API Testing

This document describes the steps to perform manual testing of a specific API in an API file.

There can be a lot to consider in testing, such as the operating system, network connectivity, different configurations options, etc. For this section of the documentation, we are only focusing on testing the API Logic.

To do thorough testing, you should examine the API logic and look for different code pathways.

Then, determine all the input parameter combinations that can be used to test each one of those pathways.

1. Navigate to the Profound.js IDE

2. Open the API file

3. At the bottom right, change to the **Test** tab

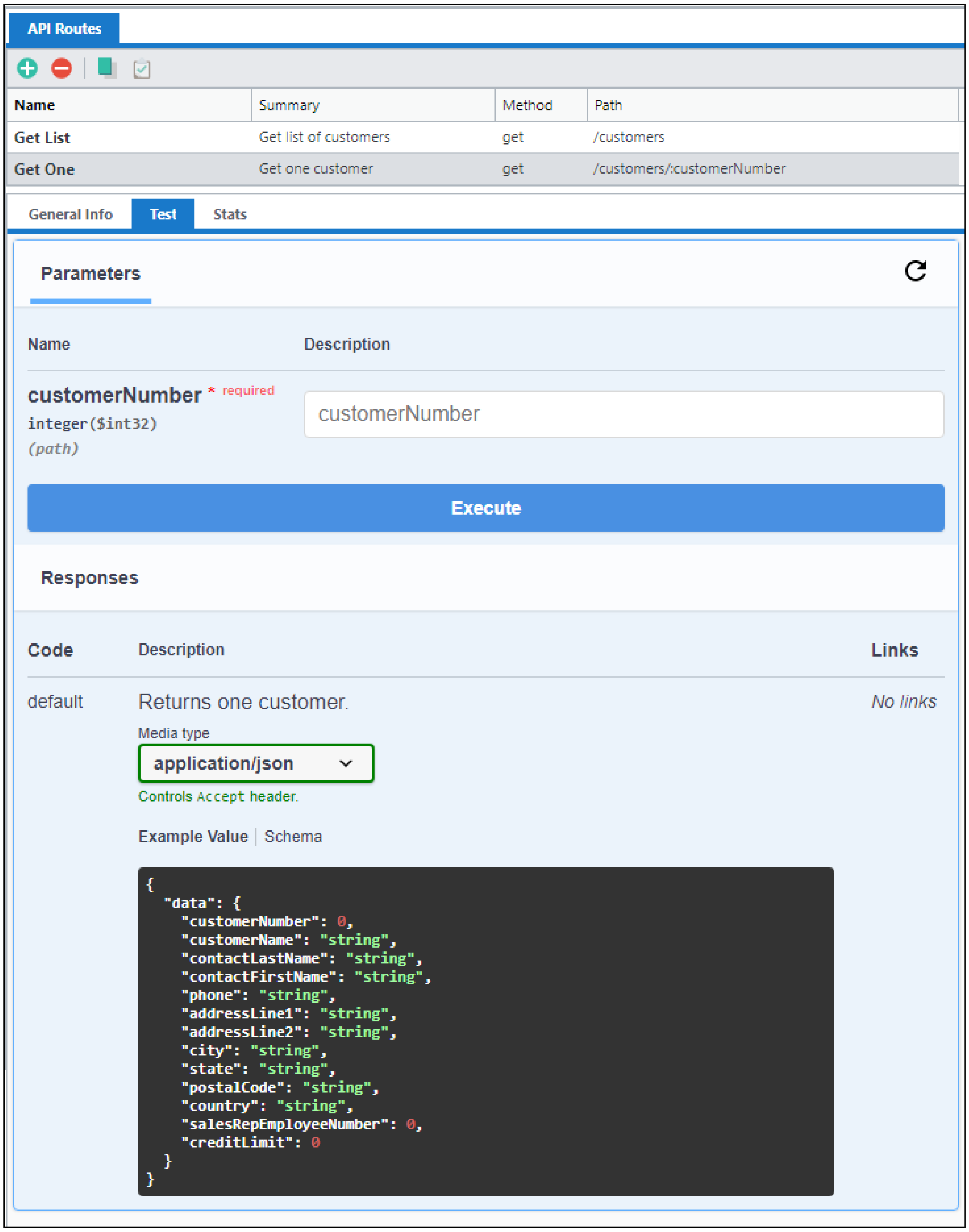
- If you are experienced with Swagger UI, then this tool will be easy to pick up.
- This tab shows the following:
  - **Parameters** section, which includes all of the input parameters along with the associated documentation and validation
  - **Responses** section, which includes the output documentation and output parameters

Below is an example of a customer.api.json file, with the **Get one customer** API selected.

From this test view, we can gather that this API:

- Has one **path** parameter named customerNumber that is required and must be an integer
  - You can also see this path parameter within the HTTP Path (this is an example of RESTful URI naming)
- It returns a JSON object that contains a property called **data**. And the data property contains several more properties of different types.

To Test the API, you would type in a valid customer number and press the Execute button.



