At a glance: The Unity V4 plugin is deprecated. We recommend migrating to the Unity V6 plugin to fully enable app attribution and reporting for mobile apps developed on the Unity development platform.

Migrating to Unity plugin V6

A Unity plugin V6 is available, allowing app owners and developers to support iOS 14. The V6 plugin also includes significant API method changes from previous versions. To update your plugin from an earlier (V4) version, requires removing/replacing the old plugin. See the full Unity V6 plugin integration guide for developers.

1. Overview - Unity V4

AppsFlyer Unity SDK V4 provides Android and iOS Unity projects with mobile app installation and event recording functionalities. Measure installs, sessions, and in-app events to evaluate ROI and app-user engagement levels.

Mobile apps developed on the Unity platform only need to integrate AppsFlyer's SDK once in order to support attribution installs for both Android and iOS generated apps.

This guide explains how to integrate AppsFlyer's SDK into your Unity code for use in iOS and Android apps.

1.1 SDK integration overview
<table>
<thead>
<tr>
<th>Tab</th>
<th>Content</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>[Mandatory]</strong> Integration</td>
<td>How to add and configure the SDK.</td>
<td>App dashboard shows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- New organic install</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- New non-organic install</td>
</tr>
<tr>
<td><strong>[Recommended]</strong> Core APIs</td>
<td>How to use SDK core APIs to:</td>
<td>App dashboard shows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- In-app events</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Revenue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ready to:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Perform deep linking</td>
</tr>
<tr>
<td>Additional APIs</td>
<td>How to implement and use optional APIs for:</td>
<td>Ready to:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Measure uninstalls</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Referrals</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- User engagements with push notifications</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Handle user privacy scenarios and more.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>API reference</td>
<td>Quick SDK API reference for developers</td>
<td></td>
</tr>
</tbody>
</table>

### 1.2 SDK compatibility

AppsFlyer SDK supports:

- Integration with Unity version 5 and later.
- **Google Play Install Referrer API** starting from Unity plugin v4.16.0.
  - To use the new Referrer API, update the plugin to v4.16.0 or later.

App developers, this tab content is for you. After you implement and initialize the AppsFlyer SDK, you'll see two installs in your app's AppsFlyer Overview dashboard—one organic, the other non-organic.

### 2. Add the SDK to your app

⭐ **Important!**

AppsFlyer Unity SDK doesn't support Unity Internal Build System.

#### 2.1 Download AppsFlyer's Unity plugin
Use this Github link to download the plugin: https://github.com/AppsFlyerSDK/Unity.

## 2.2 Install the plugin

1. Import the AppsFlyerUnityPlugin.unitypackage into your Unity project.
2. Go to Assets > Import Package > Custom Package.

## 2.3 Set up Android and iOS

### Android setup

#### Set required permissions

AndroidManifest.xml includes these permissions:

```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" />
```

#### Set Android install referrer

The Android install referrer improves attribution accuracy, protects from install fraud and more.

Once you have added the AppsFlyer plugin to Unity, the following happens automatically:

1. The `installreferrer-1.0.aar` file is added to the project (if the checkbox wasn’t unticked when importing the plugin).
2. The following line is added to the `build.gradle` file:

   ```groovy
   implementation (name: 'installreferrer-1.0', ext:'aar')
   ```

### Set BroadcastReceiver in AndroidManifest.xml

---

**Note**

Google deprecated the BroadcastReceiver in March 2020. This change doesn't affect your app but it means that implementing the installreferrer is now mandatory. You don't have to do anything though. As mentioned, if you use the latest Unity SDK, installreferrer is automatically added to your project. You might still need the
BroadcastReceiver for out-of-store attribution. To make sure, check with the store where you list your app.

**iOS setup**

**Linked Frameworks and Libraries**
You used Unity for xCode to build your project. Now, if it wasn't included earlier, add `Security.framework` to xCode's **Linked Frameworks and Libraries**.

![Linked Frameworks and Libraries](image)

**Note**
If you are using SDK version 4.8.11 and above, the AdSupport.framework is added automatically; you do not need to add the frameworks manually.

