MEDIAPANEL

Pomiar treści audio i wideo na stronach internetowych

Implementacja w odtwarzaczach kontrolowanych za pomocą JavaScript

Lipiec 2023

MEDIAPANEL

MEDIAPANEL

Spis treści:

1. Informacje ogólne

Gemius oferuje rozwiązanie oparte na JavaScript dla celów pomiaru treści audio i wideo. Użyte funkcje umożliwiają Właścicielom treści uzyskanie informacji na temat wszystkich czynności i stanów zarejestrowanych w odtwarzaczu objętym badaniem. W celu uzyskania dokładnych danych o konsumpcji emitowanych treści, Właściciele powinni zaimplementować skrypty zliczające, które stanowią uzupełnienie funkcjonalności odtwarzacza i dostarczają do Gemius informacje zwrotne na temat ich wykorzystania. Moduł Stream gemiusPrism pozwala na przeprowadzanie pomiaru aktywności użytkowników korzystających z odtwarzaczy audio i wideo osadzonych na stronie lub wewnątrz aplikacji dla każdej z dostępnych platform.

Na potrzeby pomiaru treści audio i wideo w Badaniu Jednoźródłowym Mediapanel. Gemius zakłada dedykowane do tego celu projekty w gemiusPrism. Wydawcy, którzy posiadają moduł Stream w gemiusPrism mogą skorzystać z tych projektów na rzecz Badania Jednoźródłowego. W tym celu powinni zaktualizować swoje oskryptowanie zgodnie z poniższym dokumentem. Jeżeli Wydawca nie zdecyduje się na wprowadzenie zmian do aktualnego oskryptowania, Gemius założy konta dedykowane jedynie do Badania Jednoźródłowego a Wydawca przeprowadzi oskryptowanie treści audio i wideo zgodnie z dokumentem.

W celu przeprowadzenia prawidłowego pomiaru Gemius zakłada dwa rodzaje projektów:

- projekt dla playerów osadzonych na stronach internetowych
- projekt dla playerów umieszczonych w aplikacjach (mobilnych)

Schemat nazewnictwa projektów jest następujący:

- gsm2_wydawcaA_web
- gsm2_wydawcaA_aplikacje

Projekt dedykowany dla stron internetowych (Web) posiada jeden identyfikator skryptu, którego Wydawca używa na wszystkich stronach na których posiada odtwarzacze audio i wideo.

W niniejszym dokumencie rozróżniamy 3 rodzaje parametrów:

Wymagane – jako parametry potrzebne do prezentacji danych w wynikach Badania Jednoźródłowego w uzgodnionych przecięciach i widokach:

- Lista Brandów Wydawcy (widok w drzewie)
- Content Type: Audio, Video (filtr)
- Channel Space Type: Commercial, Editorial (filtr)
- Channel Type: Website, Application, Radio, TV (filtr)
- TransmissionType: On-demand, Broadcast (filtr)
  W ramach “Broadcast” Gemius nadaje kanałom mediowym wartości: Music station, Radio station – online, Radio station – licenced, TV station – online, TV station – licenced.

Rekomendowane – jako parametry które mogą posłużyć do prezentacji danych w widokach prywatnych w gemiusPrism, nie są konieczne do prezentacji danych w wynikach Badania Jednoźródłowego.

- Kategoryzacja według rodzaju gatunku materiału (opis w dokumencie *Typologia Pozycji Programowych*, wynika z wartości przekazanych w parametrze *programGenre*)
- Kategoryzacja tematyczna (opis w dokumencie *Typologia Pozycji Programowych*, wynika z wartości przekazanych w parametrze *programThematicCategory)*
- Kategoryzacja funkcjonalna (kategoryzacja nadawana przez Gemius, opis w dokumencie *Typologia Pozycji Programowych*)
- Widoki dla serii oraz konkretnych pozycji programowych (wynikają z wartości przekazanych w parametrach: *programName i series*)

Opcjonalne - które nie są konieczne do prezentacji danych w wynikach Badania Jednoźródłowego, lecz mogą być wykorzystane w analizie danych w gemiusPrism na własne potrzeby Wydawcy.

Instancja pomiaru jest inicjowana poprzez wywołanie funkcji ***GemiusPlayer*** podczas ładowania się komponentów odtwarzacza. **gemiusPrism** obsługuje pomiar treści definiowanych jako sesja w odtwarzaczu zawierającym jeden lub więcej typów aktywności, w tym:

<table>
<tr><th colspan="10">Elementy programu</th></tr>
<tr><td colspan="2">blok reklamowy pre-roll</td><td>program net</td><td colspan="2">blok reklamowy mid-roll</td><td>program netto</td><td colspan="2">blok reklamowy mid-roll</td><td>program net</td><td colspan="2">blok reklamowy post-roll</td></tr>
<tr><td>spot reklamowy</td><td>spot reklamowy</td><td>pierwsza część</td><td>spot reklamowy</td><td>spot reklamowy</td><td>druga część</td><td>spot reklamowy</td><td>spot reklamowy</td><td>trzecia część</td><td>spot reklamowy</td><td>spot reklamowy</td></tr>
</table>

- Odtwarzacz jest ładowany na stronie internetowej i czeka na pierwszą czynność użytkownika.
- Blok reklamowy pre-roll – emitowane są spoty reklamowe, po czym wyświetlona zostaje pierwsza część materiału – czyli emisja części materiału następująca po odtworzeniu bloku reklamowego pre-roll i poprzedzająca odtworzenie bloku reklamowego mid-roll (lub do końca materiału, jeżeli nie ma zaplanowanych bloków reklamowych mid-roll).
- Emisja bloku reklamowego mid-roll – następuje emisja dodatkowych spotów reklamowych, po czym druga część i kolejne części materiału są odtwarzane po kolei po wyświetleniu bloków reklamowych mid-roll (powyższy przykład ilustruje trzy części materiału przerywanego dwoma blokami reklamowymi mid-roll).
- Koniec odtwarzania materiału, po którym może nastąpić emisja bloku reklamowego post-roll.

Oprócz opisanych stanów mierzona sesja może obejmować czynności użytkownika i stan połączenia na linii użytkownik-serwer, które prowadzą do takich akcji jak: buforowanie, pauza, szukanie, przejście do innego punktu odtwarzania, przejście do kolejnego materiału, przejście do poprzedniego materiału, zatrzymanie materiału, zakończenie materiału lub zamknięcie odtwarzacza. Pomiar może również zawierać informacje na temat zmian rozdzielczości odtwarzacza, jakości i głośności.

Obowiązkowo do skryptu Wydawca musi przekazać informację o tym gdzie zlokalizowany jest player (metoda setVideoObject). Dzięki temu wykonywane jest automatyczne pobieranie informacji o widoczności i rozmiarze playera.

2. Implementacja pomiaru

2.1. Pobieranie biblioteki

Dane zbierane w trakcie przeprowadzania pomiaru treści audio i wideo przesyłane są do dedykowanego serwera Gemius (tzw. **hitcollector**). W przypadku odtwarzaczy osadzonych na stronie, ścieżka do skryptu głównego (gplayer.js) znajdującego się na serwerze **hitcollector** musi zostać umieszczona w kodzie źródłowym strony między znacznikami <head> </head>, przed odtwarzaczem. Skrypt główny może zostać załadowany synchronicznie:

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

lub asynchronicznie:

```
<script type="text/javascript">
function gemius_pending(i) { window[i] = window[i] || function() {var x = window[i+'_pdata'] = window[i+'_pdata'] || [];
x[x.length]=arguments;};};gemius_pending('gemius_init');
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>

```

PREFIX oznaczony w powyższym przykładzie powinien zostać zamieniony na krótki ciąg liter identyfikujący serwer Gemius, który zbiera dane. Prefix można znaleźć w skrypcie zliczającym w interfejsie **gemiusPrism** (w Ustawienia / Skrypty / Instalacja podstawowa) lub otrzymać od działu Mediapanel.

Przykładowy prefix: ‘gapl’

W przypadku implementacji asynchronicznej istnieje możliwość, że odtwarzacz i materiał załadują się przed załadowaniem skryptu zliczającego. W takiej sytuacji wszystkie stany odtwarzacza i czynności użytkownika zostają zmierzone w momencie wystąpienia i przesłane do serwera **hitcollector** równocześnie, zaraz po załadowaniu się skryptu głównego.

Aby zapewnić prawidłowy pomiar viewability odtwarzacza osadzonego w ramce HTML, skrypt należy umieścić zarówno w osadzanej ramce, jak i stronie, na której została załadowana ramka. Jeśli jest kilka poziomów zagnieżdżenia, skrypt powinien zostać umieszczony na każdym z poziomów.

Jeżeli witryna korzysta z języka xhtml, należy zawrzeć dodatkowe linijki kodu po otwarciu tagu <script>:

```
<script type="text/javascript">
<!--//--><![CDATA[//><!—
```

oraz przed zamknięciem:

```
//--><!]]>
</script>
```

2.2. Tworzenie obiektu do pomiaru

Po utworzeniu komponentu odtwarzacza w przeglądarce, skrypt zliczający jest inicjowany poprzez wywołanie funkcji ***GemiusPlayer*** i utworzenie obiektu, który będzie wykorzystywany do przekazywania wszystkich informacji na temat danego odtwarzacza. W przypadku wyświetlania wielu instancji odtwarzacza, dla każdej z nich niezbędne jest utworzenie osobnego obiektu odtwarzacza.

