Adobe® Marketing Cloud

Video Analytics (Migrate to AEM)

Contents

Measuring Video in Adobe Analytics

Important: *The video documentation provided here is specific to clients utilizing version 1.5 or higher of Adobe's Video Analytics SDK for heartbeat measurement, and does not include instructions around the legacy milestone video implementation. We encourage all customers to move towards adopting the latest Video Analytics SDK to capitalize on the improvements and expanded measurement. You can view the benefits of transitioning to the SDK below. While Adobe will continue to support the Milestone method of tracking videos, there will not be any planned updates, fixes or feature improvements. Please reach out to your Adobe Account Manager if you have any further questions.*

Overview

Adobe Analytics for Video (Video Analytics) is an add-on to the base Analytics offering that provides clients with robust video measurement for content, audio and advertisements. Video Analytics provides many benefits to customers to allow for real-time monitoring, detailed analysis, actionable insights and monetization opportunities. Video tracking is enabled through SDK integrations with the most commonly used video players. The latest SDKs have the additional capability of capturing audio related metrics.

Adobe Analytics for Video enables clients to track the full customer journey across their site, which includes video consumption, and these measures are easily integrated into Analytics reporting and other Experience Cloud products. Video measurement allows you to slice and dice your data into multiple dimensions and segments, capturing all of the metadata you need to do a full detailed analysis, and to attribute success criteria to fully viewed videos, average time spent and completed ads.

The video solution not only measures vital video delivery metrics related to QoS, such as dropped frames, time spent buffering, and average bitrate, but it can also be combined with your website or app data to visualize the flow of the customer and their interests to better be able to make recommendations and personalize their experiences through the Adobe Experience Cloud.

Benefits

Some of the many benefits that Adobe's video solution provides include:

- **Timely analysis** - Make real-time, actionable decisions utilizing key video performance metrics (e.g., duration) across multiple channels. Video events are measured in **10-second** intervals to capture all activity as it occurs.
- **Drive engagement** - Fully engage users through fewer buffering events and by understanding where and when ads should display within video content to provide a smooth, less intrusive viewing experience that brings users back and delivers repeat visits.
- **Holistic picture** - Combine multiple data points across all of your content distributors to get a full view of all your video activity, and measure engagement and views across all possible channels through *Federated Analytics*.
- **Increased granularity** - Evaluate viewing behavior at the most granular level, including individual visitor time of day, concurrent viewers by minute, and average duration the content was viewed.
- **Precise measurement** - Measure across the multiple devices used for video consumption, including OTT, smartphone, tablet, desktop, and more, to monitor user engagement patterns and habits.
- **Segmentation** - Apply classifications to your players, devices, genres, chapters, and shows to see how each has an impact on your overall views and customer engagement with content, audio, ads, and combined.

Heartbeat versus Milestone Benefits

Adobe Analytics for Video is able to be measured through two means: the legacy Milestone method and the current Heartbeats method. The Heartbeats method is the preferred method of measurement and we encourage all clients to move to this version if they haven’t already, to take advantage of the benefits described below.

The legacy Milestone method is based on individual server calls to the Analytics server, for video starts, quartiles, duration, and completes. The Heartbeats method provides a more robust video tracking solution that measures video in 10 second intervals to provide enhanced, standardized video metrics. In addition, Adobe has derived learnings from our Milestone method to provide a smoother, streamlined implementation process through the Video Analytics SDK utilized by heartbeats.

Some of the many benefits of the Heartbeats method include:

- **Streamlined implementation process** - Map variables more easily through your player API and validate implementations through the Adobe Debug Tool to ensure all the necessary variables are tracked accurately.
- **Automatic Adobe Experience Cloud Integration** - Take advantage of the automatic integration with the Adobe Experience Cloud through the Marketing Cloud ID, segment your video audiences, target them, and make video recommendations based on user preferences.
- **Shared video data through Federated Analytics** - Capitalize on our industry-first video sharing capabilities, to evaluate data holistically across all of your video distribution partners—operators, programmers, and distributors.
- **Partnerships with Certified Ratings Partners** - Adobe partners with audience ratings partner Nielsen to provide neutral census third party measurement to allow for trusted, certified video ratings.
- **Standardized solution across all platforms** - Enable consistent, standardized variables across all of your videos and platforms to allow for a more efficient cross-campaign, device and vendor comparison.

Table 1: Comparison Chart

	Video Analytics- Milestone	**Video Analytics- Heartbeats**
Video Events	High-level Standard Events	Detailed and Custom Events every 10s
Metrics and Dimensions	Variances among Vendors, Non-Standardized Metrics and Dimensions	Clear, Standardized Metrics, Dimensions, and Benchmarks across Vendors
Integrations w/ Adobe Products	Individual Sessions w/ some Mappings and Integrations	Stitched Marketing Cloud ID linked to full Adobe Experience Cloud for easier cross-analysis
Pricing	Tracked and billed against each server call	Transparent tracking by video stream (single)
Implementation and Support	Longer integrations with limited support on legacy versions & no upgrades	Streamlined configuration with ongoing updates and improvements
Partner Sharing	N/A	Federated Analytics and Certified Metrics
Advanced Tracking	N/A	Error Recovery Tracking and Concurrent Viewers

Devices Supported

Adobe Analytics for Video has evolved with the industry to provide strong data collection tools to ensure each video stream is collected and reported across all meaningful devices. Our Video Analytics SDK is developed for all of the most utilized devices, including:

- iOS and Android smartphones and tablets

- OTT devices for ROKU, AppleTV, FireTV, and Android TV
- JavaScript Browser for Desktop and Laptop

The SDK's are routinely updated when new versions of devices are released, and you can use these SDKs to integrate with most of the largest video players today, including Brightcove and Ooyala.

The table below provides a list of the devices that are currently supported through our SDK implementation or Analytics API. To download the most recent version of the SDK, visit *Download the Video Heartbeat Library SDKs*. If there is a device that is not listed which you are seeking measurement against, please contact customer care or your solution consultant for the status of that device.

	Adobe Video SDK	**Video Analytics Collection API**
JavaScript Browser		
iOS Devices		
Android Devices		
Windows		
Blackberry		
Apple TV (new/legacy)		
ROKU		
OSX		
Fire TV		
Android TV		
Chromecast		
Xbox 1		
Sony PS3/PS4		

Getting Started

Before you begin your Adobe Analytics for Video implementation you have some early decisions to make, regarding which implementation scenario makes the most sense for your situation. The information in the following sections will help you determine how best to proceed:

- **Video Analytics** - Implement one of the Video Analytics implementation paths using the latest Video Analytics SDKs (recommended)
- **Milestone** - Implement with Milestone (the older Adobe tracking implementation)
- **Data Insertion APIs** - Implement without using Video Analytics SDKs (video tracking with Data Insertion APIs).

Video Analytics Implementation Paths

Video Analytics (using the Heartbeats model) is Adobe’s standardized video solution. It has replaced Adobe's older Milestone model.

For each of the paths described below, customers will need to contact their Sales Representative/Account Manager to sign a new Sales Order as Video Analytics has a unique SKU and changes from a pricing model based on server calls to a model based on video streams.

- **Video Analytics Path** - The Video Analytics path features the Video Analytics SDK, which can be used across any video player, including customer and/or OVP players such as Brightcove, Ooyala, thePlatform, and so on.

  Important: *To use Video Analytics, customers also need to use Adobe Analytics. If Video Analytics is your intended path, see Standard Implementation.*

- **Analytics and Ratings Path** - Adobe has partnered with Nielsen so that customers can opt-in to share data, collected by Adobe SDKs/tags, directly with Nielsen to provide the data that fuels their ratings solutions. For the Nielsen partnership, Adobe created a Combo SDK that comprises the Nielsen and Adobe Heartbeat SDKs. With one implementation, customers can capture data for analytics and ratings (Nielsen’s Digital Content Ratings). To leverage this partnership, a customer must complete the following steps:

  1. Sign an Adobe Sales Order for the Video Analytics SKU, based on streams.
  2. Sign an Adobe Addendum in which the customer gives permission to Adobe to share data directly with Nielsen.
  3. Contact your Nielsen Account team and complete paperwork from Nielsen to leverage Digital Content Ratings (DCR).
  4. After completing steps 1-3, contact your Adobe Account Manager to get the location where you can download the Combo SDK.

     Note: *The Combo SDK is not available on GitHub.*

  For more information about the partnership and how to implement this path, see *Digital Content Ratings, powered by Adobe*.

  With this partnership, to ensure proper implementation and to guarantee quality data based on agreed upon standardized metrics, Adobe will certify your builds before you release them to production.

  Note: *To send data to our ratings partners, customers need to contact their Sales Representative/Account Manager to purchase Adobe Certifications.*

- **Adobe Primetime Path -** Adobe Primetime is an Adobe Marketing Cloud solution that helps content programmers and distributors monetize video on every connected screen. Primetime eliminates the complexity of reaching, monetizing, and activating global audiences across devices by providing a modular platform for video publishing, advertising, personalization, and analytics. Additionally, Primetime offers solutions and value around the following:

  - Support for accurately measuring Linear and VOD content types.
  - Support for measuring ad breaks with (or without) dynamic ad insertion.
  - TVSDK's seamless ad insertion model allows for analytics that directly measures the ad playback, which increases accuracy.
  - Robust set of events and metadata to ensure accuracy across QoS buffering or mobile connectivity interruptions issues and end-user interactions such as seeking, pausing, and backgrounding on mobile.
  - Integrated support for Nielsen DTVR (linear) with ID3 metadata and DCR with CMS metadata.

  TVSDK is already integrated with the Video Analtyics (Heartbeats) SDK, which makes implementation much easier and faster across every supported platform. Primetime also supports the partnership with Nielsen. To leverage Primetime, follow the same guidelines and prerequisites found in *Video Analytics* along with the following docs for your platform(s):

  - *Video Analytics in TVSDK 1.4 for Android*
  - *Video Analytics in TVSDK 2.4 for Browser TVSDK*
  - *Video Analytics in TVDSK 1.4 for Desktop HLS*
  - *Video Analytics in TVDSK 2.3 for Desktop HLS*
  - *Video Analytics in TVSDK 1.4 for iOS*

  You should also contact your Sales Representative/Account Manger to discuss what you need to do to purchase TVSDK.

Video Milestone Implementation (Old)

The Video Milestone measurement model is the previous method Adobe supported to track video. This has since been replaced by the Video Analytics (Heartbeats) model.

The Milestone model allows a completely custom implementation where customers can decide when and how frequently to send in server calls during playback. It was called "Milestone" because the most common implementations sent in server calls at the start, 25%, 50%, 75%, and at the completion of the video. The solution uses your own custom eVars and events to track video by using Adobe's Media Module. The Heartbeats solution was designed to build upon some of the limitations of Milestones, specifically around providing more granularity and accuracy around engagement and to provide a more standardized measurement solution across our entire customer base.

Table 2: Feature Differences Between Milestone and Heartbeat Video Measurement

Feature	Milestone	Heartbeat
Quartile tracking		
Real-Time Reporting		
10 second time spent granularity		
Standardized data and benchmarking		

Feature	Milestone	Heartbeat
Chapter tracking		
Pricing based on streams		
Player Mapping implementation model		
Required variables ONLY in base objects		
Streamlined configuration		
Error state recovery		
Video ad tracking included in solution		

The Milestone model can be used at the same time as the Heartbeats model in a specific report suite, but both solutions should not be used at the same time in the same player. Data captured from both models can live and be reported on in the same report suite. This will allow you to gradually migrate to the Heartbeats model without having to update everything at the same time.

For more information, see the *Video Milestone documentation*.

Video Data Insertion APIs

The Data Insertion API is a way to track video playback on devices and platforms where Adobe does not currently have a Video Heartbeats SDK. HTTP POSTs and HTTP GETs enable a server-side method to collect data.

When using the API, during playback you will determine which events and metadata that you want to send to Adobe with server calls. However, these API calls are completely independent of Adobe's Video Analytics solution (Heartbeats). This means that you cannot leverage Adobe's solution/reserved variables to standardize your video implementation, nor will Adobe send any of this data to our ratings partners.

For more information about the Data Insertion API, see *Data Insertion: Overview*.

Prerequisites for Standard Implementation

Audience Manager Enablement

Adobe Audience Manager (AAM), a Data Management Platform (DMP), helps you bring your audience data assets together, making it easy to collect commercially relevant information about site visitors, create marketable segments, and serve targeted advertising and content to the right audience.

With AAM, you are not tied to a data seller, exchange, or demand-side platform. Additionally, AAM is completely agnostic when it comes to your partners' data assets. With access to multiple data sources, AAM offers digital publishers the ability to use a wide variety of third-party data and our private data co-op.

To learn more about AAM, see *Audience Manager Overview*.

Video Heartbeat Data in Audience Manager

For both video content and video ads, the metrics and metadata that are collected by using solution (reserved) variables can be automatically sent to AAM. The data transfer is available across all platforms including desktop, mobile, and OTT. To enable this server-side data transfer, you need to reach out to Adobe Client Care and ask for this feed to be enabled.

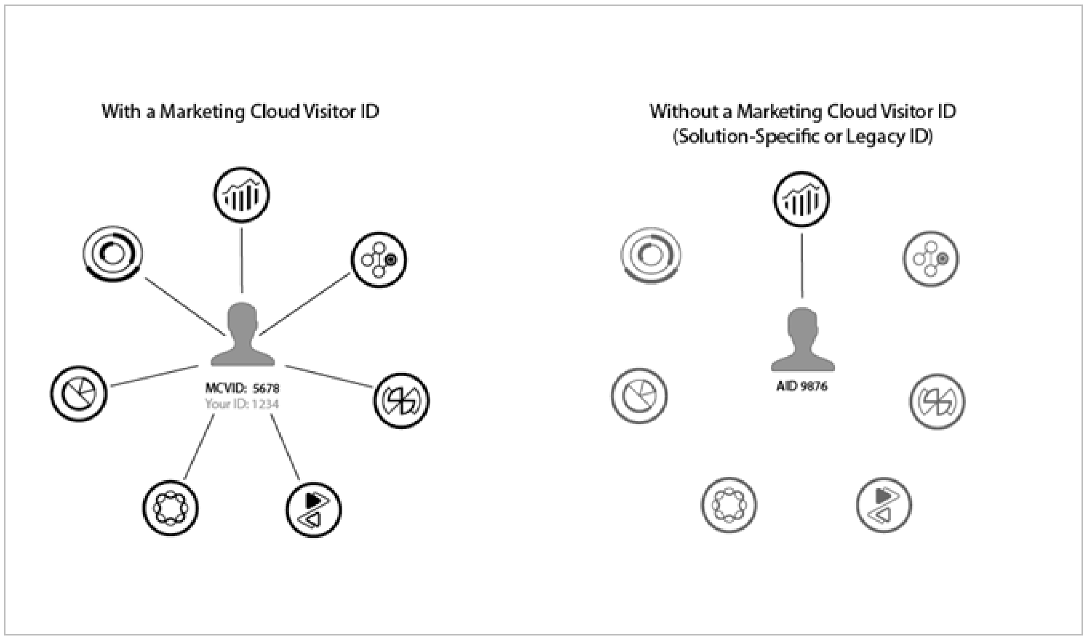
Federated Data fully supports data sharing to AAM. Please work with your Adobe team for confirmation of Federated Data settings.

Important: *To ensure the smooth transfer of data to AAM, you should be on the latest versions of the heartbeat libraries.*

Marketing Cloud Enablement

As part of the requirements to implement the Video Heartbeat SDK, you need to implement the Marketing Cloud ID Service (formerly known as Visitor ID Service).

The Marketing Cloud ID service enables the common identification framework for the Marketing Cloud Core Services, solutions, and customer attributes and audiences in the People core service. It works by assigning a unique, persistent ID to a site visitor. When your organization implements the ID service, this ID lets you identify the same site visitor and their data in different Marketing Cloud solutions.


With a Marketing Cloud Visitor ID
MCVID: 5678
Your ID: 1234
Without a Marketing Cloud Visitor ID (Solution-Specific or Legacy ID)
AID 9876

The ID service can also replace the different solution-specific IDs (for example, Analytics AID). Through the *Customer IDs and Authentication States* functionality, the ID service lets you pass in your own customer IDs to the Marketing Cloud. Keep in mind, however, that the ID service only works with the solutions to which you have already subscribed. If you are not signed up for access to other products, the ID service does not provide the access.

Going forward, the ID service is an integral component of many current and future Marketing Cloud features, enhancements, and services. Currently, the ID service supports *Analytics*, *Audience Manager*, and *Target*.

Important: *To participate in the Adobe Marketing Cloud Device Co-op, the Marketing Cloud ID service required.*

If you have not implemented the ID service, now is the time to start considering a migration strategy. For more information about the importance and role of the ID service, see *Why the Marketing Cloud ID Service Should be on Your Radar*.

Important: *In the absence of any user ID information present on the video specific calls the default analytics Fallback ID Methods will apply.*

For more information about the Marketing Cloud ID, see *Overview*.

For more information about implementing the Marketing Cloud ID Service, see *Marketing Cloud ID Service*.

Analytics Enablement

To enable video reports in Analytics and see the video and video ad data you are collecting, see *Video Reports Enablement*.

Download the Video Heartbeat Library SDKs

You need to download the VHL SDKs before you can start the implementation.

Download the 2.x SDKs

Video Analytics 2.x SDKs	Latest VHL SDK Downloads	VHL Getting Started Guides
Android/FireTV	*VHL SDK for Android v2.0.1*	*Get Started - VHL 2.x for Android*
iOS/AppleTV	*VHL SDK for for iOS v2.0.1*	*Get Started - VHL 2.x for iOS*
JavaScript	*VHL SDK for JS v2.0.2*	*Get Started - VHL 2.x for JS*
Roku	*VHL SDK for Roku v2.0.1*	*Get Started - VHL 2.x for Roku*
Chromecast	*VHL SDK for Chromecast v2.0.0*	*Get Started - VHL 2.x for Chromecast*

Download the 1.x SDKs

Video Analytics 1.x SDKs	Latest VHL SDK Downloads	VHL Getting Started Guides
Android	*VHL SDK for Android v1.5.8*	*Get Started - VHL 1.x for Android*
iOS	*VHL SDK for iOS v1.5.9*	*Get Started - VHL 1.x for iOS*
JavaScript	*VHL SDK for JavaScript v1.5.7*	*Get Started - VHL 1.x for JS*
TVML	*VHL SDK for TVML v1.0.0*	*Get Started - VHL 1.x for TVML*
Apple TV	*VHL SDK for Apple TV v1.0.0*	*Get Started - VHL 1.x for Apple TV*
Chromecast	*VHL SDK for Chromecast v1.0.0*	*Get Started - VHL 1.x for Chromecast*

Download the Adobe Nielsen 2.x SDKs

Adobe Nielsen 2.x SDKs	Latest VHL SDK Downloads	VHL Nielsen Implementation Guides
Android	*VHL for Android v.2.0.1N*	*Implement VHL for Android 2.x*
iOS	*VHL for iOS v.2.0.1N*	*Implement VHL for iOS 2.x*
JavaScript	*VHL for JavaScript v.2.0.1N*	*Implement VHL for JS 2.x*

Set up and Configure

The following instructions provide guidance for implementation across all 2.x SDKs.

Important: *Important: The following instructions provide guidance for implementation across all 2.x SDKs. If you are implementing a 1.x version of the SDK, you can download the Developers Guide further down the page.*

This topic contains the following information:

Implement

1. Import the Heartbeat libraries or for JavaScript, create local references to the classes. There are three classes/libraries to reference.

   For links to platforms and additional details, see the **Code** section below.

   - Media Heartbeat Config: The config contains the basic settings for reporting.
   - Media Heartbeat Delegate: The delegate controls playback time and the QoS object.
   - Media Heartbeat: The primary library containing members and methods.

2. Create a `MediaHeartbeatConfig` Instance.

   Set the config values on your MediaHeartbeatConfig instance for accurate tracking.

Variable Name	**Description**	**Required**	**Default Value**
`trackingServer`	Define the server for tracking media heartbeats. This is different from your analytics tracking server.	Yes	Empty String
`channel`	Channel name property	Yes	Empty String
`ovp`	Name of the online video platform through which content gets distributed.	Yes	unknown
`appVersion`	Version of the video player app/SDK.	Yes	unknown
`playerName`	Name of the video player in use, i.e., "AVPlayer", "HTML5 Player", "My Custom VideoPlayer".	Yes	Empty String
`ssl`	Property that indicates whether the heartbeat calls should be made over HTTPS.	Yes	false
`debugLogging`	Gets the preference for debug log output.	Yes	false

3. Implement the `MediaHeartbeatDelegate`.
   Here is the `MediaHeartbeatDelegate` reference:

Method name	Description	Required
`getQoSObject()`	Returns the MediaObject instance that contains the current QoS information. This method will be called multiple times during a playback session. Player implementation must always return the most recently available QoS data.	Yes
`getCurrentPlaybackTime()`	Returns the current position of the playhead. For VOD tracking, the value is specified in seconds from the beginning of the media item. For LINEAR/LIVE tracking, the value is specified in seconds from the beginning of the program.	Yes

Tip: *The QoS Object is not required. If the QoS data is available for your player, then the following variables are required to fully track Quality of Service.*

Here is the `MediaObject` (QoS Object) reference:

Variable name	Description	Required
`bitrate`	The bitrate of media in bits per second.	Yes
`startupTime`	The start up time of media in seconds.	Yes
`fps`	The frames displayed per second.	Yes
`droppedFrames`	The number of dropped frames so far.	Yes

4. Create the `MediaHeartbeat` instance.

   Use the `MediaHertbeatConfig` and `MediaHertbeatDelegate` to create the `MediaHeartbeat` instance.

