GEMIUS

Implementing Streaming Media Measurement for JavaScript-controlled video/audio players

06-06-2018

www.gemius.com

List of contents

List of contents ... 2
Introduction ... 3
Implementation ... 4
  Loading the library ... 4
  Object creation of measured player ... 6
  Passing a video object ... 7
  Sending program and ads information ... 8
  Measurement of user actions and player state changes ... 11
    Play ... 11
    Stop ... 12
    Pause ... 13
    Buffering ... 13
    Break ... 13
    Seeking ... 13
    Completion ... 14
    Close ... 14
    Skip ... 14
    Next ... 15
    Previous ... 15
    Resolution Change (optional) ... 16
    Volume Change (optional) ... 17
    Quality Change (optional) ... 18
  General comments ... 19
    Offset parameter ... 19
    Continue event ... 19
    Changing the player size and visibility ... 19
  Sending custom parameters ... 20
  Live streaming scripting ... 21
Example ... 22
Debugging Script with Gemius Debugger ... 26

Introduction

Gemius provides a JavaScript-based solution for streaming media measurement purposes. Used functions allow to collect information on all actions and states registered in the audited video/audio player. In order to obtain precise data about the streaming media consumption, participants are requested to deploy measurement codes that complement functionalities of the player and provide feedback to the study operator on their usage. Stream module of **gemiusPrism™** allows to measure user activities inside the video/audio players embedded in the web page or inside an application on any platform.

Each player (base component), which study focuses on, has a given, assigned **gemiusPrism™** project. Publishers who would like to have all activities in a single **gemiusPrism™** project may do so by defining the brand of a given player, common to the group of players they wish to monitor. Brand can be defined as a player attribute. Instance of the measurement is initiated by call of the **GemiusPlayer** function during loading of player components. **gemiusPrism™** supports measurement of the streaming content defined as a session on the player. Such a session consists of one or multiple states and actions which include:

gross program										
pre-roll break		net program	mid-roll break		net program	mid-roll break		net program	post-roll break	
spot	spot	1st part	spot	spot	2nd part	spot	spot	3rd part	spot	spot

- Player being loaded on the web page and waiting for the first user action.
- Pre-roll commercial break being played – ad spots emitted, then 1st part of material being watched by the user – broadcast of the part of streamed content consequent to pre-roll and preceding mid-roll break (or till the end of content if there are no mid-rolls planned).
- Mid-roll commercial break being played – additional ad spots emitted, then 2nd and consequent parts of material being played in turns with consequent mid-roll commercial breaks (in example displayed above there are in total 3 parts of the streamed material interrupted with 2 mid-rolls).
- End of streaming material, which can be followed by a post-roll commercial break.
- In addition to described states, measured session may include user actions and server-user's device connectivity status which result in: buffering, pause, seeking, skipping to a point in material, proceeding to next material, proceeding to previous material, stopping the material, material completion or player closure. Optionally, measurement may also include information on player's resolution, quality and volume changes.

Implementation

Loading the library

Data collected during the measurement is sent to a dedicated Gemius server called **hitcollector**. In the case of players embedded in websites, path to the main script (gplayer.js) located on the **hitcollector** has to be inserted in the source code of the website between the <head> </head> tag, before the player. The main script can be loaded either synchronously:

```
<script type="text/javascript" src="http://PREFIX.hit.gemius.pl/gplayer.js"></script>
```

or asynchronously:

```
<script type="text/javascript">
<!--//--><![CDATA[//><!--
function gemius_player_pending(obj,fun) {obj[fun] = obj[fun] || function() {var x =
window['gemius_player_data'] = window['gemius_player_data'] || [];
x[x.length]=[this,fun,arguments];};};
gemius_player_pending(window,"GemiusPlayer");
gemius_player_pending(GemiusPlayer.prototype,"newProgram");
gemius_player_pending(GemiusPlayer.prototype,"newAd");
gemius_player_pending(GemiusPlayer.prototype,"adEvent");
gemius_player_pending(GemiusPlayer.prototype,"programEvent");
gemius_player_pending(GemiusPlayer.prototype,"setVideoObject");
(function(d,t) {try {var gt=d.createElement(t),s=d.getElementsByTagName(t)[0],
l='http'+((location.protocol=='https:')?'s':''); gt.setAttribute('async','async');
gt.setAttribute('defer','defer'); gt.src=l+'://PREFIX.hit.gemius.pl/gplayer.js';
s.parentNode.insertBefore(gt,s);} catch (e) {}})(document,'script');
//--><!]]>
</script>
```

In the case of asynchronous implementation, there is a possibility that the player and material will be loaded prior to loading of the measurement script. In such a situation, all player states and actions taken by the Internet user are still measured and sent to **hitcollector** as soon as the script loads. Streaming media measurement may be run alongside audience study measurement on one or numerous dedicated accounts.

PREFIX in path should be changed to a prefix value dedicated for a given market (e.g. gapl).

In order to properly measure viewability of the player embedded in an HTML frame, the script has to be included both in the embedded HTML frame and in the embedding webpage. If there are several levels of embedding, the script has to be included on each level to ensure accurate measurement of player viewability.

