StreamhubAnalytics Swift generic plugin for iOS and tvOS applications

5th of March 2018	2.0	QoS, metadata Frontloading tracking
23rd of April 2018	3.0	Swift 4.0 compatible Updated links
5th of June 2019	4.0	Swift 5.0 compatible version
19th of August 2019	5.0	Refactoring. Removed AlamoFire dependency. Added new integration samples applications

For all kind of Apple applications (iOS, tvOS, swift app, objective-c app...), we provide a unique plugin built using the Swift programming language.

Download and extract the plugin from the archive at https://s3-eu-west-1.amazonaws.com/streamhub-static-content/plugins/genericplugins/player-plugin-ios-generic.zip

The plugin comprises the following:

1. A folder **uk.co.streamhub** that contains the 3 files you need to integrate the StreamhubAnalytics library in your project
2. A folder **integration-samples** that contains:
    a. **Integration-sample-swift** is a example app integrating the plugin in a Swift application with a video player.
    b. **IntegrationSampleObjC** is an example app integration the plugin in an Objective-C application.

Getting Started

- Unzip the archive
- Browse around the code to get familiar with the plugin files and API

Integrate The Plugin In Your Own Application

The integration process is quite easy and straightforward, because the complexity of interacting with our Backend REST API has been abstracted in the plugin logic.

If Your Application Is written In Swift

Just add the folder *uk.co.streamhub* to your project and start using the plugin API.

If Your Application Is written In Objective-C

There are plenty of resources online explaining how to use Swift code in an Objective-C application.
This article
https://medium.com/ios-os-x-development/swift-and-objective-c-interoperability-2add8e6d6887 is a good reference and you should be setup and ready to use the plugin API in 2 mn.

Just like for a Swift application, add the folder *uk.co.streamhub* to your project and start using the plugin API.

About The Integration Sample Applications

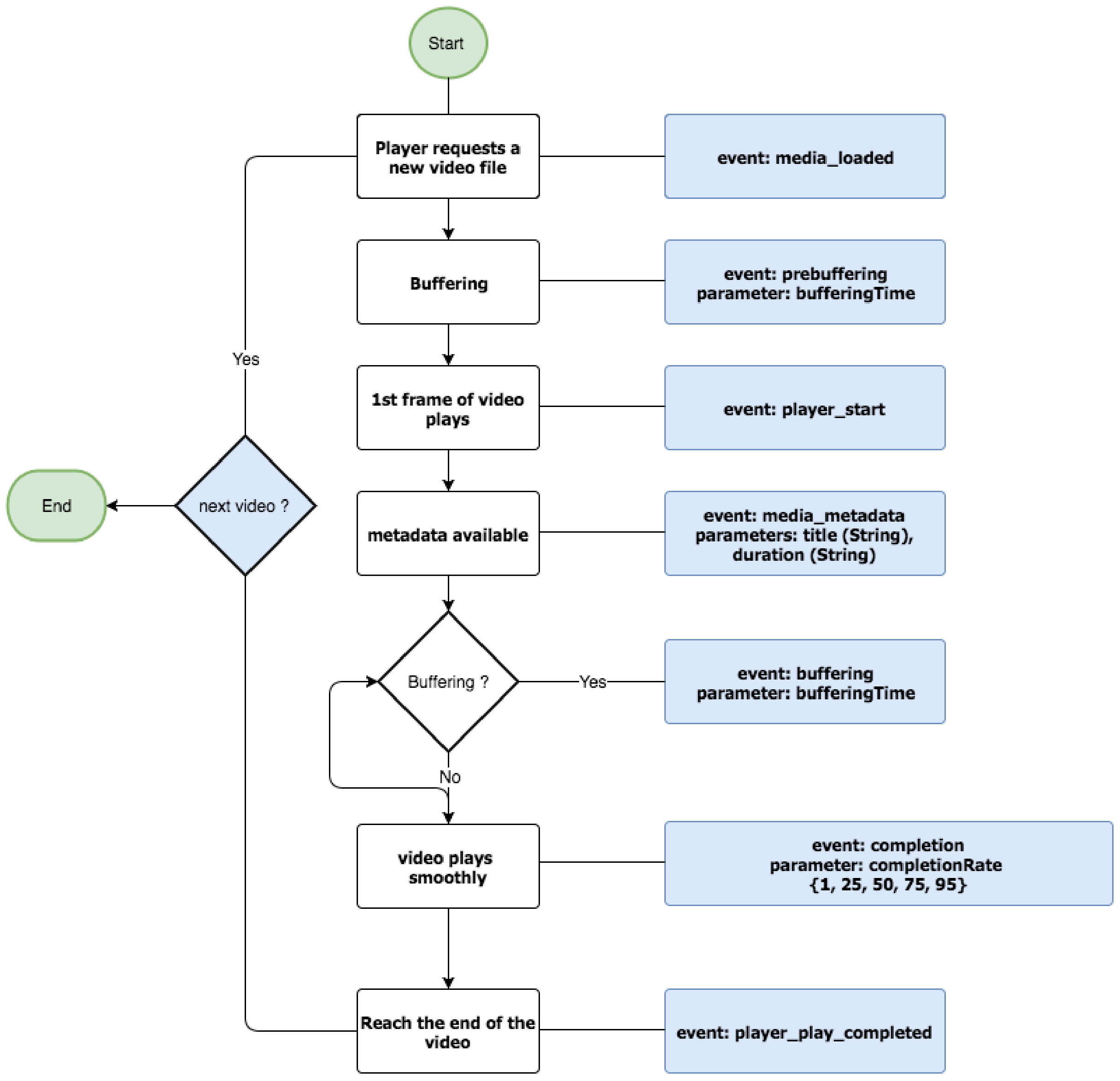
The Swift sample application is featuring a video player and all the necessary callbacks and observers to track the playback.