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

Opis argumentów		
playerID[*String*]	Wymagany	Identyfikator odtwarzacza, na podstawie którego mogą zostać zagregowane wyniki z różnych instancji odtwarzacza (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,). Parametr ten może przyjmować dowolne wartości. Wartość parametru będzie automatycznie nazwą brandu prezentowaną w wynikach badania, należy więc zwracać szczególną uwagę na to, aby nadawana była zawsze w sposób spójny i konsekwentny (np. jeśli chodzi o używanie bądź pomijanie polskich znaków)

		Identyfikatorem jest nazwa brandu Playera, np. WP Pilot, OpenFM, ipla.
gemiusID[*String*]	Wymagany	Unikalna sekwencja znaków wskazująca projekt albo sekcję projektu na którą zostanie skierowany zmierzony ruch. W zależności od ustaleń w umowie, identyfikator można uzyskać na kilka sposobów: • z interfejsu **gemiusPrism** / Ustawienia / Skrypty / Instalacja podstawowa (szczegóły w kolejnym rozdziale) • kontaktując się z działem Mediapanel Przykładowy identyfikator: 'zZCQBO8qo8FpfdK3lsjyMXZHjwzBsFt1vQ9YPinPCAX.M7'. Zmienna 'gemius_identifier' musi zostać zdefiniowana zanim zostanie załadowany Skrypt Główny. Jeśli ten warunek nie zostanie spełniony, dane nie będą zbierane poprawnie.
additionalParameters[*Dictionary*]	Wymagany	Słownik zawierający dodatkowe parametry opisujące odtwarzacz i jego domyślne ustawienia.
Opis dodatkowych parametrów		
currentDomain[*String*]	Wymagany *(w przypadku playera w ramce)*	Zdefiniowana nazwa domeny, w której odtwarzacz ma być osadzony. Przesyłany w przypadku osadzenia w wielowarstwowym elemencie frame lub iframe dla potrzeb prezentacji danych w drzewku syndykacyjnym.
volume[*Number*]	Wymagany	Wstępnie zdefiniowana wartość procentowa poziomu głośności odtwarzacza (zakres od 0 do 100). Dla opcji „wycisz” wartość powinna wynosić -1.
resolution[*String*]	Opcjonalny *(Wymagamy przekazania obiektu playera i tym samym NIE wymagamy definiowania ręcznie)*	Wstępnie zdefiniowana rozdzielczość odtwarzacza (np. 1024x768). Rzeczywisty rozmiar okna odtwarzacza (wielkość domyślna). Jeżeli lokalizacja playera zostanie przekazana do skryptu głównego za pomocą metody *setVideoObject*, rozmiar jest przekazywany automatycznie i parametr *resolution* nie musi być ręcznie definiowany. (Więcej informacji w sekcji Przekazywanie obiektu odtwarzacza.)

Dodatkowe parametry przekazywane w trakcie tworzenia obiektu dla odtwarzacza umożliwiają przekazanie informacji o domyślnych ustawieniach odtwarzacza. Następnie mogą one być modyfikowane przez czynności użytkownika lub predefiniowane w związku z emisją konkretnego materiału.

2.3. Przekazywanie obiektu odtwarzacza

Obowiązkowe jest wskazanie przekazanie do skryptu informacji o tym, gdzie zlokalizowany jest odtwarzacz. Dzięki temu możliwe będzie automatyczne pobieranie rozmiaru odtwarzacza i informacji o jego widoczności. Należy przy tym pamiętać, że aby możliwe było przekazanie jakiejkolwiek informacji na temat obiektu, najpierw musi zostać utworzony obiekt odtwarzacza.

W celu przekazania obiektu wideo/audio do skryptu należy skorzystać z metody setVideoObject, która jako jedyny parametr przyjmuje element DOM. Jeśli utworzyliśmy obiekt player (np. var player = new GemiusPlayer(.....); to przekazanie informacji o obiekcie wideo/audio może odbyć się w następujący sposób:

```
1 player.setVideoObject(document.getElementById("myplayer"));
```

Kod ten działa przy założeniu, że elementowi audio/wideo, który chcemy przekazać, nadaliśmy identyfikator – w tym przypadku o wartości *„myplayer”* np. deklarując go w następujący sposób:

```
1 <video id="myplayer" .... >
```

Jeśli element wideo/audio na stronie zostanie utworzony ponownie, ale kontynuowane jest w nim odtwarzanie tego samego materiału, to należy uaktualnić informację o lokalizacji elementu wideo/audio ponownie wywołując funkcję ***setVideoObject***.

Jeżeli nowy obiekt będzie posiadał taką samą wartość atrybutu id, również należy ponownie wykonać rejestrację, ponieważ do metody przekazywany jest element, a nie id elementu. Do skryptu należy przekazać obiekt wideo/ audio a nie element nadrzędny nad tym obiektem.

W przypadku skryptowania odtwarzaczy w ramkach należy pamiętać, że skrypt należy załączyć także na stronie, na której dana ramka została załadowana. Jeśli ta strona jest także umieszczona w ramce to trzeba skrypt załadować również na stronie, na której ta ramka jest umieszczona itd. Wklejka na stronie, która zawiera oskryptowaną ramkę jest taka sama jak ta przedstawiona wyżej. Osoba implementująca skrypt musi więc załadować skrypt (dodać powyższą wklejkę) nie tylko na stronie z playerem, ale także na stronach, które mają ramki z oskryptowanym playerem.

2.4. Uwzględnianie informacji na temat odtwarzanego materiału i reklam

2.4.1. Program

Po załadowaniu programu do odtwarzacza, jego opis powinien zostać przekazany do **gemiusPrism**. Jest to warunek konieczny niezależnie od tego, czy emisja materiału rozpoczęła się (czy to w wyniku działania

użytkownika, czy poprzez uruchomienie się funkcji autoodtwarzania (autoplay) – odtwarzacz może czekać na podjęcie działania przez użytkownika). Opis jest przekazywany poprzez wykonanie funkcji ***newProgram*** w obiekcie odtwarzacza.

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

Opis argumentów		
programID[*String*]	Wymagany	Unikalny identyfikator programu (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,). Parametr ten umożliwia rozróżnianie programów dlatego nie należy go powtarzać (wyłączając broadcast). Wartość definiowana przez użytkownika.
additionalParameters[*Dictionary*]	Wymagany	Słownik zawierający dodatkowe parametry opisujące materiał załadowany do odtwarzacza i jego ustawienia.
Opis dodatkowych parametrów		
programName[*String*]	Wymagany	Tytuł materiału (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,). Zalecamy aby zdefiniować ten parametr i nie polegać wyłącznie na rozróżnieniu na podstawie parametru programID.
programDuration[*Number*]	Wymagany	Długość materiału (bez reklam) w sekundach. Dla odtworzeń materiałów transmitowanych na żywo bez możliwości określenia harmonogramu programów w trakcie transmisji (np. bezpośrednia transmisja na żywo kanału TV), wartość powinna wynosić -1.
programType[*String*]	Wymagany	Typ materiału. Dozwolone wartości: „audio”, „video”.
transmissionType[*Number*]	Wymagany, gdy broadcast Opcjonalny, gdy on demand	Rodzaj transmisji: 1 = On demand. Materiał wyświetlany na żądanie użytkownika. 2 = Broadcast / Live program / Time shift viewing TSV. Materiał wyświetlany na żywo lub

		zgodnie z programem wydawcy (np. transmisja kanału TV). W przypadku *transmissionType = 2 (broadcast)*, należy zdefiniować również parametry *transmissionChannel* oraz *transmissionStartTime*. Jeśli parametr *transmissionType* nie zostanie zdefiniowany, zastosowana zostanie wartość „On demand”.
transmissionChannel[*String*]	Wymagany, gdy broadcast Opcjonalny, gdy on demand	Nazwa kanału TV/radio (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,). Należy zdefiniować ten parametr w przypadku *transmissionType = 2 (broadcast)*. Parametr ten może przyjmować dowolne wartości. Wartość parametru będzie automatycznie nazwą kanału prezentowaną w wynikach badania, należy więc zwracać szczególną uwagę na to, aby nadawana była zawsze w sposób spójny i konsekwentny (np. jeśli chodzi o używanie bądź pomijanie polskich znaków) np. Polsat Cafe zamiast polsatcafe lub RMF FM zamiast rmffm.
transmissionStartTime[*String*]	Wymagany, gdy broadcast Opcjonalny, gdy on demand	Czas rozpoczęcia transmisji. Należy zdefiniować ten parametr w przypadku *transmissionType = 2 (broadcast)*. Wymagany format: timestamp – liczba sekund od 00:00:00 UTC 1 stycznia 1970. Więcej informacji w sekcji Skryptowanie live streaming (broadcast).
programGenre[*Number*]	Rekomendowany	Gatunek materiału. Dozwolone wartości: 1 = Program na żywo (gdy *transmissionType = 2*) 2 = Film, Słuchowisko radiowe 3 = Serial telewizyjny, Serial radiowy, Vlog 4 = Program telewizyjny, Audycja radiowa, Podcast, Program, Audycja online 5 = Muzyka, Pasmo teledysków, Playlista.

