AppsFlyer SDK Integration - iOS V3.3.3

Print / PDF

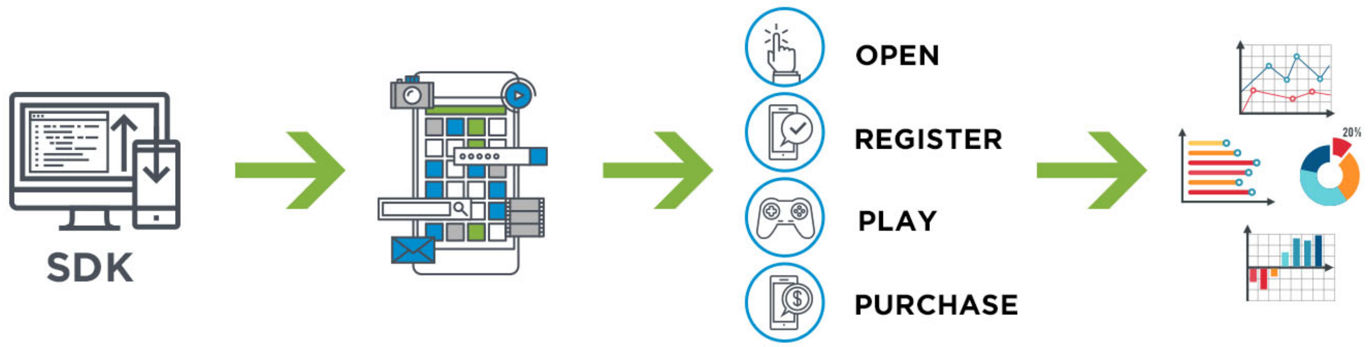
Jamie Weider
Last update: November 27, 2018 15:49

Page Contents:

This iOS SDK is out of date! For the latest version, click here.

Current Version: v3.3.3

AppsFlyer's SDK provides app installation and event tracking functionality. We have developed an SDK that is highly robust (8+ billion SDK installations to date), secure, lightweight and very simple to embed.



You can track installs, updates and sessions (by following the mandatory steps below), and also track additional in-app events beyond app installs (including in-app purchases, game levels, etc.) to evaluate ROI and user engagement levels.

The mandatory steps are explained below, followed by additional optional features.

The iOS SDK is compatible with all iOS devices (iPhone, iPod, iPad) with iOS version 6 and above (including 9).

1. What's New in this Version?

- ✓ Enabling Extension Support
- ✓ Support iOS 9
- ✓ Support TLS 1.2 for iOS 9
- ✓ In-App Purchase Receipt Validation - Updated this API to support the transaction validation as well
- ✓ The SDK is available both as a framework and as a static library

2. Embed the SDK into Your Application (Mandatory)

AppsFlyer SDK is available both as a framework and as a static library.