**iOS Apple Search Ads**

It is necessary to add the following:

**AdSupport.framework**
If included: AppsFlyer collects IDFA.
If excluded: AppsFlyer cannot attribute installs to Facebook, Twitter, and most other ad networks.

**iAd.framework**
[Recommended] Required in order to attribute Apple Search Ads.

3. Implement and initialize the SDK
This section describes how to implement and initialize AppsFlyer Unity SDK.

### 3.1 Retrieve your dev key

Your AppsFlyer dev key is unique and identifies your account. Only account admin can retrieve the dev key.

Use is mandatory as it allows the SDK to securely send and retrieve data belonging to your AppsFlyer account.

To do:

1. Go to the AppsFlyer platform.
2. In the menu bar, go to **Configuration** and click **App Settings**.
3. Under **SDK installation**, copy your dev key.

![APP SETTINGS](image)

### 3.2 Initialize the SDK

**Step 1: Create an empty AppsFlyer object**

To initialize the SDK, create an empty AppsFlyer object.

1. From the top menu select **GameObject** > **Create Empty**.
2. Select the object and name it AppsFlyerObject.
3. Click **Add Component**.
4. Choose **New script** and name it AppsFlyerObject.

**Step 2: Add init code to AppsFlyerObject**

Put the following code in the new script that you created in step 1.

**Unity Plugin 4.15.1 and later**
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AppsFlyerObject : MonoBehaviour {

    void Start () {

        /* Mandatory - set your AppsFlyer's Developer key. */
        AppsFlyer.setAppsFlyerKey("YOUR_APPSFLYER_DEV_KEY");

        /* For detailed logging */
        /* AppsFlyer.setIsDebug (true); */

        #if UNITY_IOS
            /* Mandatory - set your apple app ID
             * NOTE: You should enter the number only and not the "ID" prefix */
            AppsFlyer.setAppID("YOUR_APP_ID_HERE");

            AppsFlyer.getConversionData();
            AppsFlyer.trackAppLaunch();

            #else if UNITY_ANDROID
                /* For getting the conversion data in Android, you need to add the "AppsFlyerTrackerCallbacks" listener.*/
                AppsFlyer.init("YOUR_APPSFLYER_DEV_KEY","AppsFlyerTrackerCallbacks");
            #endif
        }
    }
}

Unity Plugin 4.14.3 and earlier

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AppsFlyerObject : MonoBehaviour {

    void Start () {

        /* Mandatory - set your AppsFlyer's Developer key. */
        AppsFlyer.setAppsFlyerKey("YOUR_APPSFLYER_DEV_KEY");

        /* For detailed logging */
        /* AppsFlyer.setIsDebug (true); */

        #if UNITY_IOS
            /* Mandatory - set your apple app ID
             * NOTE: You should enter the number only and not the "ID" prefix */
            AppsFlyer.setAppID("YOUR_APP_ID_HERE");

            AppsFlyer.getConversionData();
            AppsFlyer.trackAppLaunch();

            #else if UNITY_ANDROID
                AppsFlyer.init("YOUR_APPSFLYER_DEV_KEY");
            #endif
        }
    }
}

⭐ Important!

- **Android:** AppsFlyer.init includes calling trackAppLaunch. Therefore, do not call trackAppLaunch in the Android build of the Unity plugin.
Step 3: Create an empty AppsFlyerTrackerCallbacks object

1. From the top menu select **GameObject > Create Empty**.
2. Select the object and name it AppsFlyerTrackerCallbacks.
3. Click **Add Component**.
4. Select **AppsFlyerTrackerCallbacks** script.

Step 4: Specify Android package in Unity project

1. In Unity, Click **File > Build Settings**.
2. Choose **Android Platform > Player Settings**.
3. Under **Player**, select **Other Settings > Identification > Package Name**.
4. Specify the Android package name in the **Package Name** field.

4. Test installs

The time has come to test the SDK integration by simulating organic and non-organic installs.

4.1 Whitelist your test device

First, whitelist the device you'll use to test installs.

4.2 Simulate an organic install

Organic installs are unattributed installs; they are normally the result of installs made directly via app stores.

