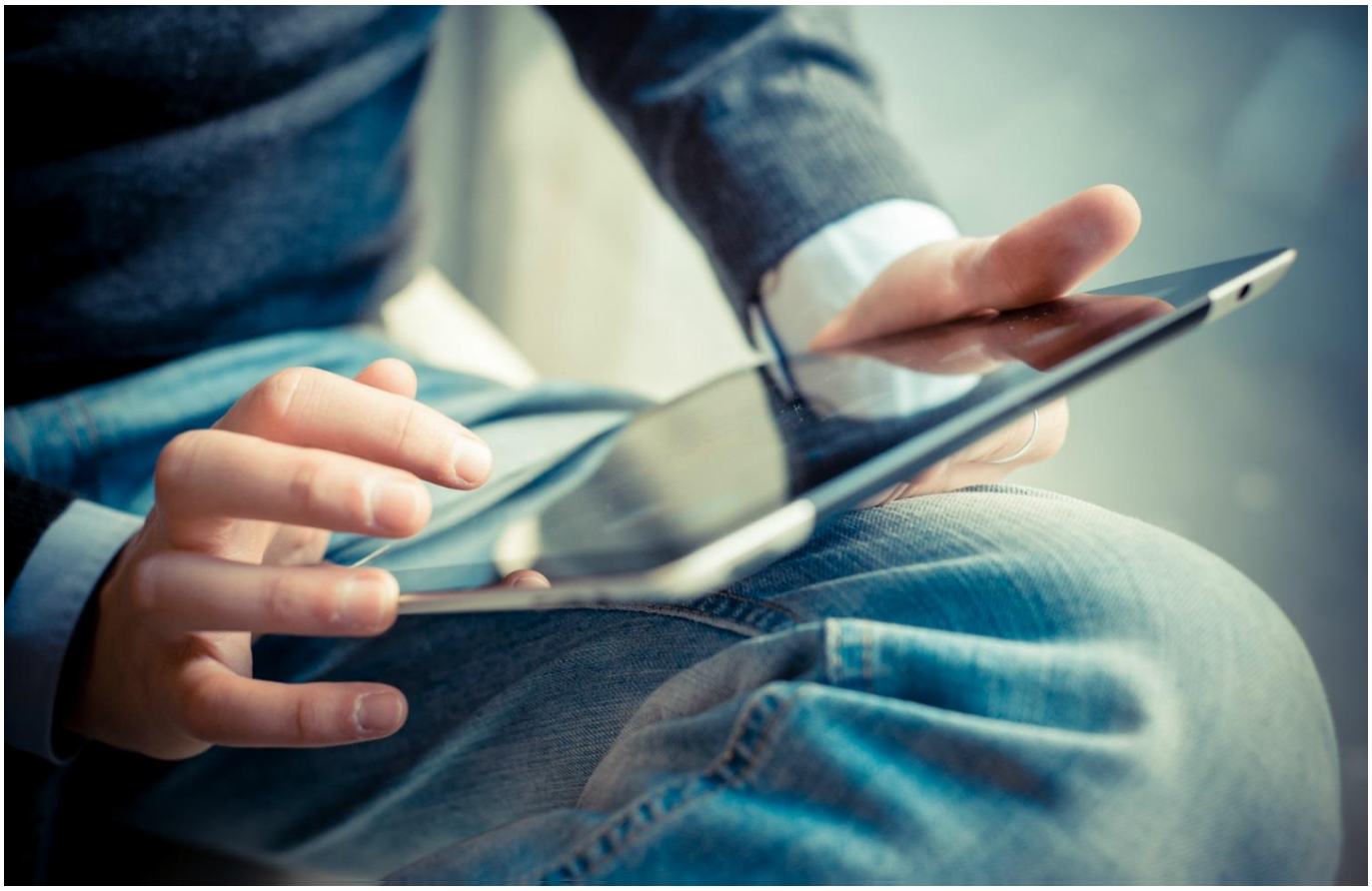


SENSIC STREAM AGENT API

Instrukcja implementacji dla pomiaru treści strumieniowych (iOS)

PLaNET PBI GfK

Ostatnia zmiana: 27.11.2015

Spis treści

1 Wprowadzenie 3
1.2 O dokumencie 3
2 Integracja 4
2.1 Integracja biblioteki SSA-A 4
2.2 Zainicjowanie klasy SSA 4
2.3 Generowanie zapytań śledzących 5
2.3.1 Pozyskanie agenta 5
2.3.2 Śledzenie czynności użytkownika za pomocą agenta 6
2.4 Testowanie 7