AppsFlyer SDK Framework

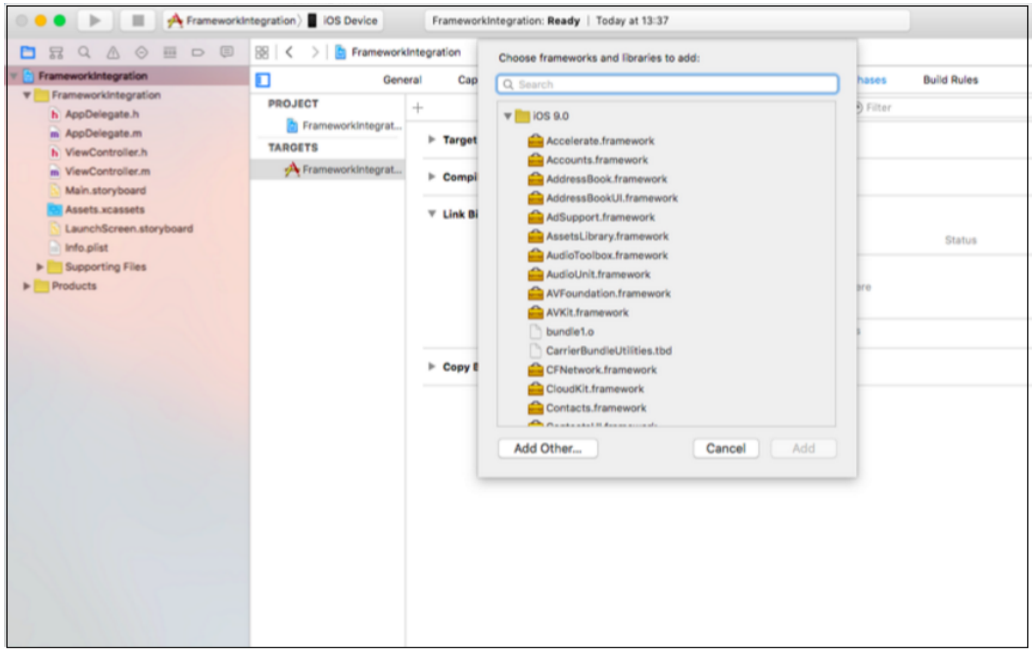
The simplest way to integrate the framework is using cocoapods:

- ✓ Add the following line to your pod file:

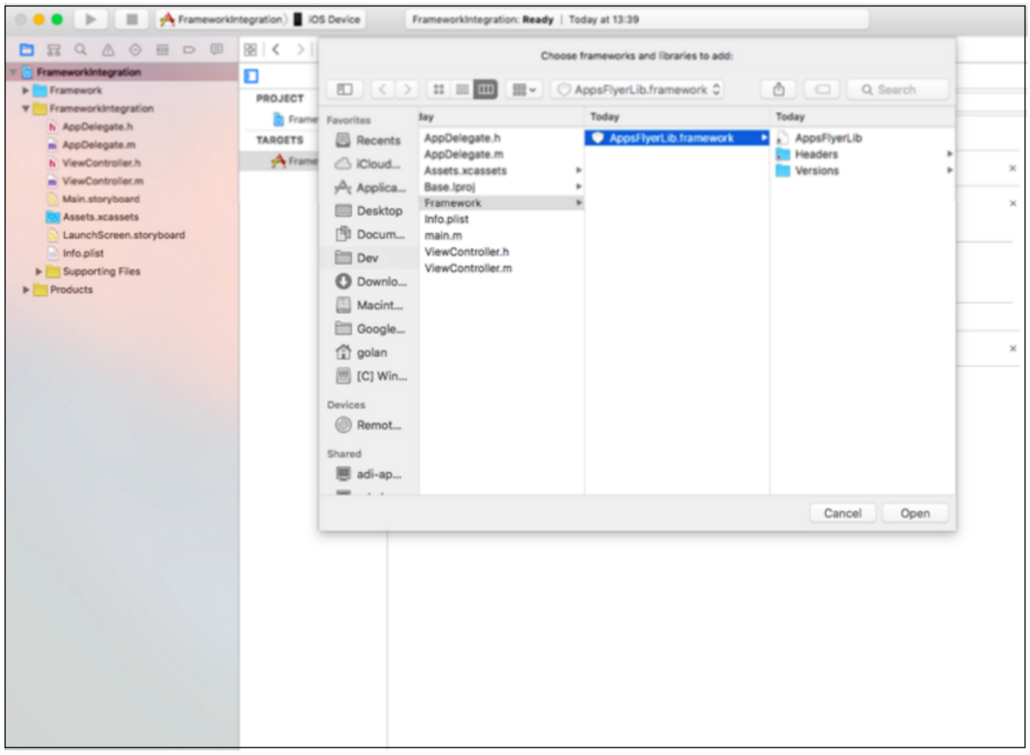
```
pod 'AppsFlyerFramework'
```

If you do not use cocoapods, follow the steps below:

In Xcode, go to **Build Phases** >> **Link Binary with Libraries** >> Click **+** sign to add a new library



Click **Add Other** and add the **AppsFlyerLib.framework**



Include the following header:

```
#import "AppsFlyerLib/AppsFlyerTracker.h"
```

2.1 AppsFlyer SDK Static Library

The simplest way to integrate the static library is by using cocoapods:

- ✓ Add the following line to your pod file:

```
pod 'AppsFlyer-SDK'
```

2.2 IDFA

If you do not use cocoapods, follow the steps below:

- ✓ Add the header and lib files to your project.

