Adobe Analytics Video milestone tracking

All you need to know from configuration to implementation

Alexis Cazes

5/26/15

Video implementation

Adobe Analytics and Video Milestones

Adobe Analytics provides two distinct type of video tracking:

- Video Milestone
- Video Heartbeat

This document will go over Video Milestone implementation. The implementation of Video implementation requires that you follow the step in the right order and that you under when to call the correct Media Methods

Requirements

The video player does not matter

Often customers will ask if the video player X is supported for Video Milestone. The answer is simple: No! Video Milestone implementation is not linked to specific video player.

The video player used does not matter. You can if you wish implement Video milestone tracking without using any video player on your webpage or your mobile app. You will just have to set up the media module correctly and call the correct methods at the right times and in the right order. This will allow you to see video data in the video reports.

The steps and logic to implement video milestone for website and mobile differs slightly but the core logic will be similar.

As mentioned above the Video Player does not matter, what matters is the language that you will use to implement Video milestone. Several libraries are available to implement Adobe Analytics, hence to implement Video Milestone. You can find a list of the different implementation on the official website:
https://marketing.adobe.com/resources/help/en_US/sc/appmeasurement/video/video_developer.html

The different type of implementation are:

- Flash
- Open source Media Framework (OSMF)
- ANDROID SDK 4.x
- IOS SDK 4.x
- Windows 8.1 Universal APP SDK 4.x
- Xamarin components SDK 4.x
- JavaScript

In this training I will provide you the steps to implement JavaScript, Android 4.x and IOS 4.x Video Milestone. If you know these 3 Video milestone implementation then you should be able to implement and other video milestone implementation.

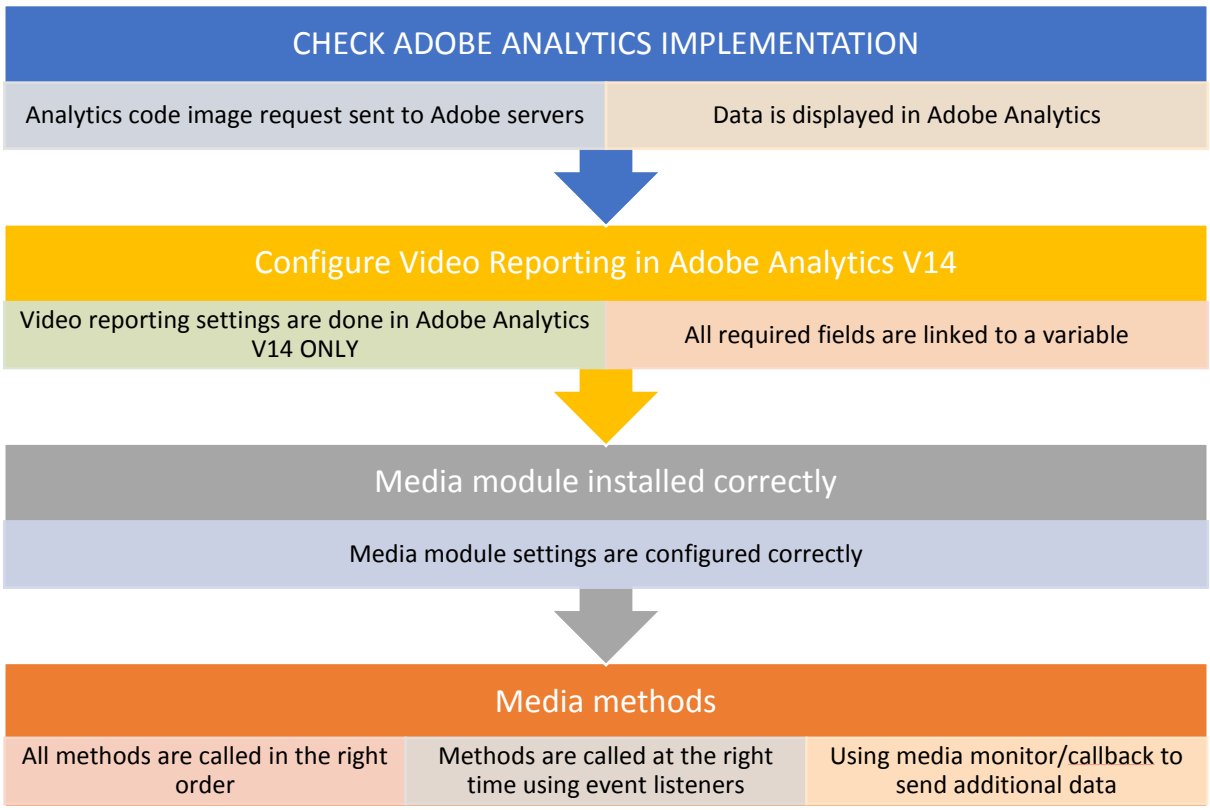
Main steps to implement Video Milestone

There are 4 main steps to have a successful video milestone implementation:

- The Adobe Analytics is implemented successfully and data is displayed in Adobe Analytics!
- The video reporting is configured correctly in **Adobe Analytics V14 !**
- The Media module is implemented successfully with the right settings!

- The Media module methods are called in the right order and at the right time!

The following diagram summaries all that needs to be done:



First Main Step: Check Adobe Analytics implementation.

It is important that your Adobe Analytics implementation is fully functional before that you attempt to implementation Adobe Analytics. Without a successful Adobe Analytics implementation, you most likely will fail your Video Milestone implementation.

It is important that you check that Adobe Analytics image requests are being sent from the page on which you will implement video tracking. It is also important to check for the mobile app implementation that the activity for which you will implement video tracking can send adobe analytics image requests and that lifecycle tracking is implemented successfully.

One of the best way is to use a packet monitor:
https://marketing.adobe.com/resources/help/en_US/sc/implement/packet_monitor.html

Second Main Step: Configure Adobe Analytics Video reporting in V14

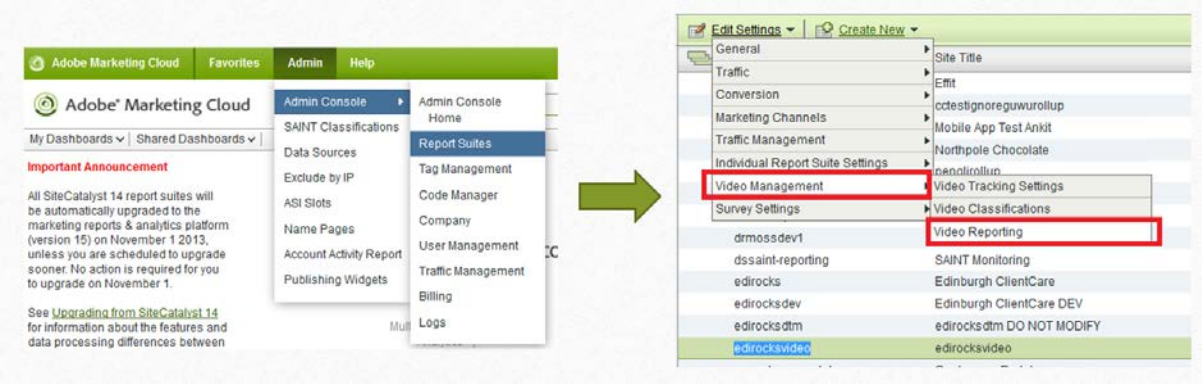
You **MUST** log in to **ADOBE ANALYTICS V14** to configure the video reporting (https://marketing.adobe.com/resources/help/en_US/sc/appmeasurement/video/video_config.html).

