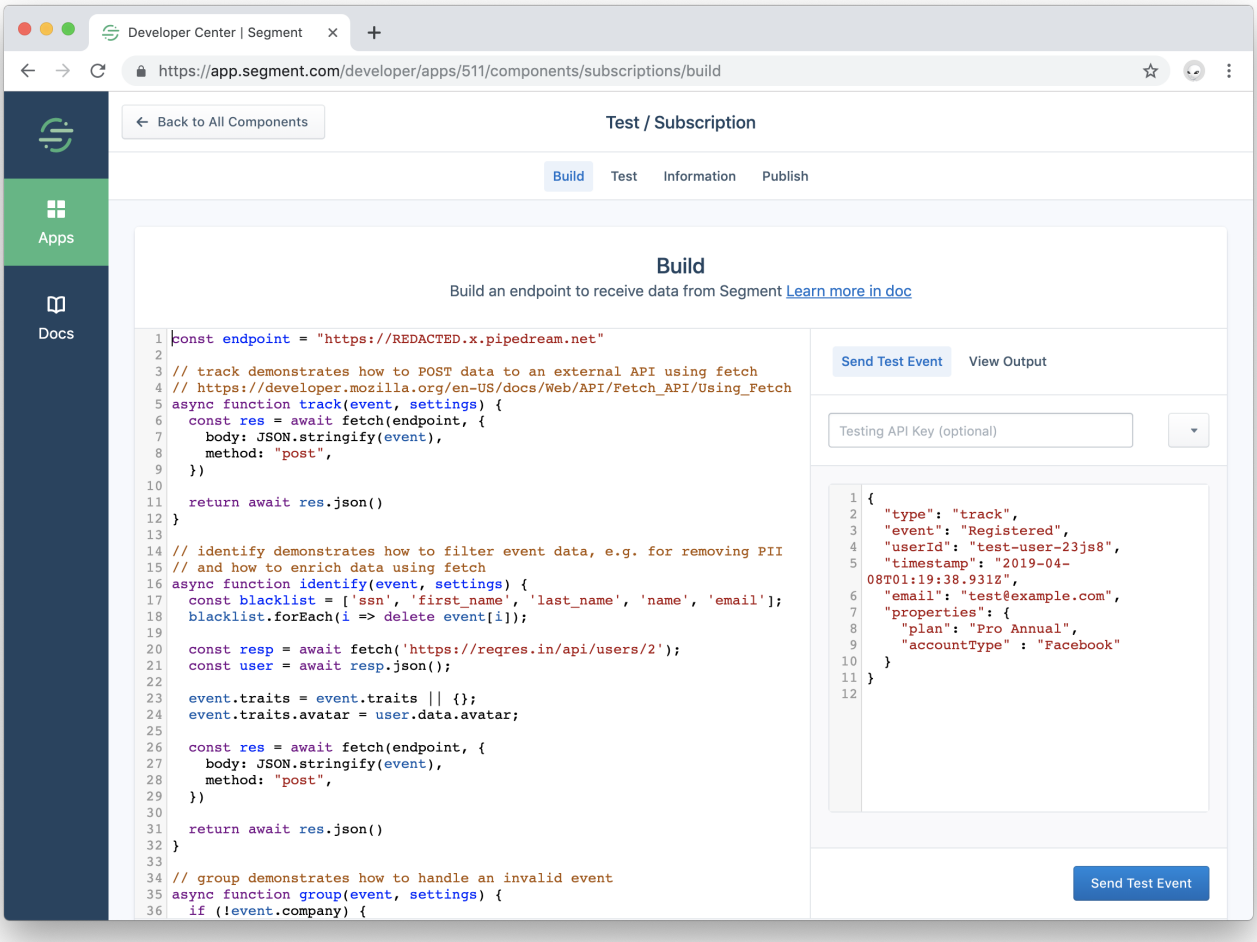
1. Understand Segment's Conceptual Model and Spec.
2. Follow Segment's security guidance.
3. Request access to the Segment Developer Center.
4. Create an App.
5. Build and test your Component(s).
6. Publish documentation.
7. Submit your App for review.
8. Launch into *Public Beta*!

Build

Begin by selecting the *Subscription* card in your Developer Center UI after creating an App and selecting *I want Segment to run functions I write*. Next, you will see the code editor where you can take full control of your Subscriptions's logic. Segment provides boilerplate functions that make it simple to send data to your API

Endpoint. You can delete the example code and implement your own functions.