To simulate an organic install, see the relevant operating system instructions:

- **Android**
- **iOS**

4.3 Simulate a non-organic install

Non-organic installs are attributed installs; they are usually the result of an ad engagement.

To simulate a non-organic install (using attribution links), see the relevant operating system instructions:
Now it's time to explain how to record in-app events and revenue, as well as setting up deep linking. Recording in-app events and revenue lets you measure the quality of your app users, while deep linking allows you to give app users a better user experience.

In this tab, app developers will find relevant instructions. However, app owner/marketer input is essential. The app owner,

- Decides which in-app events to record in order to measure user quality.
- Needs to access the AppsFlyer platform > menu bar to set up OneLink for deep linking.

5. Record in-app events

Record in-app events to measure KPIs such as ROI (Return on Investment) and LTV (Lifetime Value).
We recommend defining the in-app events you want to record.

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

To send events:

- Specify the event's name and parameters.
- See related lists:
  - Listing of recommended event names and structures
  - List found in AFInAppEvents class.

**Recommended practice!** Use event names and parameters for the following reasons:

- **Standard naming:** AppsFlyer can automatically map events to SRNs such as Facebook, Google, Twitter, and Snapchat.
- **Backward compatibility:** No issue arises if AppsFlyer changes an event name or parameter as your implementation is backward compatible.
5.2 Record revenue

Send revenue with any in-app event by using the `af_revenue` event parameter:

- Populate `af_revenue` with any numeric value, positive or negative.
- It's the ONLY event parameter considered as real revenue in raw data reports and on the dashboard.
  
  **Good practice!** Learn about currency settings, display, and currency conversion.

Consider these currency code factors when sending events with revenue:

- Default currency: USD
- Set currency codes as a 3-character ISO 4217 code (see example below).
- Set the currency code for all events by calling this API:
  ```csharp
  AppsFlyer.setCurrencyCode("ZZZ")
  ```

- Revenue values cannot contain comma separators, currency symbols, or text.
  Example revenue event: 1234.56

**Example: In-app purchase event with revenue**

This purchase event has $200 in revenue and appears as revenue on the dashboard.

```csharp
System.Collections.Generic.Dictionary<string, string> ();
purchaseEvent.Add(AFInAppEvents.CURRENCY, "USD");
purchaseEvent.Add(AFInAppEvents.REVENUE, "200");
purchaseEvent.Add(AFInAppEvents.QUANTITY, "1");
purchaseEvent.Add(AFInAppEvents.CONTENT_TYPE, "category_a");
AppsFlyer.trackRichEvent ("af_purchase", purchaseEvent);
```

Record negative revenue

Recording negative revenue is sometimes required. The below code features the following:

- Revenue value is preceded by a minus sign.
- The event name has a unique value, "cancel_purchase". This lets you identify negative revenue events in raw data reports and on the dashboard.

**Example: App user receives a refund or cancels a subscription.**
purchaseEvent.Add(AFInAppEvents.CURRENCY, "USD");
purchaseEvent.Add(AFInAppEvents.REVENUE, "-200");
purchaseEvent.Add(AFInAppEvents.QUANTITY, "1");
purchaseEvent.Add(AFInAppEvents.CONTENT_TYPE, "category_a");
AppsFlyer.trackRichEvent ("cancel_purchase", purchaseEvent);

5.3 In-app purchase validation

The AppsFlyer SDK provides server verification for in-app purchases. To validate a purchase, see the relevant operating system instructions:

Android Call

//To get the callbacks
AppsFlyer.createValidateInAppListener ("AppsFlyerTrackerCallbacks", "onInAppBillingSuccess", "onInAppBillingFailure");
AppsFlyer.validateReceipt (string publicKey, string purchaseData, string signature, string price, string currency, Dictionary additionalParameters);

Call the Listener (Android only)

AppsFlyer.createValidateInAppListener ("AppsFlyerTrackerCallbacks", "onInAppBillingSuccess", "onInAppBillingFailure");

Note! The purchaseData variable must include the JSON from the original purchase.

iOS

Ensure you test against the Apple sandbox server call:

if (Debug.isDebugBuild) {
    AppsFlyer.setIsSandbox (true);
}
AppsFlyer.validateReceipt (string productIdentifier, string price, string currency, string transactionId, Dictionary additionalParameters);

