio en ad XML, si el diseñador del anuncio así lo decide. Hay varios ejemplos de uso común de exteriorización de parámetros de anuncio en XML para anuncios avanzados, incluyendo:

• Exteriorización de URLs de vídeo desde ad SWF. Esto permite codificar/actualizar el vídeo del anuncio con independencia de su desarrollo.

• Parametrizando ciertas partes de los anuncios como los textos de llamada a la acción. La parametrización puede usarse para permitir funcionalidad avanzada como la personalización e internacionalización/localización del anuncio basado en contexto de servicio.

La especificación permite pasar al anuncio una sección designada de XML. La especificación VAST [4] XML ya contiene tal sección de datos. Se asume que el análisis XML es efectuado por el reproductor (o alguno de sus componentes), que tiene la responsabilidad de identificar la sección de datos del anuncio extrayéndola y pasándola al anuncio como específica en el initAd() API abajo.

4.6. ANUNCIO Y REGISTRO DE ACTIVIDAD

En esta especificación hemos mantenido el registro de la actividad fuera del anuncio. Esto es para mantener el anuncio adaptable a tantas plataformas como sea posible:

• Este enfoque mantiene el anuncio independiente de cómo son representados los registros de actividad *beacons* variados dentro del esquema XML.

• Diferentes plataformas de anuncios difieren en la manera en que manejan los registros de actividad. Este enfoque mantiene el anuncio independiente de estas diferencias en el formato.

• Esto también permite que el manejo del *registro de actividad* complejo (p.ej., el *buffering* y dosificación de eventos no esenciales) sea codificado en un lugar dentro del reproductor, en lugar de repetir la misma lógica dentro de cada anuncio.

El API permite al anuncio emitir eventos personalizados, registrando el ID del evento y datos adicionales. El reproductor puede coger estos eventos y registrar la actividad en XML. La situación de uso más común será trazar un mapa del ID del evento desde el anuncio al ID de todos los eventos en el ad XML, y registrando todos los correspondientes eventos o actividad.
Este enfoque no impide que un anuncio registre todos los eventos y los emita de forma propia. Los eventos registrados y completamente resueltos son *eventos a registrar* conocidos en el momento del desarrollo del anuncio y no son exteriorizados dentro del ad XML.

4.7. MÉTODOS

Todos los métodos son llamados por el reproductor.

4.7.1 handshakeVersion

handshakeVersion(playerVPAIDVersion : String) : String

El reproductor llama a la handshakeVersion inmediatamente después de cargar el anuncio para indicar al anuncio que VPAID será utilizado. El reproductor pasa su secuencia de última versión VPAID. El anuncio devuelve una secuencia de versión mínimamente fijada en “1.0”, y de la forma “parche.principal.secundario”. El reproductor debe verificar que soporta la versión particular de VPAID o cancelar el anuncio. Todas las versiones VPAID tienen una compatibilidad retroactiva dentro del mismo número de versión principal (pero sin compatibilidad hacia adelante). Así que si el reproductor soporta “2.1.05” y el anuncio indica “2.0.23”, el reproductor puede ejecutar el anuncio pero no en la situación inversa. Las implementaciones de definiciones de interfaz estático pueden requerir un acuerdo externo para el ajuste de versiones. Las implementaciones dinámicas pueden utilizar la llamada del método handshakeVersion para determinar si un anuncio soporta VPAID. Para lenguajes dinámicos, el anuncio o el reproductor pueden adaptarse para ajustarse a la versión del otro si es necesario. Una buena práctica es llamar siempre a la handshakeVersion incluso si la versión ha sido coordinada externamente, en caso de anuncios que soporten múltiples versiones y usos handshake-Version para decidir cuál actuar en el tiempo de ejecución.

4.7.2 initAd

initAd(width : Number, height : Number, viewMode : String, desiredBitrate : Number[, creativeData : String][, environmentVars : String]) : void

Después de que el anuncio es cargado y el reproductor llama a la handshakeVersion, el reproductor llama al initAd para inicializar la experiencia del anuncio. El reproductor puede precargar el anuncio y retrasar la llamada a initAd hasta la próxima reproducción del anuncio, no obstante, el anuncio no carga sus recursos hasta que se llama el initAd. El anuncio envía el evento de AdLoaded para notificar al reproductor que sus recursos están cargados y está listo para visualizar. El reproductor pasa un ancho y una altura para indicar el área de visualización para el anuncio. viewMode puede ser uno entre los siguientes: “normal”, “thumbnail” o “fullscreen” para indicar el modelo de visualización actual de los reproductores, como son definidos por el reproductor y el editor de anuncio y puede no ser aplicable a todos los anuncios. El reproductor también pasa una deseada tasa de bits en kbps (kilobits por segundo) que el anuncio puede usar al seleccionar la tasa de bits para cualquier flujo de contenido. creativeData es un parámetro opcional que puede usarse para pasar datos adicionales de inicialización de anuncio; por ejemplo, las extensiones del nodo de una respuesta VAST [4]. enviromentVars es un parámetro opcional que puede usarse para pasar variables de tiempo de ejecución de implementación específica, URL-encoded name=value pairs separado por ‘&’. (mirar resizeAd más abajo para más información sobre dimensionamiento).

4.7.3 resizeAd

resizeAd(width : Number, height : Number, viewMode : String) : void

Siguiendo un redimensionamiento del contenedor de anuncio UI, el reproductor llama al resizeAd para permitir al anuncio escalar o reposicionarse dentro de su área de visualización. El ancho y la altura siempre ajusta el área máxima de visualización asignada para el anuncio, y resizeAd sólo es llamada cuando el reproductor cambia su tamaño del contenedor del vídeo. Para anuncios que se expanden o entran en modo lineal, el área entera de visualización del contenido del vídeo está dando el ancho y altura, porque estos anuncios pueden ocupar esa área entera cuando están en modo lineal o expandido. También, el reproductor debe evitar usar el escalamiento incorporado y las propiedades de dimensionamiento o métodos para la tecnología de implementación en concreto. viewMode puede ser uno entre los siguientes: “normal”, “thumbnail” o “fullscreen” para indicar el modo de visualización actual de los reproductores, como son definidos por el reproductor y el editor de anuncio y puede no ser aplicable a todos los anuncios. El reproductor no debe fijar nunca las propiedades del anuncio de ancho, alto, escala X o escala Y, pero debe enmarcarlo en el ancho y altura previstos. No obstante, el reproductor puede fijar las propiedades x e y del anuncio a la posición. Además, el anuncio puede elegir enmarcarse en el ancho y altura para garantizar que el UI colocado fuera de la pantalla nunca es visible.