		6 = Fragment programu (wycięty z pozycji programowej)
programThematicCategory[*String*]	Rekomendowany	Kategoria tematyczna materiału. Lista wartości opisana jest w Dokumencie *Typologia Pozycji Programowych.*
series[*String*]	Rekomendowany	Nazwa serii, np. Gra o Tron (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,).
programSeason[*String*]	Opcjonalny	Nazwa sezonu – zestawu odcinków wyświetlanych pod tym samym tytułem (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,). Np.: „season 1" lub „Season 2014-2015".
programPartialName[*String*]	Opcjonalny	Tematyka fragmentu programu (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,). Np.: jeśli *programName = „Poland – France UEFA European Championship"*, to *programPartialName = „Highlights"*.
programProducer[*String*]	Opcjonalny	Nazwa producenta programu (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,).
typology[*String*]	Opcjonalny	Hierarchiczna kategoryzacja treści zastosowana przez Wydawcę: Sport/Football, Movie/Class B).
premiereDate[*String*]	Opcjonalny	Data pierwszej publikacji programu na stronie klienta (w formacie YYYYMMDD).
externalPremiereDate[*String*]	Opcjonalny	Data pierwszej publikacji poza stroną klienta. Może być wykorzystywany np. do analizy serialu TV przed premierą lub do analizy ramowej rynku filmowego przed premierą; pozwala określić, po jakim czasie od premiery w kinie film pojawił się w Internecie (w formacie YYYYMMDD).
quality[*String*]	Opcjonalny	Wstępnie zdefiniowana (np. 1920x1080) jakość załadowanego materiału. Na późniejszym etapie może być dostosowywana przez użytkownika.

resolution[*String*]	Opcjonalny *(Wymagamy przekazania obiektu playera i tym samym NIE wymagamy definiowania ręcznie)*	Wstępnie zdefiniowana rozdzielczość załadowanego materiału, która może zmieniać domyślne lub zdefiniowane przez użytkownika ustawienia okna odtwarzacza. Jeżeli lokalizacja odtwarzacza zostanie przekazana za pomocą metody ***setVideoObject***, to rozmiar jest przekazywany automatycznie i parametr ten nie musi być ręcznie definiowany. Więcej informacji w sekcji Przekazywanie obiektu odtwarzacza.
volume[*Number*]	Wymagany	Wstępnie zdefiniowany poziom głośności załadowanego materiału, który może zmieniać domyślne lub zdefiniowane przez użytkownika ustawienia głośności.
customAttributes	Opcjonalny	Dodatkowe atrybuty materiału. Ich nazwy i wartości są definiowane przez uczestnika badania i powinny być różne od nazw i wartości parametrów oficjalnych. W razie potrzeby wartość dodatkowego parametru może zostać zmieniona w dowolnym zdarzeniu przesyłanym poprzez funkcję programEvent.

Fakt zdefiniowania (utworzenia) odtwarzacza zostaje zarejestrowany i może zostać wyodrębniony w danych surowych. Jednakże wskaźniki prezentowane w ***gemiusPrism*** opierają się na danych przesłanych po kontakcie użytkownika z materiałem lub reklamą (warunkiem jest rozpoczęcie emisji reklamy lub pierwszej części programu). W momencie rozpoczęcia odtwarzania, inicjowany jest pomiar programu brutto. Oznacza to, że użytkownik chce obejrzeć dany materiał lub jego część. Z uwagi jednak na to, że użytkownik może zrezygnować z oglądania w czasie emisji bloku reklamowego pre-roll, a także ze względu na możliwość wystąpienia problemów technicznych lub sieciowych, mierzona sesja może się zakończyć jeszcze przed obejrzeniem samego materiału (netto) przez użytkownika.

2.4.2. Reklama

Dedykowane funkcje ***programEvent*** i ***adEvent*** służą do przeprowadzania pomiaru stanów i czynności podejmowanych przez użytkownika w czasie trwania sesji niezależnie od całościowej struktury programu: liczby i rodzaju przerw reklamowych (oraz spotów reklamowych w nich zawartych), a także części materiału, które zostały obejrzane i z których składa się struktura. Jeżeli przed emisją treści audio lub wideo wyświetlany jest blok reklamowy (pre-roll), informacje o reklamie powinny być przekazane poprzez funkcję ***newAd***.

Metoda ta powinna zostać wywołana tuż przed rozpoczęciem odtwarzania reklamy w odtwarzaczu:

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

Opis argumentów		
adID[*String*]	Wymagany	Unikalny identyfikator reklamy lub całego brake'u reklamowego. (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,). Zachęcamy do przekazywania unikalnych identyfikatorów per reklama. Jeżeli Wydawca przekazuje identyfikator całego break'u konieczne jest przekazanie wartości „break" w parametrze AdType.
additionalParameters[*Dictionary*]	Wymagany	Słownik zawierający dodatkowe parametry opisujące reklamę i jej ustawienia.
Opis parametrów dodatkowych		
adName[*String*]	Rekomendowany	Tytuł reklamy (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,). Zaleca się, aby zdefiniować ten parametr i nie polegać wyłącznie na rozróżnieniu na podstawie parametru *adID*.
adDuration[*Number*]	Rekomendowany	Długość reklamy w sekundach; liczba całkowita.
adType[*String*]	Opcjonalny *(Wymagany, jeżeli adID dotyczy całego break'u i gdy transmissionType =1; on demand)*	Typ reklamy. Sugerowane wartości: „break", „promo", „spot", „sponsor". Wymagana jest wartość „break", jeżeli w adID Wydawca przekazuje jeden identyfikator.
campaignClassification[*String*]	Opcjonalny	Hierarchiczna kategoryzacja kampanii, z uwzględnieniem nazwy kampanii, gatunku, producenta, oddzielonych ukośnikiem (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,).
adFormat[Number]	Opcjonalny	Format reklamy. Dozwolone wartości: 1 - video 2 - audio Jeśli parametr *adFormat* nie zostanie zdefiniowany, wysłana zostanie wartość „1".

quality[String]	Opcjonalny	Wstępnie zdefiniowana jakość (np. 1920x1080) załadowanej reklamy. Na późniejszym etapie może być dostosowywany przez użytkownika.
resolution[String]	Opcjonalny *(Wymagamy przekazania obiektu playera i tym samym NIE wymagamy definiowania ręcznie)*	Wstępnie zdefiniowana rozdzielczość załadowanej reklamy, która może zmieniać domyślne lub zdefiniowane przez użytkownika ustawienia okna odtwarzacza. Jeżeli lokalizacja odtwarzacza zostanie przekazana za pomocą metody ***setVideoObject*** to rozmiar jest przekazywany automatycznie i parametr ten nie musi być ręcznie definiowany. Więcej informacji w sekcji Przekazywanie obiektu odtwarzacza.
volume[*String*]	Wymagany	Wstępnie zdefiniowany poziom głośności załadowanej reklamy, który może zmieniać domyślne lub zdefiniowane przez użytkownika ustawienia głośności.
customAttributes	Opcjonalny	Dodatkowe atrybuty reklamy. Ich nazwy i wartości są definiowane przez uczestnika badania i powinny być różne od nazw i wartości parametrów oficjalnych. W razie potrzeby wartość dodatkowego parametru może zostać zmieniona w dowolnym zdarzeniu przesyłanym poprzez funkcję adEvent.

Kiedy informacja o załadowanej treści jest gotowa, funkcje przekazujące informacje na temat czynności podejmowanych przez użytkowników i na temat stanów odtwarzacza mają dostateczny zasób danych wejściowych.

2.5. Pomiar aktywności użytkowników i zmian stanu odtwarzacza

Po zdefiniowaniu obiektu odtwarzacza oraz opisaniu reklamy i materiału, informacje o aktywności użytkowników i zmianach stanu odtwarzacza mogą zacząć być przekazywane. Informacje te są przekazywane poprzez wywołanie funkcji ***programEvent*** i ***adEvent***.

2.5.1. Play

Zainicjowanie odtwarzania materiału lub reklamy poprzedzone jest przesłaniem informacji o zdarzeniu ***play***. Może być to wynikiem zarówno ustawienia auto-play (automatyczne odtwarzanie), jak i aktywności użytkownika.

W zależności od tego, czy blok reklamowy pre-roll występuje w materiale czy nie, emisja może się rozpocząć jako rezultat zarejestrowania opcji *play* w wyniku wywołania funkcji ***adEvent*** lub ***programEvent***.

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

Opis argumentów		
programID[*String*]	Wymagany	Unikalny identyfikator programu (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,), deklarowany we wcześniejszym wywołaniu funkcji ***newProgram***.
adID[*String*]	Wymagany	Unikalny identyfikator reklamy (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,), deklarowany we wcześniejszym wywołaniu funkcji ***newAd***.
offset[*Number*]	Wymagany	Przesunięcie czasu w sekundach – moment w treści programu, w którym nastąpiło dane zdarzenie. Wartość powinna być taka sama dla wszystkich zdarzeń w ramach tego samego bloku reklamowego. Offset dla zdarzenia ***play*** reklamy służy do określenia typu reklamy: pre, mid, post, live lub unknown. Użycie parametru offset dla materiałów typu broadcast (live) zostało szczegółowo opisane w sekcji Skryptowanie live streaming (broadcast).
additionalParameters[*Dictionary*]	Wymagany	Słownik zawierający dodatkowe parametry definiujące zdarzenie *play* dla danej reklamy.
Opis parametrów dodatkowych		
autoPlay[*Bool*]	Wymagany	Informacja o trybie, w którym rozpoczyna się odtwarzanie reklamy – autoplay czy ręczne uruchomienie. Dozwolone wartości: „true”, „false”.
adPosition[*Number*]	Opcjonalny	Numer porządkowy reklamy w bloku reklamowym.
breakSize[*Number*]	Opcjonalny	Liczba reklam w bloku reklamowym, do którego wlicza się dana reklama.
resolution[*String*]	Opcjonalny *(Wymagamy przekazania obiektu playera i*	Rozdzielczość, na którą odtwarzacz przełączył się automatycznie w momencie rozpoczęcia odtwarzania reklamy. Jeżeli lokalizacja odtwarzacza zostanie przekazana za pomocą metody ***setVideoObject*** to rozmiar jest przekazywany automatycznie i parametr ten

	tym samym NIE wymagamy definiowania ręcznie)	nie musi być ręcznie definiowany. Więcej informacji w sekcji Przekazywanie obiektu odtwarzacza.
volume[Number]	Wymagany	Poziom głośności, na którą odtwarzacz przełączył się automatycznie w momencie rozpoczęcia odtwarzania reklamy.
adDuration[Number]	Opcjonalny	Długość reklamy w sekundach. W razie potrzeby wartość parametru przekazywana w funkcji newAd może zostać zmieniona w zdarzeniu przesyłanym poprzez funkcję adEvent np. w zdarzeniu play. Nowa wartość parametru będzie widoczna w kolejnych hitach tej samej sesji.
customAttributes	Opcjonalny	Dodatkowe atrybuty reklamy. Ich nazwy i wartości są definiowane przez uczestnika badania i powinny być różne od nazw i wartości parametrów oficjalnych. W razie potrzeby wartość parametru przekazywana w funkcji newAd może zostać zmieniona w zdarzeniu przesyłanym poprzez funkcję adEvent np. w zdarzeniu play. Nowa wartość parametru będzie widoczna w kolejnych hitach tej samej sesji.

