1. Overview

AppsFlyer Android SDK provides app install and event recording functionality to Android apps. The SDK can be used with Java/Kotlin.

Embed the SDK in your app to record:

- App installs
- User engagement (for example, sessions and in-app events)

1.1 SDK integration—what you need to do

<table>
<thead>
<tr>
<th>Tab</th>
<th>Purpose</th>
<th>After completion</th>
</tr>
</thead>
</table>

SDK Version: 5.4.6 (Release Notes)

Deprecated SDK versions

https://support.appsflyer.com/hc/en-us/articles/360014302437#introduction
<table>
<thead>
<tr>
<th>Tab</th>
<th>Purpose</th>
<th>After completion</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SDK integration</strong></td>
<td>Add and configure the SDK</td>
<td>• A new organic install in your app dashboard.</td>
</tr>
<tr>
<td>(Mandatory)</td>
<td></td>
<td>• A new non-organic install in your app's dashboard.</td>
</tr>
<tr>
<td><strong>Core APIs</strong></td>
<td>Measure in-app events and revenue, Enable deep linking and gather conversion data</td>
<td>• In-app events and revenue appear on your dashboard.</td>
</tr>
<tr>
<td>(Recommended)</td>
<td></td>
<td>• You are able to perform deep linking.</td>
</tr>
<tr>
<td><strong>Additional APIs</strong></td>
<td>Implement and use optional APIs. For example, Uninstall measurement and referrals attribution.</td>
<td>• You are able to measure uninstalls, referrals, and user engagement with push notifications.</td>
</tr>
<tr>
<td>(Recommended)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>API reference</strong></td>
<td>SDK APIs reference document</td>
<td></td>
</tr>
</tbody>
</table>

### 1.2 SDK compatibility with Android platforms

- Starting Android V4.0
- Non-mobile Android-based platforms such as Smart TVs, including Amazon Fire TV
- Out-of store markets for Android apps, like Amazon, and Baidu

The following sections describe how to implement and initialize the AppsFlyer SDK. After you complete this section, you will see two installs in your AppsFlyer dashboard: organic and non-organic.

### 2. Add the SDK to your app

#### 2.1 Add the SDK to your project
Use one of the following methods to add the SDK to your app:

- [Best practice] Using Gradle
- Manually add the SDK

Add the SDK using Gradle

1. Add the code below to Module-level `app/build.gradle` before `dependencies`:
   ```gradle
   repositories {
     mavenCentral()
   }
   ```

   Add the latest version of AppsFlyer SDK as a dependency. You can find the latest version [here].
   ```gradle
   dependencies {
     //make sure to use the latest SDK version: https://mvnrepository.com/artifact/com.appsflyer/af-android-sdk
     implementation 'com.appsflyer:af-android-sdk:5.4.0'
   }
   ```

2. Sync the project to retrieve the dependencies - see the following screenshot:
Manually add the SDK

1. Download the AF-Android-SDK.jar
2. Add it to the project

2.2 Adding Android install referrer to your app

The Android Install Referrer improves attribution accuracy, protects from install fraud and more. It is supported from AppsFlyer Android SDK version 4.8.6.
## Note

Google deprecated the *BroadcastReceiver* in March 2020.

- This change does not affect the app.
- You may still need the BroadcastReceiver for out-of-store attribution. Check with the store where the app is listed to make sure.
- Implementing the install referrer is now mandatory.

There are two ways to add the install referrer to your app:

- Use Gradle (recommended)
- Manually add the install referrer

### Gradle

Add the Android Install Referrer as a dependency. You can find the latest version [here](https://support.appsflyer.com/hc/en-us/articles/360014302437#introduction).

```java
dependencies {
    //make sure to use the latest SDK version: https://mvnrepository.com/artifact/com.appsflyer/af-android-sdk
    implementation 'com.appsflyer:af-android-sdk:5.+'
    implementation 'com.android.installreferrer:installreferrer:1.1'
}
```

1. Sync the project to retrieve the dependencies - see the following screenshot:
If you are using ProGuard and want to use Google's new referrer API, set the following ProGuard rule:

```
-keep public class com.android.installreferrer.** { *; }
```

### Manual

1. Download the Install Referrer aar
2. Add it to the project
3. Add the following permission to the manifest:

```
com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE
```
Adding permissions helps increase the rate of Probabilistic modeling attribution. It also allows the SDK to send more data like Wifi and carrier network data. You can find this data in raw data reports and use it for analysis and optimization of campaigns.

Add Required Permissions

1. Add the following permissions to AndroidManifest.xml:

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

2.4 Set BroadcastReceiver to get data from Google Play

Note

Google deprecated the BroadcastReceiver in March 2020.

- This change does not affect the app.
- You may still need the BroadcastReceiver for out-of-store attribution. Check with the store where the app is listed to make sure.
- Implementing the install referrer is now mandatory.

The BroadcastReceiver gets information from Google Play that AppsFlyer uses for attribution. Using the BroadcastReceiver increases the attribution rate.

The following two options are available for implementing the install referrer BroadcastReceiver:

Single BroadcastReceiver
If you do not have a receiver listening on the INSTALL_REFERRER, in AndroidManifest.xml, in the application tag add the following receiver:

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

**Multiple BroadcastReceiver**

If you already have a receiver listening on the INSTALL_REFERRER, 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:

```xml
<application>
  <receiver android:name="com.appsflyer.MultipleInstallBroadcastReceiver" android:exported="true">
    <intent-filter>
      <action android:name="com.android.vending.INSTALL_REFERRER" />
    </intent-filter>
  </receiver>
</application>
```
If you get the error "Unresolved class SingleInstallBroadcastReceiver" after adding the receiver to AndroidManifest.xml, make sure to build the app first.

3. Implement and initialize the SDK

This section describes how to implement and initialize the SDK.

3.1 Retrieve your dev key

AppsFlyer uses the dev key to uniquely identify your account. The dev key is mandatory because it allows the SDK to securely send and retrieve data that belongs to your AppsFlyer account.

**Warning! Using the wrong dev key or an incorrect dev key impacts all traffic sent from the SDK and breaks attribution and reporting.**

To retrieve your dev key:

1. Go to your app's dashboard.
2. In the dashboard, under **Configuration** click **App Settings**.
3. Copy your dev key.
3.2 Initialize the SDK

We recommend initializing the SDK inside the app's global application class. This allows the SDK to initialize in all scenarios, including deep linking.

The steps listed below take place inside the app's global application class.

1. Inside the app's global class, import the following libraries

```java
import android.app.Application;
import android.util.Log;
import com.appsflyer.AppsFlyerLib;
import com.appsflyer.AppsFlyerConversionListener;
import java.util.Map;
```

2. Inside the global class, assign your dev key to a variable, preferably named AF_DEV_KEY.

   Important: it is crucial to use the correct dev key when initializing the SDK. Using the wrong dev key or an incorrect dev
key impact all traffic sent from the SDK and cause attribution and reporting issues.

Java

```java
public class AFApplication extends Application {
    private static final String AF_DEV_KEY = "qrdZGj123456789";

    // ...
}
```

Kotlin

```kotlin
class AFApplication : Application() {
    private val devKey = "qrdZGj123456789";

    // ...
}
```

3. Inside the global class, after the call to `super.onCreate()`, implement the `AppsFlyerConversionListener`. See code below in step 4.

