Adobe® Marketing Cloud

VideoHeartbeat SDK Guide for TVML - Version 1.x

Table of Contents

VideoHeartbeat SDK 1.x for TVML

VideoHeartbeat SDK 1.x for Marketing Cloud Solutions lets you measure video engagement through Video heartbeats on AppleTV applications written in JavaScript TVJS/TVML.

Getting Started

Information to help you get starting with the AppleTV VideoHeartbeat JS SDK. This section assumes that you have configured a report suite through Adobe Mobile Services to collect app data and also have the ADBMobile build for tvOS.

Get the SDK

After unzipping the VideoHeartbeat-tvml-1.*.zip download file, you'll have the following software components:

- *ReleaseNotes.txt*
- *readme.txt*: Getting started file for VideoHeartbeat TVML SDK
- *docs/VideoHeartbeat_tvml.pdf*: Documentation for VideoHeartbeat TVML SDK.
- *libs/VideoHeartbeat-tvml.min.js*: VideoHeartbeat TVML SDK.
- *libs/libAdobeVHLTVMLBridge.a, libs/ADBMediaHeartbeatJSExport.h:* TVML bridge library
- *samples*: This folder contains sample application built using client-server model of TVML framework.

Video Analytics via VideoHeartbeats

The APIs belonging to video heartbeats enable real-time dashboards and other media reporting capabilities. This guide is intended for a media integration engineer who has an understanding of the APIs and workflow of the media player being instrumented. Implementing these APIs requires that your media player provide the following:

- An API to subscribe to player events. The media heartbeat requires that you call a set of simple APIs when events occur in your player.
- An API or class that provides player information, such as media name and play-head position.

Integrating video heartbeats APIs

Integrating media analytics real-time media tracking into a media player requires including Adobe Mobile SDK, instantiating and configuring the media heartbeats instance, listening to media player events and using appropriate media heartbeats APIs in your project. Therefore the steps for a developer integrating the media heartbeats APIs are as follows:

Initial Setup

1. Acquire the required Adobe Mobile SDK for tvOS and add it into your project.
2. Acquire "libAdobeVHLTVMLBridge.a" and "ADBMediaHeartbeatJSExport.h" files under libs directory and add it to your project.
3. Install the TVML hooks using *installTVMLHooks* API on ADBMediaHeartbeatJSExport.h

   [ADBMediaHeartbeatJSExport installTVMLHooks:self.appController];

4. Acquire the VideoHeartbeat TVML SDK and load it using evaluateScripts API on launch of the application.

   ```
   evaluateScripts(["http://path_to_VideoHeartbeat_lib/VideoHeartbeat-tvml.min.js"], function(success) {
         if(success) {
            console.log("MediaHeartbeat JS lib loaded successfully.");
         } else {
            console.log("MediaHeartbeat JS lib failed to load.");
         }
      });
   ```

5. Provide the configuration for Adobe Mobile through ADBMobileConfig.json, and configuration for video heartbeat should be provided using MediaHeartbeatConfig.

Approaches to tracking Media during a Playback Session

Using Tracking APIs: This way of integration involves handling player events and calling relevant media heartbeats APIs from the application. The APIs documentation and this guide should be used to learn the details on all the media heartbeats APIs and their respective life cycle.

Configure Media heartbeats

Use ADB.MediaHeartbeatConfig instance to configure media heartbeats. Following table shows structure of MediaHeartbeatConfig:

ADB.MediaHeartbeatConfig

Config Parameter	Description
trackingServer	String representing the URL of the tracking endpoint on the backend side.
publisher	String representing the content publisher unique identifier.
channel	String representing the name of the content distribution channel.
ssl	Boolean representing whether ssl should be used for tracking calls
ovp	String representing the name of the video player provider
sdkVersion	String representing the current version of the app/sdk.
playerName	String representing the name of the player.
setDebugLogging	Method to set debug logging. Takes Boolean parameter as an input.