Despite being absolutely basic, this application should be the reference one to help integrating the Streamhub plugin.

The Objective-C sample application simulates calling the plugin API at regular intervals, and is just tracking media playback heartbeats. Please refer to the Swift sample code to understand how to track buffering or rebuffering events, send metadata or track ads.

Player Event Flow

The diagram below shows a typical playback tracking.
The final validation of your integration should include a verification that the ticks and events of your players are emitted, and in the right order.
This diagram is drawn from
https://streamhub.co.uk/wp-content/uploads/2019/08/player-side-rest-api-4.pdf which provides further informations relevant to any integrations


Start
Player requests a new video file
event: media_loaded
Buffering
event: prebuffering parameter: bufferingTime
1st frame of video plays
event: player_start
metadata available
event: media_metadata parameters: title (String), duration (String)
Buffering ?
Yes
event: buffering parameter: bufferingTime
No
video plays smoothly
event: completion parameter: completionRate {1, 25, 50, 75, 95}
Reach the end of the video
event: player_play_completed
next video ?
Yes
End

Create an instance of the StreamhubAnalytics class

Swift code:

```
var _streamhub:StreamhubAnalytics = StreamhubAnalytics(partnerId: "jwplatform",
endPoint: "http://stats.streamhub.io", playerId: "ios-swift-generic-test-player",
isLive: false, userId: "", analyticsId: "streamhub-5812d")
```

Objectice-C code:

```
StreamhubAnalytics* streamhub = [[StreamhubAnalytics alloc] initWithPartnerId:@"jwplatform"
endPoint:@"http://stats.streamhub.io" playerId:@"ios-swift-generic-test-player" isLive:NO
userId:@"" analyticsId:@"streamhub-5812d"];
```

Parameters details

name	type	comments
partnerId	String	provided to you by your Streamhub integration engineer
endPoint	String	`http://stats.streamhub.io` is the production endPoint for submitting stats. However, you could choose to use your own mockup server during the plugin development phase.
isLive	Boolean	specify if the video content is either a Live or a VoD program. If the content is Live, then, we usually require that you provide a channel identifier for the programs being played by your video player. See the API method **setChannelId** below. For any details, please contact your Streamhub integration engineer.
playerId	String	unique identifier of the player playing the video. It is a unique identifier for the player in both your system and ours. If any clarification is needed, please contact your Streamhub integration engineer.
userId	String	is used if you are able to track your user via a user identifier. If so, you will be able to see user-specific analytics (How old is the public watching the program XYZ ? Which gender ? Which location ? so forth...)
analyticsId	String	The main tracking code that has been provided to you by your Streamhub integration engineer.