4. Inside the global class, after the `ConversionListener`, initialize the SDK using `init()` method.

Java

```java
@override
public void onCreate() {
    super.onCreate();
    AppsFlyerConversionListener conversionListener = new AppsFlyerConversionListener() {
```
@Override
public void onConversionDataSuccess(Map<String, Object> conversionData) {
    for (String attrName : conversionData.keySet()) {
        Log.d("LOG_TAG", "attribute: " + attrName + " = " + conversionData.get(attrName));
    }
}

@Override
public void onConversionDataFail(String errorMessage) {
    Log.d("LOG_TAG", "error getting conversion data: " + errorMessage);
}

@Override
public void onAppOpenAttribution(Map<String, String> attributionData) {
    for (String attrName : attributionData.keySet()) {
        Log.d("LOG_TAG", "attribute: " + attrName + " = " + attributionData.get(attrName));
    }
}

@Override
public void onAttributionFailure(String errorMessage) {
    Log.d("LOG_TAG", "error onAttributionFailure : " + errorMessage);
}

AppsFlyerLib.getInstance().init(AF_DEV_KEY, conversionListener, this);
}
```kotlin
class AFApplication : Application() {

    private val devKey = "qdZGj123456789";

    override fun onCreate() {
        super.onCreate()
        val conversionDataListener = object : AppsFlyerConversionListener{
            override fun onConversionDataSuccess(data: MutableMap<String, Any>?) {
                data?.let {
                    it.map {
                        Log.i(LOG_TAG, "conversion_attribute: \${it.key} = \${it.value}"")
                    }
                }
            }

            override fun onConversionDataFail(error: String?) {
                Log.e(LOG_TAG, "error onAttributionFailure : \$error")
            }

            override fun onAppOpenAttribution(data: MutableMap<String, String>?) {
                data?.map {
                    Log.d(LOG_TAG, "onAppOpen_attribute: \${it.key} = \${it.value}")
                }
            }

            override fun onAttributionFailure(error: String?) {
                Log.e(LOG_TAG, "error onAttributionFailure : \$error")
            }
        }

        AppsFlyerLib.getInstance().init(devKey, conversionDataListener, this)
    }
}
```

5. Inside the app's global class, call `startTracking()`.
In the `AndroidManifest.xml` file, inside the `application` tag, add the following line:

```xml
android:name="APP PACKAGE NAME.AFApplication"
```

- **APP PACKAGE NAME** - replace this with the app package name.
- **AFApplication** - replace this with the name that you set for the global app class.
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.appsflyer.sample">

<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<!-- Optional : -->
<uses-permission android:name="android.permission.READ_PHONE_STATE" />

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

<uses-permission android:name="android.permission.WAKE_LOCK" />

<application
    android:name="com.appsflyer.sample.AFApplication"
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name="com.appsflyer.sample.MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
        </intent-filter>
    </activity>
</application>
</manifest>
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>

</activity>

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

</application>

</manifest>

This line in manifest.xml tells the application what is the global applications. As mentioned above, doing so makes the AppsFlyer SDK globally accessible throughout the app.

### 3.4 Delay SDK initialization

To delay initializing the SDK, you can call startTracking from the Activity class.
This option can be used, if, for example, you need to postpone startTracking until you receive user consent due to GDPR or CCPA requirements, etc.

When invoking startTracking in Activity class:

- Make sure to pass Activity instance as an argument and not Application.
- If you plan to use deep linking features, add the `startTracking()` API to any other Activity you can deep link to that doesn't have startTracking in it.

4. Test installs

You are ready to test the SDK integration by simulating organic and non-organic installs.

4.1 Register your test device

Before you start testing installs, register the test device.

4.2 Simulate an organic install

Organic installs are unattributed installs which are usually direct installs from app stores.

To simulate an organic install:

1. Make sure you have a mobile device connected to your computer.
2. In Android Studio, open the Logcat.
3. From Android Studio, install the app on the device or emulator.
4. Wait for the app to launch.
5. In the Logcat, look for your app's package name.

You should see the following:
The highlighted part in the screenshot indicates that the SDK reports an organic install. This data comes from the `onConversionDataSuccess` method in the `AFApplication` class. Getting conversion data is discussed later in this guide.

**Note:** Starting SDK V5, `onConversionDataSuccess` is the name of the method for getting conversion data. If you are using an SDK version lower than 5.0.0, the name of the method is `onInstallConversionDataLoaded`. We recommend that you upgrade to SDK 5.0.0. To learn more, click here.

An organic install should now appear on the **Overview** page of the application's dashboard.

If you don't see an install in the app's dashboard, see our [SDK troubleshooting guide](https://support.appsflyer.com/hc/en-us/articles/360014302437#introduction).

### 4.3 Simulate a non-organic install

A non-organic install is an attributed install that usually follows an ad engagement. You can simulate a non-organic install by using attribution links.

To simulate a non-organic install:

1. In the manifest, find out what is the package name of your app e.g. `com.company.app`.
   - In the URL below, replace `<app_id>` with your app's package name:

     ```
     https://app.appsflyer.com/<app_id>?pid=Test&c=Test
     ```

     - The `c` parameter specifies the name of campaign.
     - The `pid` parameter specifies the name of the media source to which the install is attributed.
You should see the following:

A non-organic install should now appear in the **Overview** page of the application's dashboard.

![Logcat](https://support.appsflyer.com/hc/en-us/articles/360014302437#introduction)

**Note**

When you're done testing and debugging the SDK integration, switch off the **SDK logs**.
Known issues with integrating the SDK

See the following to learn more about possible issues when integrating the SDK and how to overcome them.

ProGuard warning

If you are using ProGuard and you encounter a warning regarding our `AFKeystoreWrapper` class, then add the following code to your ProGuard rules file:

```
-keep class com.appsflyer.* { *; }
```

Backup rules

If you add `android:fullBackupContent="true"` inside the `<application>` tag in the AndroidManifest.xml, you might get the error:

```
Manifest merger failed : Attribute application#fullBackupContent value=(true)
```

To fix this error, add `tools:replace="android:fullBackupContent"` in the `<application>` tag in the AndroidManifest.xml file.

If you have your own backup rules specified (`android:fullBackupContent="@xml/my_rules"`), in addition to the instructions above, please merge them with AppsFlyer rules manually by adding the following rule:

```
<full-backup-content>
  ...//your custom rules
  <exclude domain="sharedpref" path="appsflyer-data"/>
</full-backup-content>
```

Missing resource files
If you are using Android SDK V5 and above, make sure that in the APK file, in addition to the `classes.dex` and `resources` files, you also have folder `com > appsflyer > internal` with files `a`- and `b`- inside.

**Note:** Before SDK 5.3.0, file names are `a.` and `b.`

Check that you have the required files by opening your APK in Android Studio. See the following screenshot for reference.

If those files are missing, the SDK can't make network requests to our server, and you need to contact your CSM or support.
This tab explains how to record in-app events and revenue, and how to set up deep linking.

Recording in-app events and revenue allows you to measure the quality of your users. Deep linking allows you to provide users with better user experience.
This tab contains instructions for developers, but input from the marketer is essential because:

- The marketer should decide which in-app events need recording to measure user quality.
- The marketer has access to AppsFlyer dashboard, which is required for setting up OneLink for deep linking.

5. Recording in-app events

In-App events provide insight into what is happening in your app. We recommend to take the time and define the events you want to record. Recording in-app events helps you to measure KPIs such as ROI (Return on Investment) and LTV (Lifetime Value).

There are several ways to record in-app events. The most common way is sending events via the SDK, which we discuss in this article. To learn about other ways to record in-app events, see our in-app events overview guide.

If your app belongs to a certain vertical, e.g. travel, gaming, eCommerce, etc., you can use the full list of recommended in-app events per vertical.