Method parameters

Android
### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String publicKey</td>
<td>Public key from Google Developer Console</td>
</tr>
<tr>
<td>String signature</td>
<td>Transaction signature; returned by Google API when a purchase is completed</td>
</tr>
<tr>
<td>String purchaseData</td>
<td>Product purchased in JSON format; returned by Google API when a purchase is completed</td>
</tr>
<tr>
<td>String revenue</td>
<td>In-app event revenue to be reported to AppsFlyer</td>
</tr>
<tr>
<td>String currency</td>
<td>In-app event currency to be reported to AppsFlyer</td>
</tr>
<tr>
<td>Dictionary&lt;String, String&gt;</td>
<td>Additional in-app event parameters which appear in the event_value field in in-app event raw data</td>
</tr>
<tr>
<td>additionalParameters</td>
<td></td>
</tr>
</tbody>
</table>

### iOS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String productIdIdentifier</td>
<td>Product identifier</td>
</tr>
<tr>
<td>String price</td>
<td>In-app event revenue to be reported to AppsFlyer</td>
</tr>
<tr>
<td>String currency</td>
<td>In-app event currency to be reported to AppsFlyer</td>
</tr>
<tr>
<td>Dictionary&lt;String, String&gt;</td>
<td>Additional in-app event parameters which appear in the event_value field in in-app event raw data</td>
</tr>
<tr>
<td>additionalParameters</td>
<td></td>
</tr>
</tbody>
</table>

---

**Note**

Call `validateReceipt` to automatically generate an `af_purchase` in-app event.

- Then, it isn't necessary to send a purchase event after validating a purchase.
If another purchase event is sent after validating, this results in duplicate event reporting.

5.4 In-app event considerations

- Event name: maximum of 45 characters
- Event value: must not exceed 1000 characters - if longer we may truncate it
- Supports non-English characters for in-app events (and other SDK APIs) beginning from version 4.8.1 for both iOS and Android
- Pricing and revenue:
  - Only use numbers and decimals such as 5 or 5.2
  - Up to 5 numbers after the decimal such as 5.12345

5.5 Examples of recording in-app events

Record in-app events by calling `trackRichEvent` and include the event name and value parameters.

See In-app events for more details.

**Example: How to record an in-app purchase event**

For a comprehensive list of ready-made code snippets per vertical, see our guide for rich in-app events per verticals.

```csharp
purchaseEvent.Add(AFInAppEvents.CURRENCY, "USD" );
purchaseEvent.Add(AFInAppEvents.REVENUE, "200" );
purchaseEvent.Add(AFInAppEvents.QUANTITY, "2" );
purchaseEvent.Add(AFInAppEvents.CONTENT_TYPE, "category_a" );
purchaseEvent.Add(AFInAppEvents.CONTENT_ID, "092" );
AppsFlyer.trackRichEvent (AFInAppEvents.PURCHASE, purchaseEvent);
```

5.6 Record offline in-app events

Sometimes users produce in-app events, while they have no internet connection. However, AppsFlyer is able to record the event. How does this work?

1. SDK sends events to AppsFlyer servers and waits for a response.
2. If the SDK doesn't get a 200 response, the event is stored in the cache.
3. After receiving the next 200 response, the cached event is re-sent to the server.
6. Deep link with OneLink

AppsFlyer’s OneLink is the solution for multi-platform attribution: **redirection** and **deep linking**.

6.1 Device detection and redirection

OneLink:

- Detects the type of device (Android and iOS, desktop, etc.) when a user clicks, then
- Redirects the user to the correct destination: Google Play, iOS app store, out-of-store markets, or web pages.

To implement multi-platform attribution links (no SDK coding required) and review the basics of deep linking, see the OneLink redirection guide.

6.2 Deep linking

Use deep linking to direct existing users to specific activities and customized content.

An app owner and a developer must work together to set up deep linking with OneLink:

- **App owner must access the AppsFlyer dashboard**
- **Developer must access the app**

See setting up deep linking with OneLink.

6.3 Deferred deep linking
Deferred deep linking lets you deep link new users to direct them to specific activities and customized content with the first launch of an app install.