Before proceeding to configure the video reporting make sure that you have at least the following amount of each variables:

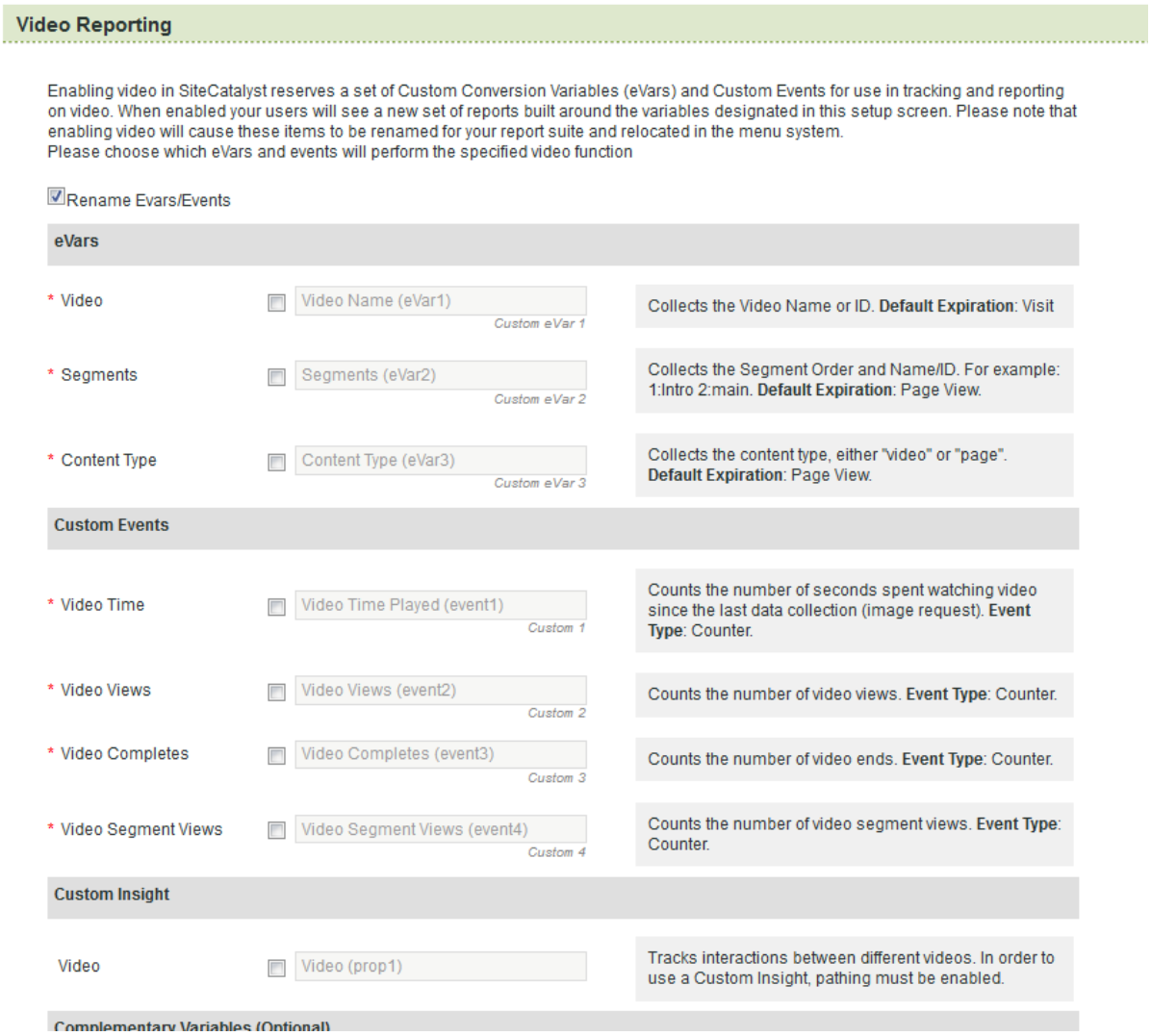
- 3 eVars (required):
  - Video name with default expiration visit
  - Segments with default expiration pageview
  - Content Type with default expiration pageview
- 7 events all configured as counter events (required):
  - Video time
  - Video Views
  - Video Completes
  - Video Segment Views
  - 25% viewed
  - 50% viewed
  - 75% viewed
- 1 prop (optional but advised, useful for real-time debugging):
  - The prop needs to have pathing enabled. It will be used to capture the video name as well
- If you decide to use Ad Video Milestone Tracking then additional variables will need to be used

Steps to access the video reporting settings

1. Connect to **Adobe Analytics v14**
2. Go to Admin ➔ Admin Console ➔ Report suites
3. Select Report suite
4. Click on Edit Settings ➔ Video Management ➔ Video reporting



Once you are in the Video reporting report make sure to map the correct variables:



Known issue

If you have accessed Video reporting settings from **Adobe Analytics V15** and you have enabled video reporting in V15 prior to enabling them in **Adobe Analytics V14**, please be aware that backend processing rules will be enabled and some of these processing rules will remain even after that you have enabled Video Reporting in **V14**. This may result in inconsistency in your reporting between what you send and what is displayed in Adobe Analytics.

The backend processing rules will be based on adobe reserved contextData variables, contained between c. a. and .a .c.

The best way to fix your issue (for JavaScript) would be to pass the correct values directly in the reserved Adobe Analytics variables:

e.g:

```
function s_doPlugins(s) {

s.contextData['a.contentType']="My content Type";

}
s.doPlugins=s_doPlugins
```

Create processing rules for mobile video milestone tracking.

When you implement video milestone tracking in any of the mobile apps using SDK 4.x, all the video data will be send in Adobe reserved variables (contextData) contained between c. a. and .a .c in the image request. For the data to be displayed in Adobe Analytics processing rules will need to be created. To be able to use processing rules you will need to be enabled. You will need to pass an exam: https://marketing.adobe.com/resources/help/kb/en_US/analytics/kb/processing-rules-authorization.html

If you use Adobe Mobile Services then you need to be an admin to make the mapping under Custom data content:

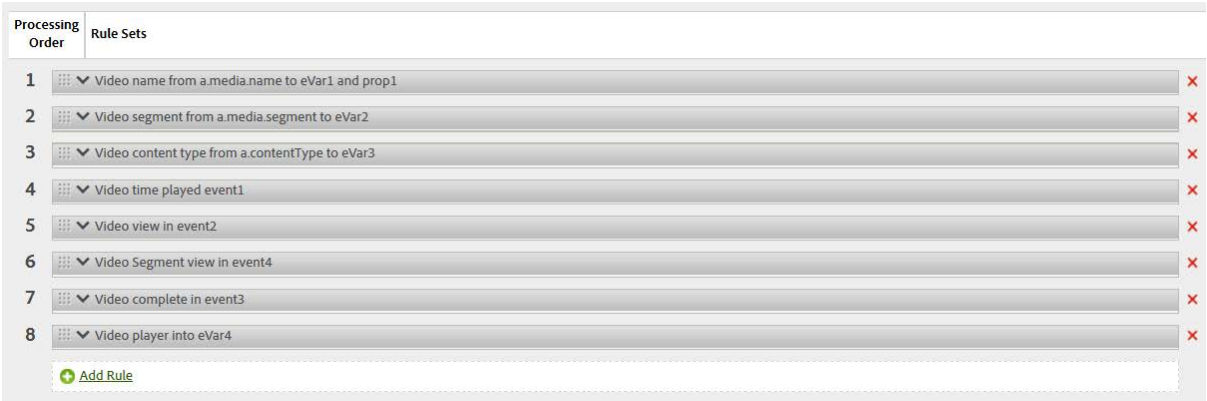
https://marketing.adobe.com/resources/help/en_US/mobile/settings_custom_data.html

The rules will be used to map the adobe reserved media variables to the video reporting variables defined in **Adobe Analytics V14**.

