SENSIC SITE TRACKER API

iOS App-Tagging Implementation Guide

PLaNET PBI GfK

Last change: 16.11.2015

Table of contents

1 Introduction 3

1.1 About this document 3

2 General concepts of SENSIC DAM 4

2.1 Page impression definition – “the golden rule” 4

3 How to integrate 5

3.1 Add Tracker Framework to your app code 5

3.2 Initiating the tracking library 5

3.3 Sending page impressions 5

3.4 Testing 6

4 Use cases to trigger a request 7

4.1 When to generate a tracking request in your app generally 7

4.2 How to define the page impression for your specific app 7

4.2.1 Code example 8

1 Introduction

1.1 About this document

This document contains **the documentation of the SENSIC Site Tracker API ("SST-A") for iOS apps**, which is used to measure the usage of apps for Apple iOS. This guide will provide you an overview of how the API looks like and how it is used for tracking of iOS apps.

It's structured in the following sections

2.) General concepts of SENSIC DAM

3.) How to integrate: This chapter describes how to integrate the SDK into your app technically

4.) Use cases and integration examples: Chapter 4 describes the uses cases in which a measurement request has to be generated. Also you find integration examples there.

2 General concepts of SENSIC DAM

Note: For a system overview, tracking parameters and privacy information please see the SENSIC administration guide.

2.1 Page impression definition – "the golden rule"

One of the base indexes to describe the reach of an online website or app is the page impression.

To assure the comparability of usage between the participating websites and apps it is important that all participants use the same definition of a page impression. Technically, every correct request on the sensic.net domain, done with the help of the SST or SSA code libraries, is registered as a page impression.

Due to this fact, it's important that all participants follow the global definition, in which cases a request (that leads to a page impression) has to be triggered. This definition is named as the "golden rule":

> *A **page impression** is defined as an **intentional interaction of the user** with a site, which **potentially** results in ad display (irrespective of the fact whether an ad is actually placed on that site or not). Every such action (user's intentional interaction with the site) **has to be counted only once**. User actions not leading to a potential display of a new ad, are not considered as page impressions.*

Generally speaking, a page impression is considered to be the event of a content change in a website (or mobile app) that exchanges most or parts of the content or information of the current view. Furthermore, page views can also cause a potential advert replacement. The event has to be intended by an active user interaction, that can be can be carried out with the help of a key, mouse or voice control, opening a browser (that is e.g. clicking on the start page) as well as reloading (refreshing) initiated by the user.

The usage of apps is slightly different from the usage of a website, due to more kinds of interaction and events, which can influence the usage scenario (e.g. an incoming call). For **apps** you can follow the practical guideline to generate a page impression whenever new content is rendered caused by a user interaction and this new content may potentially contain an ad.

The following events and actions are **not** page impressions:

- auto-refresh
- automatic content delivery
- automatic content delivery when closing the browser / application
- automatic routing on a page (except for aliases and redirects)
- scrolling
- searching within a page
- selecting content on a page (e.g. highlighting)
- activities generated by bots and the like
- switching between a browser's windows/tabs without displaying new content
- reloading iFrames

3 How to integrate

The following sections describe how to integrate the SST-A measurement library into your iOS app. Implementing and using the SENSIC Site Tracker library for iOS apps requires iOS 7.

3.1 Add Tracker Framework to your app code

Drag & Drop the folder "SST-SDK" into the Xcode project navigator view of your App. The folder should contain the following files:

- "libSST.a"
- "SST.h"

Click the project in the Project Navigator, click your app's target, then build settings, then search for "Other Linker Flags", click the + button, and add '-ObjC'.

3.2 Initiating the tracking library

You should create a single tracker instance in "didFinishLaunchingWithOptions" and reuse it in subsequent code either by using a global variable or as a member of another shared object.

During initializing, the tracking library downloads a configuration file from the SENSIC cloud backend.

```
#import "SST.h"

...

// Create a tracker instance:

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

tracker = [[SST alloc] initWithMediaId:@"yourMediaId"
andAdvertisingId:idfaString];

...
```

Set your **MediaID** here!