Metoda ta musi zostać wywołana tuż przed rozpoczęciem odtwarzania reklamy. Każda reklama musi zostać uprzednio zarejestrowana poprzez wywołanie funkcji ***newAd***. Każda reklama w bloku reklamowym, lub cały blok reklamowy muszą zostać zgłoszone do systemu (przesłane poprzez wywołanie funkcji ***newAd***) zanim rozpoczęcie odtwarzania zostanie zarejestrowane (przesłane poprzez wywołanie funkcji ***adEvent***). W podobny sposób informacja o zdarzeniu *play* jest wysyłana z wykorzystaniem funkcji ***programEvent***.

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

Opis argumentów		
programID[*String*]	Wymagany	Unikalny identyfikator programu (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,), deklarowany we wcześniejszym wywołaniu funkcji ***newProgram***.
offset[*Number*]	Wymagany	Przesunięcie czasu w sekundach – moment w treści programu, w którym nastąpiło dane zdarzenie.

		Użycie parametru offset dla materiałów typu broadcast (live) zostało szczegółowo opisane w sekcji Skryptowanie live streaming (broadcast).
additionalParameters[*Dictionary*]	Wymagany	Słownik zawierający dodatkowe parametry definiujące zdarzenie *play* dla danego materiału.
Opis parametrów dodatkowych		
autoPlay[*Bool*]	Wymagany	Informacja o trybie, w którym rozpoczyna się odtwarzanie materiału – autoplay czy ręczne uruchomienie. Dozwolone wartości: „true", „false". W przypadku intencjonalnie włączonej playlisty, wartość „false" powinna być przekazana przy kontynuowaniu każdego kolejnego materiału.
partID[*Number*]	Opcjonalny	Numer porządkowy części programu (od 1 do n), określający jego pozycję w całości programu. Jeżeli konfiguracja usługi umożliwia odtwarzanie użytkownikowi programu od np. drugiej części, liczba ta wynosi 2 – jest to pozycja bezwzględna wyświetlanej części materiału.
resolution[*String*]	Opcjonalny *(Wymagamy przekazania obiektu playera i tym samym NIE wymagamy definiowania ręcznie)*	Rozdzielczość, na którą odtwarzacz przełączył się automatycznie w momencie rozpoczęcia odtwarzania materiału. Jeżeli lokalizacja playera zostanie przekazana za pomocą metody ***setVideoObject,*** to rozmiar jest przekazywany automatycznie i parametr ten nie musi być ręcznie definiowany. Więcej informacji w sekcji Przekazywanie obiektu odtwarzacza.
volume[*Number*]	Wymagany	Poziom głośności, na którą odtwarzacz przełączył się automatycznie w momencie rozpoczęcia odtwarzania materiału.
programDuration[Number]	Opcjonalny	Długość materiału w sekundach. W razie potrzeby wartość parametru przekazywana w funkcji newProgram może zostać zmieniona w zdarzeniu przesyłanym poprzez funkcję programEvent np. w zdarzeniu play. Nowa wartość parametru będzie widoczna w kolejnych hitach tej samej sesji.

customAttributes	Opcjonalny	Dodatkowe atrybuty materiału. Ich nazwy i wartości są definiowane przez uczestnika badania i powinny być różne od nazw i wartości parametrów oficjalnych. W razie potrzeby wartość parametru przekazywana w funkcji newProgram może zostać zmieniona w zdarzeniu przesyłanym poprzez funkcję programEvent np. w zdarzeniu play. Nowa wartość parametru będzie widoczna w kolejnych hitach tej samej sesji.

Podczas emisji całkowitych elementów programu (reklam i części materiału netto) ma miejsce szereg działań podejmowanych przez użytkownika i stanów dotyczących pracy odtwarzacza. Zastosowanie funkcji ***programEvent*** i ***adEvent*** w obiekcie odtwarzacza pozwala je mierzyć. W przypadku siedmiu zdarzeń informujących o przerwie w emisji reklamy lub materiału uwzględnienie dodatkowych parametrów nie jest wymagane.

2.5.2. Stop

Użytkownik nacisnął przycisk *stop* lub wykonał akcję mającą ten sam skutek. Zdarzenie to oznacza, że odtwarzanie materiału lub reklamy zostało zatrzymane i wskaźnik na pasku postępu przesunął się do początku materiału.

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

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

2.5.3. Pause

Użytkownik nacisnął przycisk *pause* lub wykonał akcję mającą ten sam skutek. Zdarzenie to oznacza, że odtwarzanie materiału lub reklamy zostało tymczasowo wstrzymane i wskaźnik na pasku postępu pozostaje w pozycji, w której nastąpiło zdarzenie.

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

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

2.5.4. Buffering

Użytkownik nie wykonał żadnej akcji, ale odtwarzacz zakończył odtwarzanie załadowanego fragmentu materiału lub reklamy i próbuje załadować kolejny fragment z serwera przed wznowieniem odtwarzania.

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

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

2.5.5. Break

Użytkownik nie wykonał żadnej akcji, ale odtwarzacz wstrzymał odtwarzanie załadowanego materiału, aby wyświetlić blok reklamowy, po wyświetleniu którego odtwarzanie materiału zostaje wznowione (w przypadku post-roll odtwarzana jest następna część lub program jest wznawiany).

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

2.5.6. Seek

Użytkownik przeszedł do przypadkowego punktu odtwarzania materiału lub reklamy, tj. kliknął na pasek postępu lub wykonał akcję mającą ten sam skutek, próbując pominąć część materiału lub powrócić do wcześniejszej części materiału.

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

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

2.5.7. Complete

Użytkownik obejrzał ostatnią sekundę materiału lub reklamy.

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

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

2.5.8. Close

Zakończenie odtwarzania materiału lub reklamy poprzez zamknięcie aktywnego okna odtwarzacza przez użytkownika lub system. Zdarzenie *close* należy zaraportować tylko, jeśli w trakcie trwania materiału (przed jego zakończeniem) został on zmieniony na inny lub okno odtwarzacza zostało zamknięte.

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

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

Inna grupę stanowią zdarzenia i stany odtwarzacza, które przekazują dodatkowe informacje na temat aktywności użytkownika i jego interakcji z odtwarzaczem. Informacje te dotyczą interakcji z listami odtwarzania oraz manualnych zmian poziomu głośności, rozmiaru okna i ustawień dotyczących jakości w odtwarzaczu.

2.5.9. Skip

Użytkownik nacisnął przycisk *skip* lub wykonał akcję mającą ten sam skutek, w wyniku czego odtwarzanie przesunęło się do kolejnej części materiału lub punktu na pasku postępu.

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

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

2.5.10. Next

Użytkownik nacisnął przycisk *next* lub wykonał akcję mającą ten sam skutek, czego wynikiem jest przejście z aktualnie odtwarzanego materiału do odtwarzania nowego materiału (kolejnego na liście).

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

2.5.11. Previous

Użytkownik nacisnął przycisk *previous* lub wykonał akcję mającą ten sam skutek, czego wynikiem jest przejście z aktualnie odtwarzanego materiału do odtwarzania nowego materiału (poprzedniego na liście).

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

Opis argumentów		
programID[*String*]	Wymagany	Unikalny identyfikator programu programu (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,), deklarowany we wcześniejszym wywołaniu funkcji ***newProgram***.
offset[*Number*]	Wymagany	Przesunięcie czasu w sekundach – moment w treści **programu**, w którym nastąpiło dane zdarzenie. Użycie parametru offset dla materiałów typu broadcast (live) zostało szczegółowo opisane w sekcji Skryptowanie live streaming (broadcast).
additionalParameters[*Dictionary*]	Opcjonalny	Słownik zawierający dodatkowe parametry definiujące zdarzenie *prev* dla danego materiału
Opis parametrów dodatkowych		
listID[*Number*]	Opcjonalny	Unikalny identyfikator listy, której elementy zostały użyte do poruszania się między punktami treści.

2.5.12. Resolution change (opcjonalnie)

Użytkownik zmienił rozdzielczość odtwarzacza lub wykonał akcję mającą ten sam skutek podczas emisji reklamy lub materiału. Jeżeli lokalizacja playera zostanie przekazana za pomocą metody ***setVideoObject***, to rozmiar jest przekazywany automatycznie i parametr ten nie musi być ręcznie definiowany. Więcej informacji w sekcji Przekazywanie obiektu odtwarzacza.

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

Opis argumentów		
programID[*String*]	Wymagany	Unikalny identyfikator programu (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,), deklarowany we wcześniejszym wywołaniu funkcji ***newProgram***.
adID[*String*]	Wymagany	Unikalny identyfikator reklamy (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,), deklarowany we wcześniejszym wywołaniu funkcji ***newAd***.
offset[*Number*]	Wymagany	Przesunięcie czasu w sekundach – moment w treści **programu**, w którym nastąpiło dane zdarzenie. Wartość

		powinna być taka sama dla wszystkich zdarzeń w ramach tego samego bloku reklamowego. Użycie parametru offset dla materiałów typu broadcast (live) zostało szczegółowo opisane w sekcji Skryptowanie live streaming (broadcast).
additionalParameters[*Dictionary*]	Wymagany	Słownik zawierający dodatkowe parametry definiujące zdarzenie *chngRes* dla danej reklamy.
Opis parametrów dodatkowych		
resolution[*String*]	Opcjonalny	Wstępnie zdefiniowana rozdzielczość odtwarzacza (np. 1024x768). Rzeczywisty rozmiar okna odtwarzacza (wielkość domyślna).

