Analytics Tutorial

Bitmovin Analytics + Player Setup

Bitmovin Analytics works through a Data-Collection-Library called bitmovinanalytics-js that records player events and reports them to the Analytics backend. To install bitmovinanalytics-js it has to be loaded through a script tag in the page alongside the Bitmovin Player Javascript.

Include Bitanlytics + Player

- `<script type="text/javascript" src="//bitmovin-a.akamaihd.net/bitmovin-player/stable/7/bitmovinplayer.js"></script>`
- `<script type="text/javascript" src="//bitmovin-a.akamaihd.net/bitmovin-analytics/stable/1/bitmovinanalytics.min.js"></script>`

Setup Bitanalytics + Player

Player HTML Element

```
<div id="player"></div>
```

Bitanalytics Config

Before initializing the Player you should set up Bitanalytics-Js with your License-Key and some information about the played video content. The configuration is per Analytics-Object so if you are loading multiple Videos in one Page you can create multiple instances with different settings.

```
var bitanalyticsConfig = {
 key: "<analytics-key>",
 playerKey: "<player-key>",
 player: bitmovin.analytics.Players.BITMOVIN,
 cdnProvider: bitmovin.analytics.CdnProviders.AKAMAI,
 debug: false
};
var analytics = bitmovin.analytics(bitanalyticsConfig);
```

Configurable Options:

`key`	Your Bitmovin Analytics License Key
`playerKey`	You Bitmovin Player License Key

`player`	The Player tech you are using. Currently only Bitmovin is supported
`cdnProvider`	For information purposes you can set your used CDN tech. Possible available constants are: • `BITMOVIN` • `AKAMAI` • `FASTLY` • `MAXCDN` • `CLOUDFRONT` • `CHINACACHE` • `BITGRAVITY` You can always supply a custom string if your CDN technology is not present in the above.
`debug`	This will enable console.log output for debugging the Bitmovin Analytics-Js

Player Config

Now configure your Bitmovin-Player as per usual (a detailed documentation can be found here: Bitmovin-Player Configuration)

```
var playerConfig = {
 key: "<player-key>",
 source:    {
   dash: "http://bitdash-a.akamaihd.net/content/sintel/sintel.mpd",
   hls:
"https://bitdash-a.akamaihd.net/content/sintel/hls/playlist.m3u8",
   userId:  "customer#123",
   videoId: "Sintel"
 }
};
var player = bitmovin.player("player");
player.setup(playerConfig);
```

Register Player

After setting up the player you have to register the resulting player object with Bitmovin-Analytics to start recording events. This call should happen as soon as possible after the player setup call. (Preferably in the next line)

```
analytics.register(player);
```

Bitmovin Analytics + Player Licensing

Bitmovin Analytics and Player work on localhost out of the box. Any other domain have to be added to your analytics and/or player license.

Add domain to analytics license:
https://bitmovin.com/encoding-documentation/bitmovin-api/#/reference/analytics/queries/add-domain

Add domain to player license:
https://bitmovin.com/encoding-documentation/bitmovin-api/#/reference/player/domains/add-domain

Add example.com to your Analytics License with CURL:

```
curl -H "Content-Type: application/json" -X POST -d '{"url": "example.com"}'
-H X-Api-Key:<api-key>
https://api.bitmovin.com/v1/analytics/licenses/<analytics-license-key>/domain
s
```

Add example.com to your Player License with CURL:

```
curl -H "Content-Type: application/json" -X POST -d '{"url": "example.com"}'
-H X-Api-Key:<api-key>
https://api.bitmovin.com/v1/player/licenses/<player-license-key>/domains
```

Verify your Setup

To check if your Bitmovin-Analytics setup is working please visit https://analytics-dashboard.bitmovin.com/validate-setup and enter your API key to see if data is being recorded.

Bitmovin Analytics API

Documentation:
https://bitmovin.com/encoding-documentation/bitmovin-api/#/reference/analytics

Query Object

Parameter	**Type**	**Description**
dimension	string	The column you want to apply the aggregate function on. IMPRESSION_ID, VIDEO_BITRATE, ...
start	Date (iso string)	Start date. 2016-12-01T00:00:00Z
end	Date (iso string)	End date. (<= start) 2016-12-02T00:00:00Z
interval	string	Time grouping interval. MINUTE, HOUR, DAY, MONTH
groupBy	Array of strings	Grouping columns. BROWSER, STREAM_FORMAT, ...
filters	Array of objects	Filters applied to the query.
limit	number	Max . 150
offset	number	
orderBy	Array of objects	Orderby applied to the query.