5.1 In-app event names and parameters

The SDK has two in-app event-related interfaces:

- `AFInAppEventType` - constants for in-app event names.
- `AFInAppEventParameterName` - constants for in-app event parameter names.

We strongly recommend using these two interfaces for the following reasons:

- The standard naming allows AppsFlyer to automatically map events to SRNs such as Facebook, Google, Twitter, and Snapchat.
- Backward compatibility - if AppsFlyer decides to change the name of any event or event parameter, your implementation is backward compatible.

To use these two interfaces, import them:
Here is the list of recommended event names and structures.

5.2 Recording revenue

You can send revenue with any in-app event. Use the \texttt{af\_revenue (AFInAppEventParameterName.REVENUE)} event parameter to include revenue in the in-app event. You can populate it with any numeric value, positive or negative.

\texttt{af\_revenue} is the only event parameter that AppsFlyer counts as real revenue on the raw data and dashboard. For more details please click here.

When sending events with revenue, keep the following in mind:

- If you set currency code (see example below), it should be a 3 character ISO 4217 code. (default is USD).
- You can set the currency code for all events by calling the following method: \texttt{AppsFlyer.setCurrencyCode("ZZZ")} To learn about currency settings, display, and currency conversion, see our guide on revenue currency.
- The revenue value should not contain comma separators, currency sign, or text. A revenue event should be similar to 1234.56, for example.

\textbf{Example: In-app event purchase event with revenue}

\textbf{Java}

\begin{verbatim}
Map<String, Object> eventValue = new HashMap<String, Object>();
eventValue.put(AFInAppEventParameterName.REVENUE, 1234.56);
eventValue.put(AFInAppEventParameterName.CONTENT_TYPE, "Shirt");
eventValue.put(AFInAppEventParameterName.CONTENT_ID, "1234567");
eventValue.put(AFInAppEventParameterName.CURRENCY, "USD");
AppsFlyerLib.getInstance().trackEvent(getApplicationContext(), AFInAppEventType.PURCHASE, eventValue);
\end{verbatim}
Kotlin

```kotlin
val eventValue = HashMap<String, Any>()

eventValue.put(AFInAppEventParameterName.REVENUE, 1234.56)
eventValue.put(AFInAppEventParameterName.CONTENT_TYPE, "Shirt")
eventValue.put(AFInAppEventParameterName.CONTENT_ID, "1234567")

AppsFlyerLib.getInstance().trackEvent(getApplicationContext(), AFInAppEventType.PURCHASE, eventValue)
```

The purchase event above has $1234.56 in revenue associated with it, appearing as revenue in the dashboard.

Recording Negative Revenue

There may be situations where you want to record negative revenue.

For example, a user receives a refund for shoes that they purchased from you.

Java

```java
Map<String, Object> eventValue = new HashMap<String, Object>();

eventValue.put(AFInAppEventParameterName.REVENUE, -200);
eventValue.put(AFInAppEventParameterName.CONTENT_TYPE, "shoes");
eventValue.put(AFInAppEventParameterName.CONTENT_ID, "4875");

AppsFlyerLib.getInstance().trackEvent(getApplicationContext(), "cancel_purchase", eventValue);
```

Kotlin

```kotlin
val eventValue = HashMap<String, Any>()

eventValue.put(AFInAppEventParameterName.REVENUE, -200)
eventValue.put(AFInAppEventParameterName.CONTENT_TYPE, "shoes")
eventValue.put(AFInAppEventParameterName.CONTENT_ID, "4875")

AppsFlyerLib.getInstance().trackEvent(getApplicationContext(), "cancel_purchase", eventValue)
```
val eventValue = HashMap<String, Any>()
eventValue.put(AFInAppEventParameterName.REVENUE, -200)
eventValue.put(AFInAppEventParameterName.CONTENT_TYPE, "category_a")
eventValue.put(AFInAppEventParameterName.CONTENT_ID, "1234567")
eventValue.put(AFInAppEventParameterName.CURRENCY, "USD")
AppsFlyerLib.getInstance().trackEvent(applicationContext, "cancel_purchase", eventValue)

### Note

Notice the following in the code above:

1. The revenue value is preceded by a minus sign.
2. The event name has a unique value of "cancel_purchase" - to allow you to identify negative revenue events in the dashboard and raw data reports.

### 5.3 In-app purchase validation

AppsFlyer's SDK provides server receipt validation for in-app purchases. To validate a purchase, call the `validateAndTrackInAppPurchase`.

This call automatically generates an `af_purchase` in-app event, given that the purchase is validated.

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

#### Method parameters

- **String publicKey** – public key from Google Developer Console.
- **String signature** – transaction signature (returned from Google API when the purchase is completed).
Purchase validation success and failure callbacks

If you want to know if the attempt to validate the purchase is successful or not, implement registerValidatorListener in your application class. This listener has two callback blocks, one for ‘Success’ and one for ‘Failure’ (for any reason, including validation fail).

Java

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

Kotlin

```kotlin
AppsFlyerLib.getInstance().registerValidatorListener(this, object : AppsFlyerInAppPurchaseValidatorListener {
    override fun onValidateInApp() {
    }
    override fun onValidateInAppFailure(error: String) {
        Log.d(TAG, "onValidateInAppFailure called: " + error);
    }
});
```
Log.d(LOG_TAG, "Purchase validated successfully")

override fun onValidateInAppFailure(error: String) {
    Log.d(LOG_TAG, "onValidateInAppFailure called: $error")
}

Usage example of validating a purchase:

Java

// Purchase object is returned by Google API in onPurchasesUpdated() callback
private void handlePurchase(Purchase purchase) {
    Log.d(LOG_TAG, "Purchase successful!");
    Map<String, String> additional_event_values = new HashMap<>();
    additional_event_values.put("some_parameter", "some_value");
    String price = "10";
    String currency = "USD";
    AppsFlyerLib.getInstance().validateAndTrackInAppPurchase(getApplicationContext(), PUBLIC_KEY, purchase.getSignature(), purchase.getOriginalJson(), revenue, currency, {

Kotlin

// Purchase object is returned by Google API in onPurchasesUpdated() callback
private fun handlePurchase(purchase: Purchase) {

Log.d(LOG_TAG, "Purchase successful!");
val additional_event_values = HashMap<String, String>()
additional_event_values.put("some_parameter", "some_value");
val price = "10"
val currency = "USD"
AppsFlyerLib.getInstance().validateAndTrackInAppPurchase(this, PUBLIC_KEY, purchase.getSignature(), purchase.getOriginalJson(), revenue, currency, additional_event_values);

Validating in-app purchase automatically sends an in-app purchase event to AppsFlyer. See below a sample data that is passed in the event_value parameter:

```
{
  "some_parameter": "some_value", // from additional_event_values
  "af_currency": "USD", // from currency
  "af_content_id": "test_id", // from purchase
  "af_revenue": "10", // from revenue
  "af_quantity": "1", // from purchase
  "af_validated": true // flag that AF verified the purchase
}
```

Note
Calling `validateAndTrackInAppPurchase` automatically generates an `af_purchase` in-app event. Sending this event yourself creates double duplicate event reporting.

5.4 In-app events limitations

- Event name: up to 45 characters
- Event value: must not exceed 1000 characters - if longer we may truncate it
- Pricing and revenue: use only digits and decimals, e.g. 5 or 5.2
Pricing and revenue values can have up to 5 digits after the period, e.g. 5.12345
Non-English characters are supported, in in-app events, other SDK APIs, starting from Android SDK V4.8.1.