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

Opis argumentów		
programID[*String*]	Wymagany	Unikalny identyfikator programu (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,), deklarowany we wcześniejszym wywołaniu funkcji ***newProgram***.
offset[*Number*]	Wymagany	Przesunięcie czasu odtwarzania **programu** w sekundach. Moment, w którym nastąpiło dane zdarzenie. Użycie parametru offset dla materiałów typu broadcast (live) zostało szczegółowo opisane w sekcji Skryptowanie live streaming (broadcast).
additionalParameters[*Dictionary*]	Wymagany	Słownik zawierający dodatkowe parametry definiujące zdarzenie *chngRes* dla danego materiału
Opis parametrów dodatkowych		
resolution[*String*]	Opcjonalny	Rozdzielczość odtwarzacza (np. 1024x768) zdefiniowana przed użytkownika. Rzeczywisty rozmiar okna odtwarzacza.

2.5.13. Volume change

Użytkownik użył suwaka zmiany głośności, przycisku *wycisz* (*mute*) lub wykonał akcję mająca ten sam skutek podczas regulacji głośności. Zdarzenie opcjonalne.

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

Opis argumentów		
programID[*String*]	Wymagany	Unikalny identyfikator programu (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,), deklarowany we wcześniejszym wywołaniu funkcji ***newProgram***.
adID[*String*]	Wymagany	Unikalny identyfikator reklamy (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,), deklarowany we wcześniejszym wywołaniu funkcji ***newAd***.
offset[*Number*]	Wymagany	Przesunięcie czasu w sekundach – moment w treści programu, w którym nastąpiło dane zdarzenie. Wartość powinna być taka sama dla wszystkich zdarzeń w ramach tego samego bloku reklamowego. Użycie parametru offset dla materiałów typu broadcast (live) zostało szczegółowo opisane w sekcji Skryptowanie live streaming (broadcast).
additionalParameters[*Dictionary*]	Wymagany	Słownik zawierająca dodatkowe parametry definiujące zdarzenie *chngVol* dla danej reklamy.
Opis parametrów dodatkowych		
volume[*Number*]	Wymagany	Zdefiniowana przez użytkownika wartość procentowa poziomu głośności odtwarzacza (zakres od 0 do 100). Dla opcji *Wycisz* wartość powinna wynosić -1.

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

Opis argumentów		
programID[*String*]	Wymagany	Unikalny identyfikator programu (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @

		# * ( ) - / ? : ; ~ $,), deklarowany we wcześniejszym wywołaniu funkcji ***newProgram***.
offset[*Number*]	Wymagany	Przesunięcie czasu odtwarzania **programu** w sekundach. Moment, w którym nastąpiło dane zdarzenie. Użycie parametru offset dla materiałów typu broadcast (live) zostało szczegółowo opisane w sekcji Skryptowanie live streaming (broadcast).
additionalParameters[*Dictionary*]	Wymagany	Słownik zawierający dodatkowe parametry definiujące zdarzenie *chngVol* dla danego materiału
Opis parametrów dodatkowych		
volume[*Number*]	Wymagany	Zdefiniowana przez użytkownika wartość procentowa poziomu głośności odtwarzacza (zakres od 0 do 100). Dla opcji *Wycisz* wartość powinna wynosić -1.

2.5.14. Quality change (opcjonalnie)

Użytkownik zmienił jakość treści odtwarzanych w odtwarzaczu lub wykonał akcję mającą ten sam skutek podczas emisji reklamy lub materiału

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

Opis argumentów		
programID[*String*]	Wymagany	Unikalny identyfikator programu (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,), deklarowany we wcześniejszym wywołaniu funkcji ***newProgram***.
adID[*String*]	Wymagany	Unikalny identyfikator reklamy (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,), deklarowany we wcześniejszym wywołaniu funkcji ***newAd***.
offset[*Number*]	Wymagany	Przesunięcie czasu w sekundach – moment w treści **programu**, w którym nastąpiło dane zdarzenie. Wartość powinna być taka sama dla wszystkich zdarzeń w ramach tego samego bloku reklamowego. Użycie parametru offset dla materiałów typu broadcast (live) zostało szczegółowo opisane w sekcji Skryptowanie live streaming (broadcast).

additionalParameters[*Dictionary*]	Opcjonalny	Słownik zawierający dodatkowe parametry definiujące zdarzenie *chngQual* dla danej reklamy.
Opis parametrów dodatkowych		
quality [*String*]	Opcjonalny	Jakość (np. 1920x1080) załadowanej reklamy zdefiniowana przez użytkownika.

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

Opis argumentów		
programID[*String*]	Wymagany	Unikalny identyfikator programu (max. 64 znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 oraz znaki specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,), deklarowany we wcześniejszym wywołaniu funkcji ***newProgram***.
offset[*Number*]	Wymagany	Przesunięcie czasu odtwarzania **programu** w sekundach. Moment, w którym nastąpiło dane zdarzenie. Użycie parametru offset dla materiałów typu broadcast (live) zostało szczegółowo opisane w sekcji Skryptowanie live streaming (broadcast).
additionalParameters[*Dictionary*]	Opcjonalny	Słownik zawierający dodatkowe parametry definiujące zdarzenie *chngQual* dla danego materiału.
Opis parametrów dodatkowych		
quality[*String*]	Opcjonalny	Jakość (np. 1920x1080) załadowanej reklamy zdefiniowana przez użytkownika.

2.6. Informacje dodatkowe

2.6.1. Parametr offset

Każde zdarzenie dotyczące odtwarzacza jest wykonywane w kontekście bieżącego przesunięcia (***offset***) materiału. Przesunięcie to czas (konkretna sekunda programu), w którym czynność jest wykonywana. Jeżeli czynność ma miejsce przed emisją pierwszej części materiału, przesunięcie ma wartość **0**. Każda czynność, która przerywa odtwarzanie materiału, wymaga przesłania o niej informacji. Po wznowieniu odtwarzania powinno nastąpić przesłanie zdarzenia ***play***. Pozwala to na poprawne (automatyczne) zliczenie czasu odtwarzania materiału.

Offset dla zdarzenia play reklamy służy do określania typu bloku reklamowego.

Użycie parametru offset dla materiałów typu broadcast (live) zostało szczegółowo opisane w sekcji Skryptowanie live streaming (broadcast).

2.6.2. Typ bloku reklamowego

Typ bloku reklamowego jest określany przez porównanie parametru przesunięcia (***offset***) przekazanego ze zdarzeniem ***play*** reklamy i czasu trwania programu. W przypadku materiału dłuższego niż 100 sekund przyjmowany jest 5-sekundowy margines. Rodzaje bloków reklamowych są określane według następujących zasad:

- Live: transmissionType = 2 (patrz sekcja Skryptowanie live streaming (broadcast))
- Unknown: czas trwania < 0 oraz transmissionType ≠ 2
- Pre: offset <= 0 (dla czasu trwania > 100, offset <= 5)
- Post: offset >= czasu trwania (dla czasu trwania > 100, offset >= czas trwania minus 5)
- Mid: 0 < offset < czas trwania (dla czasu trwania > 100, 5 < offset < czas trwania minus 5)

2.6.3. Zdarzenie continue

W przypadku, gdy odtwarzanie materiału nie jest przerywane żadną czynnością przez 5 minut, zdarzenie ***continue*** jest generowane automatycznie w celu utrzymania czasu trwania wizyt i dokładniejszego pomiaru czasu. W przypadku zamknięcia przeglądarki, w której osadzony jest odtwarzacz objęty badaniem, generowane jest zdarzenie ***unload***, zawierające **gemiusPrism** informację na temat bieżącego przesunięcia (***offset***) emitowanego materiału. Należy mieć na uwadze, że podczas trwania bloku reklamowego i emisji poszczególnych reklam, rozpoczęty program i dana część programu powinny znajdować się w stanie ***break***.

2.6.4. Zmiana wymiarów i widoczności playera

Jeśli funkcja ***setVideoObject*** została wywołana ze wskazaniem elementu strony, który jest odtwarzaczem, przesyłana jest informacja o rozdzielczości odtwarzacza. Jeśli rozmiar odtwarzacza zmieni się w trakcie odtwarzania, w zdarzeniu ***continue*** przesyłane są pierwotne wymiary odtwarzacza.

Jeśli skrypt jest w stanie zweryfikować widoczność (zostały spełnione warunki opisane w sekcji Przekazywanie obiektu odtwarzacza i obiekt wideo/audio został przekazany do skryptu), jest ona sprawdzana co sekundę i wysyłana w każdym zdarzeniu. Parametr widoczności może przyjmować wartość **0** (odtwarzacz niewidoczny) lub **1** (odtwarzacz widoczny).

Zakłada się, że odtwarzacz jest widoczny, gdy obiekt wideo/audio przekazywany metodą ***setVideoObject*** jest widoczny w co najmniej 50%. Jeśli widoczność odtwarzacza zmieni się w trakcie odtwarzania, zdarzenie ***continue*** jest wysyłane z poprzednią wartością widoczności. Dzięki temu możliwy jest pomiar czasu odtwarzania w czasie gdy odtwarzacz jest widoczny.

Należy pamiętać, że zmiana widoczności odtwarzacza zostaje zarejestrowana, gdy:

- okno przeglądarki zostaje zminimalizowane lub jest otwarte w tle,
- użytkownik zmienia zakładkę przeglądarki,
- użytkownik przewija stronę,
- użytkownik przewija zawartość ramki HTML.

Zmiana widoczności odtwarzacza nie zostaje zarejestrowana, gdy:

- ekran zostaje wyłączony,
- wyświetlany jest wygaszacz ekranu,
- ekran zostaje zablokowany,
- okno przeglądarki jest przykryte innym oknem.