Filters

Parameter	Type	Description
name	string	The column that will be filtered. BROWSER, STREAM_FORMAT, ...
operator	string	Operator used. EQ, NE, GT, ...
value	String, number, boolean	The value of the filter. “Chrome”, “dash”, ...

OrderBy

Parameter	Type	Description
name	string	The column after which will be ordered. If a interval and/or groupBy columns are present, only these columns and FUNCTION are allowed as orderBy columns. Use FUNCTION to order by the aggregate function (e.g. avg VIDEO_BITRATE + orderBy FUNCTION will order the results depending on the results of avg VIDEO_BITRATE for each column.
order	string	DESC, ASC

Query Examples

Analytics requests are `POST` requests with the query object as payload. To authenticate with your API-Key you can send the X-Api-Key as header or as a request parameter to authenticate your requests.

Impressions Count

The total number of times the player was loaded in the specified timeframe.

`POST https://api.bitmovin.com/v1/analytics/queries/ count`

```
{
      "dimension":"IMPRESSION_ID",
      "start":"2016-12-01T00:00:00Z",
      "end":"2016-12-02T00:00:00Z"
}
```

Active Impressions Count

The total number of times the player was loaded and the user actually started playback in the specified timeframe.

`POST https://api.bitmovin.com/v1/analytics/queries/ count`

```
{
      "dimension":"IMPRESSION_ID",
      "start":"2016-12-01T00:00:00Z",
      "end":"2016-12-02T00:00:00Z",
      "filters": [{
            "name": "PLAYED",
            "operator": "GT",
            "value": 0
      }]
}
```

Note: The ratio between these two is also the Bounce-Rate for the Video (how many users did not start playback).

Active Impression Count grouped by hour

Same as before but this time grouped by hour.

`POST https://api.bitmovin.com/v1/analytics/queries/ count`

```
{
      "dimension":"IMPRESSION_ID",
      "start":"2016-12-01T00:00:00Z",
      "end":"2016-12-02T00:00:00Z",
      "interval": "HOUR",
      "filters": [{
            "name": "PLAYED",
            "operator": "GT",
            "value": 0
      }]
}
```

`Sample Output:`


```
{
   "requestId":"57a9d6af-1e49-43c2-8e8d-b5ed0de6b5b7",
   "status":"SUCCESS",
   "data":{
      "result":{
         "rowCount":5,
         "rows":[
            [ 1481511600000, 15 ],
            [ 1481536800000, 57 ],
            [ 1481479200000, 19 ],
            [ 1481497200000, 10 ],
            [ 1481522400000, 33 ]
         ]
      },
      "messages":[
         {
            "type":"INFO",
            "text":"Queried successfully."
         }
      ]
   }
}
```

Dash Active Impressions Count grouped by Country

This example demonstrates how to filter for different stream formats (possible values are dash, hls, progressive).

`POST https://api.bitmovin.com/v1/analytics/queries/ count`

```
{
      "dimension":"IMPRESSION_ID",
      "start":"2016-12-01T00:00:00Z",
      "end":"2016-12-02T00:00:00Z",
      "groupBy": ["COUNTRY"],
      "filters": [{
            "name": "PLAYED",
            "operator": "GT",
            "value": 0
      },{
            "name": "STREAM_FORMAT",
            "operator": "EQ",
            "value": "dash"
      }]
}
```

Top 10 videos played by users

`POST https://api.bitmovin.com/v1/analytics/queries/ count`

```
{
      "dimension":"IMPRESSION_ID",
      "start":"2016-12-01T00:00:00Z",
      "end":"2016-12-02T00:00:00Z",
      "groupBy": ["VIDEO_ID"],
      "filters": [{
            "name": "PLAYED",
            "operator": "GT",
            "value": 0
      }],
      "limit": 10,
      "orderBy": [{
            "name": "FUNCTION",
            "order": "DESC"
      }]
}
```