For every event you send to Segment, Segment invokes a function you provide for the event type. So you must define functions named after every type in the Segment Spec that you support:

- `onIdentify`
- `onTrack`
- `onPage`
- `onScreen`
- `onGroup`
- `onAlias`
- `onDelete`

The two items passed into the functions are the *event payload* and the *settings*. All subscriptions have an *apiKey* setting by default. To add more custom settings, go to the `Settings Builder` page under `App Info`. Use your custom setting *key* (which is generated for you from your custom setting label) to access your custom setting from the *settings* argument.

- The *Event* argument to the function is the Segment Event Data
- The *Settings* argument to the function contains user settings like *apiKey* and any custom settings you have added.

The functions are "async/await" style JavaScript, and should use the Fetch API using the pre-loaded `fetch` package.

Here's a basic example of a function that POSTs the event to a "request bin" for introspection. You can go to

RequestBin to create your own `endpoint` to experiment with. This builds a query string for the URL, sets a basic auth header, and sends a JSON body:

```
const endpoint = "https://REDACTED.x.pipedream.net"

async function onTrack(event, settings) {
  const url = new URL(endpoint);
  url.searchParams.set("ts", event.timestamp);

  const res = await fetch(url.toString(), {
    body: JSON.stringify(event),
    headers: new Headers({
      "Authentication": 'Basic ' + btoa(`${settings.apiKey}:`),
      "Content-Type": "application/json",
    }),
    method: "post",
  })

  return await res.text() // or res.json() for JSON APIs
}
```

The function should return data to indicates a success. In the above example Segment returns the request body.

You can also `throw` an error to indicate a failure.

In the above example, try changing the endpoint to `https://foo` and you'll see it throws a `FetchError` with the message `request to https://foo/ failed, reason: getaddrinfo ENOTFOUND foo foo:443`

There are three pre-defined error types that you can `throw` to indicate the function ran as expected, but data could not be delivered:

- EventNotSupported
- InvalidEventPayload
- ValidationError

Here are basic examples using these error types:

```
async function onGroup(event, settings) {
  if (!event.company) {
    throw new InvalidEventPayload("company is required")
  }
}

async function onPage(event, settings) {
  if (!settings.accountId) {
    throw new ValidationError("Account ID is required")
  }
}

async function onAlias(event, settings) {
  throw new EventNotSupported("alias is not supported")
}
```

If you do not supply a function for an event type, Segment will throw an implicit `EventNotSupported` error.

Built-in Dependencies

lodash

A modern JavaScript utility library delivering modularity, performance & extras.

See Docs

AWS

The official Amazon Web Services SDK.

See Docs

Crypto

The crypto module provides cryptographic functionality that includes a set of wrappers for OpenSSL's hash, HMAC, cipher, decipher, sign, and verify functions.

See Docs

Fetch API

The Fetch API provides a JavaScript interface for accessing and manipulating parts of the HTTP pipeline, such as requests and responses. It also provides a global `fetch()` method that provides an easy, logical way to fetch resources asynchronously across the network.

See Docs

`fetch()`

The `fetch()` method starts the process of fetching a resource from the network, returning a promise which is fulfilled once the response is available.

See Docs

`Request`

The `Request` interface of the Fetch API represents a resource request.

See Docs

`Response`

The `Response` interface of the Fetch API represents the response to a request.

See Docs

`Headers`

The `Headers` interface of the Fetch API allows you to perform various actions on HTTP request and response headers. These actions include retrieving, setting, adding to, and removing. A Headers object has an associated header list, which is initially empty and consists of zero or more name and value pairs.

See Docs

`URL`

The `URL` interface is used to parse, construct, normalize, and encode URLs. It works by providing properties which allow you to easily read and modify the components of a URL.

See Docs

`URLSearchParams`

The `URLSearchParams` interface defines utility methods to work with the query string of a URL.

See Docs

`atob()`

The `atob()` function decodes a string of data which has been encoded using base-64 encoding.

See Docs

`btoa()`

The `btoa()` method creates a base-64 encoded ASCII string from a binary string.