   Important: *Make sure that your `MediaHeartbeat` instance is accessible and does not get deallocated until the end of the video session. This instance will be used for all the following video tracking events.*

   Tip: *`MediaHeartbeat` requires an instance of `AppMeasurement` to send calls to Adobe Analytics.*

5. Combine all of the pieces.

   The following sample code utilizes our JavaScript 2.x SDK for an HTML5 video player:

```
// Create local references to the heartbeat classes
var MediaHeartbeat = ADB.va.MediaHeartbeat;
var MediaHeartbeatConfig = ADB.va.MediaHeartbeatConfig;
var MediaHeartbeatDelegate = ADB.va.MediaHeartbeatDelegate;

//Media Heartbeat Config
var mediaConfig = new MediaHeartbeatConfig();
mediaConfig.trackingServer = "namespace.hb.omtrdc.net";
mediaConfig.playerName = "HTML5 Basic";
mediaConfig.channel = "Video Channel";
mediaConfig.debugLogging = true;
mediaConfig.appVersion = "2.0";
mediaConfig.ssl = false;
mediaConfig.ovp = "HTML5";

// Media Heartbeat Delegate
var mediaDelegate = new MediaHeartbeatDelegate();

// Set mediaDelegate CurrentPlaybackTime
mediaDelegate.getCurrentPlaybackTime = function() {
    return video.currentTime;
};

// Set mediaDelegate QoSObject - OPTIONAL
mediaDelegate.getQoSObject = function() {
    return MediaHeartbeat.createQoSObject(video.bitrate,
                                          video.startuptime,
                                          video.fps,
                                          video.droppedframes);
}
// Create mediaHeartbeat instance
this.mediaHeartbeat =
  new MediaHeartbeat(mediaDelegate, mediaConfig, appMeasurementInstance);
```

Code

Video Analytics 2.x SDKs	Developer Guides
Android/FireTV	*Configure for Android*
iOS/AppleTV	*Configure for iOS*
JavaScript	*Configure for JavaScript*
Roku	*Configure for Roku*

Video Analytics 1.x SDKs*	Developer Guides
Android	*Configure for Android*
AppleTV	*Configure for AppleTV*
Chromecast	*Configure for Chromecast*
iOS	*Configure for iOS*
JavaScript	*Configure for JavaScript*
Primetime	• **Android**: *Configure Video Analytics* • **DHLS**: *Configure Video Analytics* • **iOS**: *Configure Video Analytics*
TVML	*Configure for TVML*

***** For all 1.x SDKs, the links are for the full PDF download of the documentation.

Validate

Video implementations are composed of two types of tracking calls:

• Video and Ad Start calls are sent directly to the AppMeasurement server.

- Heartbeat calls are sent on start and every ten seconds to the Heartbeat tracking server.

Video tracking will behave the same across all platforms, desktop and mobile.

Across all tracking calls there are a few key universal variables to be validated:

- **AppMeasurement (Analytics)**

  For more information about tracking server options, see *Correctly populate the trackingServer and trackingServerSecure variable*.

  Important: *An RDC tracking server or CNAME resolving to an RDC server is required for Marketing Cloud Visitor ID service.*

  The analytics tracking server should end in `.sc.omtrdc.net` or be a CNAME.

- **Video Heartbeat**

  Always has the format `[namespace].hb.omtrdc.net`, where `[namespace]` is defined by your login company and is provided by Adobe.

Track Core Video Playback

The following instructions provide guidance for implementation across all 2.x SDKs.

Important: *If you are implementing a 1.x version of the SDK, you can download the 1.x Developers Guide further down the page.*

This topic contains the following information:

- *Overview*
- *Implement*
- *Code*
- *Validate*

Overview

Core video playback includes tracking video starts, video completes, pausing, and scrubbing. Utilize the video player API to identify key player events and to populate the required and optional video variables. The following are the key elements of tracking video playback; details for each item are below:

On video load:

- Create the media object
- Populate the metadata
- Call `trackSessionStart(mediaObject, contextData)`

On video start:

- Call `trackPlay()`

On video complete:

- Call `trackComplete()`
- Call `trackSessionEnd()`

On video pause:

- Call `trackPause()`
- Call `trackPlay()` when the video resumes

On video scrub:

- Call `trackEvent(MediaHeartbeat.Event.SeekStart)`
- Call `trackEvent(MediaHeartbeat.Event.SeekComplete)`

On video buffer:

- Call `trackEvent(MediaHeartbeat.Event.BufferStart);`
- Call `trackEvent(MediaHeartbeat.Event.BufferComplete);`

Tip: *The playhead position is set as part of the set-up and configuration code. For more information about* `getCurrentPlayheadTime()`*, see Set up and Configure.*

Implement

To implement core video playback:

1. Identify when the user triggers the intention of playback (user clicks play and/or autoplay is on) and create a `MediaObject` instance using the video information for video name, video ID, video length, and stream type.

   Here is the `MediaObject` reference:

Variable Name	Description	Required
`name`	Video name	Yes
`mediaid`	Video unique identifier	Yes
`length`	Video length	Yes
`streamType`	Stream type (see `constants MediaHeartbeat.StreamType.VOD`)	Yes

   Here is the `MediaHeartbeat.StreamType.VOD` constants reference:

Constant Name	Description
`VOD`	Stream type for Video on Demand.
`LIVE`	Stream type for LIVE content.
`LINEAR`	Stream type for LINEAR content.

   The general format for the `MediaObject` is `MediaHeartbeat.createMediaObject(<VIDEO_NAME>, <VIDEO_ID>, <VIDEO_LENGTH>, <STREAM_TYPE>.VOD);`

```
var mediaObject =
  MediaHeartbeat.createMediaObject("Name", "ID", VIDEO_LENGTH, MediaHeartbeat.StreamType.VOD);
```

2. Attach all custom video metadata and standard video metadata to the video tracking session through context data variables.

   For custom metadata, create a variable object for the custom variables and populate with the data for this video. For example:

```
/* Set custom context data */
var customVideoMetadata = {
    isUserLoggedIn: "false",
    tvStation: "Sample TV station",
    programmer: "Sample programmer"
};
```

   For standard metadata, instantiate the `standardVideoMetdata` object and populate the desired variables. For a complete list of standard metadata variables, see *Metrics and Metadata*. For example:

```
var standardVideoMetadata = {};
standardVideoMetadata[MediaHeartbeat.VideoMetadataKeys.EPISODE] = "Sample Episode";
standardVideoMetadata[MediaHeartbeat.VideoMetadataKeys.SHOW] = "Sample Show";
mediaObject.setValue(MediaHeartbeat.MediaObjectKey.StandardVideoMetadata,
standardVideoMetadata);
```

   Tip: *Attaching `standardVideoMetadata` to `mediaObject` is optional.*

3. To begin tracking a video session, call `trackSessionStart` in the `MediaHeartbeat` instance.

   Tip: *The second value is the custom video metadata object name that you created in step 2.*

   ```
   mediaHeartbeat.trackSessionStart(mediaObject, customVideoMetadata);
   ```

   Important: *`trackSessionStart()` tracks the user intention of playback, not the beginning of the playback. This API is used to load the video data/metadata and to estimate the time-to-start QoS metric (the time duration between `trackSessionStart()` and `trackPlay()`).*

   Tip: *If you are not using custom video metadata, send a null value for the data argument in `trackSessionStart()`. For example:*

   `mediaHeartbeat.trackSessionStart(mediaObject, null)`

4. Identify the event from the video player for the beginning of the video playback, where the first frame of the video is rendered on the screen, and call `trackPlay()`:

   ```
   mediaHeartbeat.trackPlay();
   ```

5. Identify the event from the video player for the completion of the video playback, where the user has watched the content until the end, and call `trackComplete()`:

   ```
   mediaHeartbeat.trackComplete();
   ```

6. Identify the event from the video player for the unloading/closing of the video playback, where the user closes the video and/or the video is completed and has been unloaded, and call `trackSessionEnd()`:

   ```
   mediaHeartbeat.trackSessionEnd();
   ```

   Important: *`trackSessionEnd()` marks the end of a video tracking session. If the session was successfully watched to completion, where the user watched the content until the end, ensure that `trackComplete()` is called before `trackSessionEnd()`. Any other `track*()` API call is ignored after `trackSessionEnd()`, except for `trackSessionStart()` for a new video tracking session.*

7. Identify the event from the video player for video pause and call `trackPause()`.

   ```
   mediaHeartbeat.trackPause();
   ```

   Tip: *Identify any scenario in which the Video Player will pause and make sure that `trackPause()` is properly called. Sample scenarios include when an application goes to the background or the player automatically pauses because of a mobile interrupt.*

8. Identify the event from the video player for video play and/or video resume from pause and call `trackPlay()`.

   ```
   mediaHeartbeat.trackPlay();
   ```

9. Identify the event from the video player for scrubbing/seeking and utilize a custom `MediaHeartbeat` event to capture the action.

   ```
   mediaHeartbeat.trackEvent(MediaHeartbeat.Event.SeekStart);
   ```

10. Identify the event from the video player for video play and/or video resume from scrubbing/seeking and use a second custom `MediaHeartbeat` event.

    ```
    mediaHeartbeat.trackEvent(MediaHeartbeat.Event.SeekComplete)
    ```

11. Listen for playback buffering to start and use a custom `MediaHeartbeat` event to capture the change.

    ```
    mediaHeartbeat.trackEvent (MediaHeartbeat.Event.BufferStart);
    ```

12. Identify when buffering ends and call the `MediaHeartbeat` event to capture the change.

    ```
    mediaHeartbeat.trackEvent (MediaHeartbeat.Event.BufferComplete);
    ```

The following sample code uses the JavaScript 2.x SDK for an HTML5 video player:

```
/* Call on video start */
if (e.type == "play") {

    // Check for start of video
    if (mediaOffset == 0) {
        /* Set media info */
        /* MediaHeartbeat.createMediaObject(<VIDEO_NAME>,
                                            <VIDEO_ID>,
                                            <VIDEO_LENGTH>,
                                            MediaHeartbeat.StreamType.VOD);*/
        var mediaInfo = MediaHeartbeat.createMediaObject(
          document.getElementsByTagName('video')[0].getAttribute("name"),
          document.getElementsByTagName('video')[0].getAttribute("id"),
          video.duration,
          MediaHeartbeat.StreamType.VOD);

        /* Set custom context data */
        var customVideoMetadata = {
            isUserLoggedIn: "false",
            tvStation: "Sample TV station",
            programmer: "Sample programmer"
        };

        /* Set standard video metadata */
        var standardVideoMetadata = {};
        standardVideoMetadata[MediaHeartbeat.VideoMetadataKeys.EPISODE] = "Sample Episode";
        standardVideoMetadata[MediaHeartbeat.VideoMetadataKeys.SHOW] = "Sample Show";
        mediaInfo.setValue(MediaHeartbeat.MediaObjectKey.StandardVideoMetadata,
                           standardVideoMetadata);

        // Start Session
        this.mediaHeartbeat.trackSessionStart(mediaInfo, customVideoMetadata);

        // Track play
        this.mediaHeartbeat.trackPlay();

    } else {
        // Track play for resuming playack
        this.mediaHeartbeat.trackPlay();
    }
};

/* Call on video complete */
if (e.type == "ended") {
    console.log("video ended");
    this.mediaHeartbeat.trackComplete();
    this.mediaHeartbeat.trackSessionEnd();
    mediaOffset = 0;
};

/* Call on pause */
if (e.type == "pause") {
    this.mediaHeartbeat.trackPause();
```

```
};

/* Call on scrub start */
if (e.type == "seeking") {
    this.mediaHeartbeat.trackEvent(MediaHeartbeat.Event.SeekStart);
};

/* Call on scrub stop */
if (e.type == "seeked") {
    this.mediaHeartbeat.trackEvent(MediaHeartbeat.Event.SeekComplete);
};

/* Call on buffer start */
if (e.type == “buffering”) {
    this.mediaHeartbeat.trackEvent(MediaHeartbeat.Event.BufferStart);
};

/* Call on buffer complete */
if (e.type == “buffering end”) {
    this.mediaHeartbeat.trackEvent(MediaHeartbeat.Event.BufferComplete);
};
```

Code

Video Analytics 2.x SDKs	Developer Guides
Android/FireTV	*Track Core for Android*
iOS/AppleTV	*Track Core for iOS*
JavaScript	*Track Core for JavaScript*
Roku	*Track Core for Roku*
Chromecast	*Track Core for Chromecast*

Video Analytics 1.x SDKs*	Developer Guides
Android	*Track Core for Android*
AppleTV	*Track Core for AppleTV*
Chromecast	*Track Core for Chromecast*
iOS	*Track Core for iOS*
JavaScript	*Track Core for JavaScript*
Primetime	• **Android**: *Configure Video Analytics* • **DHLS**: *Configure Video Analytics* • **iOS**: *Configure Video Analytics*
TVML	*Track Core for TVML*

***** For all 1.x SDKs, the links are for the full PDF download of the documentation.

Validate

Video Start

On start of a video player, these key calls are sent in the following order:

1. Video analytics start*****

2. Heartbeat start*****
3. Heartbeat analytics start

*****These calls contain additional metadata variables for both custom and standard.

Content Play

During regular main content playback, Heartbeat calls are sent to the Heartbeat server every ten seconds.

Video Complete

At the 100% point, on a video or at a show boundary on a linear stream, a Heartbeat complete call will be sent.

Content Pause

When the video player pauses, video player pause event calls will be sent every 10 seconds. After pause ends, the play events should resume.

Content Scrub/Seek

On scrubbing of the video playhead, no special tracking calls are sent. However, when video playback resumes after scrubbing, the playhead value should reflect the new position in the main content.

Content Buffer

When the video player buffers, video player buffer event calls will be sent every 10 seconds. After buffer ends, the play events should resume.

Track Ads

The following instructions provide guidance for implementation across all 2.x SDKs.

Important: *The following instructions provide guidance for implementation across all 2.x SDKs. If you are implementing a 1.x version of the SDK, you can download the Developers Guide further down the page.*

This topic contains the following information:

- *Overview*
- *Implement*
- *Code*
- *Validate*

Overview

Ad playback includes tracking ad breaks, ad starts, ad completes, and ad skips. You can use the video player's API to identify key player events and to populate the required and optional ad variables.

Here are the key elements to track ad playback:

On ad break start, including pre-roll

- Create the `adBreak` object instance for the ad break, for example, `adBreakObject`.
- Call `trackEvent(MediaHeartbeat.Event.AdBreakStart, adBreakObject);`.

On every ad asset start

- Create the ad object instance for the ad asset, for example, `adObject`.
- Populate the ad metadata, `adCustomMetadata`.
- Call `trackEvent(MediaHeartbeat.Event.AdStart, adObject, adCustomMetadata);`.

On every ad asset complete

- Call `trackEvent(MediaHeartbeat.Event.AdComplete);`.

On ad skip

- Call `trackEvent(MediaHeartbeat.Event.AdSkip);`.

On ad break complete

- Call `trackEvent(MediaHeartbeat.Event.AdBreakComplete);`.

Implement

To implement ad playback:

1. Identify when the ad break boundary begins, including pre-roll, and create an `AdBreakObject` by using the ad break information.

   Here is the Ad break object reference:

Variable Name	Description	Required
`name`	Ad break name such as pre-roll, mid-roll, and post-roll.	Yes
`position`	The number position of the ad break starting with 1.	Yes
`startTime`	Playhead value at the start of the ad break.	Yes

The general format for the ad break object is:

```
var adBreakObject = MediaHeartbeat.createAdBreakObject(<ADBREAK_NAME>, <POSITION>,
<START_TIME>);
```

2. Call `trackEvent()` with `AdBreakStart` in the `MediaHeartbeat` instance to begin tracking the ad break.

```
mediaHeartbeat.trackEvent(MediaHeartbeat.Event.AdBreakStart, adBreakObject);
```

3. Identify when the ad asset starts and create an `AdObject` instance using the ad information.

   Here is the `AdObject` reference:

Variable Name	Description	Required
`name`	Friendly name of the ad asset.	Yes
`adId`	Unique identified for the ad asset.	Yes
`position`	The number position of the asset in the ad break, starting with 1.	Yes
`length`	Ad asset length	Yes

The general format for the ad object is:

```
var adObject = MediaHeartbeat.createAdObject(<AD_NAME>, <AD_ID>, <POSITION>, <LENGTH>);
```

4. Attach all the custom ad metadata to the video tracking session through context data variables.

   For custom metadata, create a variable object for the custom data variables and populate with the data for the current ad asset. For example:

```
/* Set custom context data */
var adCustomMetadata = {
    affiliate: "Sample affiliate",
    campaign: "Sample ad campaign",
    creative: "Sample creative"
};
```

5. Call `trackEvent()` with the `AdStart` event in the `MediaHeartbeat` instance to begin tracking the ad playback.

   Be sure to include a reference to your custom metadata variable as the third parameter in the event call:

```
mediaHeartbeat.trackEvent(MediaHeartbeat.Event.AdStart, adObject, adCustomMetadata);
```

6. When the ad asset playback reaches the end of the ad, call `trackEvent()` with the `AdComplete` event.

   ```
   mediaHeartbeat.trackEvent(MediaHeartbeat.Event.AdComplete);
   ```

7. If ad playback did not complete because the user chose to skip the ad, track the `AdSkip` event.

   ```
   mediaHeartbeat.trackEvent(MediaHeartbeat.Event.AdSkip);
   ```

8. When the ad break is complete, use the `AdBreakComplete` event to track.

   ```
   mediaHeartbeat.trackEvent(MediaHeartbeat.Event.AdBreakComplete);
   ```

The following sample code utilizes our JavaScript 2.x SDK for an HTML5 video player.

```
/* Call on ad break start */

if (e.type == "ad break start") {
    var adBreakObject = MediaHeartbeat.createAdBreakObject("mid-roll", 2, 500);
    this.mediaHeartbeat.trackEvent(MediaHeartbeat.Event.AdBreakStart, adBreakObject);
};

/* Call on ad start */
if (e.type == "ad start") {
    var adObject = MediaHeartbeat.createAdObject("PepsiOne", "123456ab", 1, 30);
    /* Set custom context data */
    var adCustomMetadata = {
        affiliate:"Sample affiliate",
        campaign:"Sample ad campaign",
        creative:"Sample creative"
    }
    this.mediaHeartbeat.trackEvent(MediaHeartbeat.Event.AdStart, adObject, adCustomMetadata);
};

/* Call on ad complete */
if (e.type == "ad complete") {
    this.mediaHeartbeat.trackEvent(MediaHeartbeat.Event.AdComplete);
};

/* Call on ad skip */
if (e.type == "ad skip") {
    this.mediaHeartbeat.trackEvent(MediaHeartbeat.Event.AdSkip);
};

/* Call on ad break complete */
if (e.type == "ad break complete") {
    this.mediaHeartbeat.trackEvent(MediaHeartbeat.Event.AdBreakComplete);
};
```

Code

Video Analytics 2.x SDKs	Developer Guides
Android/FireTV	*Track Ads for Android*
iOS/AppleTV	*Track Ads for iOS*
JavaScript	*Track Ads for JavaScript*
Roku	*Track Ads for Roku*
Chromecast	*Track Ads for Chromecast*

Video Analytics 1.x SDKs*	Developer Guides
Android	*Track Ads for Android*

Video Analytics 1.x SDKs*	Developer Guides
AppleTV	*Track Ads for AppleTV*
Chromecast	*Track Ads for Chromecast*
iOS	*Track Ads for iOS*
JavaScript	*Track Ads for JavaScript*
Primetime	• **Android**: *Configure Video Analytics* • **DHLS**: *Configure Video Analytics* • **iOS**: *Configure Video Analytics*
TVML	*Track Ads for TVML*

***** For all 1.x SDKs, the links are for the full PDF download of the documentation.

Validate

Ad Start

On start of an individual ad playback, three key calls are sent in the following order:

1. Video ad analytics start*****
2. Heartbeat ad start*****
3. Heartbeat analytics start

*****These calls contain additional metadata variables for both custom and standard.

Ad Play

During ad playback, Heartbeat ad play calls are sent to the Heartbeat server every ten seconds.

Ad Complete

At the 100% point on an ad, a Heartbeat ad complete call will be sent.

Ad Skip

When an ad is skipped, no events are sent, so the tracking calls will not include the ad information.

Tip: *No unique calls are sent on ad break start and ad break complete.*

Track Chapters and Segments

The following instructions provide guidance for implementation across all 2.x SDKs.

Important: *The following instructions provide guidance for implementation across all 2.x SDKs. If you are implementing a 1.x version of the SDK, you can download the Developers Guide further down the page.*

This topic contains the following information:

- *Overview*
- *Implement*
- *Code*
- *Validate*

Overview

Chapter and segment tracking is available for custom-defined video chapters or segments. Some common uses for chapter tracking are to define custom segments based on video content, such as baseball innings, or to define content segments between ad breaks. Chapter tracking is **not** required for core video heartbeat implementations.

Chapter tracking includes chapter starts, chapter completes, and chapter skips. You can use the video player API with customized segmentation logic to identify chapter events and to populate the required and optional chapter variables. Here are the key elements of tracking chapter playback:

Tip: *Additional details for each section is available in the Implement section.*

On chapter start:

- Create the chapter object instance for the chapter, `chapterObject`
- Populate the chapter metadata, `chapterCustomMetadata`
- Call `trackEvent(MediaHeartbeat.Event.ChapterStart, chapterObject, chapterCustomMetadata);`

On chapter complete:

- Call `trackEvent(MediaHeartbeat.Event.ChapterComplete);`

On chapter skip:

- Call `trackEvent(MediaHeartbeat.Event.ChapterSkip);`

Implement

To implement custom video chapters:

1. Identify when the chapter start event occurs and create the `ChapterObject` instance by using the chapter information.