Average startup time (player + video) per video

The filter PAGE_LOAD_TYPE filters the results to impressions that actually happened in the foreground (Certain browsers will slow down loading when the page is loaded in a background tab)

`POST https://api.bitmovin.com/v1/analytics/queries/ avg`

```
{
      "dimension":"STARTUPTIME",
      "start":"2016-12-01T00:00:00Z",
      "end":"2016-12-02T00:00:00Z",
      "groupBy": ["VIDEO_ID"],
      "filters": [{
            "name": "STARTUPTIME",
            "operator": "GT",
            "value": 0
      }, {
           "name": "PAGE_LOAD_TYPE",
           "operator": "EQ",
           "value": 1
      }],
      "orderBy": [{
            "name": "FUNCTION",
            "order": "DESC"
      }]
}
```

Average time to first frame per video

Milliseconds passed between the play command and the first frame played.

`POST https://api.bitmovin.com/v1/analytics/queries/ avg`

```
{
      "dimension":"VIDEO_STARTUPTIME",
      "start":"2016-12-01T00:00:00Z",
      "end":"2016-12-02T00:00:00Z",
      "groupBy": ["VIDEO_ID"],
      "filters": [{
            "name": "VIDEO_STARTUPTIME",
            "operator": "GT",
            "value": 0
      }],
      "orderBy": [{
            "name": "FUNCTION",
            "order": "DESC"
      }]
}
```

Rebuffer time per day

Total milliseconds your users spent rebuffering grouped by day. Combining this metric with the total milliseconds played in a day gives you the rebuffer percentage.

`POST https://api.bitmovin.com/v1/analytics/queries/ sum`

```
{
      "dimension":"BUFFERED",
      "start":"2016-12-01T00:00:00Z",
      "end":"2016-12-08T00:00:00Z",
      "interval":"DAY",
      "filters":[{
            "name":"BUFFERED",
            "operator":"GT",
            "value":0
      },{
            "name":"VIDEOTIME_START",
            "operator":"GT",
            "value":0
      }],
      "orderBy": [{
            "name": "DAY",
            "order": "DESC"
      }]
}
```

Rebuffer time per day

Total milliseconds your users spent rebuffering grouped by day. Combining this metric with the total milliseconds played in a day gives you the rebuffer percentage.

`POST https://api.bitmovin.com/v1/analytics/queries/ sum`

```
{
      "dimension":"BUFFERED",
      "start":"2016-12-01T00:00:00Z",
      "end":"2016-12-08T00:00:00Z",
      "interval":"DAY",
      "filters":[{
            "name":"BUFFERED",
            "operator":"GT",
            "value":0
      },{
            "name":"VIDEOTIME_START",
            "operator":"GT",
            "value":0
      }],
      "orderBy": [{
            "name": "DAY",
            "order": "DESC"
      }]
}
```

Last 20 Impressions

`POST https://api.bitmovin.com/v1/analytics/queries/ min`

```
{
      "dimension":"MINUTE",
      "start":"2016-12-01T00:00:00Z",
      "end":"2016-12-02T00:00:00Z",
      "groupBy": ["IMPRESSION_ID"],
      "filters": [{
            "name": "PLAYED",
            "operator": "GT",
            "value": 0
      }],
      "limit": 20,
      "orderBy": [{
```

```
            "name": "FUNCTION",
            "order": "DESC"
      }]
}
```

Last 20 Impressions that saw a certain Error Code

This can be a very useful query in combination with our deep inspection of individual impressions.

`POST https://api.bitmovin.com/v1/analytics/queries/ min`

```
{
      "dimension":"MINUTE",
      "start":"2016-12-01T00:00:00Z",
      "end":"2016-12-02T00:00:00Z",
      "groupBy": ["IMPRESSION_ID"],
      "filters": [{
            "name": "ERROR_CODE",
            "operator": "EQ",
            "value": 3032
      }],
      "limit": 20,
      "orderBy": [{
            "name": "FUNCTION",
            "order": "DESC"
      }]
}
```

Deep Inspection of one Impression

`GET https://api.bitmovin.com/v1/analytics/impressions/<impressionId>`

This will return a list of Analytics Samples for this particular impression so you can retrace all steps the user has taken within the video (where buffering was encountered, how the bitrate was adapted, seeking behaviour etc).