4.7.4 startAd

startAd() : void

El reproductor llama a startAd cuando quiere que el anuncio empiece a visualizarse. El anuncio responde mandando un evento AdStarted notificando al reproductor que la publicidad se está reproduciendo. El reproductor no puede volver a ejecutar un anuncio llamando múltiples veces a startAd + stopAd.

4.7.5 stopAd

stopAd() : void

El reproductor llama a stopAd cuando no visualizará más el anuncio. También se llama a stopAd si el reproductor necesita cancelar una publicidad. No obstante, el anuncio puede tomarse algún tiempo en cerrar y limpiar recursos antes de enviar un evento AdStopped al reproductor.

4.7.6 pauseAd

pauseAd() : void

Se llama a pauseAd para pausar la reproducción de la publicidad. El anuncio envía un evento AdPaussed cuando ha sido pausado. La publicidad debe apagar todo el audio y suspender cualquier animación o vídeo. El reproductor puede usar **pausa** para esconder el anuncio configurando su visibilidad del contenedor de la visualización. Se deja a criterio de la publicidad quitar los elementos UI o solo parar su animación y, quizás, atenuar su brillo.

4.7.7 resumeAd

resumeAd() : void

Se llama a resumeAd después de la reproducción del anuncio tras una llamada a pauseAD. El anuncio envía un evento AdPlaying cuando el anuncio ha reanudado la reproducción.

4.7.8 expandAd

expandAd() : void

El reproductor llama a expandAd para pedir que el anuncio cambie a su tamaño UI mayor. Por ejemplo, el reproductor puede implementar un botón de abrir que llame a expandAd cuando sea cliclado (ver la propiedad adExpanded más abajo) El reproductor puede usar el valor de la propiedad **adExpanded**, así como el evento AdExpandedChange, para establecer cuándo visualizar un botón de abrir o de cerrar, si fuese necesario. expandAd puede no ser aplicable a todos los anuncios.

4.7.9 collapseAd

collapseAd() : void

El reproductor llama a collapseAd para pedir que el anuncio vuelva a su tamaño de UI más pequeño. Por ejemplo, el reproductor puede implementar un botón de abrir que llame a collapseAd cuando sea cliclado y sea visualizado solo cuando el anuncio se encuentre en un estado expandido (mirar la propiedad **adExpanded** más abajo). collapseAd puede no ser aplicable en todos los casos.

4.8. PROPIEDADES

El reproductor puede acceder a todas las propiedades. Si la propiedad es *getProperty*, el anuncio escribe a la propiedad y el reproductor lee de la propiedad. Si la propiedad es *setProperty*, el reproductor escribe a la propiedad y el anuncio lee de la propiedad.

4.8.1 adLinear

get adLinear : Boolean

El Boolean adLinear indica el modo lineal presente del anuncio vs. el modo de operación no lineal. Con el valor verdad, adLinear indica que el anuncio está en modo de reproducción lineal, y con el valor falso en modo no lineal. El reproductor controla adLinnear inicialmente así como cada vez que se recibe un evento AdLinearChange, y actualiza su estado de acuerdo con los detalles de la colocación del anuncio. Mientras la publicidad está en modo lineal, el reproductor tiene el vídeo del contenido en pausa. Si adLinear está colocada en el valor verdad inicialmente y el anuncio está designado como *pre-roll* (definido externamente), el reproductor puede elegir retrasar la carga del vídeo de contenido hasta cerca del final de la reproducción del anuncio.

4.8.2 adExpanded

get adExpanded : Boolean

El valor Boolean adExpanded indica si el anuncio está en un estado que ocupa más área UI que su área más pequeña. Si el anuncio tiene múltiples estados expandidos, todos los estados expandidos muestran adExpanded en valor verdad. Un evento AdExpandedChange indica que el valor ha cambiado, pero el reproductor puede controlar la propiedad en cualquier momento. Si el anuncio es estáticamente dimensionado, adExpanded es colocado en valor falso.

4.8.3 adRemainingTime

get adRemainingTime : Number

El reproductor puede usar la propiedad adRemainingTime para actualizar el UI del reproductor durante la reproducción del anuncio. La propiedad adRemainingTime está en segundos y se refiere al tiempo en que se puede acceder a la propiedad. El reproductor puede interrogar de manera periódica a la propiedad adRemainingTime, pero debe controlarla siempre que recibe un evento AdRemainingTimeChange. Si no implementado, returns -1 (volver -1). Si desconocido, returns -2 (volver -2). El reproductor puede utilizar el valor de la propiedad adRemainingTime para visualizar un indicador de tiempo restante u otro indicador de duración del anuncio.

4.8.4 adVolume

get adVolume : Number
set adVolume : Number

El reproductor utiliza la propiedad adVolume para intentar fijar u obtener el volumen del anuncio. El valor adVolume está entre 0 y 1 y es lineal. El reproductor es responsable de mantener el estado mudo y de fijar el volumen del anuncio en consecuencia. Si no está implementado el *get* (obtener), siempre volver -1. Si el *set* (fijar) no está implementado, no hacer nada.

4.9. EVENTOS ENVIADOS

Todos los eventos son enviados desde el anuncio al reproductor. Debe tenerse en cuenta que se suben varios eventos como resultado de que el método llama al anuncio. Los gestores de eventos en el reproductor pueden llamar a los métodos de anuncio por turnos. Los reproductores deben escribirse de manera que impidan una sobrecarga debido a un llamamiento a métodos como resultado de un evento.

4.9.1 AdLoaded

El evento AdLoaded es enviado por el anuncio para notificar al reproductor que el anuncio ha terminado cualquier carga de recursos y está listo para visualizarse. El anuncio no intenta cargar recursos hasta que el reproductor llama al método init.

