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

Refer to it as **CleverTap** in the Integrations object

Components

Browser iOS Android

Connection Modes

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

Getting Started

Once the Segment library is integrated, toggle CleverTap on in your Segment destinations, and add your CleverTap Account ID and CleverTap Account Token which you can find in the CleverTap Dashboard under

Settings.

CleverTap supports the Identify, Track, Page (server-side only), and Screen (iOS and server-side only) methods.

You can integrate CleverTap using a server-side or mobile destination (iOS or Android). If you are interested in using CleverTap's push notifications or in-app notifications products, you should use the mobile destinations.

For server-side destination requests, CleverTap requires both the Segment `anonymousId` and `userId` in the payload.

CleverTap maintains the server-side and mobile integrations:

- Android
- iOS

For any issues with the server-side and mobile integrations, contact the CleverTap Support team.

Identify

When you identify a user, Segment passes that user's information to CleverTap with userId as CleverTap's Identity value. A number of Segment's special traits map to CleverTap's standard user profile fields. You'll pass the key on the left into Segment and Segment transforms it to the key on the right before sending to CleverTap.

- `name` maps to `Name`
- `birthday` maps to `DOB`
- `avatar` maps to `Photo`
- `gender` maps to `Gender`
- `phone` maps to `Phone`
- `email` maps to `Email`

All other traits will be sent to CleverTap as custom attributes. The default logic will lower case and snake_case any user traits - custom or special - passed to CleverTap.

> In cloud mode, CleverTap uses Segment anonymous ID as the CleverTap ID. In device mode, CleverTap ignores the anonymous ID and CleverTap injects its own ID.

Alias

> Alias is supported by Device-mode Web connections

When you send an Alias call to CleverTap, CleverTap updates the user's profile with the contents of the Alias call.

Track

When you `track` an event, Segment sends that event to CleverTap as a custom event. Note that CleverTap does not support arrays or nested objects for custom track event properties.

> CleverTap requires `identify` traits such as `userId` or `email` to record and associate the Track event. Without these traits, the Track event does not appear in CleverTap.

The default logic for the cloud mode connection to CleverTap will lower case and snake_case any event properties passed from Segment's servers to CleverTap. The device mode connection will not lower case or

snake_case any event properties passed directly to CleverTap from the client.

Order Completed

When you `track` an event using the server-side destination with the name `Order Completed` using the e-commerce tracking API, Segment maps that event to CleverTap's Charged event.

Page

When you send a `page` event using the server-side destination, Segment sends that event to CleverTap as a Web Page Viewed event.

Screen

When you send a `screen` event using the server-side destination or the iOS bundled SDK, Segment sends that event to CleverTap as an App Screen Viewed event.

Android

Integrating

1. Add the CleverTap Segment Destination dependency to your app build.gradle:

```
compile 'com.clevertap.android:clevertap-segment-android:+'
```

Note: The group Id is `com.clevertap.android`, not `com.segment.analytics.android.integrations`.

2. Next, declare CleverTap's destination in your Analytics instance:

```
Analytics analytics = new Analytics.Builder(context, "YOUR_WRITE_KEY_HERE")
  .use(CleverTapIntegration.FACTORY)
  ...
  .build();
```

Integrating Push

1. In your AndroidManifest.xml, register the following CleverTap services.

```
<service
    android:name="com.clevertap.android.sdk.FcmTokenListenerService">
    <intent-filter>
        <action android:name="com.google.firebase.INSTANCE_ID_EVENT"/>
    </intent-filter>
</service>

<service
    android:name="com.clevertap.android.sdk.FcmMessageListenerService">
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT"/>
    </intent-filter>
</service>
```

2. For more in-depth information, visit CleverTap's Android push integration documentation.

In-App Notifications

1. In your AndroidManifest.xml, add the CleverTap InAppNotificationActivity.

```
<activity
        android:name="com.clevertap.android.sdk.InAppNotificationActivity"
        android:configChanges="orientation|keyboardHidden"
        android:theme="@android:style/Theme.Translucent.NoTitleBar" />
```

No further action is required to integrate in-app notifications, which are registered for and requested by default by the CleverTap Segment integration.

Sample App

CleverTap has created a sample Android application that integrates CleverTap using Segment. Check it out at the GitHub repository.

iOS

Integrating

1. Add the CleverTap Segment Pod to your Podfile:

```
`pod 'Segment-CleverTap'`
```

Use the latest version on CocoaPods since it will contain the most up to date features and bug fixes.

2. Next, declare CleverTap's integration in your app delegate instance:

```
SEGAnalyticsConfiguration *config = [SEGAnalyticsConfiguration configurationWithWriteKey:@"YOUR_WRITE_KEY_HERE"];
[config use:[SEGCleverTapIntegrationFactory instance]];
[SEGAnalytics setupWithConfiguration:config];
```

Integrating Push

1. Follow the directions to register for push at: /docs/connections/sources/catalog/libraries/mobile/ios/#how-do-i-use-push-notifications.
2. In your application's application:didReceiveRemoteNotification: method, add the following:
`[[SEGAnalytics sharedAnalytics] receivedRemoteNotification:userInfo];`
3. If you integrated the application:didReceiveRemoteNotification:fetchCompletionHandler: in your app, add the following to that method:
`[[SEGAnalytics sharedAnalytics] receivedRemoteNotification:userInfo];`
4. If you implemented handleActionWithIdentifier:forRemoteNotification:, add the following to that method:
`[[SEGAnalytics sharedAnalytics] handleActionWithIdentifier:identifier forRemoteNotification:userInfo];`

In-App Notifications

No further action is required to integrate in-app notifications, which are registered for and requested by default by the CleverTap Segment integration.

Sample App

CleverTap has created a sample iOS application that integrates CleverTap using Segment. Check it out at the GitHub repository.

Server-Side

Push Tokens

If you chose not to bundle the CleverTap Mobile SDK, then you will have to implement your own Push Message processors (and you won't have access to CleverTap's In-App feature).

If you decide to implement your own Push Message processors, then you can pass push tokens to CleverTap using the server-side destination. You can do this by sending it inside context.device.token.

Troubleshooting

Verbose Logging

When using Web Device-mode, you can enable verbose logging of all communication with CleverTap servers by

setting the `theWZRK_D` variable in `sessionStorage`. In the developer console of your browser, enter `sessionStorage['WZRK_D'] = '';`, you'll see error messages and warnings logged. See the CleverTap Web Quickstart Guide for more details.

Engage

You can send computed traits and audiences generated using Engage to this destination as a **user property**. To learn more about Engage, schedule a demo.

For user-property destinations, an identify call is sent to the destination for each user being added and removed. The property name is the snake_cased version of the audience name, with a true/false value to indicate membership. For example, when a user first completes an order in the last 30 days, Engage sends an Identify call with the property `order_completed_last_30days: true`. When the user no longer satisfies this condition (for example, it's been more than 30 days since their last order), Engage sets that value to `false`.

When you first create an audience, Engage sends an Identify call for every user in that audience. Later audience syncs only send updates for users whose membership has changed since the last sync.

Real-time to batch destination sync frequency

Real-time audience syncs to CleverTap may take six or more hours for the initial sync to complete. Upon completion, a sync frequency of two to three hours is expected.

Settings

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

SETTING	DESCRIPTION
Account ID *(required)*	`string`. Add your CleverTap Account ID which you can find in the CleverTap Dashboard under Settings.
Account Token	`string`. **Mobile Only:**Add your CleverTap Account Token which you can find in the CleverTap Dashboard under Settings.
Region	`select`. **Server Only**: Your dedicated Clevertap region.

This page was last modified: 17 Oct 2023

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