# How to track youtube videos using DTM?

DTM (Dynamic Tag Management) is a tool that allows you to do things faster. We can track HTML5 videos using event based rules in DTM. However it doesn't have any out of the box functionality to track videos provided by other video providers (Brightcove, Youtube, Vimeo, VideoJS, Flowplayer, etc.)

If someone uses the video service that has some sort of JavaScript API (Brightcove, Youtube, Vimeo, VideoJS, Flowplayer, etc.), we can use DTM to add the API code to listen for the different video events. It is required to load the code you need in a page load rule or App measurement or s_code file (make sure it loads at the correct spot, i.e. page top or page bottom), and that will setup the API and/or listeners to track the videos. Then we have to define what we want to do when it comes to capturing and sending the video data. We can use the Media Module implementation which won't require any additional settings in DTM.

Please see the following steps for tracking the "YouTube Video" through Media Module Implementation.

**Step 1:** YouTube videos can be embedded in a web page either through an iframe or an object tag.

**Object Embed:** Embed the video with with the API turned on e.g. http://www.youtube.com/v/8pB5sRsX5OU?fs=1&rel=0&enablejsapi=1&playerapiid= myytplayer on your webpage.

**Note -** Please make sure you don't have two tags with the same ID, but make sure the tag has an ID that matches the playerapiid parameter

```
<p>Sample YouTube in OBJECT embed tag</p>
 <object id="myytplayer" width="425" height="344"
classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000">
    <param name="movie"
value="http://www.youtube.com/v/AhgtoQIfuQ4?fs=1&rel=0&enablejsapi=1&p
layerapiid=myytplayer"></param>
    <param name="allowFullScreen" value="true"></param>
    <param name="allowscriptaccess" value="always"></param>
    <embed id="myytplayer2"
  src="http://www.youtube.com/v/AhgtoQIfuQ4?fs=1&rel=0wcv8zQo9HyU&enable
  jsapi=1&playerapiid=myytplayer2" type="application/x-shockwave-flash"
  allowscriptaccess="always" allowfullscreen="true" width="425"
  height="344"></embed>
  </object>
```

**Iframe Embed:** Embed the video in an iframe with with the API turned on e.g. http://www.youtube.com/v/8pB5sRsX5OU?fs=1&rel=0&enablejsapi=1 on your webpage.

**Note:** It must provide an "Id" to the iframe. Please see below code for more help

```
<p>YouTube  video in iframe embed tag</p>
<iframe id="player1" frameborder="0" width="425" height="344"
allowfullscreen="1" title="YouTube video player"
src="https://www.youtube.com/embed/wcv8zQo9HyU?enablejsapi=1">
```

**Step 2:** Add Adobe Analytics Media tracking module to your Appmeasurement (or similar). It should be placed somewhere after your config section and before the "DO NOT ALTER ANYTHING BELOW THIS LINE !". In case of legacy implementation, you need to have code version of H.15.1 or newer. You'll get this Media module from your Adobe Analytics Account by going to the following: Admin ----> Code Manager---->App Measurement ------> App Measurement Media Module.js

***Note:*** *If you do not see 'Code Manager' under Admin you do not have access. To get access, have an Admin user add your account to a group that has access to the Code Manager.*

**Step 3:** Add the following code into Head of the web page on which youtube video is embedded just below DTM header code.

```
<script type="text/javascript"
src="https://www.youtube.com/iframe_api"></script>

<script type="text/javascript">
    var video_obj=null;
    var video_length=0;
    var video_name='Movie name ' + new Date().getTime();


    // 3. This function creates an <iframe> (and YouTube player)
    //     after the API code downloads.
    var player;
    function onYouTubeIframeAPIReady() {
        console.log('*** iFrame embed onYouTubeIframeAPIReady');

        player = new YT.Player("player1", {

          events: {

            'onReady': onPlayerReady,
            'onStateChange': onPlayerStateChange
          }
        });
    }

    // 4. The API will call this function when the video player is
ready.
    function onPlayerReady(event) {
        // event.target.playVideo();
        console.log('*** iFrame embed onPlayerReady ', player);
    }

    // 5. The API calls this function when the player's state
changes.
    //     The function indicates that when playing a video
(state=1),
    //     the player should play for six seconds and then stop.
    var done = false;
    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 = player.getVideoData();
        video_name = video_name.title;
        video_length = player.getDuration();

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