Call the API methods

setUserAgent (required)

The plugin being generic for all kind of Apple devices, there is cases where the user agent can't be detected automatically at runtime.
As a result, we ask you to hardcode it, according to your specific device.

Swift:

```
_streamhub?.setUserAgent( "My-Platform-User-Agent" )
```

<u>Objectice-C:</u>

```
[streamhub setUserAgent:@"My-Platform-User-Agent"];
```

setChannelId (required for live only)

If your are playing Live video content, then you probably have EPG metadata relating to it. Usually, in the EPG metadata, a program belong to a channel. We require you to provide the channel identifier associated with the playing program if you have initialized the StreamhubAnalytics library with *isLive: true*.

<u>Swift</u>

```
_streamhub?.setChannelId("a-channel-unique-identifier-mapping-to-some-epg-content")
```

<u>Objective-C</u>

```
[streamhub setChannelId: @"a-channel-unique-identifier-mapping-to-some-epg-content"];
```

setDuration (optional, vod or live with time-shifting)

If your video content has a known duration, then you should provide it to the plugin so that it will be able to report stats with completion rates events. Completion rates will give you insight on how much of your videos are actually watched by your end users.

<u>Swift</u>

```
_streamhub?.setDuration(seconds)
```

<u>Objective-C</u>

```
[streamhub setDuration:68.9f];
```

onMediaReady (required)

onMediaReady must be called to provide the unique identifier of the media about to be played.

<u>Swift</u>

```
_streamhub?.onMediaReady("sara-screen-recording")
```

<u>Objective-C</u>

```
[streamhub onMediaReady:@"sara-screen-recording"];
```

onMediaLoaded

The onMediaLoaded() API signals the beginning of a playback session.

That is the first event your application needs to send and that needs to happen *before* the app is requesting the first video bytes from the network (in order to measure some QoS metrics accurately)

Swift
```
_streamhub?.onMediaLoaded();
```

Objective-C
```
[streamhub onMediaLoaded];
```

onMediaBufferedComplete

The onMediaBufferedComplete(bufferingTime: Int, prebuffering: bool) API signals that a buffering window has been detected.
That API needs to called at the end of the buffering sequence (e.g. when the playback has resumed) and it must be sent with a bufferingTime parameter and a boolean signaling if the buffering has occured before (prebuffering: true) the video's first frame has been played (e.g before calling the *onMediaStart()* API) or after (prebuffering: false)the playback has started (e.g signaling a rebuffering)
In the case of a rebuffering, the event should be called each times it happens.

Swift
```
_streamhub?.onMediaBufferedComplete(bufferingTime: Int(timeInterval), prebuffering: true)
```

Objective-C
```
[streamhub onMediaBufferedComplete:3 prebuffering:true];
```

addMediaMetadata, onMediaMetadata (required)

The addMediaMetadata() and onMediaMetadata() API function works together so that your app can provide media metadata to StreamhubAnalytics.
First you need to call addMediaMetadata(String key, String value) to provide the metadata field name and field value.
Valid keys are ; \`duration\`, \`title\`, \`playerTitle\`, \`categories\`.
All values must be provided as strings. You should not care about providing \`duration\` explicitly since this metadata will be added to the metadata object when you call the setDuration API. However, should you wish to provide that metadata explicitly, you have to provide the \`duration\` parameter as an **Integer**
The API also takes care of url-encoding the values to use them in HTTP requests.
Finally, the \`categories\` metadata should be provided as a comma-delimited string of category.