5.5 Examples for recording in-app events

You can record in-app events by calling `trackEvent` with event name and value parameters. See In-App Events documentation for more details.

Below is a simple example of how to record a purchase event. For a comprehensive list of ready-made code snippets per vertical, see our guide for rich in-app events per verticals.

**Example: In-app purchase event**

**Java**

```java
Map<String, Object> eventValues = new HashMap<>();
eventValues.put(AFInAppEventParameterName.REVENUE, 1200);
eventValues.put(AFInAppEventParameterName.CURRENCY, "JPY");
eventValues.put(AFInAppEventParameterName.CONTENT_TYPE, "Shoes");
AppsFlyerLib.getInstance().trackEvent(this, AFInAppEventType.PURCHASE, eventValues);
```

**Kotlin**

```kotlin
val eventValues = HashMap<String, Any>()
eventValues.put(AFInAppEventParameterName.REVENUE, 1200)
eventValues.put(AFInAppEventParameterName.CURRENCY, "JPY")
eventValues.put(AFInAppEventParameterName.CONTENT_TYPE, "Shoes")
AppsFlyerLib.getInstance().trackEvent(this, AFInAppEventType.PURCHASE, eventValues)
```
5.6 Recording offline in-app events

If a user initiates an event when the internet connection is unavailable, Appsflyer is still able to record it. This is how it works:

1. SDK sends the events to AppsFlyer's servers and waits for a response.
2. If the SDK doesn't receive a 200 response, the event is stored in the cache.
3. Once the next 200 response is received, the stored event is re-sent to the server.
4. If there are multiple events in the cache, they are sent to the server one promptly after another.

Note

SDK's cache can store up to 40 events, which means that only the first 40 events that happen offline are saved. Everything that comes afterwards until the next 200 response, gets discarded.

The event time that appears in the raw data is the time the event is sent to AppsFlyer after the device goes online again. It is not the actual time that the event takes place.

5.7 Handle success and failure when recording in-app events

You can set a listener when recording in-app events. The listener allows you to define logic for two scenarios:

- In-app event recorded successfully.
- An error occurred when recording in-app event.

Java
In the event that an error occurs when recording the in-app event, an error code and string description are provided, as indicated in the table that follows.

<table>
<thead>
<tr>
<th>Error code</th>
<th>String description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>&quot;Event timeout. Check 'minTimeBetweenSessions' param&quot;</td>
</tr>
<tr>
<td>Error code</td>
<td>String description</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>11</td>
<td>&quot;Skipping event because 'isStopTracking' enabled&quot;</td>
</tr>
<tr>
<td>40</td>
<td>Network error: Error description comes from Android</td>
</tr>
<tr>
<td>41</td>
<td>&quot;No dev key&quot;</td>
</tr>
<tr>
<td>50</td>
<td>&quot;Status code failure&quot; + actual response code from the server</td>
</tr>
</tbody>
</table>

6. Deep linking with OneLink

OneLink is AppsFlyer's solution for multi-platform attribution, redirection, and deep linking.

6.1 Device detection and redirection

OneLink detects the device type upon click and redirects the user to the matching destination, e.g. Google Play, iOS app store, out-of-store markets, or web pages.

The OneLink redirections guide discusses implementing multi-platform attribution links (no SDK coding required). It's also the basis for deep linking.

6.2 Deep linking

Deep linking allows you to send users to specific activities and serve them with customized content. This is especially useful when running retargeting campaigns.
To set up deep linking with OneLink, a marketer with access to AppsFlyer dashboard and a developer with access to the app must work together.

See our guide on setting up deep linking with OneLink.

### 6.3 Deferred deep linking

Deferred deep linking allows you to deep link new users and serve them with customized content after they install the app. This is unlike regular deep linking where the app needs to be already installed on the user's device.

To set up deferred deep linking with OneLink, the developer also needs access to the AppsFlyer dashboard.

The setup for deferred deep linking is the same as deep linking. The only difference is that you need to implement additional logic in the app in order to deep link the users and serve them with customized content after they install and launch the app.

See our guide on deferred deep linking to learn more.

### 6.4 Getting deep link data

The SDK provides you with the conversion or engagement data following every install or deep linking event. You can use this data to customize content and the app's behavior programmatically.

To get deep linking data when the direct deep link is used and the app is opened, implement the `onAppOpenAttribution` method.

To get deep linking re-engagement data manually at any time, implement the `performAppAttribution` method. This allows access to re-engagement data without recording a new re-engagement.

See our guide on deep linking data to learn more.

### 7. Get conversion data
You can access user attribution data in real-time for each new install, from within the SDK.

By doing this, you can serve users with personalized content, or send them to specific activities within the app (see deferred deep linking in this article). This enhances user engagement with the app.

To obtain AppsFlyer conversion data from the Android SDK, implement `AppsFlyerConversionListener`.

### Java

```java
import android.app.Application;
import com.appsflyer.AppsFlyerLib;
import com.appsflyer.AppsFlyerConversionListener;
import java.util.Map;
import android.util.Log;

public class AFApplication extends Application {
    private static final String AF_DEV_KEY = "qrdZGj123456789";

    @Override
    public void onCreate() {
        super.onCreate();
        AppsFlyerConversionListener conversionListener = new AppsFlyerConversionListener() {

            @Override
            public void onConversionDataSuccess(Map<String, Object> conversionData) {

                for (String attrName : conversionData.keySet()) {
                    Log.d("LOG_TAG", "attribute: " + attrName + " = " + conversionData.get(attrName));
                }
            }

            @Override
            public void onConversionDataFail(String errorMessage) {
                Log.d("LOG_TAG", "error getting conversion data: " + errorMessage);
            }
        }
    }
}
```

https://support.appsflyer.com/hc/en-us/articles/360014302437#introduction
```java
public void onAppOpenAttribution(Map<String, String> conversionData) {
    for (String attrName : conversionData.keySet()) {
        Log.d("LOG_TAG", "attribute: " + attrName + " = " + conversionData.get(attrName));
    }
}

@Override
public void onAttributionFailure(String errorMessage) {
    Log.d("LOG_TAG", "error onAttributionFailure : " + errorMessage);
}
};

AppsFlyerLib.getInstance().init(AF_DEV_KEY, conversionListener, getApplicationContext());
AppsFlyerLib.getInstance().startTracking(this);
```

Kotlin
import com.appsflyer.AppsFlyerConversionListener
import com.appsflyer.AppsFlyerLib
import com.appsflyer.AppsFlyerLibCore.LOG_TAG

class AFApplication : Application() {
    private val devKey = "qrdZGj123456789";

    override fun onCreate() {
        super.onCreate()

        val conversionDataListener = object : AppsFlyerConversionListener{
            override fun onConversionDataSuccess(data: MutableMap<String, Any>?) {
                data?.let {
                    cvData ->
                    cvData.map {
                        Log.i(LOG_TAG, "conversion_attribute: ${it.key} = ${it.value}"")
                    }
                }
            }

            override fun onConversionDataFail(error: String?) {
                Log.e(LOG_TAG, "error onAttributionFailure : $error")
            }

            override fun onAppOpenAttribution(data: MutableMap<String, String>?) {
                data?.map {
                    Log.d(LOG_TAG, "onAppOpen_attribute: ${it.key} = ${it.value}"")
                }
            }

            override fun onAttributionFailure(error: String?) {
                Log.e(LOG_TAG, "error onAttributionFailure : $error")
            }
        }

        AppsFlyerLib.getInstance().init(devKey, conversionDataListener, applicationContext)
        AppsFlyerLib.getInstance().startTracking(this)
    }
}
The most important APIs in the `AppsFlyerConversionListener` interface are:

- `onInstallConversionData` - provides conversion data for new installs.
  