4.9.2 AdStarted

El evento AdStarted es enviado por el anuncio para notivicar al reproductor que el anuncio se está visualizando.

4.9.3 AdStopped

El evento AdStopped es enviado por el anuncio para notivicar al reproductor que el anuncio ha parado de visualizarse, y se han limpiado todos los recursos del anuncio.

4.9.4 AdLinearChange

El evento AdLinearChange es enviado por el anuncio para notificar al reproductor que el anuncio ha cambiado el modo de reproducción. El reproductor debe conseguir la propiedad adLinear y actualizar su UI en consecuencia (ver la propiedad **adLinear** para más información).

4.9.5 AdExpandedChange

El evento AdExpandedChange es enviado por el anuncio para notificar al reproductor que el estado expandido del anuncio ha cambiado. El reproductor puede conseguir la propiedad adExpanded y actualizar su UI en consecuencia (ver método **expandAd** más abajo).

4.9.6 AdRemainingTimeChange

El evento AdRemainingTimeChange es enviado por el anuncio para notificar al reproductor que el tiempo de reproducción restante del anuncio ha cambiado. El reproductor puede conseguir la propiedad adRemainingTime y actualizar su UI en consecuencia. El anuncio solo manda este evento cuando adRemainingTime cambia en relación con el tiempo actual, significando que si el anuncio extiende o contrae su tiempo de visualización general el evento será enviado, pero si el tiempo de visualización general no está cambiado ningún evento será enviado. La propiedad adRemainingTime puede leerse una vez y el reproductor fijar un temporizador basado en ella que no necesite ser actualizado a menos que reciba un evento AdRemainingTimeChange.

4.9.7 AdVolumeChange

El anuncio envía el evento AdVolumeChange para notificar al reproductor que el anuncio ha cambiado su volumen, si el anuncio soporta volumen. El reproductor puede conseguir la propiedad adVolume y actualizar su UI en consecuencia.

4.9.8 AdImpression

El evento AdImpression se utiliza para notificar al reproductor que la fase de usuario-visible ha comenzado. El evento AdImpression puede enviarse usando diferentes criterios dependiendo del tipo de formato que el anuncio esté implementando. Para un *mid-roll* lineal la impresión debe coincidir con el evento AdStart. No obstante, para un anuncio *overlay* no lineal, la impresión debería ocurrir cuando el *banner* de invitación es visualizado, lo que normalmente se produce antes de que el vídeo sea mostrado.

4.9.9 AdVideoStart, AdVideoFirstQuartile, AdVideoMidpoint, AdVideoThirdQuartile, AdVideoComplete

Estos cinco eventos se envían para notificar al reproductor acerca del progreso del vídeo del anuncio. Ellos ajustan los eventos VAST [4] del mismo nombre, así como los eventos de “Porcentaje completo” en función de las *Métricas Publicitarias en Vídeo In-Stream*, y deben implementarse para ser compatibles con IAB.

4.9.10 AdClickThru

El anuncio envía el evento AdClickThru cuando un *click-thru* ocurre. Se pueden incluir parámetros para dar al reproductor la opción de manejar el evento. Son incluidos tres parámetros con el evento, String url, String ID y el Boolean playerHandles. Si playerHandles tiene valor verdad, el reproductor debe manejar el evento abriendo una nueva ventana de navegador en la url, cuando tiene valor falso, el anuncio manejará el evento. Puede usarse la url por el anuncio para especificar la url del *click-thru*, y la ID puede usarse para propósitos de monitoreo. Este evento replica otro del mismo nombre en las *Métricas Publicitarias en Vídeo In-Stream*, y debe implementarse para ser compatible con IAB.

4.9.11 AdUserAcceptInvitation, AdUserMinimize, AdUserClose

El anuncio envía los eventos AdUserAcceptInvitation, AdUserMinimize y AdUserClose cuando se encuentra con el requerimiento con los mismos nombres dados en las Métricas Publicitarias en Vídeo In-Stream. Cada uno de ellos es un *reporting* del anuncio al reproductor sobre una acción iniciada por el usuario. El reproductor puede usarlos para informar externamente, pero no realiza ninguna otra acción. El anuncio debe implementar las acciones UI pero también dispara estos eventos para notificar al reproductor.

4.9.12 AdPaused, AdPlaying

Se envían los eventos AdPaused y AdPlaying en respuesta a llamadas del método al pauseAd y al resumeAd, respectivamente, para indicar que la acción ha surtido efecto (ver las descripciones de los métodos **pauseAd** y **resumeAd** para más detalle).

