MAT iOS SDK v2.7.1

SDK Integration Guide

Version 2.7.1 | January 2014

Table of Contents

Introduction

The MobileAppTracking (MAT) SDK for Apple iOS provides basic application install and event tracking functionality. The iOS SDK is provided in the form of a framework that you simply include in your iOS project. Our SDK is compatible with iPhone®, iPad® and iPod Touch® devices. To track installs, you must first integrate the iOS SDK with your Apple app. You may also add and track additional events beyond an app install (such as purchases, game levels, and any other user engagement) as well as app “opens” to determine user engagement.

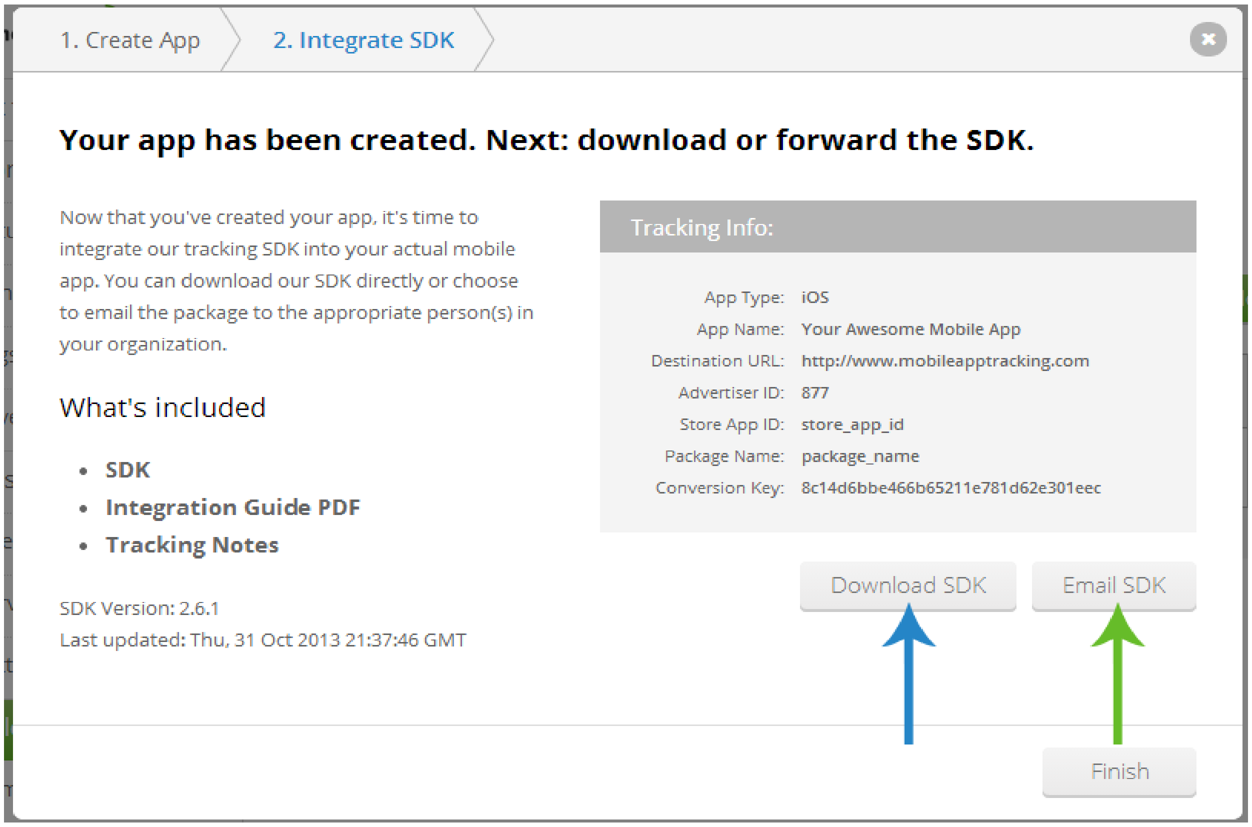
This document outlines the MAT iOS SDK integration.

Compatibility

The current version of the MAT iOS SDK is compatible with iOS 4.3 and above using Xcode 4.5. The SDK is compatible with ARC and non-ARC projects.

Downloading the iOS SDK

There are several methods you can use to download the MAT iOS SDK. The main method of obtaining the MAT iOS SDK is directly downloading it during the second step (“Integrate SDK”) of the “Add Mobile App” once you have logged into the MobileAppTracker platform. On this same page, you also have the ability to have the SDK emailed to any email address you provide. See the screenshot below for reference.


1. Create App
2. Integrate SDK

Your app has been created. Next: download or forward the SDK.

Now that you've created your app, it's time to integrate our tracking SDK into your actual mobile app. You can download our SDK directly or choose to email the package to the appropriate person(s) in your organization.

