Mobile	Mobile
Server	Server

Castle monitors every step of the customer journey to help visualize and proactively block fraud that would otherwise fly under the radar. Types of fraud or abuse that can be managed include bots, fake accounts, multi-accounting, and account sharing.

The Castle destination source code is available on GitHub. Source code for the following integrations are available:

- iOS
- Android
- Web

Castle maintains this destination. For any issues with the destination, contact the Castle support team.

Getting Started

1. Navigate to **Connections > Catalog** in the Segment web app.
2. Search for *Castle* in the **Destinations** tab of the catalog, and select it, and click **Configure Castle**.
3. Choose the sources you want to connect the destination to.
4. Enter the "Publishable Key" the Publishable Key field. Find the Publishable Key on the Castle dashboard. Calls are now visible in Castle dashboards in real-time.

> Castle ingests Segment Client-side events. Server-side events are dropped and not processed. Castle supports web, Android and iOS integrations through Segment.

iOS

1. Add the Castle Segment dependency

   With Xcode:
   In the Xcode **File** menu, click **Add Packages**. In the resulting dialog where you can search for Swift packages, enter the following repository URL: `https://github.com/castle/analytics-ios-integration-castle`
   You can optionally pin the package to a specific branch or version, and select the project in your workspace to which you'll add the package. When you're done, click **Add Package**.

   With Package.swift:

```
.package(
    name: "SegmentCastle",
    url: "https://github.com/castle/analytics-ios-integration-castle",
    from: "1.0.0"
),
```

2. Next, add the Castle destination to your analytics instance:

```
let analytics = Analytics(configuration: Configuration(writeKey: "<YOUR_WRITE_KEY_HERE>"))

let castleDestination = CastleDestination(userJwt: "<USER_JWT>")
analytics.add(plugin: castleDestination)
```

Android

1. You can add the Castle Segment dependency two ways:

   Add this line to your gradle file:

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

   Add the following for Kotlin DSL:

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

2. Next, add the Castle destination to your analytics instance:

```
analytics = Analytics("<YOUR WRITE KEY>", applicationContext)
analytics.add(plugin = CastleDestination(userJwt = "<USER_JWT>"))
```

Page

If you're not familiar with the Segment Specs, take a look to understand what the Page call does. An example call looks like:

```
analytics.page()
```

Segment sends Page calls to Castle as `$page` events.

Track

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

```
analytics.track('Added to Cart')
```

Segment sends Track calls to Castle as a `$custom` events.

Secure Mode

Send user information as a signed JWT when you use Castle in production. This prevents bad actors from spoofing any user information.

In your backend code, encode the user as a JWT and sign it using your Castle "API Secret". When Castle receives the JWT, the integrity of the user data is verified to ensure that the data wasn't tampered with.

Below is an example of how to generate a JWT on your backend using the Ruby language:

```
jwt_from_backend = JWT.encode({
  id: '97980cfea0067',
  email: 'peter@example.com'
}, ENV.fetch('CASTLE_API_SECRET'), 'HS256')
```

Transfer the `user_jwt` object to your frontend through a separate API call, or by injecting the code using a templating language:

```
var userJwt = "<%= jwt_from_backend %>";

// Then use the `userJwt` argument instead of `user` when using any of the tracking methods
Castle.page({userJwt: userJwt});

analytics.identify('97980cfea0067', {
  email: 'peter@example.com',
}, {
  Castle: {
    userJwt: userJwt
  }
});
```

When Castle receives a JWT version of the user object, its contents override the user object sent the standard Segment way.

Settings

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

SETTING	DESCRIPTION
Automatic Page tracking	`boolean`, defaults to `FALSE`. When you enable automatic page tracking, Castle will track a page view whenever the url of the site changes as opposed to mapping explicitly to your implementation of Segment `.page()` calls.
Custom Cookie Domain	`string`. If your authenticated area is located at a different domain, use the cookie domain setting to change on which url the cookie is set.

SETTING	DESCRIPTION
API Publishable Key *(required)*	`string`. You can find your publishable key under **Settings** in the Castle dashboard. It should look something like this: `pk_KmoUyttyEiHCdFTWSqhAF1SL1z9Fi1yg`. This is required and will be used to initialize Castle's library on your device as well as when you make mobile/server calls through our server side integration.

This page was last modified: 26 Jun 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