For successful initiation of the tracking library and measurement of the app usage, you have to set the correct **MediaID** provided to you by GfK.

3.3 Sending page impressions

A tracking request should be triggered by the SDK, whenever a *page view* happens in the app. You can trigger a tracking request like this:

```
// Sending events:

[tracker sendPageImpressionWithContentId:@"yourContentId"];
```

Set the **ContentID** here!

For successful measurement of the page impression, you have to set the corresponding ContentID provided to you by the media owner. **Please keep in mind that measurement requests without a ContentID or with a blank ContentID are ignored by the SENSIC backend and don't generate page impressions!**

To define when you have to generate a tracking request please refer to chapter *2.1 Page impression definition – "the golden rule"* and the examples in chapter *4 Use cases to trigger a request.*

3.4 Testing

Once the SST library has been added to your app, the best way to test or debug the tracking requests is by using a web debugging proxy such as *Charles*. This way, the measurement can be tested on PC or Laptop.

For successful debugging of the SST integration in your app using Charles, you should use the same WiFi for both your (test-) device and PC/Laptop and set up the PC as Proxy in your mobile device (plus, adding the correct port). After the first contact, Charles should ask if the connection can be trusted and will monitor all of the traffic which can be observed for debugging and testing.

A description of this process can also be found on the Charles website:
http://www.charlesproxy.com/documentation/faqs/using-charles-from-an-iphone/

Alternatively, other web debugging proxies can be used as well, following the same process as described.

4 Use cases to trigger a request

4.1 When to generate a tracking request in your app generally

The usage of apps is slightly different from the usage of a website. Apps provide more kinds of user interactions that can cause the reload of content. The event based architecture of mobile apps lead to the fact, that an app may influence other apps running on the device. The usage scenario isn't under complete control of the app and may be influenced by external events like an incoming call, push notices and so on.

In practice this leads to a broad range of how apps are implemented and how user interaction is handled by the app, so that it's impossible to predefine when to send a tracking request (that leads to a counted page impression) in a global manner.

The app owner, who knows his app, the user interactions and the behavior in several situations, is responsible for defining when to send a tracking request. To do this, you can follow this practical guideline:

> *Generate a tracking request whenever **new content is rendered** caused by a **user interaction** and this new content may **potentially contain an ad**.*

The "golden rule" of the page impression definition (see chapter *2.1*) contains several examples for actions and events, which are **not** page impressions. These events **must not create a tracking request**:

- auto-refresh of content
- automatic content delivery
- automatic content delivery when closing the browser / app
- scrolling
- searching within a page
- selecting content on a page (e.g. highlighting)
- activities generated by bots and the like
- switching between views and tabs without displaying new content

4.2 How to define the page impression for your specific app

The specific definition, which events represent a page view and generate a tracking request, have to be done by the app owner. To come to this specific definition, these steps may be helpful:

- write down a list of events that are handled by your app
- define in which situations the events are triggered
- Question 1: is the event caused by user interaction?
- Question 2: does the event lead to rendering new content?
- Question 3: may the new content potentially contain an ad?

If the three questions are answered with yes, you have to generate a tracking request at this point of your app.

This is an example table which may be used as template for the definition:

Event (Example)	In which situation?	Question 1: Caused by an user interaction?	Question 2: Is new content rendered?	Question 3: Content may potentially contain an ad	Trigger tracking request (Yes if Questions 1 – 3 are answered with yes)
ViewAppeared	Display start screen	Yes (App start)	Yes	Yes	**Yes**
ViewRefreshed	Timebased automatically refresh of newsfeed	No	Yes	Yes	**No**
	User pushes reload button in newsfeed	Yes	Yes	Yes	**Yes**
GestureShake	Newsfeed display (will be refreshed after shaking)	Yes	Yes	Yes	**Yes**
ViewDisappeared	App is going to background	Yes (Menu button pushed)	No	No	**No**
…					

4.2.1 Code example

The following code example sends a tracking request to the SENSIC backend with the example ContentID “sports”. This request may be generated for example in case of the iOS event “ViewAppeared”.

```
// Sending events:

[tracker sendPageImpressionWithContentId:@"sports"];
```