  **Note**: Starting SDK V5, `onConversionDataSuccess` is the name of the method for getting conversion data. If you are using an SDK version lower than 5.0.0, the name of the method is `onInstallConversionDataLoaded`. You should upgrade to SDK 5.0.0. To learn more, click [here](https://support.appsflyer.com/hc/en-us/articles/360014302437#introduction).

- `onAppOpenAttribution` - provides retargeting conversion data when an existing app is launched, either manually or through deep linking.

To learn more about conversion data, see our guide on [conversion data scenarios](https://support.appsflyer.com/hc/en-us/articles/360014302437#introduction).

### 8. Attribution

#### Out-of-store Android apps

With AppsFlyer, you can attribute installs for out-of-store Android apps. This allows you to promote your apps and reach larger audiences in markets where Google Play is not available.

For more details on how to attribute installs for out-of-store apps, read [here](https://support.appsflyer.com/hc/en-us/articles/360014302437#introduction).

#### Pre-installed apps

In pre-install campaigns, app owners can ask device manufacturers to pre-install their apps on devices before they leave the factory.

With AppsFlyer, you can easily attribute installs of pre-installed apps. When users launch your app for the first time, AppsFlyer attributes the install to the manufacturer as a media source.

For details, click [here](https://support.appsflyer.com/hc/en-us/articles/360014302437#introduction).

#### Measure uninstalls
AppsFlyer allows you to measure the uninstalls rate of users coming from different sources. Uninstall measurement can help you to analyze and optimize your campaigns according to this significant KPI.
To learn how to setup uninstall measurement, read here.

Setting a tracking request listener

If you want to receive a confirmation that the tracking request was successfully received by the AppsFlyer servers, implement the AppsFlyerTrackingRequestListener listener.

The `onTrackingRequestSuccess()` callback method is invoked for every 200 response to an attribution request made by the SDK.

The `onTrackingRequestFailure(String error)` callback method is invoked for any other response and returns the response as the error string.

Implementation example

**Java**

```java
AppsFlyerLib.GetInstance().startTracking(getApplicationContext(), "devKey", new AppsFlyerRequestListener() {
    @Override
    public void onSuccess() {
        Log.d(LOG_TAG, "Launch sent successfully, got 200 response code from server");
    }
    @Override
    public void onError(int i, @NonNull String s) {
        Log.d(LOG_TAG, "Launch failed to be sent:
" + "Error code: " + i + "\n" + "Error description: " + s);
    }
});
```

**Kotlin**

```kotlin
AppsFlyerLib.getInstance().startTracking(getApplicationContext(), "devKey", object : AppsFlyerRequestListener {
    override fun onSuccess() {
        Log.d(LOG_TAG, "Launch sent successfully, got 200 response code from server");
    }
    override fun onError(i: Int, s: String) {
        Log.d(LOG_TAG, "Launch failed to be sent:
" + "Error code: " + i + "\n" + "Error description: " + s);
    }
});
```
AppsFlyerLib.getInstance().startTracking(this, "devKey", object : AppsFlyerRequestListener {
    override fun onSuccess() {
        Log.d(LOG_TAG, "Launch sent successfully")
    }
    override fun onError(errorCode: Int, errorDesc: String) {
        Log.d(LOG_TAG, "Launch failed to be sent:
            " + errorCode + "\n            Error code: " + errorCode + "\n            Error description: " + errorDesc)
    }
})

In the event that an error occurs during the request listener, an error code and string description are provided, as indicated in the table that follows.

<table>
<thead>
<tr>
<th>Error code</th>
<th>String description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>&quot;Event timeout. Check 'minTimeBetweenSessions' param&quot;</td>
</tr>
<tr>
<td>11</td>
<td>&quot;Skipping event because 'isStopTracking' enabled&quot;</td>
</tr>
<tr>
<td>40</td>
<td>Network error: Error description comes from Android</td>
</tr>
<tr>
<td>41</td>
<td>&quot;No dev key&quot;</td>
</tr>
<tr>
<td>50</td>
<td>&quot;Status code failure&quot; + actual response code from the server</td>
</tr>
</tbody>
</table>

Setting additional custom data
The `setAdditionalData` API is required to integrate on the SDK level with several external partner platforms, including Segment, Adobe and Urban Airship. Use this API only if the integration article of the platform specifically states `setAdditionalData` API is needed.

`setAdditionalData` code example:

**Java**

```java
HashMap<String, Object> CustomDataMap = new HashMap<>();
CustomDataMap.put("custom_param_1","value_of_param_1");
AppsFlyerLib.getInstance().setAdditionalData(CustomDataMap);
```

**Kotlin**

```kotlin
val customDataMap = HashMap<String, Any>()
customDataMap.put("custom_param_1","value_of_param_1")
AppsFlyerLib.getInstance().setAdditionalData(customDataMap)
```

### 9. Sessions

**Custom time between sessions**

By default, at least 5 seconds must pass between two app launches to count as two separate sessions (more about counting sessions).

Use the following API to set the minimum time between sessions:
AppsFlyerLib.setMinTimeBetweenSessions(int seconds);

Setting a **high value to the custom time between launches may badly impact APIs** relying on sessions data, such as deep linking.

---

**Background sessions for utility apps**

You can report new user sessions using this SDK method. For example, this may be handy for utility apps that run in the background.

Use this API in your activity's onCreate():

```java
public void reportTrackSession(Context context);
```

**Usage Example:**

```java
AppsFlyerLib.getInstance().reportTrackSession(context);
```

---

**10. Owned media**

**Resolving wrapped deep link URLs**

Some 3rd party services such as email service providers wrap links in emails with their own click recording domains. Some even allow you to set your own click recording domains. If OneLink is wrapped in such domains, it might limit its functionality.

To overcome this issue you can use the `setResolveDeepLinkURLs` API. Use this API to get the OneLink from click domains that launch the app. Make sure to call this API before SDK initialization.

For example, you have three click domains that redirect to your OneLink which is https://mysubdomain.onelink.me/abCD. Use this API to get the OneLink that your click domains redirect to. This API method receives a list of domains that the SDK resolves. Add the
following code before SDK initialization.

```java
AppsFlyerLib.getInstance().setResolveDeepLinkURLs("clickdomain.com", "myclickdomain.com", "anotherclickdomain.com");
```

The code above allows you to use your click domain while preserving OneLink functionality. The click domains are responsible for launching the app. The API, in turn, gets the OneLink from these click domains and then you can use the data from this OneLink to deep link and customize user content.

### Recording push notifications

With AppsFlyer, you can record push notifications as part of retargeting campaigns.

To enable this feature, call the `sendPushNotificationData` method inside the `onCreate` method of every activity that is launched after clicking the notification:

```java
AppsFlyerLib.getInstance().sendPushNotificationData(this);
```

For more information on push notification measurement, read here.

### User invite attribution

Allowing your existing users to invite their friends and contacts as new users to your app, can be a key growth factor for your app. With AppsFlyer, you can attribute and record installs that originate from user invites within your app.

For details, see the User Invite Attribution article.

### Cross promotion attribution
Cross promoting apps can be a major growth factor in driving additional installs for your apps. AppsFlyer enables you to attribute and record installs originating from a cross-promotion of one of your apps from within the current app the user has.

For details, see the Cross Promotion Attribution article, here.