1 Wprowadzenie

1.1 Ogólne informacje nt. pomiaru DAM (Digital Audience Measurement) na platformie SENSIC

Celem platformy SENSIC jest zapewnienie infrastruktury niezbędnej do zbierania danych w ramach pomiaru site-centric za pomocą technologii dostarczanych przez GfK. Wydawcy treści internetowych (stron WWW/aplikacji mobilnych/treści strumieniowych) implementują skrypty/SDK-i (Software Development Kit) zliczające bezpośrednio do treści zamieszczanych na swoich serwisach oraz aplikacjach. Odpowiednia struktura kodu oparta na parametrach Media ID oraz Content ID pozwala na klasyfikację mierzonych treści zgodnie z obowiązującą taksonomią (typologią treści).

Platforma SENSIC oferuje kilka różnych bibliotek oraz skryptów zliczających dla najczęściej używanych typów treści oraz platform. Biblioteki/skrypty zliczające powinny zostać zintegrowane z aplikacją (stroną WWW w przypadku skryptów), aby zapewnić przesłanie danych o odsłonach na platformę SENSIC, w przypadku, gdy odpowiednie zdarzenie zostało wywołane przez użytkownika w aplikacji. Tak zliczone odsłony są następnie procesowane, walidowane oraz wzbogacane dodatkowymi informacjami przed zapisaniem ich na serwerze.

1.2 O dokumencie

Instrukcja ta zawiera dokumentację biblioteki Sensic Stream Agent („SSA-A") dla systemu iOS, która pozwala na pomiar treści strumieniowych w aplikacjach mobilnych działających na systemie iOS. Niniejsza instrukcja przedstawia w poglądowy sposób informacje na temat budowy API oraz sposobu jego integracji z aplikacją.

2 Integracja

Aby śledzić wykorzystanie treści strumieniowych za pomocą SSA, należy zintegrować bibliotekę śledzącą z kodem aplikacji i powiązać ze zdarzeniami używanego odtwarzacza multimediów.

Niektóre parametry muszą zostać ustawione przez wydawcę w osadzonym kodzie biblioteki. Inne informacje są automatycznie ustawiane przez logikę śledzenia SSA.

Poniższe punkty ogólnie opisują techniczną integrację biblioteki SSA-A.

2.1 Integracja biblioteki SSA-A

Pierwszą rzeczą, którą należy zrobić, jest zintegrowanie biblioteki SSA-A z kodem aplikacji.

W środowisku Xcode za pomocą metody przeciągnij i upuść (Drag & Drop) należy dodać do nawigatora projektu (project navigator) aplikacji katalog `SSA-SDK`.
Folder powinien zawierać następujące pliki:

- `SSA-SDK/libSSA.a`
- `SSA-SDK/SSA.h`

Kliknij projekt w Nawigatorze projektu, następnie kliknij docelową aplikację, a następnie **Build Settings**, po czym wyszukaj pozycję „Other Linker Flags”, kliknij przycisk + i dodaj '-ObjC'.

2.2 Zainicjowanie klasy SSA

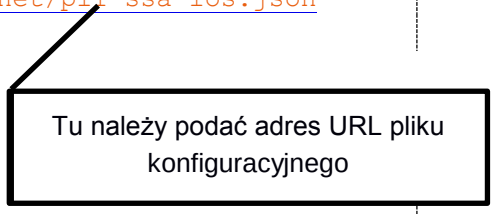
Do pracy z biblioteką SSA konieczne jest zainicjowanie klasy SSA. Mimo że może istnieć wiele wystąpień tej klasy, zazwyczaj nie będzie potrzeby tworzenia więcej niż jednej instancji.

Zainicjuj bibliotekę za pomocą polecenia `[[SSA alloc] initWithAdvertisingID: "someAdvertisingId"]` podczas uruchamiania aplikacji. Podany identyfikator `advertisingID` służy do identyfikacji użytkownika.

W tym celu można użyć klasy `AppDelegate`. Oto przykład:

```
#import "AppDelegate.h"
#import "SSA.h"
#import "Constants.h"
#import <AdSupport/ASIdentifierManager.h>


@implementation AppDelegate


SSA *SSA = nil;
// some methods and fields omitted for brevity

- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // ...
```

```
    NSString *idfaString = [[[ASIdentifierManager sharedManager]
advertisingIdentifier] UUIDString];

    if (idfaString) {
        SSA = [[SSA alloc]
                initWithAdvertisingId: idfaString
                andConfigUrl: @"https://config.sensic.net/pl1-ssa-ios.json"
               ];
    }
    else {
        // do not track
        SSA = [[SSA alloc] init];
    }

    return YES;
}
// ...
```



2.3 Generowanie zapytań śledzących

2.3.1 Pozyskanie agenta

Do śledzenia czasu przeglądania użytkownika konieczne jest pozyskanie agenta. W tym celu należy wywołać metodę `getAgent` klasy `SSA` i przekazać jej wartość *MediaID*. W takim przypadku będzie dokładnie jedno wystąpienie agenta na jedno *MediaID*.

