MAT JavaScript SDK

SDK Integration Guide

Version 1.0 | June 2013

Table of Contents

_____

Introduction

The MobileAppTracking (MAT) SDK for JavaScript provides basic application install and event tracking functionality. The JavaScript SDK is provided in the form of a single .js file that you simply include in your HTML file. To track installs, you must first integrate the JavaScript SDK with your app. You may also add and track additional events beyond an app install (such as purchases, game levels, and any other user engagement).

This document outlines the MAT JavaScript SDK integration and use cases.

Downloading the JavaScript SDK

The JavaScript SDK can be downloaded from the url:
https://s3.amazonaws.com/hasfiles/scripts/attribution.js

It is provided as a single .js file. You may download the file to integrate locally or use our hosted version.

Implementation

1. Add **attribution.js** to your app's js folder.

2. Include the JavaScript file as a script in your HTML file.

```
<script language="javascript" type="text/javascript"
    src="https://s3.amazonaws.com/hasfiles/scripts/attribution.js" />
```

3. You'll then need to initialize the "MobileAppTracker" class to allow you to call the functions from the SDK. Generally, the initialization should be called inside a function that runs on window.onload of the first screen of your app.

4. You will need to pass the following fields to the constructor:
- Advertiser ID (your_advertiser_id) of your app in the MobileAppTracking platform
- Package name/Bundle identifier
- App platform, either MobileAppTracker.PLATFORM_ANDROID or MobileAppTracker.PLATFORM_IOS
- Current app version (used for detecting updates)

```
MobileAppTracker.init("your_advertiser_id", "com.example.packagename",
        MobileAppTracker.PLATFORM_ANDROID, "1");
```

5. Now that you have added the SDK to your project, you can call the functions to track installs and other events. A sample implementation can be found here: https://github.com/MobileAppTracking/sdk-plugins/tree/master/sdk-js

Set Device Identifiers

The JavaScript SDK does not collect any device identifiers by default. You should set these before making any tracking calls, in order to attribute device installs and events to clicks.

Android
Android ID
Call MobileAppTracker.setAndroidId(), passing in the value of:
```
Secure.getString(getContentResolver(), Secure.ANDROID_ID);
```

Device ID (Requires Permission: READ_PHONE_STATE)
For phones, call MobileAppTracker.setDeviceId(), passing in the value of the IMEI or MEID from TelephonyManager's getDeviceId():
```
((TelephonyManager)
getSystemService(Context.TELEPHONY_SERVICE)).getDeviceId();
```

iOS
IFA
Call MobileAppTracker.setAppleAdvertisingIdentifier(), passing in the value:
```
ASIdentifierManager *adMgr = [ASIdentifierManager sharedManager];
NSString *appleAdId = [adMgr.advertisingIdentifier UUIDString];
```

IFV
Call MobileAppTracker.setAppleVendorIdentifier(), passing in the value:
```
NSString *appleVendorId = [[[UIDevice currentDevice]
identifierForVendor] UUIDString];
```

Ad Tracking Enabled
Call MobileAppTracker.setAppleAdTrackingEnabled(), passing in the value:
```
ASIdentifierManager *adMgr = [ASIdentifierManager sharedManager];
NSString *isEnabled = [NSString stringWithFormat:@"%d",
adMgr.advertisingTrackingEnabled];
```

4

Installs and Updates

As the success of attributing app events after the initial install is dependent upon first tracking that install, we require that the install is the first event tracked. To track the install of your mobile app, use the "trackInstall" method. If users have already installed your app prior to SDK implementation, then these users should be tracked as updates.

Track Install

The "trackInstall" method is used to track when users install your mobile app on their device and will only record one conversion per install in reports. You should call trackInstall() in the main load function after initializing the MobileAppTracker class.

```
MobileAppTracker.trackInstall();
```

The "trackInstall" method automatically tracks updates of your app if the app version differs from the last app version it saw.

Handling Installs Prior to SDK Implementation

What if your app already has thousands or millions of users prior to implementing the SDK? After integrating with the MAT SDK, the platform would record each user as a new app install when those users updated their app.

The MAT SDK provides you two methods to force an update event to be tracked instead of an install event:

1. "trackUpdate" instead of "trackInstall"
2. Import prior installs to the platform

These methods are useful if you already have a live app and plan to add the MAT SDK in a new version. Learn how to handle installs prior to SDK implementation here.

