Exercise 8: Stripe API

The exercises so far have used imaginary, fairly simple APIs. Let's now take a piece of an actual API and create an OAS file for it. Expect this exercise to take some time, at least a couple of hours.

Stripe is a company that has an API for credit card transactions. Online companies often use it as a way to charge their customers. Customers will enter credit card information at a company's website, and then the company's servers will make calls to the Stripe API to make those transactions happen. This way, the companies do not need to set up relationships with credit card companies. Stripe is used by a very large number of companies and handles billions of dollars every year.

Stripe pioneered a form of API documentation where the page is divided into three vertical slices: a navigation bar, the documentation, and sample code. The sample code shows multiple languages as tabs that you can switch between. For this exercise, you will only use the curl tab.

There's no reason to think that Stripe uses OAS, but in this exercise, imagine that they do. It would take weeks to create an OAS file for the entire Swagger API, so we are going to just use one piece of it, which is the Tax Codes section resource. The Tax Codes API requests are used to obtain the codes that determine a product's tax rate. I chose this because it's smaller than many of the resources, but it will still allow you to make a real-world example.

The documentation for this endpoint is at: https://docs.stripe.com/api/tax_codes?lang=curl. Bring that up in a webpage so you can refer to it. Also bring up other OAS files that you can reference.

Important! This exercise is based on the Stripe website, which could change at any time if they change their API. ***Please*** notify me if you see something different, or if the links no longer work, etc. You can do this by sending me a message through Udemy or posting it in the Q&A section.

General information

Open up the Swagger editor and from the **File** menu, clear the editor.

1. Start with the Swagger version of 2.0.
2. Now add the **info** section. You will need to find the **version**. Open https://stripe.com/docs/api/versioning?lang=curl and notice the current version. As of the writing of this exercise, that version is **2019-02-19**.
3. For the title, say "Tax Codes".
4. For the description, take the description from https://docs.stripe.com/api/tax_codes?lang=curl. Use a link to https://docs.stripe.com/tax/tax-codes. Use this HTML to indicate the link, placing it where the text says "Tax codes":

```
<a href="https://docs.stripe.com/tax/tax-codes">Tax codes</a>
```

5. Now you are done with the **info** section. Next, add the **host**, **basePath**, and **scheme**. To figure that out, look at one of the Refunds endpoints: https://docs.stripe.com/api/tax_codes/retrieve?lang=curl. To the right, under the Curl tab, you will see an example request. After the word "curl", you can see the URL, which contains the **host**, **basePath**, and **scheme**. As of the writing of this exercise, the host is **api.stripe.com**, the **basePath** is **/v1**, and the **schemes** are just one item: **https**
6. Finally, add **consumes** and **produces**. If you look at the introduction (https://stripe.com/docs/api?lang=curl), you will see that the API accepts form-encoded data and returns JSON. This means that **consumes** should have one item of **application/x-www-form-urlencoded** and **produces** should have one item of **application/json**

Although you will get an error because you don't have path information, you should be able to see the title and description with the link, as well as the base URL, which should mention the Stripe documentation. If you are stuck, you can look at my solution so far at: http://sdkbridge.com/swagger/stripe1.yaml

Security

If you read the Authentication section (https://stripe.com/docs/api/authentication?lang=curl), you will find that authentication is performed using BasicAuth. So you will need to create a security definition which you can call **basicAuth** that has a type of **basic**.

Add that to the bottom of your OAS file.

Finally add a security section under the produces section so that we use the basicAuth definition:

```
security:
  - basicAuth: []
```