2.6.5. Czyszczenie obiektu GemiusPlayer – metoda dispose()

Obiekt odtwarzacza jest automatycznie czyszczony po zamknięciu przeglądarki lub jej całkowitym przeładowaniu. W niektórych sytuacjach, m.in. w przypadku implementacji stream Webview w aplikacji mobilnej lub skryptowania witryn Single Page Application, może zaistnieć potrzeba wyczyszczenia zasobów obiektu GemiusPlayer bez zamykania/przeładowywania instancji przeglądarki.

Wywołanie metody **dispose()** powoduje, że obiekt odtwarzacza **zwalnia** odpowiednie zasoby i jest gotowy do **usunięcia**. Od tego momentu przestanie również wysyłać kolejne hity, zarówno generowane automatycznie (np. 'continue', 'unload'), jak i te wywoływane przez aplikację. Nie dotyczy to hitów, które zostały umieszczone w kolejce do wysłania przed wywołaniem dispose().

Jeśli podczas wywoływania dispose() nadal istnieją aktywne materiały w stanie **'play'**, zostanie wysłane dodatkowe zdarzenie **'dispose'** (analogicznie do **'unload'**). Jednakże, gdy odtwarzanie materiału zostanie przerwane, **należy zawsze użyć zdarzenia 'close'**.

```
var player = new GemiusPlayer("192038","xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.xx",
{"currentDomain":"example.com", "volume":100} );
player.newProgram(123456,{
    'programName':'The North Remembers',
    'programDuration':3030,
    'programType':'video',
});
player.programEvent(123456,0,"play",{"autoPlay":true, "partID":1});
player.programEvent(123456,20,"close");
player.dispose();
delete player;
```

2.7. Wysyłanie dodatkowych parametrów

Korzystając z **gemiusPrism** wydawcy mają możliwość zdefiniowania i zarządzania dowolną formą wewnętrznej kategoryzacji. Funkcje wywoływane podczas pomiaru stream posiadają pośród ***additionalParameters*** dedykowaną przestrzeń służącą do definiowania i przekazywania argumentu ***customAttributes***. Można w nim zawrzeć informacje niezbędne do sformułowania kategorii będących częścią opisu materiału i reklamy. Powinny one przyjmować format: 'key':value.

```
1 'customAttribute1':attributeValue1,
2 'customAttribute2':attributeValue2,
3 'customAttribute3':attributeValue3,
4 'customAttribute4':attributeValue4,
5 'customAttribute5':attributeValue5;
```

Dla każdego materiału istnieje możliwość zdefiniowania 5 dodatkowych atrybutów ***customAttributes***, których analiza będzie możliwa w **gemiusPrism**, bądź poprzez vSender (jeżeli Wydawca posiada wykupioną tę usługę). Dodatkowe (prywatne) parametry Wydawcy nie będą wykorzystane w Badaniu Jednoźródłowym, służą jedynie wewnętrznej analityce Wydawcy.

Przy konstruowaniu wszystkich parametrów dodatkowych należy przestrzegać następujących reguł:

- dozwolone są znaki: a-z, A-Z oraz znaki diakrytyczne, 0-9 i oraz specjalne: _ . ! @ # * ( ) - / ? : ; ~ $,
- dla każdej pary (‘key’:value) łączna liczba znaków może wynieść maksymalnie 200,
- nazwy zaczynające się od znaku „_” nie są dozwolone (są one przeznaczone wyłącznie do użytku wewnętrznego firmy Gemius),
- nazwy atrybutów zdefiniowane dla materiałów i reklam powinny się różnić.

Aby zapewnić pełny zakres danych opisujących sesję, wyświetlane treści i reklamy, Gemius proponuje zestaw obowiązkowych, rekomendowanych i opcjonalnych parametrów, które są zawarte w informacji wysyłanej w momencie zarejestrowania stanu lub akcji w odtwarzaczu i zawsze są uwzględniane na liście ***additionalAttributes***.

2.8.Skryptowanie live streaming (broadcast)

Materiały transmitowane na żywo, tj. emisja koncesjonowanych stacji telewizyjnych oraz programów radiowych oraz ich internetowe odpowiedniki powinny być skryptowane zgodnie z wcześniej zamieszczonymi wytycznymi, z poniższymi tylko wyjątkami:”riały transmitowane na żywo powinny być skryptowane zgodnie z wcześniej zamieszczonymi wytycznymi, z poniższymi tylko wyjątkami:

1. Parametr ***transmissionType[Number]***
   W przypadku treści transmitowanych na żywo, parametr *transmissionType* powinien przyjmować wartość „2” (broadcast).

2. Parametr ***programDuration[Number]***
   Jeśli nie jest możliwe wyznaczenie czasu trwania materiału (np. w przypadku transmisji na żywo lub transmisji na żywo kanału TV), wartość parametru *programDuration* powinna być ustawiona na -1.

3. Parametry ***offset[Number]*** oraz ***transmissionStartTime[String]***
   W przypadku treści transmitowanych na żywo, parametr *offset* powinien być taki sam jak timestamp wskazujący, kiedy dany materiał był nadawany. Taka sama zasada dotyczy parametru *transmissionStartTime*. Dla materiałów, które mogą być oglądane z opóźnieniem (TSV – time shifted viewing) wartość parametrów *offset* oraz *transmissionStartTime* powinna równać się bieżącemu czasowi pomniejszonemu o przesunięcie spowodowane opóźnieniem, a dla materiałów, których nie można oglądać z opóźnieniem – bieżącemu czasowi.

   Podsumowując, w przypadku materiałów typu broadcast, programów na żywo oraz TSV przekazywany timestamp powinien być powiązany z czasem emisji materiału.

   Przykładowe wartości parametrów:

   Użytkownik zaczyna oglądanie materiału 1 lutego 2020 o godzinie 12:00. *Offset* dla rozpoczęcia odtwarzania materiału przyjmuje więc wartość 1580558400 (2020-02-01 12:00:00).
   Użytkownik rozpoczął oglądanie materiału 1 lutego 2020 o godzinie 12:00, jednak oglądany przez niego fragment materiału był transmitowany o 10:00. W takiej sytuacji parametr *offset* dla rozpoczęcia odtwarzania materiału powinien przyjmować wartość 1580551200 (2020-02-01 10:00:00).

   W przypadku materiałów stream nadawanych w czasie rzeczywistym, np. w radiach internetowych, każde zatrzymanie odtwarzania materiału przez użytkownika (niezależnie od funkcji, którą zostało wywołane czyli tez jeżeli przycisk zatrzymujący ma znak „pauzy”), które nie daje możliwości powrotu do odtwarzania w miejscu, w którym odtwarzanie zostało przerwane, powinno być traktowane jako zakończenie odtwarzania. Wznowienie odtworzenia powinno w takim przypadku powinno być traktowane jako kolejne odtworzenie (views).

4. W przypadku transmitowania kanału TV lub radio, należy zdefiniować parametry ***transmissionChannel[String]***

5. Parametr ***programGenre[Number]***

   Jeśli nie jest możliwe określenie gatunku materiału nie należy wysyłać parametru *programGenre*. W przypadku treści transmitowanych na żywo, parametr *programGenre* powinien przyjmować wartość „1” (Program na żywo).

3. Przykład implementacji

Przeanalizujmy przykładowy scenariusz implementacji kodu. Mamy do dyspozycji poniższe informacje o materiale, a także wiemy, że nie ma bloku reklamowego pre-roll, ale są dwa spoty reklamowe w bloku reklamowym mid-roll i jeden spot reklamowy w bloku reklamowym post-roll. Tym samym program jest podzielony na dwie części. Ma również włączone ustawienie auto-play i emisja rozpoczyna się zaraz po załadowaniu się odtwarzacza i treści.

```
gemiusID = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.xx
  programID = 123456
  progamName = The_North_Remembers
  programDuration = 3030 // length of the material in seconds
  transmissionType = on demand (czyli wartość = 1)
  programGenre = serial (czyli wartość = 3)
  series = Game_of_Thrones
  programSeason = season_2
  programProducer = HBO
  intName = Furbe
  intCategory = Comedy
  intType = vod
  intStatus = public
  intCanal = canal1
```

intName, *intCategory*, *intType*, *intStatus* oraz *intCanal* to przykładowe zdefiniowane własne zmienne użytkownika.

Na początku skrypt zliczający powinien zostać załadowany, a obiekt odtwarzacza utworzony.

```
<script type="text/javascript">
function gemius_pending(i) { window[i] = window[i] || function() {var x = window[i+'_pdata'] = window[i+'_pdata'] || [];
x[x.length]=arguments;};};
gemius_pending('gemius_init');
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("IPLA","xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.xx",
{"currentDomain":"example.com", "volume":100} );
```

Następnie niezbędne jest przekazanie informacji o obiekcie wideo/audio, aby wielkość i widoczność odtwarzacza mogły zostać wykryte automatycznie:

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

Po załadowaniu się materiału w odtwarzaczu następuje wywołanie funkcji ***newProgram***.

```
player.newProgram(123456,{
    'programName':'The_North_Remembers',
    'programDuration':3030,
    'programType':'video',
    'transmissionType':1,
    'programGenre':3,
    'series':'Game_of_Thrones',
    'programSeason':'season_2',
    'programProducer':'HBO',
    'quality':'1920x1080',
            'resolution':'1280x720',
            'volume':80,
    'intName':'Furbe',
    'intCategory':'Comedy',
    'intType':'vod',
```

```
        'intStatus':'public',
        'intCanal':'canal1'
});
```

Ponieważ funkcja auto-play jest włączona, odtwarzanie następuje natychmiast, co zostaje zgłoszone poprzez wywołanie funkcji ***programEvent***. Ustawienia emisji dotyczące poziomu głośności i rozdzielczości pozostają niezdefiniowane.

```
player.programEvent(123456,0,"play",{"autoPlay":true,"volume":100});
```

Po pięciu sekundach użytkownik pauzuje odtwarzanie za pomocą przycisku pause. To powoduje wywołanie kolejnych funkcji – najpierw ***pause***, a po wznowieniu odtwarzania - ***play***.

```
player.programEvent(123456,5,"pause");
player.programEvent(123456,5,"play",{"autoPlay":false, "volume":100});
```

Następnie po 90 sekundach materiał zostaje przerwany przez blok reklamowy mid-roll, a po 2 sekundach wyświetlania reklamy użytkownik wywołuje zdarzenie ***close*** poprzez wybranie innego materiału.

```

player.programEvent(123456,90,"break");
player.newAd(49327505,{
        'adName':'ColaT',
        'adType':'promo',
        'adDuration':5,
        'volume':100
});
player.adEvent(123456,49327505,90,"play",{"autoPlay":true,
 "addPosition":1, "breakSize":2 'volume':100});
player.adEvent(123456,49327505,90,"close");
player.programEvent(123456,90,"close");

```

Wartości przesunięcia czasu (***offset***) mają zastosowanie tylko do programu.