What's included

- **SDK**
- **Integration Guide PDF**
- **Tracking Notes**

SDK Version: 2.6.1
Last updated: Thu, 31 Oct 2013 21:37:46 GMT

Tracking Info:

App Type:	iOS
App Name:	Your Awesome Mobile App
Destination URL:	http://www.mobileapptracking.com
Advertiser ID:	877
Store App ID:	store_app_id
Package Name:	package_name
Conversion Key:	8c14d6bbe466b65211e781d62e301eec

Once you’ve downloaded the MAT iOS SDK, decompress the .zip file and extract the files to your development computer. The MAT iOS SDK consists of a framework folder that makes it easier to integrate into your mobile app and contains the necessary header file as well as appledoc style documentation.

Implementation

1. Create a new project in Xcode.

2. In Xcode, Build Phases → Link Binary with Libraries, link the following files:
   1. MobileAppTracker.framework.
      Be sure to choose the plus to “add” the file to your project. This will create the necessary linking for a static library in Xcode. Additionally, you can drag and drop the framework file into the list of libraries you want to link.
   2. CoreTelephony.framework
   3. SystemConfiguration.framework
   4. MobileCoreServices.framework
   5. iAd.framework
   6. AdSupport.framework (set it's status as Optional if your deployment target is < iOS 6.0)

3. Open your AppDelegate.m file.

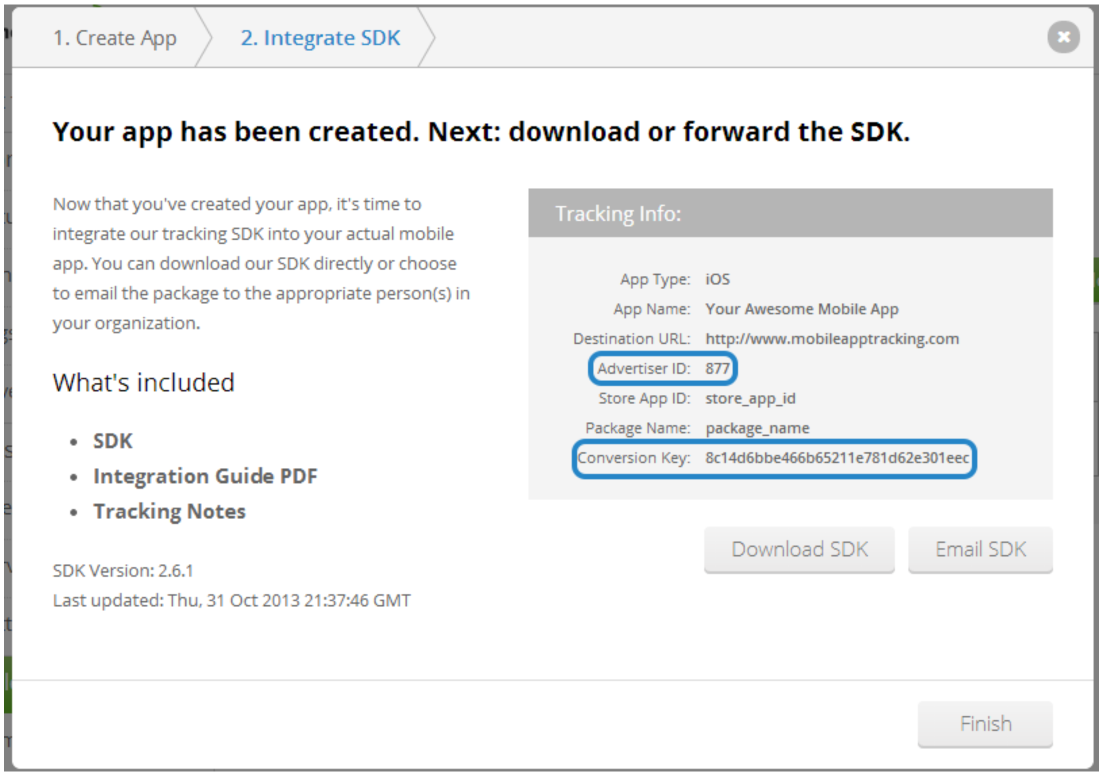
4. Add the following to the top of your project:

```
#import <MobileAppTracker/MobileAppTracker.h>
```

5. Initialize the MobileAppTracker.m class by pasting the following inside your "application:didFinishLaunchingWithOptions" method:

```
NSString * const MAT_CONVERSION_KEY = @"your MAT conversion key";
[[MobileAppTracker sharedManager] startTrackerWithMATAdvertiserId:MAT_ADVERTISER_ID
                                          MATConversionKey:MAT_CONVERSION_KEY];
```

The "MAT_ADVERTISER_ID" and the "MAT_CONVERSION KEY" values correlate to the ***Advertiser Id*** and ***Conversion Key*** provided to you when you created the Mobile App (Step 3 in "Create Mobile App") in platform. See screenshots below for reference.



Your app has been created. Next: download or forward the SDK.

Now that you've created your app, it's time to integrate our tracking SDK into your actual mobile app. You can download our SDK directly or choose to email the package to the appropriate person(s) in your organization.

