# Creating your own Flash plugin for the Streamhub Analytics platform

## Overview

We provide a Actionscript 3.0 library that you can use to create your own plugin bridge between a particular type of video player you use and our analytics platform.

The Streamhub Analytics Actionscript 3.0 library ease that development task by exposing  a simple and comprehensible set of predefined Actionscript methods for sending video usage reporting data to the Streamhub platform.

The library is provided as a set of Actionscript 3.0 files (no compiled .swc, no external dependancies) so that it's quite easy to just drop the files in your Flash or Flex project and build your own bridge or OSMF plugin from it.

Download the library.

## Integration

### Copy / Paste the library Actionscript 3.0 files into your project.

Here is the library file structure.
The only class you have to deal with is **StreamhubAnalytics.as**

```
|--- com
       |--- sh
            |--- events
                    |-------- StreamhubEvent.as
            |--- model
                    |-------- StreamhubConstants.as
            |--- utils
                    |-------- SessionUtils.as
                    |-------- Logger.as
            |--- StreamhubAnalytics.as
```

## Create an instance of the StreamhubAnalytics prototype

The next step is to create a StreamhubAnalytics object.

```
var streamhub:StreamhubAnalytics = new StreamhubAnalytics(partnerId, endPoint, playerId,
isLive, userId, analyticsId);
```

### 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.<br>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.<br>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 methods within the library to track events and ticks

Once a library instance has been created, you can access the API methods.
Example: `streamhub.onMediaStart();`

## API Reference

As a minimum, we require you to call the onMediaReady, onMediaStart, onTick and onMediaComplete API functions for each viewing of a video.

### setChannelId (required for live only)

If your are playing Live video content, then you probably have EPG metadata related 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*.

Actionscript
```
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.

Actionscript
```
streamhub.setDuration(seconds);
```

### pageReferrer (optional)

Set the referrer of the page on which you flash-based video player is embedded.

Actionscript
```
shAnalytics.pageReferrer = ExternalInterface.call("window.document.referrer.toString") as
String;
```

### pageLocation(optional)

Set the location of the page on which you flash-based video player is embedded.

A little bit of coding might be required to retrieve the location url, in particualr if your player is embedded in an iframe and cross-domain policy restriction apply.

Actionscript

```
shAnalytics.pageLocation = ExternalInterface.call("window.document.location.toString") as
String;
```

### onMediaReady (required)

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

Actionscript

```
streamhub.onMediaReady("sara-screen-recording");
```

### onMediaStart (required)

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

Actionscript

```
streamhub.onMediaStart();
```

### onTick (required)

*onTick* is a convenient method that abstract the logic of measuring accurately the video watching time.
The sample app create a thread to read the video playback current time at regular interval.
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.

Actionscript

```
streamhub.onTick(seconds);
```

### setPaused (optional)

Call this method when media is paused or unpaused, providing  the appropriate boolean value.

Actionscript

```
streamhub.setPaused( true );
```

### 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.

Actionscript

```
streamhub.onMediaComplete()
```

## Code sample

```
var partnerId:String     = "some-partner";
var endPointUrl:String   = "https://stats.streamhub.io";
var playerId:String      = "some-player-identifier";
var isLive:Boolean        = true; // whether your player is providing a live service or a
onDemand service
var userId:String       = undefined; // can be used to track your logged users if your
service requires authentication
var analyticsId:String   = "your-analytics-id";
var firstplay:Boolean       = true; // a convenient variable to track the first time the
media is played and call the onMediaStart function accordingly


var shAnalytics:StreamhubAnalytics = new StreamhubAnalytics( partnerId, endPointUrl,
playerId, isLive, userId, analyticsId );

shAnalytics.pageReferrer = ExternalInterface.call("window.document.referrer.toString") as
String;

shAnalytics.pageLocation = ExternalInterface.call("window.document.location.toString") as
String;


/**
 * Here we handle a fictional mediaLoaded event callbacks for a fictional video player
accessible via the js variable player
 */
player.on( 'mediaLoaded', function( metadata:Object ):void {
    shAnalytics.setChannelId( metadata.channelId );
    // if the show is a live-dvr and the duration is known, provide it through the
setDuration to enable completion rates reporting
    shAnalytics.setDuration( metadata.duration );
    shAnalytics.onMediaReady( metadata.videoId );
});

/**
 * Here we handle a fictional playStateChange event callback. We track usage of play and
pause event and call the onMediaStart function if the media is played for the first time.
 */
player.on( 'playStateChange', function( isPlaying:Boolean ):void{
    if(isPlaying == true){
    if(firstplay){
        firstplay = false;
        shAnalytics.onMediaStart();
    }
        shAnalytics.setPaused(false);
    }
    else
        shAnalytics.setPaused(true);
});

/**
 * Here we handle the fictional event callback 'timeUpdate' providing the current time in
the video playback
 * make sure currentTimeInSecond has a second resolution (some players API provide this
value in milliseconds)
 * This is the most important part of the integration as it will keep track of the video
views and engagement of the person watching the video.
 */
player.on( 'timeUpdate', function( currentTimeInSecond:Number /* usually a float value */
):void{
```

```
    shAnalytics.onTick( currentTimeInSecond );
});

player.on( 'complete', function( event:Event ):void{
    shAnalytics.onMediaComplete();
});
```