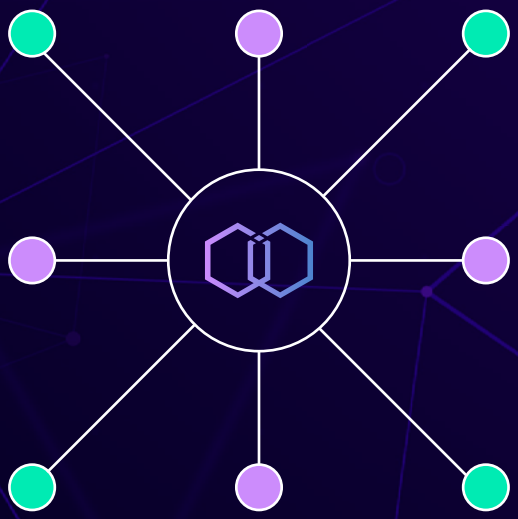
The Ultimate Guide To API Ingestion with Integrate.io

STEP-BY-STEP GUIDE



Integrate.io

Table Of Contents

01

INTRODUCTION

In an ideal world, ETL providers have natively built data connectors for every system that customers want to ingest data from.

With an estimated 30,000+ SaaS applications alone, and the average ETL provider supporting 100–150 native data connectors, we are a long way off this pipedream.

For times where there isn't a native built connector available for your data source of choice, a universal API connector provides an ideal solution for ingesting your data. With many ETL providers now offering their own variations of API connectors, it's critical to weigh up the pros and cons of each and to ensure you choose a solution that is right for your current and future needs. Two of the main points to evaluate an API connector on are ease of use (how easy is it to use and how much time is this saving us) and flexibility (can this handle my current and future API ingestion needs).

While native connectors will typically be the preferred option if they are available, there are also many times where beng able to ingest directly from the source using a universal API connector is a better option.

Some of these times include:

1 LONGER-TAIL DATA SOURCES

These are systems where native connectors are not currently available and are too down the pecking order to have a native connector built out any time soon.

2 INTERNAL/PRIVATE APIS

These are company-specific APIs which will never be supported via native connectors

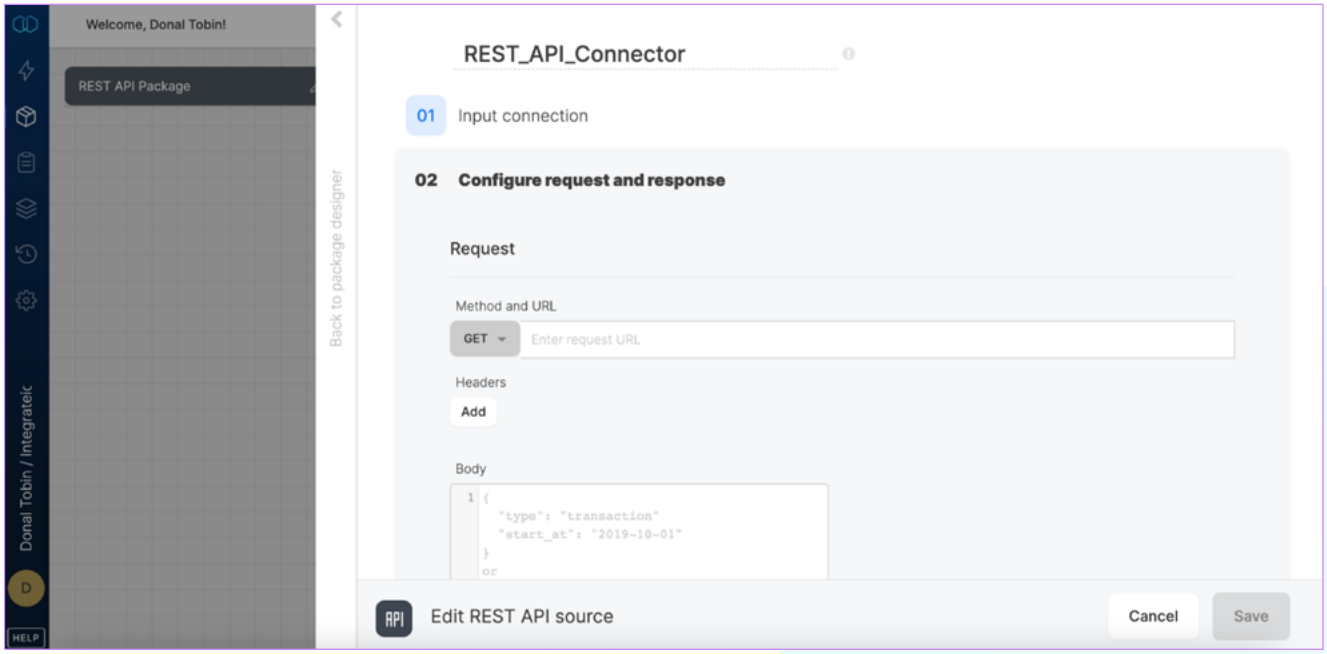
3 FREQUENTLY CHANGING APIS

Unfortunately, many companies make significant updates to their APIs at will and with little to no forewarning. Having to maintain native connectors for these frequently changing APIs results in significant time being spent updating connectors.

4 UNRELIABLE NATIVE CONNECTORS

Native connectors don't always live up to expectations. It's common for native connectors to have issues, be missing fields, data type mismatches, and many other issues.

Integrate.io prides itself on its universal API connector's flexibility and ease of use. Within minutes, you can be ingesting from any API. No two APIs are the same but the following guide walks you through the most common types of APIs Integrate.io customers ingest from. By the end of this guide, you will be an API ingestion pro and ready to tackle ingesting the most complex of APIs!



02

GETTING STARTED

Integrate.io's universal REST API source component gives you the ability to access data from an infinite number of applications and systems.

This guide will walk you through the process of accessing your API data. We will start with reading the API documentation, finding the pertinent information you need, and showing you how to configure the REST API source component with that information.

STEPS

1. **Find the API documentation.** Many APIs have their documentation publicly available on the internet, although occasionally some will only be accessible to users after logging in to the platform/console. If you can't find it, reach out to the customer support of your application.
2. **Seek out the particular information and credentials you need.** We need a few pieces of information from the documentation. They can typically be found in an Authentication section and a section that describes the objects/tables of the API and the ways you can interact with them (i.e. endpoints):
   - How does the API require us to pass our authentication credentials?
   - What is the URL of the API endpoint we want to retrieve data from?
   - What method does your endpoint use? (Such as GET, POST, PUT, etc)
   - Does the API require any headers, query parameters, or body?
   - How is the data returned?
3. **Configure the REST API source component inside an Integrate.io ETL package.** See the following examples for instructions.

03

INGESTING API DATA IN INTEGRATE.IO ETL – TUTORIAL

These three sample API calls build upon one another in complexity while demonstrating the flexibility of the REST API source component and covering the majority of the types of APIs that Integrate.io customers call within the platform. We will refer to the **STEPS ABOVE ^** as we walk through the process of accessing the data from each of the following APIs.

#1 Stripe API – GET method with an API key

STEP 1

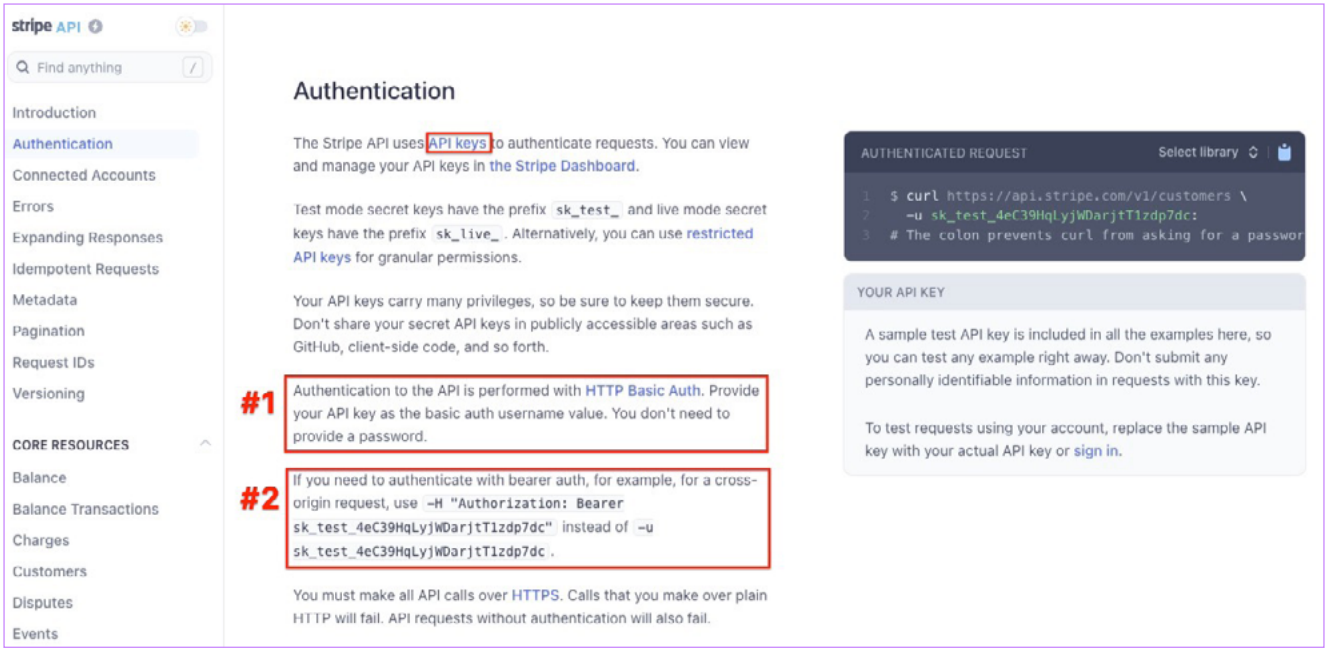
My company uses Stripe for billing and I'm trying to ingest that data into my data warehouse for better visibility for my Customer Success team. I search for "Stripe API documentation" and find it at stripe.com/docs/api.

