Device-mode	Cloud-mode
Web	Web
Mobile	Mobile
Server	Server

AppsFlyer is the world's leading mobile attribution & marketing analytics platform, helping app marketers around the world make better decisions. The AppsFlyer destination code is open-source. You can browse the code on GitHub for iOS and Android.

Segment's Appsflyer destination code is open source and available on GitHub. You can view these repositories:

- Android
- iOS
- Kotlin
- Swift

Getting Started

1. From the Segment web app, click **Catalog**.
2. Search for "AppsFlyer" in the Catalog, select it, and choose which of your sources to connect the destination to.
3. In the destination settings, enter your `AppsFlyer Dev Key`, which can be retrieved from the App Settings section of your AppsFlyer account.
4. Follow the instructions in the GitHub repositories: iOS SDK and Android SDK.
5. After you build and release to the app store, Segment starts translating and sending your data to AppsFlyer automatically.

Important: If you plan on using the server-side destination for an Android project, make sure to enter your **Android App ID**. If you are using only the mobile SDK, Android projects only require the **AppsFlyer Dev Key**. iOS projects always require both the **AppsFlyer Dev Key** and the **Apple App ID**. Also, note that if you do use the server-side destination, you will not be able to selectively disable calls sent to AppsFlyer using your Segment dashboard.

Additional device-mode set up for iOS 14 support

Segment updated the AppsFlyer iOS SDK to use version `6.0 beta` to prepare for tracking changes in iOS 14. The SDK beta version is compatible with the beta version of iOS 14 released by Apple, and supports both AppsFlyer's aggregate attribution, and Apple's `AppTrackingTransparency` framework, and more. See the AppsFlyer blog post about AppsFlyer's new privacy-centric attribution model.

To use the latest AppsFlyer SDK to collect IDFAs, do the following:

1. Upgrade to use Xcode12.
2. Update your Segment AppsFlyer SDK to version 6.0.2 or later.
3. Import and implement the AppTrackingTransparency (ATT) Framework.
   - Navigate to your project `Info.plist` and add a "Privacy - Tracking Usage Description". This description appears in a popup when the application initializes in iOS 14. Users are prompted to indicate whether or not they want to allow tracking.
4. Add and customize the following code in your `AppDelegate.m` file on `didFinishLaunchingWithOptions` to allow AppsFlyer collect IDFAs.

```
// The following block is for applications wishing to collect IDFA.
// for iOS 14 and later - The user will be prompted for permission to collect IDFA.
// If permission granted, the IDFA will be collected by the SDK.
// for iOS 13 and earlier - The IDFA will be collected by the SDK. The user will NOT be prompted for permission.
if #available(iOS 14, *) {
    // Set a timeout for the SDK to wait for the IDFA collection before handling app launch
    AppsFlyerLib.shared().waitForATTUserAuthorization(withTimeoutInterval: 60)
    // Show the user the Apple IDFA consent dialog (AppTrackingTransparency)
    // Can be called in any place
    ATTrackingManager.requestTrackingAuthorization { (status) in
    }
}
```

5. Follow Segment's guide for collecting IDFA

Additional iOS Cloud Mode Set up for iOS 14

With the release of Segment's latest Analytics-iOS SDK, which includes support for upcoming iOS 14 tracking changes, you must decide if you *need* to collect the user's IDFA or not. If you do not need to collect IDFA, you can update your Analytics-iOS SDK to the next version, and Segment sets `device.adTrackingEnabled` to `false`, and starts deleting the `device.advertisingId` from the context object in your payloads. If you *do* need to collect the IDFA, you must import the IDFA closure as a config to the library, or import the Ad Tracking Transparency

framework from Apple.

If you have the **Can Omit AppsFlyerID** setting enabled, but aren't sending an IDFA (either because you aren't passing one, or the user denied permission to collect it), AppsFlyer rejects the event.

To prevent this, you can enable the new **Fallback to send IDFV when advertisingId key not present** setting in your AppsFlyer destination settings. With this enabled, when you send data using cloud-mode (through the Segment servers), Segment sends the user's IDFV (the `device.id`) when `device.advertisingId` is missing or blank AND "Can Omit AppsFlyerID" is enabled.