Example:

Swift

```
var duration = 4
var added = _streamhub?.addMediaMetadata( key: "title", value: "SARA Screen Recording" )
added = _streamhub?.addMediaMetadata( key: "duration", value: String(_duration) )
added = _streamhub?.addMediaMetadata( key: "playerTitle", value: "My Beautiful Player" )
added = _streamhub?.addMediaMetadata( key: "categories", value:
"some,meaningful,categories" )
```

The addMediaMetadata method returns a boolean value (true, if the metadata has been correctly added. false otherwise)

Once you have provided all the metadata you wanted, you will need to call the onMediaMetadata() API method to actually send a metadata event to our collector service.

Swift

```
_streamhub?.onMediaMetadata();
```

Objective-C

```
[streamhub onMediaMetadata];
```

onMediaStart (required)

onMediaStart must be called the first time your video content starts to play.

Swift

```
_streamhub?.onMediaStart()
```

Objective-C

```
[streamhub onMediaStart];
```

onTick (required)

onTick is a convenient method that abstract the logic of measuring accurately the video watching time.
the AVFoundation framework provides different way to read the video playback current time (see sample apps).
This is the perfect place to call the *onTick* api method**.**
You should call *onTick* during the whole duration of the video playback.
The unique parameter has to be provided in seconds.

Swift

```
_streamhub?.onTick(seconds)
```

Objective-C

```
[streamhub onTick:1.0f];
[streamhub onTick:2.33f];
[streamhub onTick:3.4f];
[streamhub onTick:5.7f];
```

onMediaComplete (required)

Call *onMediaComplete* to report a video completed event.
The state of the plugin is reset when you call *onMediaComplete*.
As a result, *onMediaReady, onMediaStart, setDuration* should be called again to process the next video if any.

Swift
```
_streamhub?.onMediaComplete()
```

Objective-C
```
[streamhub onMediaComplete];
```

Tracking Ads

If you need to track Ads into our system to get Ad specific reports, we provide 2 utility functions that you can call to report the viewing of ads in your video player and user engagement.

Note: you must call onMediaReady before you can call any of the ad tracking utility function. This is too ensure that the system can compute for which main content you are tracking ads.

trackAd

Call this method during the Ad playback. You need to provide an **adInfo** of type Dictionary<String, String> which contains the identifiers for your ad and the percentage of the ad that has been watched as an Integer value.

adInfo depending on how your Ad metadata are stored, you might need to provide 1 or more identifier to track your ad.
At the minimum, you could provide an id following the Ad-id standard (http://www.ad-id.org/) but, if your ad id is not universal unique identifier, then you could also rely on a couple of other fields to identify your ad like a combination of an Ad id + campaignId + advertiserId.

Below is the list of fields you can provide as key within the adInfo dictonary. Use the ones you depend on.

Key	**Comments**
id	Required, String
campaignId	Optional, String
distributorId	Optional, String

advertiserId	Optional, String
code	Optional, String
customField	Optional, String. Use this field to add any other identifiers not listed above

Percentile how much of your ad has been watched. The standard in the ad industry is to send an event every time 25% of the Ad has been watched.
Possible values are then integer values: 0, 25, 50, 75, 100.

Swift
```
public func trackAd(adInfo: [String: String], percentile: Int)

Examples:

let adInfo:[String: String]  = [ "id": "6128", "advertiserId": "487" ]

… and later...
_streamhub?.trackAd(adInfo, percentile: 0)
```

onAdClick

Call *onAdClick* to report the user clicking on the Ad during playback.
The parameters are the same as the trackAd function.

Swift
```
_streamhub?.onAdClick(adInfo:Dictionary<String, String>, percentile:int )

examples:

let adInfo:[String: String]  = [ "id": "6128", "advertiserId": "487" ]

… and later...
_streamhub?.trackAd(adInfo, percentile: 0)
```