Standard deep linking also directs users to specific activities and customized content, but an app must already be installed on the user's device.

To set up deferred deep linking with OneLink:

- Developer needs access to the AppsFlyer platform.
- AppsFlyer platform setup for deferred and standard deep linking is the same.
- Must implement additional logic in the app to deep link users and provide them with customized content after they install and launch the app.

Check deferred deep linking for more information.

6.4 Get deep-link data

The AppsFlyer SDK provides conversion or engagement data after every install or deep-linking event. Use this data to customize content and/or the app's programmatic behavior.

To receive deep-link data:

- Implement callback `onAppOpenAttribution` (found in `AppsFlyerTrackerCallbacks` class); it is called by the AppsFlyer SDK.
- Returned OneLink/attribution link parameters trigger an app to open.
- Parse the values and apply the logic to trigger the relevant app page.

```csharp
public void onAppOpenAttribution(string validateResult) {
    print("AppsFlyerTrackerCallbacks:: got onAppOpenAttribution = " + validateResult);
}
```

Check deep linking data for more information.

7. Get conversion data

The SDK provides access to real-time user attribution data for every new install. This enhances user engagement with the following:

- Personalized content
- Sending them to specific activities within an app. See deferred deep linking in this article.

To obtain AppsFlyer conversion data:

1. Add an empty `GameObject` and name it `AppsFlyerTrackerCallbacks`.
2. Attach to `GameObject` the `AppsFlyerTrackerCallbacks.cs`. 
See API reference for `onConversionDataSuccess`.

### Android Unity Plugin > 4.15.1

```csharp
/*To get conversion data for Android, add this listener to the init() method*/
AppsFlyer.init("YOUR_DEV_KEY", "AppsFlyerTrackerCallbacks");
```

- Move methods from `AppsFlyerTrackerCallbacks.cs` to another class.
- Name the new class as the listener.

**Note!** Must use the exact same method namespaces that appear on `AppsFlyerTrackerCallbacks`.

### Android Unity Plugin < 4.14.3

```csharp
/*To get conversion data for Android, add this listener.*/
AppsFlyer.loadConversionData("AppsFlyerTrackerCallbacks");
```

- Move methods from `AppsFlyerTrackerCallbacks.cs` to another class.
- Name the new class as your listener.

**Note!** Must use the exact same method namespaces that appear on `AppsFlyerTrackerCallbacks`.

### iOS

```csharp
/*To get conversion data. This is triggered on AppsFlyerTrackerCallbacks.cs file */
AppsFlyer.getConversionData();
```

**Example:**

`AppsFlyerTrackerCallbacks.cs class`

```csharp
public void didReceiveConversionData(string conversionData) {
    print("AppsFlyerTrackerCallbacks:: got conversion data = " + conversionData);
}
```

Check conversion data scenarios for more information.

### 8. Attribution

**Measure uninstalls**

Measure the uninstall rate of users coming from different sources. This significant KPI helps you analyze and optimize your campaigns.

To learn how to setup uninstall measurement, see the relevant operating system instructions.

---

**Android - Firebase**

https://support.appsflyer.com/hc/en-us/articles/213766183-Unity-plugin-V4-deprecated-#introduction
1. Download the Unity Firebase SDK from https://firebase.google.com/docs/unity/setup.
2. Import `FirebaseMessaging.unitypackage` into the project.
3. Import `google-services.json` into the project (obtained from the Firebase console).

**Note**

Unity Firebase SDK automatically adds manifest receivers.

4. Add this code to the Unity class handling the AppsFlyer code:

```csharp
using Firebase.Messaging;
using Firebase.Unity;

FirebaseMessaging.FirebaseMessaging.TokenReceived += OnTokenReceived;

public void OnTokenReceived(object sender, Firebase.Messaging.TokenReceivedEventArgs token) {
    AppsFlyer.updateServerUninstallToken(token.Token);
}
```

Add to Start() method:

```csharp
public void Start() {
    // register to push notifications
    AppsFlyer.registerUninstall("device_push_notification_token");
}
```

**Warning**

**Scenario:** Implementing a Unity Firebase SDK.

**Requirement:** Do not add the following method call to `enableUninstallTracking("SenderID")` if you added the Firebase Unity SDK to your project.

