AppsFlyer SDK Integration - Android V.2.3.1.18

Print / PDF

Jamie Weider
Last update: November 19, 2018 11:30

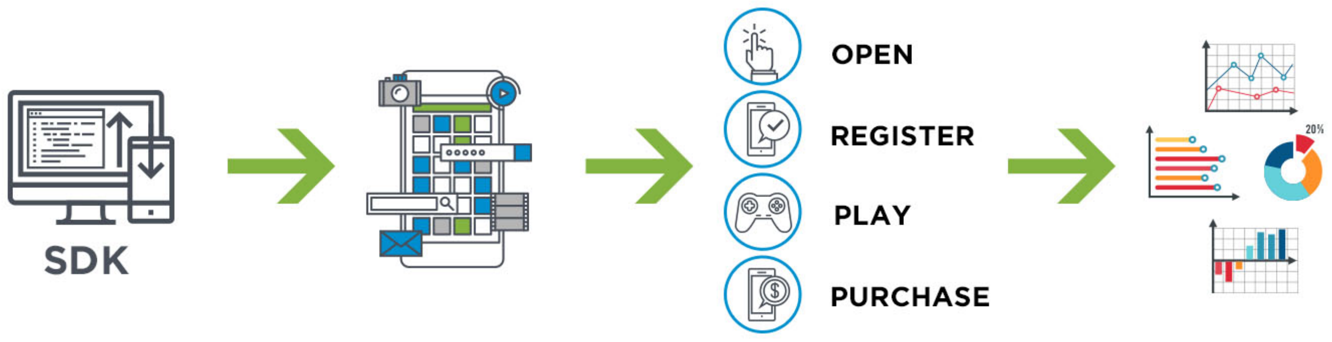
Page Contents:

This Android SDK is out of date!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

Current Version: v2.3.1.18

AppsFlyer's SDK provides app installation and event tracking functionality. We have developed an SDK that is highly robust (7+ 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 detailed in sections 2 and 3 below, followed by additional optional features in section 4.

The AppsFlyer Android SDK is compatible with Android 2.3 and above.

The following sections are covered in this guide:

Android SDK Download

To download the iOS SDK as a **static library** click here.

To download the iOS SDK as a **framework** click here.

1. What's New in This Version?

- ✓ Added **validateAndTrackInAppPurchase** for in-app purchase validation with Google Play
- ✓ Performance improvements and bug fixes

2. Embedding the SDK into Your Application (Mandatory)

The following steps are mandatory to measure installs and sessions.

2.1 Add the AppsFlyer SDK to your Project

Add **AF-Android-SDK.jar** (download here) to the project's class path.

2.2 Set the Required Permissions

The **AndroidManifest.xml** should include the following permissions:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

* READ_PHONE_STATE permission is optional.

Adding this permission enables carrier tracking of IMEI (required for tracking outside of Google Play).

2.3 Set an Install Referrer Broadcast Receiver in AndroidManifest.xml

Android apps cannot have multiple receivers with the same intent-filtered action.

AppsFlyer provides a solution that broadcasts INSTALL_REFERRER to all other receivers automatically. In the AndroidManifest.xml, add the following receiver as the FIRST receiver for INSTALL_REFERRER, and ensure the receiver tag is within the application tag:

```
<receiver android:name="com.appsflyer.MultipleInstallBroadcastReceiver" an
  <intent-filter>
     <action android:name="com.android.vending.INSTALL_REFERRER" />
  </intent-filter>
</receiver>
```

If you want to use multiple receivers, the Manifest.xml must appear, as follows:

```
<!—The AppsFlyer Install Receiver is first and will broadcast to all recei
<receiver android:name="com.appsflyer.MultipleInstallBroadcastReceiver" an
  <intent-filter>
     <action android:name="com.android.vending.INSTALL_REFERRER" />
  </intent-filter>
</receiver>
<!—All other receivers should follow right after -->
<receiver android:name="com.google.android.apps.analytics.AnalyticsReceive
 <intent-filter>
      <action android:name="com.android.vending.INSTALL_REFERRER" />
 </intent-filter>
</receiver>
<receiver android:name="com.admob.android.ads.analytics.InstallReceiver" a
      <intent-filter>
          <action android:name="com.android.vending.INSTALL_REFERRER" />
      </intent-filter>
</receiver>
```

For more information about how to add an additional receiver to access app install data from your app context, click here.