See Docs

Test

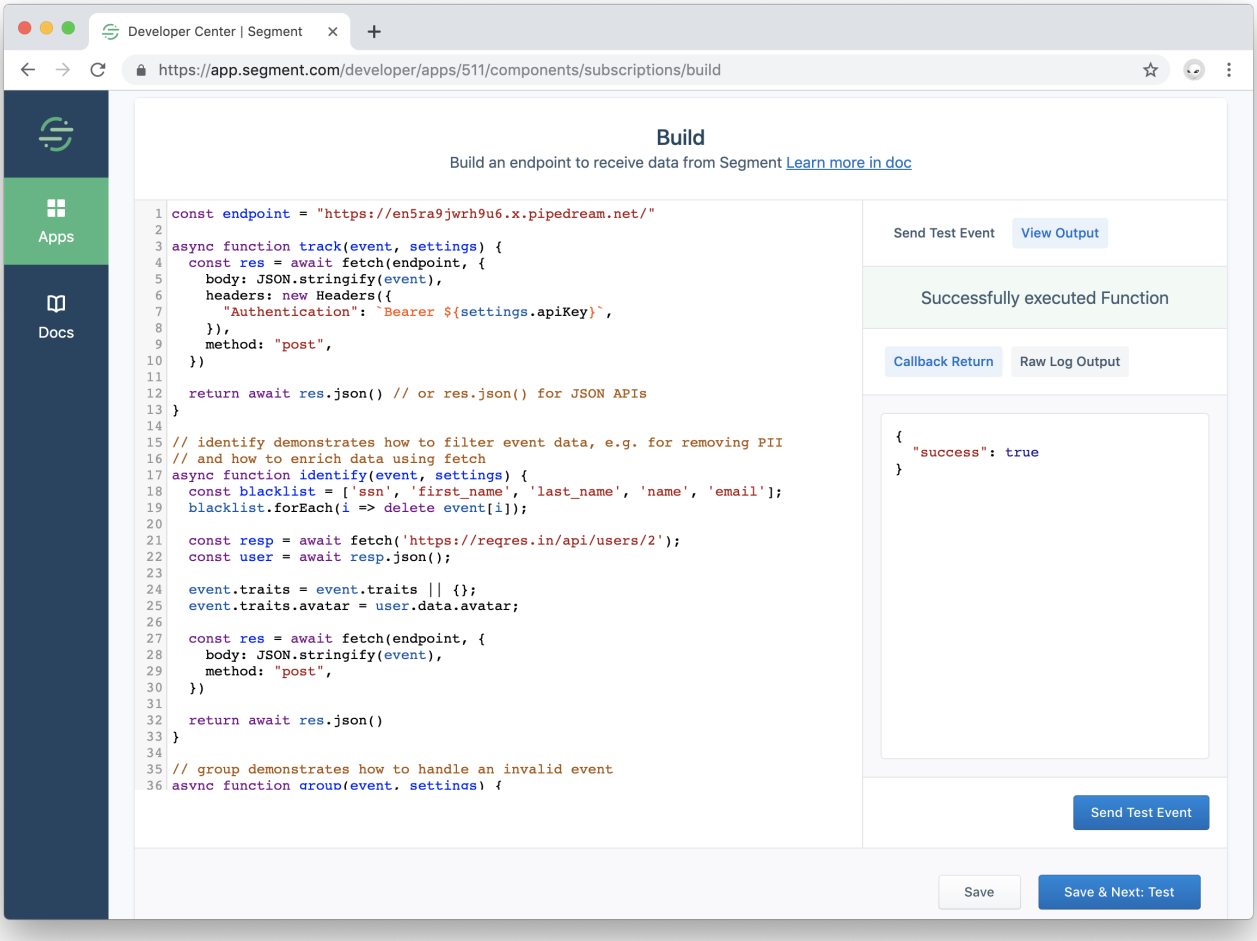
When testing your integration, proceed through two separate flows:

1. Test that your endpoint successfully ingests data in the way you would expect.

2. Mimic a user implementing your integration within their Segment workspace.

Your Endpoint

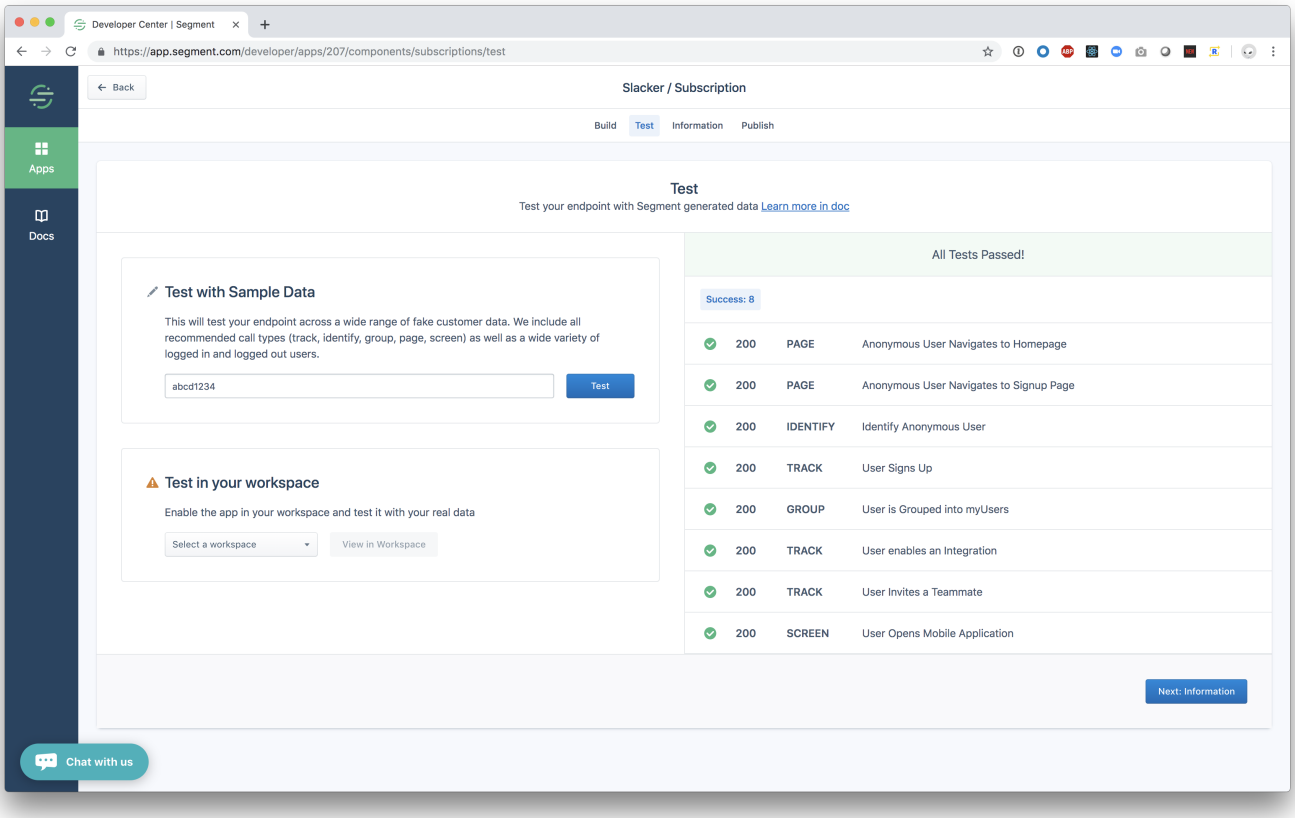
Test your code directly from the Developer Center UI. Use the `Send Test Event` button and review the test event to make sure your function works as expected.



In the debugger panel, check the two outputs. The **Callback Return** and the **Log Output**.

- **Callback Return** - What data your function returned or error it threw.
- **Log Output** - The raw log. Any messages to `console.log()` from your function appear here.

When your code is working with one event you can test it with a suite of more Segment events. Click `Save and Next: Test`, fill in an `API Key` and click `Test`. You will see the results of additional types of Segment data.



The User Flow

The ultimate goal is for Partners like yourself to create and publish high quality Destinations in the Segment Catalog. Your Segment account doubles as a sandbox account to test your destination while you are still in a private "building" state.