Name	Summary	Method	Path
Get List	Get list of customers	get	/customers
Get One	Get one customer	get	/customers/:customerNumber

Name	Description
customerNumber * required integer($int32) (path)	customerNumber

Code	Description	Links
default	Returns one customer.	No links

Media type: application/json
Controls Accept header.

```
{
  "data": {
    "customerNumber": 0,
    "customerName": "string",
    "contactLastName": "string",
    "contactFirstName": "string",
    "phone": "string",
    "addressLine1": "string",
    "addressLine2": "string",
    "city": "string",
    "state": "string",
    "postalCode": "string",
    "country": "string",
    "salesRepEmployeeNumber": 0,
    "creditLimit": 0
  }
}
```

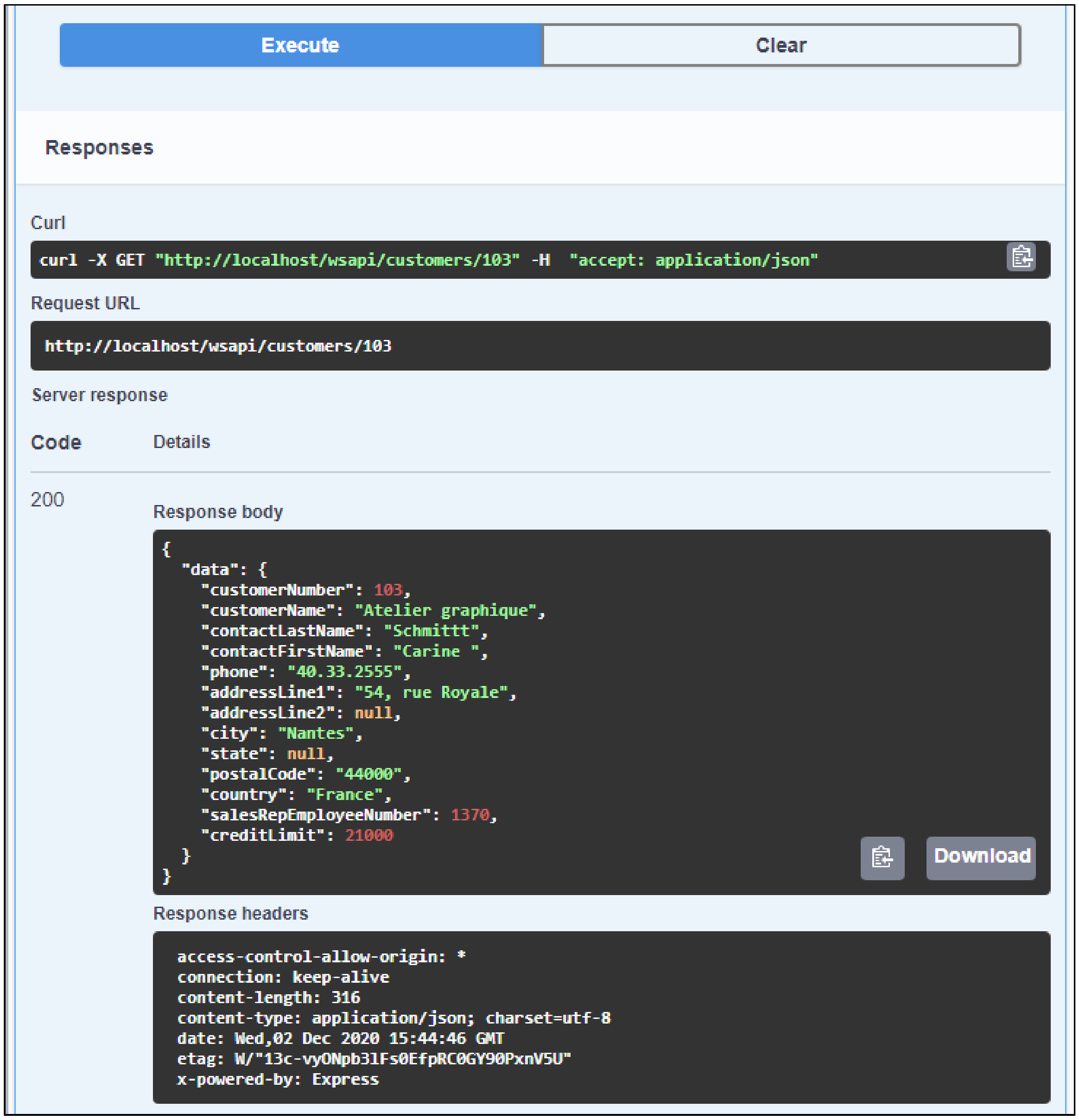
The **Execute** function validates the input parameters and then will attempt to call your API logic.

You will see a *loading* spinner while it calls your API.

Once the spinner goes away, this **Responses** area will be updated with the actual results.

You will see a Curl request, a Requested URL, and the actual Server response Status Code and Body.

You will notice that the Response body is not an exact match to the documented output parameters, because we're now looking at live response data:



Responses

Curl

```
curl -X GET "http://localhost/wsapi/customers/103" -H  "accept: application/json"
```

Request URL

```
http://localhost/wsapi/customers/103
```

Server response

Code Details

200

Response body

```
{
  "data": {
    "customerNumber": 103,
    "customerName": "Atelier graphique",
    "contactLastName": "Schmittt",
    "contactFirstName": "Carine ",
    "phone": "40.33.2555",
    "addressLine1": "54, rue Royale",
    "addressLine2": null,
    "city": "Nantes",
    "state": null,
    "postalCode": "44000",
    "country": "France",
    "salesRepEmployeeNumber": 1370,
    "creditLimit": 21000
  }
}
```

Response headers

```
 access-control-allow-origin: *
 connection: keep-alive
 content-length: 316
 content-type: application/json; charset=utf-8
 date: Wed,02 Dec 2020 15:44:46 GMT
 etag: W/"13c-vyONpb31Fs0EfpRC0GY90PxnV5U"
 x-powered-by: Express
```