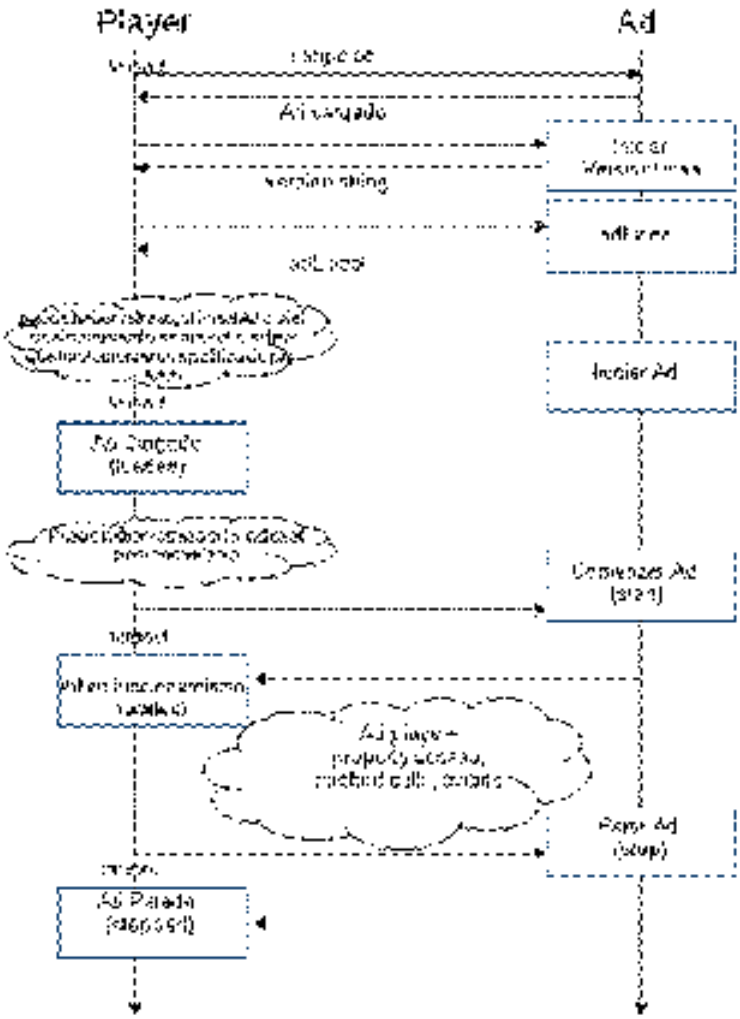
4.9.13 AdLog

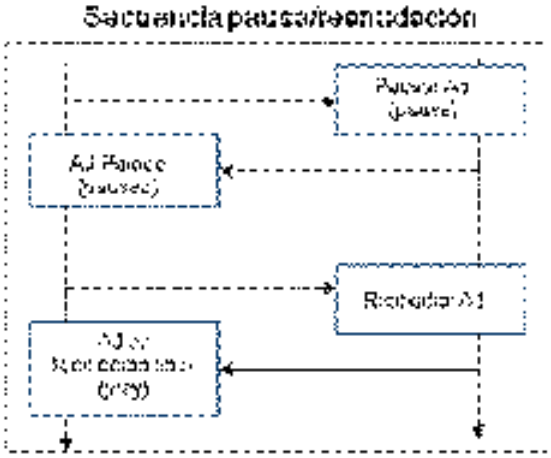
El anuncio envía opcionalmente el evento AdLog al reproductor para transmitir información de depuración, en un parámetro mensaje String. No se requiere que el anuncio provea eventos AdLog, pero puede ser conveniente para ayudar a los ingenieros del reproductor a depurar anuncios particulares.

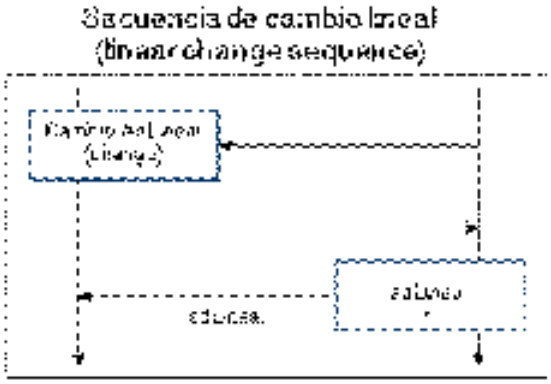
4.9.14 AdError

Se envía el evento AdError cuando el anuncio ha experimentado un error fatal. Antes de que el anuncio envíe AdError se deben limpiar todos los recursos y cancelar cualquier reproducción de anuncio pendiente. El reproductor debe eliminar cualquier anuncio UI, y recuperar su estado de reproducción de contenido regular. El parámetro mensaje String está incluido para que se pase al reproductor información más específica.

4.10. DIAGRAMA DE SECUENCIA







4.11. GESTIÓN DE ERRORES E INTERVALOS

Cualquier error fatal detectado por el anuncio inicia un evento AdError que cancela la reproducción del anuncio y devuelve el reproductor al estado de reproducción del contenido. Asimismo, el reproductor implementa un mecanismo de intervalo para los siguientes aspectos:

1. Petición del archivo de anuncio al archivo de anuncio exitosamente cargado

    a. Acción de recuperación: cancelar la carga del anuncio, saltar anuncio

2. Llamando al initAd y recibiendo el evento adLoaded

    a. Acción de recuperación: eliminar el anuncio de UI, saltar anuncio

3. Llamando al startAd para recibir el evento AdStarted

    a. Acción de recuperación: llamar al stopAd, eliminar anuncio de UI, saltar anuncio

4. Recibiendo el evento AdLinearChange con el conjunto adLinear con valor de verdad para recibir el evento AdLinearChange con el conjunto adLinear con valor de falso (excluyendo pausa de anuncio/reanudar tiempo)

    a. Acción de recuperación: llamar a stopAd, eliminar anuncio de UI

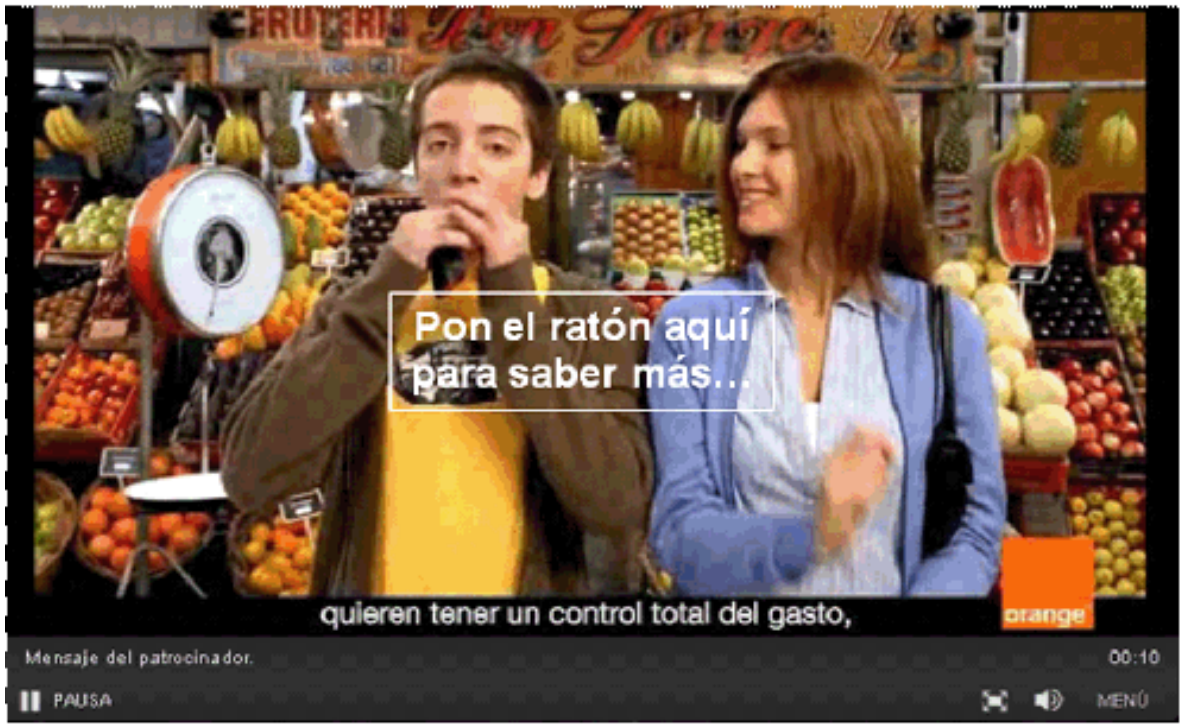
5. Llamando a stopAd y recibiendo el evento AdStopped