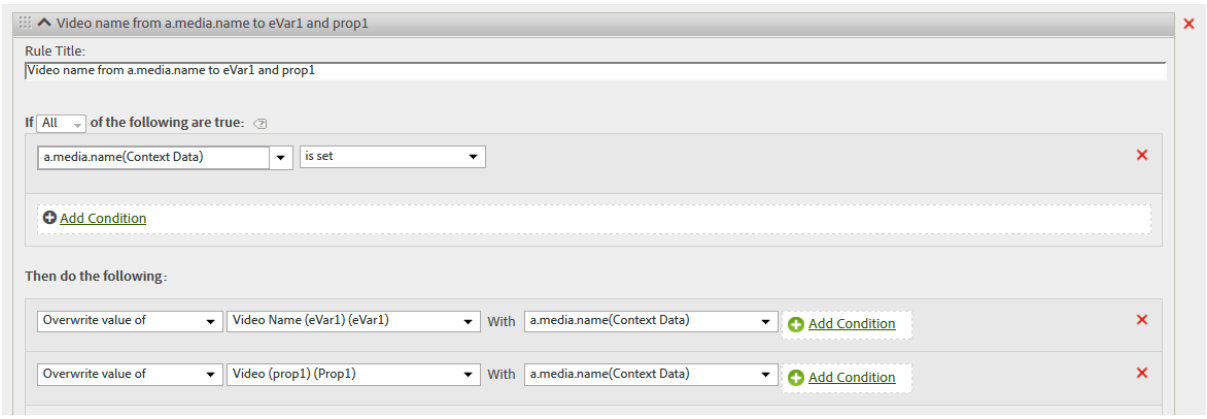
Note: If you decide to not do all the variables mapping in the JavaScript implementation then Media data will be sent in the same format as in the mobile implementation and will require processing rules so the data is displayed in Adobe Analytics. This can be useful if you have different video reporting mapping across different report suites (multi-suite tagging) and you want to use processing rules to make sure to map the correct data to the correct video report. This can also be used to have identical mobile and website implementation.