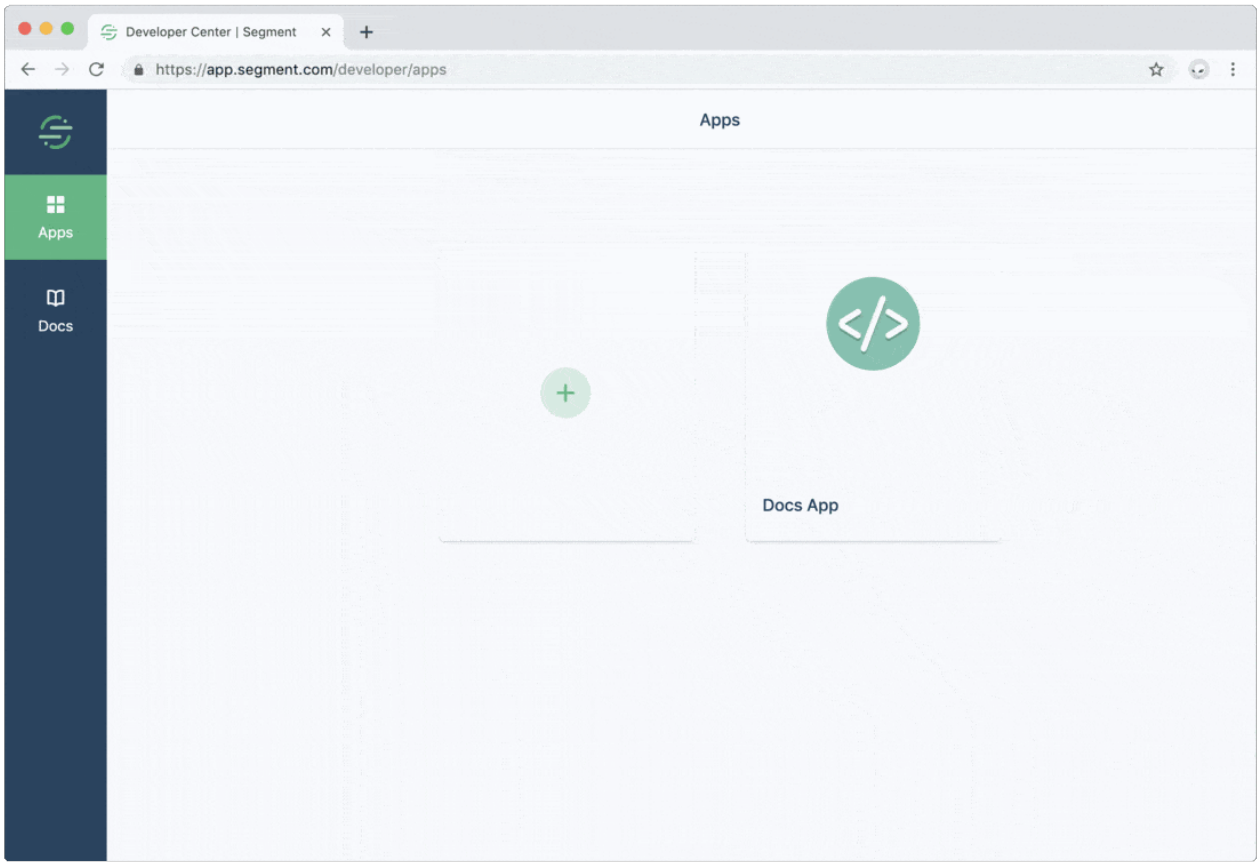
To test your Destination in the Catalog, click the "Test" tab in the Developer Center Component builder. In the "Test in your workspace" section, select your personal workspace and click "view". This redirects to you a URL like https://app.segment.com/WORKSPACE-SLUG/destinations/catalog/APP-SLUG, which is your catalog entry.

From here, click "Configure App", select a Source, and click "Confirm Source". You can now configure your destination by setting the "API Key", then clicking the toggle to enable the destination.

Next you can click the "Event Tester" tab to send data to your destination. Here you can see what requests Segment sends to your destination and introspect the response you are returning.

Now you can use the JavaScript SDK in a browser to generate real analytics events.

Finally you should verify the data in your service.



Handling deletions

In addition to the five primary spec methods, Segment forwards partners a sixth message type for customer-requested deletions. Destination Partners with access to the Developer Center are *required* to implement and document support for this federated user deletion.

Here's what a payload for deletion request looks like.

```
{
  "type": "delete",
  "channel": "server",
  "messageId": "delete-022bb90c-bbac-11e4-8dfc-aa07a5b093db",
  "projectId": "abcd123",
  "userId": "5678",
  "context": [],
  "integrations": [],
  "receivedAt": "2019-02-19T23:58:54.387Z",
  "sentAt": "2019-02-19T21:58:54.387Z",
  "originalTimestamp": "2019-02-19T23:58:54.387Z",
  "timestamp": "2019-02-19T23:58:54.387Z"
}
```

Next Steps

Complete the remaining steps as outlined in the Developer Center Overview

This page was last modified: 21 Apr 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