Sample Implementation:

```
var MediaHeartbeatConfig = ADB.MediaHeartbeatConfig;

MediaHeartbeatConfig.setDebugLogging(true);
MediaHeartbeatConfig.trackingServer = "example.com";
MediaHeartbeatConfig.playerName = "sample-player";
MediaHeartbeatConfig.publisher = "sample-publisher";
MediaHeartbeatConfig.channel = "sample-channel";
MediaHeartbeatConfig.sdkVersion = "test-sdk";
MediaHeartbeatConfig.ssl = false;
MediaHeartbeatConfig.ovp = "test-ovp";
```

If *MediaHeartbeat* is incorrectly configured, media module (VHL) will go into disabled state and will not send any tracking calls.

MediaHeartbeat track methods

ADB.MediaHeartbeat

Method	Description
trackLoad	Media playback tracking method to track Media Load, and set the current session active Ex: *' Create a media info object* *mediaInfo = new ADB.MediaHeartbeat.MediaInfo()* *mediaInfo.id = "sample-media-id"* *mediaInfo.name = "VideoName"* *mediaInfo.playerName = "AppleTV sample player"* *mediaInfo.length = "600"* *mediaInfo.streamType = ADB.MediaHeartbeat.StreamType.VOD*

	‘ Create context data if any
	  mediaContextData = {}
	  mediaContextData["cmk1"] = "cmv1"
	  mediaContextData["cmk2"] = "cmv2"
	
	ADB.MediaHeartbeat.trackLoad(mediaInfo, mediaContextData)

trackUnload	Media playback tracking method to track Media Unload and deactivate current session Ex: *ADB.MediaHeartbeat.trackUnload()*
trackPlay	Media playback tracking method to track Media Play Ex: *ADB.MediaHeartbeat.trackPlay()*
trackPause	Media playback tracking method to track Media Pause Ex: *ADB.MediaHeartbeat.trackPause()*
trackComplete	Media playback tracking method to track Media Complete Ex: *ADB.MediaHeartbeat.trackComplete()*
trackError	Error tracking method to track Player Error Ex: *ADB.MediaHeartbeat.trackError(msg.GetMessage())*
trackEvent	Media tracking method to track events that do not belong to media lifecycle and are optional. Example: AD_START/AD_COMPLETE, CHAPTER_START/CHAPTER_COMPLETE; Refer Events section for detailed list of events. This method takes three arguments: *event constant, event info, and context data (send empty object if there is no context data)*

setDelegate	Method to set delegate on media heartbeat to receive playhead position and QoSInfo. Media Heartbeat will invoke “getCurrentPlaybackTime” and “getQoSInfo” methods on this delegate to get playhead and QoS values respectively.
	
	Ex:
	ADB.MediaHeartbeat.setDelegate({
	  “getCurrentPlaybackTime” : function() {
	    return playhead;
	  },
	  “getQoSInfo” : function(){
	    var qosInfo = new ADB.MediaHeartbeat.QoSInfo();
	    qosInfo.bitrate = player.getBitrate();
	    qosInfo.fps = player.getFPS();
	    qosInfo.droppedFrames = player.getDroppedFrames();
	    return qosInfo;
	  }
	});

In addition to the above methods, the ADB.MediaHeartbeat also provides constants to track media events.

ADB.MediaHeartbeat.Event

Constant	**Description**
BufferStart	EventType for Buffer Start

BufferComplete	EventType for Buffer Complete
SeekStart	EventType for Seek Start
SeekComplete	EventType for Seek Complete
BitrateChange	EventType for Bitrate change
ChapterStart	EventType for Chapter Start
ChapterComplete	EventType for Chapter Complete
ChapterSkip	EventType for Chapter skip
AdBreakStart	EventType for AdBreak Start
AdBreakComplete	EventType for AdBreak Complete
AdBreakSkip	EventType for AdBreak Skip
AdStart	EventType for Ad Start
AdComplete	EventType for Ad Complete
AdSkip	EventType for Ad Skip