Example of processing rule to map mobile video variables to video reports

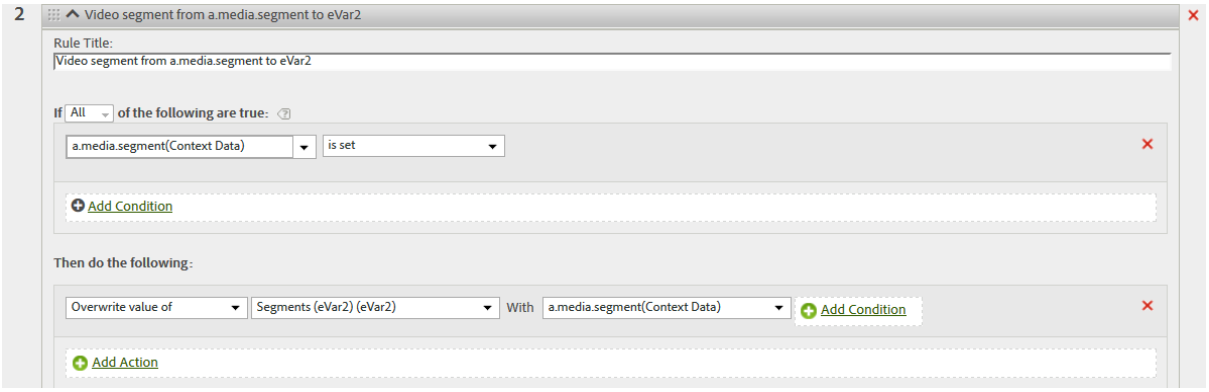
Main Mapping



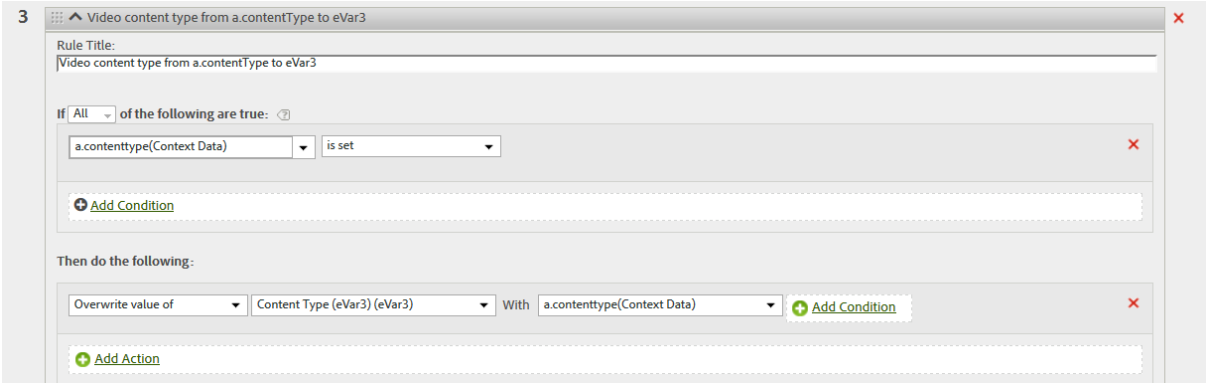
Video name



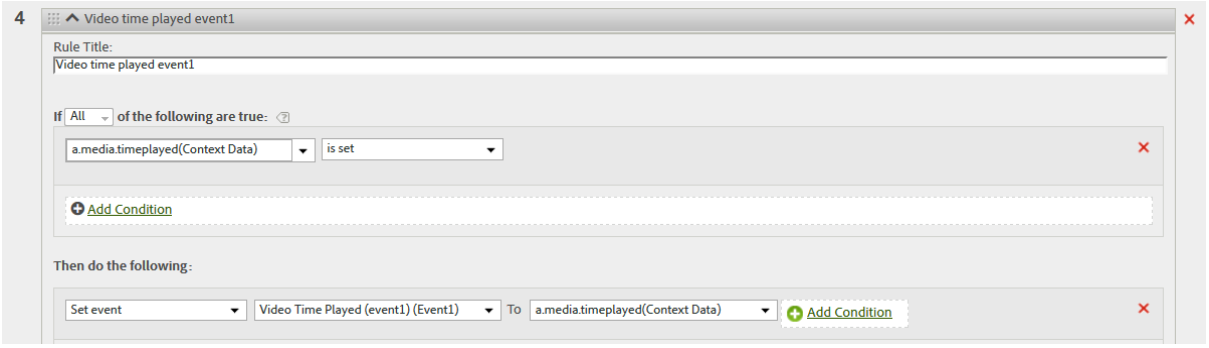
Video Segment



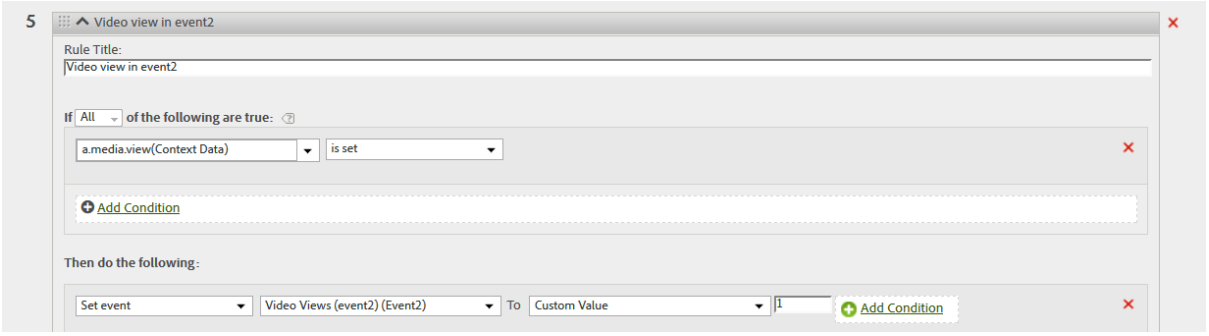
Video content type



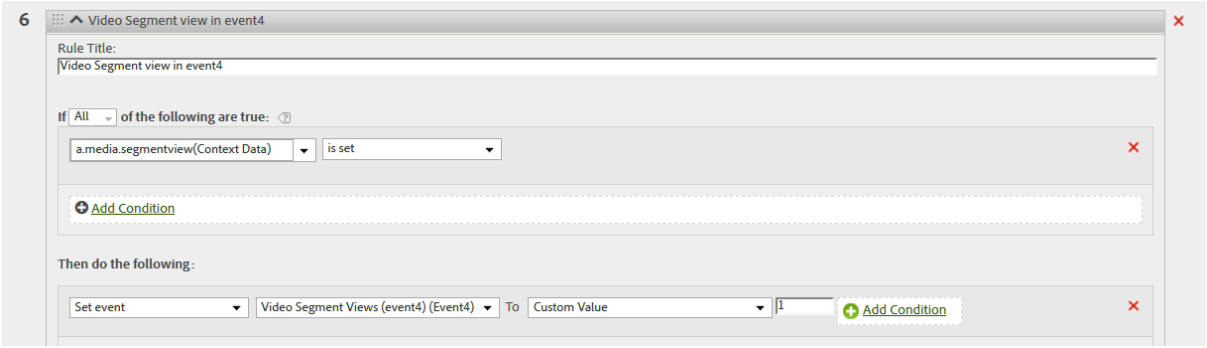
Video time played



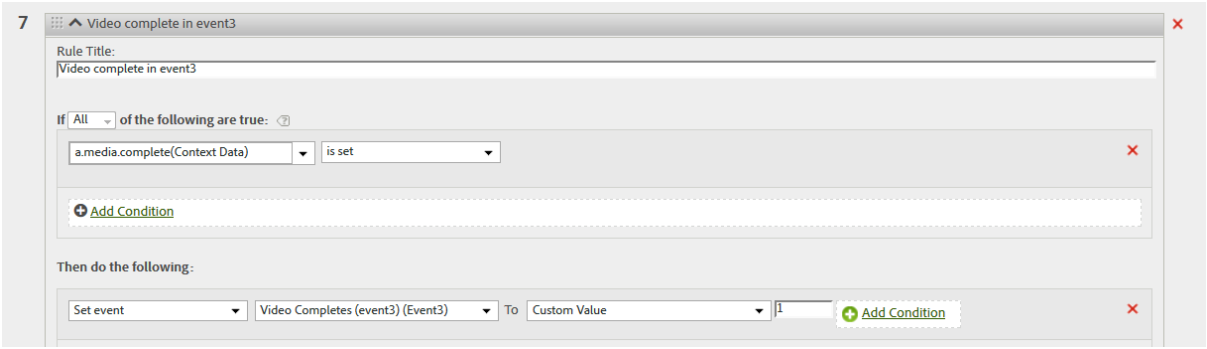
Video view



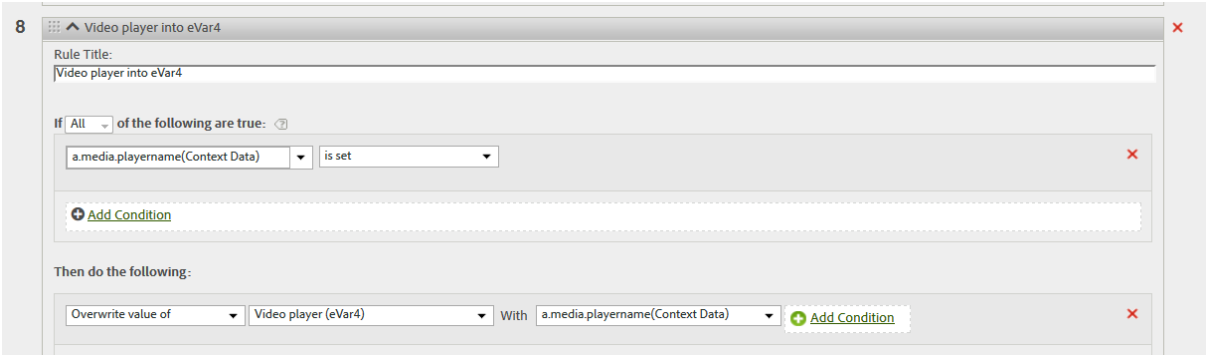
Video segment view



Video complete

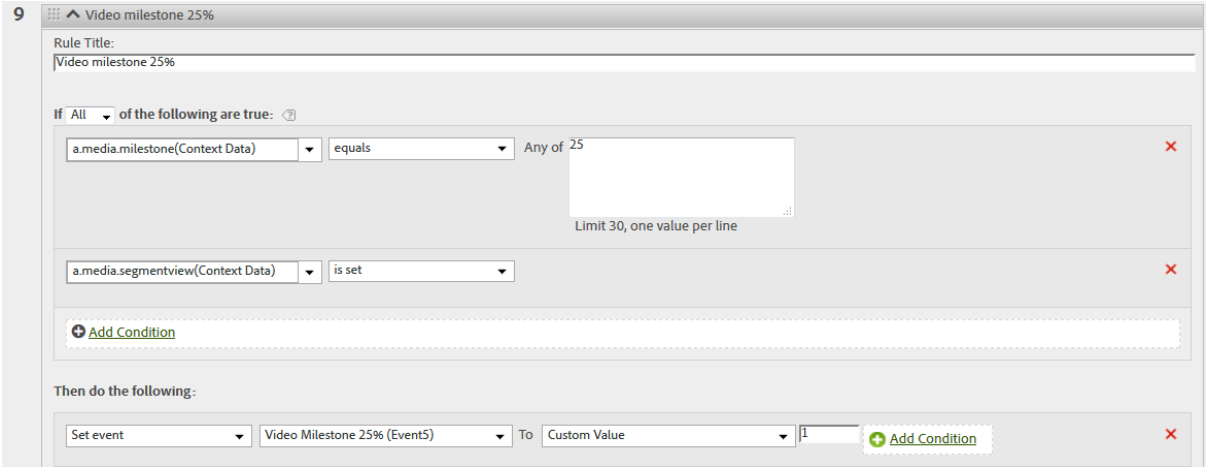


Video Player

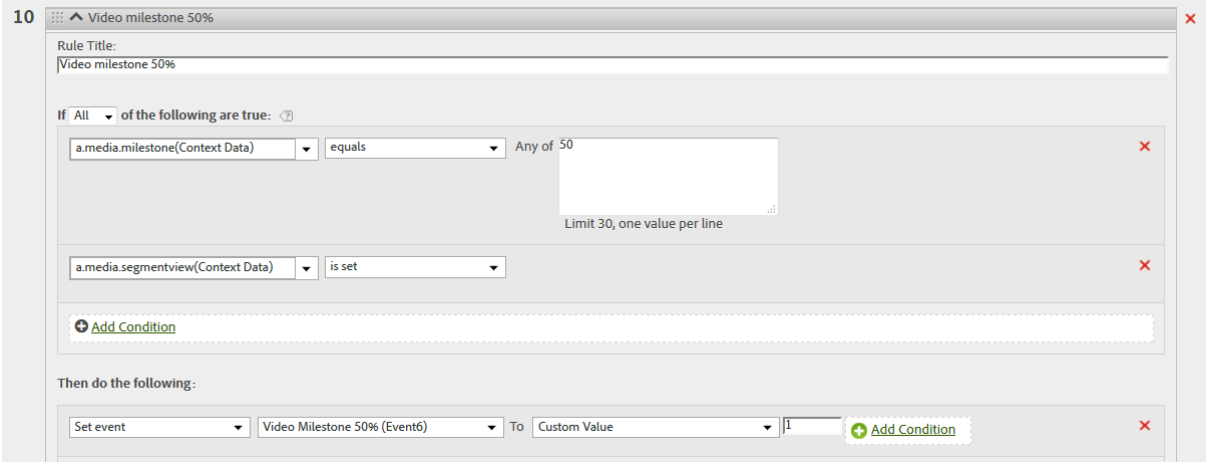


Video milestones

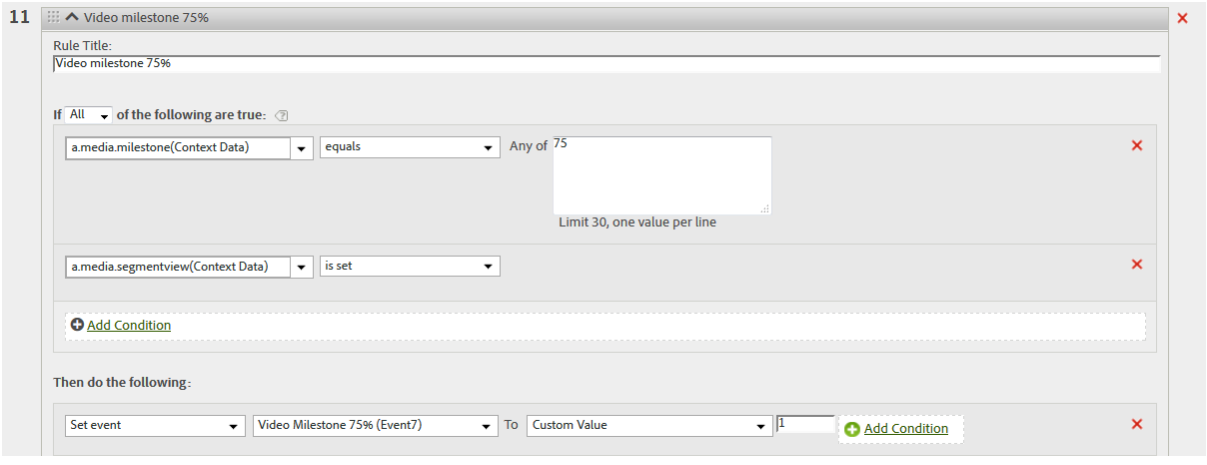
Video milestone 25%



Video milestone 50%



Video milestone 75%



Third Main Step: Configure the Media Module

Choose the right tracking library for your video player

As mentioned above the video player used does not matter but the language of the code used for the video player does matter. You will find that there are a multitude of players out there:

- Flash
- HTML5
- Hybrid Flash + HTML5
- Mobile APP video players
- Other : Silverlight, Windows Media, QuickTime
- Partner players
- Paid providers : Brightcove, Ooyala, Kaltura, The platform
- Open source: JW Player, FlowPlayer
- Free providers : Youtube, Vimeo, Facebook
- Custom: Home Built Players

For website usage if you can implement an HTML5 video player using the JavaScript AppMeasurement library and a flash player then you should be able to implement Video Milestone tracking for any players out there. Some of the video players out there are using Flash but also provide a JavaScript API. This could allow you to create reusable JavaScript function that can be called to send Adobe Analytics media milestone data across Video players!

For mobile if you can implement an Android App SDK 4.x Video milestone tracking then you should be able to implement for any Video Milestone SDK 4.x tracking that Adobe Analytics provides.

Website video milestone media module configuration

AutoTrack vs Manual Tag

When you implement video milestone tracking using the JavaScript AppMeasurement or the JavaScript H code, you will come across the option to track the video milestone using autotrack or using manual tagging.

Our Media module needs to be told when:

- The video is beginning
- The video has ended
- Someone paused or resumed the video
- The video has paused to buffer and when the buffering is done
- The use has skipped ahead/back

All of these information is handled by the video player's event handler.

Autotrack will be when our media module "already knows" which event listeners to listen for, and therefore the media module will know when the send the right data.

Manual tagging will be used when our media module is unable to figure out what is happening at the video player level. A developer will have to add the correct media module methods to be called at the right time during the video player lifecycle.

Autotrack is only supported by a handful of players for our media module. Most of them use flash.

You will have to use manually tracking in 99% of the cases:

- The player is not supported by AutoTrack
- Classifying a video ID to a friendly name isn't a viable option (if you have to classify more than 10 video names then you should opt for manual tagging)
- Live Streams (video with no defined end)
- Use of custom segment names
- And finally if AutoTrack simply refuses to work.

Media Module Settings

Mobile settings for SDK 4.x

All of the mobile SDK 4.x follow the same logic to initialise the media module settings. You will find below the steps to take to setup media module for Android SDK 4.x and IOS SDK 4.x. You will notice that in both cases the steps are similar

IOS SDK 4.x

