



Getting Started

- What is Segment?
- [How Segment Works](#)
- Getting Started Guide
- A Basic Segment Installation
- Planning a Full Installation
- A Full Segment Installation
- Sending Data to Destinations
- Testing and Debugging
- What's Next
- Use Cases

Guides

Connections

Unify

Engage

Privacy

Protocols

Segment App

API

Partners

Glossary

Config API

Help

sending data from any of the Segment [libraries](#) to an Adobe report suite. When you send events from Segment's mobile SDKs or Cloud-mode libraries, Segment translates that data using a mapping that you configure, and then passes it to the Adobe Analytics [Data Insertion API](#). For more information, you can browse the code on GitHub in the [@segment-integrations/analytics-kotlin-adobe-analytics](#) repository.

Planning for Adobe Analytics

Adobe Analytics uses a slightly different approach to tracking than Segment, and it's important to understand the difference so you can effectively set up your integration. Segment uses a user-action data model, which uses different types of calls to track different activities of a user on a website or app. Adobe Analytics uses page views as the basic unit of activity, and variables like custom traffic variables (also called 'props'), eVars, list variables, and hierarchy variables to add details for more nuanced analysis.

For example, if one of your end users dismissed a welcome dialog in your app, Segment would generate a `Welcome Dialog Dismissed` event with properties that contain the user ID (`user123`) and the dialog name (`welcome-dialog`), while Adobe Analytics would model the same action as a `pageView` with variables that represent the dialog name, `visitorID`, and the event name, and an eVar ("dismissed").

Both Segment and Adobe Analytics have recommended standard data for tracking events. Segment has [the Spec](#), and Adobe uses predefined events. Segment automatically maps incoming event data and some product level properties to Adobe's predefined events, when the event data is in the correct Segment Ecommerce Spec (</docs/connections/spec/ecommerce/v2/>) format. Video calls using the format described in this document are also automatically mapped. If you're using the Mobile SDKs, mobile lifecycle events are also automatically mapped. If you need to create Page and Track events that are outside the scope of the Ecommerce Spec, you need to map those in your Segment destinations settings UI.

Segment strongly recommends that you create a Tracking Plan for both your Segment and Adobe Analytics events before you send any events or properties to Adobe. This helps you map your Segment events to Adobe events and Segment properties to Adobe variables. If you decide to set up Adobe Analytics for mobile, you must set up this mapping in both the Segment settings and the Adobe Mobile Services dashboard, so it's good to stay consistent.

Setting Up the Adobe Analytics SDK

Before you start sending data from your Kotlin application to Adobe Analytics, complete the following setup steps:

1. Enable the Segment-Adobe Analytics destination in your Segment workspace.
2. From your Adobe Mobile Services dashboard, check and customize the settings on the "Manage App Settings" tab.
3. Download these settings as the `ADBMobileConfig.json` file by clicking the **Config JSON** link at the bottom of the same tab. Follow the instructions in Adobe's [Core implementation and lifestyle](#) documentation.
4. Follow the instructions below for each mobile environment to add the Adobe Analytics dependency to your project.



Tip: Mobile implementations use the `ADBMobileConfig.json` file to store the settings that you would otherwise enter in the Adobe Analytics destination settings in the Segment app. You can change these settings from the Manage App Settings tab in your Adobe Mobile Services dashboard, and can download the file from that same tab. This file includes the Report Suite ID, Timestamp Option, Tracking Server Secure URL, Tracking Server URL, and Use Secure URL for Server-side settings.

Adding the dependency

To install the Segment-Adobe Analytics integration, add this line to your gradle file:

```
implementation 'com.segment.analytics.kotlin.destinations:adobe-analytics:<latest_version>'
```

Or the following for Kotlin DSL

```
implementation("com.segment.analytics.kotlin.destinations:adobe-analytics:<latest_version>")
```

Using the Plugin in your App

Open the file where you set up and configured the Analytics-Kotlin library. Add this plugin to the list of imports.

```
import com.segment.analytics.kotlin.destinations.adobeanalytics.AdobeAnalyticsDestination
```

Just under your Analytics-Kotlin library setup, call `analytics.add(plugin = ...)` to add an instance of the plugin to the Analytics timeline.

```
analytics = Analytics("<YOUR WRITE KEY>", applicationContext) {
    this.flushAt = 3
    this.trackApplicationLifecycleEvents = true
}
analytics.add(plugin = AdobeAnalyticsDestination(adobeAppID = "<WRITE YOUR ENVIRONMENT-FILE-ID>"))
```

Your events will now begin to flow to Adobe Analytics in device-mode.