ADB.MediaHeartbeat.StreamType

LIVE	Constant for Stream Type LIVE
VOD	Constant for Stream Type VOD

There is also convenience methods as described below for creating various info objects sent through the ADB.MediaHeartbeat API methods. Please refer to the table below

Method	**Description**
MediaInfo	This method returns an initialized Media Information object *Ex:* *mediaInfo = new ADB.MediaHeartbeat.MediaInfo()*
AdInfo	This method returns initialized Ad Information object *Ex:* *adInfo = new ADB.MediaHeartbeat.AdInfo()*
ChapterInfo	This method returns initialized Chapter Information object *Ex:* *chapterInfo = new ADB.MediaHeartbeat.ChapterInfo()*
AdBreakInfo	This method returns initialized AdBreak Information object *Ex:* *adBreakInfo = new ADB.MediaHeartbeat.AdBreakInfo()*
QoSInfo	This method returns initialized QoS Information object *Ex:* *QoSInfo = new ADB.MediaHeartbeat.QoSInfo()*

Once the application developer is familiar with all the above APIs, integrating media heartbeats with the application media player can be achieved by calling raw tracking APIs directly.

Using Media Tracking APIs

App developers can use the media tracking APIs described above directly to track media life cycle and ad/chapter events. Please refer to the sample app for sample implementation on implementing these APIs.

Opt-Out and Privacy Settings

You can control whether or not Video Analytics data is sent on a specific device, for this VideoHeartbeats relies on Adobe Mobile Privacy policy, which can be done using the following settings:

- *privacyDefault* setting in ADBMobile JSON Config. This controls the initial setting that persists until it is changed in code.
- *ADBMobile.setPrivacyStatus()* method. After the privacy setting is changed using this method, the change is permanent until it is changed again using this method, or the app is completely uninstalled and re-installed.

Note: Media heartbeat tracking calls are also disabled if the privacy status is set to opt-out

Set this to ADBMobilePrivacyStatusOptIn/ADBMobilePrivacyStatusOptOut if user wants to opt-in/opt-out, or ADBMobilePrivacyStatusUnknown if it is unknown.

```
ADBMobile.setPrivacyStatus(ADBMobilePrivacyStatusOptOut);
```

This method will return the current value for privacy status constant (ADBMobilePrivacyStatusOptIn or ADBMobilePrivacyStatusOptOut or ADBMobilePrivacyStatusUnknown)

```
ADBMobile.privacyStatus();
```

Debug Logging

ADBMobile library provides debug logging through the *setDebugLogging* method. Debug logging should be set to false for all the production apps. Note, this debug logging API is not for VideoHeartbeats.

```
ADBMobile.setDebugLogging(true);
```

Using Bloodhound to Test AppleTV Applications

During application development, Bloodhound lets you view server calls locally, and optionally forward the data to Adobe collection servers.

Bloodhound can be downloaded from any app configuration page in Adobe Mobile services.

Bloodhound 3 Beta for Mac documentation

Bloodhound 2 for Windows documentation

Contact and Legal Information

Information to help you contact Adobe and to understand the legal issues concerning your use of this product and documentation.

Help & Technical Support

The Adobe Marketing Cloud Customer Care team is here to assist you and provides a number of mechanisms by which they can be engaged:

• Check the Marketing Cloud help pages for advice, tips, and FAQs
• Ask us a quick question on Twitter @AdobeMktgCare
• Log an incident in our customer portal
• Contact the Customer Care team directly
• Check availability and status of Marketing Cloud Solutions

Service, Capability & Billing

Dependent on your solution configuration, some options described in this documentation might not be available to you. As each account is unique, please refer to your contract for pricing, due dates, terms, and conditions. If you would like to add to or otherwise change your service level, or if you have questions regarding your current service, please contact your Account Manager.

Feedback

We welcome any suggestions or feedback regarding this solution. Enhancement ideas and suggestions for the Analytics suite can be added to our Customer Idea Exchange.

Legal