    a. Acción de recuperación: eliminar anuncio de UI

5. ANUNCIOS EJEMPLO

5.1. PRE-ROLL CLICABLE

Los anuncios clicables son un formato de vídeo *pre-roll*, con un *overlay* interactivo adicional. El *overlay* puede ser un texto o una imagen, y cuando el usuario hace clic en el anuncio, se le llevará a una página que el anunciante especifica. Esta página se abrirá en una ventana separada.



Con VPAID:	**Sin VPAID:**
1) El reproductor llama a la handshakeVersion("1.0") -> "1.0" 2) El reproductor consigue adLinear = true, retrasa la carga del vídeo del contenido 3) El reproductor player llama a initAd(400, 300, "normal", 500) 4) El anuncio envía una AdLoaded 5) El reproductor llama a startAd 6) El anuncio envía un AdStarted 7) El reproductor obtiene un adRemainingTime* 8) [set adVolume | pauseAd…resumeAd | resizeAd]* 9) El reproductor protege el vídeo del contenido, el reproductor determina el tiempo 10) El anuncio envía una AdStopped porque se ha completado 11) El reproductor reproduce el vídeo del contenido	Usar un pre-roll linear con una duración predefinida. - No interactividad que pudiera ampliar la duración del anuncio - Problemas potenciales de escala - No control de volumen, info de posición, etc. si viene empaquetado como un anuncio avanzado

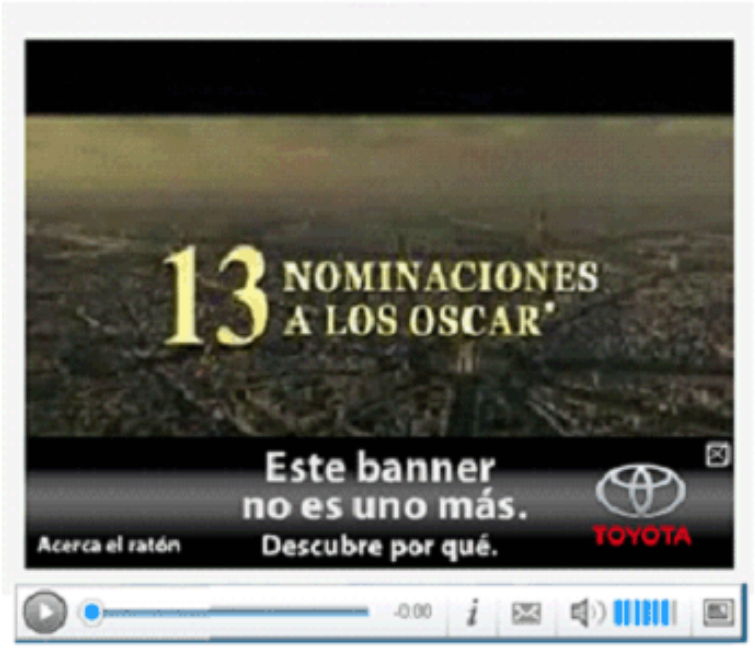
5.2. BANNER ACOMPAÑANTE

Este formato consiste en una publicidad en vídeo con la sección interactiva externa al reproductor de vídeo. Puesto que el área del vídeo no es clicable, es necesario el banner acompañante para determinar si el usuario ha interactuado con el anuncio.

VPAID no se requiere para este caso de uso.	**Sin VPAID:** Usar un vídeo lineal más un banner acompañante. - No interactividad que pudiera ampliar la duración del anuncio - Problemas potenciales de escala - No control de volumen, etc. si viene empaquetado como un anuncio avanzado

5.3. BANNER OVERLAY

El formato *overlay* visualiza las imágenes del patrocinador sobre el vídeo durante la reproducción del mismo. Puede haber reglas añadidas relacionadas con el tiempo de visualización del *banner* así como con la posición.



Con VPAID:	**Sin VPAID:**
1) El reproductor llama a la handshakeVersion("1.0") -> "1.0" 2) El reproductor consigue adLinear, = false, empieza a cargar el vídeo del contenido 3) El reproductor llama al initAd(80, 300, "normal", 500) 4) El anuncio envía la AdLoaded 5) El reproductor reproduce el vídeo del contenido, quizá antes que 4 6) El reproductor llama al startAd() 7) El anuncio envía el AdStarted 8) El reproductor llama al stopAd() cuando el tiempo de visualización del anuncio ha transcurrido 9) El anuncio envía la AdStopped 10) La reproducción del vídeo del contenido continúa	Usar un anuncio no lineal con duración definida. El reproductor debe definir la posición del anuncio. - No interactividad que pudiera ampliar la duración del anuncio - Problemas potenciales de escala - No control de volumen, etc. si viene empaquetado como un anuncio avanzado

5.4. BANNER OVERLAY CON CLIC LINEAL A LA PUBLICIDAD EN VÍDEO

El formato *overlay* superpone las imágenes del patrocinador sobre el vídeo durante la reproducción del mismo. Si el usuario hace clic en el *banner*, el anuncio muestra una publicidad lineal en vídeo seguida de la desaparición del anuncio. Si el usuario no hace clic, el *banner overlay* se visualiza hasta que su tiempo de visualización ha transcurrido.