11. User identifiers

Get AppsFlyer ID

An AppsFlyer ID is created for every new install of an app. You can use AppsFlyer ID for various purposes:

- Send server-to-server in-app events.
- Match AppsFlyer ID with user records in your backend systems.
- Map entries when merging data from pull and push API.

Use the following API to obtain AppsFlyer ID:

```java
public String getAppsFlyerUID(Context context);
```

Usage example:

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

Set Customer User ID

Setting your own Customer User ID in the AppsFlyer SDK enables you to cross-reference your own unique ID with the AppsFlyer ID and other identifiers. The customer user ID is available in AppsFlyer raw data reports. You can also use it in postback APIs for cross-
referencing with your internal IDs.

To set your Customer User ID:

```java
public void setCustomerUserId(String id);
```

**Usage example:**

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

We recommend setting the Customer User ID early in the app's flow, as it is only associated with events reported after its setup:

- If `setCustomerUserId` is called before calling `startTracking`, the Customer User ID appears in the raw data reports for installs and for events.
- If it is set after, Customer User ID is only associated with events that are recorded after setting the Customer User ID.

**Getting Customer User ID:**

To avoid setting the Customer User ID value again beyond the first launch, and to reduce calls to your server to get the customer user ID, you can check if its value is empty or not by using:

```java
AppsFlyerProperties.getInstance().getString(AppsFlyerProperties.APP_USER_ID)
```

For more information about the Customer User ID, [click here](#).

**Delay SDK init for customerUser ID**

You can delay the SDK Initialization until the customerUser ID is set.

To indicate that the SDK should delay initialization for the Customer User ID call:

```java
AppsFlyerLib.getInstance().waitForCustomerUserId(true);
```

immediately before the `init()` method. The rest of the SDK initialization should remain unchanged.
Once the customerUserID has been provided, call

```java
AppsFlyerLib.getInstance().setCustomerIdAndTrack("customer_id", this);
```

to provide the SDK with the relevant Customer User ID and start the SDK.

The code should appear as follows:

### Java

```java
public class AFApplication extends Application {
    private static final String AF_DEV_KEY = "qrdZGj123456789";
    @Override
    public void onCreate() {
        super.onCreate();
        AppsFlyerConversionListener conversionDataListener = new AppsFlyerConversionListener()
            ...
        AppsFlyerLib.getInstance().waitForCustomerUserId(true);
        //WARNING! Removing above line doesn't cancel its effect.
        // Replace with this to stop waiting for CUID:
        // AppsFlyerLib.getInstance().waitForCustomerUserId(false);
        AppsFlyerLib.getInstance().init(AF_DEV_KEY, getConversionListener(), getApplicationContext());
        AppsFlyerLib.getInstance().startTracking(this);
        // Do your magic to get the customerUserID
        // ...
        // Call the following API once the customerUserID is available:
        AppsFlyerLib.getInstance().setCustomerIdAndTrack("customer_id", this);
    }
}
```

### Kotlin

```kotlin
public class AFApplication extends Application {
    private static final String AF_DEV_KEY = "qrdZGj123456789";
    @Override
    public void onCreate() {
        super.onCreate();
        AppsFlyerConversionListener conversionDataListener = object()
            ...
        AppsFlyerLib.getInstance().waitForCustomerUserId(true);
        //WARNING! Removing above line doesn't cancel its effect.
        // Replace with this to stop waiting for CUID:
        // AppsFlyerLib.getInstance().waitForCustomerUserId(false);
        AppsFlyerLib.getInstance().init(AF_DEV_KEY, getConversionListener(), getApplicationContext());
        AppsFlyerLib.getInstance().startTracking(this);
        // Do your magic to get the customerUserID
        // ...
        // Call the following API once the customerUserID is available:
        AppsFlyerLib.getInstance().setCustomerIdAndTrack("customer_id", this);
    }
}
```
```kotlin
class AFApplication: Application() {

    private val afDevKey = ""

    override fun onCreate() {
        super.onCreate()
        val conversionDataListener = object: AppsFlyerConversionListener {
            ...
        }
        AppsFlyerLib.getInstance().waitForCustomerUserId(true);  
        //WARNING! Removing above line doesn't cancel its effect. 
        // Replace with this to stop waiting for CUID:
        // AppsFlyerLib.getInstance().waitForCustomerUserId(false);
        AppsFlyerLib.getInstance().init(afDevKey, conversionDataListener, this)
        AppsFlyerLib.getInstance().startTracking(this)
        // Do your magic to get the customerUserID
        // ...
        // any AppsFlyer SDK code invoked here will be discarded
        // Call the following API once the customerUserID is available:
        AppsFlyerLib.getInstance().setCustomerIdAndTrack("customer_id", this)
    }
}
```

To learn more about delaying the SDK initialization until the Customer User ID is available, go [here](https://support.appsflyer.com/hc/en-us/articles/360014302437#introduction).

⚠️ **Warning**

Use this API only in cases where it is appropriate for your business logic. Using this API increases the chance for discrepancies and might make the app more exposed to fraud.

### Google Advertising ID

From SDK Version 4.8.0 AppsFlyer automatically collects the *google_advertising_id*.
The requirement to collect the Google Advertising ID is only relevant for SDK versions 4.7.X and below.

**OAID, IMEI, and Android ID**

At least one unique device identifier must be collected to enable attribution. The following identifiers are available: GAID, Android ID, IMEI, and OAID.

**GAID - Google Advertising ID**

Collected automatically from apps that contain Google Play Services. **If GAID is available IMEI and Android ID should NOT be collected by the app to avoid violation of the Google Play policy.**

**IMEI and Android ID**

Disabled by default. Can be collected ONLY for apps that do NOT contain Google Play Services. **Note:** Starting with Android 10 (API level 29), released in late 2019, access to the IMEI parameter is restricted.

**To send these IDs to AppsFlyer:**

1. On app open collect the device IMEI and/or Android ID.
2. Call the following APIs BEFORE calling the `startTracking` method:

   ```java
   AppsFlyerLib.getInstance().setImeiData("IMEI_HERE");
   AppsFlyerLib.getInstance().setAndroidIdData("ANDROID_ID_HERE");
   ```

**OAID - Open Advertiser Identifier**

**Guide to implementing OAID**

**Opting out of device ID collection**

Developers can opt-out of collecting IMEI, Android ID, and OAID by using the following APIs:
AppsFlyerLib.getInstance().setCollectIMEI(false);
AppsFlyerLib.getInstance().setCollectAndroidID(false); AppsFlyerLib.getInstance().setCollectOaid(false);

12. User privacy

Opt-out

In some extreme cases, you might want to shut down all SDK tracking due to legal and privacy compliance. This can be achieved with the `isStopTracking` API. Once this API is invoked, the SDK stops functioning and no longer communicates with AppsFlyer servers.

AppsFlyerLib.getInstance().stopTracking(true, context);

There are several different scenarios for user opt-out. **We highly recommend following the exact instructions for the scenario, that is relevant for your app.**

In any event, the SDK can be reactivated by calling the same API, by passing false.

**Important**

Do not call `startTracking` if `isStopTracking` is set to `true`.

To start tracking again once `stopTracking` is set to `false`, use the following SDK API:

AppsFlyerLib.getInstance().startTracking(getApplicationContext(), AF_DEV_KEY);

**Warning**
Use the stopTracking API only in cases where you want to fully ignore this user from any and all tracking. Using this API SEVERELY impacts your attribution, data collection and deep linking mechanism.

### Anonymize user data

AppsFlyer provides you with a method to anonymize specific user identifiers in AppsFlyer analytics. This method complies with the latest privacy requirements and with Facebook data and privacy policies. Its default value is NO, meaning anonymization isn't performed by default.