NOTE: If you are using other mechanisms to broadcast multiple receivers, use the com.appsflyer.AppsFlyerLib class below. This class does not broadcast to other receivers.

```
<receiver android:name="com.appsflyer.AppsFlyerLib" android:exported="true
          <intent-filter>
              <action android:name="com.android.vending.INSTALL_REFERRER"
          </intent-filter>
</receiver>
```

2.4 Embed Google Play Services into Your App

AppsFlyer can track Google Advertising ID to improve tracking.

To add Google Advertising ID:

1. Install the Google Play Services SDK and import it into your project.
2. Add the following entry to the **AndroidManifest.xml** as the last entry under application (before </application>):

```
<meta-data android:name="com.google.android.gms.version"
android:value="@integer/google_play_services_version" />
```

NOTE: AppsFlyer uses the Google Play Services library for retrieving the Google Advertising ID.

Although the library provides additional services, if you use ProGuard as part of your build process, it has a light footprint. We only use the ads packages from Google Services. If you want to optimize the size of your project you can exclude any other packages.

For more details see https://developers.google.com/android/guides/setup.

Source: https://developer.android.com/google/play-services/setup.html

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

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

To initialize the SDK, add the following code to your **onCreate** function:

```
AppsFlyerLib.setAppsFlyerKey("[Dev_Key]");
AppsFlyerLib.sendTracking(getApplicationContext());
```

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

This API enables AppsFlyer to detect installations, sessions, and updates.

IMPORTANT NOTES:

- ✓ The **sendTracking** call must be called from the first activity in launch sequence. For example, if you have a splash screen activity or a tutorial activity which is called before the main activity, the **sendTracking** API must be called at the beginning of this activity's **onCreate** function.
- ✓ To track app sessions (opens), you must call this API on every app session.

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 analyze 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 want to measure to allow you to track ROI (Return on Investment) and LTV (Lifetime Value).

Syntax:

```
public static void trackEvent(Context context, String eventName, Map event
```

context - use getApplicationContext()

eventName is any string to define the event name.

eventValues is a map of event parameters that comprise a rich event.

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

`AFInAppEventParameterName.REVENUE`
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**

`AFInAppEventParameterName.PRICE`
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

```
Map<String, Object> eventValue = new HashMap<String, Object>();
eventValue.put(AFInAppEventParameterName.LEVEL,9);
eventValue.put(AFInAppEventParameterName.SCORE,100);
AppsFlyerLib.trackEvent(content,AFInAppEventType.LEVEL_ACHIEVED,eventValue
```

This generates an event of type **af_level_achieved** with the following event values:

```
{af_level: 9, af_score: 100}
```

Example 2: Purchase Event

```
Map<String, Object> eventValue = new HashMap<String, Object>();
eventValue.put(AFInAppEventParameterName.REVENUE,200);
eventValue.put(AFInAppEventParameterName.CONTENT_TYPE,"category_a");
eventValue.put(AFInAppEventParameterName.CONTENT_ID,"1234567");
eventValue.put(AFInAppEventParameterName.CURRENCY,"USD");
AppsFlyerLib.trackEvent(content,AFInAppEventType.PURCHASE,eventValue);
```

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,
```

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 Android, click here.

5. Advanced Integration

The APIs below are optional and are part of the advanced integration with the 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 sent to AppsFlyer. The global currency code is used when **af_currency**

```
AFInAppEventParameterName.CURRENCY
```

is not sent as part of an in-app event.

USD is the default value. You can find acceptable ISO currency codes here.

Use the following API to set the currency code:

```
AppsFlyerLib.setCurrencyCode("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 to obtain AppsFlyer's unique ID:

```
String appsFlyerId = AppsFlyerLib.getAppsFlyerUID(this);
```

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 Postback APIs for cross-referencing with your internal IDs.

To set your Customer User ID:

```
AppsFlyerLib.setCustomerUserId("myId");
```

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 to use AppsFlyer's integrations with Analytics platforms such as Mixpanel and Swrve.