5.4. BANNER OVERLAY CON CLIC LINEAL A LA PUBLICIDAD EN VÍDEO

Con VPAID:	**Sin VPAID:**
1) El reproductor llama a la handshakeVersion(“1.0”) -> ”1.0” 2) El reproductor consigue adLinear, = false, empieza la carga del vídeo del contenido 3) El reproductor llama al initAd (400, 300, “normal”, 500) 4) El anuncio envía la AdLoaded 5) El reproductor reproduce el vídeo del contenido, quizá antes que 4 6) El reproductor llama al startAd() 7) El anuncio envía el AdStarted 8) Si el usuario hace clic a. El anuncio envía la AdLinearChange, adLinear = true b. El reproductor pausa el vídeo del contenido c. Se reproduce la publicidad en vídeo, se completa d. El anuncio envía el AdLinearChange, adLinear = false e. El reproductor continúa la reproducción del vídeo del contenido 9) Si no hace clic a. El reproductor llama al stopAd() cuando el tiempo de visualización del anuncio ha transcurrido 10) El anuncio envía la AdStopped 11) Continúa reproduciéndose el vídeo del contenido	No es posible

6.IMPLEMENTACIÓN ACTIONSCRIPT 3

6.1. CARACTERÍSTICAS DEL API

El anuncio swf debe ser cargado en su propia ApplicationDomain y, por lo tanto, ser tratado como un tipo de dato *. Tanto el anuncio como el reproductor necesitan una definifición de interfaz VPAID (IVPAID) ya que se están ejecutando en ApplicationDomains separados y, por ello, la interfaz no se puede compartir. El reproductor accede a VPAID llamando al método getVPAID del objeto del anuncio. Para que sea un llamada segura el reproductor puede elegir cómo contener el objeto * devuelto de modo que implemente VPAID explícitamente, como se muestra a continuación. La clase VPAID debe extender EventDispatcher. El reproductor debe llamar a AddEventHandler en el objecto VPAID mediante eventos con nombres De tipo cadena como figuran en la sección de eventos de arriba, por ejemplo, “AdStarted”. Ver sección de eventos personalizados para obtener más explicaciones sobre los eventos.
Stage.frameRate debe ser pasado a initAd en el parámetro environmentVars, por ejemplo, “frameRate = 15”.

```
package
{
   public interface IVPAID
   {
   // Propiedades
   function get adLinear() : Boolean;
   function get adExpanded() : Boolean;
   function get adRemainingTime() : Number;
   function get adVolume() : Number;
   function set adVolume(value : Number) : void;
   // Métodos
   function handshakeVersion(playerVPAIDVersion : String) : String;
   function initAd(width : Number, height : Number, viewMode : String, desiredBitrate : Number, creativeData
:  String, environmentVars : String) : void;
    function resizeAd(width : Number, height : Number, viewMode : String) : void;
    function startAd() : void;
    function stopAd() : void;
    function pauseAd() : void;
    function resumeAd() : void;
   function expandAd() : void;
   function collapseAd() : void;
   }
}
package
{
   // Contenedor del reproductor para swf cargados sin tipo
   public class VPAIDWrapper extends EventDispatcher implements IVPAID
   {
      private var _ad:*;
      public function VPAIDWrapper(ad:*) {
      _ad = ad;
   }
```

6.1. CARACTERÍSTICAS DEL API

```
// Propiedades
public function get adLinear():Boolean {
return _ad.adLinear;
}
public function get adExpanded():Boolean {
return _ad.adExpanded;
}
public function get adRemainingTime():Number {
return _ad.adRemainingTime;
}
public function get adVolume():Number {
return _ad.adVolume;
}
public function set adVolume(value:Number):void {
_ad.adVolume = value;
}
// Métodos
public function handshakeVersion(playerVPAIDVersion : String):String {
return _ad.handshakeVersion(playerVPAIDVersion);
}
public function initAd(width:Number, height:Number, viewMode:String,   desiredBitrate:Number,creativeDa
ta:String, environmentVars : String):void {
_ad.initAd(width, height, viewMode, desiredBitrate, creativeData, environmentVars);
}
public function resizeAd(width:Number, height:Number, viewMode:String):void {
_ad.resizeAd(width, height, viewMode);
}
public function startAd():void {
_ad.startAd();
}
public function stopAd():void {
_ad.stopAd();
}
public function pauseAd():void {
_ad.pauseAd();
}
public function resumeAd():void {
_ad.resumeAd();
}
public function expandAd():void {
_ad.expandAd();
}
public function collapseAd():void {
_ad.collapseAd();
}
```

```
   // EventDispatcher sobreescrito
   override public function addEventListener(type:String, listener:Function,   useCapture:Boolean=false,priori
ty:int=0, useWeakReference:Boolean=false):void {
_ad.addEventListener(type, listener, useCapture, priority, useWeakReference);
   }
   override public function removeEventListener(type:String, listener:Function,
useCapture:Boolean=false):void {_ad.removeEventListener(type, listener, useCapture);
  }
   override public function dispatchEvent(event:Event):Boolean {
   return _ad.dispatchEvent(event);
   }
   override public function hasEventListener(type:String):Boolean {
   return _ad.hasEventListener(type);
   }
   override public function willTrigger(type:String):Boolean {
   return _ad.willTrigger(type);
   }
 }
}
```