First of all, all the details are described in the official IOS SDK 4.x documentation:

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

First you will need to call:

```
ADBMediaSettings *mediaSettings = [ADBMobile mediaCreateSettingsWithName:MEDIA_NAME
length:MEDIA_LENGTH playerName:PLAYER_NAME playerID:PLAYER_ID];
```

Media name, media length, media player and player id are required parameters.

Once you called the above, choose if you want to track the standard milestone which correspond to percentage:

```
mediaSettings.milestones = @"25,50,75";
mediaSettings.segmentByMilestones = YES;
```

Or you want to track offset milestone which correspond to specific seconds that elapsed since the video started to play:

```
mediaSettings.offsetMilestones = @"60,120";
mediaSettings.segmentByOffsetMilestones = YES;
```

If you want a hit to be sent every X seconds then implement the following:

```
mediaSettings.trackSeconds = 30; // sends a hit every 30 seconds
```

THIS IS ALL THE MEDIA MODULE CONFIGURATION THAT IS NEEDED. YOU CAN MIX AND MATCH THE OPTIONS BUT THEY NEED TO BE PRESENT BEFORE THE NEXT STEP.

ANDROID SDK 4.x

First of all, all the details are described in the official IOS SDK 4.x documentation:

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

First you will need to call:

```
MediaSettings settings =Media.settingsWith(MEDIA_NAME, MEDIA_LENGTH, PLAYER_NAME,
PLAYER_ID);
```

Media name, media length, media player and player id are required parameters.

Once you called the above, choose if you want to track the standard milestone which correspond to percentage:

```
settings.milestones = "25,50,75";
settings.segmentByMilestones = true;
```

Or you want to track offset milestone which correspond to specific seconds that elapsed since the video started to play:

```
settings.offsetMilestones = "60,120";
settings.segmentByOffsetMilestones  = true;
```

If you want a hit to be sent every X seconds then implement the following:

```
settings.trackSeconds = 30; // sends a hit every 30 seconds
```

THIS IS ALL THE MEDIA MODULE CONFIGURATION THAT IS NEEDED. YOU CAN MIX AND MATCH THE OPTIONS BUT THEY NEED TO BE PRESENT BEFORE THE NEXT STEP.

Web media module settings

Download the media module

First of all you will need to download the media module code. In some versions of H code it is not present by default and in the AppMeasurement.js, the media module is in a separate JavaScript file.

Follow the following steps:

- Go to Admin ➔Code manager
  - AppMeasurement JavaScript: in the zip file downloaded you should see "AppMeasurement_Module_Media.js" ☐ copy to AppMeasurement.j
  - Legacy JavaScript code (H version): should be present in the s_code.js file
  - For Flash add the corresponding library as present in the zip folder.
- For JavaScript implementation place the Media module code above the line DO NOT ALTER BELOW THIS LINE either in the s_code.js file of AppMeasurement.js file.
- **Initialise the Media module : s.loadModule("Media")**

You can find the documentation for all code versions for video milestone implementation on the official documentation:
https://marketing.adobe.com/resources/help/en_US/sc/appmeasurement/video/video_developer.html

Unlike the mobile video milestone implementation, the default implementation requires that you create the variables mapping.

Variable mapping

For the web implementation you need to reference required variables in your implementation. Most of the variables are required. For the default implementation, you will have to map contextData variable to the variables used in the video reporting in **Adobe Analytics V14.** It is also possible to not use the variable mapping for advance reporting. This will be treated in the next section.