The Tax Code object

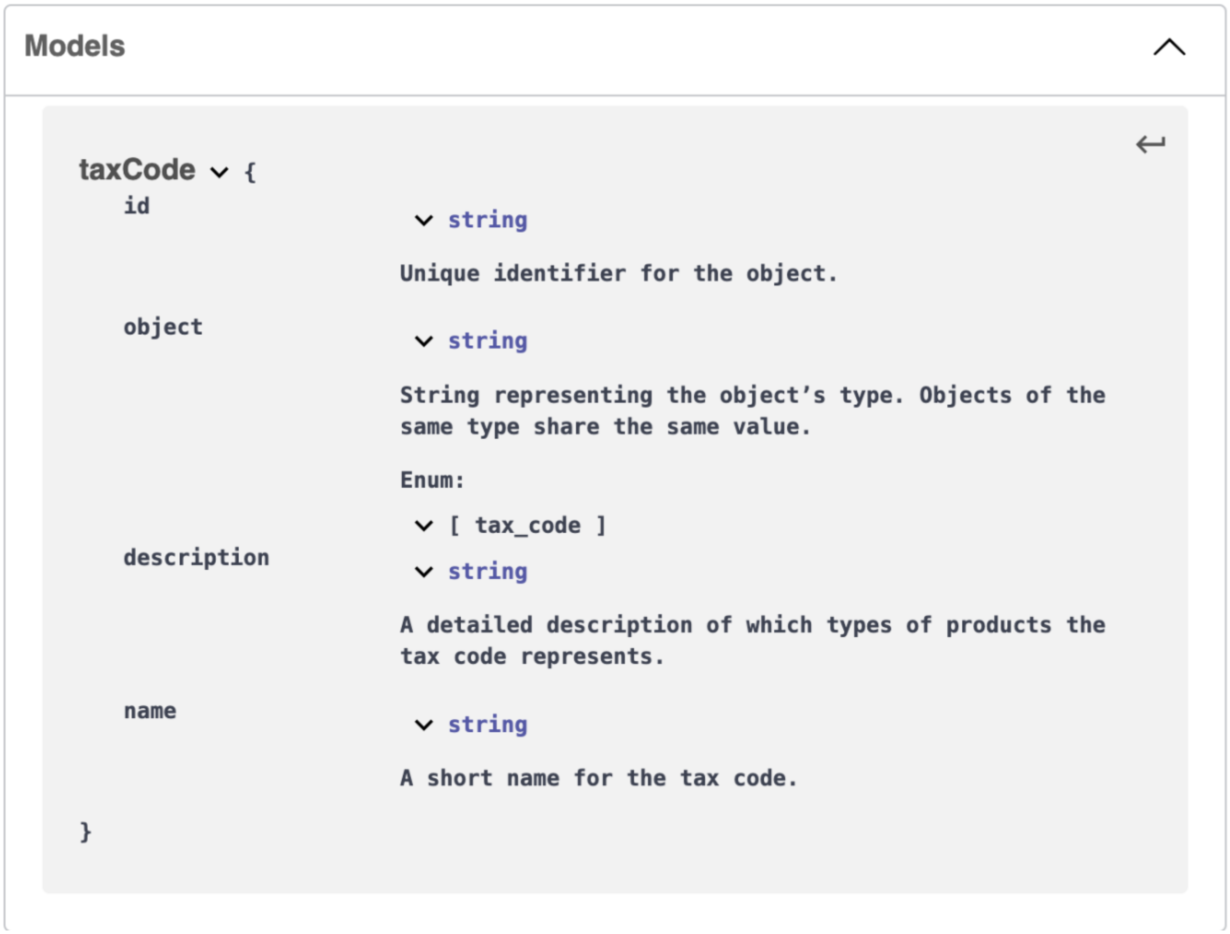
The first part of the Tax Code section of the documentation is the tax code object (https://docs.stripe.com/api/tax_codes?lang=curl). We're going to use some advanced OAS features that I didn't cover to make it match the Stripe API better. To add it to your OAS file:

1. Add a **definitions** section above the **securityDefinitions** key.
2. Within that, add an object called **taxCode**.
3. Within that, add a **properties** key, indented.
4. Look at the documentation at the link above, and you will see there are 4 properties to add: **id**, **object**, **description**, and **name**. Add an **id** property of type **string**, and for the **description**, copy the text from the documentation. ("Unique identifier for the object.")
5. Add an **object** property. For type, use string. Again, take the description text from the documentation. Note that object always has a value of "tax_code". There are two ways you can handle this. You can simply add the following sentence at the end of the description: value is "tax_code". Or you can add an "enumeration", which lists the valid values it can have. In this case, there is only one valid value, which is "tax_code".

```
          object:
            type: string
            description: String representing the object's type.
    Objects of the same type share the same value.
            enum:
              - tax_code
```

6. Add a **description** property of type string. Take the description text from the documentation.
7. Do the same for the **name** property.

