

GUIDE

Online Video Advertisement

V4.3 March 2016





Table of contents

Document Change History.....	1
Introduction	2
Overview of the implementation process.....	2
How to implement the SDK.....	3
Events and variables that are tracked.....	3
When to trigger events and variables.....	5
Report suite settings.....	5
Manually tracking a player.....	7
Manually tracking a player - Javascript.....	8
Manually tracking a player - iOS.....	11
Manually tracking a player - Android.....	13
Validation Process	15
Initial validation by the media owner	15
Final validation.....	15
Publication/release.....	16
What tools do you need?.....	16
Tests	17
Contacts.....	19



Document Change History

Revision Date	Summary of Changes
November 30, 2015	<ul style="list-style-type: none">• Updated subsite example, table on p.4• Added general configuration of AppMeasurement.js, p8
March 21, 2016	<ul style="list-style-type: none">• Updated contact details



Introduction

MMS has developed an online video currency - one standardized way of measuring video for the industry, by the industry. Together with our partner Adobe, MMS provides a toolkit (SDK) for tracking video advertisements, which is implemented on video players and available for both desktop and mobile platforms. This document describes how to implement the SDK and how to validate and test it for correctness.

Overview of the implementation process

Every new source of video ads must go through the initial implementation process in order to ensure a qualitative ad measurement. Examples of new sources of video ads:

- A new party entering the measurement standard
 - A new player or app
 - A new 3rd party ad delivery system
1. **Preparation**
 - a. Identifying and coordinating people involved, including product owners and developers and setting deadlines.
 - b. Sharing the necessary resources, including technical documentation, SDK and validation guide.
 2. **Startup meeting**
 - a. Walkthrough of the overall implementation process, the technical documents, validation and challenges and goals.
 3. **Implementation**
 - a. Actual implementation with the presence, either on site or via screen-sharing session, of the measuring company and MMS in order to enable direct feedback.
 - b. Pre validation using the validation protocol
 4. **Final validation**
 - a. Final validation by Adobe and thereby a permission to go live with the player or app.

In addition, changes to the video player that may affect ad tracking requires a new validation to ensure that the player is functioning correctly.



How to implement the SDK

This section describes in detail how to implement the SDK on various platforms.

Events and variables that are tracked

Below are the variables used in the solution.

NOTE!

The variables to implement are the ones marked “Context data”. Other columns are just noted as a reference.

Analytics variable	Context data	Used for	Where/when to set
Event 1	ad.play	Ad start/resume	When an ad starts playing or resumes after pause
Event 2	ad.stop	Ad stop	When an ad is paused or closed
Event 7	ad.impression	Ad impression	When an ad starts playing, i.e. post-buffering when the ad is shown to the end-user
Event 12	ad.complete	Ad complete	When an ad is played through to the end
Event 20	ad.timeplayed	Ad second watched	Tracks the total time of the ad played. If 15 sec played then "event20=15"
Event 21	ad.segmentstart	Ad segment view	Sent on quartile start (start, 25%, 50%, 75%)

NOTE regarding time played:

Time played should always be in whole seconds and should be implemented as “time played since last entry”

I.e if a video is 10 seconds long and paused after 5 seconds then resumed, than ad.timeplayed should be implemented as:

stop: `s.contextData['ad.timeplayed'] = '5';`

complete: `s.contextData['ad.timeplayed'] = '5';`

To give a total of 10 seconds.

The metadata about the ad and the video player are critical for identifying the video. All variables should be sent together with the first request (Event 1, Event 7, and Event 21). **eVar2 (Custom ad id / External ID) and eVar15 (Ad ID) must be set on all requests.**