What's included

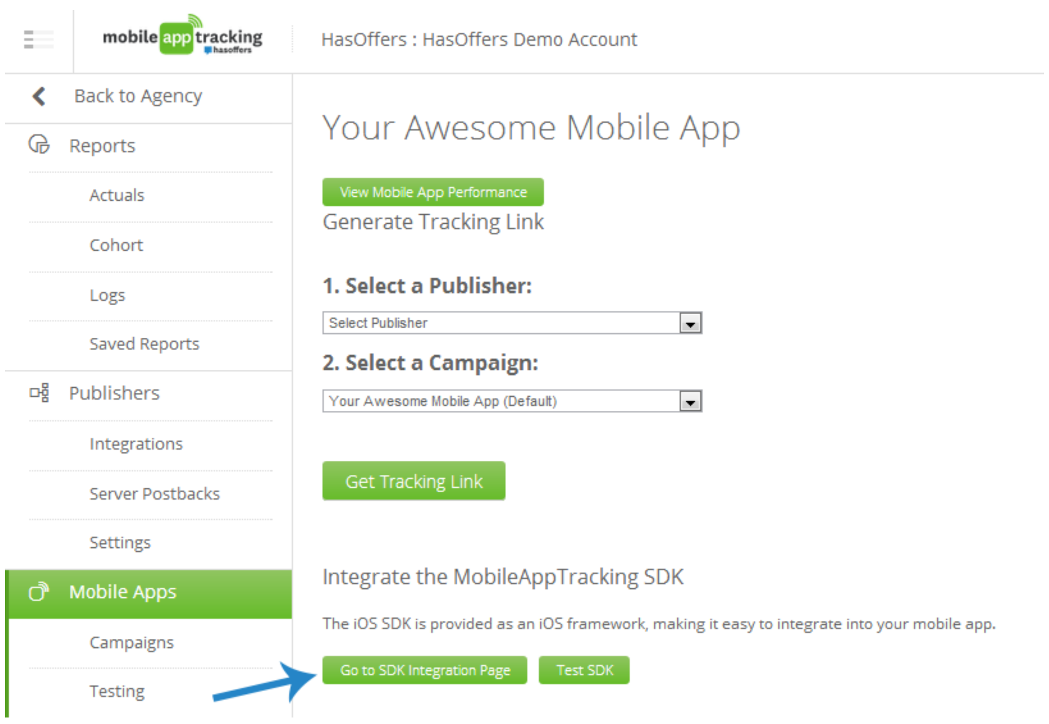
- SDK
- Integration Guide PDF
- Tracking Notes

SDK Version: 2.6.1
Last updated: Thu, 31 Oct 2013 21:37:46 GMT

Tracking Info:

App Type:	iOS
App Name:	Your Awesome Mobile App
Destination URL:	http://www.mobileapptracking.com
Advertiser ID:	877
Store App ID:	store_app_id
Package Name:	package_name
Conversion Key:	8c14d6bbe466b65211e781d62e301eec

These values may also be found on the "Tracking Code for XXXApp" page by clicking on the "Download SDK" button and clicking on the appropriate mobile app. See screenshots below for reference.