   Here is the `ChapterObject` chapter tracking reference:

Variable Name	Description	Required
name	Chapter name	Yes
position	Chapter position	Yes

Variable Name	Description	Required
length	Chapter length	Yes
startTime	Chapter start time	Yes

Tip: *These variables are only required if you are planning to track chapters.*

The general format of the chapter object is:

```
var chapterObject =
  MediaHeartbeat.createChapterObject(<CHAPTER_NAME>, <POSITION>, <LENGTH>, <START_TIME>);
```

2. If you include custom metadata for the chapter, create the context data variables for the metadata.

```
/* Set custom context data */
var chapterCustomMetadata = {
    segmentType: "Sample segment type",
    segmentName: "Sample segment name",
    segmentInfo: "Sample segment info"
};
```

3. To begin tracking the chapter playback, call the `ChapterStart` event in the `MediaHeartbeat` instance.

```
mediaHeartbeat.trackEvent(MediaHeartbeat.Event.ChapterStart,
                                      chapterObject,
                                      chapterCustomMetadata);
```

4. When playback reaches the chapter end boundary, as defined by your custom code, call the `ChapterComplete` event in the MediaHeartbeat instance.

```
mediaHeartbeat.trackEvent(MediaHeartbeat.Event.ChapterComplete);
```

5. If chapter playback did not complete because the user chose to skip the chapter (for example, if the user seeks out of the chapter boundary), call the `ChapterSkip` event in the MediaHeartbeat instance.

```
mediaHeartbeat.trackEvent(MediaHeartbeat.Event.ChapterSkip);
```

The following sample code uses the JavaScript 2.x SDK for an HTML5 video player. You should use this code with the core video playback code.

```
/* Call on chapter start */
if (e.type == "chapter start") {
    var chapterObject = MediaHeartbeat.createChapterObject("Inning 5",5,500,2500);
    /* Set custom context data*/
    var chapterCustomMetadata = {
        segmentType:"Baseball Innings",
        segmentName:"Inning 5",
        segmentInfo:"Game Six"
    }
    this.mediaHeartbeat.trackEvent(MediaHeartbeat.Event.ChapterStart,
                                   chapterObject,
                                   chapterCustomMetadata);
};

/* Call on chapter complete */
if (e.type == "chapter complete") {
    this.mediaHeartbeat.trackEvent(MediaHeartbeat.Event.ChapterComplete);
};

/* Call on chapter skip */
if (e.type == "chapter skip") {
    this.mediaHeartbeat.trackEvent(MediaHeartbeat.Event.ChapterSkip);
};
```

Code

Video Analytics 2.x SDKs	**Developer Guides**
Android/FireTV	*Track Chapters for Android*
iOS/AppleTV	*Track Chapers for iOS*
JavaScript	*Track Chapters for JavaScript*
Roku	*Track Chapters for Roku*
Chromecast	*Track Chapters for Chromecast*

Video Analytics 1.x SDKs*	**Developer Guides**
Android	*Track Chapters for Android*
AppleTV	*Track Chapters for AppleTV*
Chromecast	*Track Chapters for Chromecast*
iOS	*Track Chapters for iOS*
JavaScript	*Track Chapters for JavaScript*
Primetime	• **Android**: *Configure Video Analytics* • **DHLS**: *Configure Video Analytics* • **iOS**: *Configure Video Analytics*
TVML	*Track Chapters for TVML*

***** For all 1.x SDKs, the links are for the full PDF download of the documentation.

Validate

Chapter Start

On start of an individual chapter playback, one key calls are sent:

• Heartbeat chapter start*****

*****This call contains additional chapter metadata variables.

Chapter Complete

At the chapter boundary end, a Heartbeat chapter complete call will be sent.

Chapter Skip

When a chapter is skipped, a Heartbeat chapter skip call will be sent.

Track Quality of Experience

The following instructions provide guidance for implementation across all 2.x SDKs.

Important: *The following instructions provide guidance for implementation across all 2.x SDKs. If you are implementing a 1.x version of the SDK, you can download the Developers Guide further down the page.*

This topic contains the following information:

- *Overview*
- *Implement*
- *Code*
- *Validate*

Overview

Quality of experience tracking includes quality of service (QoS) and error tracking, both are optional elements and are **not** required for core video heartbeat implementations. You can use the video player API to identify the variables related to QoS and error tracking. Here are the key elements of tracking quality of experience:

Tip: *Additional details for each section is available in the Implement section.*

On all bitrate change events:

- Create/update the QoS object instance for the playback, `qosObject`
- Call `trackEvent(Media.Heartbeat.Event.BitrateChange, qosObject);`

On player errors:

- Call `trackError(“video error id”);`

Implement

To implement quality of experience and error tracking:

1. Identify when the bitrate changes during video playback and create the MediaObject instance using the QoS information.

   Here is the QoSObject reference:

Variable	Description	Required
`bitrate`	Current bitrate	Yes
`startupTime`	Startup time	Yes
`fps`	FPS value	Yes
`droppedFrames`	Number of dropped frames	Yes

Tip: *These variables are only required if you are planning to track QoS.*

The general format of the QoS object is:

```
var qosObject =
  MediaHeartbeat.createQoSObject(<bitrate>, <startuptime>, <fps>, <droppedFrames>);
```

2. When playback switches bitrates, call the `BitrateChange` event in the `MediaHeartbeat` instance.

```
mediaHeartbeat.trackEvent(MediaHeartbeat.Event.BitrateChange, qosObject);
```

Important: *Update the QoS object and call the bitrate change event on every bitrate change. This provides the most accurate QoS data.*

3. Make sure the `MediaHeartbeatDelegate.getQoSObject()` method returns the most updated QoS information.
4. When the video player encounters an error, and the error event is available to the player API, use the `trackError()` MediaHeartbeat event to capture the error information.

```
mediaHeartbeat.trackError("videoErrorId");
```

Tip: *Tracking video player errors will not stop the video tracking session. If the video player error prevents the playback from continuing, make sure that the video tracking session is closed by calling* `trackSessionEnd()` *after calling* `trackError()`.

The following sample code uses the JavaScript 2.x SDK for an HTML5 video player. You should use this code with the core video playback code.

```
/* Call on bitrate change */
if (e.type == "bitrate change") {
    var qosObject = MediaHeartbeat.createQoSObjectt(24,5,29,2);
    this.mediaHeartbeat.trackEvent(MediaHeartbeat.Event.BitrateChange, qosObject);
};

/*Call on player error*/
if (e.type == "error") {
    this.mediaHeartbeat.trackError("video error 10345");
};
```

Code

Video Analytics 2.x SDKs	Developer Guides
Android/FireTV	*Track Quality for Android*
iOS/AppleTV	*Track Quality for iOS*
JavaScript	*Track Quality for JavaScript*
Roku	*Track Quality for Roku*
Chromecast	*Track Quality for Chromecast*

Video Analytics 1.x SDKs*	Developer Guides
Android	*Track Quality for Android*
AppleTV	*Track Quality for AppleTV*
Chromecast	*Track Quality for Chromecast*
iOS	*Track Quality for iOS*
JavaScript	*Track Quality for JavaScript*

Video Analytics 1.x SDKs*	**Developer Guides**
Primetime	• **Android**: *Configure Video Analytics* • **DHLS**: *Configure Video Analytics* • **iOS**: *Configure Video Analytics*
TVML	*Track Quality for TVML*

***** For all 1.x SDKs, the links are for the full PDF download of the documentation.

Validate

Bitrate change

On each bitrate change, a Heartbeat `bitrate_change` call will be sent, which includes the QoS variables.

Error

On player error, a Heartbeat error call will be sent with the error value included.

VHL 1.x to 2.x Migration

Migration Overview

The migration from VHL 1.x to VHL 2.x is straightforward, with the new version featuring simplified APIs for initialization, configuration, and player delegates.

Here are the primary differences between 1.x and 2.x:

- **Plugins, Delegates** - You no longer need to implement plugins and delegates for Analytics, VideoPlayer, and Heartbeat.
- **Configuration** - You no longer need to instantiate configurations for the 1.x plugins.

In versions 2.x, all of the public methods are consolidated into the `MediaHeartbeat` class to make it easier on developers. Also, all configs are now consolidated into the `MediaHeartbeatConfig` class. These new APIs are described in detail here: *VHL 1.x to 2.x API Conversion*.

In version 2.x, you do not need to implement plugins or delegates for Analytics, VideoPlayer, or Heartbeat. Also, you no longer need to instantiate configs for all of these plugins. In the 2.x SDK you only need to instantiate the `MediaHeartbeat` class with `MediaHeartbeatDelegate` and `MediaHeartbeatConfig` instances. This is the only implementation that is required to initialize Video Analytics.

With the initialization of `MediaHeartbeat`, you can safely delete all of the implementation for Analytics Plugin, VideoPlayer Plugin and Heartbeat Plugin. Also, remove all the existing implementation for VideoHeartbeat initialization that takes in an array of plugins as an input. You can see side-by-side comparisons of the 1.x and 2.x implementations here: *VHL Code Comparison: 1.x to 2.x*

VHL Code Comparison: 1.x to 2.x

All of the VHL configuration parameters and tracking APIs are now consolidated into the `MediaHeartbeats` and `MediaHeartbeatConfig` classes.

Configuration API changes:

- `AdobeHeartbeatPluginConfig.sdk` - Renamed to `MediaConfig.appVersion`
- `MediaHeartbeatConfig.playerName` - Now set through `MediaHeartbeatConfig` instead of `VideoPlayerPluginDelegate`
- (For JavaScript only): The `AppMeasurement` instance - Now sent through the `MediaHeartbeat` constructor.

Configuration properties changes:

- `sdk` - Renamed to `appVersion`
- `publisher` - Removed; Marketing Cloud Org ID is used instead as a publisher
- `ovp` - Removed
- `quiteMode` - Removed

The following tables provide side-by-side code comparisons between VHL 1.x and VHL 2.x, covering Initialization, Core Playback, Ad Playback, Chapter Playback, and some additional events.

Table 3: VHL Code Comparison: INITIALIZATION

VHL 1.x API	VHL 2.x API
Object Initialization **1.x:** • `Heartbeat()` • `VideoPlayerPlugin()` • `AdobeAnalyticsPlugin()` • `HeartbeatPlugin()`	... **2.x:** • `MediaHeartbeat()` • `MediaHeartbeatConfig()`

Set up the video player plugin:
`this._playerPlugin =`
`  new VideoPlayerPlugin(`
`    new`
`SampleVideoPlayerPluginDelegate(this._player));`
`var playerPluginConfig =`
`  new VideoPlayerPluginConfig();`
`playerPluginConfig.debugLogging = true;`
 	
`// Set up the AppMeasurement plugin`
`this._aaPlugin =`
`  new AdobeAnalyticsPlugin(`
`    appMeasurement,`
`    new SampleAdobeAnalyticsPluginDelegate());`
`var aaPluginConfig = new`
`AdobeAnalyticsPluginConfig();`
 	
`aaPluginConfig.channel =`
`  Configuration.HEARTBEAT.CHANNEL;`
 	
`aaPluginConfig.debuglogging = true;`
`this._aaPlugin.configure(aaPluginConfig);`
 	
`// Set up the AdobeHeartbeat plugin`
`var ahPlugin =`
`  new AdobeHeartbeatPlugin(`
`    new SampleAdobeHeartbeatPluginDelegate());`
`var ahPluginConfig = new AdobeHeartbeatPluginConfig(`
 	
`    configuration.HEARTBEAT.TRACKING_SERVER,`
`    configuration.HEARTBEAT.PUBLISHER);`
`ahPluginConfig.ovp = configuration.HEARTBEAT.OVP;`
`ahPluginConfig.sdk = configuration.HEARTBEAT.SDK;`
`ahPluginConfig.debugLogging = true;`
`ahPlugin.configure(ahPluginConfig);`
 	
`var plugins =`
`  [this._playerPlugin, this._aaPlugin, ahPlugin];`
 	
`// Set up and configure the heartbeat library`
`this._heartbeat =`
`  new Heartbeat(new SampleHeartbeatDelegate(),`
`              plugins);`
`var configData = new HeartbeatConfig();`
`configData.debugLogging = true;`
`this._heartbeat.configure(configData);`
1.x Sample Player
...
	Media Heartbeat initialization:
	`var mediaConfig =`
	`  new MediaHeartbeatConfig();`
	`mediaConfig.trackingServer =`
	`  Configuration.HEARTBEAT.TRACKING_SERVER;`
	`mediaConfig.playerName =`
	`  Configuration.PLAYER.NAME;`
	`mediaConfig.debugLogging = true;`
	`mediaConfig.channel =`
	`  Configuration.HEARTBEAT.CHANNEL;`
	`mediaConfig.ssl = false;`
	`mediaConfig.ovp =`
	`  Configuration.HEARTBEAT.OVP;`
	`mediaConfig.appVersion =`
	`  Configuration.HEARTBEAT.SDK;`
	
	`this._mediaHeartbeat = new MediaHeartbeat(`
	`    new SampleMediaHeartbeatDelegate(this._player),`
	
	`     mediaConfig,`
	`     appMeasurement);`
	2.x Sample Player
	...

Delegates	...

VHL 1.x API	VHL 2.x API
1.x: • `VideoPlayerPluginDelegate()` • `VideoPlayerPluginDelegate().getVideoInfo` • `VideoPlayerPluginDelegate().getAdBreakInfo` • `VideoPlayerPluginDelegate().getAdInfo` • `VideoPlayerPluginDelegate().getChapterInfo` • `VideoPlayerPluginDelegate().getQoSInfo` • `VideoPlayerPluginDelegate().get.onError` • `AdobeAnalyticsPluginDelegate()` • `AdobeHeartbeatPluginDelegate()`	**2.x:** • `MediaHeartbeatDelegate()` • `MediaHeartbeatDelegate().getCurrentPlaybackTime` • `MediaHeartbeatDelegate().getQoSObject`

VideoPlayerPluginDelegate:

```
$.extend(SampleVideoPlayerPluginDelegate.prototype,

         VideoPlayerPluginDelegate.prototype);

function SampleVideoPlayerPluginDelegate(player) {
    this._player = player;
}

SampleVideoPlayerPluginDelegate.prototype.getVideoInfo
 =
  function() {
      return this._player.getVideoInfo();
  };

SampleVideoPlayerPluginDelegate.prototype.getAdBreakInfo
 =
  function() {
      return this._player.getAdBreakInfo();
  };

SampleVideoPlayerPluginDelegate.prototype.getAdInfo
 =
  function() {
      return this._player.getAdInfo();
  };

SampleVideoPlayerPluginDelegate.prototype.getChapterInfo
 =
  function() {
      return this._player.getChapterInfo();
  };

SampleVideoPlayerPluginDelegate.prototype.getQoSInfo
 =
  function() {
      return this._player.getQoSInfo();
  };
```

Sample 1.x Player

...

AdobeAnalyticsPluginDelegate:

```
$.extend(SampleAdobeAnalyticsPluginDelegate.prototype,

         AdobeAnalyticsPluginDelegate.prototype);

function SampleAdobeAnalyticsPluginDelegate() {}
```

MediaHeartbeatDelegate:

```
ADB.core.extend(SampleMediaHeartbeatDelegate.prototype,

  MediaHeartbeatDelegate.prototype);

function SampleMediaHeartbeatDelegate(player) {
   this._player = player;
}

SampleMediaHeartbeatDelegate.prototype.getCurrentPlaybackTime
 =
  function() {
      return this._player.getCurrentPlaybackTime();
  };

SampleMediaHeartbeatDelegate.prototype.getQoSObject
 =
  function() {
      return this._player.getQoSInfo();
  };

this._mediaHeartbeat =
  new MediaHeartbeat(new
        SampleMediaHeartbeatDelegate(this._player),

        mediaConfig,
        appMeasurement);
```

Sample 2.x Player

...

VHL 1.x API	VHL 2.x API

```
SampleAdobeAnalyticsPluginDelegate.prototype.onError
 =
  function(errorInfo) {
      console.log("AdobeAnalyticsPlugin error: " +
                   errorInfo.getMessage() +
                   " | " +
                   errorInfo.getDetails());
  };
```
 Sample 1.x Player
 	
...
 	
HeartbeatDelegate:
```
$.extend(SampleHeartbeatDelegate.prototype,
         HeartbeatDelegate.prototype);

function SampleHeartbeatDelegate() {}

SampleHeartbeatDelegate.prototype.onError =
  function(errorInfo) {
      console.log("Heartbeat error: " +
                  errorInfo.getMessage() +
                  " | " +
                  errorInfo.getDetails());
};
```
 Sample 1.x Player	

Table 4: VHL Code Comparison: CORE PLAYBACK

VHL 1.x	VHL 2.x
Session Start **1.x:** • `VideoPlayerPluginDelegate.trackVideoLoad()` • `VideoPlayerPluginDelegate.getVideoInfo()`	... **2.x:** • `MediaHeartbeat.createMediaObject()` • `MediaHeartbeat.trackSessionStart()`

```
VideoAnalyticsProvider.prototype._onLoad =
  function() {
    this._playerPlugin.trackVideoLoad();
};

SampleVideoPlayerPluginDelegate.prototype.getVideoInfo

  = function() {
    return this._player.getVideoInfo();
};

VideoPlayer.prototype.getVideoInfo = function() {
    this._videoInfo.playhead = vTime;
    return this._videoInfo;
};
```
 1.x Sample Player - trackVideoLoad()
 	
1.x Sample Player - getVideoInfo()
	```
	VideoAnalyticsProvider.prototype._onLoad =
	  function() {
	    var contextData = {};
	    var videoInfo = this._player.getVideoInfo();
	    var mediaInfo =
	      MediaHeartbeat.createMediaObject(
	        videoInfo.name,
	        videoInfo.id,
	        videoInfo.length,
	        videoInfo.streamType);

	    this._mediaHeartbeat.trackSessionStart(
	      mediaInfo,
	      contextData);
	};
	```
	 2.x Sample Player - createMediaObject()
	
	...

VHL 1.x	VHL 2.x
...	
Standard Video Metadata **1.x:** • `VideoMetadataKeys()` • `AdobeAnalyticsPlugin.setVideoMetadata90`	... **2.x:** • `MediaHeartbeat.createMediaObject()` • `MediaHeartbeat.trackSessionStart()`

`VideoAnalyticsProvider.prototype._onLoad =`
`  function() {`
`    console.log('Player event: VIDEO_LOAD');`
 	
`    var contextData = {};`
`    // Setting Standard Video Metadata`
`    contextData[VideoMetadataKeys.SEASON] =`
`      "sample season";`
`    contextData[VideoMetadataKeys.SHOW] =`
`      "sample show";`
`    contextData[VideoMetadataKeys.EPISODE] =`
`      "sample episode";`
`    contextData[VideoMetadataKeys.ASSET_ID] =`
 	
`      "sample asset id";`
`    contextData[VideoMetadataKeys.GENRE] =`
`      "sample genre";`
 	
`contextData[VideoMetadataKeys.FIRST_AIR_DATE]`
` =`
`      "sample air date";`
`    // Etc.`
 	
`this._aaPlugin.setVideoMetadata(contextData);`
`    this._playerPlugin.trackVideoLoad();`
`  };`
Sample 1.x Player
	`VideoAnalyticsProvider.prototype._onLoad =`
	`  function() {`
	`    console.log('Player event: VIDEO_LOAD');`
	`    var contextData = {};`
	
	`    var mediaInfo =`
	
	`MediaHeartbeat.createMediaObject(videoInfo.name,`
	
	`videoInfo.id,`
	
	`videoInfo.length,`
	
	`videoInfo.streamType);`
	
	`    // Set standard Video Metadata`
	`    var standardVideoMetadata = {};`
	
	`standardVideoMetadata[VideoMetadataKeys.SEASON]`
	` =`
	`      "sample season";`
	
	`standardVideoMetadata[VideoMetadataKeys.SHOW]`
	`=`
	`      "sample show";`
	
	`standardVideoMetadata[VideoMetadataKeys.EPISODE]`
	` =`
	`      "sample episode";`

VHL 1.x	VHL 2.x
...	`standardVideoMetadata[VideoMetadataKeys.ASSET_ID] =     "sample asset id";` `standardVideoMetadata[VideoMetadataKeys.GENRE] =     "sample genre";` `standardVideoMetadata[VideoMetadataKeys.FIRST_AIR_DATE] =     "sample air date";` `    // Etc.` `mediaInfo.setValue(MediaHeartbeat.MediaObjectKey.StandardVideoMetadata,                     standardVideoMetadata);` `this._mediaHeartbeat.trackSessionStart(mediaInfo, contextData);` `  };` *Sample 2.x Player* ...
Note: *Insetad of setting the Standard Video Metadata through the `AdobeAnalyticsPlugin.setVideoMetadata()` API, in VHL 2.0, the Standard Video Metadata is set through the MediaObject key `MediaObject.MediaObjectKey.StandardVideoMetadata()`.*	
Custom Video Metadata **1.x:** • `VideoMetadataKeys()` • `AdobeAnalyticsPlugin.setVideoMetadata()`	... **2.x:** • `MediaHeartbeat.createMediaObject()` • `MediaHeartbeat.trackSessionStart()`
`VideoAnalyticsProvider.prototype._onLoad =` `  function() {` `    var contextData = {` `        isUserLoggedIn: "false",` `        tvStation: "Sample TV station",` `        programmer: "Sample programmer"` `    };` `this._aaPlugin.setVideoMetadata(contextData);` `    this._playerPlugin.trackVideoLoad();` `  };` *Sample 1.x Player*	`VideoAnalyticsProvider.prototype._onLoad =` `  function() {` `    var contextData = {` `        isUserLoggedIn: "false",` `        tvStation: "Sample TV station",` `        programmer: "Sample programmer"` `    };` `    var videoInfo =` `this._player.getVideoInfo();` `    var mediaInfo =` `MediaHeartbeat.createMediaObject(videoInfo.name,`

VHL 1.x	VHL 2.x

...	`videoInfo.id,`
	
	`videoInfo.length,`
	
	`videoInfo.streamType);`
	
	`mediaInfo.setValue(MediaHeartbeat.MediaObjectKey.StandardVideoMetadata,`
	`standardVideoMetadata);`
	
	`this._mediaHeartbeat.trackSessionStart(mediaInfo,`
	` contextData);`
	`  };`
	
	Sample 2.x Player
	
	...

Note: *Instead of setting the Custom Video Metadata through the `AdobeAnalyticsPlugin.setVideoMetadata()` API, in VHL 2.0, the Standard Video Metadata is set through the `MediaHeartbeat.trackSessionStart()` API.*	
Playback **1.x:** • `VideoPlayerPlugin.trackPlay()`	... **2.x:** • `MediaHeartbeat.trackPlay()`

`VideoAnalyticsProvider.prototype._onSeekStart`
` =`
`  function() {`
`    console.log('Player event: SEEK_START');`
`    this._playerPlugin.trackSeekStart();`
`  };`
 	
Sample 1.x Player
 	
...
	`VideoAnalyticsProvider.prototype._onSeekStart`
	`=`
	`  function() {`
	`    console.log('Player event: SEEK_START');`
	
	`this._mediaHeartbeat.trackEvent(MediaHeartbeat.Event.SeekStart);`
	`};`
	
	Sample 2.x Player
	
	...

