9. Launch into *Public Beta*!

Build

Begin by selecting "Build a direct destination" within the Developer Portal. Next, you will see a field to input your destination name and slug (once created you will not be able to change the destination slug). Within the "Build and test" section of the portal UI, add the destination endpoint where Segment data will be forwarded to.

Continue reading below to understand what is expected when accepting and responding to Segment data.

Accepting Segment Data

To receive data from Segment, you must provide a server with a static endpoint that can accept HTTPS requests.

The endpoint must:

- *Accept POST requests.* Segment sends customer data to the endpoint you designate in POST requests.
- *Accept JSON data.* Segment sends data in JSON.

Use HTTPS. Segment transmits potentially sensitive data on behalf of customers, and HTTPS is the first step in making sure their data stays safe.

Authorization

Segment sends your user's API key with requests, and you can use it to authenticate requests. This is the API key *you* give to your users; it is not a Segment API key.

Segment sends the key in the `Authorization` header using the `Basic` authentication type. It is Base64 encoded with your user's API key as the username, and an empty password.

For example, if your user's API key was `segment`, Segment would Base64 encode the string `'segment:'` and prepend the string `'Basic '`. The colon is always present, even when the password is absent.

This would result in a final string of `'Basic c2VnbWVudDo='`. This is what is contained in the `Authorization` header. Like any Authorization header, you must decode the string when you receive it.

See the headers section for more details.

Custom settings

All direct destinations have an API key setting by default, which Segment will send in the Authorization Header. To add more custom settings, go to the `Settings Editor` page. Any custom settings you add will be sent in the custom header `X-Segment-Settings` (See the headers section for more details.)

Headers

Segment sends you the following HTTP headers with all requests:

HEADER	DESCRIPTION	EXAMPLE
`Accept`	Segment accepts any content type, but ignores responses unless this header is set to `application/json`.	`Accept: */*`
`Authorization`	Segment sends your user's API token in this header, with the `Basic` authentication type.	`Authorization: Basic c2VnbWVudDo=`
`Cache-Control`	Each request Segment sends is a new event. Segment does not expect your application to cache.	`Cache-Control: no-cache`
`Connection`	Segment uses HTTP/1.1's keep-alive functionality whenever possible, however this is optional.	`Connection: Keep-Alive`
`Content-Length`	Segment always sends you the length of the request in bytes.	`Content-Length: 348`
`Content-Type`	Segment indicates the type of data it sent you (this will always be JSON), along with Segment's vendor type.	`Content-Type: application/json`

HEADER	DESCRIPTION	EXAMPLE
`User-Agent`	Segment sends you this field every time.	`User-Agent: Segment.io/1.0`
`X-Segment-Settings`	Except for the API key (which is sent in the Authorization header), Segment will send the base 64 encoding of the rest of your custom settings encoded in this header.	`X-Segment-Settings: eyJjdXNOb21TZXROaW5nT25lIjoiY3VzdG9tIHNldHRpbmcgdmFsdWUifQ==`

Request body

Segment's Spec standardizes the data that you can expect from Segment. You can choose to implement four types types of calls:

- Who is this? `.identify(userId, traits)`
- What are they doing? `.track(userId, event, properties)`
- Where are they doing it? `.page(userId, pageName, properties)`
- What group are they part of? `.group(userId, groupId, groupTraits)`

For example, you might implement the `.identify(userId, traits)` call to create contacts in an email marketing application. You can expect the following customer information as a JSON object in the call body:

```
{
  "anonymousId": "1234",
  "context": {
    "ip": "8.8.8.8",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/40.0.22
14.115 Safari/537.36"
  },
  "messageId": "022bb90c-bbac-11e4-8dfc-aa07a5b093db",
  "receivedAt": "2015-02-23T22:28:55.387Z",
  "sentAt": "2015-02-23T22:28:55.111Z",
  "traits": {
    "name": "John Doe",
    "email": "john.doe@email.com",
    "plan": "premium",
    "logins": 5
  },
  "type": "identify",
  "userId": "5678",
  "version": "1.1"
}
```

The casing on these fields will vary by customer, so be ready to accept any casing.

Status code

Segment uses standard HTTP status code conventions to help diagnose problems quickly and give better insight into how the destination is working.

Upon receiving data, your endpoint should reply with one of the following status codes:

CODE	REASON
200	You've accepted and successfully processed the message.
202	You've accepted the message, but have not yet processed it.
400	The message is malformed, or otherwise contains an error that is the client's fault.

CODE	REASON
401	The client's API key is malformed, has expired, or is otherwise no longer valid.
403	The client's API key is valid, but has been rejected due to inadequate permissions.
500	If you encounter an internal error when processing the message, reply with this code. (Hopefully you won't have to send too many of these.)
501	If Segment sends you an API call type (indicated by the `type` property included on all messages) you don't support, reply with this code. Read more about the API call types Segment supports in the Spec docs.
503	Send Segment this code when your endpoint is temporarily down for maintenance or otherwise not accepting messages. This helps Segment avoid dropping users' messages during your downtime.

Response body

You can normally send back an empty body, but when sending back a `4xx`- or `5xx`-class error, you can optionally send Segment a diagnostic message that explains the error. This message is displayed to the user in the Segment debugger, and is be used in Segment's Event Delivery summaries.

Be sure to send JSON (and set your `Content-Type` header to `application/json`), and send your message in the `message` property.

Here's an example of a `401` response that helps a user track down why their calls aren't appearing in your tool's UI:

```
{
  "message": "API token expired"
}
```

Or, if your tool requires an email address in order to accept calls, use this example `400` reply:

```
{
  "message": "Missing email address"
}
```

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 Portal UI. Use the `Send Test Events` tab and review the test events to make sure your destination works as expected.

In the debugger panel, check the two outputs. The **Request from Segment** and the **Response from destination**.

- **Request from Segment** - What Segment posted to your endpoint
- **Response from destination** - How your server responded

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 "View in workspace" button in the "Test in your workspace"

section. 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. Learn more about the event tester in the Event Tester docs.

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

Finally you should verify the data in your service.

Write documentation

Documentation is integral to enabling Segment’s users to self-serve and onboard with your integration. Segment’s documentation team will work with you during this part of the process to ensure your documentation matches the Segment style and is as instructive as possible.

To create your documentation, follow the instructions outlined in this template

This page was last modified: 12 Aug 2024