STEP 2

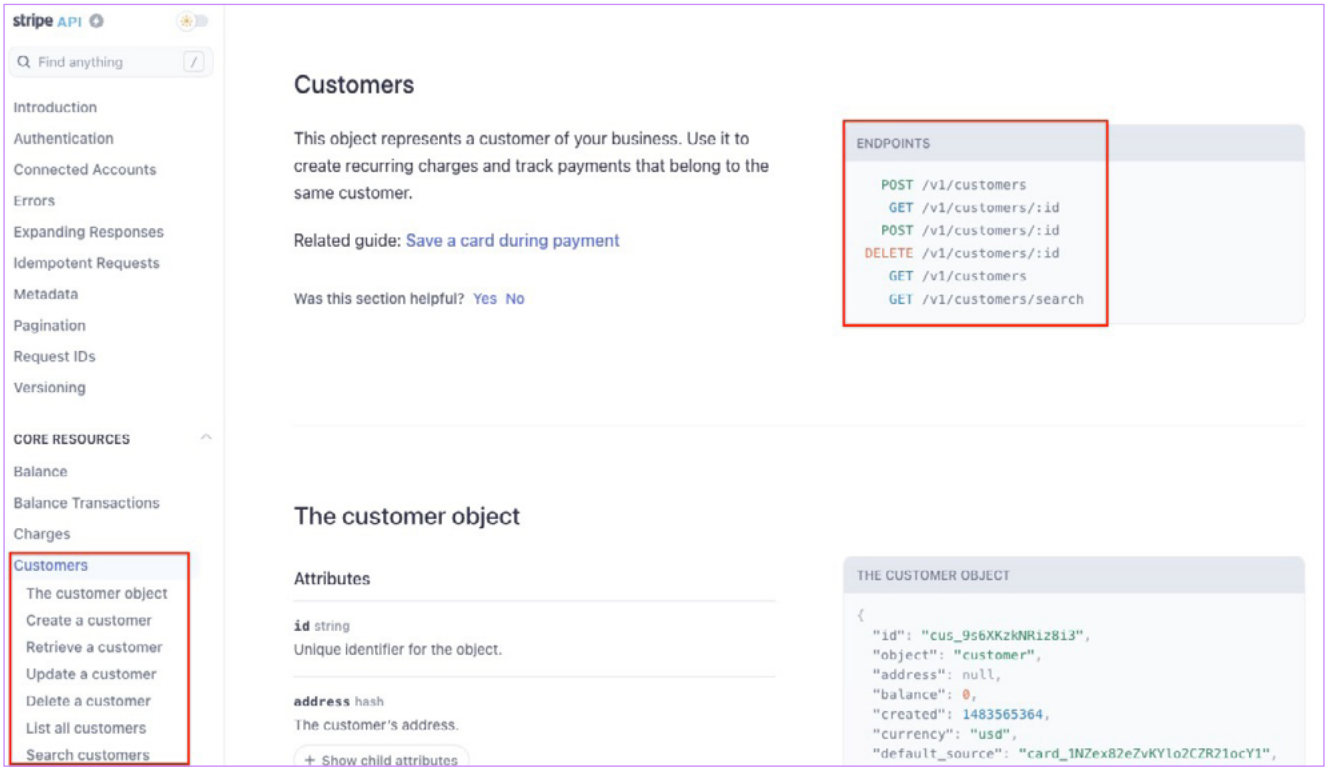
I see "Authentication" in the left sidebar. This section of the documentation gives us the information I need for **STEP 2A ABOVE ^**. It explains that Stripe uses an API key for authentication, and that there are two ways you can pass that API key to the API:

1. Using Basic Authentication it is the username and no password is required.
2. Or you can pass it as the token in an Authorization header like this:

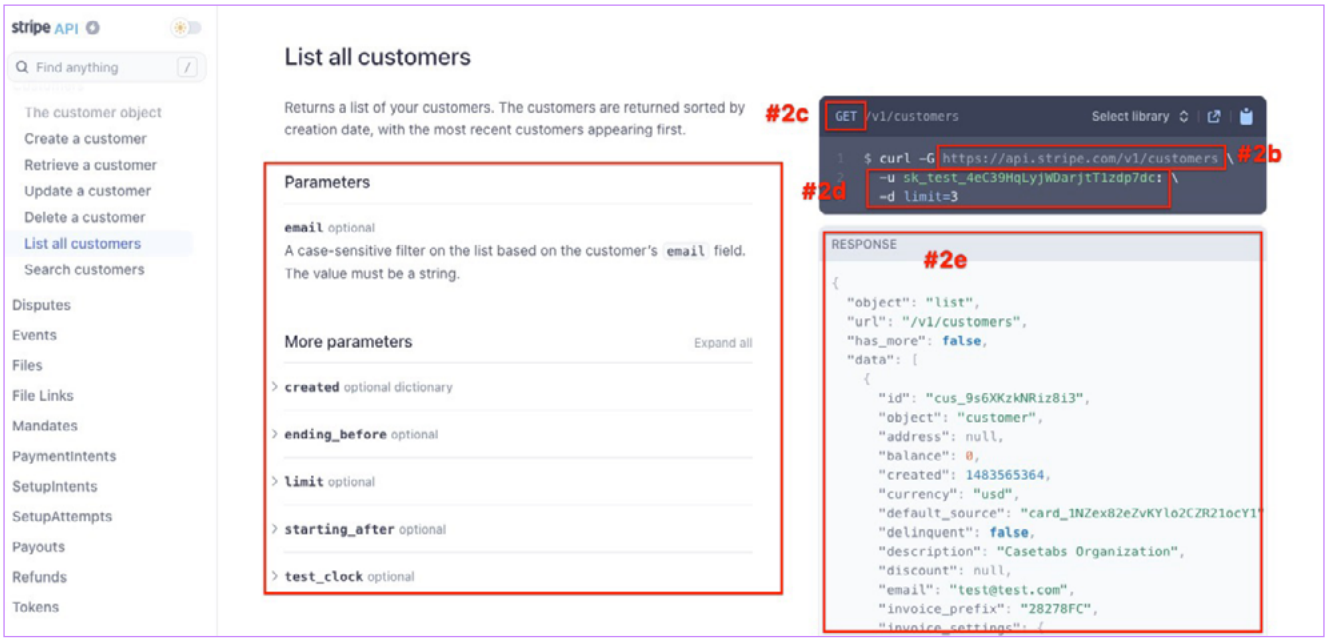
   Authorization : Bearer <your API key>



Next, I need to find the particular object/table where the data I want is stored. I'm interested in customer data and I see Customers in the left sidebar. When I click on it, it expands to show me actions that are available with this data and takes me to this page where I can see the endpoints that are associated with those actions:



I want a list of all the customers. I can find that if I scroll down:

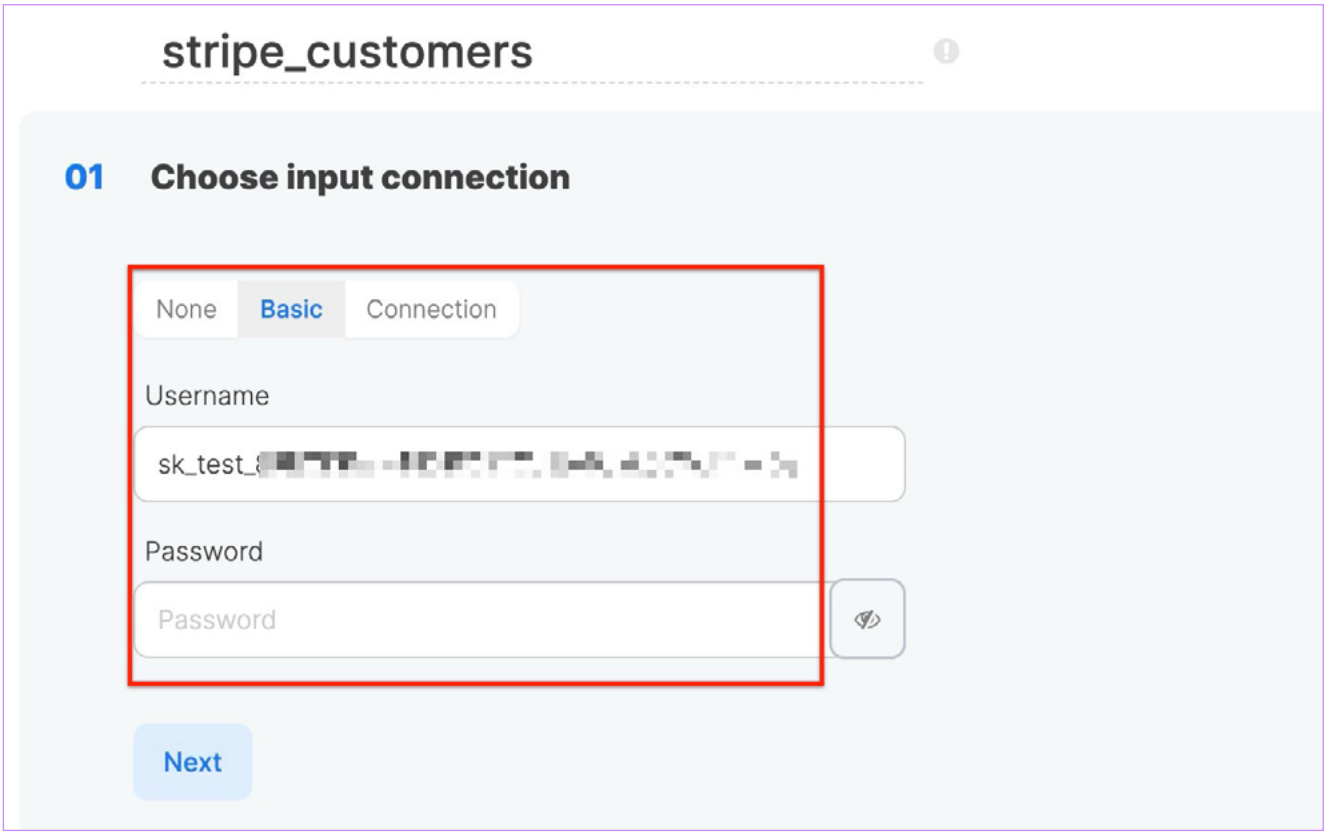


This section of the documentation will give me the rest of the information I need for **STEPS 2B–2E**. It tells me:

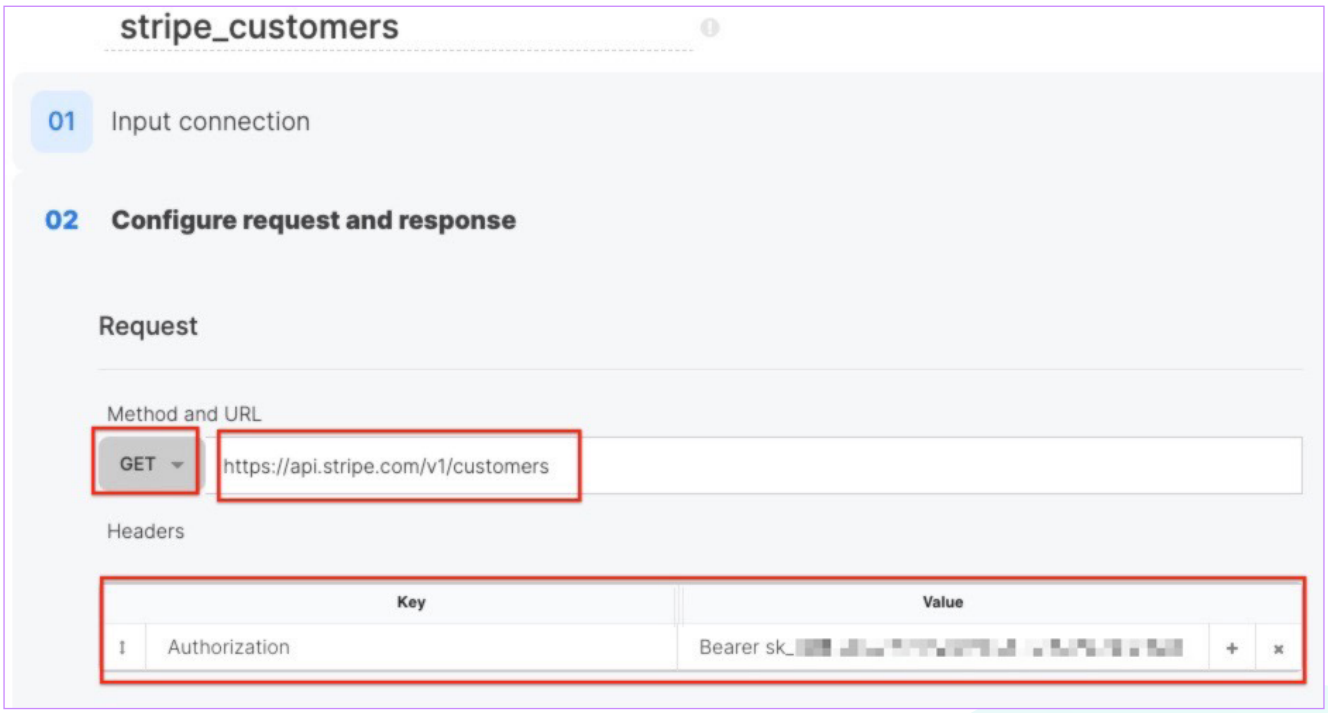
- **2B:** The URL is https://api.stripe.com/v1/customers
- **2C:** The method is GET
- **2D:** There are no required parameters or headers (they would show on the right hand dark gray box in the Curl function as -h). All of the parameters that are listed on the left are optional.
- **2E:** The data from the API is returned in a JSON object (indicated by the curly brackets). However, you can verify that in the Introduction section of the API documentation as well.

STEP 3

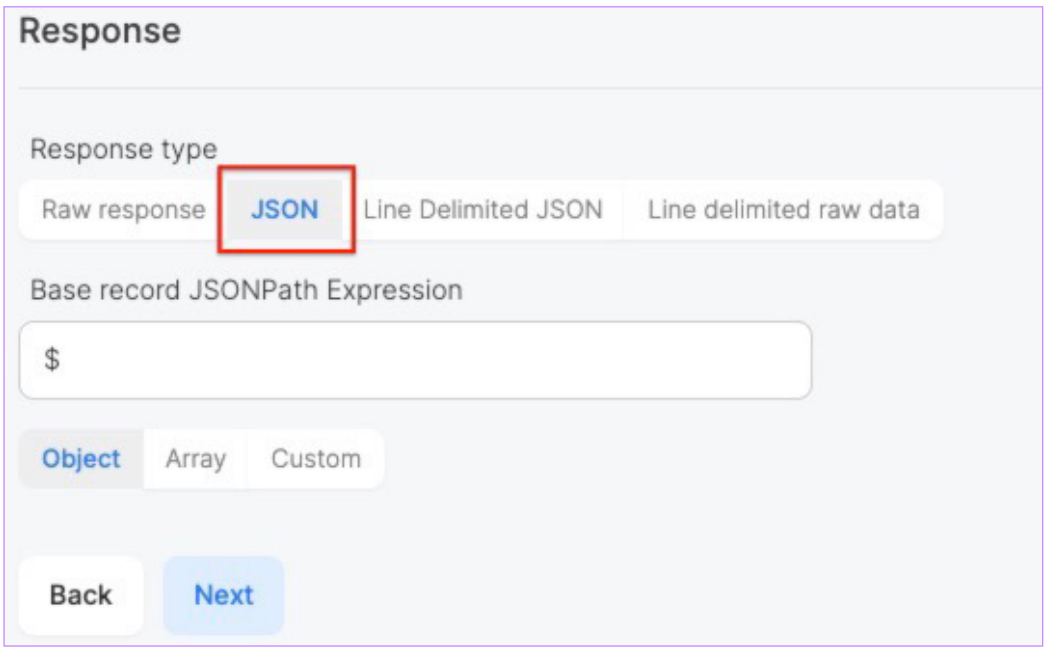
Now I can configure the REST API source component in a dataflow package within Integrate.io. If I'm going to pass the API key as the username in Basic Authentication, I select "Basic" in **STEP 1** of the REST API component and place the API key in the username field, leaving the password field blank:



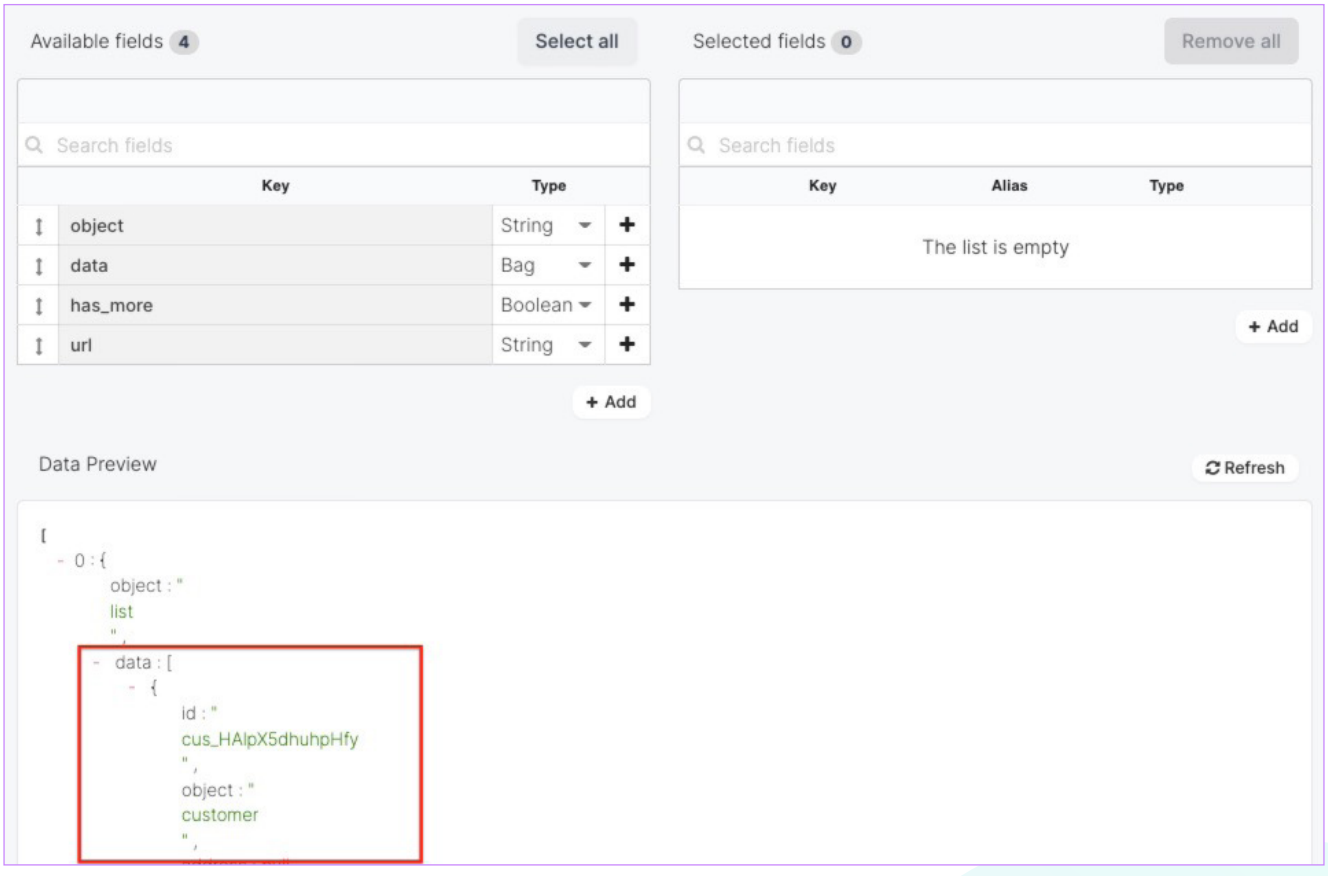
Alternatively, if I'm going to pass the API key as the token in the Authorization header I select "None" in **STEP 1** and click "Add" below the word Headers in **STEP 2**. Then I fill in "Authorization" on the key side and "Bearer <my API key>" on the value side. This is also where I'll place the URL and select GET in the method dropdown:



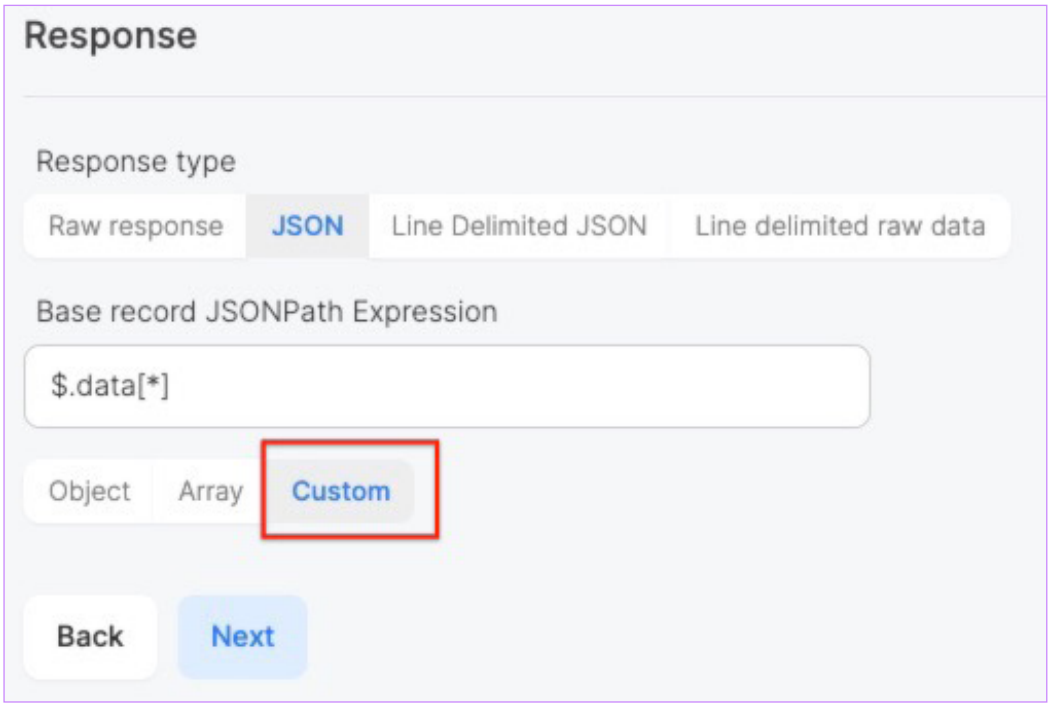
Then I scroll down to the Response section, select JSON since I know that's how my API will return its data, and click Next.



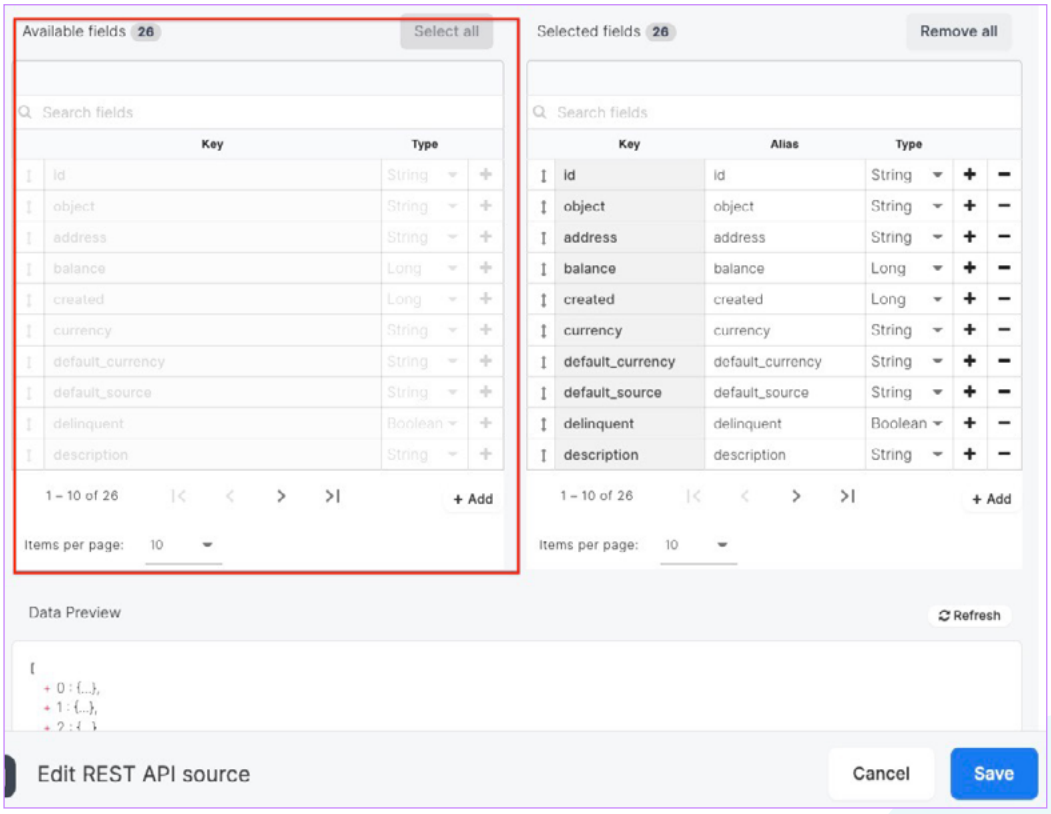
I can see the four fields being returned by the API. In the Data Preview, I can see the customer data that I'm most interested in is inside the data bag (i.e. array).



I go back to **STEP 2** and edit the Base record JSONPathExpression selection from Object to Custom. The component automatically adds **$.data[*]** to the field. This path tells the component to parse all the fields inside the data bag/array. Click Next.



The component is now parsing the customer fields that I want. I click Select All and Save the component.