Additional React Native device-mode set up

To add the AppsFlyer device-mode SDK to a React Native project using Segment's new `2.0` release, please reference the Destination Plugin documentation.

Server

AppsFlyer offers an **augmentative** server-side HTTP API intended for use along side the AppsFlyer mobile SDK. Use the cloud-mode destination *with* the mobile SDK to link out-of-app events (such as website or offline purchases) with attributed users and devices.

Important: The cloud-mode destination is not meant to replace the device-mode destination, and you should not use the cloud-mode destination by itself. AppsFlyer requires that you bundle the mobile SDK to correctly attribute user actions. Remember that if you pass in an `appsFlyerId` on cloud-mode calls, you cannot prevent events from sending to AppsFlyer from the Segment app.

If you want to use AppsFlyer server-side only, contact your AppsFlyer representative, as this is an Enterprise Customer Feature.

Configuring Server-side Delivery

If you'd like to attribute offline events with a certain user or device, the server-side destination may be employed.

AppsFlyer requires the following properties for this attribution:

AppsFlyer ID

Send the **AppsFlyer ID** with each event at `integrations.AppsFlyer.appsFlyerId`, see example below. This identifier is unique to each device and can be retrieved using the AppsFlyer SDK. It is a good idea to store this value in an external database where it may be easily accessible by a server or website environments.

Device Type

AppsFlyer requires the user's device type as either `'ios'` or `'android'`, passed at `context.device.type` object, see example below.

Advertising ID

AppsFlyer requires the passing of an **Advertising ID** (referred to as **IDFA** on iOS and **Advertising ID** on Android) at `context.device.advertisingId`, see example below:

```
// node.js library example
analytics.track({
  event: 'Membership Upgraded',
  userId: '97234974',
  context: {
    device: {
      type: 'ios',
      advertisingId: '159358'
    }
  },
  integrations: {
    AppsFlyer: {
      appsFlyerId: '1415211453000-6513894'
    }
  }
});
```

Check your specific server-side library docs for specifics on how to format the method properly.

> See the **Can Omit AppsFlyerId** and **Fallback to send IDFV when advertisingId key not present (Server-Side Only)** settings descriptions for details on excluding the previous fields in server-side events.

Finally, the server-side component will look for the following `properties` and handle them specially:

- `ip` (this should be the `ip` of your customer–this is not collected by Segment's libraries out-of-the-box. Pass this into the payload under context so that AppsFlyer can properly attribute it.)
- `timestamp` (refer to AppsFlyer's docs on how they process timestamps. Since the libraries generate a timestamp, Segment always sets this value)
- `currency` (defaults to `"USD"`)
- `revenue` (For `Order Completed` events, precedence is given to `total`, falling back to `properties.revenue`)

All other `properties` will be sent to AppsFlyer as custom properties inside `eventValue`.

> Be sure to calibrate/update the time window in AppsFlyer's dashboard to see your events!

Send in-app events to Appsflyer v3 Endpoint

When transmitting data serverside to Appsflyer, you have the option to enhance security by enabling the transmission of in-app events to Appsflyer's v3 endpoint, which authenticates requests using a more secure S2S token.

To activate this feature, simply input your S2S token in the destination settings and toggle the "Use API v3" switch to the enabled position.

Send User Consent Preferences Server-side

To transmit user consent data server-side, incorporate the consent preferences into the `integrations.AppsFlyer.consent_data` object. This can be done in either TCF or manual format, as outlined in the AppsFlyer Send Event documentation.

```
// node.js library example with tcf
analytics.track({
  event: 'Membership Upgraded',
  userId: '97234974',
  context: {
    device: {
      type: 'ios',
      advertisingId: '159358'
    }
  },
  integrations: {
    AppsFlyer: {
      appsFlyerId: '1415211453000-6513894'
    },
    consent_data: {
      tcf: {
        tcstring: "string",
        cmp_sdk_version: 1,
        cmp_sdk_id: 1,
        gdpr_applies: 0,
        policy_version: 1
    }
  }
  }
});

// node.js library example with manual consent
analytics.track({
  event: 'Membership Upgraded',
  userId: '97234974',
  context: {
    device: {
      type: 'ios',
      advertisingId: '159358'
    }
  },
  integrations: {
    AppsFlyer: {
      appsFlyerId: '1415211453000-6513894'
    },
    consent_data: {
      manual: {
        ad_personalization_enabled: 'true',
        ad_user_data_enabled: 'true',
        gdpr_applies: 'true'
      }
  }
  }
});
```