**Consequences if added:**

- Firebase Unity SDK will get the sender ID from the `google-services.json` file (that was added earlier).
- As a result, this may cause a debug warning from Android.

**iOS**

Get your device token from `UnityEngine.iOS.NotificationServices.deviceToken`.

Call this method when you receive your device token (tab content issues):

```csharp
AppsFlyer.registerUninstall("device_push_notification_token");
```

**Example:**

```csharp
void Start () {
    // register to push notifications
}```
To integrate on the SDK level with several external partner platforms (including Segment, Adobe, and Urban Airship), it is necessary to use the `setAdditionalData` API. Only use this API if the partner's integration article specifically states the `setAdditionalData` API is required.

**Example:** `setAdditionalData` code

```csharp
Dictionary<string, string> CustomDataMap = new Dictionary<string, string>();
CustomDataMap.Add("custom_param_1", "value_of_param_1");
AppsFlyer.setAdditionalData(CustomDataMap);
```

9. Sessions

**Custom time between sessions**

By default, at least 5 seconds must pass between two app launches for them to count as two separate sessions. Use this API to set the minimum time between sessions:

```csharp
 AppsFlyer.setMinTimeBetweenSessions(custom_time_ins_seconds);
```

**Note!** If you set a high value for the custom time between launches, this can negatively impact APIs that rely on session data (such as deep linking).

See more about counting app sessions.

**Background sessions for utility apps**
10. Owned media

Resolve wrapped deep-link URLs

Some 3rd party services (such as email service providers):

- Wrap links in emails using their own click recording domains.
- Allow you to set your own click-recording domains.

However, if OneLink is wrapped in such domains, it may limit its functionality. To overcome this:

- Call the `setResolveDeepLinkURLs` API before SDK initialization.
- The API gets the OneLink from click domains that launch the app.

**Scenario:** Three click domains redirect to your OneLink at [https://mysubdomain.onelink.me/abCD](https://mysubdomain.onelink.me/abCD).

**Solution:**

- Use the `setResolveDeepLinkURLs` API to get the OneLink that is reached after click domains redirect an app user.
- This API method receives a list of domains that are resolved by the SDK.
- The following code lets you use your click domain but, also, preserve OneLink functionality:

  ```javascript
  Example
  AppsFlyer.setResolveDeepLinkURLs("example.com", "click.example.com");
  ```

Now, use the data from this OneLink to deep link and customize user content.

Record push notifications

Record push notifications as part of your retargeting campaigns.

**To record push notifications:**

- Call the `sendPushNotificationData` method.
- This method is inside the `onCreate` method of every activity that is launched after clicking this notification:
**User invite attribution**

If you let existing app users invite their friends and contacts to become new users, this 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. See user invite attribution.

**Cross-promotion attribution**

Cross promoting apps can be a major growth factor in driving additional installs for your apps.

With AppsFlyer, you can attribute and record installs originating from a cross-promotion campaign—promote one of your apps from within another of your apps already launched by one of your users. See cross-promotion attribution.

### 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 it to user records in your back-end systems.
- Map entries when merging data from pull and push API.

Use this API to obtain a unique AppsFlyer ID:

```csharp
string AppsFlyerUID = AppsFlyer.getAppsFlyerId();
```

**Set customer user ID**

Set your own unique customer user ID (CUID) and cross-reference it with a unique AppsFlyer ID.
Unique CUID:

- Appear in AppsFlyer raw data CSV reports.
- Can be used in postback APIs to cross-reference with internal IDs.

To set your CUID, use:

```javascript
AppsFlyer.setCustomerUserID("someId");
```

**Good practice!** Set the CUID early in the app flow—it is only associated with events reported after its setup.

Call `setCustomerUserID` before calling `startTracking`

- Recorded events will be associated with the CUID.
- Related data will appear in the raw data reports for installs and events.