Pause **1.x:** • `VideoPlayerPlugin.trackPause()`	... **2.x:** • `MediaHeartbeat.trackPausel()`

`VideoAnalyticsProvider.prototype._onPause =`
`  function() {`
`    console.log('Player event:X PAUSE');`
`    this._playerPlugin.trackPause();`
`  };`
 	
Sample 1.x Player
	`VideoAnalyticsProvider.prototype._onBufferComplete`
	` =`
	`  function() {`
	`    console.log('Player event:`
	`BUFFER_COMPLETE');`
	
	`this._mediaHeartbeat.trackEvent(MediaHeartbeat.Event.BufferComplete);`
	
	`  };`

VHL 1.x	VHL 2.x
...	Sample Player 2.x ...
Seek Complete **1.x:** • `VideoPlayerPlugin.trackSeekComplete()`	... **2.x:** • `MediaHeartbeat.trackEvent(MediaHeartbeat.Event.SeekComplete)`
`VideoAnalyticsProvider.prototype._onSeekComplete =` `function() {` `console.log('Player event: SEEK_COMPLETE');` `this._playerPlugin.trackSeekComplete();` `};` *Sample 1.x Player* ...	`VideoAnalyticsProvider.prototype._onSeekComplete =` `function() {` `console.log('Player event: SEEK_COMPLETE');` `this._mediaHeartbeat.trackEvent(MediaHeartbeat.Event.SeekComplete);` `};` *Sample Player 2.x* ...
Buffer Start **1.x:** • `VideoPlayerPlugin.trackBufferStart()`	... **2.x:** • `MediaHeartbeat.trackEvent(MediaHeartbeat.Event.BufferStart)`
`VideoAnalyticsProvider.prototype._onBufferStart =` `function() {` `console.log('Player event: BUFFER_START');` `this._playerPlugin.trackBufferStart();` `};` *Sample 1.x Player* ...	`VideoAnalyticsProvider.prototype._onBufferStart =` `function() {` `console.log('Player event: BUFFER_START');` `this._mediaHeartbeat.trackEvent(MediaHeartbeat.Event.BufferStart);` `};` *Sample Player 2.x* ...
Buffer Complete **1.x:** • `VideoPlayerPlugin.trackBufferComplete()`	... **2.x:** • `MediaHeartbeat.trackEvent(MediaHeartbeat.Event.BufferComplete)`
`VideoAnalyticsProvider.prototype._onBufferComplete =` `function() {` `console.log('Player event: BUFFER_COMPLETE');` `this._playerPlugin.trackBufferComplete();` `};` *Sample 1.x Player*	`VideoAnalyticsProvider.prototype._onBufferComplete =` `function() {` `console.log('Player event: BUFFER_COMPLETE');` `this._mediaHeartbeat.trackEvent(MediaHeartbeat.Event.BufferComplete);` `};`

VHL 1.x	VHL 2.x
...	*Sample Player 2.x* ...
Playback Complete **1.x:** • `VideoPlayerPlugin.trackComplete()`	... **2.x:** • `MediaHeartbeat.trackComplete()`

`VideoAnalyticsProvider.prototype._onComplete`
`=`
`  function() {`
`    console.log('Player event: COMPLETE');`
`   this._playerPlugin.trackComplete(function()`
` {`
`        console.log(`
`          "The completion of the content has`
`been tracked.");`
`    });`
`  };`
 	
Sample 1.x Player
 	
...
	`VideoAnalyticsProvider.prototype._onComplete =`
	
	`  function() {`
	`    console.log('Player event: COMPLETE');`
	`    this._mediaHeartbeat.trackComplete();`
	`  };`
	
	Sample Player 2.x
	
	...

Table 5: VHL Code Comparison: AD PLAYBACK

VHL 1.x	VHL 2.x
Ad Start **1.x:** • `VideoPlayerPlugin.trackAdStart()` • `VideoPlayerPluginDelegate.getAdBreakInfo()` • `VideoPlayerPluginDelegate.getAdInfo()`	... **2.x:** • `MediaHeartbeat.createAdBreakObject()` • `MediaHeartbeat.createAdObject()` • `MediaHeartbeat.trackEvent(MediaHeartbeat.Event.AdBreakStart)` • `MediaHeartbeat.trackEvent(MediaHeartbeat.Event.AdStart)`

`VideoAnalyticsProvider.prototype._onAdStart =`
 	
`  function() {`
`    console.log('Player event: AD_START');`
`    this._playerPlugin.trackAdStart();`
`  };`
 	
Sample 1.x Player
 	
...
 	
`SampleVideoPlayerPluginDelegate.prototype.getAdInfo`
` =`
`  function() {`
`    return this._player.getAdInfo();`
`  };`
 	
Sample 1.x Player
	`VideoAnalyticsProvider.prototype._onAdStart =`
	
	`  function() {`
	`    console.log('Player event: AD_START');`
	`    var adContextData = {};`
	
	`    // AdBreak Info - getting the adBreakInfo`
	`from player and creating`
	`    // AdBreakInfo Object from MediaHeartbeat`
	`    var _adBreakInfo =`
	`this._player.getAdBreakInfo();`
	`    var adBreakInfo =`
	
	`MediaHeartbeat.createAdBreakObject(_adBreakInfo.name,`
	
	`_adBreakInfo.position,`

VHL 1.x	VHL 2.x

...	`_adBreakInfo.startTime);`
	
	`   // Ad Info - getting the adInfo from player and creating`
	`    // AdInfo Object from MediaHeartbeat`
	`    var _adInfo = this._player.getAdInfo();`
	`    var adInfo =`
	`MediaHeartbeat.createAdObject(_adInfo.name,`
	
	` _adInfo.id,`
	
	` _adInfo.position,`
	
	` _adInfo.length);`
	
	`this._mediaHeartbeat.trackEvent(MediaHeartbeat.Event.AdBreakStart,`
	
	`adBreakInfo);`
	`this._mediaHeartbeat.trackEvent(MediaHeartbeat.Event.AdStart,`
	`                                    adInfo,`
	
	`adContextData);`
	`  };`
	
	Sample 2.x Player
	
	...

Standard Ad Metadata **1.x:** • `AdMetadataKeys()` • `AdobeAnalyticsPlugin.setAdMetadata()`	... **2.x:** • `MediaHeartbeat.createAdObject()` • `MediaHeartbeat.trackAdStart()`

`VideoAnalyticsProvider.prototype._onAdStart =`
 	
`  function() {`
`    console.log('Player event: AD_START');`
 	
`    var contextData = {};`
`    // setting Standard Ad Metadata`
`    contextData[AdMetadataKeys.ADVERTISER] =`
`      "sample advertiser";`
`    contextData[AdMetadataKeys.CAMPAIGN_ID] =`
 	
`      "sample campaign";`
`    contextData[AdMetadataKeys.CREATIVE_ID] =`
 	
`      "sample creative";`
`    contextData[AdMetadataKeys.CREATIVE_URL]`
`=`
`      "sample url";`
`    contextData[AdMetadataKeys.SITE_ID] =`
`      "sample site";`
`    contextData[AdMetadataKeys.PLACEMENT_ID]`
`=`
`      "sample placement";`
	`VideoAnalyticsProvider.prototype._onAdStart =`
	
	`  function() {`
	`    console.log('Player event: AD_START');`
	`    var adContextData = { };`
	
	`    // AdBreak Info - getting the adBreakInfo from player and creating`
	`    // AdBreakInfo Object from MediaHeartbeat`
	`    var _adBreakInfo =`
	`this._player.getAdBreakInfo();`
	`    var adBreakInfo =`
	
	`MediaHeartbeat.createAdBreakObject(_adBreakInfo.name,`
	
	`_adBreakInfo.position,`
	
	`_adBreakInfo.startTime);`
	
	`   // Ad Info - getting the adInfo from player and creating`
	`    // AdInfo Object from MediaHeartbeat`
	`    var _adInfo = this._player.getAdInfo();`

VHL 1.x	VHL 2.x
`this._aaPlugin.setAdMetadata(contextData);` `this._playerPlugin.trackAdStart();` `};` *Sample 1.x Player* ...	`var adInfo =` `MediaHeartbeat.createAdObject(_adInfo.name,` `_adInfo.id,` `_adInfo.position,` `_adInfo.length);` `// Set standard Ad Metadata` `var standardAdMetadata = {};` `standardAdMetadata[MediaHeartbeat.AdMetadataKeys.ADVERTISER] =` `"Sample Advertiser";` `standardAdMetadata[MediaHeartbeat.AdMetadataKeys.CAMPAIGN_ID] =` `"Sample Campaign";` `adInfo.setValue(MediaHeartbeat.MediaObjectKey.StandardAdMetadata,` `standardAdMetadata);` `this._mediaHeartbeat.trackEvent(MediaHeartbeat.Event.AdBreakStart,` `adBreakInfo);` `this._mediaHeartbeat.trackEvent(MediaHeartbeat.Event.AdStart,` `adInfo,` `adContextData);` `};` *Sample 2.x Player* ...
Note: *Instead of setting the Standard Ad Metadata through the* `AdobeAnalyticsPlugin.setVideoMetadata()` *API, in VHL 2.0, the Standard Ad Metadata is set through the* `AdMetadata` *key* `MediaObject.MediaObjectKey.StandardVideoMetadata`	
Custom Ad Metadata **1.x:** • `AdobeAnalyticsPlugin.setAdMetadata()`	... **2.x:** • `MediaHeartbeat.createAdObject()` • `MediaHeartbeat.trackAdStart()`
`VideoAnalyticsProvider.prototype._onAdStart =` `function() {` `console.log('Player event: AD_START');`	`VideoAnalyticsProvider.prototype._onAdStart =` `function() {` `console.log('Player event: AD_START');` `var adContextData = {`

VHL 1.x	VHL 2.x

VHL 1.x

```
    var contextData = {};
    // setting Standard Ad Metadata
    contextData[AdMetadataKeys.ADVERTISER] =
      "sample advertiser";
    contextData[AdMetadataKeys.CAMPAIGN_ID] =

      "sample campaign";
    contextData[AdMetadataKeys.CREATIVE_ID] =

      "sample creative";
    contextData[AdMetadataKeys.CREATIVE_URL]
=
      "sample url";
    contextData[AdMetadataKeys.SITE_ID] =
      "sample site";
    contextData[AdMetadataKeys.PLACEMENT_ID]
=
      "sample placement";

   this._aaPlugin.setAdMetadata(contextData);

    this._playerPlugin.trackAdStart();
  };
```

Sample 1.x Player

...

VHL 2.x

```
        affiliate: "Sample affiliate",
        campaign: "Sample ad campaign"
    };

    // AdBreak Info - getting the adBreakInfo
from player and creating
    // AdBreakInfo Object from MediaHeartbeat
    var _adBreakInfo =
this._player.getAdBreakInfo();
    var adBreakInfo =

MediaHeartbeat.createAdBreakObject(_adBreakInfo.name,


_adBreakInfo.position,

_adBreakInfo.startTime);

    // Ad Info - getting the adInfo from player
 and creating
    // AdInfo Object from MediaHeartbeat
    var _adInfo = this._player.getAdInfo();
    var adInfo =

MediaHeartbeat.createAdObject(_adInfo.name,
                                   _adInfo.id,


_adInfo.position,

_adInfo.length);


this._mediaHeartbeat.trackEvent(MediaHeartbeat.Event.AdBreakStart,


adBreakInfo);

this._mediaHeartbeat.trackEvent(MediaHeartbeat.Event.AdStart,

                                    adInfo,

adContextData);
  };
```

Sample 2.x Player

...

Note: *Instead of setting the Custom Ad Metadata through the* `AdobeAnalyticsPlugin.setVideoMetadata` *API, in VHL 2.0, the Standard Ad Metadata is set through the* `MediaHeartbeat.trackAdStart()` *API.*

Ad Skip **1.x:** • `AdobeAnalyticsPlugin.setAdMetadata()`	... **2.x:** • `MediaHeartbeat.createAdObject()`

VHL 1.x	VHL 2.x
	• `MediaHeartbeat.trackAdStart()`

`SampleVideoPlayerPluginDelegate.prototype.getAdInfo =`
`  function() {`
`    return this._player.getAdInfo();`
`  };`
 	
Sample 1.x Player
 	
...
	`VideoAnalyticsProvider.prototype._onAdSkip =`
	`  function() {`
	`    console.log('Player event: AD_SKIP');`
	`this._mediaHeartbeat.trackEvent(MediaHeartbeat.Event.AdSkip);`
	`  };`

Note: *In VHL 1.5.X APIs;* `getAdinfo()` *and* `getAdBreakInfo()` *must return null if the player is outside the Ad break boundaries.*	
Ad Complete **1.x:** • `VideoPlayerPlugin.trackAdComplete()`	... **2.x:** • `MediaHeartbeat.trackEvent(MediaHeartbeat.Event.AdComplete)` • `MediaHeartbeat.trackEvent(MediaHeartbeat.Event.AdBreakComplete)`

`VideoAnalyticsProvider.prototype._onAdComplete =`
`  function() {`
`    console.log('Player event: AD_COMPLETE');`
`    this._playerPlugin.trackAdComplete();`
`  };`
 	
Sample 1.x Player
 	
...
	`VideoAnalyticsProvider.prototype._onAdComplete =`
	`  function() {`
	`    console.log('Player event: AD_COMPLETE');`
	`this._mediaHeartbeat.trackEvent(MediaHeartbeat.Event.AdComplete);`
	`this._mediaHeartbeat.trackEvent(MediaHeartbeat.Event.AdBreakComplete);`
	`  };`
	
	Sample 2.x Player
	
	...

Table 6: VHL Code Comparison: CHAPTER PLAYBACK

VHL 1.x	VHL 2.x
Chapter Start **1.x:** • `VideoPlayerPluginDelegate.getChapterInfo()` • `VideoPlayerPlugin.trackChapterStart()`	... **2.x:** • `MediaHeartbeat.createChapterObject` • `MediaHeartbeat.trackEvent(MediaHeartbeat.Event.ChapterStart)`

`VideoAnalyticsProvider.prototype._onChapterStart =`
`  function() {`
`    console.log('Player event: CHAPTER_START');`
	`VideoAnalyticsProvider.prototype._onChapterStart =`
	`  function() {`
	`    console.log('Player event: CHAPTER_START');`

VHL 1.x	VHL 2.x

`    this._playerPlugin.trackChapterStart();`
`  };`
Sample 1.x Player
...
`SampleVideoPlayerPluginDelegate.prototype.getChapterInfo`
` =`
`  function() {`
`    return this._player.getChapterInfo();`
`  };`
Sample 1.x Player
...
	`    var chapterContextData = { };`
	
	`    // Chapter Info - getting the chapterInfo from player and creating`
	`    // ChapterInfo Object from MediaHeartbeat`
	`    var _chapterInfo = this._player.getChapterInfo();`
	`    var chapterInfo =`
	`MediaHeartbeat.createChapterObject(_chapterInfo.name,`
	`_chapterInfo.position,`
	`_chapterInfo.length,`
	`_chapterInfo.startTime);`
	
	`this._mediaHeartbeat.trackEvent(MediaHeartbeat.Event.ChapterStart,`
	`chapterInfo,`
	`chapterContextData);`
	`  };`
	Sample 2.x Player
	...

Chapter Skip **1.x:** • `VideoPlayerPluginDelegate.getChapterInfo()`	... **2.x:** • `MediaHeartbeat.trackEvent(MediaHeartbeat.Event.ChapterSkip)`

`SampleVideoPlayerPluginDelegate.prototype.getChapterInfo`
` =`
`  function() {`
`    return this._player.getChapterInfo();`
`  };`
Sample 1.x Player
...
	`VideoAnalyticsProvider.prototype._onChapterSkip`
	` =`
	`  function() {`
	`this._mediaHeartbeat.trackEvent(MediaHeartbeat.Event.ChapterSkip);`
	`  };`

Note: *In VHL 1.5.X APIs;* `getChapterinfo()` *must return null if the player is outside the Chapter boundaries.*	
Chapter Custom Metadata **1.x:** • `VideoPlayerPlugin.trackChapterStart()` • `AdobeAnalyticsPlugin.setChapterMetadata()`	... **2.x:** • `MediaHeartbeat.createChapterObject()` • `MediaHeartbeat.trackEvent(MediaHeartbeat.Event.ChapterStart)`

`VideoAnalyticsProvider.prototype._onChapterStart`
` =`
	`VideoAnalyticsProvider.prototype._onChapterStart`
	` =`

VHL 1.x

```
    function() {
      console.log('Player event: CHAPTER_START');

        this._aaPlugin.setChapterMetadata({
            segmentType: "Sample segment type"
        });
        this._playerPlugin.trackChapterStart();
    };
```

Sample 1.x Player

...

VHL 2.x

```
    function() {
      console.log('Player event: CHAPTER_START');

        var chapterContextData = {
            segmentType: "Sample segment type"
        };

        // Chapter Info - getting the chapterInfo
from player and creating
        // ChapterInfo Object from MediaHeartbeat
        var _chapterInfo =
this._player.getChapterInfo();
        var chapterInfo =

MediaHeartbeat.createChapterObject(_chapterInfo.name,


_chapterInfo.position,

_chapterInfo.length,

_chapterInfo.startTime);


this._mediaHeartbeat.trackEvent(MediaHeartbeat.Event.ChapterStart,


chapterInfo,

chapterContextData);
    };
```

Sample Player 2.x

...

Note:

Chapter Complete

1.x:

- `trackChapterComplete()`

...

2.x:

- `trackEvent(MediaHeartbeat.Event.ChapterComplete)`

```
VideoAnalyticsProvider.prototype._onChapterComplete
 =
    function() {
        console.log('Player event:
CHAPTER_COMPLETE');
       this._playerPlugin.trackChapterComplete();

    };
```

Sample 1.x Player

```
VideoAnalyticsProvider.prototype._onChapterComplete
 =
    function() {
        console.log('Player event:
CHAPTER_COMPLETE');

this._mediaHeartbeat.trackEvent(MediaHeartbeat.Event.ChapterComplete);

    };
```

Sample Player 2.x

VHL 1.x	VHL 2.x
...	...

Table 7: VHL Code Comparison: OTHER EVENTS

VHL 1.x	VHL 2.x
Bitrate Change **1.x:** • `VideoPlayerPlugin.trackBitrateChange()`	... **2.x:** • `MediaHeartbeat.trackEvent(MediaHeartbeat.Event.BitrateChange)`
`VideoAnalyticsProvider.prototype._onBitrateChange` `=` `function() {` `console.log('Player event:` `BITRATE_CHANGE');` `// Update getQosInfo to return the updated` `bitrate` `this._playerPlugin.trackBitrateChange();` `};`	`VideoAnalyticsProvider.prototype._onBitrateChange` `=` `function() {` `console.log('Player event:` `BITRATE_CHANGE');` `// Update getQosObject to return the` `updated bitrate` `this._mediaHeartbeat.trackEvent(MediaHeartbeat.Event.BitrateChange);` `};` *Sample 2.x Player* ...
Video Resume **1.x:** • `VideoInfo.resumed()` • `VideoPlayerPluginDelegate.getVideoInfo()` • `VideoPlayerPlugin.trackVideoLoad()`	... **2.x:** • `MediaObject()` • `MediaHeartbeat.trackSessionStart()`
`this._videoInfo.resumed = true;` *Sample 1.x Player* ... `VideoPlayer.prototype.getVideoInfo =` `function() {` `this._videoInfo.playhead = vTime;` `return this._videoInfo;` `};` *Sample 1.x Player*	`VideoAnalyticsProvider.prototype._onLoad =` `function() {` `console.log('Player event: VIDEO_LOAD');` `var contextData = {};` `var videoInfo =` `this._player.getVideoInfo();` `var mediaInfo =` `MediaHeartbeat.createMediaObject(videoInfo.playerName,` `videoInfo.id,`

VHL 1.x	VHL 2.x

...	`videoInfo.length,`
	`videoInfo.streamType);`
	`mediaInfo.setValue(MediaHeartbeat.MediaObjectKey.VideoResumed,`
	`true);`
	`this._mediaHeartbeat.trackSessionStart(mediaInfo,`
	`contextData);`
	`};`
	Sample 2.x Player
	...

For more information on tracking video with 2.x, see *Track Core Video Playback* in *Measuring Video in Adobe Analytics*.

VHL 1.x to 2.x API Conversion

Table 8: Required Track APIs:

VHL 1.x	VHL 2.x
`videoPlayerPlugin.trackVideoLoad()`	N/A
`videoPlayerPlugin.trackSessionStart()`	*`mediaHeartbeat.trackSessionStart(mediaObject, mediaCustomMetadata)`*
`videoPlayerPlugin.trackPlay()`	*`mediaHeartbeat.trackPlay()`*
`videoPlayerPlugin.trackPause()`	*`mediaHeartbeat.trackPause()`*
`videoPlayerPlugin.trackComplete()`	*`mediaHeartbeat.trackComplete()`*
`videoPlayerPlugin.trackVideoUnload()`	*`mediaHeartbeat.trackSessionEnd()`*
`videoPlayerPlugin.trackApplicationError()`	N/A
`videoPlayerPlugin.trackVideoPlayerError()`	*`mediaHeartbeat.trackError()`*

All of the optional tracking APIs such as (Ads, Chapters, Bitrate change, Seeking, and Buffering) are now part of a single `trackEvent` API. The *`trackEvent`* API receives a constant parameter that represents the type of event that it is intended to track:

Table 9: Optional trackEvent APIs:

VHL 1.x	VHL 2.x
Return a valid `AdBreakInfo` in `VideoPlayerPlugin.getAdBreakInfo()`	`trackEvent(Event.AdBreakStart)`
Return null in `VideoPlayerPlugin.getAdBreakInfo()`	`trackEvent(Event.AdBreakComplete)`

VHL 1.x	VHL 2.x
`playerPlugin.trackAdStart()`	`trackEvent(Event.AdStart, adObject, adCustomMetadata)`
`playerPlugin.trackAdComplete()`	`trackEvent(Event.AdComplete)`
Return null in `VideoPlayerPlugin.getAdInfo()`	`trackEvent(Event.AdSkip)`
`playerPlugin.trackChapterStart()`	`trackEvent(Event.ChapterStart, chapterObject, chapterCustomMetadata)`
`playerPlugin.trackChapterComplete()`	`trackEvent(Event.ChapterComplete)`
Return null in `VideoPlayerPlugin.getChapterInfo()`	`trackEvent(Event.ChapterSkip)`
`playerPlugin.trackSeekStart()`	`trackEvent(Event.SeekStart)`
`playerPlugin.trackSeekComplete()`	`trackEvent(Event.SeekComplete)`
`playerPlugin.trackBufferStart()`	`trackEvent(Event.BufferStart)`
`playerPlugin.trackBufferComplete()`	`trackEvent(Event.BufferComplete)`
`playerPlugin.trackBitrateChange()`	`trackEvent(Event.BitrateChange)`
`playerPlugin.trackTimedMetadata()`	`trackEvent(Event.TimedMetadataUpdate)`

Metrics and Metadata

List of video content data, including context data values, that Adobe collects via solution variables.