Object creation of measured player

Once player component has been created in the browser, stream measurement script is initialized through the call of the **GemiusPlayer** function and creation of an object, which will be used to pass on all information about the given player. A separate object has to be created for each measured player instance present in the document displayed by the browser.

```
var player = new GemiusPlayer(playerID,gemiusID,additionalParameters);
```

Arguments description		
playerID[*String*]	Obligatory	Brand of the player used to aggregate results of all measured instances of the player with the single brand (max. 64 characters; allowed are: a-z, A-Z and national characters, 0-9, and special characters:_ . ! @ # * ( ) - / ? : ; ~ $,).
gemiusID[*String*]	Obligatory	Tag identifier (provided by Gemius), assigned to unique **gemiusPrism™** project, on which is recorded conducted measurement result of given player.
additionalParameters[*Dictionary*]	Optional	Dictionary with additional parameters describing the player and its default settings.
Additional parameters description		
currentDomain[*String*]	Optional	Set name of domain in which given player were to be embedded. Used in syndication in case of multi layered frame and iframe embedding.
volume[*Number*]	Optional	Pre-set % value of volume (range between 0 and 100). In case of mute option, value should be set to -1.
resolution[*String*]	Optional	Pre-set value of player resolution (e.g. 1024x768). Actual, physical size of player component (default size). If a video object is passed to the main script, the resolution parameter is redundant. For more information, please refer to the Passing a video object section.

Additional parameters included in object creation allow to pass default settings of the player, which can then be further modified by either actions of the user or pre-set emission characteristics of the content.

Passing a video object

Automatic detection of the player size and its visibility is only possible if information about the location of the player is passed to the main script. Still, before any information about a video object is passed, first a player object has to be created. In order to pass a video object to the main script, the following method has to be used:

```
player.setVideoObject(document.getElementById("videoId"));
```

where `videoId` is any ID assigned to the video element, e.g.:

```
<video id="videoId" .... >
```

If a new player object is created on the same webpage and the playback continues in a new instance of the player, the location of the video element should be updated by calling the `setVideoObject` function again.

Sending program and ads information

After streaming content is loaded into the player, it is necessary to pass its description to **gemiusPrism™**. For this purpose, the **newProgram** function of the player object has to be called. This has to be done before and regardless of whether or not actual emission of content or advertising spots started (be it via user's action or auto-play function).

```
player.newProgram(programID,additionalParameters);
```

Arguments description		
programID[*String*]	Obligatory	The unique identifier of the content broadcasted in the net parts of the program (max. 64 characters; allowed are: a-z, A-Z and national characters, 0-9, and special characters:_ . ! @ # * ( ) - / ? : ; ~ $,). E.g. "123456" for the "Game of Thrones", season 2, episode 1
additionalParameters[*Dictionary*]	Obligatory	Dictionary with additional parameters describing the material loaded to the player and its settings.
Additional parameters description		
programName[*String*]	Obligatory	Title of the content broadcasted in the net parts of the program (max. 64 characters; allowed are: a-z, A-Z and national characters, 0-9, and special characters:_ . ! @ # * ( ) - / ? : ; ~ $,). E.g. "The North Remembers" – title of the episode of the "Game of Thrones" series.
programDuration[*Number*]	Obligatory	The total duration of the content broadcasted in the net parts of the program in seconds. When it is not possible to evaluate duration (e.g. in live streaming, like direct streaming of TV channel) the value of programDuration should be set to -1.
programType[*String*]	Obligatory	Type of content. Allowed values are: 'Audio', 'Video'.
series[*String*]	Optional	Hierarchical description of Series or other content broadcasted in Series/Season model, slash separated (max. 64 characters; allowed are: a-z, A-Z and national characters, 0-9, and special characters:_ . ! @ # * ( ) - / ? : ; ~ $,). E.g. 'Game of Thrones/Season 1' or 'Champions League/Season 2014-2015'
typology[*String*]	Optional	Hierarchical categorization of the content, which can be common for the market aligned to TV study provider (i.e. like Nielsen's: Sport/Football, Movie/Class B).
premiereDate[*String*]	Optional	Date of the first publication of the program content on client web site (YYYYMMDD format).

externalPremiereDate[*String*]	Optional	Date of the first publication out-site the service. For example can be used for pre-premiere analysis of TV series or movie market publication frame analysis, how long it takes from cinema to Internet (YYYYMMDD format).
quality[*String*]	Optional	Pre-set value (e.g. 1920x1080) of loaded material's quality. Can be further adjusted by the internet user.
resolution[*String*]	Optional	Pre-set value of loaded material's resolution, which may alter default or user settings of player container (e.g. 1280x720). If a video object is passed to the main script, the resolution parameter is redundant. For more information, please refer to the Passing a video object section.
volume[*Number*]	Optional	Pre-set value of loaded material's volume, which may alter default or user settings of volume.
customAttributes$_i$	Optional	Additional attributes of a material. Its names and values are defined by the study participant.