Identify

If you're not familiar with the Segment Spec, take a look to understand what the Identify method does. An example iOS call would look like:

```
[[SEGAnalytics sharedAnalytics] identify:@"12091906-01011992"
                             traits:@{ @"email": @"john.doe@example.com" }];
```

When you call Identify, Segment uses AppsFlyer's `setCustomerUserID` to send the `userId` that was passed in.

Note: Identify calls are not supported using AppsFlyer's HTTP API at the moment. You can only send `.identify` calls if you have the AppsFlyer SDK bundled.

Track

If you're not familiar with the Segment Spec, take a look to understand what the Track method does. An example iOS call would look like:

```
[[SEGAnalytics sharedAnalytics] track:@"Article Completed"
                         properties:@{ @"title": @"How to Create a Tracking Plan", @"course": @"Intro to Analytics"
 }];
```

When you call Track, Segment translates it automatically and sends the event to AppsFlyer.

Segment includes all the event properties as callback parameters on the AppsFlyer event, and automatically translates `properties.revenue` to the appropriate AppsFlyer purchase event properties based on the spec'd properties.

Segment uses AppsFlyer's `transactionId` deduplication when you send an `orderId` (see Segment's e-commerce spec for more details).

Install Attributed

Client

Segment will automatically trigger an `Install Attributed` event if you have **trackAttributionData** enabled in your settings, and the Segment-AppsFlyer integration installed in your app. The event payload will adhere to the `Install Attributed` event specification explained in Segment's Mobile Spec and will propagate to your other downstream destinations.

Server

If you track events server-side, AppsFlyer can still send attribution postbacks, but you need to configure this functionality in your AppsFlyer account. To enable this:

1. Navigate to your AppsFlyer app and on the sidebar of the main screen select **Configuration > Partner Marketplace** and search for Segment.
2. Enter the configuration options and input your Segment Write Key. Once enabled, successfully attributed app installs begin showing up as `Install Attributed` events similar to the client side behavior documented above.

If you are sending in the attribution data yourself, for iOS be sure the following properties are sent in within the campaign object on the `Install Attributed` or `Application Opened` event so Appsflyer can correctly attribute it as an Apple Search Ad event. These values should be returned by the Apple Search Ads API:

```
"campaign": {
        "content": "keyword1keyword2",
        "ad_creative": "OrgName",
        "conversion_date": "2018-03-07T04:05:50Z",
        "ad_group": "US-iOS-campaign-Exact",
        "id": "123",
        "ad_group_id": "456",
        "name": "US-iOS-campaign",
        "click_date": "2018-03-06T04:05:50Z",
        "lineitem_id":"789",
        "attribution":"true",
        "lineitem_name":"US-iOS-campaign-Name"
      }
```

For example, an attribution event coming from an attribution partner would look like:

```
[[SEGAnalytics sharedAnalytics] track:@"Install Attributed", properties: @{
    @"provider" : @"Appsflyer/Tune/Kochava/Branch",
    @"campaign" : @{
        @"source" : @"Network/FB/AdWords/MoPub/Source",
        @"name" : @"Campaign Name",
        @"content" : @"Organic Content Title",
        @"ad_creative" : @"Red Hello World Ad",
        @"ad_group" : @"Red Ones",
        @"conversion_date": @"2018-03-07T04:05:50Z",
        @"id": @"123",
        @"ad_group_id": @"456",
        @"click_date": @"2018-03-06T04:05:50Z",
        @"lineitem_id":@"789",
        @"attribution":@"true",
        @"lineitem_name":@"US-iOS-campaign-Name"
    }
}];
```

Attribution and install counts might differ between Segment and attribution providers like AppsFlyer

For more information about the factors that contribute to these differences, see the Segment's Role in Attribution documentation.

Other Features

Revenue Tracking

The destination automatically recognizes spec'd `revenue` property and translates them to AppsFlyer's revenue tracking method.