Variable name	Context data	Used for	Report name
s.eVar1	ad.videoplayer	Name of the video/ad player. E.g. "FreeWheel Player", "Videoplaza Player"	Video player
s.eVar2	ad.customid	The unique id of the Ad (Film code). Begins with [S]1-9, [5]1-9, [H]1-9, use the sample values "SiMYME202B", "S2TRER1001", "S5TRER1001", "51TRER1001"	Custom ad id / External ID
s.eVar3	ad.duration	The total time of the ad in whole seconds, displayed as "5", "7", "30" etc.	Ad duration
s.eVar4	ad.format	Video format, specified as "linear video ad", "non-linear video ad" and "companion ad"	Ad Format
s.eVar5	ad.type	Used to determine what type of ad is displayed, use "pre-roll", "post-roll" or "mid-roll"	Ad type
s.eVar6	ad.segment	Use format "1:M:0-25" for first quartile (0-25%), "2:M:25-50" for the second quartile etc	Ad segment
s.eVar7	ad.sitename	Set eVar7 to "site name value", value will automatically be overwritten with site domain value (using Processing Rules available within SiteCatalyst)	Site Name
s.eVar8	ad.subsite	Set to [product].["android"/"ios"/"flash"/"html"]. Use small characters and "." as a delimiter. Sample: "dplay.iphone", "tv3play.flash", "tv4play.android" etc.	Sub site
s.eVar9	ad.platform	Platform the ad is being played on, use "desktop", "mobile" or "app". Device will be captured automatically	Platform
s.eVar13 (only applicable for Videoplaza)	ad.campaignid	Campaign id	Campaign ID
s.eVar14 (only applicable for Videoplaza)	ad.goalid	Goal id provided by the ad server	Goal ID
s.eVar15	ad.id	Set to the internal Ad ID provided by the Ad Server	Ad ID / Ad Unit ID