This state is recorded and can be separated in raw data. However, in **gemiusPrism™** metrics pertain to data post contact of a user with the streaming content or ad (start of an ad or 1st part of program is required). When the playing process begins, measurement has to be initiated, which begins the gross viewing of the whole program. This indicates that the user is willing to see given content. However, it does not yet constitute actual start of viewing the content (net part of the program). Due to possibility of user quitting during pre-roll ad block or occurrence of technical or network problems, measured session may end without the user viewing the net part of the program at all.

Dedicated **programEvent** and **adEvent** functions are used to measure all states and user actions taken during the session regardless of gross program structure: number and type of ad breaks (and spots in them) and actual viewed net program parts which it is split into. If there is a pre-roll ad played before the actual streaming content information about the spot has to be sent via the **newAd** function.

This method has to be called just before playing any given ad in the player.

```
player.newAd(adID,additionalParameters);
```

Arguments description		
adID[*String*]	Obligatory	A unique identifier of an advertisement (max. 64 characters; allowed are: a-z, A-Z and national characters, 0-9, and special characters:_ . ! @ # * ( ) - / ? : ; ~ $,).
additionalParameters[*Dictionary*]	Optional	Dictionary with additional parameters describing the advertisement and its settings.
Additional parameters description		
adName[*String*]	Optional	Title of the advertisement (i.e. 'Ferrari test drive'). Strongly advised to define it, to not rely only on separation via adID (max. 64 characters; allowed are: a-z, A-Z and national characters, 0-9, and special characters:_ . ! @ # * ( ) - / ? : ; ~ $,).
adDuration[*Number*]	Optional	Total length of the advertisement in seconds, integer value.
adType[*String*]	Optional	Type of advertisement. Proposed values are: 'promo', 'spot', 'sponsor'.
campaignClassification[*String*]	Optional	Hierarchical classification of the campaign, including: campaign name, brand, producer, slash separated (max. 64 characters; allowed are: a-z, A-Z and national characters, 0-9, and special characters:_ . ! @ # * ( ) - / ? : ; ~ $,).
quality[*String*]	Optional	Pre-set value (e.g. 1920x1080) of loaded commercial's quality. Can be further adjusted by the internet user.
resolution[*String*]	Optional	Pre-set value of loaded advertisement's resolution, which may alter default or user settings of player container (e.g. 1280x720). If a video object is passed to the main script, the resolution parameter is redundant. For more information, please refer to the Passing a video object section.
volume[*Number*]	Optional	Pre-set value of loaded advertisement's volume, which may alter default or user settings of volume.
customAttribute$_i$	Optional	Additional attributes of an advertisement. Its names and values are defined by the study participant.

Once information on loaded content is ready, functions carrying information on internet users actions and player states changes have sufficient information input.

Measurement of user actions and player state changes

After player object is defined and proper ad and material is described, everything is ready to pass information about internet user actions and player state changes. Such information is passed with use of **programEvent** and **adEvent** functions.

Play

To initiate viewing of material or ad first information on **play** event needs to be sent. This can be result of auto-play setting or internet user action.

```
player.adEvent(programID,adID,offset,”play”,additionalParameters);
```

Arguments description		
programID[*String*]	Obligatory	A unique program identifier declared in the earlier **newProgram** function call (max. 64 characters; allowed are: a-z, A-Z and national characters, 0-9, and special characters:_ . ! @ # * ( ) - / ? : ; ~ $,).
adID[*String*]	Obligatory	A unique advertising spot identifier declared in the earlier **newAd** function call (max. 64 characters; allowed are: a-z, A-Z and national characters, 0-9, and special characters:_ . ! @ # * ( ) - / ? : ; ~ $,).
offset[*Number*]	Obligatory	Offset in content in seconds where the event occurs.
additionalParameters[*Dictionary*]	Obligatory	Dictionary with additional parameters describing the play event for given commercial.
Additional parameters description		
autoPlay[*Bool*]	Optional	Information on mode in which ad is being started. Allowed values are: ‘true’, ‘false’.
adPosition[*Number*]	Optional	Position of advertisement in break.
breakSize[*Number*]	Optional	Number of ads in break given ad belongs to.
resolution[*String*]	Optional	Resolution to which player auto-switched on start of viewing the advertisement. If a video object is passed to the main script, the resolution parameter is redundant. For more information, please refer to the Passing a video object section.
volume[Number]	Optional	Volume to which player auto-switched on start of viewing the advertisement.

This method must be called just before playing an ad. Each advertisement must be previously registered by call of **newAd** function. Each ad inside the commercial break has to be notified to the system (sent via **newAd** function separately) before its start is registered (sent via **adEvent** function). In similar fashion **play** event information is sent with use of **programEvent** function.

```
player.programEvent(programID,offset,”play”,additionalParameters);
```

Arguments description		
programID[*String*]	Obligatory	A unique program identifier declared in the earlier **newProgram** function call (max. 64 characters; allowed are: a-z, A-Z and national characters, 0-9, and special characters:_ . ! @ # * ( ) - / ? : ; ~ $,).
offset[*Number*]	Obligatory	Offset in content in seconds where the event occurs.
additionalParameters[*Dictionary*]	Obligatory	Dictionary with additional parameters describing the play event for given material.
Additional parameters description		
autoPlay[*Bool*]	Optional	Information on mode in which material is being started. Allowed values are: ‘true’, ‘false’.
partID[*Number*]	Optional	Position of the partial in program from 1..n. If due to configuration of the service, user can start program from the middle of second partial this number is 2 – absolute position of the part viewed.
resolution[*String*]	Optional	Resolution to which player auto-switched on start of viewing the material. If a video object is passed to the main script, the resolution parameter is redundant. For more information, please refer to the Passing a video object section.
volume[Number]	Optional	Volume to which player auto-switched on start of viewing the material.