Your Awesome Mobile App

Generate Tracking Link

1. Select a Publisher:

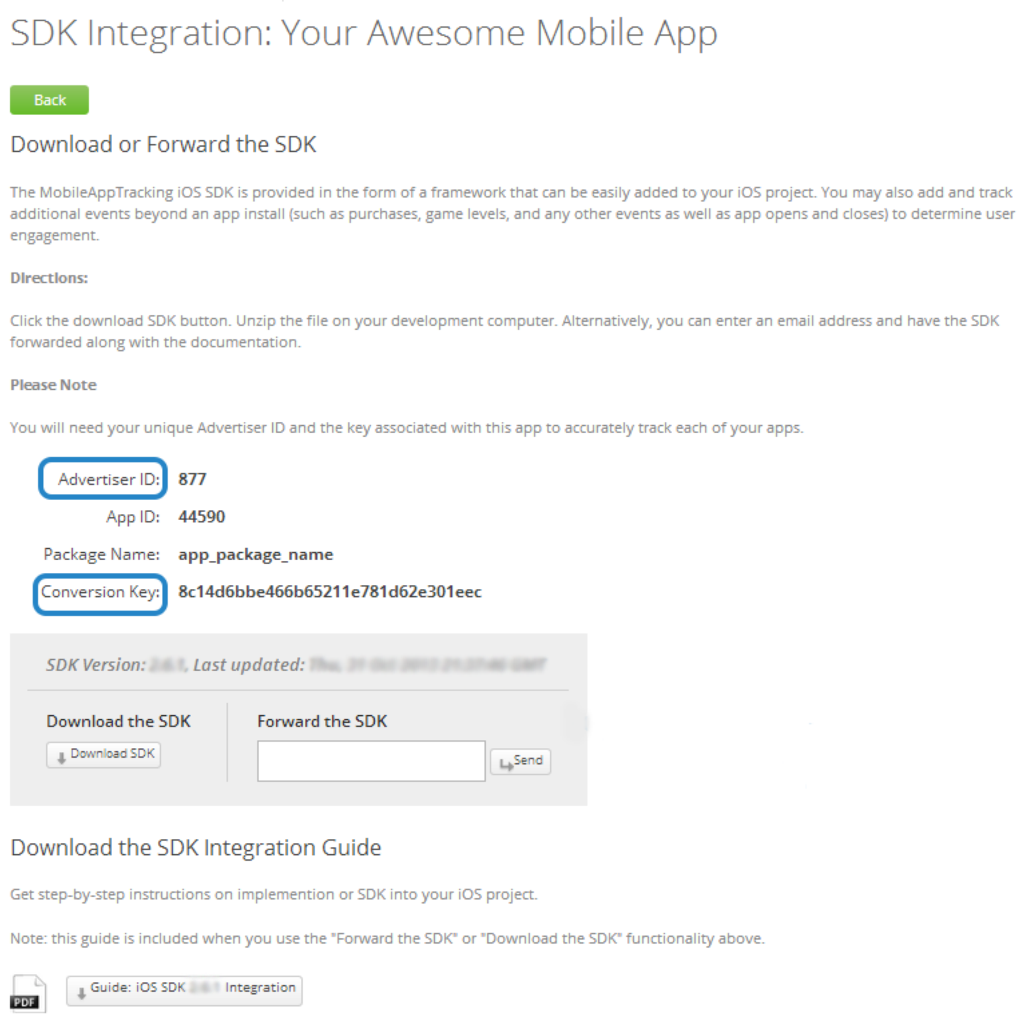
Select Publisher

2. Select a Campaign:

Your Awesome Mobile App (Default)

Integrate the MobileAppTracking SDK

The iOS SDK is provided as an iOS framework, making it easy to integrate into your mobile app.



SDK Integration: Your Awesome Mobile App

Download or Forward the SDK

The MobileAppTracking iOS SDK is provided in the form of a framework that can be easily added to your iOS project. You may also add and track additional events beyond an app install (such as purchases, game levels, and any other events as well as app opens and closes) to determine user engagement.

Directions:

Click the download SDK button. Unzip the file on your development computer. Alternatively, you can enter an email address and have the SDK forwarded along with the documentation.

Please Note

You will need your unique Advertiser ID and the key associated with this app to accurately track each of your apps.

Advertiser ID:	877
App ID:	44590
Package Name:	app_package_name
Conversion Key:	8c14d6bbe466b65211e781d62e301eec

SDK Version: [redacted]*, Last updated:* [redacted]

Download the SDK

Forward the SDK

Download the SDK Integration Guide

Get step-by-step instructions on implemention or SDK into your iOS project.

Note: this guide is included when you use the "Forward the SDK" or "Download the SDK" functionality above.

6. You are now ready to call the functions to track installs and other events. ***IMPORTANT***: Always call "trackInstall" or "trackUpdate" if the user is either installing the app for the first time or updating their app after a previous install.

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 install of your iOS 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. We recommend calling "trackInstall" after instantiating the "startTracker" method in which your "advertiser ID" and "conversion key" are given.

```
[[MobileAppTracker sharedManager] 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 SDK implementation? What happens when these users update the app to the new version that contains the MAT SDK?

MAT provides you two ways to make sure that the existing users do not count towards new app installs.

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

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

If the code used to differentiate installs versus app updates is not properly implemented, then you will notice a spike of total installs on the first day of the SDK implementation.

Sample Install Code

Now that you know how “trackInstall” vs. “trackUpdate” works, your AppDelegate.m file should now look similar to below.

The sample install code below also includes debugging code. For more details on how debugging works, please refer to Debug Mode and Duplicates.

Only on devices with OS versions lower than iOS 7, it is recommended to set the MAC address, otherwise publishing partners that rely on MAC Address for attribution will be impacted. You should generate the MAC address with the format “MM:MM:MM:SS:SS:SS” and set it in MAT using setMACAddress: before the first trackXXX method is called.

```
-(BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:
(NSDictionary *)launchOptions
{
   // Override point for customization after application launch.

   // Set this class as the delegate to receive response from platform
   // Make sure that you have this class adopt the protocol MobileAppTrackerDelegate
   // [[MobileAppTracker sharedManager] setDelegate:self];

   // REQUIRED
   // The minimum data needed to initialize the MobileAppTracker is: advertiser_id,
conversion_key
   [[MobileAppTracker sharedManager] startTrackerWithMATAdvertiserId:MAT_ADVERTISER_ID
                                                        MATConversionKey:MAT_CONVERSION_KEY];

    // Strongly recommended: set the Apple IFA in MAT
   NSUUID *ifa = [[ASIdentifierManager sharedManager] advertisingIdentifier];
   [[MobileAppTracker sharedManager] setAppleAdvertisingIdentifier:ifa];

     // Recommended: When iOS ver < 7.0, generate and include the device MAC address in
MAT
    // NSString *macAddress = @"<device mac address MM:MM:MM:SS:SS:SS generated by you>”;
    // [[MobileAppTracker sharedManager] setMACAddress:macAddress];

#if DEBUG
      // FOR DEBUG ONLY: Turn this on to see debug messages
      //[[MobileAppTracker sharedManager] setDebugMode:YES];

      // FOR DEBUG ONLY: Turn this on to allow duplicate events
      //[[MobileAppTracker sharedManager] setAllowDuplicateRequests:YES];
#endif
```

```
    // Optional user identifier that that you generate that may match with your internal
systems
    // NSString *userId = @"your_internal_id_for_the_user”;
    // [[MobileAppTracker sharedManager] setUserId:userId];

    // track first app open as Install
    [[MobileAppTracker sharedManager] trackInstall];
    return YES;
}
```

Events

After the install has been tracked, the “trackAction” method is intended to be used to track user actions such as — reaching a certain level in a game or making an in-app purchase. The “trackAction” method allows you to dynamically define the event name.

Registration

If you have a registration process, its recommended to track it by calling trackAction set to “registration”.

