
# Video Tracking for YouTube 2.0.1

# Pre-Requisites

Each Launch property will need the Adobe Analytics, Experience Cloud Visitor ID Service, and Core extensions installed, and configured from the Extension screen.

Per https://developers.google.com/youtube/player_parameters, use the "Embed a player using an iFrame tag" code snippet in the HTML of each web page where a video player is to render.

This extension, version 2.0.1, supports embedding one or more YouTube videos on a single web page by inserting an "`id`" attribute with a unique value in the iFrame script tag, and appending "`enablejsapi=1`" and "`rel=0`" to the end of the "`src`" attribute value, if not already included, as such:

```
id="player1" width="560" height="315"
src="https://www.youtube.com/embed/xpatB77BzYE?enablejsapi=1&rel
=0" frameborder="0" allow="accelerometer; autoplay; encrypted-
media; gyroscope; picture-in-picture" allowfullscreen
```

Please note that this extension is also designed to dynamically check for a unique "`id`" attribute value, like "`player1`," whether the "`enablejsapi`" and "`rel`" query string parameters exist, and if their expected values are correct. As a result, the YouTube script tag can be added to a web page with or without the "`id`" attribute and the "`enablejsapi`" and "`rel`" query string parameters are included, or not.

On pages with more than one video, note that each video will use the same configuration set in the Launch rule executing on that page. For example, if you create a rule with an event that triggers on video 50% complete, each video on the page will trigger the rule at the 50% cue point.

The Extension relies on the following logic to rewrite the iFrames:

```
document.onreadystatechange = function () { ?if
(document.readyState === 'complete') {
```

Therefore, there will be a slight flicker after the page loads. This behavior is expected.

# Data Elements

There are six data elements available within the extension, none of which require configuration.

1. Playhead Position: This data element records the place, in seconds, of the playhead position on the video timeline, when it is called upon within a Launch Rule.
2. Video ID: This data element specifies the YouTube ID associated with the video.
3. Video Name: This data element specifies the descriptive, or friendly name of the video.
4. Video URL: This data element returns the YouTube.com URL for the currently loaded/playing video.
5. Video Duration: This data element records the total duration, in seconds, of the video content.
6. Extension Version: This data element records the YouTube Tracking Extension version, like "Video Tracking|YouTube|2.0.0," for example.

# Events

There are eight events available within the extension, only Custom Cue Point Tracking requires configuration.

1. Video Ready: This event will trigger when the video is cued, and ready to play.
2. Video Start: This event will trigger when the video is first started, and when `player.getCurrentTime() === 0`
3. Video Pause: This event will trigger when the video is paused.
4. Video Resume: This event will trigger when the video is resumed, and when `player.getCurrentTime() !== 0`
5. Custom Cue Tracking: This event will trigger when the video reaches the specified video threshold percentage. For example, if a video is 60 seconds and the specified cue point is 50%, the event with trigger when playhead position equals 30 seconds. Cue point tracking applies to both initial play, and replay. Note that if user seeks across a cue point, the event will not fire. Cue point events only fire when playhead crosses calculated cuepoint location on timeline, and video player is playing.
6. Video Buffer: This event will trigger when player downloads a certain amount of data before it begins playing the video.
7. Video Ended: This event will trigger when a video fully-completes.
8. Video Replay: This event will trigger every time a video is replayed.

# Usage

There will be one Launch Rule for every Video Event (listed above). As such, the user of this extension needs to create a specific Launch rule for each event they want to track. In other words, if they don't want to track Video Pause, they wouldn't create a rule for it.

Rules have 3 actions:
1. Set variables: Set the Adobe Analytics variables (map to all or some included Data Elements).
2. Send beacon: Send the Adobe Analytics beacon as a custom link tracking call, and provide a "Link Name" value.
3. Clear variables: Clear the Adobe Analytics variables.

Example Launch Rule for "Video Start":
The following Video Extension objects are to be included:

Events:
"Video Start" (this event will cause the rule to fire when the visitor starts playing a YouTube video)

Condition: None

Actions: Using Analytics extension, use the Analytics extension to:

1. "Set Variables" action, to map the event for Video Start, a prop/eVar for the Video Duration data element, a prop/eVar for the Video ID data element, a prop/eVar for the Video Name data element, a prop/eVar for the Video URL data element, a prop/eVar for the Extension Version data element, and
2. Then include the "Send Beacon" action (s.tl) with link name "video start,"
3. Followed by a "Clear Variables" action.

Tip: For implementations where multiple eVars or props for each video element can't be used, data elements values can be concatenated within Launch, parsed into Classification reports using the Classification Rule Builder Tool, per https://docs.adobe.com/content/help/en/analytics/components/classifications/classifications-rulebuilder/classification-rule-builder.html, and then applied as a segment in Analysis Workspace.

To concatenate video information values, create a new data element called "Video Meta Data," and program it to pull in all the Video Data Elements (listed above) and assemble them together, as such:

```
var r = "";
r.push('YouTube');
//Player Name
```

```
r.push(_satellite.getVar('Video ID'));
r.push(_satellite.getVar('Video Name'));
r.push(_satellite.getVar('Video Duration'));
r.push(_satellite.getVar('Extension Version'));

return r.join('|');
```