Transaction De-duplication

The destination automatically recognizes the spec'd `orderId` property, and sends it as the Transaction ID to AppsFlyer for revenue de-duplication.

In-App Purchase Receipts

The destination does not currently support in-app purchase receipts. If this is important to you, email support@appsflyer.com.

Deeplinking

The destination does not automatically support out-of-the-box deeplinking (you need to write code here regardless!).

Therefore, you can use AppsFlyer's OneLink integration which is a single, smart, tracking link that can be used to track on both Android and iOS. OneLink tracking links can launch your app when it is already installed instead of redirecting the user to the app store.

For more details, review the AppsFlyer OneLink set up Guide. More information is available in the AppsFlyer SDK Integration Guides (iOS, Android) and Segment's mobile FAQs (iOS, Android).

FAQ

Can I send my AppsFlyer attribution data to destinations like GA4 and Salesforce?

Yes, you can use Source Functions to send attribution data to destinations. Source Functions let you create a custom source that ingests AppsFlyer data through a Webhook and transforms it into Track, Identify, Page, or other event calls. These events can then be sent to your connected destinations.

Settings

Segment lets you change these destination settings from the Segment app without having to touch any code.

SETTING	DESCRIPTION
Android App ID	`string`. Your Android App's ID. Find this in your AppsFlyer's 'My App' dashboard. It should look something like 'com.appsflyer.myapp'. This is required for Android projects if you want to send events using the server side integration.
Apple App ID (iOS)	`string`. Your App's ID, which is accessible from iTunes or in AppsFlyer's 'My App' dashboard. This is optional for Android projects, and only required for iOS projects.
AppsFlyer Dev Key *(required)*	`string`. Your unique developer ID from AppsFlyer, which is accessible from your AppsFlyer account.

SETTING	DESCRIPTION
AppsFlyer S2S Token	`string`. Your unique S2S token from AppsFlyer, accessible in your AppsFlyer account. Required when "Use API V3" is set to "true."
Can Omit AppsFlyerId	`boolean`, defaults to `FALSE`. *Only applicable for Appsflyer's Business Tiers customers using server-side or cloud mode destination.* Please contact your AppsFlyer representative for more information. This setting allows to use the advertising ID as appsflyer ID.
Fallback to send IDFV when advertisingId key not present (Server-Side Only)	`boolean`, defaults to `FALSE`. With the update to use analytics-ios v4.x SDK if adTrackingEnabled is set to false, the advertisingId key will be deleted from the event. If you have the setting enabled "Can Omit AppsFlyerId", these events will fail when sent to AppsFlyer API. To prevent these event failures in this scenario enable this send the IDFV instead. When the "Can Omit AppsFlyerId" setting is enabled if the IDFA is zeroed out, we will also send an IDFV when this setting is enabled.
Enable HTTP fallback (Android)	`boolean`, defaults to `FALSE`. If selected, HTTPS calls will fallback on HTTP
Roku App ID	`string`. **IMPORTANT**: In order to send Roku data, you **must** contact your AppsFlyer representative as this type of data stream requires a full server to server integration which is available but is gated as a AppsFlyer Enterprise Customer feature. Without AppsFlyer's consent we are unable to forward your Roku data. Your Roku App's ID. Find this in your AppsFlyer's 'My App' dashboard. This is required for Roku projects if you want to send events using the server side integration.
Track Attribution Data	`boolean`, defaults to `FALSE`. Send attribution data to Segment and other tools as a track call (mobile libraries only).
Use API v3	`boolean`, defaults to `FALSE`. Enable to post in-app events to AppsFlyer V3 endpoint. Do not enable if you have not provided a value for the "AppsFlyer S2S Token" setting.

This page was last modified: 23 Jan 2025

Need support?

Questions? Problems? Need more info? Contact Segment Support for assistance!

Visit our Support page

Help improve these docs!

Edit this page

Request docs change

Was this page helpful?

Yes

No

Get started with Segment

Segment is the easiest way to integrate your websites & mobile apps data to over 300 analytics and growth tools.

Your work e-mail

Request Demo

or

Create free account

Privacy

Terms

Website Data Collection Preferences