- **Label:** The name of the parameter.
- **Implementation:** Information on implementation values and requirements
  - *Key* - Variable, set either manually in your app, or automatically by the Adobe VHL SDK.
  - *Required* - Indicates whether the parameter is required for basic video tracking.
  - *Type* - Specifies the type of the variable to be set, string or number.
  - *Sent With* - Indicates when the data is sent: *Initiate* is the analytics call sent on video start, *Ad Start* is the analytics call sent on ad start, *Chapter Start* is the analytics call sent on chapter start, and *Close* is the compiled analytics call sent directly from the heartbeat server to the analytics server at the end of the video session, or the end of the ad. The Close calls are not available in network packet calls.
  - *Min. SDK Version* - Indicates which SDK version you would need to access the parameter.
  - *Sample Value* - Provides example of common variable usage.
- **Network Parameters:** Displays the values that are passed to Adobe Analytics or Heartbeat servers. This column shows the names of the parameters that are seen in the network calls generated by Adobe VHL SDKs.
- **Reporting:** Information on how to view and analyze the video data.
  - *Available* - Indicates whether the data is available in reporting by default (*Yes*), or requires custom set-up (*Custom*)
  - *Reserved Variable* - Indicates whether the data is captured as an event, eVar, prop, or classification in a reserved variable.
  - *Report Name* - Name of Adobe Aanlytics report for variable
  - *Context Data* - Name of the Adobe Analytics context data passed to the reporting server and used in processing rules.
  - *Data Feed* - Column name for variable found in Clickstream or Live Stream data feeds
  - *Audience Manager* - Trait name found in Adobe Audience Manager

Video Parameters

Table 10: Core Video Data

Label	Implementation	Network Parameters	Reporting

Video Name	• **Key:** `name*`
	• **Required:** No
	• **Type:** string
	• **Sent with:** Initiate, Close
	• **Min. SDK Version:** 1.5.1
	• **Sample value:** `"The Big Bang Theory"`
		• **Adobe Analytics:**
		`a.media.friendlyName`
		• **Heartbeats:**
		`s:asset:name`
			• **Available:** Yes
			• **Reserved Variable:** eVar and classification
			• **Expiration:** On HIT
			• **Report Name:** Video Name, or Video Name (variable)
			• **Context Data:** `a.media.friendlyName`
			• **Data Feed:** `videoname`

Label	Implementation	Network Parameters	Reporting
			• **Audience Manager:** `c_contextdata.a.media.friendlyName`
	This is the "friendly" (human-readable) name of the video, equal to the last value of `s:asset:name`. In reporting, "Video Name" is the classification and "Video Name (variable)" is the eVar. ***** `createMediaObject(`***`name`***`, mediaId, length, streamType)`		

Content ID	• **Key:** **`mediaId*`**
	• **Required:** Yes
	• **Type:** string
	• **Sent with:** Initiate, Close
	• **Min. SDK Version:** Any
	• **Sample value:** `"4586695ABC"`
		• **Adobe Analytics:**
		`a.media.name`
		• **Heartbeats:**
		`s:asset:video_id`
			• **Available:** Yes
			• **Reserved Variable:** eVar
			• **Expiration:** On VISIT
			• **Report Name:** Content
			• **Context Data:** `a.media.name`
			• **Data Feed:** `video`
			• **Audience Manager:** `c_contextdata.a.media.name`

	Content ID of the content, which can be used to tie back to other industry / CMS IDs, equal to the last value of `s:asset:video_id`. Any integer and/or letter combination. ***** `createMediaObject(name, `***`mediaId`***`, length, streamType)`		

Video Length	• **Key:** **`length*`**
	• **Required:** Yes
	• **Type:** number
	• **Sent with:** Initiate, Close
	• **Min. SDK Version:** Any**
	• **Sample value:**
	  • VOD: 128
	  • Live: 86400
	  • Linear: 1800
		• **Adobe Analytics:**
		`a.media.length`
		• **Heartbeats:**
		`l:asset:length`
			• **Available:** Yes
			• **Reserved Variable:** eVar and classification
			• **Expiration:** On HIT
			• **Report Name:** Video Length, or [variable]
			• **Context Data:** `a.media.length`
			• **Data Feed:** `videolength`
			• **Audience Manager:** `c_contextdata.a.media.length`

	Clip Length/Runtime - This is the maximum length (or duration) of the content being consumed (in seconds). It equals the last value of `l:asset:length` from events of type Main. If `l:asset:length` is not set, then the last value of `l:asset:duration` is used. ***Important:*** *This property is used to compute several metrics, such as progress tracking metrics and Average Minute Audience. If this is not set, or not greater than zero, then these metrics are not available.* For Live videos with an unknown duration, the value of 86400 is the default.		

Label	Implementation	Network Parameters	Reporting
	In Reporting, "Video Length" is the classification and "Video Length (variable)" is the eVar. * `createMediaObject(name, mediaId, length, streamType)` ** Pre Version 1.5.1, this was `l:asset:duration`; after 1.5.1, this is `l:asset:length`.		
Content Type	• **Key:** `streamType*` • **Required:** Yes • **Type:** restricted string • **Sent with:** Initiate, Close • **Min. SDK Version:** Any • **Sample value:** `"vod"`	• **Adobe Analytics:** `a.contentType` • **Heartbeats:** `s:stream:type`	• **Available:** Yes • **Reserved Variable:** eVar • **Expiration:** On HIT • **Report Name:** Content Type • **Context Data:** `a.contentType` • **Data Feed:** `videocontenttype` • **Audience Manager:** `c_contextdata.a.contentType`
	Restricted to three values: `streamType.VOD`, `streamType.LIVE`, or `streamType.LINEAR`. This equals `s:stream:type`. If that is unset, this equals `missing_content_type`. * `createMediaObject(name, mediaId, length, streamType)`		
Video Session ID	• **Key:** Automatically set • **Required:** Yes • **Type:** number • **Sent with:** Initiate, Close • **Min. SDK Version:** 1.5.8 • **Sample value:** `1482236761294786918253`	• **Adobe Analytics:** `a.media.vsid` • **Heartbeat:** `s:event:sid`	• **Available:** Use processing rule • **Reserved Variable:** N/A • **Report Name:** Custom • **Context Data:** `a.media.vsid` • **Data Feed:** `vsid` • **Audience Manager:** `c_contextdata.a.media.vsid`
	This identifies an instance of a content stream unique to an individual playback.		
Content Player Name	• **Key:** `playerName*` • **Required:** Yes • **Type:** string • **Sent with:** Initiate, Close • **Min. SDK Version:** Any • **Sample value:** `"Brightcove"`, `"Primetime"`, etc.	• **Adobe Analytics:** `a.media.playerName` • **Heartbeats:** `s:sp:player_name`	• **Available:** Yes • **Reserved Variable:** eVar • **Expiration:** On HIT • **Report Name:** Content Player Name • **Context Data:** `a.media.playerName` • **Data Feed:** `videoplayername` • **Audience Manager:** `c_contextdata.a.media.playerName`
	Name of the player.		

Label	Implementation	Network Parameters	Reporting
	* *MediaHeartbeatConfig*.***playerName***		

Content Channel	• **Key:** `channel*`
	• **Required:** Yes
	• **Type:** string
	• **Sent with:** Initiate, Close
	• **Min. SDK Version:** Any
	• **Sample value:** `"Sports"`
		• **Adobe Analytics:**
		`a.media.channel`
		• **Heartbeats:**
		`s:sp:channel`
			• **Available:** Yes
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:** Content Channel
			• **Context Data:** `a.media.channel`
			• **Data Feed:** `videochannel`
			• **Audience Manager:** `c_contextdata.a.media.channel`

	Distribution Station/Channels or where the content is played. Any string value is accepted here. * *MediaHeartbeatConfig*.***channel***		
Content Segment	• **Key:** Automatically set • **Required:** Yes • **Type:** string • **Sent with:** Close • **Min. SDK Version:** Any • **Sample value:** `"[0-10]"` (minutes)	• **Adobe Analytics:** N/A • **Heartbeats:** N/A	• **Available:** Yes • **Reserved Variable:** eVar • **Expiration:** On HIT • **Report Name:** Content Segment • **Context Data:** `a.media.segment` • **Data Feed:** `videosegment` • **Audience Manager:** `c_contextdata.a.media.segment`
	The interval that describes the part of the content that has been viewed (in minutes). The segment is computed as min and max of the playhead values during a playback session.		

Video Path	• **Key:** Automatically set
	• **Required:** No
	• **Type:** string
	• **Sent with:** Initiate
	• **Min. SDK Version:** Any
	• **Sample value:** `"4586695ABC"`
		• **Adobe Analytics:**
		`a.media.name`
		• **Heartbeats:**
		`s:asset:video_id`
			• **Available:** Yes
			• **Reserved Variable:** prop
			• **Report Name:** Video Path
			• **Context Data:** `a.media.name`
			• **Data Feed:** `videopath`
			• **Audience Manager:** `c_contextdata.a.media.name`

	Ability to track path of viewer across site and/or App to see path they took to view a particular video. Any integer and/or letter combination.		

Label	Implementation	Network Parameters	Reporting
SDK Version	• **Key: `appVersion*`** • **Required:** No • **Type:** string • **Sent with:** Close • **Min. SDK Version:** 1.5.7 • **Sample value:** `"2.62.0_release"`	• **Adobe Analytics:** `a.media.sdkVersion` • **Heartbeats:** `s:sp:sdk`	• **Available:** Use custom processing rule • **Reserved Variable:** N/A • **Report Name:** • **Context Data:** `a.media.sdkVersion`` • **Data Feed:** `N/A` • **Audience Manager:** `c_contextdata.a.media.sdkVersion`
	The SDK version used by the player. This could have any custom value that makes sense for your player. Customers will have to create their own processing rules to have the value available for reporting. ***** *`MediaHeartbeatConfig`***`.appVersion`**		
VHL Version	• **Key:** Automatically set* • **Required:** No • **Type:** string • **Sent with:** Close • **Min. SDK Version:** 1.5.7 • **Sample value:** `"js-2.0.1.88-c8c0b1"`	• **Adobe Analytics:** `a.media.vhlVersion` • **Heartbeats:** `s:sp:hb_version`	• **Available:** Use custom processing rule • **Reserved Variable:** N/A • **Report Name:** Custom • **Context Data:** `a.media.vhlVersion` • **Data Feed:** `N/A` • **Audience Manager:** `c_contextdata.a.media.vhlVersion`
	The heartbeat SDK version used for the tracking session. Customers will have to create their own processing rules to have the value available for reporting. * *`MediaHeartbeat`*`.version();`		

Table 11: Standard Video Metadata

Label	Implementation	Network Parameters	Reporting
Show	• **Key:** `SHOW` • **Required:** No • **Type:** string • **Sent with:** Initiate, Close • **Min. SDK Version:** 1.5.7 • **Sample value:** `"Modern Family"`, `"Blacklist"`, `"New Girl"`, etc.	• **Adobe Analytics:** `a.media.show` • **Heartbeats:** `s:meta:a.media.show`	• **Available:** Yes • **Reserved Variable:** eVar • **Expiration:** On HIT • **Report Name:** Show • **Context Data:** `a.media.show` • **Data Feed:** `videoshow` • **Audience Manager:** `c_contextdata.a.media.show`

Label	Implementation	Network Parameters	Reporting
	Program/Series Name ***Note:*** *Program Name is required only if the show is part of a series.* *MediaHeartbeat.VideoMetadataKeys*		

Stream Format	• **Key:** `STREAM_FORMAT`
	• **Required:** No
	• **Type:** string
	• **Sent with:** Initiate, Close
	• **Min. SDK Version:** 1.5.7
	• **Sample value:** `"Live"`
		• **Adobe Analytics:**
		`a.media.format`
		• **Heartbeats:**
		`s:meta:a.media.format`
			• **Available:** Use custom processing rule
			• **Reserved Variable:** N/A
			• **Report Name:** Custom
			• **Context Data:** `a.media.format`
			• **Data Feed:** `N/A`
			• **Audience Manager:** `c_contextdata.a.media.format`

	Format of the stream (Live, VOD, Linear). *MediaHeartbeat.VideoMetadataKeys*		

Season	• **Key:** `SEASON`
	• **Required:** No
	• **Type:** string
	• **Sent with:** Initiate, Close
	• **Min. SDK Version:** 1.5.7
	• **Sample value:** `"2"` (an integer representing the season number)
		• **Adobe Analytics:**
		`a.media.season`
		• **Heartbeats:**
		`s:meta:a.media.season`
			• **Available:** Yes
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:** Season
			• **Context Data:** `a.media.season`
			• **Data Feed:** `videoseason`
			• **Audience Manager:** `c_contextdata.a.media.season`

	The season number the show belongs to. ***Note:*** *Season Series is required only if the show is part of a series.* *MediaHeartbeat.VideoMetadataKeys*		

Episode	• **Key:** `EPISODE`
	• **Required:** No
	• **Type:** string
	• **Sent with:** Initiate, Close
	• **Min. SDK Version:** 1.5.7
	• **Sample value:** `"13"` (an integer representing the episode number)
		• **Adobe Analytics:**
		`a.media.episode`
		• **Heartbeats:**
		`s:meta:a.media.episode`
			• **Available:** Yes
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:** Episode
			• **Context Data:** `a.media.episode`
			• **Data Feed:** `videoepisode`
			• **Audience Manager:** `c_contextdata.a.media.episode`

Label	Implementation	Network Parameters	Reporting
	The number of the episode. *MediaHeartbeat.VideoMetadataKeys*		

Asset ID	• **Key:** `ASSET_ID`
	• **Required:** No
	• **Type:** string
	• **Sent with:** Initiate, Close
	• **Min. SDK Version:** 1.5.7
	• **Sample value:** `"89745363"` (any integer and/or letter combination)
		• **Adobe Analytics:**
		`a.media.asset`
		• **Heartbeats:**
		`s:meta:a.media.asset`
			• **Available:** Use custom processing rule
			• **Reserved Variable:** N/A
			• **Report Name:** Custom
			• **Context Data:** `a.media.asset`
			• **Data Feed:** `N/A`
			• **Audience Manager:** `c_contextdata.a.media.asset`

	This is the unique identifier for the content of the video asset, such as the TV series episode identifier, movie asset identifier, or live event identifier. Typically these IDs are derived from metadata authorities such as EIDR, TMS/Gracenote, or Rovi. These identifiers can also be from other proprietary or in-house systems. *MediaHeartbeat.VideoMetadataKeys*		

Genre	• **Key:** `GENRE`
	• **Required:** No
	• **Type:** string
	• **Sent with:** Initiate, Close
	• **Min. SDK Version:** 1.5.7
	• **Sample value:** `"Drama"`, `"Comedy"`, etc.
		• **Adobe Analytics:**
		`a.media.genre`
		• **Heartbeats:**
		`s:meta:a.media.genre`
			• **Available:** Yes
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:** Genre
			• **Context Data:** `a.media.genre`
			• **Data Feed:** `videogenre`
			• **Audience Manager:** `c_contextdata.a.media.genre`

	Type or grouping of content as defined by content producer. *MediaHeartbeat.VideoMetadataKeys*		

First Air Date	• **Key:** `FIRST_AIR_DATE`
	• **Required:** No
	• **Type:** string
	• **Sent with:** Initiate
	• **Min. SDK Version:** 1.5.7
	• **Sample value:** `"2016-01-25"`
		• **Adobe Analytics:**
		`a.media.airDate`
		• **Heartbeats:**
		`s:meta:a.media.airDate`
			• **Available:** Use custom processing rule
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:** Custom
			• **Context Data:** `a.media.airDate`
			• **Data Feed:** `N/A`
			• **Audience Manager:** `c_contextdata.a.media.airDate`

Label	Implementation	Network Parameters	Reporting
	The date when the content first aired on television. Any date format is acceptable, but Adobe recommends: YYYY-MM-DD *MediaHeartbeat.VideoMetadataKeys*		

First Digital Date	• **Key:** `FIRST_DIGITAL_DATE`
	• **Required:** No
	• **Type:** string
	• **Sent with:** Initiate, Close
	• **Min. SDK Version:** 1.5.7
	• **Sample value:** `"2016-01-25"`
		• **Adobe Analytics:**
		`a.media.digitalDate`
		• **Heartbeats:**
		`s:meta:a.media.digitalDate`
			• **Available:** Use custom processing rule
			• **Reserved Variable:** N/A
			• **Report Name:** Custom
			• **Context Data:** `a.media.digitalDate`
			• **Data Feed:** `N/A`
			• **Audience Manager:** `c_contextdata.a.media.digitalDate`

	The date when the content first aired on any digital channel or platform. Any date format is acceptable but Adobe recommends: YYYY-MM-DD *MediaHeartbeat.VideoMetadataKeys*		

Content Rating	• **Key:** `RATING`
	• **Required:** No
	• **Type:** string
	• **Sent with:** Initiate, Close
	• **Min. SDK Version:** 1.5.7
	• **Sample value:** `TVY, TVG, TVPG, TVMA`, etc.
		• **Adobe Analytics:**
		`a.media.rating`
		• **Heartbeats:**
		`s:meta:a.media.rating`
			• **Available:** Use custom processing rule
			• **Reserved Variable:** N/A
			• **Report Name:** Custom
			• **Context Data:** `a.media.rating`
			• **Data Feed:** `N/A`
			• **Audience Manager:** `c_contextdata.a.media.rating`

	Rating as defined by TV Parental Guidelines. *MediaHeartbeat.VideoMetadataKeys*		

Originator	• **Key:** `ORIGINATOR`
	• **Required:** No
	• **Type:** string
	• **Sent with:** Initiate, Close
	• **Min. SDK Version:** 1.5.7
	• **Sample value:** `"Warner Brothers", "Sony", "Disney"`, etc.
		• **Adobe Analytics:**
		`a.media.orginator`
		• **Heartbeats:**
		`s:meta:a.media.orginator`
			• **Available:** Use custom processing rule
			• **Reserved Variable:** N/A
			• **Report Name:** Custom
			• **Context Data:** `a.media.orginator`
			• **Data Feed:** `N/A`
			• **Audience Manager:** `c_contextdata.a.media.orginator`

	Creator of the content. *MediaHeartbeat.VideoMetadataKeys*		

Label	Implementation	Network Parameters	Reporting

Network	• **Key:** `NETWORK`
	• **Required:** No
	• **Type:** string
	• **Sent with:** Initiate, Close
	• **Min. SDK Version:** 1.5.7
	• **Sample value:** `"Fox", "Bravo", "ESPN"`, etc.
		• **Adobe Analytics:**
		`a.media.network`
		• **Heartbeats:**
		`s:meta:a.media.network`
			• **Available:** Yes
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:** Network
			• **Context Data:** `a.media.network`
			• **Data Feed:** `videonetwork`
			• **Audience Manager:** `c_contextdata.a.media.network`

	The network/channel name. *`MediaHeartbeat.VideoMetadataKeys`*		

Show Type	• **Key:** `SHOW_TYPE`
	• **Required:** No
	• **Type:** string
	• **Sent with:** Initiate, Close
	• **Min. SDK Version:** 1.5.7
	• **Sample value:**
	• "0" = Full episode
	• "1" = Preview/trailer
	• "2" = Clip
	• "3" = Other
		• **Adobe Analytics:**
		`a.media.type`
		• **Heartbeats:**
		`s:meta:a.media.type`
			• **Available:** Yes
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:** Show Type
			• **Context Data:** `a.media.type`
			• **Data Feed:** `videoshowtype`
			• **Audience Manager:** `c_contextdata.a.media.type`

	Type of content, expressed as an integer between 0 and 3. *`MediaHeartbeat.VideoMetadataKeys`*		

MVPD	• **Key:** `MVPD`
	• **Required:** No
	• **Type:** string
	• **Sent with:** Initiate, Close
	• **Min. SDK Version:** 1.5.7
	• **Sample value:** `"Comcast", "DirecTV", "Dish"`, etc.
		• **Adobe Analytics:**
		`a.media.pass.mvpd`
		• **Heartbeats:**
		`s:meta:a.media.pass.mvpd`
			• **Available:** Yes
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:** MVPD
			• **Context Data:** `a.media.pass.mvpd`
			• **Data Feed:** `videomvpd`
			• **Audience Manager:** `c_contextdata.a.media.pass.mvpd`

	MVPD provided via Adobe authentication. *`MediaHeartbeat.VideoMetadataKeys`*		

Label	Implementation	Network Parameters	Reporting

Authorized	• **Key:** `AUTHORIZED`
	• **Required:** No
	• **Type:** string
	• **Sent with:** Initiate, Close
	• **Min. SDK Version:** 1.5.7
	• **Sample value:** `"TRUE"`
		• **Adobe Analytics:**
		`a.media.pass.auth`
		• **Heartbeats:**
		`s:meta:a.media.pass.auth`
			• **Available:** Yes
			• **Reserved Variable:** event
			• **Report Name:** Authorized
			• **Context Data:** `a.media.pass.auth`
			• **Data Feed:** `videoauthorized`
			• **Audience Manager:** `c_contextdata.a.media.pass.auth`

	The user has been authorized via AdobePass. ***Important:*** *This can only be true if it is set. If it is not set, no value is returned.* *`MediaHeartbeat.VideoMetadataKeys`*		

Day Part	• **Key:** `DAY_PART`
	• **Required:** No
	• **Type:** string
	• **Sent with:** Initiate, Close
	• **Min. SDK Version:** 1.5.7
	• **Sample value:**
		• **Adobe Analytics:**
		`a.media.dayPart`
		• **Heartbeats:**
		`s:meta:a.media.dayPart`
			• **Available:** Yes
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:** Day Part
			• **Context Data:** `a.media.dayPart`
			• **Data Feed:** `videodaypart`
			• **Audience Manager:** `c_contextdata.a.media.dayPart`

	A property that defines the time of the day when the content was broadcast or played. This could have any value set as necessary by customers. *`MediaHeartbeat.VideoMetadataKeys`*		

Video Feed Type	• **Key:** `FEED`
	• **Required:** No
	• **Type:** string
	• **Sent with:** Initiate, Close
	• **Min. SDK Version:** 1.5.7
	• **Sample value:** `"East-HD"`, `"West-HD"`, `"East-SD"`, etc.
		• **Adobe Analytics:**
		`a.media.feed`
		• **Heartbeats:**
		`s:meta:a.media.feed`
			• **Available:** Yes
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:** Video Feed Type
			• **Context Data:** `a.media.feed`
			• **Data Feed:** `videofeedtype`
			• **Audience Manager:** `c_contextdata.a.media.feed`