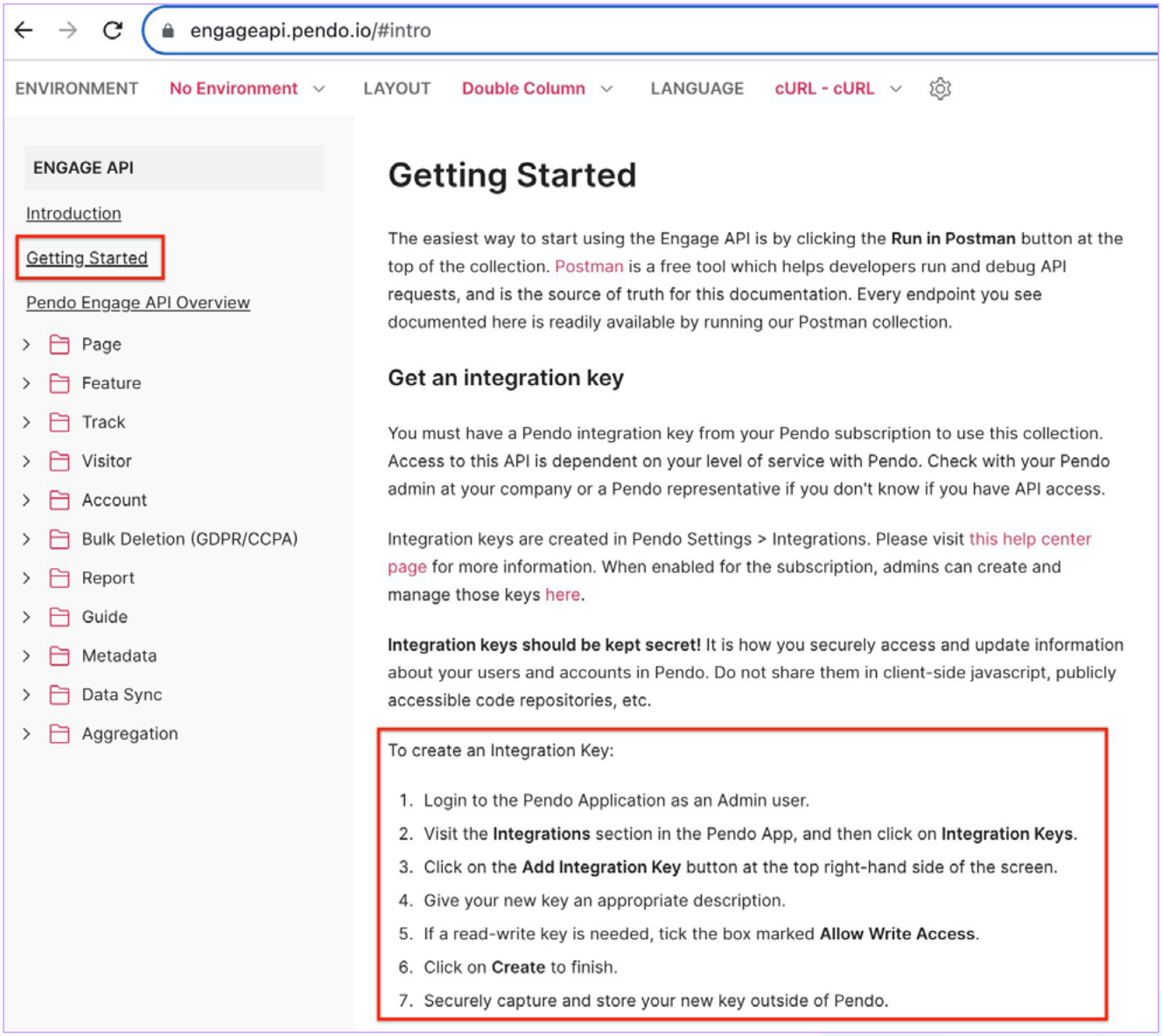
#2 Pendo API – POST method with an API key

STEP 1

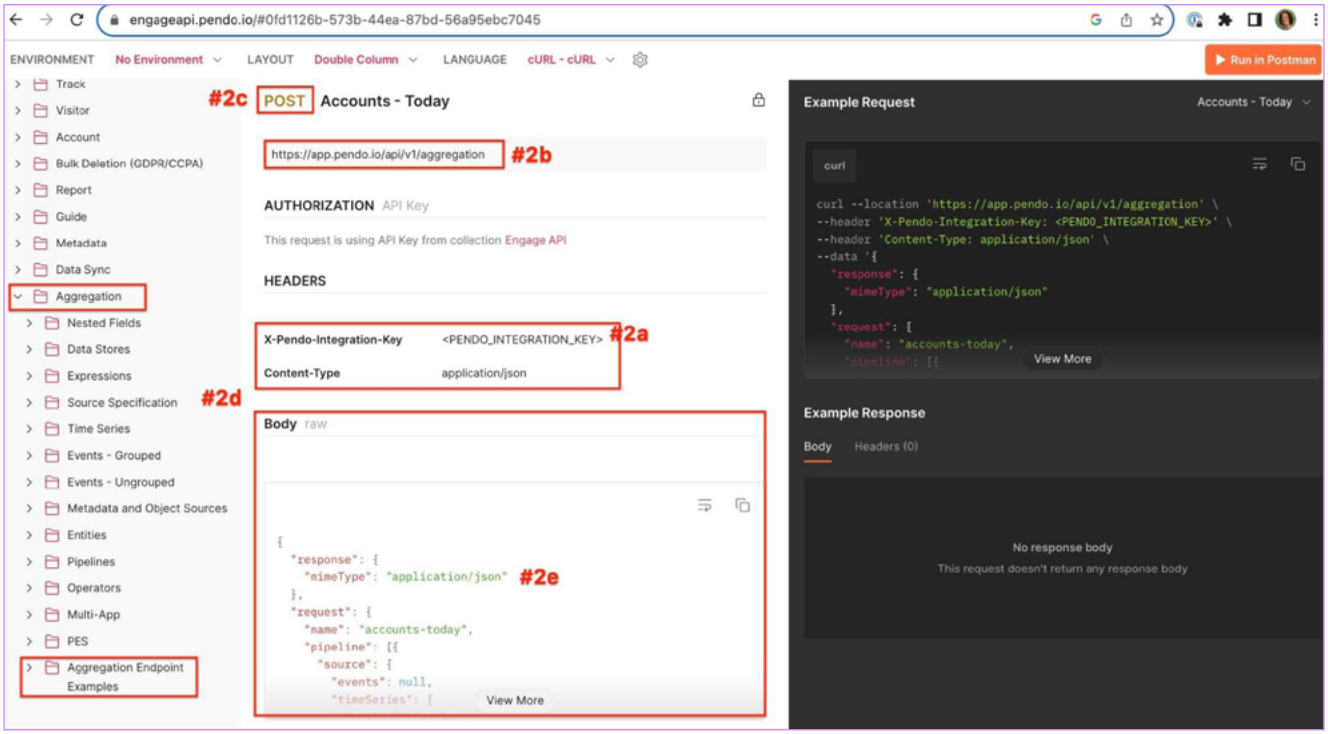
In this next example, I need to extract data from Pendo. I search for “Pendo API documentation” on the internet and find it at https://engageapi.pendo.io.

STEP 2

I see “Getting Started” in the left sidebar. This section of the documentation tells me that I need an Integration Key in order to access the API and walks me through the steps I need to follow to get one.



Next, I know I want aggregation data so I click on “Aggregation” in the sidebar. I see “Aggregation Endpoint Examples” so I click on that and look at the “Accounts – Today” example:



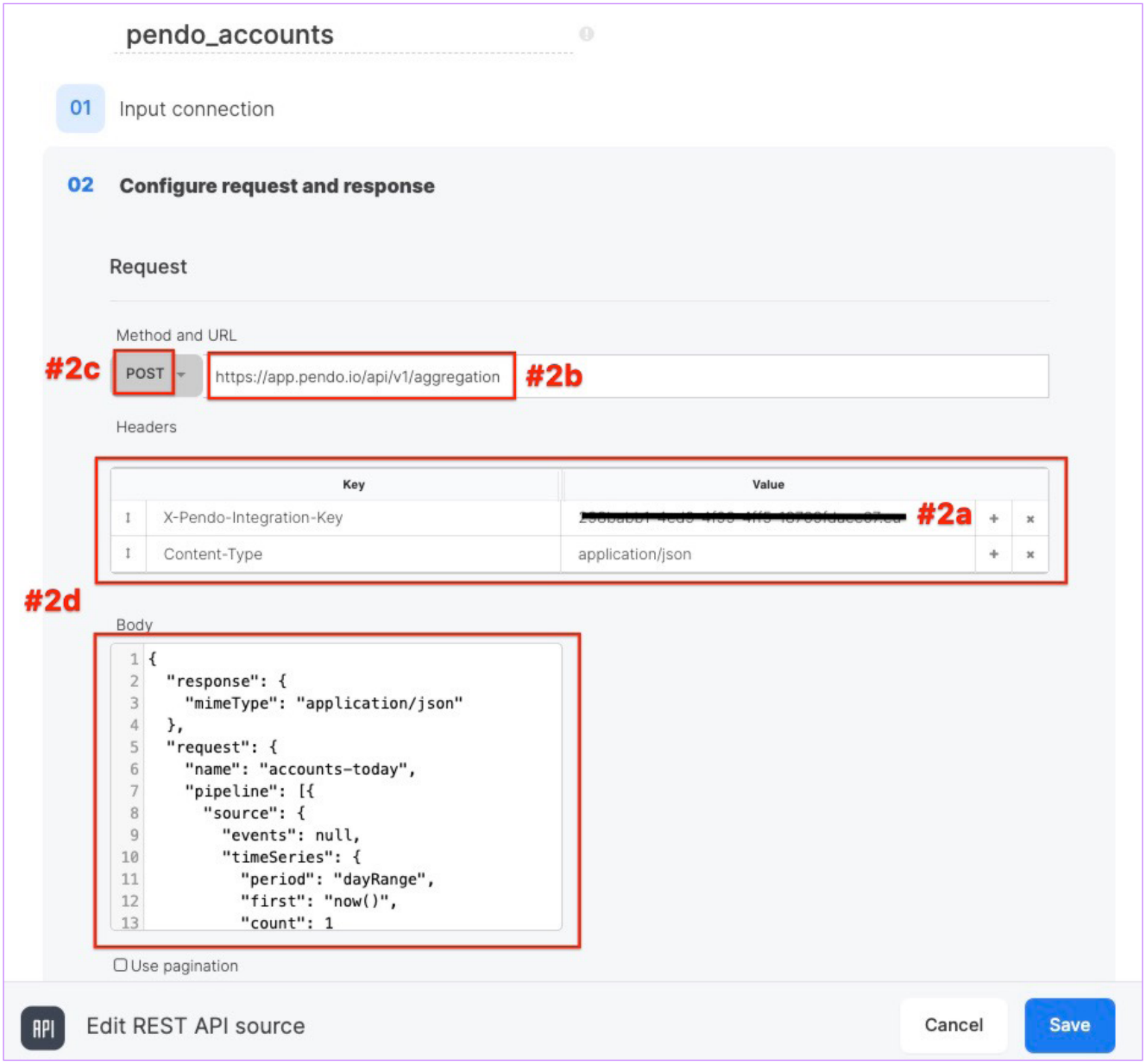
This section of the documentation will give me the rest of the information I need for **STEPS 2A–2E**. It tells me:

- **2A:** The API key is passed in a X-Pendo-Integration-Key header.
- **2B:** The URL is https://app.pendo.io/api/v1/aggregation
- **2C:** The method is POST. **A POST method means we are sending data to the API. This will mean that you are sending a Body field and a Content-type header to tell the API what format to expect the data in the Body field to take**. Common types are:
  - JSON which requires an application/json content-type header
  - Url-encoded which requires an application/x-www-form-urlencoded content-type header
- **2D:** There are two required headers and a body field:
  - **X-Pendo-Integration-Key : <PENDO_INTEGRATION_KEY>**
  - **Content-type : application/json**
  - You can click on View More to see the entire Body field of this example call.
- **2E:** Within the Body field, we are passing a request to return the data in JSON format.

I can see the Curl function in the dark gray box on the right and how each of those pieces of information is being passed to the API.

STEP 3

Now I can configure the REST API source component in a dataflow package within Integrate.io.



After configuring this part of the component, the rest of the steps are the same as in the first example with the Stripe API.

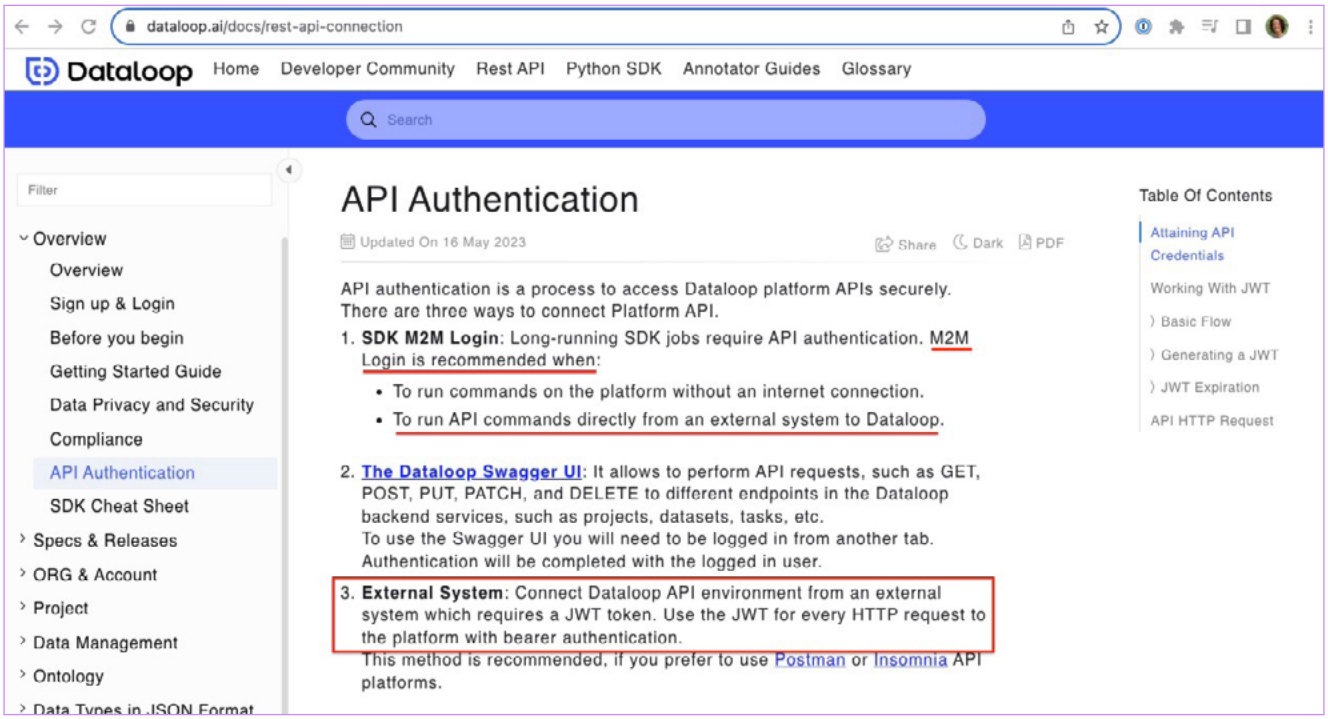
#3 Dataloop API – OAuth2.0 with a short-lived JWT

STEP 1

In this example, I need to extract data from the Dataloop API. I search for “Dataloop API documentation” on the internet and find it at https://dataloop.ai/docs/rest-api-connection.

STEP 2

The Dataloop API documentation opens right up to API Authentication. I read that for a M2M (machine to machine) login from an external system (like Integrate.io ETL) the API requires a JWT (java web token) to be passed with every request for data.



Reading on I find out that I need to pass my username and password to the API in order to get a JWT, but the JWT will expire in 24 hours.

Generating a JWT

The request schema is as follows:

POST https://gate.dataloop.ai/token?default
Content-Type: application/json
Body

JSON | Copy

```
{
    "username": "<user name>",
    "password": "<user password>",
    "type": "user_credentials"
}
```

Assuming a successful response, the body of the API will contain an access token, id token, and refresh token. I will need to pass the id token.

JSON | Copy

```
{
    "access_token": "...",
    "id_token": "...",
    "refresh_token": "...",
}
```

Use the **id_token** (JWT) from the response as your authorization JWT for future requests.

This means that I need to call this ^ token endpoint before I can make a request for data. Given these requirements, the setup for ingesting from this API will involve a couple of additional steps compared to the above examples.

STEP 3

OAuth2.0 flows like this one that the Dataloop API uses are becoming increasingly common. OAuth 2.0 uses Access Tokens. An Access Token is a piece of data that represents the

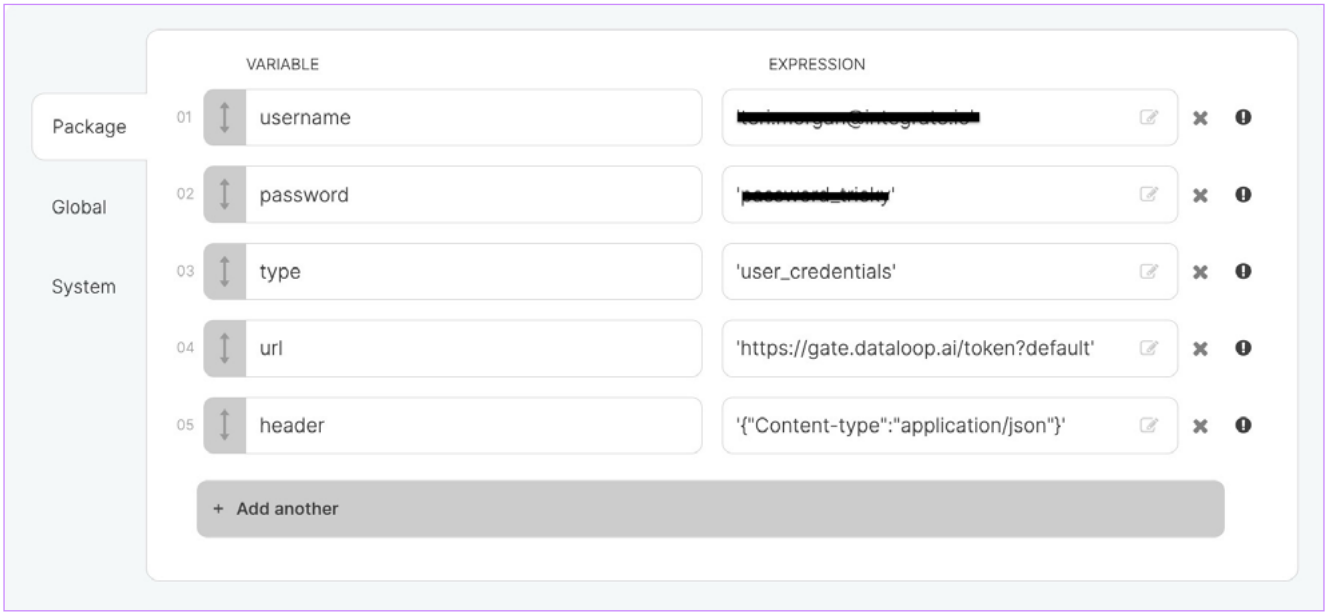
authorization to access resources on behalf of the end-user. At a high level, this process follows this flow:

1. The Client requests authorization (authorization request) from the Authorization server, supplying the client id and secret as identification and defining the grant type. Grants are the set of steps a Client has to perform to get resource access authorization. A system like Integrate.io ETL platform will often use the Client Credentials grant type which is used for non-interactive applications e.g., automated processes, microservices, etc.
2. The Authorization server authenticates the Client and responds with either an Authorization Code or Access Token, depending on the grant type. With the Client Credentials grant type it will be an access token.
3. With the Access Token, the Client requests access to the resource from the Resource server.