Użycie tej metody zostało pokazane w poniższym fragmencie kodu:

```
#import "VideoPlayerController.h"
#import "SSA.h"

// some imports omitted for brevity

extern SSA *SSA;

@interfaceVideoPlayerController () {
        Agent *agent = nil;
    }
    // some properties omitted for brevity
@end

@implementation VideoPlayerController

// some methods omitted for brevity

- (void)viewDidLoad
{
    [super viewDidLoad];
```

```
    // ...
self.agent = [SSA getAgentForMediaId: @"someMediaId"];
}
@end
```

Ustaw tutaj **MediaID**!

Jeśli dla jednego identyfikatora *MediaID* potrzeba wielu agentów, należy również podać wartość `playerId`. Wartość `playerId` musi być unikatowa w obrębie aplikacji.

2.3.2 Śledzenie czynności użytkownika za pomocą agenta

Zanim można będzie śledzić czas oglądania treści strumieniowych, agent musi zostać zainicjowany za pomocą identyfikatora *ContentId* odpowiadającemu śledzonemu materiałowi wideo/audio. Opcjonalnie można też podać zestaw parametrów niestandardowych.

Zawsze, gdy użytkownik rozpocznie lub wznowi oglądanie treści, musi zostać wywołana metoda `notifyPlay`. Wywołanie tej metody, gdy treści odtwarzane w playerze są już mierzone, nie przyniesie żadnego efektu.

Gdy użytkownik zatrzyma lub wstrzyma oglądanie treści, musi zostać wywołana metoda `notifyIdle`. Wywołanie tej metody, gdy player jest już wstrzymany, nie przyniesie żadnego efektu.

Zastosowanie opisanych metod zostało pokazane w poniższym kodzie:

```
#import "VideoPlayerController.h"
#import "SSA.h"

     // some imports omitted for brevity

extern SSA *SSA;
@interfaceVideoPlayerController () {
    Agent *agent = nil;
     }
    // some properties omitted for brevity
@end

@implementationVideoPlayerController
    // some methods omitted for brevity
- (void)viewDidLoad
{
    [super viewDidLoad];
    // ...
    [self.agent notifyLoadedWithContentId: @"someContentId"
     andCustomParameters: @{@"cp1" :"customParamValue 1"}];
}
- (void)playbackStateChanged
{
    switch (self.moviePlayer.playbackState) {
```

Ustaw tutaj **ContentID**

Ustaw tutaj własne parametry niestandardowe (maks. 4)

```
        case MPMoviePlaybackStatePlaying:
            [self.agent notifyPlay]
            break;

        case MPMoviePlaybackStatePaused:
        case MPMoviePlaybackStateInterrupted:
        case MPMoviePlaybackStateSeekingBackward:
        case MPMoviePlaybackStateSeekingForward:
        case MPMoviePlaybackStateStopped:
              [self.agent notifyIdle]
              break;
    }
}
@end
```

2.4 Testowanie

Gdy odpowiednia biblioteka Sensic Site Tracker została dodana do aplikacji, najlepszą metodą, aby sprawdzić implementację lub zdebugować poprawność zapytań wysyłanych do serwera GfK jest użycie webowego serwera pośredniczącego, np. narzędzia *Charles*. W ten sposób można przetestować poprawność pomiaru (wysyłanych zapytań) na laptopie lub komputerze stacjonarnym.

Aby skutecznie przeprowadzić proces testowania prawidłowości implementacji naszej biblioteki w Państwa aplikacji należy użyć tej samej sieci WiFi dla urządzenia, na którym testowana będzie aplikacja oraz dla PC/laptopa. Następnie należy ustawić PC/laptopa jako proxy na urządzeniu mobilnym (a także pamiętać o ustawieniu właściwego portu). Charles zapyta czy połączenie jest zaufane oraz zacznie monitorować dane wysyłane przez aplikację mobilną.

Opis procesu można znaleźć na stronie:

http://www.charlesproxy.com/documentation/faqs/using-charles-from-an-iphone/

Alternatywnie, inne serwery pośredniczące mogą być również stosowane w procesie debugowania, według tej samej procedury jak ta opisana powyżej.