On the right side of the editor, you will still see an error because we not put in the **paths** key yet. But if you scroll down to the bottom, you will see your model, which should look like this once you open all the sections:


Models

```
taxCode ∨ {
   id              ∨ string
                   Unique identifier for the object.
   object          ∨ string
                   String representing the object's type. Objects of the
                   same type share the same value.
                   Enum:
                   ∨ [ tax_code ]
   description     ∨ string
                   A detailed description of which types of products the
                   tax code represents.
   name            ∨ string
                   A short name for the tax code.
}
```

Note that I clicked on **> Array [1]** to see the **[tax_code]**.

If you are stuck, you can look at my solution at: http://sdkbridge.com/swagger/stripe2.yaml

Create a tax code

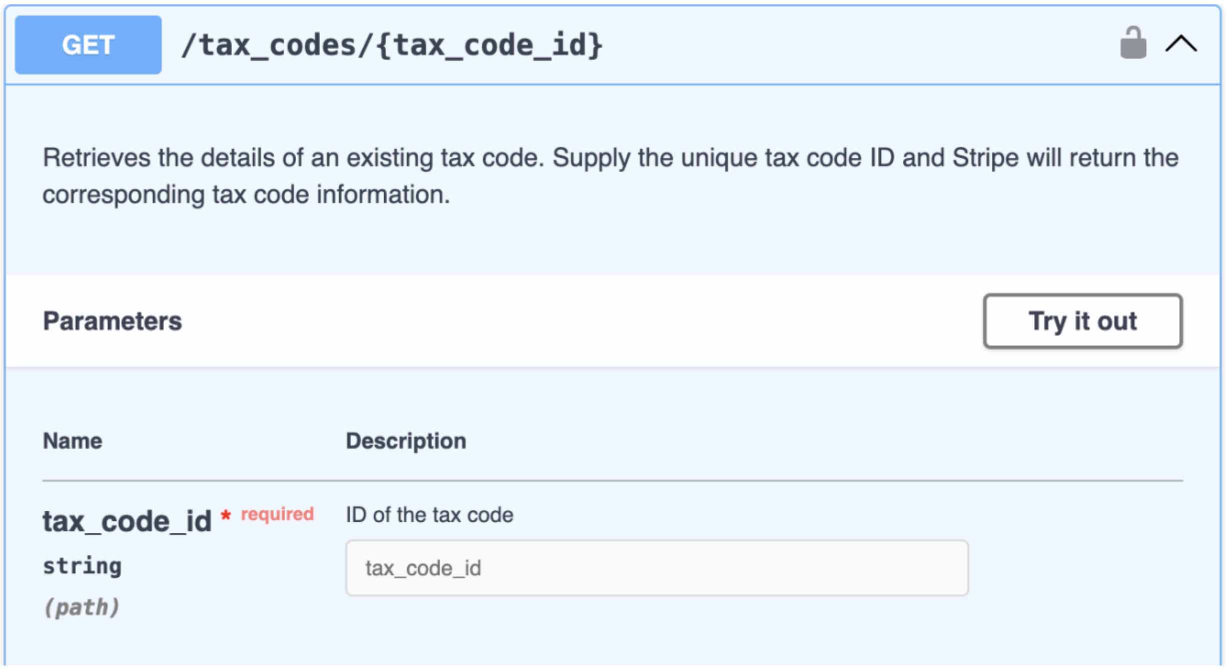
If you keep scrolling down, you will see the first API request, which is to retrieve a tax code. (https://docs.stripe.com/api/tax_codes/retrieve?lang=curl). In this section, you will figure out information from the curl sample section of the documentation. curl is a command-line tool used to make HTTP requests. Follow these steps to add your first path:

1. Under the **security** section, add a **paths** key, like you have done in previous exercises.
2. Look at the right side, under the **curl** tab. Note that the URL is: **https://api.stripe.com/v1/tax_codes/txcd_99999999.** In this case, the **api.stripe.com** is the host, the **v1** is the base URL, and the path is **tax_codes/txcd_99999999**. However, **txcd_99999999** is clearly an ID, not part of the API. Because it follows **tax_codes**, it will be a tax code ID. So the path needs to include that ID. Use this as your path:

   ```
   /tax_codes/{tax_code_id}:
   ```

3. Next, notice that above the curl sample, it says that the method is GET, so add the get method.
4. Copy the description from the documentation.
5. Now add a **parameters** key.
6. Add the one parameter, which is the ID. It will have a **name** of **tax_code_id**, an **in** of **path**, a **required** of **true**, a **type** of **string**, and a **description** of **ID of the tax code.**
7. Finally, add the response by adding a **responses** section under the **parameters** section of the **post** section. Then add a key of **200** and a description of **Successful response**. It returns a tax code, so add a **schema** key and a **$ref** that points to that object.

At this point, your errors should be totally gone. If you have errors, read them and see if you can figure them out. Often it's indentation problems, or something in the wrong section. This is what your documentation should look like for the GET:



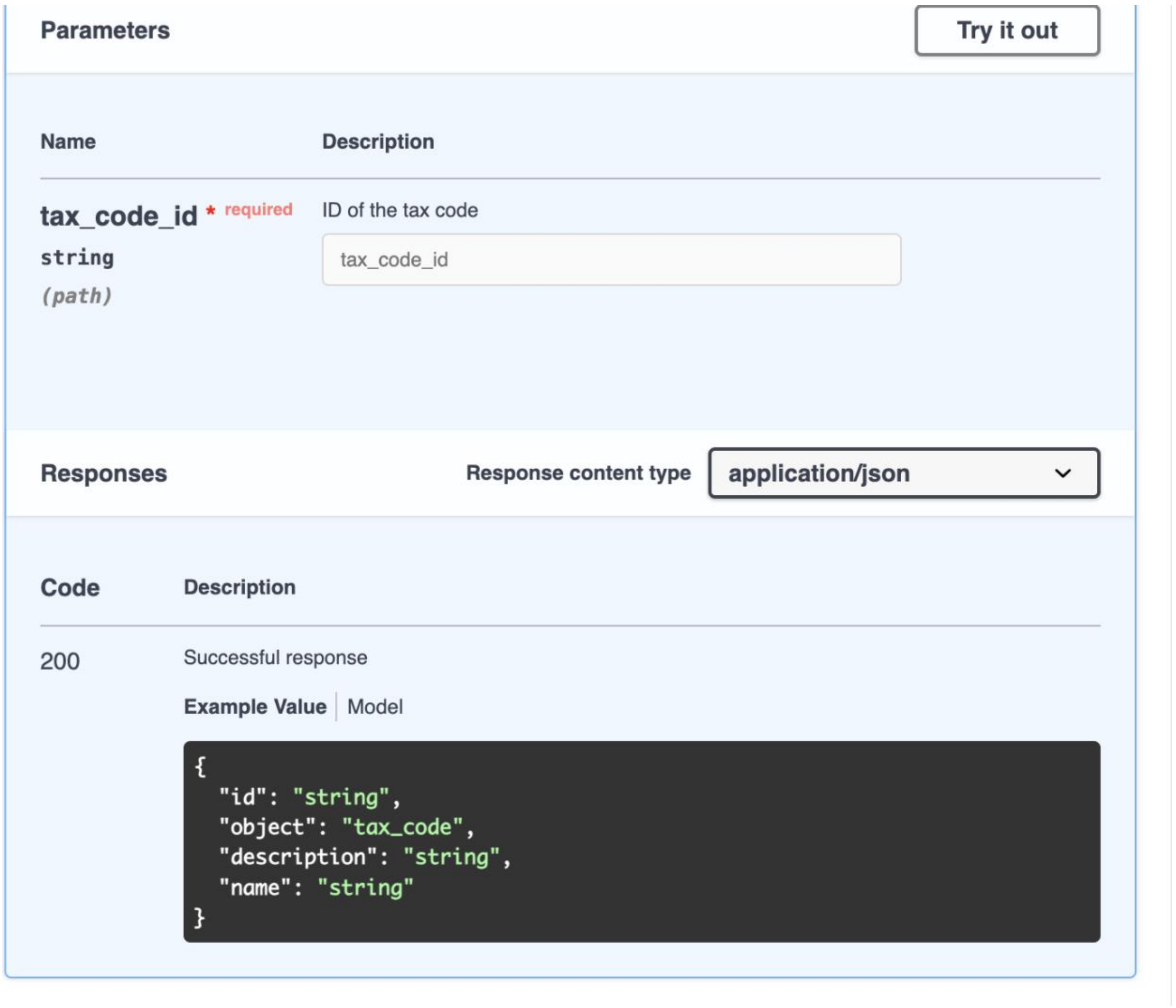
/tax_codes/{tax_code_id}

Retrieves the details of an existing tax code. Supply the unique tax code ID and Stripe will return the corresponding tax code information.

Parameters

Name	Description
tax_code_id * required string (path)	ID of the tax code

tax_code_id


Parameters

Try it out

Name	Description
tax_code_id * required string (path)	ID of the tax code tax_code_id

Responses

Response content type: application/json

Code	Description
200	Successful response

Example Value | Model

```
{
  "id": "string",
  "object": "tax_code",
  "description": "string",
  "name": "string"
}
```

If you click on **Try it out**, you'll see that you have a place to enter the tax code ID. Note the red asterisk indicating that the parameter is required. At the top, you will find an authorization section.

Schemes
HTTPS
Authorize

Click on **Authorize**, and it will bring up a dialog to enter basic authentication credentials (i.e., username and password). Of course, this won't work because you don't have a Stripe account.

If you need help, you can look at my solution so far at: http://sdkbridge.com/swagger/stripe3.yaml

List all tax codes

Scroll down again, and you will see the second request, which is **List all tax codes**. To understand what some of the parameters mean, read the section on Pagination: https://stripe.com/docs/api/pagination?lang=curl

Follow these steps to create the second request:

1. Copy everything from your first path (**/tax_codes/{tax_code_id}**) and paste it below that section to create the second path.
2. To get the path name, look at the URL of curl sample request. It has the same host and base URL, so we just need everything between that and the question mark. Rename the path to be **/tax_codes**
3. This is another GET method, so leave that as is.
4. Change the description to match the documentation. Use the HTML link tag to link to the available tax codes page, like we did previously.
5. Look under **Parameters** in the documentation to find the parameters. Add an **ending_before** parameter with an **in** of **query**, a **required** of **false** (it's optional), and a **type** of **string**. Finally, copy the description from the documentation.
6. Add a **limit** parameter. You should be able to figure out its **in**, **required**, and **type**. In addition, you can add a **maximum** and **minimum**.
7. Copy the **ending_before** parameter and add it to the end of the parameters list. Change its name to **starting_after** and update its description. (Everything else is the same.)
8. For the 200 response, copy the Returns information in the **description**.
9. Also for the response, you will need to create a new object in the **definitions** section. Let's call it **taxCodeList**. First change the schema to

```
$ref: '#/definitions/taxCodeList'
```

10. Now create it in the **definitions** section. For the **description**, use the documentation.
11. The best way to figure out what goes into this object is to look at the curl example. There are four properties: **object**, **url**, **has_more**, and **data**.
12. **object** is a **string** with a value of "list". You can copy the object from **taxCode**, but change the **enum** value.
13. **url** is a **string** that refers to the URL of these tax code. So have a **description** of **URL of the tax codes.**
14. **has_more** is a **boolean** that is described here: https://stripe.com/docs/api/pagination?lang=curl#pagination-has_more
15. **data** should refer to an array of new objects. You can tell it's an array because of the square brackets in the sample. This means that it will be of type **array**, and the times will be objects of type taxCode.

Here is my solution for the final document: http://sdkbridge.com/swagger/stripe4.yaml