Getting Started with HLS Interstitials

Preliminary Specification and Implementation Guide

Version 1.0b4

July 12, 2021

Introduction

The HTTP Live Streaming (HLS) protocol delivers live and on-demand audiovisual content to global-scale audiences. Vendors typically insert separate interstitial content into their primary presentations in order to display advertising, branding, or other information to viewers.

High-level interstitial support in HLS makes it easier to create and deliver features such as bumpers and mid-roll ads, as well as more advanced experiences that are difficult to achieve with earlier techniques, such as late binding to current ad inventory and dynamic scheduling.

This document outlines how backend production tools can adopt a new playlist metadata schema to insert interstitial content into HLS streams. It also describes new AVFoundation APIs that applications on Apple platforms can use to discover, monitor, and schedule interstitials for playback.

Overview

Servers can schedule interstitials by placing metadata (EXT-X-DATERANGE tags) into the Media Playlists of the primary asset. Clients can schedule interstitials using AVFoundation APIs. Interstitials can be scheduled (and unscheduled) dynamically during playback.

Interstitial themselves are self-contained media assets. They can be scheduled anywhere on the timeline of a primary media asset. Clients can schedule interstitials against unmodified primary assets. While interstitials must be VOD assets, they can be scheduled against either VOD or live primary content (including Low-Latency HLS streams).

Interstitials are specified by URL. The player loads the resource specified by the URL after it buffers the primary asset to the scheduled interstitial playback time. This allows for late binding to ad inventory. Since an interstitial is described by a single URL, a server can respond to it with a limited number of HTTP redirects without a major impact on performance.

An interstitial request can either be for a single interstitial asset or a list of assets. In the second case, the composition of the list can be determined when the server responds to the interstitial request.

Although content providers are encouraged to provide interstitial assets with a suitable ladder of bitrate tiers, it is not necessary for interstitials to match the tier structure of the primary content. Similarly it is not necessary to provide matching alternative language renditions, although it is recommended. Interstitials do not need to use the same codecs as the primary content, although using different codecs will cause transition delays on certain devices such as some third-party AirPlay receivers.

It is possible to specify that user navigation be restricted while playing interstitials. These restrictions are enforced at the player UI level, which is AVKit or custom application code while playing on-device, or the AirPlay receiver while AirPlay is active,

Devices that do not implement HLS interstitial support will ignore server-generated interstitial events when playing a primary asset. Content vendors seeking backward compatibility can continue to use static interstitial insertion techniques, or simply encode slates or national ads directly into the primary asset, and override them with HLS interstitials for newer devices. Or they can forgo interstitial playback on older devices if the population is small enough to allow it.

Interstitials scheduled inside other interstitials are ignored by clients.

Execution Model

Interstitial playback on Apple devices is accomplished using two players: a primary AVPlayer created by the client application which plays the primary asset, and an interstitial AVQueuePlayer that is created automatically by the primary AVPlayer when it is necessary to play interstitial assets.

The use of two players allows interstitial content to be scheduled anywhere on the primary item timeline, even in the middle of a video GOP.

AVFoundation arranges the buffering and playback transitions between primary and interstitial content to minimize disruption. For best results, interstitial events should be scheduled early enough in advance of playback to allow time for buffering.

For video playback, AVFoundation uses the same AVPlayerLayer(s) provided to the primary AVPlayer to display the interstitial content as well. Interstitial content should use the same aspect ratio as the primary content to minimize disruption.

Clients that wish to monitor interstitial playback to perform quartile reporting, offer custom controls or perform other tasks can do so by observing the interstitial AVQueuePlayer.

New AttributeValue type

The following type definition will be added to the HLS spec, section 4.2 Attribute Lists:

o enumerated-string-list: a quoted-string containing a comma-separated list of enumerated-strings from a set that is explicitly defined by the AttributeName. Each enumerated-string in the list is a string consisting of characters from the set [A-Z] and “-“. The list SHOULD NOT repeat any enumerated-string. To support forward compatibility, clients MUST ignore any unrecognized enumerated-strings in an enumerated-string-list.

Interstitial DATERANGE Schema

The server can insert EXT-X-DATERANGE tags to tell the player to schedule interstitial playback. The EXT-X-DATERANGE CLASS “com.apple.hls.interstitial” specifies how an interstitial is to be scheduled.

Recall that a live Playlist reload can update an existing DATERANGE with new attributes by including a new EXT-X-DATERANGE tag with the same ID. (Existing attribute values cannot be changed.) Also note that all Renditions of a particular Master Playlist must have the same set of EXT-X-DATERANGE tags.

Interstitial EXT-X-DATERANGE tags can have the following attributes:

CLASS

The CLASS attribute is required. Its value must be “com.apple.hls.interstitial”.

X-ASSET-URI

The value of the X-ASSET-URI is a quoted-string absolute URL for a single interstitial asset.