Dependent of whether or not the pre-roll commercial break is present, streaming may start either with a play registered via **adEvent** or **programEvent** function. If there is no pre-roll present, viewing of 1st part of content is not going to be between two blocks of commercials but at the beginning. Separate content parts do not need to be viewed in consecutive order (user may freely start with or switch between any of the parts). Regardless of positioning and order in which they are viewed.

During emission of any of the gross program elements (ads and net program parts), there is a number of actions user may take and states which may pertain to player performance. Applying dedicated function to player object allows to measure any of them. Seven events notifying about emission interruption of an advertisement or material do not contain any additional parameters.

Stop

User pressed ‘**stop**’ button on the player or took an action with result equal to it; view of material ceased and progress returned to the start of the material.

```
player.adEvent(programID,adID,offset,"stop");
```

```
player.programEvent(programID,offset,"stop");
```

Pause

User pressed '**pause**' button on the player or took an action with result equal to it; viewing of material temporarily ceased and progress remains at position this event occurred.

```
player.adEvent(programID,adID,offset,"pause");
```

```
player.programEvent(programID,offset,"pause");
```

Buffering

buffering – user did not take any action but player concluded playing already loaded chunk of material and attempts to load another chunk from the server before resuming emission.

```
player.adEvent(programID,adID,offset,"buffer");
```

```
player.programEvent(programID,offset,"buffer");
```

Break

break – user did not take any action but player ceased playing loaded material in order to emit commercial block, after which material emission resumes (next part is played or program concludes in case of post-roll).

```
player.programEvent(programID,offset,"break");
```

Seeking

seeking – user switched to inconsequent time point of material, e.g. clicked in progress bar or took an action with result equal to it, attempting to skip part of material or return to earlier portion of it in comparison to current position in material.

```
player.adEvent(programID,adID,offset,"seek");
```

```
player.programEvent(programID,offset,"seek");
```

Completion

completion – last second of material or ad has been emitted.

```
player.adEvent(programID,adID,offset,"complete");
```

```
player.programEvent(programID,offset,"complete");
```

Close

close – conclusion of material viewing by closing a browser window. Event **close** has to be reported only when the material is not yet finished and it was changed to another material or player window closed.

```
player.adEvent(programID,adID,offset,"close");
```

```
player.programEvent(programID,offset,"close");
```

Different group of events and player states are these, which pass additional information on internet user's behavior and interaction with the player. These include information on interactions with the play lists, and manual changes to volume, window size and quality settings in the player.

Skip

skip – user used a skip button or took an action with results equal to it, which moved emission to next part of material or point in progress within the single material.

```
player.programEvent(programID,offset,"skip");
```

```
player.adEvent(programID,adID,offset,"skip");
```

Next

next – user used a next button or took an action with result equal to it, which changes played material to pre-set new material (next item on the list).

```
player.programEvent(programID,offset,”next”,additionalParameters);
```

Previous

previous – user used a previous button or took an action with result equal to it, which changes played material to pre-set new material (previous item on the list).

```
player.programEvent(programID,offset,”prev”,additionalParameters);
```

Arguments description		
programID[*String*]	Obligatory	A unique program identifier declared in the earlier **newProgram** function call (max. 64 characters; allowed are: a-z, A-Z and national characters, 0-9, and special characters:_ . ! @ # * ( ) - / ? : ; ~ $,).
offset[*Number*]	Obligatory	Offset in content in seconds where the event occurs.
additionalParameters[*Dictionary*]	Optional	Dictionary with additional parameters describing the previous event for given material.
Additional parameters description		
listID[*Number*]	Optional	Unique identifier of the list which items were used to navigate the content.

Resolution Change (optional)

resolution change – user changed resolution of the player or took an action with results equal to it, during emission of an ad or material. If a video object is passed to the main script, the resolution change event is redundant. For more information, please refer to the Passing a video object section.

```
player.adEvent(programID,adID,offset,"chngRes",additionalParameters);
```

Arguments description		
programID[*String*]	Obligatory	A unique program identifier declared in the earlier **newProgram** function call (max. 64 characters; allowed are: a-z, A-Z and national characters, 0-9, and special characters:_ . ! @ # * ( ) - / ? : ; ~ $,).
adID[*String*]	Obligatory	A unique advertising spot identifier declared in the earlier **newAd** function call (max. 64 characters; allowed are: a-z, A-Z and national characters, 0-9, and special characters:_ . ! @ # * ( ) - / ? : ; ~ $,).
offset[*Number*]	Obligatory	Offset in content in seconds where the event occurs.
additionalParameters[*Dictionary*]	Optional	Dictionary with additional parameters describing the resolution change event for given commercial
Additional parameters description		
resolution[*String*]	Optional	Value of player resolution (e.g. 1024x768) set by the user. Actual, physical size of player window.

```
player.programEvent(programID,offset,"chngRes",additionalParameters);
```