When to trigger events and variables

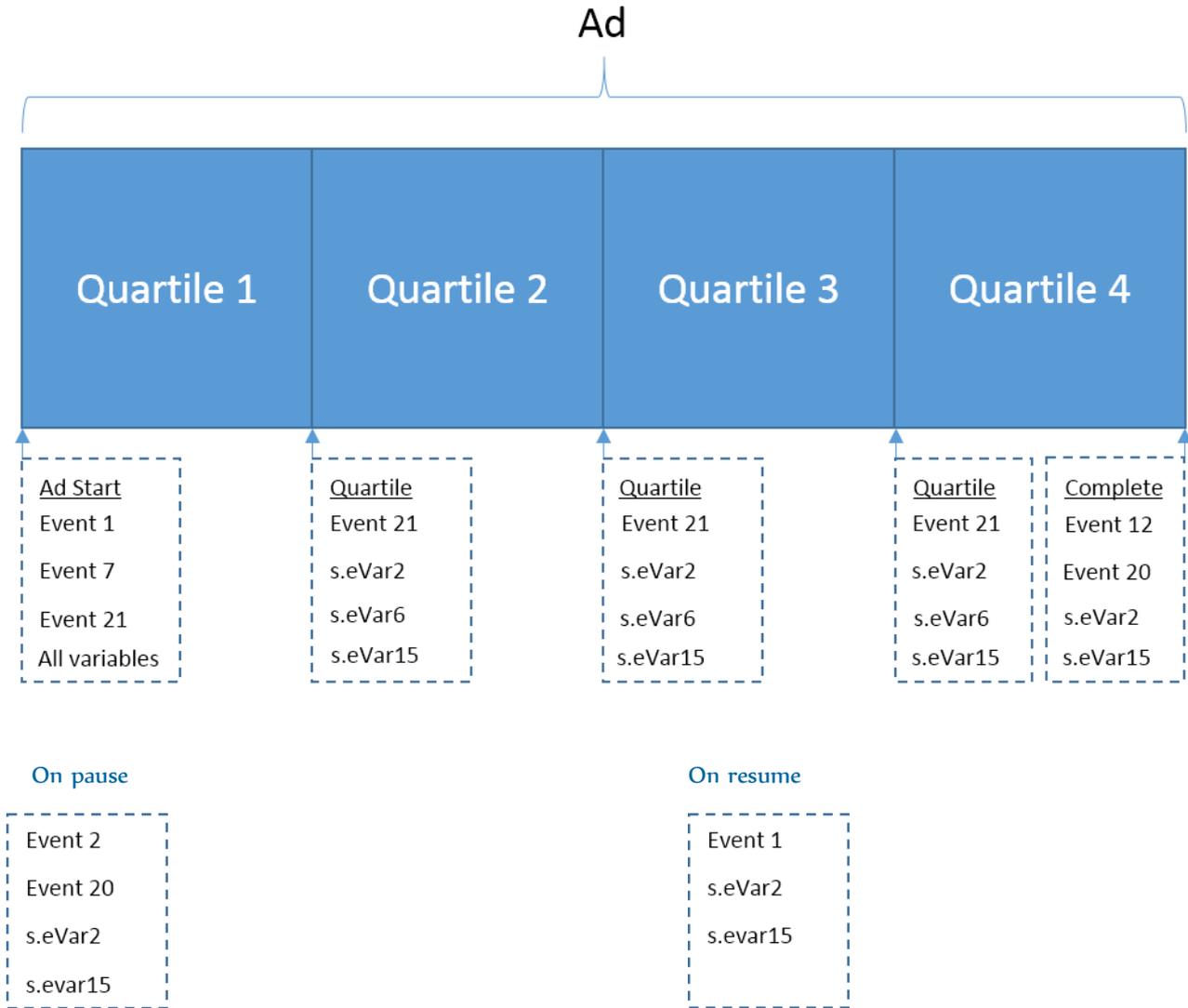


Figure 1: Events and variables tracked for an ad

Report suite settings

Report suite ID determines where data is sent. This value should have separate values in development and production. The actual values to use will be provided from either MMS or Adobe.

Development: <mmsdev>

Production: <mmsprod>



You may also need to set “tracking server” if this is not already configured.

This should always be “mms.d2.sc.omtrdc.net”

The settings for the different platforms:

- Javascript

Report suite: s.account = “<Report suite ID>”;

Tracking server: s.trackingserver = “mms.d2.sc.omtrdc.net”;

- iOS

Set in “ADBMobileConfig.json”

```
"analytics": {  
    "rsids": "<Report suite ID>",  
    "server": "mms.d2.sc.omtrdc.net",
```

- Android

Set in “ADBMobileConfig.json”

```
"analytics": {  
    "rsids": "<Report suite ID>",  
    "server": "mms.d2.sc.omtrdc.net",
```

NOTE! You should receive pre-populated configuration files with the above settings already set.

But for example report suite ID needs to be updated before launching in production.



Manually tracking a player

The implementation in this guide is based on a manual implementation. That is NOT to use any of the media modules that are available in the different SDKs. The reason is to have as clean and easy implementation as possible which should be easier to maintain and also better aligned between platforms and players.

Files required:

AppMeasurement.js (javascript)

This is the core tracking file containing the main tracking functions and should be referenced on all pages where Adobe Analytics should be implemented.

- ADBMobile.h, AdobeMobileLibrary.a, ADBMobileConfig.json (iOS)

Core Adobe Analytics library that should be imported

General help section: https://marketing.adobe.com/resources/help/en_US/mobile/ios/dev_qs.html

- adobeMobileLibrary.jar (Android), ADBMobileConfig.json

Core Adobe Analytics library that should be imported

General help section: https://marketing.adobe.com/resources/help/en_US/mobile/android/dev_qs.html

If possible Adobe Consulting will deliver the necessary files to you. The file(s) will have all the necessary settings included as shown below. What need to be adjusted are the report suites data should be sent to.

Also the functions shown below should be considered as examples. How and where to implement these functions can be customized by the client to best fit their environment and solution.

SDK reference

Javascript:

When logged in to Analytics go to “Admin > Code Manager”. Download Appmeasurement.js

Mobile:

Log in to Mobile Services (mobilemarketing.adobe.com) and choose “Manage App Settings”.

Here you will find the latest SDK as well as already customized config files.

All the latest versions, including general mobile implementation samples and documentation, can also be found here:

<https://github.com/Adobe-Marketing-Cloud/mobile-services/releases>

Manually tracking a player - Javascript

General configuration of AppMeasurement.js:

```
s.linkTrackVars="contextData.ad.impression,contextData.ad.segmentstart,contextData.ad.play"+  
",contextData.ad.stop,contextData.ad.videoplayer,contextData.ad.customid"+  
",contextData.ad.duration,contextData.ad.format,contextData.ad.type"+  
",contextData.ad.segment,contextData.ad.sitename,contextData.ad.subsite"+  
",contextData.ad.platform,contextData.ad.campaignid,contextData.ad.goalid"+  
",contextData.ad.complete,contextData.ad.timeplayed,contextData.ad.id"
```

```
/*Call on video start*/  
function startMovie(){  
s.contextData['ad.play'] = '1'; //event1  
s.contextData['ad.impression'] = '1'; //event7  
s.contextData['ad.segmentstart'] = '1'; //event21  
s.contextData['ad.videoplayer'] = '<Video player>';  
s.contextData['ad.customid'] = '<Custom AD ID>';  
s.contextData['ad.duration'] = '<Ad duration>';  
s.contextData['ad.format'] = '<Ad format>';  
s.contextData['ad.type'] = '<Ad type>';  
s.contextData['ad.segment'] = '1:M:0-25';  
s.contextData['ad.sitename'] = '<Site name>';  
s.contextData['ad.subsite'] = '<Sub site>';  
s.contextData['ad.platform'] = '<Platform>';  
s.contextData['ad.campaignid'] = '<Campaign ID>';  
s.contextData['ad.goalid'] = '<Goal ID>';  
s.contextData['ad.id'] = '<Ad ID>';  
s.contextData['ad.impression'] = ""; //clear event7 to avoid being set on following calls  
s.contextData['ad.segmentstart'] = ""; //clear event21 to avoid being set on following calls  
s.tl(true, "o", "Video tracking - start");  
}  
/*Call on video pause and slider grab*/  
function stopMovie(){
```

```
s.contextData['ad.stop'] = '1'; //event2
s.contextData['ad.timeplayed'] = '<current offset>'; //event20, whole seconds
s.tl(true, "o", "Video tracking - stop");
s.contextData['ad.stop'] = ""; //clear to avoid being set on following calls
}
```

*/*Call on video resume from pause and slider release*/*

```
function playMovie(){
s.contextData['ad.play'] = '1'; //event1
s.tl(true, "o", "Video tracking - resume");
s.contextData['ad.play'] = ""; //clear to avoid being set on following calls
}
```

*/*Call on video end*/*

```
function endMovie(){
s.contextData['ad.complete'] = '1'; //event12
s.contextData['ad.timeplayed'] = '<time played>'; //Whole seconds
s.tl(true, "o", "Video tracking - complete");
}
```

*/*Call on quartiles*/*

```
function quartiles(iQuartile){
  switch(iQuartile) {
    case 25:
      s.contextData['ad.segment'] = '2:M:25-50';
      s.contextData['ad.segmentstart'] = '1'; //event21
      break;
    case 50:
      s.contextData['ad.segment'] = '3:M:50-75';
      s.contextData['ad.segmentstart'] = '1'; //event21
      break;
    case 75:
      s.contextData['ad.segment'] = '4:M:75-100';
```



```
s.contextData['ad.segmentstart'] = '1';           //event21
break;
}
s.tl(true, "o", "Video tracking - segment");
s.contextData['ad.segment'] = "";                 //clear to avoid being set on following calls
s.contextData['ad.segmentstart'] = "";           //clear to avoid being set on following calls
}
```

You can find a simple (javascript) example on

http://dahlberg06.businesscatalyst.com/mms/javascript_nomedia.html



Manually tracking a player - iOS

First step is core implementation and lifecycle tracking. Please refer to:

https://marketing.adobe.com/resources/help/en_US/mobile/ios/dev_qs.html

- Initial play

```
NSMutableDictionary *contextData = [NSMutableDictionary dictionary];
[contextData setObject:@"1" forKey:@"ad.play"];
[contextData setObject:@"1" forKey:@"ad.impression"];
[contextData setObject:@"1" forKey:@"ad.segmentstart"];
[contextData setObject:@"<Video player>" forKey:@"ad.videoplayer"];
[contextData setObject:@"<Custom AD ID>" forKey:@"ad.customid"];
[contextData setObject:@"<Ad duration>" forKey:@"ad.duration"];
[contextData setObject:@"<Ad format>" forKey:@"ad.format"];
[contextData setObject:@"<Ad type>" forKey:@"ad.type"];
[contextData setObject:@"1:M:0-25" forKey:@"ad.segment"];
[contextData setObject:@"<Site name>" forKey:@"ad.sitename"];
[contextData setObject:@"<Sub site>" forKey:@"ad.subsite"];
[contextData setObject:@"<Platform>" forKey:@"ad.platform"];
[contextData setObject:@"<Campaign ID>" forKey:@"ad.campaignid"];
[contextData setObject:@"<Goal ID>" forKey:@"ad.goalid"];
[contextData setObject:@"<Ad ID>" forKey:@"ad.id"];
[ADBMobile trackAction:@"Video tracking - start" data:contextData];
```