X-ASSET-LIST

The value of the X-ASSET-LIST is a quoted-string URL to a JSON object. The JSON object must contain a key/value pair whose key is “ASSETS” and whose value is a JSON array of Asset-Description JSON objects. (Note that keys in a JSON object are case-sensitive.) Each Asset-Description JSON object MUST have a “URI” member whose value is a quoted-string absolute URL for a single interstitial asset, and a DURATION member whose value is a decimal-floating-point indicating the duration of the interstitial asset in seconds. The client is expected to play the interstitial assets back-to-back in the order that they appear in the ASSETS array.

X-RESUME-OFFSET

The value of X-RESUME-OFFSET is a decimal-floating-point of seconds that specifies where primary playback should resume following the playback of the interstitial. It is expressed as a time offset from where the interstitial playback was scheduled on the primary player timeline. A typical value for X-RESUME-OFFSET is zero. If the X-RESUME-OFFSET is not present, the player uses the duration of interstitial playback for the resume offset, which is appropriate for live playback where playback is to be kept at a constant delay from the live edge, or for VOD playback where the HLS interstitial is intended to replace content in the primary asset.

X-SNAP

The value of the X-SNAP attribute is an enumerated-string-list of Snap Identifiers. The defined Snap Identifiers are: OUT and IN.

If the list contains OUT then the client SHOULD locate the segment boundary closest to the START-DATE of the interstitial in the Media Playlist of the primary content and transition to the interstitial at that boundary. If more than one Media Playlist is contributing to playback (audio plus video for example), the player SHOULD transition at the earliest segment boundary.

If the list contains IN then the client SHOULD locate the segment boundary closest to the scheduled resumption point from the interstitial in the Media Playlist of the primary content and resume playback of primary content at that boundary. If more than one Media Playlist is contributing to playback, the player SHOULD transition at the latest segment boundary.

X-PLAYOUT-LIMIT

The value of X-PLAYOUT-LIMIT is a decimal-floating-point of seconds that specifies a limit for the playout time of the entire interstitial. If it is present, the player should end the interstitial if playback reaches that offset from its start. Otherwise the interstitial should end upon reaching the end of the interstitial asset(s).

X-RESTRICT

The value of the X-RESTRICT attribute is an enumerated-string-list of Navigation Restriction Identifiers. The defined Navigation Restriction Identifiers are: SKIP and JUMP.

If the list contains SKIP then while the interstitial is being played, the client should not allow the user to seek forward from the current playhead position or set the rate to greater than the regular playback rate until playback reaches the end of the interstitial.

If the list contains JUMP then the client should not allow the user to seek from a position in the primary asset earlier than the START-DATE attribute to a position after it without first playing the interstitial asset, even if the interstitial at START-DATE was played through earlier. If the user attempts to seek across more than one interstitial, the client should choose at least one interstitial to play before allowing the seek to complete.

Vendor-defined Attributes

Content vendors may define additional attributes for the com.apple.hls.interstitial CLASS. Vendor-defined attributes should be prefixed by X- and should use a reverse-DNS syntax to avoid collisions

Class rules

A client with specific knowledge of the presentation rules for an asset MAY override restrictions specified by EXT-X-DATERANGE RESTRICT attributes if such an action is consistent with those rules.

Two or more interstitials scheduled at the same START-DATE should be played in the order that their EXT-X-DATERANGE tags appear in the playlist. In that case, any X-RESUME-OFFSET values are cumulative.

An EXT-X-DATERANGE tag with a CLASS of “com.apple.hls.interstitial” has the following constraints:

• It must contain either an X-ASSET-URI attribute or an X-ASSET-LIST attribute but not both.

Interstitial query parameters

Packagers producing “com.apple.hls.interstitial” EXT-X-DATERANGE tags should ensure that X-ASSET-URI and X-ASSET-LIST requests contain an _HLS_interstitial_id query parameter

whose value is the (quoted) ID attribute value of the EXT-X-DATERANGE tag. This supports interoperability between content vendors and decisioning servers.

Certain clients support setting the X-PLAYBACK-SESSION-ID request header with a common, globally-unique value on every HTTP request associated with a particular playback session. Such clients should add an _HLS_primary_id query parameter to X-ASSET-URI and X-ASSET-LIST requests whose value matches the X-PLAYBACK-SESSION-ID of the primary playback session. This provides useful context for decisioning servers.

Clients that cannot set the the X-PLAYBACK-SESSION-ID request header should create a globally-unique value for every primary playback session, and provide this value as an _HLS_primary_id query parameter on both the Master Playlist request for the primary asset and the X-ASSET-URI and X-ASSET-LIST requests made on behalf of that asset.

AVFoundation API For Client-side Insertion

Clients can use AVFoundation API to observe and modify the interstitial playback schedule.

AVPlayerInterstitialEvent

An AVPlayerInterstitialEvent represents a single period of interstitial playback that can be scheduled on a primary asset timeline. Its properties include the time (or date) at which the interstitial is scheduled on the primary timeline, its resumption offset, navigation restrictions, and an array of template AVPlayerItems that will be used to instantiate AVPlayerItems on the interstitial AVQueuePlayer during playback.