	Type of feed. *`MediaHeartbeat.VideoMetadataKeys`*		

Table 12: Video Metrics

Label	Implementation	Network Parameters	Reporting
Video Initiates	• **Key:** Automatically set • **Type:** string • **Sent with:** Initiate • **Min. SDK Version:** Any • **Sample value:** `TRUE`	• **Adobe Analytics:** `a.media.view` • **Heartbeats:** `s:event:type=start`	• **Available:** Yes • **Reserved Variable:** event • **Report Name:** Video Initiates • **Context Data:** `a.media.view` • **Data Feed:** `videostart` • **Audience Manager:** `c_contextdata.a.media.view`
	Load event for the video. (This occurs when the viewer clicks the **Play** button). This would count even if there are pre-roll ads, buffering, errors, and so on. ***Important:*** *This can only be true if it is set. If it is not set, no value is returned.*		
Content Starts	• **Key:** Automatically set • **Type:** string • **Sent with:** Close • **Min. SDK Version:** Any • **Sample value:** `TRUE`	• **Adobe Analytics:** N/A • **Heartbeats:** N/A	• **Available:** Yes • **Reserved Variable:** event • **Report Name:** Content Starts • **Context Data:** `a.media.play` • **Data Feed:** `videoplay` • **Audience Manager:** `c_contextdata.a.media.play`
	First frame of video is viewed. If viewer drops during ad, buffering, etc., then there would be no "Content Start" event. ***Important:*** *This can only be true if it is set. If it is not set, no value is returned.*		
Content Complete	• **Key:** Automatically set • **Type:** string • **Sent with:** Close • **Min. SDK Version:** Any • **Sample value:** `TRUE`	• **Adobe Analytics:** N/A • **Heartbeats:** `s:event:type=complete`	• **Available:** Yes • **Reserved Variable:** event • **Report Name:** Content Completes • **Context Data:** `a.media.complete` • **Data Feed:** `videocomplete` • **Audience Manager:** `c_contextdata.a.media.complete`
	A stream that was watched to completion - This does not necessarily mean the viewer watched the whole video; viewer could have skipped ahead. This only means viewer reached the end of the video, 100%. ***Important:*** *This can only be true if it is set. If it is not set, no value is returned.*		

Label	Implementation	Network Parameters	Reporting
Content Time Spent	• **Key:** Automatically set • **Type:** number • **Sent with:** Close • **Min. SDK Version:** Any • **Sample value:** `105`	• **Adobe Analytics:** N/A • **Heartbeats:** N/A	• **Available:** Yes • **Reserved Variable:** event • **Report Name:** Content Time Spent • **Context Data:** `a.media.timePlayed` • **Data Feed:** `videotime` • **Audience Manager:** `c_contextdata.a.media.timePlayed`
	Sums the event duration (in seconds) for all events of type PLAY on the main content.		
Video Time Spent	• **Key:** Automatically set • **Type:** number • **Sent with:** Close • **Min. SDK Version:** Any • **Sample value:** `120`	• **Adobe Analytics:** N/A • **Heartbeats:** N/A	• **Available:** Yes • **Reserved Variable:** event • **Report Name:** Video Time Spent • **Context Data:** `a.media.totalTimePlayed` • **Data Feed:** `videototaltime` • **Audience Manager:** `c_contextdata.a.media.totalTimePlayed`
	Sums the event duration (in seconds) for all events of type PLAY, both main and ad content.		
10% Progress Marker	• **Key:** Automatically set • **Type:** string • **Sent with:** Close • **Min. SDK Version:** Any • **Sample value:** `TRUE`	• **Adobe Analytics:** N/A • **Heartbeats:** N/A	• **Available:** Yes • **Reserved Variable:** event • **Report Name:** 10% Progress Marker • **Context Data:** `a.media.progress10` • **Data Feed:** `videoprogress10` • **Audience Manager:** `c_contextdata.a.media.progress10`
	Playhead passes the 10% marker of video based on video length. The marker is only counted once, even if seeking backwards. If seeking forward, markers that are skipped are not counted. ***Important:*** *This can only be true if it is set. If it is not set, no value is returned.*		
25% Progress Marker	• **Key:** Automatically set • **Type:** string • **Sent with:** Close • **Min. SDK Version:** Any • **Sample value:** `TRUE`	• **Adobe Analytics:** N/A • **Heartbeats:** N/A	• **Available:** Yes • **Reserved Variable:** event • **Report Name:** 25% Progress Marker • **Context Data:** `a.media.progress25`

Label	Implementation	Network Parameters	Reporting
			• **Data Feed:** `videoprogress25` • **Audience Manager:** `c_contextdata.a.media.progress25`
	Playhead passes the 25% marker of video based on video length. Marker only counted once, even if seeking backwards. If seeking forward, markers that are skipped are not counted. ***Important:*** *This can only be true if it is set. If it is not set, no value is returned.*		
50% Progress Marker	• **Key:** Automatically set • **Type:** string • **Sent with:** Close • **Min. SDK Version:** Any • **Sample value:** `TRUE`	• **Adobe Analytics:** N/A • **Heartbeats:** N/A	• **Available:** Yes • **Reserved Variable:** event • **Report Name:** 50% Progress Marker • **Context Data:** `a.media.progress50` • **Data Feed:** `videoprogress50` • **Audience Manager:** `c_contextdata.a.media.progress50`
	Playhead passes the 50% marker of video based on video length. Marker only counted once, even if seeking backwards. If seeking forward, markers that are skipped are not counted. ***Important:*** *This can only be true if it is set. If it is not set, no value is returned.*		
75% Progress Marker	• **Key:** Automatically set • **Type:** string • **Sent with:** Close • **Min. SDK Version:** Any • **Sample value:** `TRUE`	• **Adobe Analytics:** N/A • **Heartbeats:** N/A	• **Available:** Yes • **Reserved Variable:** event • **Report Name:** 75% Progress Marker • **Context Data:** `a.media.progress75` • **Data Feed:** `videoprogress75` • **Audience Manager:** `c_contextdata.a.media.progress75`
	Playhead passes the 75% marker of video based on video length. Marker only counted once, even if seeking backwards. If seeking forward, markers that are skipped are not counted. ***Important:*** *This can only be true if it is set. If it is not set, no value is returned.*		
95% Progress Marker	• **Key:** Automatically set • **Type:** string • **Sent with:** Close • **Min. SDK Version:** Any	• **Adobe Analytics:** N/A • **Heartbeats:** N/A	• **Available:** Yes • **Reserved Variable:** event • **Report Name:** 95% Progress Marker

Label	Implementation	Network Parameters	Reporting
	• **Sample value:** `TRUE`		• **Context Data:** `a.media.progress95` • **Data Feed:** `videoprogress95` • **Audience Manager:** `c_contextdata.a.media.progress95`
	Playhead passes the 95% marker of video based on video length. Marker only counted once, even if seeking backwards. If seeking forward, markers that are skipped are not counted. ***Important:*** *This can only be true if it is set. If it is not set, no value is returned.*		
Average Minute Audience	• **Key:** Automatically set • **Type:** number • **Sent with:** Close • **Min. SDK Version:** Any • **Sample value:** Greater than or equal to 1	• **Adobe Analytics:** N/A • **Heartbeats:** N/A	• **Available:** Yes • **Reserved Variable:** event • **Report Name:** Average Minute Audience • **Context Data:** `a.media.averageMinuteAudience` • **Data Feed:** `videoaverageminuteaudience` • **Audience Manager:** `c_contextdata.a.media.averageMinuteAudience`
	Average Minute Audience metric is computed as Total Content Time Spent, for one specific video, divided by that Video's Length for all of its playback sessions: `average_minute_audience = timeSpent / videoLength;`		
Estimated Streams	• **Key:** Automatically set • **Type:** number • **Sent with:** Close • **Min. SDK Version:** Any • **Sample value:** • 1 - For a 19 minutes playback • 2 - For a 31 minutes playback • 3 - For a 78 minutes playback	• **Adobe Analytics:** N/A • **Heartbeats:** N/A	• **Available:** Use custom processing rule • **Reserved Variable:** N/A • **Report Name:** Custom • **Context Data:** `a.media.estimatedStreams` • **Data Feed:** `N/A` • **Audience Manager:** `c_contextdata.a.media.estimatedStreams`
	The estimated number of video streams per each individual content. This value is increased for each 30 minutes of video play time (content + ads). Customers must create their own processing rules to have the value available for reporting. A stream is counted at every 30 minutes, based on the `ms_s` (or `totalTimePlayed = Video Total Time`), similar to: `estimatedStreams = FLOOR(ms_s/1800) +1`		

Label	Implementation	Network Parameters	Reporting
Paused Impacted Streams	• **Key:** Automatically set • **Type:** string • **Sent with:** Close • **Min. SDK Version:** 1.5.6 • **Sample value:** `TRUE`	• **Adobe Analytics:** N/A • **Heartbeats:** `s:event:type=pause`	• **Available:** Yes • **Reserved Variable:** event • **Report Name:** Paused Impacted Stream • **Context Data:** `a.media.pause` • **Data Feed:** `videopause` • **Audience Manager:** `c_contextdata.a.media.pause`
	This value is either true or false. It is true if one or more pauses occurred during playback of a single video. ***Important:*** *This can only be true if it is set. If it is not set, no value is returned.*		
Pause Events	• **Key:** Automatically set • **Type:** number • **Sent with:** Close • **Min. SDK Version:** 1.5.6 • **Sample value:** `2` (Integer)	• **Adobe Analytics:** N/A • **Heartbeats:** `s:event:type=pause`	• **Available:** Yes • **Reserved Variable:** event • **Report Name:** Pause Events • **Context Data:** `a.media.pauseCount` • **Data Feed:** `videopausecount` • **Audience Manager:** `c_contextdata.a.media.pauseCount`
	This metric is computed as a count of pause periods that occurred during a playback session.		
Total Pause Duration	• **Key:** Automatically set • **Type:** number • **Sent with:** Close • **Min. SDK Version:** 1.5.6 • **Sample value:** `190`	• **Adobe Analytics:** N/A • **Heartbeats:** N/A	• **Available:** Yes • **Reserved Variable:** event • **Report Name:** Total Pause Duration • **Context Data:** `a.media.pauseTime` • **Data Feed:** `videopausetime` • **Audience Manager:** `c_contextdata.a.media.pauseTime`
	Sums the duration (in seconds) of all events of type PAUSE.		
Content Resumes	• **Key:** Automatically set • **Type:** string • **Sent with:** Close • **Min. SDK Version:** 1.5.6 • **Sample value:** `TRUE`	• **Adobe Analytics:** N/A • **Heartbeats:** `s:event:type=resume`	• **Available:** Yes • **Reserved Variable:** event • **Report Name:** Content Resumes • **Context Data:** `a.media.resume` • **Data Feed:** `videoresume` • **Audience Manager:** `c_contextdata.a.media.resume`

Label	Implementation	Network Parameters	Reporting
	A Resume is counted for each playback that resumes after more than 30 minutes of buffer, pause, or stall period OR if this value is set by the player on the `VideoInfo` object before `trackPlay`. ⚠ ***Important:*** *This can only be true if it is set. If it is not set, no value is returned.*		
Content Segment Views	• **Key:** Automatically set • **Type:** string • **Sent with:** Close • **Min. SDK Version:** Any • **Sample value:** `TRUE`	• **Adobe Analytics:** N/A • **Heartbeats:** N/A	• **Available:** Yes • **Reserved Variable:** event • **Report Name:** Content Segment Views • **Context Data:** `a.media.segmentView` • **Data Feed:** `videosegmentviews` • **Audience Manager:** `c_contextdata.a.media.segmentView`
	The number of views of the main content. A Content Segment View is counted when there is at least one frame viewed. ⚠ ***Important:*** *This can only be true if it is set. If it is not set, no value is returned.*		

Ad Parameters

List of video ad data, including context data values, that Adobe collects via solution variables.

Table 13: Ad Video Data

Label	Implementation	Network Parameters	Reporting
Ad Name	• **Key:** ***`name*`*** • **Required:** No • **Type:** string • **Sent with:** Ad Start, Ad Close • **Min. SDK Version:** 1.5.1 • **Sample value:** `"Ford F-150"`	• **Adobe Analytics:** `a.media.ad.friendlyName` • **Heartbeat:** `s:asset:ad_name`	• **Available:** Yes • **Reserved Variable:** eVar and classification • **Expiration:** On HIT • **Report Name:** *Ad Name and Ad Name (variable)* • **Context Data:** `a.media.ad.friendlyName` • **Data Feed:** `N/A` • **Audience Manager:** `c_contextdata.a.media.ad.friendlyName`
	Friendly name of the ad. In reporting, "Ad Name" is the classification and "Ad Name (variable)" is the eVar.		

Label	Implementation	Network Parameters	Reporting
	* `createAdObject(`***`name`***`, adId, position, length)`		

Ad ID	• **Key:** ***`adId*`***
	• **Required:** Yes
	• **Type:** string
	• **Sent with:** Ad Start, Ad Close
	• **Min. SDK Version:** Any
	• **Sample value:** `"2125"`
		• **Adobe Analytics:**
		`a.media.ad.name`
		• **Heartbeat:**
		`s:asset:ad_id`
			• **Available:** Yes
			• **Reserved Variable:** eVar
			• **Expiration:** On VISIT
			• **Report Name:** *Ad*
			• **Context Data:** `a.media.ad.name`
			• **Data Feed:** `videoad`
			• **Audience Manager:** `c_contextdata.a.media.ad.name`

	ID of the ad. (Any integer and/or letter combination) * `createAdObject(name, `***`adId`***`, position, length)`		

Ad In Pod Position	• **Key:** ***`position*`***
	• **Required:** Yes
	• **Type:** number
	• **Sent with:** Ad Start, Ad Close
	• **Min. SDK Version:** Any
	• **Sample value:** `1`
		• **Adobe Analytics:**
		`a.media.ad.podPosition`
		• **Heartbeat:**
		`s:asset:pod_position`
			• **Available:** Yes
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:** *Ad In Pod Position*
			• **Context Data:** `a.media.ad.podPosition`
			• **Data Feed:** `videoadinpod`
			• **Audience Manager:** `c_contextdata.a.media.ad.podPosition`

	The position (index) of the ad inside the parent ad break. The first ad has index 0, the second ad has index 1, etc. * `createAdObject(name, adId, `***`position`***`, length)`		

Ad Length	• **Key:** ***`length*`***
	• **Required:** Yes
	• **Type:** number
	• **Sent with:** Ad Start, Ad Close
	• **Min. SDK Version:** 1.5.1
	• **Sample value:** `"15"`
		• **Adobe Analytics:**
		`a.media.ad.length`
		• **Heartbeat:**
		`l:asset:ad_length`
			• **Available:** Yes
			• **Reserved Variable:** eVar and classification
			• **Expiration:** On HIT
			• **Report Name:** *Ad Length and Ad Length (variable)*
			• **Context Data:** `a.media.ad.length`
			• **Data Feed:** `videoadlength`
			• **Audience Manager:** `c_contextdata.a.media.ad.length`

Label	Implementation	Network Parameters	Reporting
	Length of video ad in seconds. * *createAdObject*(name, adId, position, ***length***)		

Ad Player Name	• **Key:** ***playerName****
	• **Required:** Yes
	• **Type:** string
	• **Sent with:** Ad Start, Ad Close
	• **Min. SDK Version:** Any
	• **Sample value:** `"Freewheel"`, etc.
		• **Adobe Analytics:**
		`a.media.ad.playerName`
		• **Heartbeat:**
		`s:sp:player_name`
			• **Available:** Yes
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:** *Ad Player Name*
			• **Context Data:**
			`a.media.ad.playerName`
			• **Data Feed:** `videoadplayername`
			• **Audience Manager:**
			`c_contextdata.a.media.ad.playerName`

	The name of the player responsible for rendering the ad. * *MediaHeartbeatConfig*.***playerName***		

Ad Break Name	• **Key:** ***name****
	• **Required:** Yes
	• **Type:** string
	• **Sent with:** Ad Start, Ad Close
	• **Min. SDK Version:** Any
	• **Sample value:** `"pre-roll"`
		• **Adobe Analytics:**
		`a.media.ad.podFriendlyName`
		• **Heartbeat:**
		`s:asset:pod_name`
			• **Available:** Yes
			• **Reserved Variable:** Classification
			• **Report Name:** *Pod Name*
			• **Context Data:**
			`a.media.ad.podFriendlyName`
			• **Data Feed:** `videoadpod`
			• **Audience Manager:**
			`c_contextdata.a.media.ad.podFriendlyName`

	The friendly name of the Ad Break. * *createAdBreakObject*(***name***, position, startTime)		
Ad Break Index	• **Key:** ***position**** • **Required:** Yes • **Type:** number • **Sent with:** • **Min. SDK Version:** Any • **Sample value:** `1`	• **Adobe Analytics:** • **Heartbeat:**	• **Available:** No • **Reserved Variable:** N/A • **Report Name:** *N/A* • **Context Data:** • **Data Feed:** • **Audience Manager:**
	The index of the ad break inside the content starting at 1. This property is used ***only*** by the VHL SDK to generate the Pod ID. * *createAdBreakObject*(name, ***position***, startTime)		

Label	Implementation	Network Parameters	Reporting
Ad Break Position	• **Key:** ***`startTime*`*** • **Required:** Yes • **Type:** number • **Sent with:** Ad Start, Ad Close • **Min. SDK Version:** Any • **Sample value:** `90`	• **Adobe Analytics:** `a.media.ad.podSecond` • **Heartbeat:** `l:asset:pod_offset`	• **Available:** Yes • **Reserved Variable:** Classification • **Report Name:** *Pod Position* • **Context Data:** `a.media.ad.podSecond` • **Data Feed:** • **Audience Manager:** `c_contextdata.a.media.ad.podSecond`
	The offset of the ad break inside the content, in seconds. ***** *`createAdBreakObject`*`(name, position, `***`startTime`***`)`		
Ad Break ID	• **Key:** *`Automatically set`* • **Required:** Yes • **Type:** string • **Sent with:** Ad Start, Ad Close • **Min. SDK Version:** Any • **Sample value:** `c4a577424c84067899b807c76722d495_1`	• **Adobe Analytics:** `a.media.ad.pod` • **Heartbeat:** `l:asset:pod_id`	• **Available:** Yes • **Reserved Variable:** eVar • **Expiration:** On HIT • **Report Name:** *Ad Pod* • **Context Data:** `a.media.ad.pod` • **Data Feed:** `videoadpod` • **Audience Manager:**

Table 14: Standard Ad Metadata

Label	Implementation	Network Parameters	Reporting
Advertiser	• **Key:** `ADVERTISER` • **Required:** No • **Type:** string • **Sent with:** Ad Start, Ad Close • **Min. SDK Version:** 1.5.7 • **Sample value:**	• **Adobe Analytics:** `a.media.ad.advertiser` • **Heartbeat:** `s:meta:a.media.ad.advertiser`	• **Available:** Yes • **Reserved Variable:** eVar • **Expiration:** On HIT • **Report Name:** *Advertiser* • **Context Data:** `a.media.ad.advertiser` • **Data Feed:** `videoadvertiser` • **Audience Manager:** `c_contextdata.a.media.ad.length`
	Company/Brand whose product is featured in the ad. *`MediaHeartbeat.AdMetadataKeys`*		

Label	Implementation	Network Parameters	Reporting

Campaign ID	• **Key:** `CAMPAIGN_ID`
	• **Required:** No
	• **Type:** string
	• **Sent with:** Ad Start, Ad Close
	• **Min. SDK Version:** 1.5.7
	• **Sample value:** Integer, or name (string).
		• **Adobe Analytics:**
		`a.media.ad.campaign`
		• **Heartbeat:**
		`s:meta:a.media.ad.campaign`
			• **Available:** Yes
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:** *Campaign ID*
			• **Context Data:** `a.media.ad.campaign`
			• **Data Feed:** `videocampaign`
			• **Audience Manager:** `c_contextdata.a.media.ad.campaign`

	ID of the ad campaign. *`MediaHeartbeat.AdMetadataKeys`*		

Creative ID	• **Key:** `CREATIVE_ID`
	• **Required:** No
	• **Type:** string
	• **Sent with:** Ad Start, Ad Close
	• **Min. SDK Version:** 1.5.7
	• **Sample value:** Integer, or name (string).
		• **Adobe Analytics:**
		`a.media.ad.creative`
		• **Heartbeat:**
		`s:meta:a.media.ad.creative`
			• **Available:** Yes
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:** *Creative ID*
			• **Context Data:** `a.media.ad.creative`
			• **Data Feed:** `adclassificationcreative`
			• **Audience Manager:** `c_contextdata.a.media.ad.creative`

	ID of the ad creative. *`MediaHeartbeat.AdMetadataKeys`*		

Site ID	• **Key:** `SITE_ID`
	• **Required:** No
	• **Type:** string
	• **Sent with:** Ad Start, Ad Close
	• **Min. SDK Version:** 1.5.7
	• **Sample value:**
		• **Adobe Analytics:**
		`a.media.ad.site`
		• **Heartbeat:**
		`s:meta:a.media.ad.site`
			• **Available:** *Use custom processing rule*
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:**
			• **Context Data:** `a.media.ad.site`
			• **Data Feed:** `N/A`
			• **Audience Manager:** `c_contextdata.a.media.ad.site`

	ID of the ad site. *`MediaHeartbeat.AdMetadataKeys`*		
Creative URL	• **Key:** `CREATIVE_URL` • **Required:** No	• **Adobe Analytics:** `a.media.ad.creativeURL`	• **Available:** *Use custom processing rule* • **Reserved Variable:** eVar

Label	Implementation	Network Parameters	Reporting
	• **Type:** string • **Sent with:** Ad Start, Ad Close • **Min. SDK Version:** 1.5.7 • **Sample value:**	• **Heartbeat:** `s:meta:a.media.ad.creativeURL`	• **Expiration:** On HIT • **Report Name:** • **Context Data:** `a.media.ad.creativeURL` • **Data Feed:** `N/A` • **Audience Manager:** `c_contextdata.a.media.ad.creativeURL`
	URL of the ad creative. *`MediaHeartbeat.AdMetadataKeys`*		

Placement ID	• **Key:** `PLACEMENT_ID`
	• **Required:** No
	• **Type:** string
	• **Sent with:** Ad Start, Ad Close
	• **Min. SDK Version:** 1.5.7
	• **Sample value:**
		• **Adobe Analytics:**
		`a.media.ad.placement`
		• **Heartbeat:**
		`s:meta:a.media.ad.placement`
			• **Available:** *Use custom processing rule*
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:**
			• **Context Data:** `a.media.ad.placement`
			• **Data Feed:** `N/A`
			• **Audience Manager:** `c_contextdata.a.media.ad.placement`

	Placement ID of the ad. *`MediaHeartbeat.AdMetadataKeys`*		

Table 15: Ad Metrics

Label	Implementation	Network Parameters	Reporting
Ad Start	• **Key:** *Automatically set* • **Required:** Yes • **Type:** string • **Sent with:** Ad Start • **Min. SDK Version:** Any • **Sample value:** `TRUE`	• **Adobe Analytics:** `a.media.ad.view` • **Heartbeat:** • `s:event:type=start` • `s:asset:type=ad`	• **Available:** Yes • **Reserved Variable:** event • **Report Name:** *Ad Starts* • **Data Feed:** `videoadstart` • **Context Data:** `a.media.ad.view` • **Audience Manager:** `c_contextdata.a.media.ad.view`
	Number of video ad starts.		
Ad Complete	• **Key:** *Automatically set* • **Required:** Yes	• **Adobe Analytics:** `a.media.ad.complete`	• **Available:** Yes • **Reserved Variable:** event

Label	Implementation	Network Parameters	Reporting
	• **Type:** string • **Sent with:** Ad Close • **Min. SDK Version:** Any • **Sample value:** `TRUE`	• **Heartbeat:** • `s:event:type=complete` • `s:asset:type=ad`	• **Report Name:** *Ad Completes* • **Data Feed:** `videoadcomplete` • **Context Data:** `a.media.ad.complete` • **Audience Manager:** `c_contextdata.a.media.ad.complete`
	Number of video ad completes.		
Ad Time Spent	• **Key:** *Automatically set* • **Required:** Yes • **Type:** string • **Sent with:** Ad Close • **Min. SDK Version:** Any • **Sample value:** `15`	• **Adobe Analytics:** `a.media.ad.timePlayed` • **Heartbeat:**	• **Available:** Yes • **Reserved Variable:** event • **Report Name:** *Ad Time Spent* • **Data Feed:** `videoadtime` • **Context Data:** `a.media.ad.timePlayed` • **Audience Manager:** `c_contextdata.a.media.ad.timePlayed`
	The total amount of time, in seconds, spent watching the ad (i.e., the number of seconds played).		