5.4 Get Conversion Data (Optional)

AppsFlyer allows you to access the user attribution data in real-time directly at 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.

5.5 Set User Email (Optional)

AppsFlyer enables you to report one or more of the device's associated email addresses. You must collect the email addresses and report it to AppsFlyer according to your required encryption method.

The following encryption methods are available: Sha1, MD5 and plain.

Example:

```
AppsFlyerLib.setUserEmails(AppsFlyerProperties.EmailsCryptType.MD5, "email
```

5.6 Reporting Deep Links for Retargeting Attribution (Optional)

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

If the deep link is opening an activity other than the main activity, set the dev key on that activity too and implement the following code to make sure the **SendTracking** API is not called multiple times in the same session (for the main and other activities):

NOTE: You must use the activity context and not the application context. This code must be placed in the Activity **onCreate**.

```
AppsFlyerLib.setAppsFlyerKey("[Dev_Key]");
Intent intent = ((Activity)context).getIntent();
String action = intent.getAction();
if (action == Intent.ACTION_VIEW) {
    AppsFlyerLib.sendTracking(context);
}
```

5.7 In-App Purchase Validation (Optional)

AppsFlyer’s SDK provides server verification for in-app purchases. To set purchase validation tracking, call the **validateAndTrackInAppPurchase** method inside the **onActivityResult** function.

This call automatically generates an **af_purchase** in-app event.

```
public static void validateAndTrackInAppPurchase(Context context,
String publicKey, String signature, String purchaseData, String price,
String currency, HashMap<String, String> additionalParameters);
```

This call has two callback blocks, one for 'Success' and one for 'Failure' (for any reason, including validation fail).

```
AppsFlyerLib.registerValidatorListener(this,new AppsFlyerInAppPurchaseVali
          public void onValidateInApp() {
              Log.d(TAG, "Purchase validated successfully");
          }
          public void onValidateInAppFailure(String error) {
              Log.d(TAG, "onValidateInAppFailure called: " + error);
          }
      });
```

5.8 Enable HTTPS to HTTP Fallback (Optional)

AppsFlyer Android SDK uses HTTPS at all times, without HTTP fallback. If you want to allow the SDK to use HTTP fallback, add the following setting:

```
AppsFlyerLib.setUseHTTPFalback(true);
```

5.9 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:

```
AppsFlyerLib.setDeviceTrackingDisabled(true);
```

5.10 Measuring Session Duration (Optional)

AppsFlyer can track the duration of a user session. To enable session duration measurement you must override **onResume** and **onPause** for each Activity of your app:

```
@Override
public void onResume() {
super.onResume();
AppsFlyerLib.onActivityResume(this);
}
@Override
public void onPause() {
super.onPause();
AppsFlyerLib.onActivityPause(this);
}
```

5.11 Measure App Uninstalls (Optional)

To allow AppsFlyer to measure uninstalls of your app, add the following entry to the app **manifest.xml**:

```
<receiver android:name="com.appsflyer.AppsFlyerLib">
<intent-filter>
<action android:name="android.intent.action.PACKAGE_REMOVED"/>
         <data android:scheme="package"/>
</intent-filter>
</receiver>
```

***NOTE*:** This allows other packages to receive this event and report to AppsFlyer's servers that a certain package name was uninstalled from the device.

5.12 Track App Installs Out of Google Play (Optional)

***IMPORTANT NOTE*:** Google Play is the default store. If you are publishing your app only on Google Play, skip this section.

To track installs out of Google Play, set the channel/store at the app **AndroidManifest.xml** with a unique channel or any store name for each APK. The **CHANNEL** value is case sensitive.

Examples:

Amazon:

```
<meta-data android:name="CHANNEL" android:value="Amazon" />
```

Standalone:

```
<meta-data android:name="CHANNEL" android:value="Standalone"/>
```

Verizon (Pre-Installed):

```
<meta-data android:name="CHANNEL" android:value="Verizon" />
```

NOTE: You must configure the CHANNEL value at the AppsFlyer dashboard when setting up the app.

Place the meta-data tag before the </application> tag.

For more details on how to track installs for out-of-store apps, read here.

6. Testing the SDK Integration

To learn how to test the SDK integration before submitting to the Google Play Store, click here and after submitting to the Google Play Store, click here.