6.2. EVENTOS PERSONALIZADOS

Al igual que con la interfaz VPAID por sí misma, las definiciones de eventos de clases no se comparten entre el reproductor y el anuncio. La siguiente definición de clase se puede realizar tanto en el reproductor como en el anuncio, pero como están en ApplicationDomains separadas no puede ser compartido. El anuncio debe crear una clase derivada flash.events.Event que implemente un recogedor de datos de propiedades como se muestra a continuación.

```
package
{
  import flash.events.Event;
  public class VPAIDEvent extends Event
  {
    public static const AdLoaded : String = “AdLoaded”;
    public static const AdStarted : String = “AdStarted”;
    public static const AdStopped : String = “AdStopped”;
    public static const AdLinearChange : String = “AdLinearChange”;
    public static const AdExpandedChange : String = “AdExpandedChange”;
    public static const AdRemainingTimeChange : String= “AdRemainingTimeChange”;
    public static const AdVolumeChange : String = “AdVolumeChange”;
    public static const AdImpression : String = “AdImpression”;
    public static const AdVideoStart : String = “AdVideoStart”;
    public static const AdVideoFirstQuartile : String= “AdVideoFirstQuartile”;
    public static const AdVideoMidpoint : String = “AdVideoMidpoint”;
    public static const AdVideoThirdQuartile : String= “AdVideoThirdQuartile”;
    public static const AdVideoComplete : String = “AdVideoComplete”;
    public static const AdClickThru : String = “AdClickThru”;
    public static const AdUserAcceptInvitation : String= “AdUserAcceptInvitation”;
    public static const AdUserMinimize : String = “AdUserMinimize”;
    public static const AdUserClose : String = “AdUserClose”;
    public static const AdPaused : String = “AdPaused”;
    public static const AdPlaying : String = “AdPlaying”;
    public static const AdLog : String = “AdLog”;
    public static const AdError : String = “AdError”;
    private var _data:Object;
    public function VPAIDEvent(type:String, data:Object=null, bubbles:Boolean=false,
    cancelable:Boolean=false) {

     super(type, bubbles, cancelable);
     _data = data;
  }
  public function get data():Object {
    return _data;
    }
  }
}
```

6.2. EVENTOS PERSONALIZADOS

```
// ejemplo de llamada a ad dispatch desde una function dentro de la clase VPAID del anuncio
   dispatchEvent(new VPAIDEvent(VPAIDEvent.AdStarted));
   dispatchEvent(new VPAIDEvent(VPAIDEvent.AdClickThru,
   {url:myurl,id:myid,playerHandles:true}));
```

El reproductor utiliza addEventListener con una función controladora que recibe un parámetro tipo * que será el evento personalizado. Para continuar con el ejemplo anterior:

```
   public function onAdClickThru(event:*) : void
   {
   trace("Ad url is: " + event.data.url);
   }
   _VPAID.addEventListener(VPAIDEvent.AdClickThru, onAdClickThru);
```

6.3. SEGURIDAD

Para llevar a cabo secuencias de comandos unidireccionales en ActionScript 3, use Security.allowDomain ("<playerdomain o *>"). El anuncio swf también debe ser servido desde un dominio donde / crossdomain.xml permita al anuncio swf ser cargado por el dominio del reproductor o *. El reproductor debe cargar el anuncio swf en dominio de seguridad y aplicación independiente.

7. IMPLEMENTACIÓN ACTIONSCRIPT 2

7.1. CARACTERÍSTICAS DEL API

El anuncio swf cargado debe ser tratado como un tipo sin tipo de datos, su método getVPAID devolverá el objeto VPAID como el tipo de datos sin tipo. Los métodos y propiedades VPAID son accedidos directamente en el objeto VPAID devuelto. Para una llamada segura, el reproductor debe optar por ajustar el objeto sin tipo VPAID con una clase que implemente VPAID de forma explícita. La clase VPAID debe utilizar el código ActionScript 2 class EventDispatcher. El reproductor debe llamar a AddEventHandler en la propiedad VPAID mediante eventos con nombres De tipo cadena que figuran en la sección de eventos anterior, por ejemplo, “STARTED”. Tenga en cuenta que es especialmente importante que el reproductor nunca ajuste las propiedades ancho, alto, scaleX, scaleY de los anuncios directamente o se tiene el riesgo de que efecto de escala sea impredecible en el anuncio.
El reproductor debe llamar a la función para cambiar el tamaño de su lugar.

7.2. EVENTOS PERSONALIZADOS

Los eventos personalizados en AS2 están implementados de forma similar que en AS3 usando la clase EventDispatcher, auque el evento puede ser simplementoun objecto con las propiedades correctas, por ejemplo:

```
// ejemplo de llamada a ad dispatch desde una function dentro de la clase VPAID del anuncio
dispatchEvent({target:this, type:”AdClickThru”, data:{url:myurl, id:myid, playerHandles:true}});
```

7.3. SEGURIDAD

Para llevar a cabo secuencias de comandos unidireccionales en ActionScript 2, use Security.allowDomain (“<playerdomain o *>”). El anuncio swf también debe ser servido desde un dominio donde / crossdomain.xml permita al anuncio swf ser cargado por el dominio del reproductor o *. El reproductor debe cargar el anuncio swf en dominio de seguridad y aplicación independiente.

8.IMPLEMENTACIÓN SILVERLIGHT

8.1. CARACTERÍSTICAS DEL API

El anuncio Silverlight expondrá las API de la parte superior para apoyar la comunicación desde el reproductor. Como no es muy fácil acceder al método getVPAID en el objeto del anuncio, el objeto del anuncio pondrá en marcha la interfaz “IVPAID”. El reproductor puede determinar si el anuncio implementa la interfaz “IVPAID” y si lo hace, se entenderá que el anuncio es compatible con VPAID. La combinación reproductor / anuncio puede decidir utilizar esta
interfaz VPAID como no seguros, pero es recomedable utilizar una interfaz de tipo segurida que define la API de arriba. Este tipo de seguridad se puede lograr mediante la interfaz IVPAID binaria que se puede implementar en el anuncio. El reproductor también utilizará esta interfaz binaria para comprobar si el anuncio implementa la interfaz y por lo tanto estar seguro de que el anuncio es compatible con VPAID. Se puede encontrar más informaciónacerca de esta interfaz en: http:/ /www.iab.net/iab_products_and_industry_services/508676/508950/vpaid