For more information on how OAuth2.0 works, see this article.

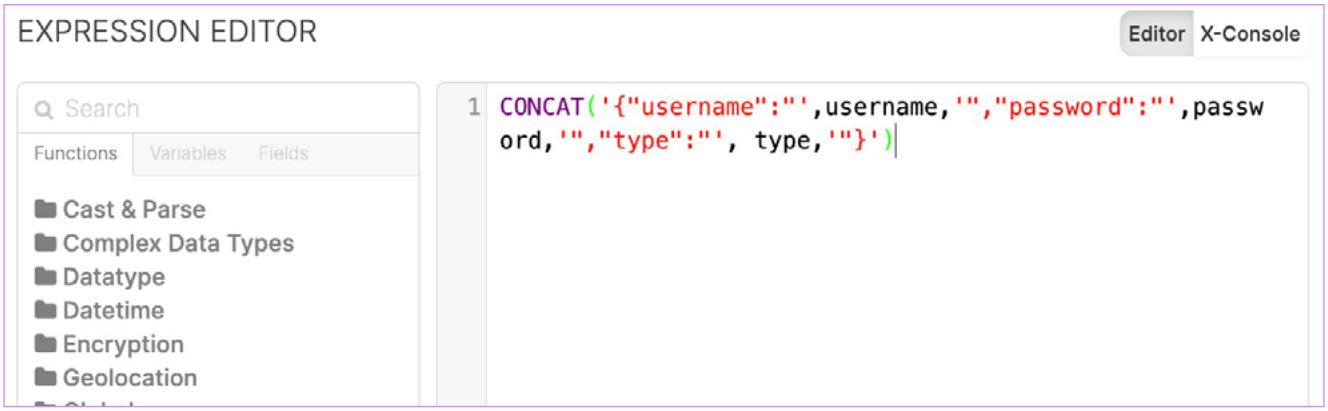
Now we will configure this in Integrate.io ETL: In order to facilitate calling the authorization endpoint before making a request for the data, I will use a Curl function in the variables section. When a job runs, the variables evaluate first. Thus, I will make a call to the authorization endpoint, retrieve my access token, and then pass the token variable into the dataflow which executes next. Let me show you how.

First I open the variables section (the button with the three dots on it in the upper right section of the dashboard.) I create a package variable to hold each of the credentials I will be passing to the authorization endpoint. For instance, in our Dataloop example **ABOVE ^** I need to pass in a username, password, and type. Keep in mind that these values all need to be string data types in order to be passed into the Curl function so enclose them in single quotes. Next, I will create a variable for the URL and the Content-type header. The header must be formatted as JSON so I follow this syntax: **{“key”:”value”}**

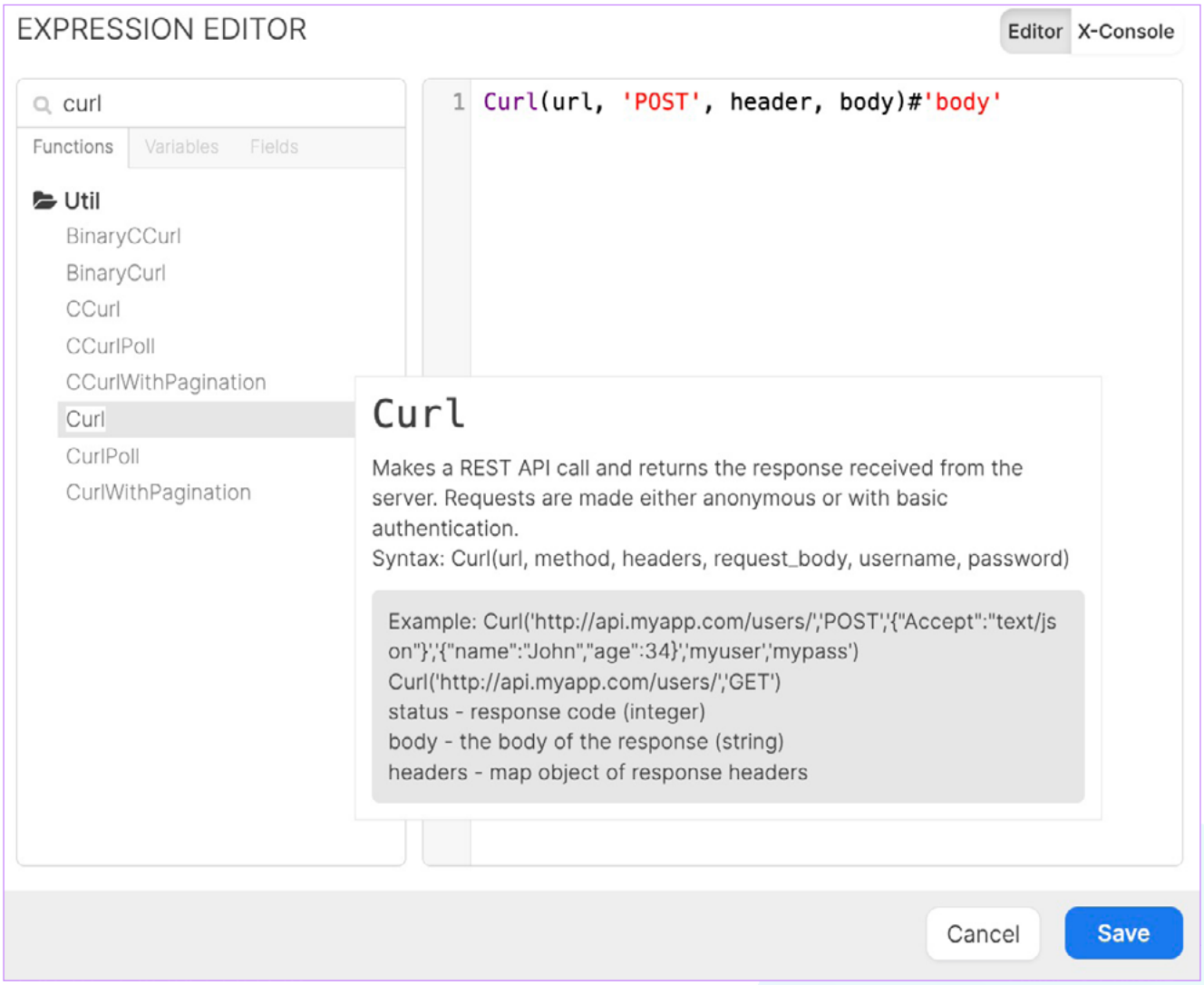


Next I will create a variable named “body” that will hold the JSON body field with our credentials that we’re sending to the authorization endpoint. After naming the variable, I click on the pencil and paper icon in the Expression field to open the Expression Editor.