```
                console.log("*-* Player is on play mode " +
event.data + ' ' + player.getCurrentTime(), s);
                if(player.getCurrentTime() === 0) {
                    s.Media.open(video_name, video_length, 'Youtube
Object Embed');
                    s.Media.play(video_name,
player.getCurrentTime());
                } else {
                    s.Media.play(video_name,
player.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+' '+player.getCurrentTime());
                s.Media.stop(video_name,
player.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,
player.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, player.getCurrentTime());
                s.Media.close(video_name);
            }
    }

        function onYouTubePlayerReady(playerId) {
            video_obj=document.getElementById(playerId);//playerId

            video_obj.addEventListener("onStateChange",
"onytplayerStateChange");

            video_length= video_obj.getDuration();

            console.log('*-* Youtube Video Ready - in call back
function --- ' + playerId, 'video duration= ' +
video_obj.getDuration(), 'video current time= ' +
video_obj.getCurrentTime());
        }


        function onytplayerStateChange(newState) {
            console.log("*-* Player's new state: " + newState);
            //-1 --> <0 is not started
            if(newState===1){
                //*-* PLAY
                console.log("*-* Player is on play mode " + newState
+ ' ' + video_obj.getCurrentTime(), s);
```

```
                if(video_obj.getCurrentTime() === 0) {
                    s.Media.open(video_name, video_length, 'Youtube
Object Embed');
                    s.Media.play(video_name,
video_obj.getCurrentTime());

                    // s.Media.play(video_name,
video_obj.getCurrentTime(),5,'a segment name', 30);
                } else {
                    s.Media.play(video_name,
video_obj.getCurrentTime());
                }
            }else if(newState===2){
                //*-* PAUSE --- CAN USE THIS FOR ENDING TOO =-- check
on time -5 sec!!
                console.log("*-* Player is on pause mode " +
newState+' '+video_obj.getCurrentTime());
                s.Media.stop(video_name,
video_obj.getCurrentTime());//this will cause the monitor to have
media.event='STOP'

                // s.Media.stop(video_name,
video_obj.getCurrentTime());//this will cause the monitor to have
media.event='STOP'
                //*-*if not used (ie. not pausing) then CLOSE will
kick in when the video completes otherwise video completes will also
send 'STOP'
            }else if(newState===3){
                //*-* SKIPPING
                console.log("*-* Player is on skipping mode " +
newState);
                s.Media.stop(video_name,
video_obj.getCurrentTime());//this will cause the monitor to have
media.event='STOP'
            }else if(newState===0){
                //*-* Completed
                console.log("*-* Player has been completed " +
newState);
                s.Media.stop(video_name, video_obj.getCurrentTime());
                s.Media.close(video_name);
            }
        }

    </script>
```

9
0
9
1
9
2
9
3
9
4
9
5
9
6
9
7
9
8
9
9
1
0
0
1
0
1
1
0
2
1
0
3
1
0
4
1
0
5
1
0
6
1
0
7
1
0
8
1
0
9
1
1
0
1
1
1

1
1
2
1
1
3
1
1
4
1
1
5
1
1
6
1
1
7

**Step 4:** After you insert the code in your project, you need to map the conversion variables and events you are using to track video. Please add the following code in AppMeasurement or s_code file

More information on video tracking available here:

- https://marketing.adobe.com/resources/help/en_US/sc/appmeasurement/video/video_js.html
- https://marketing.adobe.com/resources/help/en_US/sc/appmeasurement/video/video_ref_methods.html
- https://marketing.adobe.com/resources/help/en_US/sc/appmeasurement/video/video_ref_variables.html
- https://marketing.adobe.com/resources/help/en_US/sc/appmeasurement/video/video_js_sample.html