```javascript
AppsFlyer.setCustomerUserID("someId");
```

**Get Customer User ID**
Check CUID for more information.

**Delay SDK init for customerUserID**
Set the customer user ID (CUID) and only then initialize the SDK. This is useful if you want install and event data to contain your CUID.

See the relevant operating system instructions:

1. **Android**
2. **iOS**

⚠️ **Warning**

Only delay setting the CUID if it suits your business logic. If you delay setting the CUID, this can increase the chance for discrepancies and may expose your app to fraud attempts.

## 12. User privacy

**Opt-out**

Different scenarios, such as legal and privacy compliance issues, may lead to a decision to opt-out and stop all SDK tracking.
Best practice! Follow the exact instructions for the scenario relevant to your app.

To stop tracking:

- Call isStopTracking and set to true.
  
  ```
  AppsFlyer.stopTracking(true);
  ```

- SDK stops functioning and no longer communicates with AppsFlyer servers.

To reactivate tracking: Call isStopTracking in and set to false.

⚠️ Warning

Only use the stopTracking API if you want to completely stop all tracking of a specific app user. This API severely impacts attribution, data collection, and the deep linking mechanism.

Anonymize user data

It is possible to anonymize specific user identifiers within AppsFlyer analytics. This complies with both the latest privacy requirements (GDPR, COPPA) and Facebook's data and privacy policies.

To anonymize an app user:

1. Set the API during SDK initialization
2. Call deviceTrackingDisabled and set to true.

   ```
   AppsFlyer.setDeviceTrackingDisabled(true);
   ```

3. App user install, events, and sessions are anonymized.

To restart tracking: Call deviceTrackingDisabled and set to false.

⚠️ Warning

Anonymizing users severely impacts your attribution information. Only use this option for regions that legally prevent you from collecting user information.

currencyCode

<table>
<thead>
<tr>
<th>Description</th>
<th>Use for events with revenue. Accepts ISO currency codes.</th>
</tr>
</thead>
</table>

https://support.appsflyer.com/hc/en-us/articles/213766183-Unity-plugin-V4-deprecated-#introduction
<table>
<thead>
<tr>
<th>Description</th>
<th>Use for events with revenue. Accepts ISO currency codes.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Method signature</strong></td>
<td><code>public void setCurrencyCode(String currencyCode);</code></td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td><code>AppsFlyer.setCurrencyCode(&quot;ZZZ&quot;);</code></td>
</tr>
</tbody>
</table>

### deviceTrackingDisabled

<table>
<thead>
<tr>
<th>Description</th>
<th>Anonymize a user’s installs, events, and sessions. Learn more about anonymizing user data.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Method signature</strong></td>
<td><code>public void setDeviceTrackingDisabled(boolean isDisabled);</code></td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td><code>AppsFlyer.setDeviceTrackingDisabled(true);</code></td>
</tr>
</tbody>
</table>

### handlePushNotification

<table>
<thead>
<tr>
<th>Description</th>
<th>Measure and get data from push notification campaigns. Learn more about recording push notifications.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Method signature</strong></td>
<td><code>handlePushNotification(Dictionary&lt;string, string&gt; payload)</code></td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td><code>AppsFlyer.handlePushNotification(payload)</code></td>
</tr>
</tbody>
</table>

### setIsDebug

| Description | Show AppsFlyer SDK logs in the Xcode console or Android Studio logs.  
- Restrict debugging to the development phase.  
- Disable debugging before distributing the app to app stores.  
- Failure to disable debugging poses major security and privacy risks. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Method signature</strong></td>
<td><code>setIsDebug(boolean debug);</code></td>
</tr>
</tbody>
</table>

https://support.appsflyer.com/hc/en-us/articles/213766183-Unity-plugin-V4-deprecated-#introduction
### Description
Show AppsFlyer SDK logs in the Xcode console or Android Studio logs.
- Restrict debugging to the development phase.
- Disable debugging before distributing the app to app stores.
- Failure to disable debugging poses major security and privacy risks.

### Example
AppsFlyer.setIsDebug(true);  

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Method signature</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>stopTracking</td>
<td>Shuts down all SDK functionality. Learn more about user privacy and opt-out.</td>
<td>stopTracking(true);</td>
<td>AppsFlyer.stopTracking(true);</td>
</tr>
<tr>
<td>onAppOpenAttribution</td>
<td>Get deep-link data when an app opens via a deep link.</td>
<td>public void onAppOpenAttribution(string validateResult)</td>
<td></td>
</tr>
<tr>
<td>onAppOpenAttributionFailure</td>
<td>Obtains errors while getting deep-link data.</td>
<td>public void onAppOpenAttributionFailure (string error)</td>
<td></td>
</tr>
</tbody>
</table>

### onConversionDataSuccess
Method used to get conversion data. Useful for deferred deep linking.
- **From SDK v5:** onConversionDataSuccess
- **Before SDK v5:** onConversionDataReceived

Best practice! Upgrade to SDK 5.0.0.
<table>
<thead>
<tr>
<th>Description</th>
<th>Method used to get conversion data. Useful for deferred deep linking.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>From SDK v5:</strong></td>
<td>onConversionDataSuccess</td>
</tr>
<tr>
<td><strong>Before SDK v5:</strong></td>
<td>onConversionDataReceived <strong>Best practice! Upgrade to SDK 5.0.0.</strong></td>
</tr>
</tbody>
</table>

**Method signature**