```
[[MobileAppTracker sharedManager] trackActionForEventIdOrName:@”registration"
eventIsId:NO];
```

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

While our platform always blocks the tracking of duplicate installs, by default it does not block duplicate event requests. However, a registration event may be an event that you only want tracked once per device/user. Please see block duplicate requests setting for events for further information.

Irrespective of the Block Duplicate Length setting referred to in the above link, duplicate event tracking requests are allowed if you call setAllowDuplicateRequests:YES in code during testing or have an active Test Profile for the device/simulator being used.

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

```
[[MobileAppTracker sharedManager] trackActionForEventIdOrName:@"purchase"
                                                     eventIsId:NO
                                                 revenueAmount:10.00f
                                                  currencyCode:@"GBP"];
```

Note: Pass the revenue in as a Double 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 platform by viewing Reports > Logs > Events. Then filter the report by the “purchase” event.

Track App Store Purchase State

The SDK also allows you to track purchase events that occur inside your app by tying in your events to iTunes in-App Purchase system. Learn about tracking purchase events with Apple iTunes in-App Purchases here.

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” in the trackAction call. The following code snippet shows a sample implementation for use in AppDelegate.m.

```
- (void)applicationDidBecomeActive:(UIApplication *)application
{
   [[MobileAppTracker sharedManager] trackActionForEventIdOrName:@"open" eventIsId:NO];
}
```

When your app is launched from some other app, e.g. Safari, you receive a callback in the UIApplicationDelegate. Its recommended you track the referrer app package name and url by calling the SDK method:

```
- (void)applicationDidOpenURL:(NSString *)urlString sourceApplication:(NSString
*)sourceApplication;
```

The following code snippet shows a sample implementation for use in AppDelegate.m.

```
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:
(NSString *)sourceApplication annotation:(id)annotation
{
   [[MobileAppTracker sharedManager] applicationDidOpenURL:[url absoluteString]
sourceApplication:sourceApplication];
   return YES;
}
```

The applicationDidOpenURL:sourceApplication method is independent of the “open” event and may be called multiple times on the same date.

You can see the aggregated count of Opens by going to Reports > Actuals, clicking Edit and enabling the "Opens" checkbox. 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 or id” 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 sharedManager] trackActionForEventIdOrName:@"event name or id"
                                                    eventIsId:YES/NO];
```

You can pass in an event name or event id. If you pass in an event name and eventIsId:NO, then you are indicating to the SDK that you want your own event name passed in. If you pass in an event id and eventIsId:YES, then you are indicating that you have a pre-defined event id in the platform that you associate the action with.

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

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

While our platform always blocks the tracking of duplicate installs, by default it does not block duplicate event requests. However, there may be other types of events that you only want tracked once per device/user. Please see block duplicate requests setting for events for further information.

Testing SDK

The SDK is designed to track conversions through the Apple iTunes AppStore as well as from outside market places 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 iTunes AppStore after testing that tracking is fully operational - saving you time and resources. Learn how to test the iOS SDK integration here.

After you are done testing, you should review the iOS SDK Implementation Troubleshooting guide before submitting your app with the SDK.

Debug Mode and Duplicates

Debugging

When the debug mode is enabled in the SDK, the platform responds with debug information about success or failure of the tracking requests. You can implement the MobileAppTrackerDelegate and and set the "setDebugMode" to receive the receive the response from the platform.

```
#if DEBUG
   // enable debug mode
   [[MobileAppTracker sharedManager] setDebugMode:YES];
#endif
```

Allow Duplicates

The platform 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. There are two ways to do that:
(1) Call the "setAllowDuplicateRequests:" method in code

- easy to control in source code
- must not be used in app store builds

```
#if DEBUG
   // allow duplicate requests from the same device
   [[MobileAppTracker sharedManager] setAllowDuplicateRequests:YES];
#endif
```

(2) Set up a test profile.

- no code change required, no need to call “setAllowDuplicateRequests:”
- can be controlled from the platform
- lets you fire duplicate test requests from released apps

To test multiple installs from the same device/simulator, you first have to delete the existing app from the device/simulator and reinstall it, since the MAT SDK allows only one install request to be fired per app installation. Other in-app events may be fired multiple times from the same app installation.

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

Viewing Server Responses

You can implement the optional callback methods in the protocol MobileAppTrackerDelegate to receive success/failure messages from the server for each SDK tracking request. This comes in handy when debugging MAT SDK integration.