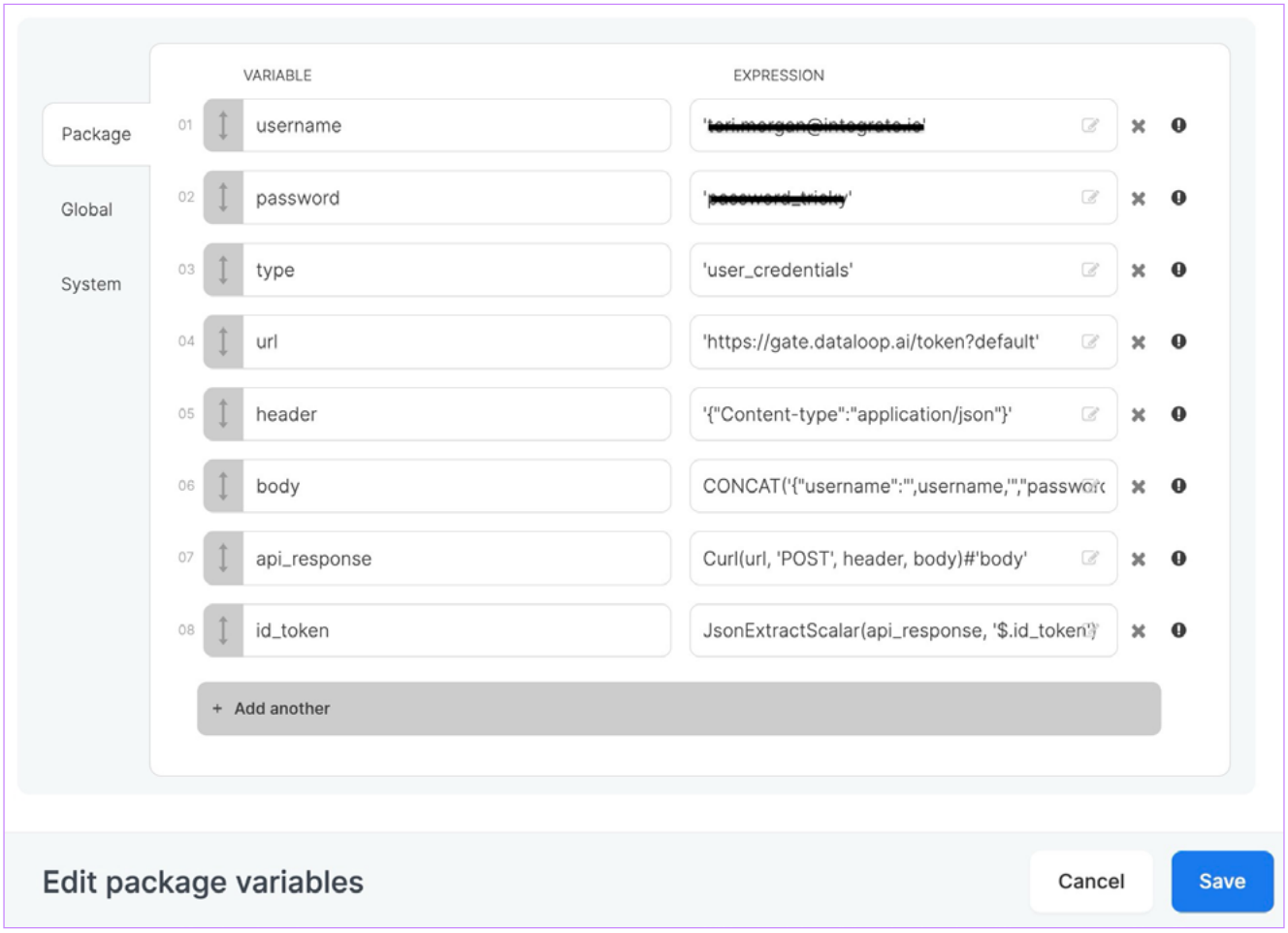
I will use a CONCAT function to combine the literal string parts with the variables I’m using to store the credentials. Make sure you remember to enclose each literal string in single quotes and separate the arguments with commas. See this doc for more information on the CONCAT function.



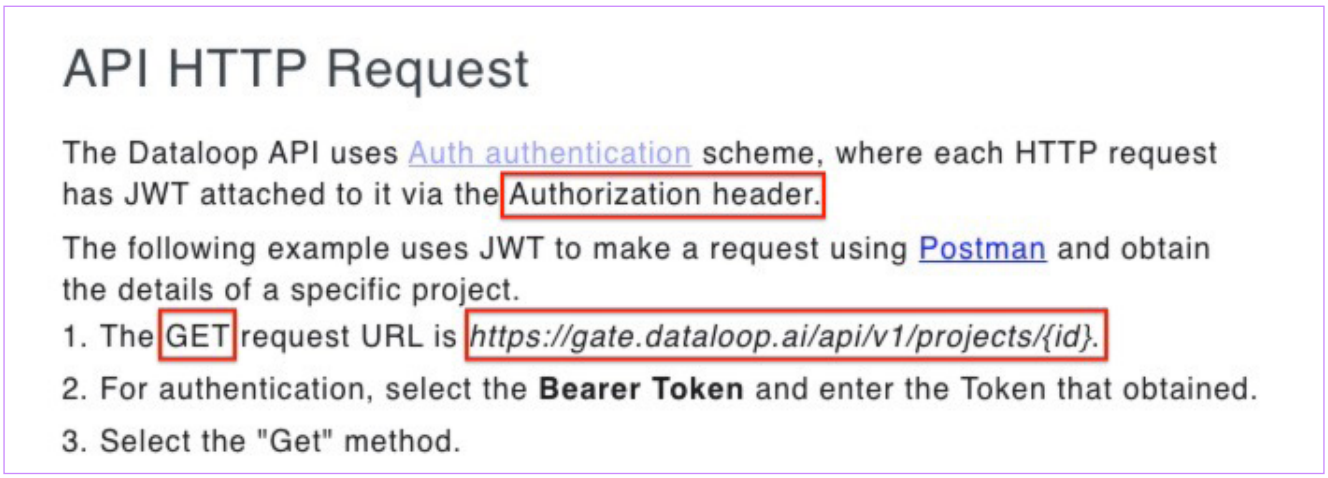
Next I will create the variable where I'll write the Curl function to call the authorization endpoint. You can see the syntax for this function in the tool tip. The username and password parameters mentioned here are for Basic authentication so I won't be using them. The only two required parameters are the URL and method. The Curl function will return a map that includes headers, body, and status. After the closing parentheses of the function you'll see #'body'. That syntax tells the function that I only want the body of the API response. See this doc for more information on the Curl function.



Here are the eight variables I created. Keep in mind that the variables evaluate from top to bottom so I can't use a variable in a subsequent step if I haven't declared it above.



Now that I have my id_token I'm ready to make a request for data. I will use this sample request described here in the API documentation:



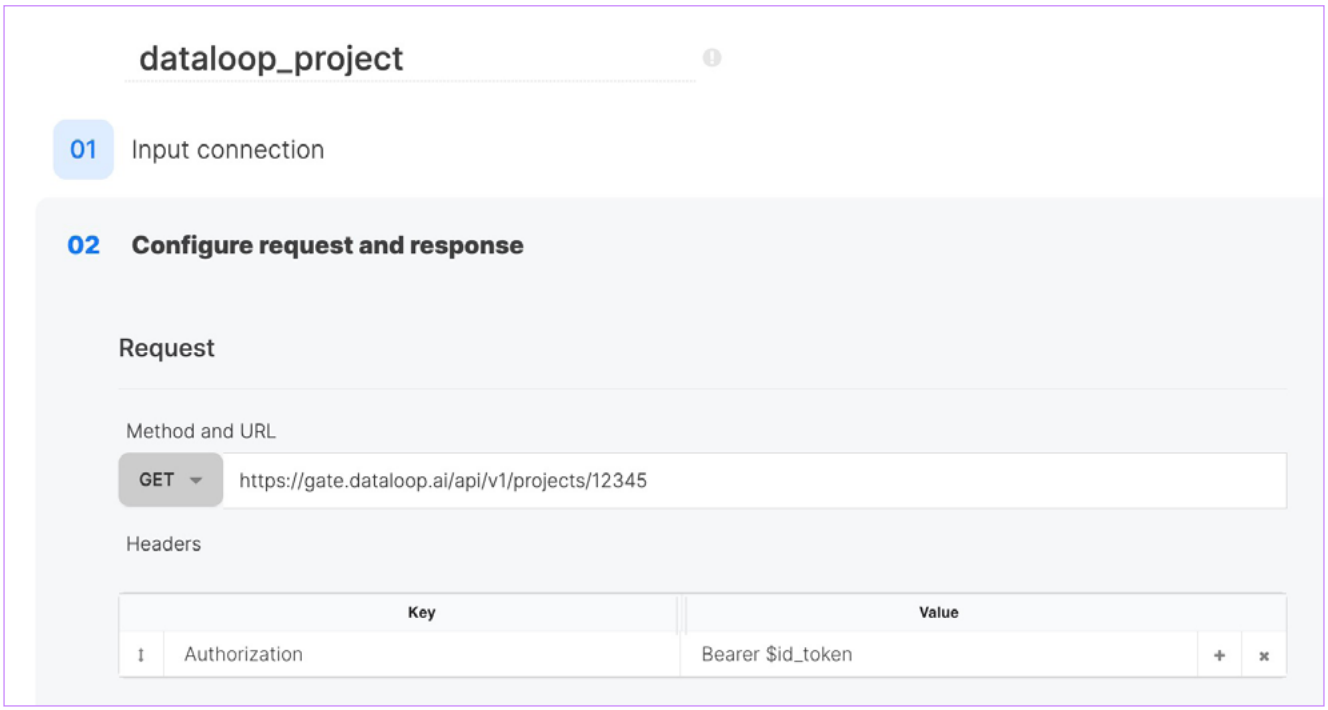
API HTTP Request

The Dataloop API uses Auth authentication scheme, where each HTTP request has JWT attached to it via the Authorization header.

The following example uses JWT to make a request using Postman and obtain the details of a specific project.

1. The GET request URL is *https://gate.dataloop.ai/api/v1/projects/{id}*.
2. For authentication, select the **Bearer Token** and enter the Token that obtained.
3. Select the "Get" method.

Finally I can configure the REST API source component and call the id_token variable. The $ is used in the Integrate.io ETL platform to indicate that something is a variable.



When the job runs, the variables will evaluate and by the time this component in the dataflow executes, that id_token variable will be replaced with the id_token value. However, while you are setting up the REST API source component you need it to call the API and return the data in order to be able to complete the setup and save the component. Place a hardcoded token value in place of the $id_token so that the component will load. Then once you are finished building the package, go back and replace the hardcoded value with the variable name and re-save.

Congratulations, you're now an API ingestion expert! We're always looking for new types of APIs to ingest from, ways we can improve our API ingestion capabilities, and of course just talking shop if that's of interest! Reach out to any of our team should there be anything we can help you with or drop us an email at hello@integrate.io.

Happy API ingesting!

We hope that you have enjoyed reading this guide.

If so, be sure to check out our additional content related to "The Ultimate Guide To API Ingestion with Integrate.io".

Interested in learning more about Integrate.io?

Chat with an Expert!