Sending data to Adobe Analytics

Segment strongly recommends that you create a tracking plan for both your Segment and Adobe Analytics events *before* you send any events or properties to Adobe. This helps you map your Segment events to Adobe events and Segment properties to Adobe eVars or props, since you'll have to do this in both the Segment settings UI and your Adobe Mobile Services dashboard.

Sending Events

You can map Segment events in your **Events v2** settings to any event variable you already defined in your Adobe Analytics Mobile Services dashboard.



Note: Do not use the deprecated **Events** settings. These no longer forward events to Adobe.

Here's an example of how you might map Segment events to Adobe Analytics events connected in device mode:

The screenshot shows the 'Settings' tab in the Segment mobile app. Under 'Event Delivery', the 'EventsV2 Mobile' tab is selected. It displays a table for mapping Segment Events to Adobe Analytics Events. The first row shows 'Clicked a Button' mapped to 'myapp.adobe.Clicked'. There is an 'Add Row' button and a 'Save' button at the bottom.

Here's an example of how you would implement the same mapping in Adobe's Mobile Services Dashboard:

The screenshot shows the 'Custom Metrics' section in the Adobe Mobile Services Dashboard. It includes a table with the following data:

Metric	Name	Context Data	Type
event1	Clicked A Button	myapp.adobe.Clicked	Whole Number
event2	Name	Context Data	Whole Number

Sending Custom Properties

You can use the Context Data Variables settings to map Segment properties to any context data variable defined in your Adobe Analytics Mobile Services dashboard. This includes both Adobe props and eVars. You can see a list of the Adobe variable types in your Adobe Mobile Services dashboard.

Settings

Filters

Event Tester

Event Delivery

General

Other

Mappings

Identity Resolution

Timestamps

Context Data Variables

List Variables

eVars

Other

Events

EventsV2 Mobile

Hierarchy Variables

Merchandising

Props

Docs

SEGMENT PROPERTY

CONTEXT DATA VARIABLE

color

▶

myapp.adobe.color

×

⊕ Add Row

Context Data Property Prefix

If you would like to prefix your Segment properties before sending them as contextData, enter a prefix here.

Save

Here’s an example of how you would implement the same mapping in Adobe’s Mobile Services Dashboard:

Standard Variables & Metrics

Custom Variables

Custom Metrics

Custom Properties

Properties (or props) answer the question "which one?" Props can be set to a text value that will be associated with other variables and metrics that are sent in the same hit. The values can be used to filter reports, or can be listed in rank order by an associated metric.

When a value is set for a property in a tracking call (or hit), it applies only to that call.

Variable	Name	Context Data	List Support	
prop1	Color	myapp.adobe.color	<input type="checkbox"/> List	

SEGMENT PAYLOAD FIELD	IOS MAPPING NOTATION	ANDROID MAPPING NOTATION
anonymousId	anonymousId	.anonymousId
messageId	messageId	.messageId
event Track calls only	event	.event
name Screen calls only	name	.name
context.traits.key	traits.key	.context.traits.key
context.key	key	.context.key

SEGMENT PAYLOAD FIELD	IOS MAPPING NOTATION	ANDROID MAPPING NOTATION
context.arrayKey.key for example: context.device.id	arrayKey.key for example: device.id	.context.arrayKey.key
properties.key	key	.key

Adobe Lifecycle events

Segment implements Adobe Lifecycle Events automatically – you don't have to enable any additional settings! Lifecycle events gather important information such as app launches, crashes, session length, and more. See the [list of all Adobe lifecycle metrics and dimensions](#) to learn more.

Identify

When you make an Identify call, Segment sets the Adobe `visitorId` to the value of the user's Segment `userId`. The snippets below show what Segment does with this information.

```
Config.setUserIdentifier("123");
```

Screen

When you call Screen, Segment sends an Adobe `trackState` event, and passes the screen name and any properties you mapped to Adobe as context data values.

For example:

```
Analytics.trackState("Home Screen", <properties mapped in contextData>);
```

Track

When you call Track, Segment sends an Adobe `trackAction` event, and passes your event name and any properties you mapped to Adobe as context data values.

Fore example:

```
Analytics.trackEvent("Clicked A Button", <properties mapped in contextData>);
```

Reset

For exmple:

```
Config.setUserIdentifier(null);
```

Flush

```
Analytics.sendQueuedHits();
```

This page was last modified: 12 Aug 2024

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

© 2025 Segment.io, Inc.

Privacy

Terms

Website Data Collection Preferences