- Complete

```
NSMutableDictionary *contextData = [NSMutableDictionary dictionary];
[contextData setObject:@"1" forKey:@"ad.complete"];
[contextData setObject:@"<Time played>" forKey:@"ad.timeplayed"];
[ADBMobile trackAction:@"Video tracking - complete" data:contextData];
```



- Segment start

```
NSMutableDictionary *contextData = [NSMutableDictionary dictionary];  
[contextData setObject:@"<Segment info>" forKey:@"ad.segment"];  
[contextData setObject:@"1" forKey:@"ad.segmentstart"];  
[ADBMobile trackAction:@"Video tracking – segment start" data:contextData];
```

- Stop

```
NSMutableDictionary *contextData = [NSMutableDictionary dictionary];  
[contextData setObject:@"<Time played>" forKey:@"ad.timeplayed"];  
[contextData setObject:@"1" forKey:@"ad.stop"];  
[ADBMobile trackAction:@"Video tracking – stop" data:contextData];
```

- Resume

```
NSMutableDictionary *contextData = [NSMutableDictionary dictionary];  
[contextData setObject:@"1" forKey:@"ad.play"];  
[ADBMobile trackAction:@"Video tracking – resume" data:contextData];
```

Manually tracking a player - Android

First step is core implementation and lifecycle tracking. Please refer to:

https://marketing.adobe.com/resources/help/en_US/mobile/android/dev_qs.html

- Initial play

```
HashMap<String, Object> exampleContextData = new HashMap<String, Object>();
exampleContextData.put("ad.play", "1");
exampleContextData.put("ad.impression", "1");
exampleContextData.put("ad.segmentstart", "1");
exampleContextData.put("ad.videoplayer", "<Video player>");
exampleContextData.put("ad.customid", "<Custom AD ID>");
exampleContextData.put("ad.duration", "<Ad duration>");
exampleContextData.put("ad.format", "<Ad format>");
exampleContextData.put("ad.type", "<Ad type>");
exampleContextData.put("ad.segment", "<Ad segment>");
exampleContextData.put("ad.sitename", "<Site name>");
exampleContextData.put("ad.subsite", "<Sub site>");
exampleContextData.put("ad.platform", "<Platform>");
exampleContextData.put("ad.campaignid", "<Campaign ID>");
exampleContextData.put("ad.goalid", "<Goal ID>");
exampleContextData.put("ad.id", "<Ad ID>");
Analytics.trackAction("Video - start", exampleContextData);
```

- Complete

```
HashMap<String, Object> exampleContextData = new HashMap<String, Object>();
exampleContextData.put("ad.complete", "1");
exampleContextData.put("ad.timeplayed", "<Time played>");
Analytics.trackAction("Video - complete", exampleContextData);
```



- Segment start

```
HashMap<String, Object> exampleContextData = new HashMap<String, Object>();  
exampleContextData.put("ad.segmentstart", "<Segment info");  
exampleContextData.put("ad.segmentstart", "1");  
Analytics.trackAction("Video - segment start", exampleContextData);
```

- Stop

```
HashMap<String, Object> exampleContextData = new HashMap<String, Object>();  
exampleContextData.put("ad.stop", "1");  
exampleContextData.put("ad.timeplayed", "<Time played>");  
Analytics.trackAction("Video - stop", exampleContextData);
```

- Resume

```
HashMap<String, Object> exampleContextData = new HashMap<String, Object>();  
exampleContextData.put("ad.play", "1");  
Analytics.trackAction("Video - resume", exampleContextData);
```

Validation Process

The validation process is initialized by a change or modification to the player that may affect MMS video ad tracking. With that said, it's up to the player owner, in consultation with the engineers, to decide if the modification affects the ad tracking.

Examples of changes that may affect the advertisement tracking:

- changes to ad format of any kind
- changes on how the ads are delivered and interacted with
- changes of player functionality
- updates to the measurement SDK

Initial validation by the media owner

This step is key to keep up efficiency in this process cycle. A designated validator should be assigned to test the player every time. The designated validator must have insight in the player development process and close contact with the player engineers in order to effectively resolve any issues. By using the validation protocol it's possible to, in an early stage, detect any faults in the ad tracking and correct them before sending the player version to a final validation. The protocol is available on the MMS website, http://mms.se/?page_id=2521. Run all tests in the "Tests" section of this document, if all events are triggered as expected then you mark them in the protocol. Save the protocol and give it to Adobe in the next step.