Descript of the different variables:

- s.Media.autoTrack= true; ➔ set to true when using AutoTrack. In most of the case set to false for manual tracking
- s.Media.trackVars="events,prop2,eVar1,eVar2,eVar3";
- s.Media.trackEvents="event1,event2,event3,event4,event5,event6,event7“
    - All variables set in video reporting in Adobe Analytics v14 should be listed in the 2 variables above
- s.Media.trackMilestones="25,50,75"; ➔ declare which milestone you want to track
- s.Media.playerName="My Media Player"; ➔ name of the player, not tracked by default in any report but can be tracked using processing rules
- s.Media.segmentByMilestones = true; ➔ Tells if when milestone met send image request or not
- s.Media.trackUsingContextData = true; ➔always set to true
- s.Media.contextDataMapping = { […] } ➔Map all the variables declared in Video reporting settings

Map video reporting settings from Adobe Analytics V14 to s.Media.contextDataMapping

All of the variables used in Adobe Analytics V14 video reporting needs to be mapped in the code. You will need to map these variable in s.Media.contextDataMapping. All the eVars, events and props present in the video reporting in V14 needs to be mapped. Only the additional eVars, props and events under additional reporting in the video reporting interface will not need to be mapped here. They will be used in Media.monitor

Example:

```
"a.media.name":"eVar1,prop1",
"a.media.segment":"eVar2",
"a.contentType":"eVar3",
"a.media.timePlayed":"event1",
"a.media.view":"event2",
"a.media.segmentView":"event4",
"a.media.complete":"event3",
"a.media.milestones":{
        25:"event5",
        50:"event6",
        75:"event7"
}
```

Example of image requests:

- First Play (Start of video):

```
c.
a.
media.
playerName=My Media Player
length=96.711111
.media
.a
.c
pe=m_s
pev3=video
events=event2
c1=testhtml5video
v1=testhtml5video
v2=1:M:0-25
v3=video
```

- Milestone 25%:

```
c.
a.
media.
playerName=My Media Player
length=96.711111
.media
.a
.c
pe=m_i
pev3=video
events=event1=24,event4,event5
c1=testhtml5video
v1=testhtml5video
v2=1:M:0-25
v3=video
```

- Milestone 50%:

```
c.
a.
media.
playerName=My Media Player
length=96.711111
.media
.a
.c
pe=m_i
pev3=video
events=event1=24,event4,event6
c1=testhtml5video
v1=testhtml5video
v2=2:M:25-50
v3=video
```

- Milestone 75%:

```
c.
a.
media.
playerName=My Media Player
length=96.711111
.media
.a
.c
pe=m_i
pev3=video
events=event1=24,event4,event7
c1=testhtml5video
v1=testhtml5video
v2=3:M:50-75
v3=video
```

- Video complete:

```
c.
a.
media.
playerName=My Media Player
length=96.711111
.media
.a
.c
pe=m_i
pev3=video
events=event1=24,event4,event3
c1=testhtml5video
v1=testhtml5video
v2=4:M:75-100
v3=video
```

No variable mapping

If you omit a variable to be mapped in s.Media.contextDataMapping then this variable will be present in the image request in c. a.[variable name]. The variables in the image request will be contextData variables using the Adobe reserved variable a. processing rules will be required to map these variables to the video reports. The processing rules will be similar to the one used for mobile video variable.

Example of image requests:

- First Play (Start of video):

```
c.
a.
contentType=video
media.
name=testhtml5video
playerName=My Media Player
length=96.711111
view=true
segmentNum=1
segment=M:0-25
.media
.a
.c
pe=m_s
pev3=video
```

- Milestone 25%:

```
c.
a.
contentType=video
media.
name=testhtml5video
playerName=My Media Player
length=96.711111
timePlayed=25
segmentNum=1
segment=M:0-25
segmentView=true
milestone=25
.media
.a
.c
pe=m_i
pev3=video
```

- Milestone 50%:

```
c.
a.
contentType=video
media.
name=testhtml5video
playerName=My Media Player
length=96.711111
timePlayed=24
segmentNum=2
segment=M:25-50
segmentView=true
milestone=50
.media
.a
.c
pe=m_i
pev3=video
```

- Milestone 75%:

```
c.
a.
contentType=video
media.
name=testhtml5video
playerName=My Media Player
length=96.711111
timePlayed=24
segmentNum=3
segment=M:50-75
segmentView=true
milestone=75
.media
.a
.c
pe=m_i
pev3=video
```

- Video complete:

```
c.
a.
contentType=video
media.
name=testhtml5video
playerName=My Media Player
length=96.711111
timePlayed=23
segmentNum=4
segment=M:75-100
segmentView=true
complete=true
.media
.a
.c
pe=m_i
pev3=video
```

You will notice that for the success events like a.media.segmentViews, a.media.view and a.media.complete, the value sent is not a number but a Boolean. This means that for processing rules you need to make that you set the condition to check if this variable returns true and if does increment the corresponding event by one and NOT assign the variable value to the event.

List of Adobe video reserved variables.

Context Data Variables	Description
a.contentType	Collects data about the type of content viewed by a visitor. Hits sent by video measurement are assigned a content type of "video".
a.media.name	Name of the video
a.media.playerName	Name of the video player used
a.media.length	Full length of the video
a.media.timePlayed	Counts the time, in seconds, spent watching a video since the last data collection process
a.media.segmentNum	Number of the video segment
a.media.segment	Name of the video segment
a.media.milestone	Specific milestone number

a.media.segmentView	Indicates that a visitor has viewed some portion of a video segment. However, it does not provide any information about how much, or what part, of a video the visitor viewed.
a.media.view	Indicates that a visitor has viewed some portion of a video. However, it does not provide any information about how much, or what part, of a video the visitor viewed.
a.media.complete	Indicates that a user has viewed a complete video. By default, the complete event is measured 1 second before the end of the video.

Modify content of reserved variables dynamically

Some of the reserved contextData return predefined values. For example a.contentType will always return “video” unless you specify a custom value. Not all the variables returns default values.

As a.contentType is set after that Media.monitor is ran you cannot define a new value for a.contentType in Media.monitor.

The solution is to set the custom value using the doPlugins function in your AppMeasurement.js file. This function is ran every single time an image request is sent to Adobe Analytics servers. As all of the video variables (with no variable mapping) are send via Adobe reserved variables, you will need to use s.contextData[‘NAME OF VARIABLE’] to change the value of a specific variable.

In my example, I have decided to change the contentType, but you could also decide to happen a specific value before or after the video name to specific from which site section the video comes from.

Code example:

```
function s_doPlugins(s) {
        s.contextData['a.contentType']="Test content Type video";
}
```

Fourth and Last Main Step: Send Media Data to Adobe Analytics

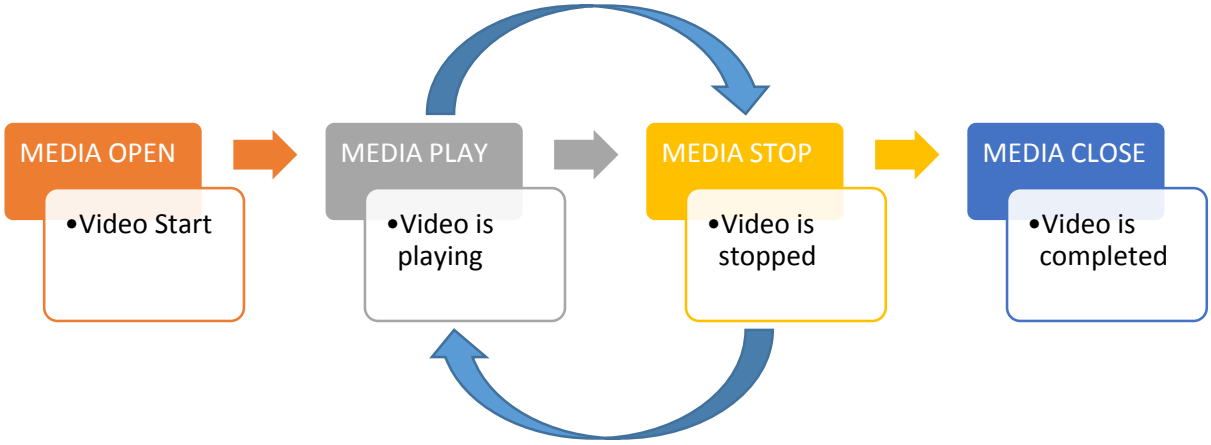
Now that the video reporting is configured successfully in **Adobe Analytics V14** and that the media settings are configured correctly, we are ready to use the media module methods to send video data to Adobe Analytics for reporting.

Before that we explain for what each methods are used for, we need to understand in which order the methods needs to be called.