AppsFlyer SDK components (**libAppsFlyerLib.a & AppsFlyerTracker.h**) are availablehere.

- ✓ Add the AppsFlyer header import:

```
#import "AppsFlyerTracker.h"
```

- ✓ Add the **AdSupport.framework** to your project and set it as “Optional”. You can find the submission instructions here.

NOTE: AppsFlyer collects IDFA only if you include **AdSupport.framework**

- ✓ Add the **iAd.framework** to your project.

3. SDK Initialization & Installation Event (Minimum Requirement for Tracking)

NOTE**: This is the minimum requirement to start tracking your app installs.**

You must initialize the SDK on the first launch of the app. Make sure that the SDK is initialized before sending the tracking event below.

3.1 Initializing the SDK

To initialize the SDK, add the following to your **“didFinishLaunchingWithOptions”**function:

```
[AppsFlyerTracker sharedTracker].appsFlyerDevKey = @"[Dev_Key]";
[AppsFlyerTracker sharedTracker].appleAppID = @"REPLACE THIS WITH YOUR App
```

Replace [**Dev_Key]** with your own **Dev_Key** (accessible from your account, see **Settings**>> **Integrate the SDK into...** in the AppsFlyer dashboard).

3.2 Adding Code

Add the following code to your **AppDelegate.m** source file at**“applicationDidBecomeActive”** function:

```
#import "AppsFlyerTracker.h"
-(void)applicationDidBecomeActive:(UIApplication *)application
{
     // Track Installs, updates & sessions(app opens) (You must include th
     [[AppsFlyerTracker sharedTracker] trackAppLaunch];
```

4. In-App Events Tracking API (Optional)

This API enables AppsFlyer to track post install events. These events are defined by the advertiser and include an event name in addition to optional event values.

Tracking in-app events helps you measure and analyse how loyal users discover your app, and attribute them to specific campaigns/media sources. It is recommended to take the time and define the events you would like to measure to allow you to track ROI (Return on Investment) and LTV (Lifetime Value).

Syntax:

```
- (void) trackEvent:(NSString *)eventName withValues:(NSDictionary*)values
```

eventName is any string to define the event name.

values is a dictionary of event parameters that comprise a rich event.

Counting revenue as part of a rich in-app event: Use the 'af_revenue" (constant)

`AFEventParamRevenue`
event parameter to count revenue as part of an in-app event. You can populate it with any numeric value, positive or negative.

NOTE: "af_price"

`AFEventParamPrice`
is not counted as revenue and is a descriptive parameter which does not affect revenue and LTV measurements.

Example 1: Level Achieved In-App Event

```
[[AppsFlyerTracker sharedTracker] trackEvent: AFEventLevelAchieved withVal
    AFEventParamLevel: @9,
    AFEventParamScore : @100 }];
```
This generates an event of type "af_level_achieved" with the following event values:

{af_level: 9 , af_score: 100}

Example 2: Purchase Event

```
[[AppsFlyerTracker sharedTracker] trackEvent:AFEventPurchase withValues: @
    AFEventParamContentId:@"1234567",
    AFEventParamContentType : @"category_a",
    AFEventParamRevenue: @200,
    AFEventParamCurrency:@"USD"}];
```
This generates an event of type "af_purchase" with the following event values:

{af_content_id: "1234567" , af_content_type: "category_a", af_revenue: 200, af_currency: "USD"}

The purchase event above contains a $200 revenue, appearing as revenue in the dashboard.