Arguments description		
programID[*String*]	Obligatory	A unique program identifier declared in the earlier **newProgram** function call (max. 64 characters; allowed are: a-z, A-Z and national characters, 0-9, and special characters:_ . ! @ # * ( ) - / ? : ; ~ $,).
offset[*Number*]	Obligatory	Offset in content in seconds where the event occurs.
additionalParameters[*Dictionary*]	Optional	Dictionary with additional parameters describing the resolution change event for given material.
Additional parameters description		
resolution[*String*]	Optional	Value of player resolution (i.e. 1024x768) set by the user. Actual, physical size of player window.

Volume Change (optional)

volume change – user used a change volume slider, mute button or took an action with results equal to it, adjusting player's volume.

```
player.adEvent(programID,adID,offset,”chngVol”,additionalParameters);
```

Arguments description		
programID[*String*]	Obligatory	A unique program identifier declared in the earlier **newProgram** function call (max. 64 characters; allowed are: a-z, A-Z and national characters, 0-9, and special characters:_ . ! @ # * ( ) - / ? : ; ~ $,).
adID[*String*]	Obligatory	A unique advertising spot identifier declared in the earlier **newAd** function call (max. 64 characters; allowed are: a-z, A-Z and national characters, 0-9, and special characters:_ . ! @ # * ( ) - / ? : ; ~ $,).
offset[*Number*]	Obligatory	Offset in content in seconds where the event occurs.
additionalParameters[*Dictionary*]	Optional	Dictionary with additional parameters describing the volume change event for given commercial.
Additional parameters description		
volume[*Number*]	Optional	% value of set volume (range between 0 and 100) by the user. In case of mute, value should be set to -1.

```
player.programEvent(programID,offset,”chngVol”,additionalParameters);
```

Arguments description		
programID[*String*]	Obligatory	A unique program identifier declared in the earlier **newProgram** function call (max. 64 characters; allowed are: a-z, A-Z and national characters, 0-9, and special characters:_ . ! @ # * ( ) - / ? : ; ~ $,).
offset[*Number*]	Obligatory	Offset in content in seconds where the event occurs.
additionalParameters[*Dictionary*]	Optional	Dictionary with additional parameters describing the volume change event for given material.
Additional parameters description		
volume[*Number*]	Optional	% value of set volume (range between 0 and 100) by the user. In case of mute, value should be set to -1.

Quality Change (optional)

quality change – user changed the quality of content in the player or took an action with results equal to it, during emission of an ad or material.

```
player.adEvent(programID,adID,offset,”chngQual”,additionalParameters);
```

Arguments description		
programID[*String*]	Obligatory	A unique program identifier declared in the earlier **newProgram** function call (max. 64 characters; allowed are: a-z, A-Z and national characters, 0-9, and special characters:_ . ! @ # * ( ) - / ? : ; ~ $,).
adID[*String*]	Obligatory	A unique advertising spot identifier declared in the earlier **newAd** function call (max. 64 characters; allowed are: a-z, A-Z and national characters, 0-9, and special characters:_ . ! @ # * ( ) - / ? : ; ~ $,).
offset[*Number*]	Obligatory	Offset in content in seconds where the event occurs.
additionalParameters[*Dictionary*]	Optional	Dictionary with additional parameters describing the quality change event for given commercial.
Additional parameters description		
quality [*String*]	Optional	Value (e.g. 1920x1080) of content quality set by the user.

```
player.programEvent(programID,offset,”chngQual”,additionalParameters);
```

Arguments description		
programID[*String*]	Obligatory	A unique program identifier declared in the earlier **newProgram** function call (max. 64 characters; allowed are: a-z, A-Z and national characters, 0-9, and special characters:_ . ! @ # * ( ) - / ? : ; ~ $,).
offset[*Number*]	Obligatory	Offset in content in seconds where the event occurs.
additionalParameters[*Dictionary*]	Optional	Dictionary with additional parameters describing the quality change event for given material.
Additional parameters description		
quality[*String*]	Optional	Value (i.e. 1920x1080) of content quality set by the user.

General comments

Offset parameter

Each of actions on the player is done in context to the current **offset** of the content broadcasted in the net part of the program. **Offset** means the second within which action is executed. If the action takes place before 1st part of the content **offset** is set to 0. Each of the actions, which interrupts playing should be followed by appropriate **play** event.

Continue event

If material is being played without any other action interrupting it for 5 minutes, **continue** event is being automatically generated to sustain duration of visits and provide better granularity of time measurement. In the event of closing the browser containing measured player **unload** event is triggered which returns to **gemiusPrism™** details on current **offset** and appropriate identifiers. Please note that for the duration of commercial break and emission of separate ads, started program and given view of the parts should be in **break** state.