Final validation

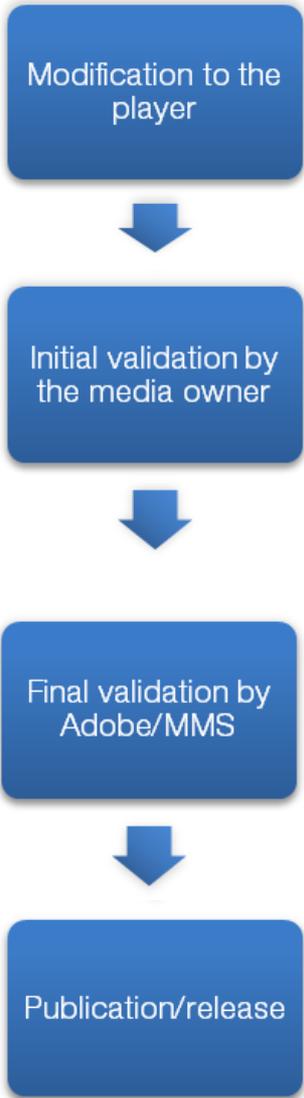
A final validation by Adobe is required in order to get permission to releasing the new player version. This guaranties the continuous quality of the ad tracking. Any critical events or variables that fail to pass the validation must be corrected before releasing the player version. A validation review takes 5 working days starting from the date complete information is submitted. Validation requests should be sent to Adobe with a cc to MMS. Contact information is available under Contacts in this document.

Please use the following template when requesting a validation:

"I would like some help with validating [insert video player version name here] which is an "ios"/"android"/"html5 AND flash based player". We have done pre-validations and the results are attached.

How to access the video player and the different kinds of ads to do the validation requested:

1. Log in to:[insert location] with username [insert username] and password [insert the password]
2. Download the: [Insert video player version name here]
3. We've ensured that Adobe can access the player and play the ads: "whenever between 1-3/5"
4. What to look for: "we only have pre-roll, and the pre-rolls should be present in all shows and all clips.", "we have both pre-rolls, mid-rolls and post-rolls. Pre-rolls are available on all clips, but mid-rolls/post-rolls are only available on...", "play a few video clips and wait, mid-rolls will appear after X video/X minutes. No pre-rolls or post-rolls are available". "Please note that..."
5. We use the following list of Ad Providers for this video player: "Videoplaza", "Freewheel" etc.



Publication/release

Only after all the critical events and variables are approved from a final validation from Adobe the player owner is permitted to go public with the player version. You are absolutely obligated to correct any errors before the final step of publishing the player. Send a copy of the final validation protocol to Adobe and MMS, and stand by for an approval to release the player version.

What tools do you need?

By monitoring the network traffic from the video player we can make sure that all events and variables are fired correctly.

Web: Most internet browsers have either built in or add-on capabilities to monitor tracking in web players.

Apps: A proxy server is needed in order to monitor tracking in apps. MMS recommends Charles (freeware).