Chapter Parameters

List of chapter and/or segment data, including context data values, that Adobe collects via solution variables.

Table 16: Chapter Metadata

Label	Implementation	Network Parameters	Reporting
Chapter Name	• **Key:** ***name**** • **Required:** No • **Type:** string • **Sent with:** Chapter Start, Close • **Min. SDK Version:** 1.3 • **Sample value:** `"The Big Bang Chapter 2 – Dating"`	• **Adobe Analytics:** `a.media.chapter.friendlyName` • **Heartbeat:** `s:stream:chapter_name`	• **Available:** *Created by default...* • **Reserved Variable:** Classification • **Report Name:** *Chapter Name* • **Context Data:** `a.media.ad.friendlyName` • **Data Feed:** `N/A` • **Audience Manager:** `c_contextdata.a.media.chapter.friendlyName`
	The name of the chapter and/or segment. ***** `createChapterObject(name, position, length, startTime)`		

Label	Implementation	Network Parameters	Reporting

Chapter Position	• **Key:** ***`position*`***
	• **Required:** No
	• **Type:** number
	• **Sent with:** Close
	• **Min. SDK Version:** 1.3
	• **Sample value:** `2`
		• **Adobe Analytics:**
		`a.media.chapter.position`
		• **Heartbeat:**
		`l:stream:chapter_pos`
			• **Available:** Yes
			• **Reserved Variable:** Classification
			• **Report Name:** *Chapter Position*
			• **Context Data:** `a.media.ad.position`
			• **Data Feed:**
			• **Audience Manager:** `c_contextdata.a.media.chapter.position`

	The position (index, integer) of the chapter inside the content. ***** *`createChapterObject`*`(name,` ***`position`***`, length, startTime)`		

Chapter Offset	• **Key:** ***`startTime*`***
	• **Required:** No
	• **Type:** number
	• **Sent with:** Close
	• **Min. SDK Version:** 1.3
	• **Sample value:** `58`
		• **Adobe Analytics:**
		`a.media.chapter.offset`
		• **Heartbeat:**
		`l:stream:chapter_offset`
			• **Available:** Yes
			• **Reserved Variable:** Classification
			• **Report Name:** *Chapter Offset*
			• **Context Data:** `a.media.chapter.offset`
			• **Data Feed:**
			• **Audience Manager:** `c_contextdata.a.media.chapter.offset`

	The offset of the chapter inside the content (in seconds) from the start. ***** *`createChapterObject`*`(name, position, length,` ***`startTime`***`)`		

Chapter Length	• **Key:**
	• **Required:** No
	• **Type:** number
	• **Sent with:** Close
	• **Min. SDK Version:** 1.3
	• **Sample value:** `486`
		• **Adobe Analytics:**
		`a.media.chapter.length`
		• **Heartbeat:**
		`l:stream:chapter_length`
			• **Available:** Yes
			• **Reserved Variable:** Classification
			• **Report Name:** *Chapter Length*
			• **Context Data:** `a.media.chapter.length`
			• **Data Feed:**
			• **Audience Manager:** `c_contextdata.a.media.chapter.length`

	The length of the chapter, in seconds. ***** *`createChapterObject`*`(name, position,` ***`length`***`, startTime)`		

Chapter	• **Key:** *Automatically set*
	• **Required:** No
	• **Type:** string
	• **Sent with:** Close
		• **Adobe Analytics:**
		`a.media.chapter.name`
		• **Heartbeat:**
		`s:stream:chapter_id`
			• **Available:** Yes
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:** *Chapter*

Label	Implementation	Network Parameters	Reporting
	• **Min. SDK Version:** 1.3 • **Sample value:**		• **Context Data:** `a.media.chapter.name` • **Data Feed:** `videochapter` • **Audience Manager:** `c_contextdata.a.media.chapter.name`
	The auto-generated ID of the chapter.		

Table 17: Chapter Metrics

Label	Implementation	Network Parameters	Reporting
Chapter Start	• **Key:** *Automatically set* • **Required:** Yes • **Type:** string • **Sent with:** Chapter Start • **Min. SDK Version:** 1.3 • **Sample value:** `TRUE`	• **Adobe Analytics:** `a.media.chapter.view` • **Heartbeat:** `s:event:type=chapter_start`	• **Available:** Yes • **Reserved Variable:** event • **Report Name:** *Chapter Starts* • **Context Data:** `a.media.chapter.view` • **Data Feed:** `videochapterstart` • **Audience Manager:** `c_contextdata.a.media.chapter.view`
	The number of chapter starts. ***Important:*** *If this event is set, the only possible value is TRUE. If this event is not set, no value is sent.*		
Chapter Complete	• **Key:** *Automatically set* • **Required:** Yes • **Type:** string • **Sent with:** Close • **Min. SDK Version:** 1.3 • **Sample value:** `TRUE`	• **Adobe Analytics:** `a.media.chapter.complete` • **Heartbeat:** `s:event:type=chapter_complete`	• **Available:** Yes • **Reserved Variable:** event • **Report Name:** *Chapter Completes* • **Context Data:** `a.media.chapter.complete` • **Data Feed:** `videochaptercomplete` • **Audience Manager:** `c_contextdata.a.media.chapter.complete`
	The number of chapter completes. ***Important:*** *If this event is set, the only possible value is TRUE. If this event is not set, no value is sent.*		

Label	Implementation	Network Parameters	Reporting
Chapter Time Spent	• **Key:** *Automatically set* • **Required:** Yes • **Type:** number • **Sent with:** Close • **Min. SDK Version:** 1.3 • **Sample value:**	• **Adobe Analytics:** `a.media.chapter.timePlayed` • **Heartbeat:**	• **Available:** Yes • **Reserved Variable:** event • **Report Name:** *Chapter Time Spent* • **Context Data:** `a.media.chapter.timePlayed` • **Data Feed:** `videochaptertime` • **Audience Manager:** `c_contextdata.a.media.chapter.timePlayed`
	The time spent on the chapter.		

Quality Parameters

List of quality of experience (QoE/Qos) data, including context data values, that Adobe collects via solution variables.

Table 18: Quality Metadata

Label	Implementation	Network Parameters	Reporting

Average Bitrate	• **Key:** ***`bitrate*`***
	• **Required:** No
	• **Type:** number
	• **Sent with:** Close
	• **Min. SDK Version:** Any
	• **Sample value:** `800-899`
		• **Adobe Analytics:**
		`a.media.qoe.bitrateAverageBucket`
		• **Heartbeat:**
		`l:stream:bitrate`
			• **Available:** Yes
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:** *Average Bitrate*
			• **Context Data:** `a.media.qoe.bitrateAverageBucket`
			• **Data Feed:** `videoqoebitrateaverageevar`
			• **Audience Manager:** `c_contextdata.a.media.qoe.bitrateAverageBucket`

	The average bitrate (in kbps). The value is predefined buckets at 100kbps intervals. The Average Bitrate is computed as a weighted average of all bitrate values related to the play duration that occurred during a playback session ***** `createQoSObject(`***`bitrate`***`, startupTime, fps, droppedFrames)`		

Time to Start	• **Key:** ***`startupTime*`***
	• **Required:** No
	• **Type:** number
	• **Sent with:** Initiate, Close
	• **Min. SDK Version:** Any
	• **Sample value:** `5`
		• **Adobe Analytics:**
		`a.media.qoe.timeToStart`
		• **Heartbeat:**
		`l:stream:startup_time`
			• **Available:** Yes
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:** *Time to Start*
			• **Context Data:** `a.media.qoe.timeToStart`
			• **Data Feed:** `videoqoetimetostartevar`

Label	Implementation	Network Parameters	Reporting
			• **Audience Manager:** `c_contextdata.a.media.qoe.timeToStart`
	The amount of time (in seconds, integer) taken for the video to start. This metric is computed as the time difference between the time the viewer clicks Play and when the first frame was rendered. This value is taken from the last value of `l:stream:startup_time` and transformed into seconds. ***** `createQoSObject(bitrate, startupTime, fps, droppedFrames)`		
FPS	• **Key:** ***`fps*`*** • **Required:** No • **Type:** number • **Sent with:** Initiate, Close • **Min. SDK Version:** Any • **Sample value:** `24`	• **Adobe Analytics:** • **Heartbeat:** `l:stream:fps`	• **Available:** No • **Reserved Variable:** N/A • **Report Name:** *N/A* • **Context Data:** • **Data Feed:** • **Audience Manager:**
	The current value of the stream frame-rate (in frames per second). ***** `createQoSObject(bitrate, startupTime, fps, droppedFrames)`		

Dropped Frames	• **Key:** ***`droppedFrames`***
	• **Required:** No
	• **Type:** number
	• **Sent with:** Close
	• **Min. SDK Version:** Any
	• **Sample value:** `3`
		• **Adobe Analytics:**
		`a.media.qoe.droppedFrameCount`
		• **Heartbeat:**
		`l:stream:dropped_frames`
			• **Available:** Yes
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:** *Dropped Frames*
			• **Context Data:**
			`a.media.qoe.droppedFrameCount`
			• **Data Feed:**
			`videoqoedroppedframecountevar`
			• **Audience Manager:**
			`c_contextdata.a.media.qoe.droppedFrameCount`

	The number of dropped frames (Integer). This value is computed as a sum of all frames dropped during a playback session. This value is taken from the last value of `l:stream:dropped_frames`. ***** `createQoSObject(bitrate, startupTime, fps, droppedFrames)`		

Buffer Events	• **Key:** *`Automatically set`*
	• **Required:** No
	• **Type:** number
	• **Sent with:** Close
	• **Min. SDK Version:** Any
	• **Sample value:** `2`
		• **Adobe Analytics:**
		`a.media.qoe.bufferCount`
		• **Heartbeat:**
		`s:event:type=buffer`
			• **Available:** Yes
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:** *Buffer Events*
			• **Context Data:**
			`a.media.qoe.bufferCount`
			• **Data Feed:**
			`videoqoebuffercountevar`

Label	Implementation	Network Parameters	Reporting
			• **Audience Manager:** `c_contextdata.a.media.qoe.bufferCount`
	The number of buffer events. This metric is computed as a count of the different buffer states that occurred during a playback session. This is a count of how many times the player enters a buffer state from other states, e.g., playing or pausing.		

Total Buffer Duration	• **Key:** *Automatically set*
	• **Required:** No
	• **Type:** number
	• **Sent with:** Close
	• **Min. SDK Version:**
	• **Sample value:** `30` (Seconds)
		• **Adobe Analytics:**
		`a.media.qoe.bufferTime`
		• **Heartbeat:**
		`l:event:duration`
			• **Available:** Yes
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:** *Total Buffer Duration*
			• **Context Data:** `a.media.qoe.bufferTime`
			• **Data Feed:** `videoqoebuffertimeevar`
			• **Audience Manager:** `c_contextdata.a.media.qoe.bufferTime`

	The total amount of time, in seconds, spent buffering. This value is computed as a sum of all buffer events durations that occurred during a playback session.		

Bitrate Changes	• **Key:**
	• **Required:** No
	• **Type:** number
	• **Sent with:** Close
	• **Min. SDK Version:** Any
	• **Sample value:** `3`
		• **Adobe Analytics:**
		`a.media.qoe.bitrateChangeCount`
		• **Heartbeat:**
		`s:event:type=bitrate_change`
			• **Available:** Yes
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:** *Bitrate Changes*
			• **Context Data:** `a.media.qoe.bitrateChangeCount`
			• **Data Feed:** `videoqoebitratechangecountevar`
			• **Audience Manager:** `c_contextdata.a.media.qoe.bitrateChangeCount`

	The number of bitrate changes (Integer). This value is computed as a sum of all bitrate change events that occurred during a playback session.		

Errors / Error Events	• **Key:**
	• **Required:** No
	• **Type:** number
	• **Sent with:** Close
	• **Min. SDK Version:** Any
	• **Sample value:** `1`
		• **Adobe Analytics:**
		`a.media.qoe.errorCount`
		• **Heartbeat:**
		`s:event:type=error`
			• **Available:** Yes
			• **Reserved Variable:** eVar
			• **Expiration:** On HIT
			• **Report Name:** *Errors*
			• **Context Data:** `a.media.qoe.errorCount`
			• **Data Feed:** `videoqoeerrorcountevar`

Label	Implementation	Network Parameters	Reporting
			• **Audience Manager:** `c_contextdata.a.media.qoe.errorCount`
	The number of errors occurred (Integer). This value is computed as a sum of all error events that occurred during a playback session.		

Table 19: Quality Metrics

Label	Implementation	Network Parameters	Reporting

Time To Start	• **Key:** *Automatically set*
	• **Required:** No
	• **Type:** number
	• **Sent with:** Close
	• **Min. SDK Version:** Any
	• **Sample value:** `1`
		• **Adobe Analytics:**
		`a.media.qoe.timeToStart`
		• **Heartbeat:**
		`l:stream:startup_time`
			• **Available:** Yes
			• **Reserved Variable:** event
			• **Report Name:** *Time to Start*
			• **Context Data:** `a.media.qoe.timeToStart`
			• **Data Feed:** `videoqoetimetostart`
			• **Audience Manager:** `c_contextdata.a.media.qoe.timeToStart`

	The amount of time (in seconds) taken for the video to start. This metric is computed as the time difference between the time the viewer clicks Play and when the first frame was rendered. We get the last value of `l:stream:startup_time` and transform it in seconds.		

Buffer Events	• **Key:** *Automatically set*
	• **Required:** No
	• **Type:** number
	• **Sent with:** Close
	• **Min. SDK Version:** Any
	• **Sample value:** `2`
		• **Adobe Analytics:**
		`a.media.qoe.bufferCount`
		• **Heartbeat:**
		`s:event:type=buffer`
			• **Available:** Yes
			• **Reserved Variable:** event
			• **Report Name:** *Buffer Events*
			• **Context Data:** `a.media.qoe.bufferCount`
			• **Data Feed:** `videoqoebuffercount`
			• **Audience Manager:** `c_contextdata.a.media.qoe.bufferCount`

	The number of buffer events (Integer). This metric is computed as a count of buffer events that occurred during a playback session.		

Total Buffer Duration	• **Key:** *Automatically set*
	• **Required:** No
	• **Type:** number
	• **Sent with:** Close
	• **Min. SDK Version:** Any
	• **Sample value:** `15`
		• **Adobe Analytics:**
		`a.media.qoe.bufferTime`
		• **Heartbeat:**
		`l:event:duration`
			• **Available:** Yes
			• **Reserved Variable:** event
			• **Report Name:** *Total Buffer Duration*
			• **Context Data:** `a.media.qoe.bufferTime`
			• **Data Feed:** `videoqoebuffertime`

Label	Implementation	Network Parameters	Reporting
			• **Audience Manager:** `c_contextdata.a.media.qoe.bufferTime`
	The total amount of time spent buffering (seconds; integer). This value is computed as a sum of all buffer events durations that occurred during a playback session.		

Bitrate Changes	• **Key:** *Automatically set*
	• **Required:** No
	• **Type:** Event
	• **Sent with:** Close
	• **Min. SDK Version:** Any
	• **Sample value:** `"3"` (Integer)
		• **Adobe Analytics:**
		`a.media.qoe.bitrateChangeCount`
		• **Heartbeat:**
		`s:event:type=bitrate_change`
			• **Available:** Yes
			• **Reserved Variable:** event
			• **Report Name:** *Bitrate Changes*
			• **Context Data:** `a.media.qoe.bitrateChangeCount`
			• **Data Feed:** `videoqoebitratechangecount`
			• **Audience Manager:** `c_contextdata.a.media.qoe.bitrateChangeCount`

	The number of bitrate changes. This value is computed as a sum of all bitrate change events that occurred during a playback session.		

Errors	• **Key:** *Automatically set*
	• **Required:** No
	• **Type:** number
	• **Sent with:** Close
	• **Min. SDK Version:** Any
	• **Sample value:** `1`
		• **Adobe Analytics:**
		`a.media.qoe.errorCount`
		• **Heartbeat:**
		`s:event:type=error`
			• **Available:** Yes
			• **Reserved Variable:** event
			• **Report Name:** *Error Events*
			• **Context Data:** `a.media.qoe.errorCount`
			• **Data Feed:** `videoqoeerrorcount`
			• **Audience Manager:** `c_contextdata.a.media.qoe.errorCount`

	The number of errors occurred (Integer). This value is computed as a sum of all error events that occurred during a playback session.		

Dropped Frames	• **Key:** *Automatically set*
	• **Required:** No
	• **Type:** number
	• **Sent with:** Close
	• **Min. SDK Version:** Any
	• **Sample value:** `1`
		• **Adobe Analytics:**
		`a.media.qoe.droppedFrameCount`
		• **Heartbeat:**
		`l:stream:dropped_frames`
			• **Available:** Yes
			• **Reserved Variable:** event
			• **Report Name:** *Dropped Frames*
			• **Context Data:** `a.media.qoe.droppedFrameCount`
			• **Data Feed:** `videoqoedroppedframecount`
			• **Audience Manager:** `c_contextdata.a.media.qoe.droppedFrameCount`

	The number of dropped frames (Integer). This value is computed as a sum of all frames dropped during a playback session.		

Label	Implementation	Network Parameters	Reporting

Drops Before Start	• **Key:** *Automatically set*
	• **Required:** No
	• **Type:** string
	• **Sent with:** Close
	• **Min. SDK Version:** Any
	• **Sample value:** `TRUE`
		• **Adobe Analytics:**
		`a.media.qoe.dropBeforeStart`
		• **Heartbeat:**
		`s:event:type=aa_start`
			• **Available:** Yes
			• **Reserved Variable:** event
			• **Report Name:** *Drops before Start*
			• **Context Data:** `a.media.qoe.dropBeforeStart`
			• **Data Feed:** `videoqoedropbeforestart`
			• **Audience Manager:** `c_contextdata.a.media.qoe.dropBeforeStart`

	The number of times a user quit the video before its start. This metric is set to 1 only if no content was rendered, regardless of ads. ***Important:*** *If this event is set, the only possible value is TRUE. If this event is not set, no value is sent.*		

Buffer Impacted Streams	• **Key:** *Automatically set*
	• **Required:** No
	• **Type:** string
	• **Sent with:** Close
	• **Min. SDK Version:** Any
	• **Sample value:** `TRUE`
		• **Adobe Analytics:**
		`a.media.qoe.buffer`
		• **Heartbeat:**
		`s:event:type=buffer`
			• **Available:** Yes
			• **Reserved Variable:** event
			• **Report Name:** *Buffer Impacted Streams*
			• **Context Data:** `a.media.qoe.buffer`
			• **Data Feed:** `videoqoebuffer`
			• **Audience Manager:** `c_contextdata.a.media.qoe.buffer`

	The number of streams impacted by buffering. This metric is set to 1 only if at least one buffer event occurred during a playback session. ***Important:*** *If this event is set, the only possible value is TRUE. If this event is not set, no value is sent.*		

Bitrate Change Impacted Streams	• **Key:** *Automatically set*
	• **Required:** No
	• **Type:** string
	• **Sent with:** Close
	• **Min. SDK Version:** Any
	• **Sample value:** `TRUE`
		• **Adobe Analytics:**
		`a.media.qoe.bitrateChange`
		• **Heartbeat:**
		`s:event:type=bitrate_change`
			• **Available:** Yes
			• **Reserved Variable:** event
			• **Report Name:** *Buffer Change Impacted Streams*
			• **Context Data:** `a.media.qoe.bitrateChange`
			• **Data Feed:** `videoqoebitratechange`
			• **Audience Manager:** `c_contextdata.a.media.qoe.bitrateChange`

Label	Implementation	Network Parameters	Reporting
	The number of streams in which bitrate changes occurred. This metric is set to 1 only if at least one bitrate change event occurred during a playback session. ***Important:*** *If this event is set, the only possible value is TRUE. If this event is not set, no value is sent.*		

Average Bitrate	• **Key:** *Automatically set*
	• **Required:** No
	• **Type:** number
	• **Sent with:** Close
	• **Min. SDK Version:** Any
	• **Sample value:** `3200`
		• **Adobe Analytics:**
		`a.media.qoe.bitrateAverage`
		• **Heartbeat:**
		`l:stream:bitrate`
			• **Available:** Yes
			• **Reserved Variable:** event
			• **Report Name:** *Average Bitrate*
			• **Context Data:** `a.media.qoe.bitrateAverage`
			• **Data Feed:** `videoqoebitrateaverage`
			• **Audience Manager:** `c_contextdata.a.media.qoe.bitrateAverage`

	The average bitrate (in kbps, integer). This metric is computed as a weighted average of all bitrate values related to the play duration that occurred during a playback session.		