Media methods workflow

The media methods will always be called in a specific order. If the methods are called out of order then the video tracking will not work successfully and/or the data sent to Adobe Analytics will be incorrect.

Media module workflow diagram:



- When the video is opened/started for the first time, Media Open will be called followed by Media Play
- Every single time the video is played, the Media Play needs to be called. (Make sure to specify the correct offset of the video)
- Every single time the video is stopped, the Media Stop needs to be called. (Make sure to specify the correct offset of the video)
- Once the video is completed, call Media Stop followed by Media Close
- Media Open and Media Close will send an image request to Adobe Analytics. Media Play and Media Stop do not send an image request to Adobe Analytics. If you want send an image request when these methods are called, you need to use Media Monitor.

Here is a table that summarise what you need to call and when you use manual tagging.

Function to call	When to call it	How often to call per Video View	Send image request to Adobe Analytics
Media Open	Video Load/First Play	1	YES
Media Play	Any time the video begins to play, resume, buffer completed, scrub release or segment change	As often as needed	NO
Media Stop	Any time the video stops, pauses, buffers, scrubber grab or segment changes	As often as needed	NO
Media Close	Video Complete	1	Yes

Note: If you are using video ad tracking, you will also need to use the corresponding video ad open, video ad play, video ad stop and video ad complete. These methods follow the same logic as default media methods.

You can find an example of Video ad tracking here:

https://marketing.adobe.com/resources/help/en_US/sc/appmeasurement/video/video_ads.html

Open the Media Tracking

Every single time video is opened/started for the first time the Media Open methods need to be called. Media Open will always be followed by Media Start methods.

Mobile

For mobile implementation, Media Open can be called with or without a callback. The callback will correspond to Media Monitor in the mobile world.

IOS SDK 4.x

Media open without callback:

```
[ADBMobile mediaOpenWithSettings:mediaSettings callback:nil];
```

Media open with callback:

```
[ADBMobile mediaOpenWithSettings:settings callback:^(ADBMediaState *state) {
    NSLog(@"mediaEvent = %@", state.mediaEvent);
    if ([state.mediaEvent isEqualToString:@"PLAY"]) {
      [ADBMobile mediaTrack:MEDIA_NAME data:@{@"myKey":@"myVal"}];
    }
    else if (state.milestone == 50) {
      [ADBMobile mediaTrack:MEDIA_NAME data:@{@"MyMilestone":@"50"}];
    }
  }];
```

Android SDK 4.x

Media open without callback:

```
Media.open(settings, null);
```

Media open with callback:

```
Media.open(mySettings, new Media.MediaCallback<MediaState>() {
@Override
        public void call(MediaState state) {
            if(state.mediaEvent ==PLAY && state.eventFirstTime){
                //DO SOMETHING!
            }
            if(state.Milestone == 50){
                //DO SOMETHING
            }
        }
    });
```

Website

JavaScript

Example of code:

```
s.Media.open(mediaName,mediaLength,mediaPlayerName);
```

Parameters:

- **mediaName**: (required) The name of the video as you want it to appear in video reports.
- **mediaLength**: (required) The length of the video in seconds.
- **mediaPlayerName**: (required) The name of the media player used to view the video, as you want it to appear in video reports.

Media Play

Media play will be called when the video start and thereafter each time the video is playing.

Note:

When the video starts the offset needs to be set to zero

When Media play is called at any other time that the video start, you need to specify the correct offset.

IOS SDK 4.x

Video start:

```
[ADBMobile mediaPlay:MEDIA_NAME offset:0];
```

Other than video start:

```
[ADBMobile mediaPlay:MEDIA_NAME offset:mediaController.currentPlaybackTime];
```

ANDROID SDK 4.x

Video start:

```
Media.play(MEDIA_NAME, 0);
```

Other than video start:

```
Media.play(MEDIA_NAME, mediaPlayer.getCurrentPosition());
```

JavaScript

Video start:

```
s.Media.play(name,0);
```

Other than video start:

```
s.Media.play(name,offset);
```

Media Stop

Media Stop need to be called each time the video is stopped for any reasons. It should always be called when the video is completed just before the Media Close. The correct offset when it stops needs to be provided.

IOS SDK 4.x

```
[ADBMobile mediaStop:MEDIA_NAME offset:mediaController.currentPlaybackTime];
```

ANDROID SDK 4.x

```
Media.stop(MEDIA_NAME, mediaPlayer.getCurrentPosition());
```

JavaScript:

```
s.Media.stop(mediaName,mediaOffset);
```

Media Close

Media close should be called when the video is completed. It should be called just after Media Stop. Once called, the video tracking stops and a complete event is sent to Adobe Analytics.

IOS SDK 4.x

```
[ADBMobile mediaStop:MEDIA_NAME offset:mediaController.currentPlaybackTime];

[ADBMobile mediaClose:MEDIA_NAME];
```

Android SDK 4.x

```
Media.stop(MEDIA_NAME, mediaPlayer.getCurrentPosition());

Media.close(MEDIA_NAME);
```

JavaScript

```
s.Media.stop("testhtml5video",currentTime);

s.Media.close("testhtml5video");
```

Media Monitor

During your video implementation you might feel the need to send additional data with your video calls. For example if your app or website show videos about a specific series, you might want to send the series number and the episode number in the Adobe Analytics call.

One of the solution would be to concatenate this data into one sting and send it as the video name and then use classification to classify the data in the correct reports, but due to the fact that the video variable is limited to 255 char it might not be possible. You will then need to send these data in separate eVars.

To do so you can use Media monitor, which allows you to send additional variables and events with your video adobe analytics server calls.

For a website implementation using JavaScript, you should place the Media Monitor code in the same file as the media module (s_code.js or AppMeasurement.js).

Media Monitor is used in conjunction with:

- Media.trackVars: to add the additional variables being sent
- Media.trackEvents: to add the additional events being sent
- Media.track: used for media calls that are not media open or media close.
- Doc: https://marketing.adobe.com/resources/help/en_US/sc/appmeasurement/video/video_mediamonitor.html

For a mobile implementation you will need to make sure to use Media open with a callback. This will allow you to specify additional logic to send additional variables.

In all the case Media track should be used if it is not a Media Open or Media Close.

Media variables usable with media monitor

There are several variable that are available to you to use with Media monitor. This will allow to send data only when a specific criteria are met like specific video is playing or video has reached a specific milestone.

A detailed list is provide in this documentation:
https://marketing.adobe.com/resources/help/en_US/sc/appmeasurement/video/video_mediamonitor.html

JavaScript example:

```
s.Media.monitor = function (s,media){

        //When Media Open is called

        if(media.event=="OPEN") {

                console.log("Media open");
```

```
        }
        //When Media Play is called
        if(media.event=="PLAY") {
                console.log("Media play");
                if(s.Media.trackEvents.indexOf('event8') < 0){
                        s.Media.trackEvents = s.Media.trackEvents + ",event8";
                }
                s.events = "event8";
                s.Media.track(media.name);
  }
        //When Media Stop is called
        if(media.event=="STOP") {
                console.log("Media stop");
                if(s.Media.trackEvents.indexOf('event9') < 0){
                        s.Media.trackEvents = s.Media.trackEvents + ",event9";
                }
                s.events = "event9";
                s.Media.track(media.name);
        }
        //When Media Close is called
        if(media.event=="CLOSE") {
                console.log("Media completed");
        }
}
```

IOS SDK 4.x

All of variable usable in the callback can be found here:
https://marketing.adobe.com/resources/help/en_US/mobile/ios/video_qs.html