NOTE: An In-app event name must be no longer than 45 characters. Events names with more than 45 characters do not appear in the dashboard, but only in the raw Data, Pull and Push APIs.

For details of the AppsFlyer Rich In-App Events for iOS, click here.

5. Advanced Integration

The APIs below are optional and are part of the advanced integration with AppsFlyer SDK.

5.1 Set Currency Code (Optional)

You can set a global currency code using the API below, in addition to specific currency codes that can be used as part of each in-app event that is sent to AppsFlyer. The global currency code is used when “af_currency” (*AFEventParamCurrency*) is not sent as part of an in-app event.

USD is the default value. Find acceptable ISO currency codes here.

Use the following API in order to set the currency code:

```
[[AppsFlyerTracker sharedTracker].currencyCode = @"GBP"];
```

5.2 Get AppsFlyer Unique ID (Optional)

An AppsFlyer proprietary unique ID is created for every new install of an app. AppsFlyer’s unique ID is the main ID used by AppsFlyer in the Reports and APIs.

Use the following API In order to get AppsFlyer’s unique ID:

```
(NSString *) [AppsFlyerTracker sharedTracker] getAppsFlyerUID
```

5.3 Set Customer User ID (Optional)

Setting your own customer ID enables you to cross-reference your own unique ID with AppsFlyer's unique ID and the other devices' IDs. This ID is available in AppsFlyer CSV reports along with postbacks APIs for cross-referencing with your internal IDs.

To set your customer user ID:

```
[[AppsFlyerTracker sharedTracker].customerUserID = @"YOUR_CUSTOM_DEVICE_ID
```

IMPORTANT NOTES:

- ✓ It is recommended to set your Customer User ID as soon as possible as it is only associated to events reported after setting this attribute.
- ✓ You must set the Customer User ID using this API in order to use AppsFlyer's integrations with Analytics platforms such as Mixpanel and Swrve.

5.4 Enabling Extension Support

The extension need permissions to use Internet:

1. Go to your extension's info.plist file
2. In the NSExtension / NSExtensionAttributes set the RequestsOpenAccess flag to YES.

On the Extension ViewController ViewDidLoad initialize AppsFlyerLib:

```
[AppsFlyerTracker sharedTracker].appsFlyerDevKey = @"YOUR_DEV_KEY_HERE";
 [AppsFlyerTracker sharedTracker].appleAppID = @"APP_ID_HERE";  [AppsFlyer
  [[AppsFlyerTracker sharedTracker] trackAppLaunch];
```

To receive Conversion Data:

1. Go to the Extension ViewController.h
2. Import AppsFlyer header file

style="font-weight: 400;">

```
#import "AppsFlyerTracker.h"(/pre>
```

1 Add <AppsFlyerTrackerDelegate> on the interface declaration

On the Extension ViewController.m add the following methods:

```
-(void)onConversionDataReceived:(NSDictionary*) installData {
    id status = [installData objectForKey:@"af_status"];
   if([status isEqualToString:@"Non-organic"]) {
      id sourceID = [installData objectForKey:@"media_source"];
      id campaign = [installData objectForKey:@"campaign"];
      NSLog(@"This is a none organic install. Media source: %@  Campaign:
  } else if([status isEqualToString:@"Organic"]) {
      NSLog(@"This is an organic install.");
  }
 }
-(void)onConversionDataRequestFailure:(NSError *) error {
  NSLog(@"%@",error);
}
- (void) onAppOpenAttribution:(NSDictionary*) attributionData {
  NSLog(@"attribution data: %@", attributionData);
}
- (void) onAppOpenAttributionFailure:(NSError *)error {
  NSLog(@"%@",error);
 }
```

5.5 Get Conversion Data (Optional)

AppsFlyer allows you to access the user attribution data in real time directly at the SDK level. It enables you to customize the landing page a user sees on the very first app open after a fresh app install.

For more information regarding this advanced functionality, read here.