The following sample code shows how AppDelegate class can implement the MobileAppTrackerDelegate protocol. The code assumes an ARC project.

In AppDelegate.h:

```
#import <MobileAppTracker/MobileAppTracker.h>
@interface AppDelegate : UIResponder <UIApplicationDelegate, MobileAppTrackerDelegate>
```

In AppDelegate.m:
In the application:didFinishLaunchingWithOptions: method

```
#if DEBUG
    // enable debug mode
    [[MobileAppTracker sharedManager] setDebugMode:YES];

    // allow duplicate requests from the same device
    [[MobileAppTracker sharedManager] setAllowDuplicateRequests:YES];
#endif

    // set this class as the delegate for MAT callbacks
    [[MobileAppTracker sharedManager] setDelegate:self];
```

Then implement the MAT delegate callback methods in the AppDelegate.m file:

```
#pragma mark - MobileAppTrackerDelegate Methods
// MAT tracking request success callback
- (void)mobileAppTracker:(MobileAppTracker *)tracker didSucceedWithData:(id)data
{
   NSString *response = [[NSString alloc] initWithData:data
encoding:NSUTF8StringEncoding];
   NSLog(@"MAT.success: %@", response);
}
// MAT tracking request failure callback
- (void)mobileAppTracker:(MobileAppTracker *)tracker didFailWithError:(NSError *)error
{
   NSLog(@"MAT.failure: %@", error);
}
```

Now when a server request completes, the custom code in your success / failure callback methods — mobileAppTracker:didSucceedWithData: / mobileAppTracker:didFailWithError: methods — will be run.

Version History

Latest Version Changes

v2.7.1 changes

- sdk does not auto-generate IFA, only provides a setter for the same
- adds support to display iAd banners
- requires iAd.framework
- list of new methods:

```
- (void) displayiAdInView:(UIView *)view;
- (void) removeiAd;
```

- list of new delegate callback methods:

```
- (void)mobileAppTrackerDidDisplayiAd;
- (void)mobileAppTrackerDidRemoveiAd;
- (void)mobileAppTrackerFailedToReceiveiAdWithError:(NSError *)error;
```

- list of deleted methods:

```
+ (void)setShouldAutoGenerateAppleAdvertisingIdentifier:(BOOL)yesorno;
```

v2.7 changes

- added setter methods that allow collecting user's Facebook, Google, Twitter Ids
- list of new methods:

```
- (void)setFacebookUserId:(NSString *)facebookUserId;
- (void)setGoogleUserId:(NSString *)googleUserId;;
- (void)setTwitterUserId:(NSString *)twitterUserId;
```

- list of deprecated methods:

```
- (void)setMATAdvertiserId:(NSString *)advertiserId;
- (void)setMATConversionKey:(NSString *)conversionKey;
- (void)setMACAddress:(NSString *)macAddress;
- (void)setODIN1:(NSString *)odin1;
- (void)setOpenUDID:(NSString *)openUDID;
- (void)setUIID:(NSString *)uiid;
```

Previous Version changes

v2.6.1 changes

- added support for builds that target arm64 architecture (64-bit devices)

v2.6 changes

- **added simulator support for Xcode 5, iOS 7**

v2.5.3 changes

- fixed applicationDidOpenURL:sourceApplication: method
- always use HTTPS, removed setUseHTTPS: method

v2.5.2 changes

- fixed applicationDidOpenURL:sourceApplication: method

v2.5.1 changes

- **critical: fixed request queuing used in case there is no network connectivity**

- **renamed helper methods that could possibly cause duplicate symbol errors**
- **replaced enum MATGender with corresponding integer constants**

v2.5 changes

- added trackAction method that accepts iTunes IAP transaction receipt to allow MAT server-side IAP transaction verification
- added setter to enable/disable App Level Ad-Tracking

```
- (void)setAppAdTracking:(BOOL)enable;
```

- added setter for UIID, an alternative for UDID

```
- (void)setUIID:(NSString *)uiid;
```

- stopped auto-generating device unique identifiers in SDK, added corresponding setter methods — MAC, ODIN1, OpenUDID, TRUSTe TPID, UIID
- removed methods:

Note: please remove the following setShouldAutoGenerateXXX method calls

```
- (void)setShouldAutoGenerateMacAddress:(BOOL)yesorno;
- (void)setShouldAutoGenerateODIN1Key:(BOOL)yesorno;
- (void)setShouldAutoGenerateOpenUDIDKey:(BOOL)yesorno;
```

- added setter methods:

```
- (void)setMACAddress:(NSString *)macAddress;
- (void)setODIN1:(NSString *)odin1;
```

v2.4.2 changes

- removed CoreGraphics framework dependency

v2.4.1 changes

- fixed user agent collection logic
- require CoreGraphics framework inclusion

v2.4 changes

- added support for recording user info — age, gender
- added support for recording user location — latitude, longitude, altitude
- event items now provide 5 additional attributes
- blocked tracking calls for “close” event
- removed error param from start method.