Changing the player size and visibility

If the setVideoObject function has been called indicating the webpage element that is the video player, each hit includes information about player resolution. If the size of the player changes during playback, the continue hit with the old size of the player is sent.

If the script is able to verify visibility (conditions described in the Loading the library section are met and the video object was passed to the script), the visibility of the player is checked every second and sent in every hit. The visibility parameter can take the value of **0** (player not visible) or **1** (player visible).

It is assumed that the player is visible when the video object passed by the setVideoObject method is visible at least in 50%. If visibility of the player changes during playback, the continue hit is sent with the previous visibility value. This makes it possible to count the playback time while the player is visible.

It needs to be remembered that a change in visibility is detected when:

- the browser window is minimized or moves to the background,
- switching between tabs,
- scrolling the webpage,
- scrolling the HTML frame content.

A change in visibility is not detected when:

- the screen is turned off,
- screensaver is displayed,
- screen lock is enabled
- the browser window is covered by another window.

Sending custom parameters

Using **gemiusPrism™** publishers can define and maintain any form of internal categorisation. Functions called during stream measurement have among **additionalParameters** specified space to define and pass argument called **customAttributes**. These can be used to include information needed to set up categories, as they are part of material and advertisement description. They should follow the dictionary record format of 'key':value.

```
'customAttribute1':attributeValue1,

'customAttribute2':attributeValue2,

'customAttribute3':attributeValue3,

'customAttribute4':attributeValue4,

'customAttribute5':attributeValue5;
```

Each material can be accompanied with maximum 5 of such attributes. Categorization definitions have to follow predefined rule set:

- The following characters are allowed: a-z, A-Z and national characters, 0-9, and special characters: _ . ! @ # * ( ) - / ? : ; ~ $,
- total character count may not exceed 200 characters
- names starting with '_' character are not allowed (they are restricted to formal measurement usage)
- attribute names defined for material and advertisements need to be unique in relation to one another.

To provide full scope of descriptive data regarding the session, viewed content and ads Gemius proposes set of compulsory and optional defined parameters. They are included as part of sent information on registered state or action in the player and are always accounted in the **additionalAttributes** list.

Live streaming scripting

Live materials should be scripted analogously as described above, but with two differences:

1. The **programDuration[*Number*]** parameter
   When it is not possible to determine the duration (e.g. in live streaming or live streaming of a TVchannel), the value of the *programDuration* parameter should be set to -1.

2. The ***offset[Number]*** parameter
   In the case of live streaming content, the *offset* parameter should be the same as the timestamp, which determines when the given material was broadcasted. For materials that can be watched with a delay, the offset parameter will be equal to the current time minus the time shift offset, and for materials that cannot be watched with a delay, it will be the current time.

Example

Let us analyze a scenario, in which we want to set up measurement of played material. We have available following information on material itself. And we know there is no pre-roll commercial break but there are 2 spots in mid-roll break and one spot in post-roll break. Program itself is thus broken into two parts. It also has an auto-play setting and emission starts as soon as player and content loads.

```
gemiusID = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.xx

 programID = 930245

 progamName = material1

 duration = 180 // length of the material in seconds

 intName = Furbe

 intCategory = Comedy

 intType = vod

 intStatus = public

 intCanal = canal1
```

Initially, the measurement script has to be loaded and player object created.

```
<script type="text/javascript">
<!--//--><![CDATA[//><!--
function gemius_player_pending(obj,fun) {obj[fun] = obj[fun] || function() {var x =
window['gemius_player_data']       =       window['gemius_player_data']       ||       [];
x[x.length]=[this,fun,arguments];};};
gemius_player_pending(window,"GemiusPlayer");
gemius_player_pending(GemiusPlayer.prototype,"newProgram");
gemius_player_pending(GemiusPlayer.prototype,"newAd");
gemius_player_pending(GemiusPlayer.prototype,"adEvent");
gemius_player_pending(GemiusPlayer.prototype,"programEvent");
gemius_player_pending(GemiusPlayer.prototype,"setVideoObject");
(function(d,t)    {try    {var    gt=d.createElement(t),s=d.getElementsByTagName(t)[0],
l='http'+((location.protocol=='https:')?'s':''); gt.setAttribute('async','async');
gt.setAttribute('defer','defer');        gt.src=l+'://PREFIX.hit.gemius.pl/gplayer.js';
s.parentNode.insertBefore(gt,s);} catch (e) {}})(document,'script');
//--><!]]>
```

```
</script>
```

```
var player = new GemiusPlayer(“192038”,”xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.xx”,
{"currentDomain":"example.com", "volume":100} );
```

A video object is passed in order for the player size and its visibility to be automatically detected. The video element ID is defined as “myvideo”.

```
player.setVideoObject(document.getElementById("myvideo");
```

When the material is loaded into the player, the **newProgram** function is called.

```
player.newProgram(930245,{
        'programType':'Video',
        'programDuration':180,
        'programName':'material1',
        'series':'season 2',
        'typology':'Movie/Class C',
        'premiereDate':'20150401',
        'externalPremiereDate':'20150101',
        'quality':'1920x1080',
        'volume':80,
        'intName':'Furbe',
        'intCategory':'Comedy',
        'intType':'vod',
        'intStatus':'public',
        'intCanal':'canal1'
});
```