Events

The "trackAction" method is intended for tracking user actions after the install like reaching a certain level in a game or making an in-app purchase. The "trackAction" method allows you to define the event name dynamically.

Registration

If you have a registration process, it's recommended to track it by calling trackAction set to "registration".

```
MobileAppTracker.trackAction("registration");
```

You can find these events in the platform by viewing Reports > Event Logs. Then filter the report by the "registration" event.

Purchases

The best way to analyze the value of your publishers and marketing campaigns is to track revenue from in-app purchases. By tracking in-app purchases for a user, the data can be correlated back to the install and analyzed on a cohort basis to determine revenue per install and lifetime value.

Track In-App Purchases
The basic way to track purchases is to track an event with a name of purchase and then define the revenue (sale amount) and currency code.

```
trackAction(event name, revenue, currency)
```

Here is an example of tracking an event for purchase with revenue and currency code.

```
MobileAppTracker.trackAction("purchase",1.99, "USD");
```

Pass the revenue in as a number and the currency of the amount if necessary. Currency is set to "USD" by default. See Setting Currency Code for currencies we support.

You can find these events in the platform by viewing Reports > Event Logs. Then filter the report by the "purchase" event.

Opens

The SDK allows you to analyze user engagement by tracking unique opens. The SDK has built in functionality to only track one "open" event per user on any given day to minimize footprint. All subsequent "open" events fired on the same day are ignored and will not show up on the platform.

To track an "open" event you must pass in "open" as the parameter in the "trackAction" call:

```
MobileAppTracker.trackAction("open");
```

You can find counts of Opens by viewing Reports > Mobile Apps. Include the parameter of Opens to see the aggregated count. The platform does not provide logs of Opens. If you track Opens using a name other than "open" then these tracked events will cost the same price as all other events to track.

Other Events

You can track other events in your app dynamically by calling "trackAction". The "trackAction" method is intended for tracking any user actions. This method allows you to define the event name.

To dynamically track an event, replace "Event Name" with the name of the event you want to track. The tracking engine will then look up the event by the name. If an event with the defined name doesn't exist, the tracking engine will automatically create an event for you with that name. An event name has to be alphanumeric.

```
MobileAppTracker.trackAction("Event Name");
```

You can find these events in platform by viewing Reports > Event Logs.

The max event limit per site is 100. Learn more about the max limit of events.

Testing SDK

The SDK is designed to track conversions through the app store as well as from outside marketplaces or third parties. This allows you to test it without updating your production app listing. Subsequently, you will be able to update your mobile app for store availability after testing that tracking is fully operational - saving you time and resources. Learn how to test the SDK integration here:
[Testing Android SDK Integration](#)
[Testing iOS SDK Integration](#)

Debug Mode and Duplicates

The SDK provides features to help you debug and test the SDK integration. When the Debug mode is enabled in the SDK, the server responds with debug information about the success or failure of the tracking requests. Debug mode log output can be found in LogCat under the tag "MobileAppTracker". To debug log messages that show the event status and server response, call the "setDebugMode" method with Boolean true:

```
MobileAppTracker.setDebugMode(true);
```

Our tracking server typically rejects installs from devices it has seen before. For testing purposes, you may want to bypass this behavior and fire multiple installs from the same testing device. To enable this, call the "setAllowDuplicates" method after initializing MobileAppTracker, with Boolean true:

```
MobileAppTracker.setAllowDuplicates(true);
```

The setDebugMode and setAllowDuplicates calls are meant for use only during debugging and testing. Its required to disable these for production builds.

While the platform only allows one install, the threshold for duplicate events are set in the platform on a per event basis. Learn how to [Block Duplicate Requests for Events](#).

If you want to allow duplicate requests from the platform, you can do so by setting up a [test profile](#).

Additional Resources

Custom SDK Settings

The SDK supports several custom identifiers that you can use as alternate means to identify your installs or events. Please navigate to the Custom SDK Settings page.

Event Items

While an event is like your receipt for a purchase, the event items are the individual items you purchased. Event items allow you to define multiple items per a single event. The "trackAction" method can include this event item data. Learn how to track event items.

Unique Identifiers for Attribution

Unique Identifiers for Attribution can be set to maximize your ability to work with integrated ad networks and publisher partners.

Methodologies for Attributing Installs

When the platform receives a request to track an app install, it uses two methodologies for attributing installs, conducted in the following order:

1. Unique Identifier Matching
2. Device Fingerprinting

Learn more about Methodologies for Attributing Installs.