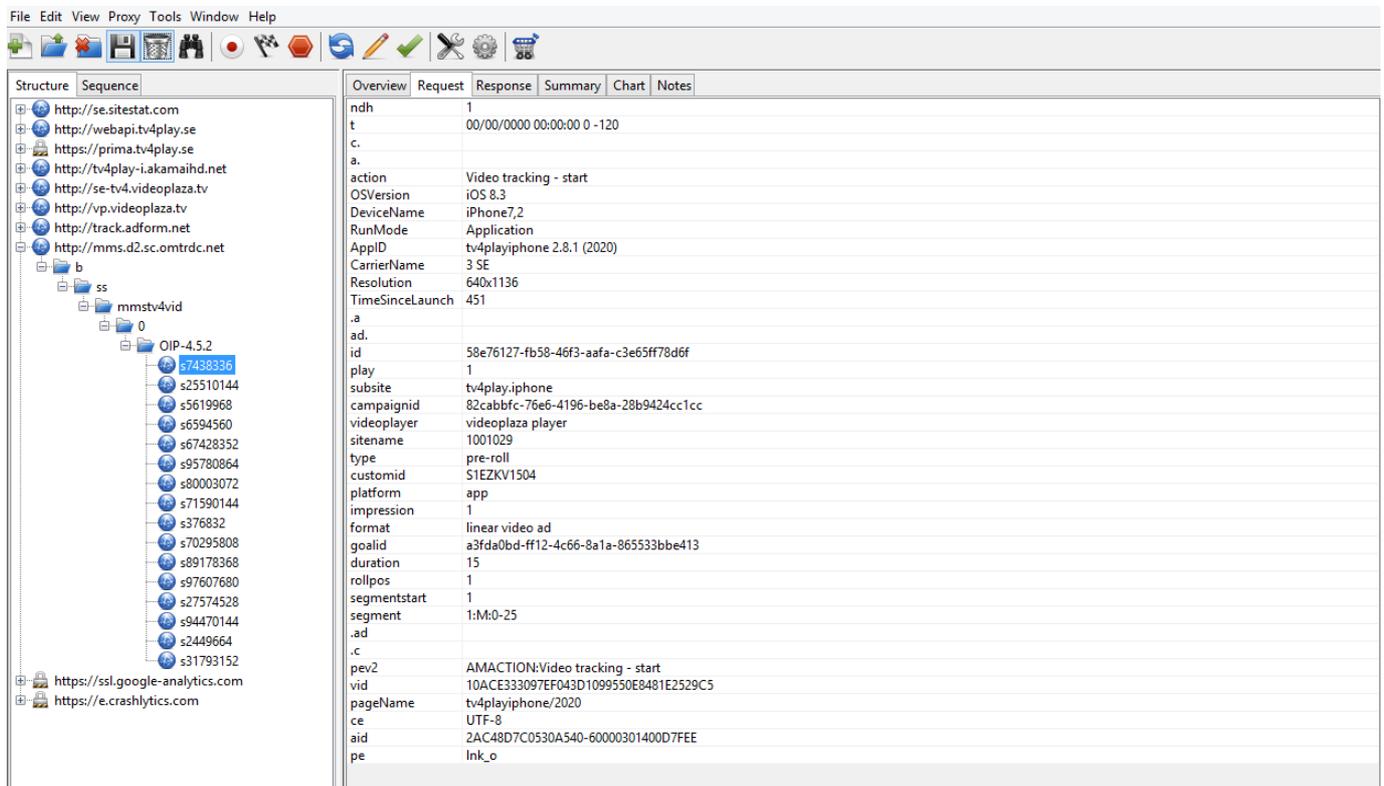


Figure 2: Screenshot from Charles Proxy

Tests

See section “When to trigger events and variables” for list of events and variables to be set.

To avoid certain hard to catch errors make sure that the app/browser is started from scratch, i.e. reboot the phone or restart the browser before performing any tests! It has been noted that measurements can disappear when the app/browser is first started but reappear after usage so make sure that events are fired during the first test performed.

Full ad block viewing (pre/mid/post-roll should all be tested)

Description: Play through all ads in an ad block

Step 1 Start play of programme

Step 2 (Optional) Scrub to mid/post-roll

Step 3 Play through full ad block

Step 4 Terminate app/player after programme/clip has started

Result: See example in “When to trigger events and variables”

User clicks on ad

Step 1 Start play of programme

Step 2 Click on ad

Result: Depends on implementation. If video is paused on click then make sure that the on pause events are fired

User returns to video player from ad

Pre-condition: User has clicked on ad

Step 1 User dismisses ad and returns to video

Result: If video was paused when ad was clicked then make sure that the on resume events are fired

User navigates to other screen in app/player interrupting the clip/programme

Step 1 Start play of clip/programme

Step 2 Click out of playing clip to home page/screen

Result: On pause events are fired

User exits app/player while it is playing

Step 1 Start play of clip/programme



Step 2 Terminate app/player after 2 seconds

Result: On pause events are fired

User pauses ad (if applicable)

Step 1 Start play of clip/programme

Step 2 Press pause after 2 seconds

Result: On pause events are fired

User resumes ad (if applicable)

Pre-condition: Clip/programme has been started and paused

Step 1 Press play

Result: On resume events are fired

NOTE! Remember to test on all relevant browsers!



Contacts

Please reach out to Katrine to request a final validation or for any questions or concerns in regards to the MMS Video Ad tracking.

MMS

Jakob Saros (1st contact)
jakob@mms.se
+46 70 7306229

Mats Lindkvist
mats@mms.se
+46 707960531

Adobe

Katrine Kiildsen (1st contact)
katrine@accrease.com

Henric Dahlberg
dhenric@adobe.com
+45 32316043