deprecated method:

```
- (BOOL)startTrackerWithMATAdvertiserId:(NSString *)aid MATConversionKey:(NSString *)key
withError:(NSError **)error;
```

added method:

```
- (BOOL)startTrackerWithMATAdvertiserId:(NSString *)aid MATConversionKey:(NSString *)key;
```

- deprecated the use of event item dictionaries in trackAction methods; use MATEventItem objects instead

v2.3 changes

- critical: fixed issue where offline queued installs/events were lost when app was killed and restarted
- fixed setAdvertiserIdentifier: to use the NSString representation of the NSUUID
- fixed issue where event items with non-English unicode characters would not get logged, e.g. ü, ç
- to count daily unique users allow only one “open” event on any given date
- the sdk now fires all asynchronous network requests on the main thread
- the sdk makes sure that the data stored by the sdk is not backed up to iCloud
- route debug requests to debug.engine.mobileapptracking.com domain
- show warning alert dialog when debug mode is enabled
- added methods setJailbroken: and setShouldAutoDetectJailbroken:
- removed method setDeviceId: that allowed you to set the Apple device uniqueIdentifier.
    - Note: Code change needed: Please remove any setDeviceId: method calls.
- list of deprecated methods:

```
- (BOOL)startTrackerWithAdvertiserId:(NSString *)aid advertiserKey:(NSString *)key
withError:(NSError **)error;
- (void)setAdvertiserId:(NSString *)advertiser_id;
- (void)setAdvertiserKey:(NSString *)advertiser_key;
- (void)setAdvertiserIdentifier:(NSUUID *)advertiser_identifier;
- (void)setVendorIdentifier:(NSUUID * )vendor_identifier;
- (void)setShouldAutoGenerateAdvertiserIdentifier:(BOOL)yesorno;
- (void)setShouldAutoGenerateVendorIdentifier:(BOOL)yesorno;
- (void)setShouldDebugResponseFromServer:(BOOL)yesorno;
- (void)setShouldAllowDuplicateRequests:(BOOL)yesorno;
```

- list of new methods:

```
- (BOOL)startTrackerWithMATAdvertiserId:(NSString *)aid MATConversionKey:(NSString *)key
withError:(NSError **)error;
- (void)setMATAdvertiserId:(NSString *)advertiser_id;
- (void)setMATConversionKey:(NSString *)conversion_key;
- (void)setAppleAdvertisingIdentifier:(NSUUID *)advertising_identifier;
- (void)setAppleVendorIdentifier:(NSUUID * )vendor_identifier;
- (void)setShouldAutoGenerateAppleAdvertisingIdentifier:(BOOL)yesorno;
- (void)setShouldAutoGenerateAppleVendorIdentifier:(BOOL)yesorno;
- (void)setDebugMode:(BOOL)yesorno;
- (void)setAllowDuplicateRequests:(BOOL)yesorno;
```

v2.2 changes

- if the internet is not reachable, then queue the request right away for retry later
- changed MobileAppTracker delegate return data from id to NSData to enable strong typing of the return data

- added delegate callbacks when app-to-app tracking request returns
- bug: event name casing was not preserved when sending the event to the server. Example: event name of “Purchase” was being changed to lowercase “purchase”
- bug: reference id was not being sent with install or update. trackInstallWithReferenceId and trackUpdateWithReferenceId were not sending up reference id to the server and reports were not showing Advertiser Ref Id values because of this.
- change: NSURL timeout for connection changed to 60 seconds as a default.

v2.1 changes

- new trackInstall and trackUpdate methods replacing the trackInstallWithUpdateOnly method
- new in app purchase tracking
- supports both compilers — Apple LLVM and LLVM-GCC
- fixed: removed revenue amount from parameters after a track action method call so revenue does not get inadvertently sent with other actions
- fixed: updated app to app tracking to support only one set tracking method and fixed a bug with setting package name
- fixed: defaulting to HTTPS was not working
- ***NOTE*** the Bundle Identifier for your app target in Xcode must match the Bundle Name in the MAT console. Otherwise, event tracking will not work.
- new optional site id parameter to identify your app to the tracking engine

v2.0 changes

- ***NOTE*** You must call trackInstall or trackUpdate in your code. These methods are new for v2.0. Previous versions of the SDK implicitly called trackInstall or trackUpdate
- simplified set methods for passing in data and setting options
- simplified start method, just pass in advertiser id, key
- control auto generation of data like mac address and other identifiers
- improved and simplified track action methods
- create custom track action events with event items to track in-app purchases
- new support for Re-Engagement
- consolidation to a single download for the framework static library file, support for arc and non-arc projects in one framework file
- sample Xcode project
- support for iOS® 4.3 and above
- support for iPhone 5 and iOS 6
- uses new iOS 6 Advertiser Identifier and Vendor Identifier
- App-to-App Tracking, track an install from your app to other apps you’ve integrated with MAT SDK

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 for a single event. The “trackAction” method can include this event item data and up to 5 custom string attributes. Learn how to track event items.