Class: ADBMediaState

```
1.  bool ad;
2.  bool clicked;
3.  bool complete;
4.  bool eventFirstTime;
5.  double length;
6.  double offset;
7.  double percent;
8.  double segmentLength;
9.  double timePlayed;
10. double timePlayedSinceTrack;
11. double timestamp;
12. NSDate *openTime
13. NSString *name
14. NSString *playerName
15. NSString *mediaEvent
16. NSString *segment
17. NSUInteger milestone
18. NSUInteger offsetMilestone
19. NSUInteger segmentNum
20. NSUInteger eventType
```

```
[ADBMobile mediaOpenWithSettings:settings callback:^(ADBMediaState *state) {
        NSLog(@"mediaEvent = %@", state.mediaEvent);
        if([state.mediaEvent isEqualToString:@"PLAY"]) {
                [ADBMobile mediaTrack:MEDIA_NAME data:@{@"myKey":@"myVal"}];
        }else if(state.milestone == 50){
                [ADBMobile mediaTrack:MEDIA_NAME data:@{@"MyMilestone":@"50"}];
        }
}];
```

Android SDK 4.x:

All of the variables usable in the callback can be found here:

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

Class: MediaState

```
1.  public Date openTime = new Date();
2.  public String name;
3.  public String segment;
4.  public String playerName;
5.  public String mediaEvent;
6.  public int offsetMilestone;
7.  public int segmentNum;
8.  public int milestone;
9.  public double length;
10. public double offset;
11. public double percent;
12. public double timePlayed;
13. public double segmentLength;
14. public boolean complete = false;
15. public boolean clicked = false;
16. public boolean ad;
17. public boolean eventFirstTime;
```

```
Media.open(mySettings, new Media.MediaCallback(MediaState state) {
@Override
        public void call(state) {
                if(state.mediaEvent == OPEN){
                        //DO SOMETHING!
                }
                if(state.Milestone == 50){
                        //DO SOMETHING
                }
        }
});
```

Data Insertion API and Video tracking

During your implementation phase you might be faced with the issue that none of the Adobe Analytics library can be used to track user's interaction. (For example tracking a gaming console like PS4 or XBOX One).

At this specific point you would have opted to use Adobe Analytics Data Insertion API. For more information about data insertion API, please check:

https://marketing.adobe.com/developer/documentation/data-insertion/c-data-insertion-api

Using the data insertion API, you can either decide to send video data in the variables defined in Video Reporting in **Adobe Analytics V14** or you can use the Adobe video reserved variables.

You can find below an example of the xml file that you can send for each solution.

Insert data using the video reporting variables

For the test below, please download curl and create an event.xml file with the details below. Open a command prompt and navigate to the folder where curl.exe is present.

Use the following command to send the hit:

```
curl -X POST -A "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_8) AppleWebKit/534.30 (KHTML, like Gecko) Chrome/12.0.742.112 Safari/534.30" -H "Content-Type: text/xml" --http1.0 -v -d @event.xml http://cardgagecorp.112.2o7.net/b/ss//6
```

Note: notice that the curl command uses a correct User-Agent. The default curl User-Agent will be considered as a bot if IAB bot filtering is applied

Note: All of the examples below are using a timestamp in the request and the report suite is configured to accept time stamped hits.

Example Video Open (First Play of Video)

```
<?xml version=1.0 encoding=UTF-8?>

<request>

        <sc_xml_ver>1.0</sc_xml_ver>

        <events>event5</events>

        <pageURL>testVideo.html</pageURL>

        <pageName>Video CURL</pageName>

        <visitorID>videoUSERID1</visitorID>

        <contextData>

                <a>

                        <contentType>Video API</contentType>
```

```
                        <media>
                                <name>VIDEO API</name>
                                <playerName>VIDEO API Player</playerName>
                                <length>96.711111</length>
                                <view>true</view>
                                <segmentNum>1</segmentNum>
                                <segment>M:0-25-API</segment>
                        </media>
                </a>
        </contextData>
        <linkType>m_s</linkType>
        <timestamp>1432641844</timestamp> >
        <reportSuiteID>lscsalexisandrdoidvideotest</reportSuiteID>
</request>
```

Example Milestone tracked

```
<?xml version=1.0 encoding=UTF-8?>
<request>
        <sc_xml_ver>1.0</sc_xml_ver>
        <events>event5</events>
        <pageURL>testVideo.html</pageURL>
        <pageName>Video CURL</pageName>
        <visitorID>videoUSERID1</visitorID>
        <contextData>
                <a>
                        <contentType>Video API</contentType>
                        <media>
                                <channel></channel>
                                <name>VIDEO API</name>
                                <playerName>VIDEO API Player</playerName>
                                <length>96.711111</length>
                                <timePlayed>25</timePlayed>
```

```
                    <segmentNum>1</segmentNum>
                    <segment>M:0-25-API</segment>
                    <segmentView>true</segmentView>
                    <milestone>25</milestone>
                </media>
            </a>
        </contextData>
        <linkType>m_i</linkType>
        <timestamp>1432641880</timestamp> >
        <reportSuiteID>lscsalexisandrdoidvideotest</reportSuiteID>
</request>
```

Example video complete

```
<?xml version=1.0 encoding=UTF-8?>
<request>
        <sc_xml_ver>1.0</sc_xml_ver>
        <events>event5</events>
        <pageURL>testVideo.html</pageURL>
        <pageName>Video CURL</pageName>
        <visitorID>videoUSERID1</visitorID>
        <contextData>
            <a>
                <contentType>Video API</contentType>
                <media>
                    <channel></channel>
                    <name>VIDEO API</name>
                    <playerName>VIDEO API Player</playerName>
                    <length>96.711111</length>
                    <timePlayed>23</timePlayed>
                    <segmentNum>4</segmentNum>
                    <segment>M:75-100-API</segment>
                    <segmentView>true</segmentView>
```

```
                                    <complete>true</complete>
                                </media>
                            </a>
                    </contextData>
                    <linkType>m_i</linkType>
                    <timestamp>1432641880</timestamp> >
                    <reportSuiteID>lscsalexisandrdoidvideotest</reportSuiteID>
</request>
```

Documentation and code examples.

Please find below a list of additional documentation that you should read and code examples.

Code Examples

- Alexis Cazes :
  - https://github.com/alcazes/Adobe-Analytics-Video-tracking-s_code
  - https://github.com/alcazes/Adobe-Analytics-Milestone-VIdeo-AppMeasurement-JS
  - Brightcove flash implementation:
    - https://github.com/alcazes/Adobe-Analytics-Video-tracking-s_code/tree/master/Brightcove-flash
  - HTML5 JavaScript:
    - https://github.com/alcazes/Adobe-Analytics-Video-tracking-s_code/tree/master/HTML5
    - https://github.com/alcazes/Adobe-Analytics-Milestone-VIdeo-AppMeasurement-JS/tree/master/HTML5
  - Youtube iframe:
    - https://github.com/alcazes/Adobe-Analytics-Video-tracking-s_code/tree/master/Youtube-iframe
    - https://github.com/alcazes/Adobe-Analytics-Milestone-VIdeo-AppMeasurement-JS/tree/master/Youtube-iframe
  - ANDROID SDK 4.x:
    - https://github.com/alcazes/Android-Milestone-Video-tracking
- Mobile code examples
  - ANDROID SDK 4.x:
    - https://github.com/alcazes/mobile-services/blob/master/samples/Android/ADBMobileSamples/src/com/adobe/adbmobilesamples/MediaActivity.java
    - https://github.com/alcazes/mobile-services/blob/master/samples/Android/ADBMobileSamples/src/com/adobe/adbmobilesamples/VideoPlayerActivity.java
  - IOS SDK 4.x:
    - https://gist.github.com/hunterpeterson/9121178
    - https://github.com/alcazes/mobile-services/blob/master/samples/iOS/ADBMobileSamples/ADBMobileSamples/MediaViewController.m

Additional documentation

- Milestone video tracking:
  - https://marketing.adobe.com/resources/help/en_US/sc/appmeasurement/video/
- Marijka Engel Adobe Blogs:
  - http://blogs.adobe.com/digitalmarketing/author/marijka-engel/
- IOS SDK 4.x Video:
  - https://marketing.adobe.com/resources/help/en_US/mobile/ios/video_qs.html
- ANDROID SDK 4.x Video:
  - https://marketing.adobe.com/resources/help/en_US/mobile/android/video_qs.html