8.1. CARACTERÍSTICAS DEL API

A continuación se especifica la interfaz que implementará el anuncio Silverlight compatible con VPAID (basado en la explicación anterior de VPAID).

```
public interface IVpaid {
#region VPAID Methods
string HandshakeVersion(string version);
void InitAd(double width, double height, string viewMode, int desiredBitrate, string
creativeData, string environmentVariables);
void StartAd();
void StopAd();
void ResizeAd(double width, double height, string viewMode);
void PauseAd();
void ResumeAd();
void ExpandAd();
void CollapseAd();
#endregion
#region VPAID Properties
bool AdLinear { get; }
bool AdExpanded { get; }
TimeSpan AdRemainingTime { get; }
double AdVolume { get; set; }
#endregion
#region VPAID Events
event EventHandler AdLoaded;
event EventHandler AdStarted;
event EventHandler AdStopped;
event EventHandler AdPaused;
event EventHandler AdResumed;
event EventHandler AdExpandedChanged;
event EventHandler AdLinearChanged;
event EventHandler AdVolumeChanged;
event EventHandler AdVideoStart;
event EventHandler AdVideoFirstQuartile;
event EventHandler AdVideoMidpoint;
event EventHandler AdVideoThirdQuartile;
event EventHandler AdVideoComplete;
event EventHandler AdUserAcceptInvitation;
event EventHandler AdUserClose;
event EventHandler AdUserMinimize;
event EventHandler<ClickThroughEventArgs> AdClickThru;
event EventHandler<VpaidMessageEventArgs> AdError;
event EventHandler<VpaidMessageEventArgs> AdLog;
event EventHandler AdRemainingTimeChange;
event EventHandler AdImpression;
#endregion
}
```

8.2. EVENTOS PERSONALIZADOS

```
#region Event Argument classes
  public abstract class VpaidMessageEventArgs : EventArgs {
  public abstract string Message { get; }
  }
  public abstract class ClickThroughEventArgs : EventArgs {
  public abstract string Url { get; }
  public abstract string Id { get; }
  public abstract bool PlayerHandles { get; }
  }
  #endregion
```

8.3. SEGURIDAD

El anuncio Silverlight XAP debe servirse desde un dominio donde / crossdomain.xml permita al anuncio XAP ser cargado por el dominio del reproductor o *.

9. IMPLEMENTACIÓN DE JAVASCRIPT BRIDGE

9. IMPLEMENTACIÓN DE JAVASCRIPT BRIDGE

9.1 CARACTERÍSTICAS DEL API

VPAID puede estar expuesto a JavaScript creando un anuncio contenedor implementado con la misma tecnología del anuncio. El contenedor debe exponer VPAID a JavaScript reenviando llamadas a los métodos JavaScript y accediendo a la propiedad en el objeto contenedor para el anuncio vía VPAID, así como manejar el envío de eventos recibidos y llamadas a funciones en el contendor del anuncio a través de VPAID a JavaScript. La *serialización* de objetos debe ser tratada a través de las capacidades tecnológicas existentes. Como convenio, el acceso a las propiedades debe ser implementado usando
funciones “getVPAIDProperty (propertyName)” y “setVPAIDProperty (propertyName, valor)” y los eventos desde el anuncio al reproductor con funciones con nombre como “VPAIDAdLoaded”.

9.2 SEGURIDAD

La seguridad para el uso de VPAID con JavaScript está dictada por las reglas de la implementación particular de la tecnología y su JavaScript bridge.

10. REFERENCIAS

10. REFERENCIAS

1. Estándares de formatos publicitarios interactivos; Formatos de Vídeo

2. Estándar para la entrega de publicidad en vídeo digital (VAST 2.0)

3. Métricas Publicitarias en Vídeo In-Stream

11. APÉNDICE A: GLOSARIO

11. APÉNDICE A: GLOSARIO

Anuncio de video: Anuncio que es emitido en el contexto del contenido del reproductor de vídeo. Hay que tener en cuenta que la parte esencial de la definición es que el contenido es el vídeo. El mismo anuncio puede o no puede ser un vídeo.

Anuncio de vídeo avanzado: Anuncio de vídeo cuya lógica de programación permite la interactividad del usuario, interacción con el contenido del vídeo o con otras opciones avanzadas.

Anuncio acompañante: Nomalmente texto, anuncio gráfico, *rich media*, o *skin* que aparece alrededor del entorno del video. Estos anuncios son de varios tamaños y formas y, generalmente, se ejecutan al lado o alrededor del reproductor de vídeo.

Anuncio de vídeo lineal: El anuncio es mostrado antes, en el medio, o después de que el vídeo haya sido visto por el usuario, en muchas ocasiones del mismo modo que una televisión puede emitirlo antes, durante o después de un programa.

Anuncio de vídeo no lineal: El anuncio se ejecuta a la vez que el vídeo se está emitiendo, de modo que los usuarios ven ambas cosas a la vez. Los anuncios no lineales pueden ser texto, anuncios gráficos o capas de video superpuestas.

Post-roll: Un anuncio de vídeo lineal que aparece antes de que el contenido del vídeo finalice.

Pre-roll: Un anuncio de vídeo que aparece antes de que el contenido del vídeo comience.

VAST (Video Ad Serving Template): Documento de IAB para la definicón de un formato XML estándar que describe un anuncio que se emitirá en, sobre o alrededor de un reproductor de vídeo.

Reproductor de vídeo: Entorno en el cual un contenido de vídeo es ejecutado. El reproductor de vídeo debe ser implementado por el soporte o suministrado por un proveedor.

12. APÉNDICE B: CONSIDERACIONES FUTURAS

12. APÉNDICE B: CONSIDERACIONES FUTURAS

1. Agregar el evento AdInteraction: El evento AdInteraction es utilizado por el anuncio para informar al reproductor de la interacción de los usuarios. Este evento permite al reproductor del soporte recopilar datos sobre los parámetros de interacción del usuario asociados al anuncio de vídeo avanzado. Es responsabilidad del reproductor del soporte encapsular la información transportada por este evento en el formato soportado por el sistema de información del soporte. El evento lleva los siguientes datos generalizados:

    a. eventType: String

    b. eventValue: String

    c. [Time : Number] (tiempo transcurrido desde la carga del anuncios, en ms, opcional)

    d. [Seq : Number] (número de secuencia del evento, opcional, basado en 0).

2. Crear e incluir enlaces a los anuncios de ejemplo, tanto en Silverlight como Flash.

Para realizar comentarios o sugerencias acerca del presente documento pueden dirigirse a comunicacion@iabspain.net