App to App Tracking

App to App tracking provides the ability for one app (the referring app) to download another app (the target app). The target app will then record an install event that contains data from the referring app. Also, you can specify that your app (AppA - referring app) redirect to the link where AppB (target app) can be downloaded (typically this is Google Play). Learn about App to App Tracking here.

Track Referrer Application (Re-Engagement)

Some app ads will open your app if already installed on the device. When your app is launched from some other app - e.g. Safari - you receive a callback in the “UIApplicationDelegate”. To track “Re-engagement”, you can track the referrer app package name and URL.

Learn how to track Re-Engagement here.

Track Events in WebView

Unique Identifiers for AttributionIf your app contains HTML content and you would like to track events in the WebView, you can do this by overriding your WebViewClient. Then handle the URL loading to find the URL request that corresponds with your event and call a trackAction before it loads. Learn how to Track Events with SDK in WebView.

Unique Identifiers for Attribution

By default the iOS SDK collects these to maximize your ability to work with integrated ad networks and publisher partners.

Methodologies for Attributing Installs

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

1. Unique Identifier Matching
2. Device Fingerprinting

Learn more about Methodologies for Attributing Installs.

Detecting Jailbroken Devices

By default the MAT SDK detects if the iOS device has been jailbroken. You can override this using Methodologies for Attributing Installsthe methods setJailbroken: and setShouldAutoDetectJailbroken:.

```
[[MobileAppTracker sharedManager] setJailbroken:YES];
```

```
[[MobileAppTracker sharedManager] setShouldAutoDetectJailbroken:YES];
```

Learn more about the techniques used by the SDK to detect jailbroken devices.

App Level Ad Tracking Opt-Out

MAT allows apps to let their users opt-out of app level ad tracking. If a user chooses to opt-out, then the install or conversion event will be used only for ad reporting and not for ad optimization. By default ad tracking is enabled (opt-in).

To opt-in (default), the following setter method may be called:

```
[[MobileAppTracker sharedManager] setAppAdTracking:YES];
```

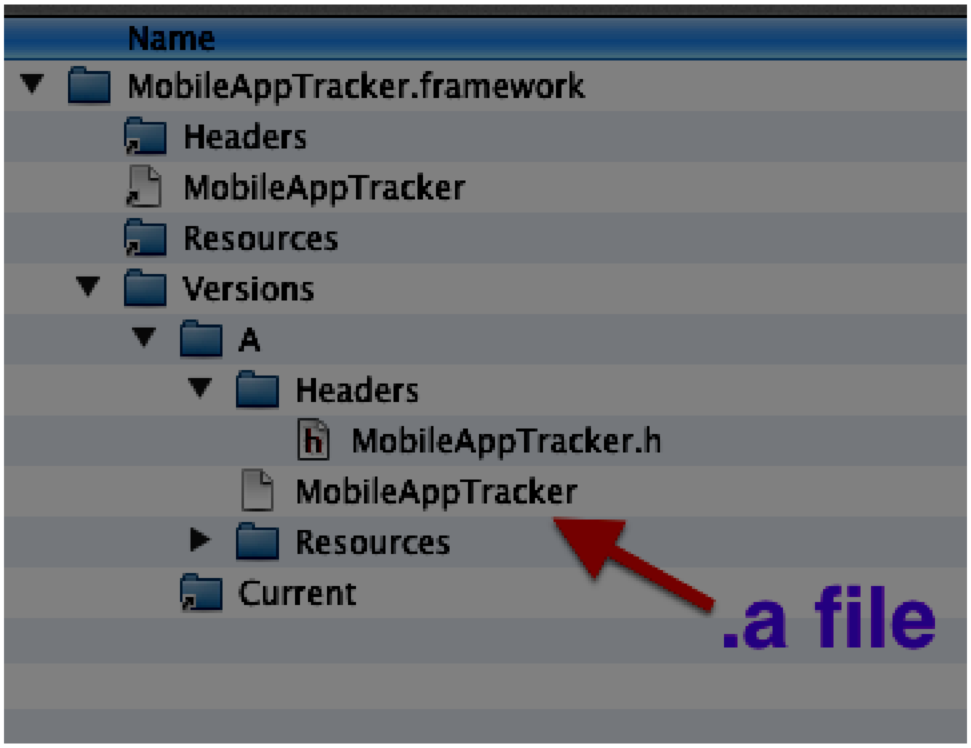
To opt-out, the following setter method may be called:

```
[[MobileAppTracker sharedManager] setAppAdTracking:NO];
```

MobileAppTracker.a file

If you are using an external SDK such as Corona, then you might not be able to directly include the MobileAppTracker.framework. In that case you need to directly include the MobileAppTracker.a file.

The MobileAppTracker.framework is a folder structure that wraps the MobileAppTracker.a and MobileAppTracker.h files. To access the MobileAppTracker.a file, open Finder and navigate to the MobileAppTracker.framework/Versions/A folder, copy out the MobileAppTracker file and rename it to MobileAppTracker.a.



SDK Implementation Troubleshooting

If you find yourself having trouble implementing our SDK, please refer to our [SDK Implementation Troubleshooting] article.