Because auto-play is enabled, the playing event occurs immediately. To report this, the **playProgram** function is called.

```
player.programEvent(930245,0,”play”,{"autoPlay":true, "partID":1});
```

After 5 seconds of playing, the user pauses the emission by pressing the pause button. This triggers the **pause** event. After material streaming is resumed, the **play** event is sent.

```
player.programEvent(930245,5,”pause”);

player.programEvent(930245,5,”play”,{"autoPlay":false, "partID":1});
```

Finally, after 90 seconds of playing, the material is interrupted by a mid-roll commercial break. After 2 seconds of an ad playing, the user changes the material to a different one. This triggers the **close** event.

```
player.programEvent(930245,90,”break”);
player.newAd(49327505,{
        'adName':'ColaT’,
        'adType':'promo',
        'adDuration':5,
        'campaignClassification':'unclassified',
        'quality':'1920x1080',
        'volume':100
});
player.adEvent(930245,49327505,90,”play”,{"autoPlay":true,
 "addPosition":1, “breakSize”:2});
player.adEvent(930245,49327505,90,”close”);
player.programEvent(930245,90,”close”);
```

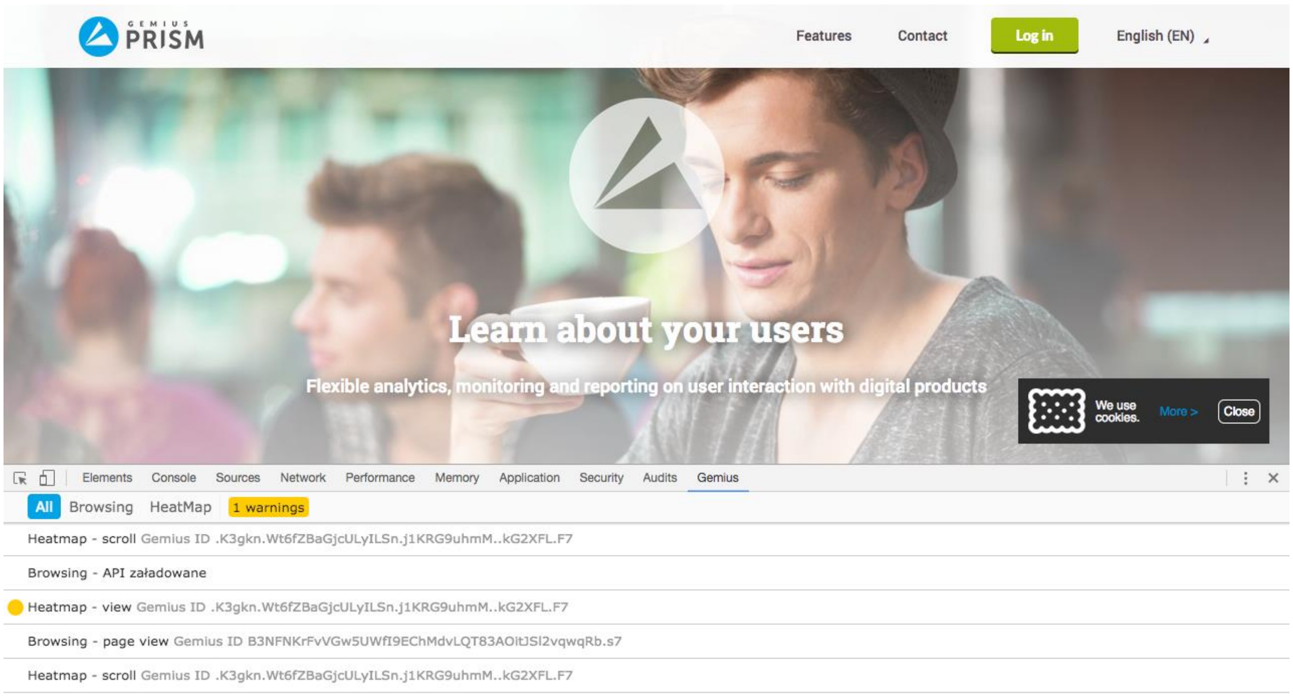
Measurement included start of first spot advertisement out of two. This ad did not have any campaign classification assigned. Finally, streaming of material concludes.

Debugging Script with Gemius Debugger

Gemius Debugger is a Chrome browser extension that helps to verify Gemius script installation.

The extension displays information whether or not Prism scripts are properly installed and working correctly on your website and all its subpages. When the debugging mode is enabled, a list of standard, e-commerce and stream hits is presented with names, comments and warnings/errors counter. A solution to each warning/error found is also provided.

To start the verification process, install Gemius Debugger from http://debugger.gemius.com.