Pomiar obejmował rozpoczęcie pierwszego spotu reklamowego z dwóch. Ten spot reklamowy nie miał przypisanej żadnej klasyfikacji dotyczącej kampanii. Emisja treści zostaje wznowiona.

4. Opóźniona inicjalizacja skryptu

Jeśli potrzebne jest opóźnienie inicjalizacji skryptu, np. w celu przekazania wartości parametrów dotyczących zgód użytkownika dopiero po załadowaniu strony, można skorzystać z funkcji `pp_gemius_init` i zmiennej `pp_gemius_init_timeout`. Wartość timeout (w milisekundach) należy zdefiniować w skrypcie zliczającym w sposób pokazany niżej:

```
<script type="text/javascript">
var pp_gemius_init_timeout = 10000;

// lines below shouldn't be edited
function gemius_pending(i) { window[i] = window[i] || function() {var x = window[i+'_pdata'] = window[i+'_pdata'] || [];
x[x.length]=arguments;};};
gemius_pending('gemius_init');
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>
```

`pp_gemius_init_timeout` wstrzymuje inicjalizację skryptu do momentu wywołania `pp_gemius_init` przez kod po stronie wydawcy.

4.1. Wywołanie pp_gemius_init

Aby zainicjalizować skrypt przed upływem czasu zdefiniowanego jako wartość timeout, należy wywołać funkcję `pp_gemius_init`. Przyjmuje ona te same parametry, co zmienne globalne (ale w formie słownikowej bez przedrostka „gemius”):

```
pp_gemius_init({
  'consent': true,
  'extraparameters' = ['language=EN', 'section_id=news'];
});
```

Parametry przekazane w wywołaniu `pp_gemius_init` mają pierwszeństwo. Pominięcie parametru z przekazanego słownika skutkuje odczytaniem zmiennej globalnej, a jeśli globalna nie jest zdefiniowana, używana jest domyślna wartość parametru.

4.2. Zmienna pp_gemius_init_timeout

Wartość zmiennej `pp_gemius_init_timeout` powinna zostać zdefiniowana na taką liczbę sekund, jaką skrypt Gemius ma czekać na wywołanie funkcji `pp_gemius_init`.

Jeśli zmienna `pp_gemius_init_timeout` nie jest zdefiniowna lub upłynął czas zdefiniowany za jej pomocą, wszystkie kolejne wywołania funkcji `pp_gemius_init` zostaną zignorowane.

Jeśli użytkownik opuści stronę zanim zostanie osiągnięty zdefiniowany timeout, skrypt zostanie zainicjalizowany z globalnymi lub domyślnymi wartościami zmiennych.

Jeśli skrypt ma oczekiwać na odpowiedź `pp_gemius_init` do skutku, wartość zmiennej `pp_gemius_init_timeout` należy ustawić na Infinity:

```
var pp_gemius_init_timeout = Infinity;
```

Jeśli użytkownik opuści stronę zanim funkcja `pp_gemius_init` zostanie wywołana, a timeout jest ustawiony na Inifinity, **nie zostaną wysłane żadne hity**.

Jeśli oprócz pomiaru Stream (gplayer.js) na stronie wykorzystywane są także skrypty Audience lub Prism (xgemius.js lub gemius.js), zmienna `pp_gemius_init_timeout` i wywołanie `pp_gemius_init` dotyczyć będą obu skryptów jednocześnie.

5. Zgoda użytkownika

5.1. Wprowadzenie

Na rynkach zrzeszonych w Unii Europejskiej możesz być zobowiązany do jasnego informowania użytkowników o celach przechowywania ich danych i zbierania potrzebnych zgód. Istnieje kilka sposobów, aby zapytać i uszanować decyzje użytkowników w tej sprawie, a skrypt Gemius ma za zadanie je wspierać. Najczęściej używane je CMP zgodne z IAB (TCF v.2, starsze wersje nie są rekomendowane) i dostosowanie skryptu Gemius do odczytywania wartości zgód poprzez dodanie odpowiedniego parametru. Jeśli CMP, z którego korzystasz, nie jest zgodne z IAB, nadal możesz przekazać zgody do skryptu Gemius w formacie zgodnym z IAB. Jeśli wykorzystujesz dane do targetowania behawioralnego AdOcean, postępuj zgodnie z dedykowaną instrukcją. Rozdział „Rozwiązanie niestandardowe” opisuje, w jaki sposób przekazać informacje o zgodach, jeśli zapytasz użytkowników w inny sposób niż przez CMP.

Uwaga: Gemius potrzebuje informacji zarówno w przypadku nieudzielenia, jak i udzielenia zgody przez użytkownika. Przypadki, w których użytkownik wyraził zgodę, a parametry przekazujące tę informację zostaną pominięte, będą traktowane jako zgoda nigdy nie wyrażona i mogą kwestionować możliwość przetwarzania przez Gemius zebranych danych.

5.2. Obsługa Consent Managament Platform

5.2.1. CMP zgodne z IAB

CMP to rozwiązanie implementowane na stronie wydawcy, służące do zarządzania zgodą użytkownika na przetwarzanie danych osobowych i pozwalające na przesyłanie informacji pomiędzy dostawcami technologii. Lista zarejestrowanych przez IAB Europe dostawców CMP (Consent Management Providers, CMPs) dostępna jest na stronie https://iabeurope.eu/cmp-list/.

Aby zapewnić obsługę CMP zgodnych z IAB Europe's GDPR Transparency & Consent Framework, udostępniamy dedykowane parametry służące do przekazywania do naszych serwerów informacji na temat udzielonej zgody lub jej braku.

Uwaga: Korzystanie z parametrów nie jest obligatoryjne.

W przypadku korzystania z CMP zgodnego z IAB Framework, należy ustawić zmienną `pp_gemius_use_cmp` na `true`. Informacje na temat zgody lub jej braku zbierane są przez CMP. Jeśli użytkownik nie wyraża zgody na przetwarzanie danych przez Gemius, nie będą do Gemius wysyłane informacje na temat cookieID ani browserID. Jeśli na stronie nie zostało zamieszczone żadne CMP, można użyć zmiennej `pp_gemius_gdpr_consent`.

Uwaga: Nasz skrypt przeszukuje strukturę DOM (w górę) w celu znalezienia ramki __tcfapiLocator. Ramka ta odpowiada za przekazanie informacji na temat zgód użytkownika. Prosimy o upewnienie się, że w kodzie HTML strony ta ramka znajduje się przed skryptem Gemius.

```
<script type="text/javascript">
var pp_gemius_use_cmp = true;

// lines below shouldn't be edited
function gemius_pending(i) { window[i] = window[i] || function() {var x = window[i+'_pdata'] = window[i+'_pdata'] || [];
x[x.length]=arguments;};};
gemius_pending('gemius_init');
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>
```

W przypadku korzystania z synchronicznej wersji skryptu:

```
<script type="text/javascript">
var pp_gemius_use_cmp = true;
</script>
<script type="text/javascript" src="https://PREFIX.hit.gemius.pl/gplayer.js"></script>
```

5.2.2. Korzystanie z danych w targetowaniu behawioralnym AdOcean

W przypadku korzystania z danych zebranych w gemiusPrism w targetowaniu behawioralnym AdOcean, należy użyć zmiennej `pp_gemius_dmp_purpose` i ustawić jej wartość na `true`.

```
<script type="text/javascript">
var pp_gemius_use_cmp = false; //can be omitted
var pp_gemius_gdpr_consent = <CONSENT_STRING>;

// lines below shouldn't be edited
function gemius_pending(i) { window[i] = window[i] || function() {var x = window[i+'_pdata'] = window[i+'_pdata'] || [];
x[x.length]=arguments;};};
gemius_pending('gemius_init');
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>
```

5.2.3. CMP timeout

Domyślnie gemiusPrism czeka na informację od CMP na temat zgody użytkownika przez 10 sekund. Aby zmienić czas oczekiwania, można użyć opcjonalnej zmiennej `pp_gemius_cmp_timeout` i przekazać w niej liczbę milisekund, przez jaką skrypt ma oczekiwać na odpowiedź CMP:

```
<script type="text/javascript">
var pp_gemius_use_cmp = true;
var pp_gemius_cmp_timeout = 10000;

// lines below shouldn't be edited
function gemius_pending(i) { window[i] = window[i] || function() {var x = window[i+'_pdata'] = window[i+'_pdata'] || [];
x[x.length]=arguments;};};
gemius_pending('gemius_init');
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>
```

Jeśli skrypt ma oczekiwać na odpowiedź CMP do skutku, należy ustawić wartość zmiennej `pp_gemius_cmp_timeout` na `Infinity`:

```
<script type="text/javascript">
var pp_gemius_use_cmp = true;
var pp_gemius_cmp_timeout = Infinity;

// lines below shouldn't be edited
function gemius_pending(i) { window[i] = window[i] || function() {var x = window[i+'_pdata'] = window[i+'_pdata'] || [];
x[x.length]=arguments;};};
gemius_pending('gemius_init');
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>
```

5.2.4. CMP niezgodne z IAB

W przypadku korzystania z CMP niezgodnego z IAB Framework lub w przypadku chęci nadpisania ustawień CMP, można użyć parametrów `pp_gemius_gdpr_consent` oraz `pp_gemius_gdpr_applies`.