```java
public void didReceiveConversionData(string conversionData)
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Handles errors when failing to get conversion data from installs.</th>
</tr>
</thead>
</table>

**Method signature**

```java
public void didReceiveConversionDataWithError(string error)
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Measure uninstalls. Learn more about uninstall measurement for Unity.</th>
</tr>
</thead>
</table>

**Method signature**

```java
registerUninstall(byte[] token);
```

**Example**

```csharp
AppsFlyer.registerUninstall(<token>);
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Resolve OneLink from click domains. Learn more about resolving wrapped deep link URLs.</th>
</tr>
</thead>
</table>

**Method signature**

```java
setResolveDeepLinkURLs(params string[] urls)
```

**Example**

```csharp
AppsFlyer.setResolveDeepLinkURLs("example.com", "click.example.com");
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Add additional data to send to external partner platforms.</th>
</tr>
</thead>
</table>

https://support.appsflyer.com/hc/en-us/articles/213766183-Unity-plugin-V4-deprecated-#introduction
### Description
Add additional data to send to external partner platforms.

### Method signature
```
setAdditionalData(Dictionary<string, string> customData)
```

### Example
Learn more about setting additional custom data.

---

### setAppInviteOneLink

<table>
<thead>
<tr>
<th>Description</th>
<th>Set the OneLink template ID used to create custom attribution links for user invites.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method signature</td>
<td><code>setAppInviteOneLinkID(string oneLinkId)</code></td>
</tr>
<tr>
<td>Example</td>
<td>Learn more about setting OneLink for user invite attribution.</td>
</tr>
</tbody>
</table>

---

### setCustomerUserID

<table>
<thead>
<tr>
<th>Description</th>
<th>Set the customer user ID (CUID).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method signature</td>
<td><code>setCustomerUserId(string id)</code></td>
</tr>
<tr>
<td>Example</td>
<td>Learn more about setting the CUID</td>
</tr>
</tbody>
</table>

---

### trackRichEvent

<table>
<thead>
<tr>
<th>Description</th>
<th>Send in-app events to AppsFlyer. Learn more about recording in-app events.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method signature</td>
<td><code>trackRichEvent(string eventName, Dictionary&lt;string, string&gt; eventValues)</code></td>
</tr>
</tbody>
</table>
purchaseEvent.Add ("af_currency", "USD");
purchaseEvent.Add ("af_revenue", "0.99");
purchaseEvent.Add ("af_quantity", "1");
AppsFlyer.trackRichEvent ("af_purchase", purchaseEvent); |
**updateServerUninstallToken (for Android)**

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>Measure uninstalls. Learn more about <a href="https://support.appsflyer.com/hc/en-us/articles/213766183-Unity-plugin-V4-deprecated-#introduction">uninstall measurement for Unity</a>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Method signature</strong></td>
<td><code>updateServerUninstallToken(string token);</code></td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td><code>AppsFlyer.updateServerUninstallToken(&quot;&lt;token&gt;&quot;);</code></td>
</tr>
</tbody>
</table>