5.6 Set User Email (Optional)

AppsFlyer enables you to report one or more of the user's email addresses. The developer should collect the email addresses from the user and report it to AppsFlyer according to the developer's desired encryption method. The following encryption methods are available: Sha1, MD5 and plain.

Example:

```
[[AppsFlyerTracker sharedTracker] setUserEmails:@[@"email1@domain.com", @"
```

5.7 Reporting Deep Links for Retargeting Attribution (Optional)

AppsFlyer enables you to report launches initiated through deep links and Universal Links and to analyse the performance of your retargeting attribution campaigns.

To report such launches, add the following code from the app delegate:

```
// Reports app open from a Universal Link for iOS 9
- (BOOL) application:(UIApplication *)application continueUserActivity:(NS
{
[[AppsFlyerTracker sharedTracker] continueUserActivity:userActivity restor
return YES;
}
// Reports app open from deeplink for iOS 8 or below
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url
      sourceApplication:(NSString*)sourceApplication annotation:(id)annota
{
[[AppsFlyerTracker sharedTracker] handleOpenURL:url
      sourceApplication:sourceApplication];
return YES;
}
```

5.8 In-App Purchase Receipt Validation (Optional)

NOTE: This function is supported for iOS7 and above.

AppsFlyer's SDK can provide in-app purchase server verification. To set receipt validation tracking you need to call the **validateAndTrackInAppPurchase** method inside the SKStoreKit's **completeTransaction**: callback. This call automatically generates an "af_purchase" in-app event.

```
- (void) validateAndTrackInAppPurchase:(NSString *) productIdentifier
price:(NSString *) price
currency:(NSString *) currency
transactionId:(NSString *) tranactionId
additionalParameters:(NSDictionary *) params
success:(void (^)(NSDictionary *response)) successBlock
failure:(void (^)(NSError *error, id reponse)) failedBlock;
```

This call has two callback blocks, one for 'success' and one for 'failure' (for any reason, including validation fail). On success, a dictionary is returned with the receipt validation data provided by Apple's servers.

IMPORTANT: For testing purposes, we recommend to set the**useReceiptValidationSandbox** flag to YES, as this redirects the requests to Apple sandbox servers.

[AppsFlyerTracker sharedTracker].useReceiptValidationSandbox = YES;

Example:

```
[[AppsFlyerTracker sharedTracker] validateAndTrackInAppPurchase:product.pr
currency:@"USD"
transactionId:trans.transactionIdentifier
additionalParameters:@{@"test": @"val" , @"test1" : @"val 1"}
success:^(NSDictionary *result){
               NSLog(@"Purcahse succeeded And verified!!! response: %@", r
} failure:^(NSError *error, id response) {
               NSLog(@"response = %@", response);
}];
```

5.9 Set HTTPS (Optional)

AppsFlyer's SDK communicates with its servers over HTTPS. In case you choose to disable this **(not recommended)**, you can set the isHTTPS property to NO. The default is YES.

```
[AppsFlyerTracker sharedTracker].isHTTPS = YES;
```

5.10 End User Opt-out (Optional)

AppsFlyer provides you a method to opt-out specific users from AppsFlyer analytics. This method complies with the latest privacy requirements and complies with Facebook data and privacy policies. Default is NO, meaning tracking is enabled by default. Use this API during the SDK initialization in Section 4 to explicitly opt-out:

```
[AppsFlyerTracker sharedTracker].deviceTrackingDisabled = YES;
```

5.11 Explicit opt-out from ID for Advertisers – IDFA/IFA (Optional)

The AppsFlyer SDK collects IDFA only if AdSupport.framework library is included in your project. There is no need to explicitly opt-in or opt-out. However, if you want to explicitly opt-out IDFA, use the following API during the SDK initialization in Section 4:

```
[AppsFlyerTracker sharedTracker].disableAppleAdSupportTracking = YES;
```

The IDFA is NOT collected to the AppsFlyer servers when the **disableAppleAdSupport**tracking is set to YES.

6. Testing the SDK Integration

To learn how to test the SDK integration before and after submitting to the App Store read here