Aby zastosować informacje na temat zgody użytkownika lub jej braku w formacie zgodnym ze standardem IAB na stronie, która nie korzysta z CMP, należy użyć zmiennej

`pp_gemius_gdpr_consent = <CONSENT_STRING>;`

na przykład: `var pp_gemius_gdpr_consent ='BORViZzORViZzABABBENBK8AAAAceADAACABgApA';`

Uwaga: Zmienna `pp_gemius_use_cmp` ma wyższy priorytet. Jeśli `pp_gemius_use_cmp` jest ustawiona na `true` i CMP jest zamieszczone na stronie, użyte zostaną wartości podane przez CMP.

Uwaga: Jeśli `pp_gemius_use_cmp` jest ustawiona na `true` , ale CMP nie jest zamieszczone na stronie oraz zmienna `pp_gemius_gdpr_consent = <CONSENT_STRING>;` nie została użyta, skrypt zachowa się tak jakby użytkownik wyraził zgodę.

```
<script type="text/javascript">
var pp_gemius_use_cmp = false; //can be omitted
var pp_gemius_gdpr_consent = <CONSENT_STRING>;

// lines below shouldn't be edited
function gemius_pending(i) { window[i] = window[i] || function() {var x = window[i+'_pdata'] = window[i+'_pdata'] || [];
x[x.length]=arguments;};};
gemius_pending('gemius_init');
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>
```

Aby zdecydować, czy zapisy EU General Data Protection Regulation (GDPR, RODO) mają zastosowanie do użytkownika, należy użyć zmiennej `pp_gemius_gdpr_applies` i ustawić jej wartość na `true` lub `false`.

Jeśli RODO nie ma zastosowania w przypadku danego użytkownika, można ustawić zmienną `pp_gemius_gdpr_applies` na `false` - nie jest wtedy sprawdzana `pp_gemius_gdpr_consent` zakładając, że użytkownik wyraził zgodę. W przeciwnym razie określenie zgody lub jej braku zależy od wartości zmiennej `pp_gemius_gdpr_consent`.

```
<script type="text/javascript">
var pp_gemius_use_cmp = false; //can be omitted
var pp_gemius_gdpr_consent = <CONSENT_STRING>;

// lines below shouldn't be edited
function gemius_pending(i) { window[i] = window[i] || function() {var x = window[i+'_pdata'] = window[i+'_pdata'] || [];
x[x.length]=arguments;};};
gemius_pending('gemius_init');
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>
```

Jeśli potrzebne jest opóźnienie inicjalizacji skryptu do czasu uzyskania informacji na temat zgody użytkownika, należy skorzystać z funkcji `pp_gemius_init` i zmiennej `pp_gemius_init_timeout`, opisanych w rozdziale 4.

Przykładowa definicja funkcji `pp_gemius_init`:

```
pp_gemius_init({
  'gdpr_consent' = <CONSENT_STRING>,
});
```

5.3. Rozwiązania niestandardowe

Jeśli Twoja witryna nie korzysta z CMP zgodnego z IAB ani parametru wymienionego wyżej, właściwym sposobem uszanowania decyzji użytkownika o udostępnieniu jego danych jest użycie parametru `pp_gemius_consent`. Aby poprawnie przechowywać informacje o wyrażeniu zgody przez użytkownika, należy podać wartość parametru `pp_gemius_consent` zarówno w przypadku, gdy zgoda została udzielona, jak i gdy jej nie udzielono.

Jeśli odwiedzający wyrazi zgodę na przetwarzanie jego danych w Twojej witrynie, jesteś zobowiązany do zmiany skryptu poprzez dodanie zmiennej `pp_gemius_consent` z wartością `true`.

W sytuacji, gdy odwiedzający sprzeciwia się przetwarzaniu jego danych, należy dodać ten sam parametr `pp_gemius_consent` z wartością `false`.

```
<script type="text/javascript">
var pp_gemius_consent = false;

// lines below shouldn't be edited
function gemius_pending(i) { window[i] = window[i] || function() {var x = window[i+'_pdata'] = window[i+'_pdata'] || [];
x[x.length]=arguments;};};
gemius_pending('gemius_init');
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>
```

Należy pamiętać, aby zmienną o wartości `false` definiować jedynie w przypadku użytkowników, którzy nie chcą być śledzeni. Zdefiniowanie tej zmiennej za każdym razem spowoduje, że cały ruch na tej stronie będzie bezcookiesowy.

Jeśli potrzebne jest opóźnienie inicjalizacji skryptu do czasu uzyskania informacji na temat zgody użytkownika, należy skorzystać z funkcji pp_gemius_init i zmiennej pp_gemius_init_timeout, opisanych w rozdziale 4.

Przykładowa definicja funkcji pp_gemius_init:

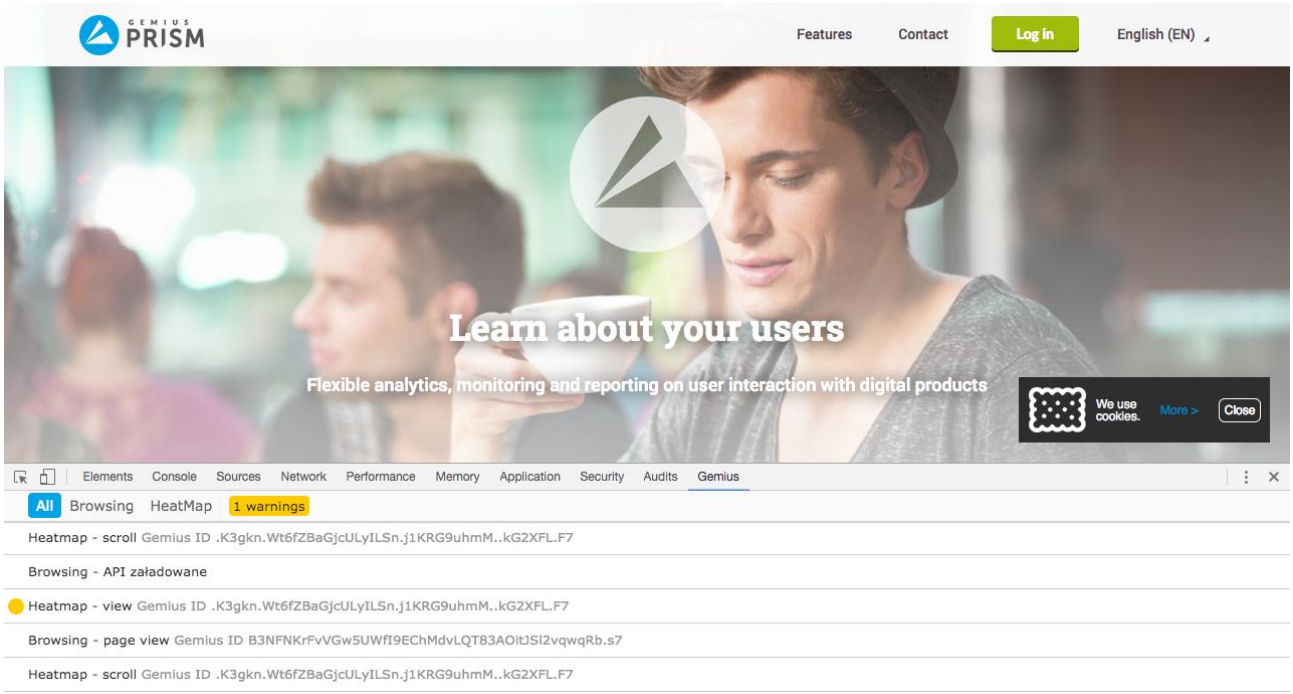
```
pp_gemius_init({
  'gdpr_consent' = <CONSENT_STRING>,
});
```

6. Weryfikacja oskryptowania za pomocą Gemius Debugger

Gemius Debugger to rozszerzenie Chrome ułatwiające weryfikację implementacji skryptów Gemius.

W panelu wtyczki wyświetlana jest informacja, czy skrypty **gemiusPrism** zostały prawidłowo zaimplementowane oraz czy z witryny przesyłane są prawidłowe dane. Kiedy tryb debugowania jest włączony, prezentowana jest lista wykonanych zapytań (standardowych, stream i e-commerce) wraz z ich nazwami, komentarzami oraz liczbą ostrzeżeń/błędów. Przedstawiane jest także sugerowane rozwiązanie dla każdego ostrzeżenia/błędu.

Aby rozpocząć proces weryfikacji, należy zainstalować wtyczkę **Gemius Debugger**, dostępną na stronie https://debugger.gemius.com.



Po zainstalowaniu **Gemius Debugger**, należy uruchomić w przeglądarce widok *Narzędzia dla programistów* (Menu > Więcej narzędzi > Narzędzia dla programistów), wybrać zakładkę Gemius, kliknąć w przycisk general i wybrać odpowiednie opcje (można to również zrobić poprzez Menu > Więcej narzędzi > Rozszerzenia: chrome://extensions/).

Za pomocą **Gemius Debugger** można także weryfikować oskryptowanie stron w wersji mobilnej – w tym celu należy kliknąć w ikonę *Toggle device toolbar* w lewym górnym rogu panelu i dostosować widok strony do konkretnego urządzenia za pomocą wbudowanych opcji.

MEDIAPANEL

Debug mobile website