Use this API during the SDK Initialization to explicitly anonymize a user's install, events and sessions:

```java
public void setDeviceTrackingDisabled(boolean isDisabled);
```

**Usage example:**

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

Tracking can be restarted by calling `deviceTrackingDisabled` set to **false**.

⚠️ **Warning**

Anonymizing users SEVERELY impacts your attribution information.
Use this option ONLY for regions which legally prevents you from collecting your users' information.

### Exclude partners from getting data

In some cases, advertisers may want to stop sharing user-level data with ad networks/partners for specific users. Reasons for this include:
AppsFlyer provides two API methods to stop sharing data with some or all partners:

- **setSharingFilter**: Used by advertisers to set some (one or more) networks/integrated partners to **exclude** from getting data.
- **setSharingFilterForAllPartners**: Used by advertisers to exclude all networks/integrated partners from getting data.

These filtering methods are supported as of SDK V5.4.1.

The filtering method **must be called every time the SDK is initialized** and affects the whole session. If it takes time to determine whether you need to set the sharing filters, then **delay the SDK initialization**.

When the method is activated before the first **startTracking** call:

- **Users from SRNs** are attributed as Organic, and their data is not shared with integrated partners.
- **Users from click ad networks** (non-SRNs) are attributed correctly in AppsFlyer, but not shared with the ad networks via postbacks, APIs, raw data reports, or by any other method.

Currently, uninstall data can't be filtered using these methods. However, you can stop sending Uninstall events to partners using their **setup pages** in AppsFlyer.

### getAppsFlyerUID

**Description**
Get AppsFlyer ID. For more information see [here](https://support.appsflyer.com/hc/en-us/articles/360014302437#introduction).

**Method signature**
```
public String getAppsFlyerUID(Context context); //
```
<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>Get AppsFlyer ID. For more information see here.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage example</strong></td>
<td>String appsFlyerId = AppsFlyerLib.getInstance().getAppsFlyerUID(this);</td>
</tr>
</tbody>
</table>

### onAppOpenAttribution

**Description** Get deep link data when the app opens via a deep link.

**Method signature**

```java
public void onAppOpenAttribution(Map<String, String> conversionData)
```

### onAttributionFailure

**Description** Handle errors in getting deep link data.

**Method signature**

```java
public void onAttributionFailure(String errorMessage)
```

### onConversionDataSuccess

**Description** Get conversion data after an install. Useful for deferred deep linking.

**Note:** Starting SDK V5, `onConversionDataSuccess` is the name of the method for getting conversion data. If you are using an SDK version lower than 5.0.0, the name of the method is `onConversionDataReceived`. We recommend that you upgrade to SDK 5.0.0. To learn more, click here.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Method signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>onConversionDataSuccess</td>
<td>Get conversion data after an install. Useful for deferred deep linking. Starting SDK V5, onConversionDataSuccess is the name of the method for getting conversion data. If you are using an SDK version lower than 5.0.0, the name of the method is onConversionDataReceived. We recommend that you upgrade to SDK 5.0.0. To learn more, click here.</td>
<td>public void onConversionDataSuccess(Map&lt;String, String&gt; conversionData)</td>
</tr>
<tr>
<td>onConversionDataFail</td>
<td>Handle errors when failing to get conversion data from installs.</td>
<td>public void onConversionDataFail(String errorMessage)</td>
</tr>
<tr>
<td>onTrackingRequestFailure</td>
<td>Callback method for AppsFlyerTrackingRequestListener. Called when the SDK fails to report app launch.</td>
<td>public void onTrackingRequestFailure(String error);</td>
</tr>
<tr>
<td>Usage example</td>
<td>See setting a tracking request listener.</td>
<td></td>
</tr>
<tr>
<td>onTrackingRequestSuccess</td>
<td>Callback method for AppsFlyerTrackingRequestListener. Called when the SDK successfully reports app launch.</td>
<td></td>
</tr>
</tbody>
</table>
### onTrackingRequestSuccess

**Description**
Callback method for AppsFlyerTrackingRequestListener. Called when the SDK successfully reports app launch.

**Method signature**
```java
public void onTrackingRequestSuccess();
```

**Usage example**
See setting a tracking request listener.

### performOnAppAttribution

**Description**
This function allows developers to manually re-trigger onAppOpenAttribution with a specific link (URI or URL), without recording a new re-engagement. This method may be required if the app needs to redirect users based on the given link, or resolve the AppsFlyer short URL while staying in the foreground/opened. This might be needed because regular onAppOpenAttribution callback is only called if the app was opened with the deep link.

**Method signature**
```java
public void performOnAppAttribution(Context context, Uri uri);
```

**Usage example**

**Java**

```java
AppsFlyerLib.getInstance().performOnAppAttribution(Context, uri);
```

**Kotlin**

```java
AppsFlyerLib.getInstance().performOnAppAttribution(Context, uri)
```

### reportTrackSession

https://support.appsflyer.com/hc/en-us/articles/360014302437#introduction
**Description**
Report sessions if your app is a utility app that runs in the background.

**Method signature**
```
public void reportTrackSession(Context context);
```

**Usage example**
```
AppsFlyerLib.getInstance().reportTrackSession(this);
```

---

**sendDeepLinkData (deprecated from V5.3.0)**

**Description**
This method is no longer used to make sure you get attribution data even if the user is deep linked into a specific activity. Instead, ensure that startTracking() has been called.
For more information, see [here](#).

**Method signature**
```
public void sendDeepLinkData(Activity activity);
```

**Usage example**
```
AppsFlyerLib.getInstance().sendDeepLinkData(this);
```

---

**sendPushNotificationData**

**Description**
Measure and get data from push notifications campaigns. For more information, see [measuring push notifications](#).

**Method signature**
```
public void sendPushNotificationData(Activity activity);
```
### Description
Measure and get data from push notifications campaigns. For more information, see [measuring push notifications](https://support.appsflyer.com/hc/en-us/articles/360014302437#introduction).

### Usage example
```java
AppsFlyerLib.getInstance().sendPushNotificationData(this);
```

---

### setAddtionalData

**Description**
Adding additional data to be sent to external partner platforms.

**Method signature**
```java
public void setAdditionalData(HashMap<String, Object> customData);
```

**Usage example**
See [setting additional data](https://support.appsflyer.com/hc/en-us/articles/360014302437#introduction).

---

### setAndroidIdData

**Description**
Send Android ID to AppsFlyer. See [OAID, IMEI and Android ID](https://support.appsflyer.com/hc/en-us/articles/360014302437#introduction).

**Method signature**
```java
public void setAndroidIdData(String aAndroidID);
```

**Usage example**
```java
AppsFlyerLib.getInstance().setImeiData("Android ID");
```

---

### setAppInviteOneLink

---
### setAppInviteOneLink

**Description**
Set the OneLink template ID that is used to create custom attribution links for user invites.

**Method signature**
```java
public void setAppInviteOneLink(String oneLinkId);
```

**Usage example**
See setting OneLink for user invite attribution.

### setCollectAndroidID

**Description**
Indicate whether Android ID should be sent to AppsFlyer.

**Method signature**
```java
public void setCollectAndroidID(boolean isCollect);
```

**Usage example**
See OAID, IMEI and Android ID.

### setCollectIMEI

**Description**
Indicate whether IMEI should be sent to AppsFlyer.

**Method signature**
```java
public void setCollectIMEI(boolean isCollect);
```

**Usage example**
See OAID, IMEI and Android ID.

### setCustomerIdAndTrack
Description
Initiates the SDK once a Customer User ID is available. For more information, see delay SDK init for Customer User ID.