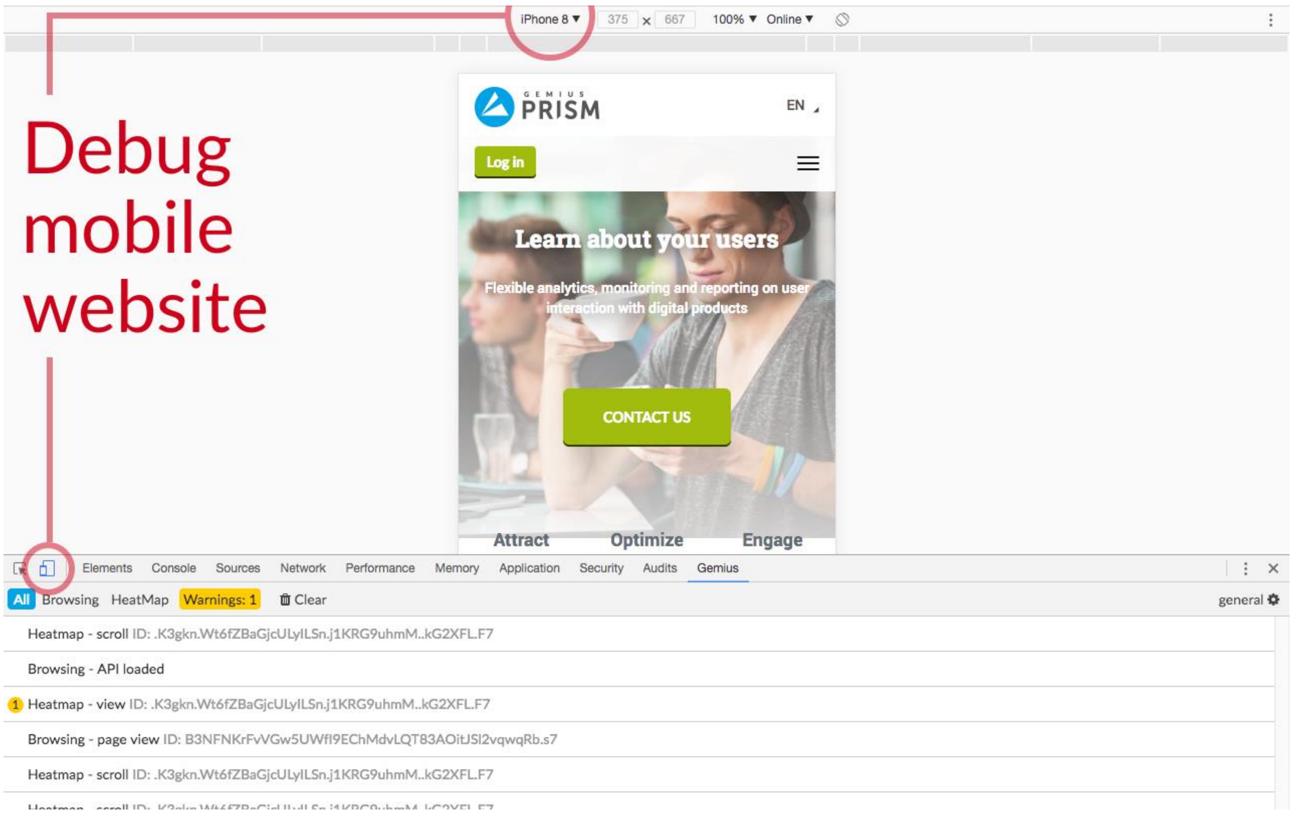


```
Heatmap - scroll Gemius ID .K3gkn.Wt6fZBaGjcULyILSn.j1KRG9uhmM..kG2XFL.F7
Browsing - API załadowane
Heatmap - view Gemius ID .K3gkn.Wt6fZBaGjcULyILSn.j1KRG9uhmM..kG2XFL.F7
Browsing - page view Gemius ID B3NFNKrFvVGw5UWfI9EChMdvLQT83AOitJSl2vqwqRb.s7
Heatmap - scroll Gemius ID .K3gkn.Wt6fZBaGjcULyILSn.j1KRG9uhmM..kG2XFL.F7
```

After Gemius Debugger is installed, go to Developer Tools (Menu > More Tools > Developer Tools), choose the Gemius tab, click general ⚙ and select options that match your project (you can also do it by going to More Tools > Extensions: chrome://extensions/).

Gemius Debugger can also be used to debug mobile versions of websites. In order to do so, click the Toggle device toolbar icon in the top-left corner and use the viewport controls to test your website on a specific device.

Debug mobile website



```
Heatmap - scroll ID: .K3gkn.Wt6fZBaGjcULyILSn.j1KRG9uhmM..kG2XFL.F7
Browsing - API loaded
1 Heatmap - view ID: .K3gkn.Wt6fZBaGjcULyILSn.j1KRG9uhmM..kG2XFL.F7
Browsing - page view ID: B3NFNKrFvVGw5UWfI9EChMdvLQT83AOitJSl2vqwqRb.s7
Heatmap - scroll ID: .K3gkn.Wt6fZBaGjcULyILSn.j1KRG9uhmM..kG2XFL.F7
```

GEMIUS

Gemius SA

contact@gemius.com

www.gemius.com