Error Impacted Streams	• **Key:** *Automatically set*
	• **Required:** No
	• **Type:** string
	• **Sent with:** Close
	• **Min. SDK Version:** Any
	• **Sample value:** `TRUE`
		• **Adobe Analytics:**
		`a.media.qoe.error`
		• **Heartbeat:**
		`s:event:type=error`
			• **Available:** Yes
			• **Reserved Variable:** event
			• **Report Name:** *Error Impacted Streams*
			• **Context Data:** `a.media.qoe.error`
			• **Data Feed:** `videoqoeerror`
			• **Audience Manager:** `c_contextdata.a.media.qoe.error`

	The number of streams in which bitrate changes occurred. This metric is set to 1 only if at least one bitrate change event occurred during a playback session. ***Important:*** *If this event is set, the only possible value is TRUE. If this event is not set, no value is sent.*		

Dropped Frame Impacted Streams	• **Key:** *Automatically set*
	• **Required:** No
	• **Type:** string
	• **Sent with:** Close
	• **Min. SDK Version:** Any
	• **Sample value:** `TRUE`
		• **Adobe Analytics:**
		`a.media.qoe.droppedFrames`
		• **Heartbeat:**
		`l:stream:dropped_frames`
			• **Available:** Yes
			• **Reserved Variable:** event
			• **Report Name:** *Dropped Frame Impacted Streams*
			• **Context Data:** `a.media.qoe.droppedFrames`
			• **Data Feed:** `videoqoedroppedframes`
			• **Audience Manager:** `c_contextdata.a.media.qoe.droppedFrames`

Label	Implementation	Network Parameters	Reporting
	The number of streams in which frames were dropped. This metric is set to 1 only if at least one frame was dropped during a playback session. ⚠ ***Important:*** *If this event is set, the only possible value is TRUE. If this event is not set, no value is sent.*		

Stalling Impacted Streams	• **Key:** *Automatically set*
	• **Required:** No
	• **Type:** string
	• **Sent with:** Close
	• **Min. SDK Version:** 1.5+
	• **Sample value:** `TRUE`
		• **Adobe Analytics:**
		`a.media.qoe.stall`
		• **Heartbeat:**
		`s:event:type=stall`
			• **Available:** *Use custom processing rule*
			• **Reserved Variable:** event
			• **Report Name:**
			• **Data Feed:** `N/A`
			• **Context Data:** `a.media.qoe.stall`
			• **Audience Manager:** `c_contextdata.a.media.qoe.stall`

	The number of streams in which a stalled event occurred. This metric is set to 1 only if at least one stall occurred during playback. Customers will have to create their own processing rules to have the value available for reporting. ⚠ ***Important:*** *If this event is set, the only possible value is TRUE. If this event is not set, no value is sent.*		

Stalling Events	• **Key:** *Automatically set*
	• **Required:** No
	• **Type:** string
	• **Sent with:** Close
	• **Min. SDK Version:** 1.5+
	• **Sample value:** `"3"` (Integer)
		• **Adobe Analytics:**
		`a.media.qoe.stallCount`
		• **Heartbeat:**
		`s:event:type=stall`
			• **Available:** *Use custom processing rule*
			• **Reserved Variable:** event
			• **Report Name:**
			• **Context Data:** `a.media.qoe.stallCount`
			• **Data Feed:** `N/A`
			• **Audience Manager:** `c_contextdata.a.media.qoe.stallCount`

	The number of times the playback was stalled during a playback session. Customers will have to create their own processing rules to have the value available for reporting.		

Total Stalling Duration	• **Key:** *Automatically set*
	• **Required:** No
	• **Type:** number
	• **Sent with:** Close
	• **Min. SDK Version:** 1.5+
	• **Sample value:** `12`
		• **Adobe Analytics:**
		`a.media.qoe.stallTime`
		• **Heartbeat:**
		`s:event:type=stall`
			• **Available:** *Use custom processing rule*
			• **Reserved Variable:** event
			• **Report Name:**
			• **Context Data:** `a.media.qoe.stallTime`
			• **Data Feed:** `N/A`
			• **Audience Manager:** `c_contextdata.a.media.qoe.stallTime`

Label	Implementation	Network Parameters	Reporting
	The total time (seconds; integer) the playback was stalled during a playback session. Customers will have to create their own processing rules to have the value available for reporting.		

Validation

List of heartbeat parameters that Adobe collects and processes on the heartbeats server.

	Name	Required / Optional	Data Source	Description
All Events	`s:event:type`	R	VHL SDK	The type of the event being tracked.
	`l:event:prev_ts`	R	VHL SDK	The timestamp of the last event of the same type in this session. The value is `-1` if this is the first event of this type in this video session.
	`l:event:ts`	R	VHL SDK	The timestamp of the event.
	`l:event:duration`	R	VHL SDK	This value is set internally (in milliseconds) by the VHL Library, not by the player. It is used to compute the time spent metrics on the backend. For example `a.media.totalTimePlayed` is computed as a sum of the duration for all the Play (`type=play`) heartbeats that are generated. ***Note:*** *For some of the HB that are sent This parameter is set to 0 for certain events because they are "state change events" (e.g., `type=complete`, `type=chapter_complete`, or `type=bitrate_change`.*
	`l:event:playhead`	R	`VideoInfo` object	The playhead was inside the currently active asset (main or ad), when the event was recorded.
	`s:event:sid`	R	VHL SDK	The session ID (a randomly generated string). All events in a certain session (video + ads) should be the same.
	`l:asset:duration / l:asset:length`	R	`VideoInfo` object	The video asset length of the main asset.

	Name	Required / Optional	Data Source	Description
	(Renamed from `length` to `duration` in version 1.5)			
	`s:asset:publisher`	R	`MediaHeartbeatConfig` object	The publisher of the asset.
	`s:asset:video_id`	R	`VideoInfo` object	An ID uniquely identifying the video in the publisher's catalog.
	`s:asset:type`	R	VHL SDK	The asset type (main or ad).
	`s:stream:type`	R	`VideoInfo` object	The stream type. Can be one of the following: • `live` • `vod` • `linear` .
	`s:user:id`	O	Config object for mobile, app measurement VisitorID	User's specifically set Visitor ID.
	`s:user:aid`	O	Marketing Cloud Org	The user's analytics Visitor ID value.
	`s:user:mid`	R	Marketing Cloud Org	The user's marketing cloud visitor ID value.
	`s:cuser:customer_user_ids_x`	O	`MediaHeartbeatConfig` object	All customer user IDs set on Audience Manager.
	`l:aam:loc_hint`	R	`MediaHeartbeatConfig` object	AAM data sent on each payload after `aa_start`.
	`s:aam:blob`	R	`MediaHeartbeatConfig` object	AAM data sent on each payload after `aa_start`.
	`s:sc:rsid`	R	Report Suit ID (or IDs)	SiteCatalyst RSID where reports should be sent.
	`s:sc:tracking_server`	R	`MediaHeartbeatConfig` object	SiteCatalyst tracking server.
	`h:sc:ssl`	R	`MediaHeartbeatConfig` object	Whether the traffic is over HTTPS (if set to 1) or over HTTP (is set to 0).

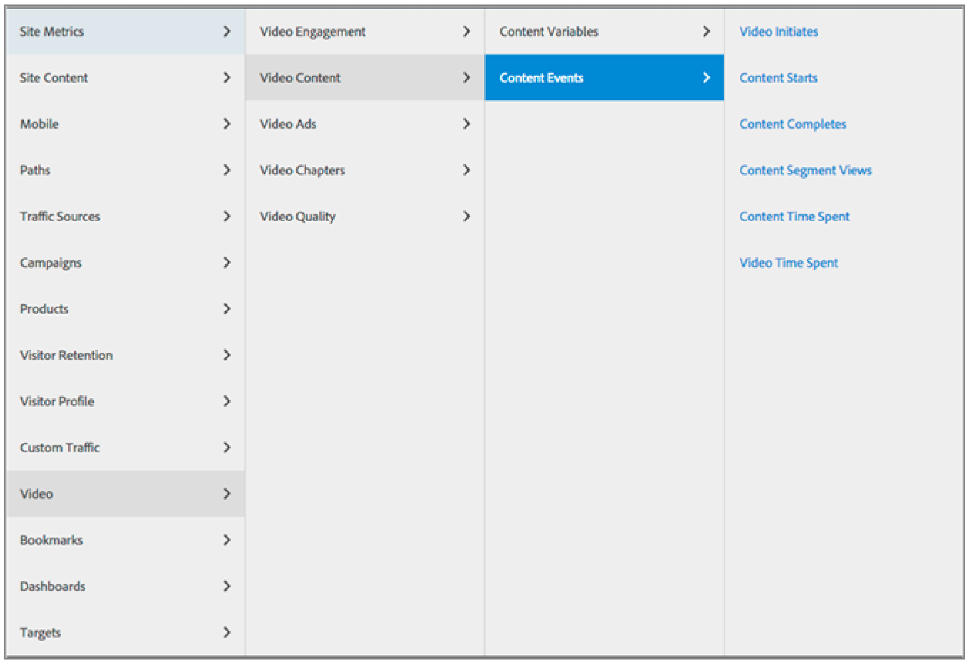
	Name	Required / Optional	Data Source	Description
	`s:sp:ovp`	O	`MediaHeartbeatConfig` object	Set to "primetime" for Primetime players, or the actual OVP for other players.
	`s:sp:sdk`	R	`MediaHeartbeatConfig` object	The OVP version string.
	`s:sp:player_name`	R	`VideoInfo` object	Video player name (the actual player software, used to identify the player).
	`s:sp:channel`	O	`MediaHeartbeatConfig` object	The channel where the user is watching the content. For a mobile app, the app name. For a website, the domain name.
	`s:sp:hb_version`	R	VHL SDK	The version number of the VideoHeartbeat library issuing the call.
	`l:stream:bitrate`	R	`QoSInfo` object	The current value of the stream bitrate (in bps).
Error Events	`s:event:source`	R	VHL SDK	The source of the error, either player-internal, or the application-level.
	`s:event:id`	R	VHL SDK	Error ID, uniquely identifies the error.
Ad Events	`s:asset:ad_id`	R	`AdInfo` object	The name of the ad.
	`s:asset:ad_sid`	R	VHL SDK	A unique identifier generated by the VHL SDK, appended to all ad-related pings.
	`s:asset:pod_id`	R	VHL SDK	Pod ID inside the video. This value is computed automatically based on the following formula: `MD5(video_id) + "_" + [pod index]`
	`s:asset:pod_position`	R	`AdBreakInfo` object	Index of the ad inside the pod (the first ad has index 0, the second ad has index 1, etc.).
	`s:asset:resolver`	R	`AdBreakInfo` object	The ad resolver.

	Name	Required / Optional	Data Source	Description
	`s:meta:custom_ad_metadata.x`	O	`MediaHeartbeat` object	The custom ad metadata.
Chapter Events	`s:stream:chapter_sid`	R	VHL SDK	The unique identifier associated to the playback instance of the chapter. ***Note:*** *A chapter can be played multiple times due to seek-back operations performed by the user.*
	`s:stream:chapter_name`	O	`ChapterInfo` object	The chapter's friendly (i.e., human readable) name.
	`s:stream:chapter_id`	R	VHL SDK	The unique ID of the chapter. This value is computed automatically based on the following formula: `MD5(video_id) + "_" + chapter_pos`
	`l:stream:chapter_pos`	R	`ChapterInfo` object	The chapter's index in the list of chapters (starting with 1).
	`l:stream:chapter_offset`	R	`ChapterInfo` object	The chapter's offset (expressed in seconds) inside the main content, excluding ads.
	`l:stream:chapter_length`	R	`ChapterInfo` object	The chapter's duration (expressed in seconds).
	`s:meta:custom_chapter_metadata.x`	O	`ChapterInfo` object	Custom chapter metadata.

Video Reports

Video variables and events are standard Analytics variables that can be reported directly and added to other Analytics reports.

Video variables and events reports appear in the **Video Content**, **Video Ads**, **Video Chapters**, and **Video Quality** entries of the **Video** menu section.



Video Default Reports

In addition to the metrics and dimensions available when you enable each of the modules, there are three additional dashboard-style reports that become available when you enable the Video Core module. Enabling the Ads module also changes the appearance of some of these dashboard-style reports by adding additional metrics and filters.

Video reports are listed in the **Reports** > **Reports & Analytics** > **Video** section.

Video Report	Description	Common Business Insights
Video Overview	Displays several aggregate measurements to quickly monitor that videos are performing as expected. A graph displays video starts next to ad impressions to let you quickly view these metrics for each video.	• Totals for top video metrics including unique viewers, completion rate, average video metrics, and average videos per visit. • Total content and ad starts for videos filtered by device type or country.
Video Detail	Displays detailed metrics for all videos including starts, concurrent viewers, completion rate, play percentage, and ad impressions.	• Totals for top video metrics including video initiates, content and ad starts, and average videos per visit. • Total content and ad starts for videos filtered by device type or country.

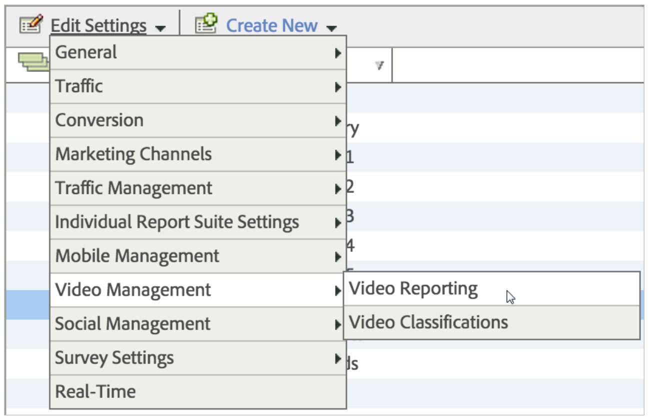
Video Report	Description	Common Business Insights
Video Daypart	Displays content starts by time of day to let you quickly view when your audience is engaged.	• Audience engagement by time of day. • Audience engagement compared to previous date ranges.
Video Events and Video Variables	Reports for additional *Video Reports* are also available. Video metrics and dimensions are standard Analytics variables that can be reported directly and added to other Analytics reports.	• Video Conversion (Events that occur after video is viewed) by generating a report with visits that include a content type of video. • Next/previous video flow using the video name prop.
Video Concurrent Viewers	Displays concurrent viewers during one day. The data can be filtered by content, device type, or country.	• Per-minute audience engagement over a 24-hour interval.

Video Reports Enablement

Each report suite that collects video metrics must be configured before video data is sent.

***Tip:** To take advantage of new capabilities, existing Video Analytics customers should re-enable video tracking for their RSIDs.*

1. In *Reports & Analytics*, click **Admin Tools** > **Report Suites**.
2. Select the report suite(s) where you are collecting video data and click **Edit Settings** > **Video Management** > **Video Reporting**.



3. On the **Video Reporting** page, enable **Video Core**, and optionally enable **Video Ads**, **Video Chapters**, and **Video Quality**.

   Video measurement includes the following modules:

   - **Video Core:** Core video measurement is used for video content. This will use Solution (or Custom) eVars to keep track of Content, Content Type, Content Player Name, and Content Channel. Solution (or Custom) events will be used for Video Initiates, Content Starts, Content Completes, and Content Time Spent.
   - **Video Ads:** Video ad measurement is used for the measurement of ads within the video content. This will use Solution eVvars to measure Ad, Ad Player Name, Ad Pod, and Ad in Pod Position. Solution events will be used for Ad Starts, Ad Completes, Ad Time Spent, and Video Time Spent.
   - **Video Chapters:** Video chapters measurement is used for the measurement of chapters. A chapter is a sub-division of content within a single video. This will use a Solution eVar to store the Chapter ID. Solution events will be used for Chapter Starts, Chapter Completes, and Chapter Time Spent. Additional chapter metadata of Chapter Name and Chapter Position will be provided as classifications of Chapter ID.
   - **Video Quality:** Video quality measurement is used for measuring the quality of the content playback. This will use Solution eVars to store Time to Start, Buffer Events, Total Buffer Duration, Bitrate Switches, Average Bitrate, Errors, and Dropped Frames. Solution events will be used for Time to Start, Drops before Start, Buffer Impacted Streams, Buffer Events, Total Buffer Duration, Bitrate Change Impacted Streams, Bitrate Changes, Avg Bitrate, Error Impacted Streams, Error Events, Dropped Frame Impacted Streams, and Dropped Frames.

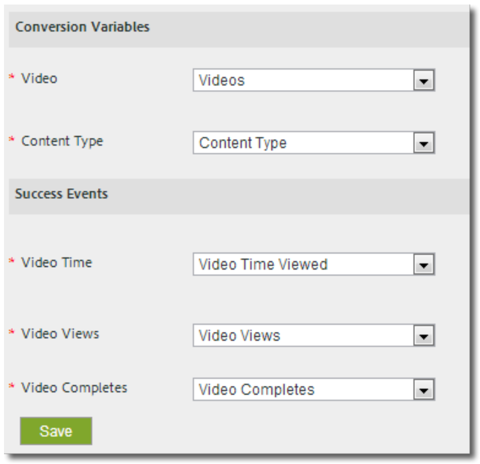
   Enabling each module reserves a set of variables and creates a new set of reports. With the exception of Quality, there will be no data in reports unless the corresponding implementation has been completed. Implementing the Core module also implements the Quality module if you enable it.

   If you are not yet tracking ads, chapters, or playback quality, you can enable additional options at any time.

4. Click **Save**.

   If this report suite is already configured to collect video data, after you click **Save**, an additional configuration page is displayed. If you see the **Video Core Measurement** page, continue to the next step.

5. (Conditional) On the **Video Core Measurement** page, select to continue using custom variables or to use solution variables.

Option	Description
Continue using custom variables.	• **Pros:** Video trending continues to work after migration. • **Cons:** Requires you to keep two custom eVars and three custom events allocated to video. You regain use of one custom eVar and one custom event. To continue using custom variables: 1. Select **Use Custom Variables**, then click **Save**. 2. When prompted, map your current custom eVars and events and then click **Save**:  **Conversion Variables** * Video: Videos * Content Type: Content Type **Success Events** * Video Time: Video Time Viewed * Video Views: Video Views * Video Completes: Video Completes Save
Migrate to solution variables. ***Important:*** *Migrating to solution variables causes you to lose **all** historical trending and comparison for video reports.*	• **Pros:** You regain use of three custom eVars and four custom events. • **Cons:** You lose **all** historical trending and comparison for video reports. This means that you cannot trend video views or video time played for any dates before you migrated to video heartbeat. ***Restriction:*** *Do not migrate to solution variables unless you are certain that you do not want to preserve this trending.* All customers should use solution variables and processing rules to put video data into existing props and eVars, only if they need to preserve historical continuity. To migrate to solution variables: 1. Select **Use Solution Variables** and click **Save**.

Ratings Partners Integration

Parter	**Documentation**
Nielsen	*Digital Content Ratings powered by Adobe*

Federated Analytics

The Federated Analytics service provides a system for sharing Adobe Video Analytics data between two partners. The standardized video measurement data created by Adobe Video Analytics is the hallmark for Federated Analytics, allowing the same data to flow into a single report from multiple sources. Through the rules and logic governing Federated Analytics, data is easily controlled and individualized to meet the needs of each partnership. Federated Analytics makes video measurement more efficient, streamlined, and actionable.

Benefits

- **Transparent:** Strip away the black box of data creation by using the same logic across companies
- **Broad:** Understand the full reach and impact of video consumption across partnerships, platforms, and devices
- **Secure:** Control server-side data sharing through rules and logic
- **Standardized:** Speak the same data language as your partners
- **Actionable:** Quantify video data to benchmark players, monitor trends, and detect anomalies through Adobe Analytics
- **Centralized:** Collect video measurement data in one Adobe location
- **Contractual:** Meet legal data sharing requirements easily
- **Timely:** Send and receive data in near real-time
- **Easy:** Tag players once with Adobe SDKs, share data to many partners

Definitions

- **Sender:** Customer generating video analytics data on owned players
- **Receiver:** Customer receiving video analytics data from sender

Requirements

- **Video Streams Contract:** Receiver and Sender must have contracted Adobe Analytics for Video Streams before gaining access to video data within Adobe Analytics. Contact your account team for more details.
- **Federated Addendum:** Each Sender and Receiver must have a signed addendum in place with Adobe before sending or receiving data. One addendum per customer is required, not one addendum per partnership. Contact your account team for more details.
- **Video Analytics Implementation:** The Sender must have video analytics implemented on all players that will be part of the federated data set. Only Video Analytics data is available for federation. See documentation: *https://marketing.adobe.com/resources/help/en_US/sc/appmeasurement/hbvideo/*
- **Adobe Consulting Contract:** For initial set-up of federated rules between receiver and sender it is valuable to work with consulting services to review data and create the data sharing agreement.

Process

1. Sender and Receiver work together to complete the Federation Rules Agreement form.

Download the current version of the form here:
https://marketing.adobe.com/resources/help/en_US/sc/appmeasurement/hbvideo/federated_analytics_form.pdf.

2. Consulting services provides a sample data file to Receiver with actual data from Sender players to further confirm correct data sharing rules are defined.
3. Sender and Receiver ensures the data sharing agreement will meet all contractual requirements between the two parties.
4. Consulting services sends the completed form to Adobe Engineering to set-up data sharing rules.
5. Data is shared to development report suite where Receiver will review and validate data.
6. Once Receiver confirms data is correct, Adobe Engineering updates the rules to point to a production report suite.
7. Receiver will review and validate data in production report suite.
8. If changes occur to the data set in the future, Sender or Receiver can submit a customer care ticket for support.

Documentation Updates

All updates (additions, deletions, and corrections) to the *Measuring Video in Adobe Analytics using Video Heartbeat Users Guide*, by date.

Last Updated: November 15, 2017

This version of the documentation is a complete rewrite of the documentation, including how to implement heartbeats and Nielsen.

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 Experience 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 @AdobeExpCare*
- *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 *can be added to our Customer Idea Exchange*.

Legal