Method signature
public void setCustomerIdAndTrack(String id, @NonNull Context context);

Usage example
AppsFlyerLib.getInstance().setCustomerIdAndTrack(customer_id", this);

setCustomerId

Description
Set the Customer User ID. For more information, see setting the Customer User ID.

Method signature
public void setCustomerId(String id)

Usage example
AppsFlyerLib.getInstance().setCustomerId("customer_id")

setDebugLog

Description
Enable debugging logs. See debugging for Android.

Method signature
public void setDebugLog(boolean shouldEnable);

Usage example
AppsFlyerLib.getInstance().setDebugLog(true);
### setDeviceTrackingDisabled

**Description**
Anonymize a user's installs, events, and sessions. For more information, see [anonymizing user data](https://support.appsflyer.com/hc/en-us/articles/360014302437#introduction).

**Method signature**
```java
public void setDeviceTrackingDisabled(boolean isDisabled);```

**Usage example**
```java
AppsFlyerLib.getInstance().setDeviceTrackingDisabled(true);
```

### setLogLevel

**Description**
Set the AppsFlyer SDK log level.

**Method signature**
```java
public void setLogLevel(AFLogger.LogLevel logLevel)
```

**Usage example**
```java
AppsFlyerLib.getInstance().setLogLevel(AFLogger.LogLevel.INFO);
```

### setMinTimeBetweenSessions

**Description**
Set the minimum time between sessions. For more information, see [custom time between sessions](https://support.appsflyer.com/hc/en-us/articles/360014302437#introduction).

**Method signature**
```java
public void setMinTimeBetweenSessions(int seconds);
```
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Usage example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>setMinTimeBetweenSessions()</code></td>
<td>Set the minimum time between sessions. For more information, see <a href="https://support.appsflyer.com/hc/en-us/articles/360014302437#introduction">custom time between sessions</a>.</td>
<td><code>AppsFlyerLib.getInstance().setMinTimeBetweenSessions(10);</code></td>
</tr>
<tr>
<td><code>setOaidData()</code></td>
<td>Send OAID to AppsFlyer. See <a href="https://support.appsflyer.com/hc/en-us/articles/360014302437#introduction">OAID, IMEI and Android ID</a>.</td>
<td><code>AppsFlyerLib.getInstance().setOaidData(&quot;OAID&quot;);</code></td>
</tr>
<tr>
<td><code>setOutOfStore()</code></td>
<td>Specify the alternative app store that the app is downloaded from.</td>
<td><code>AppsFlyerLib.getInstance().setOutOfStore(&quot;baidu&quot;);</code></td>
</tr>
</tbody>
</table>

- `AppsFlyerLib` is a library for integrating with AppsFlyer, a mobile app analytics and attribution platform.
- The `getInstance()` method is used to obtain an instance of the library.
- Method signatures are shown in the format: `public void <method name>(<parameter>);`
<table>
<thead>
<tr>
<th>Description</th>
<th>Set the SDK to report pre-install when a pre-installed app launches.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Method signature</strong></td>
<td>public void setPreinstallAttribution(String mediaSource, String campaign, String siteId);</td>
</tr>
<tr>
<td><strong>Usage example</strong></td>
<td>See pre-install campaigns for Android.</td>
</tr>
</tbody>
</table>

**setResolveDeepLinkURLs**

<table>
<thead>
<tr>
<th>Description</th>
<th>Resolve OneLink from click domains. For more information, see resolving wrapped deep link URLs.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Method signature</strong></td>
<td>public void setResolveDeepLinkURLs(String... urls);</td>
</tr>
<tr>
<td><strong>Usage example</strong></td>
<td>AppsFlyerLib.getInstance().setResolveDeepLinkURLs(&quot;example.com&quot;, &quot;click.example.com&quot;);</td>
</tr>
</tbody>
</table>

**setSharingFilter**

<table>
<thead>
<tr>
<th>Description</th>
<th>Used by advertisers to set some (one or more) networks/integrated partners to exclude from getting data. Learn more</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Method signature</strong></td>
<td>AppsFlyerLib.getInstance().setSharingFilter(String... partners);</td>
</tr>
<tr>
<td><strong>Usage example</strong></td>
<td>AppsFlyerLib.getInstance().setSharingFilter(&quot;facebook_int&quot;, &quot;googleadwords_int&quot;, &quot;snapchat_int&quot;, &quot;doubleclick_int&quot;);</td>
</tr>
</tbody>
</table>
### setSharingFilterForAllPartners

**Description**  
Used by advertisers to exclude **all** networks/integrated partners from getting data.  
[Learn more](https://support.appsflyer.com/hc/en-us/articles/360014302437#introduction)

**Method signature**  
`AppsFlyerLib.getInstance().setSharingFilterForAllPartners();`

**Usage example**  
`AppsFlyerLib.getInstance().setSharingFilterForAllPartners();`

---

### startTracking

**Description**  
Start the SDK on app launch. For more information, see [initializing the SDK](https://support.appsflyer.com/hc/en-us/articles/360014302437#introduction).

**Method signature**  
```java
public void startTracking(Application application);
```

**Usage example**  
`AppsFlyerLib.getInstance().startTracking(this);`

---

### stopTracking

**Description**  
Shut down all SDK functionality. For more information, see [user privacy - opt-out](https://support.appsflyer.com/hc/en-us/articles/360014302437#introduction).

**Method signature**  
```java
public void stopTracking(boolean isTrackingStopped, Context context);
```
Shut down all SDK functionality. For more information, see user privacy - opt-out.

AppsFlyerLib.getInstance().stopTracking(true, this);

This method is deprecated. Use startTracking instead.
Has two functions:
- Send an immediate app launch event to AppsFlyer.
- Restart SDK functionality if SDK functionality was shut down using stopTracking.

public void trackAppLaunch(Context context, String devKey);

AppsFlyerLib.getInstance().trackAppLaunch(this, AF_DEV_KEY);

Send in-app events to AppsFlyer. For more information, see recording in-app events.

public void trackEvent(Context context, String eventName, Map<String, Object> eventValues);

AppsFlyerLib.getInstance().trackEvent(this, AFINAppEventType.PURCHASE, eventValues);
**updateServerUninstallToken**

**Description**
For developers who use Firebase for other purposes other than uninstall measurement. For more information, see uninstall measurement.

**Method signature**
```java
public void updateServerUninstallToken(Context context, String token)
```

**Usage example**
```java
AppsFlyerLib.getInstance().updateServerUninstallToken(getApplicationContext(), TOKEN)
```

**waitForCustomerUserId**

**Description**
Delay SDK initialization until Customer User ID is set.

**Method signature**
```java
public void waitForCustomerUserId(boolean wait)
```

**Usage example**
```java
AppsFlyerLib.getInstance().waitForCustomerUserId(true)
```

**Deprecated APIs**

A **deprecated** API means the method will be replaced. This is a good time for developers to upgrade their code.

The **sunset date** is when the method stops working or continues working with limited capabilities.

<table>
<thead>
<tr>
<th>API name</th>
<th>Deprecated from version</th>
<th>Sunset date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>API name</td>
<td>Deprecated from version</td>
<td>Sunset date</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>sendDeepLinkData</td>
<td>5.3.0</td>
<td>TBD</td>
</tr>
<tr>
<td>trackAppLaunch</td>
<td>5.2.0</td>
<td>TBD</td>
</tr>
</tbody>
</table>