Video Tracking has been configured on passengers test site in stage -> <https://stage.sonypictures.com/movies/passengers/discanddigitaltest/>

It has been done on the trailer which pop’s up on the page and is auto played.

Additionally, an iframe has been embedded on the page, just to test the ability to track multiple videos on same page. It has been surrounded by comments //Pooja Added for AN-282. It needs to be clicked on to be played.

Following are the findings for YouTube Video Tracking in DTM:

1. To embed a YouTube video in an html page, one needs to us an ‘embed’ YouTube URL in an iframe, object or embed tag (<https://www.w3schools.com/html/html_youtube.asp>)

One can directly get the html code to embed a YouTube video in YouTube under the video->Share->Embed.

1. In order to track a YouTube Video which has been embedded using an iframe, in DTM, we need to do the following :

2.1> The iframe which has the YouTube embed video URL will need to be given an id.

2.2> An additional query parameter needs to be appended in the YouTube embed url mentioned in the iFrame 🡪 enablejsapi=1

Eg: <https://www.youtube.com/embed/7dZ9MywRJP8?rel=0&enablejsapi=1>

2.3> After all the iFrames which have youtube video URL’s are loaded on the page, the YouTube iFrame API js needs to be loaded

<script type="text/javascript" src="https://www.youtube.com/iframe\_api"></script>

2.4> This API will call the function onYouTubeIframeAPIReady() once it is loaded and ready.

Hence, we should define the function onYouTubeIframeAPIReady() on our page. This function should have code to create YouTube Players for all the embedded videos that need to be tracked. The YouTube players should only be created when the iFrame becomes available on the page

Eg:

function onYouTubeIframeAPIReady() {

console.log('\*\*\* iFrame embed onYouTubeIframeAPIReady');

//iFrame with id=” player1” is only available on the page after 2.5 secs.

var timer = setTimeout(function(){player = new YT.Player("player1", {events: {'onReady': onPlayerReady,'onStateChange': onPlayerStateChange}});},3000);

$('a').click(function(){ clearTimeout(timer); });

//iFrame with id=” player2” is only available on the page onload

player2 = new YT.Player("player2", {events: {'onReady': onPlayer2Ready,'onStateChange': onPlayer2StateChange}})

console.log('\*\*\* iFrame Player Defined '+player);

}

Note that we have specified events 'onReady' and 'onStateChange'. The YouTube API will call functions provided against these events, when the events are fired for a YouTube Player on the page.

2.5> Finally, we should define the functions specified for events 'onReady' and 'onStateChange' and call s.Media in them

Eg:

function onPlayerReady(event) {

event.target.playVideo();

console.log('\*\*\* iFrame embed onPlayerReady ', player);

}

function onPlayerStateChange(event) {

console.log('\*\*\* iFrame embed onPlayerStateChange ' + event.data + ' --- YT Player state ' + YT.PlayerState.PLAYING, player.getCurrentTime(), player);

/\*

if (event.data == YT.PlayerState.PLAYING && !done) {

setTimeout(stopVideo, 6000);

done = true;

}

\*/

video\_name = event.target.getVideoData();

video\_name = video\_name.title;

video\_length = event.target.getDuration();

// if(event.data === YT.PlayerState.PLAYING && (event.data === 1 || event.data < 0)){

console.log('\*\*\*\* Pooja \*\*\*\*');

console.log(\_satellite.getToolsByType('sc'));

console.log('\*\*\*\* Pooja \*\*\*\*');

var s = \_satellite.getToolsByType('sc')[0].getS();

if (s == null) {

console.log('\*\*\*\* s is null\*\*\*\*');

s = \_satellite.getToolsByType('sc')[1].getS();

}

if ((event.data === 1 || event.data < 0) && YT.PlayerState.PLAYING === 1) {

//\*-\* PLAY

console.log("\*-\* Player is on play mode " + event.data + ' ' + event.target.getCurrentTime(), s);

if (event.target.getCurrentTime() === 0) {

s.Media.open(video\_name, video\_length, 'Youtube Object Embed');

s.Media.play(video\_name, event.target.getCurrentTime());

} else {

s.Media.play(video\_name, event.target.getCurrentTime());

}

} else if (event.data === 2) {

//\*-\* PAUSE --- CAN USE THIS FOR ENDING TOO =-- check on time -5 sec!!

console.log("\*-\* Player is on pause mode " + event.data + ' ' + event.target.getCurrentTime());

s.Media.stop(video\_name, event.target.getCurrentTime()); //this will cause the monitor to have media.event='STOP'

} else if (event.data === 3) {

//\*-\* SKIPPING

console.log("\*-\* Player is on skipping mode " + event.data);

s.Media.stop(video\_name, event.target.getCurrentTime()); //this will cause the monitor to have media.event='STOP'

} else if (event.data === 0) {

//\*-\* Completed

console.log("\*-\* Player has been completed " + event.data);

s.Media.stop(video\_name, event.target.getCurrentTime());

s.Media.close(video\_name);

}

}

Point#2.3, 2.4 and 2.5 should be done as page load rule.

1. On DTM, we need to Add Adobe Analytics Media tracking module in Adobe Analytics tool. For this we need to move away from Managed by Adobe for adobe analytics tool, choose Custom and add the core code(available at Admin ----> Code Manager---->App Measurement ------> App Measurement Media Module.js) , right before (your Activity Module and) the AppMeasurement Core Code
2. On DTM, we need to put the tracking code for Videos into the Adobe Analytics tool in the Customize Page Code section, right above the doPlugins method. Tracking code available at 🡪 <https://helpx.adobe.com/dtm/kb/how_to_track_youtube_videos_using_DTM.html>

References :

<https://helpx.adobe.com/dtm/kb/how_to_track_youtube_videos_using_DTM.html>

<https://webanalyticsfordevelopers.com/2017/02/21/adobe-dtm-adobe-analytics-media-module-youtube-%E2%9D%A4%EF%B8%8F/>