AVPlayerInterstitialEventObserver

An AVPlayerInterstitialEventObserver is used to discover the currently-scheduled set of AVPlayerInterstitialEvents and to monitor their playback progress. It is instantiated against the AVPlayer of the primary content. Its properties include the AVQueuePlayer used for interstitial playback, an array of currently-scheduled AVPlayerInterstitialEvents, and the AVPlayerInterstitialEvent that is currently playing. The AVPlayerInterstitialEventObserver broadcasts notifications whenever these events change.

AVPlayerInterstitialEventController

The AVPlayerInterstitialEventController is a subclass of AVPlayerInterstitialEventObserver. It can be used to change the set of currently scheduled AVPlayerInterstitialEvents. It can also be used to cancel a currently playing event , supplying a resumption offset that overrides the one in the event.

Additions to AVPlayer and AVPlayerItem

AVPlayer has an additional AVPlayerWaitingReason for interstitial playback: AVPlayerWaitingDuringInterstitialEventReason. The primary AVPlayer will pause playback and wait for AVPlayerWaitingDuringInterstitialEventReason while the interstitial AVQueuePlayer plays interstitial content.

The AVPlayerItem property automaticallyHandlesInterstitialEvents is normally YES, but it can be set to NO to cause the AVPlayerItem to ignore AVPlayerInterstitialEvents specified by the server in EXT-X-DATERANGE tags.

An AVPlayerItem playing on the interstitial AVQueuePlayer will have a non-nil templatePlayerItem property that matches a template AVPlayerItem in the current AVPlayerInterstitialEvent.

Client Behavior

If an interstitial specifies a non-zero resume offset and the user tries to seek to a time between the start of the interstitial and its resumption point on the primary asset timeline, the client should begin playback from the start of the interstitial.

If a request for either an interstitial asset URL or an asset list URL returns an error, the client should cancel playback of the interstitial with a resume offset of 0.

Clients should allow a generous amount of time (up to a minute) for a server to respond to requests for interstitial assets or asset lists, to enable the server to perform back-end decisioning. Servers must respond quickly enough to avoid playback disruptions on the client.

Example: Interstitial EXT-X-DATERANGE

In this playlist an EXT-X-DATERANGE tag schedules a 15-second ad to play four seconds into a six-second primary asset. The player will play the interstitial and then resume playback of the primary asset where it left off. Seeking and scanning forward will be disabled during interstitial playback. The EXT-X-DATERANGE tag includes a vendor-defined beacon attribute that can be processed by the client.

```
#EXTM3U

#EXT-X-TARGETDURATION:6

#EXT-X-PROGRAM-DATE-TIME:2020-01-02T21:55:40.000Z

#EXTINF:6,

main1.0.ts

#EXT-X-ENDLIST

#EXT-X-DATERANGE:ID="ad1",CLASS="com.apple.hls.interstitial",START-
DATE="2020-01-02T21:55:44.000Z",DURATION=15.0,X-ASSET-URI="http://example.com/
ad1.m3u8",X-RESUME-OFFSET=0, X-RESTRICT="SKIP,JUMP",X-COM-EXAMPLE-BEACON=123
```

Example: Scheduling Interstitials

This example shows how to use an AVPlayerInterstitialEvent with an AVPlayerInterstitialEvent-Controller to schedule the playback of two interstitial assets (“ad1URL” and “ad2URL”) ten seconds after the beginning of a primary asset “movieURL.” Playback of the primary item resumes at the point where it left off.

```
let player  = AVPlayer(url: movieURL)

let controller = AVPlayerInterstitialEventController(primaryPlayer: player)

let adPodItems = [AVPlayerItem(url: ad1URL), AVPlayerItem(url: ad2URL)]

let event = AVPlayerInterstitialEvent(primaryItem: player.curentItem, time: CMTime(seconds: 10,
preferredTimescale: 1), templateItems: adPodItems, restrictions: [], resumptionOffset: .zero)

controller.events = [event]

player.play()
```

Example: Monitoring Interstitial Playback

This example shows how to use an AVPlayerInterstitialEventObserver to discover when playback of an interstitial asset begins or ends so that it can update its user interface.

```
let player  = AVPlayer(url: movieURL)

// The movieURL has interstitials inserted using EXT-X-DATERANGE tags.

let observer = AVPlayerInterstitialEventObserver(primaryPlayer: player)
```

```
NotificationCenter.default.addObserver(
    forName: AVPlayerInterstitialEventObserver.currentEventDidChangeNotification,
    object: observer,
    queue: OperationQueue.main) {
    notification_ in
    self.updateUI(observer.currentEvent)
}
```

Document Revision History

This table describes the changes to *Getting Started with HLS Interstitials*

Date	Revision	Notes
2020-12-14	0.1	First revision
2021-02-15	1.0b1	First public draft. Added PLAYOUT-LIMIT
2021-03-01	1.0b2	Added SNAP-OUT and SNAP-IN
2021-05-12	1.0b3	Introduced enumerated-string-list
2021-06-15	1.0b4	Add _HLS_primary_id and _HLS_interstitial_id