API Performance

Some APIs that you develop will have strict performance requirements.

With integrated API Statistics and a mass API test tool you can discover potential bottlenecks within your business logic, database indexes, and even application infrastructure.

Performance testing is similar to manual testing, with a key difference of using a tool to call your API in mass and under different scenarios.

Load testing and stress testing are other types of tests you can perform to find the benchmarks of your application infrastructure.

A useful tool for stress testing and determining benchmarks is Artillery, which allows mass calling of your APIs along with configurable scenarios.

Artillery is a free node package that you install on your local computer like this:

- npm install -g artillery

It can be used in a variety ways. Here are a couple quick ways to use it from a terminal:

- artillery quick --count 20 --num 100 http://pjsserver/wsapi/customers/103

  This will start 20 connections (sessions), and on each connection it will call your API 100 times, 1 at a time.

  This is a good example for performance testing, and can tell you the overall throughput of your API

- artillery quick --rate 20 --duration 60 http://pjsserver/wsapi/customers/103

  This will run for for 60 seconds, and each second 20 API calls are attempted

  This is good example for load or stress testing and can tell you how well this API will stand up under pressure.

The Artillery output provides metrics on these tests, which include:

- Scenarios launched and completed
- Response times in different forms: min, max, mean, median, top 95 and 99 percentiles
- Response status codes

With the mass tests running or finished, you can take a look at the API statistics:

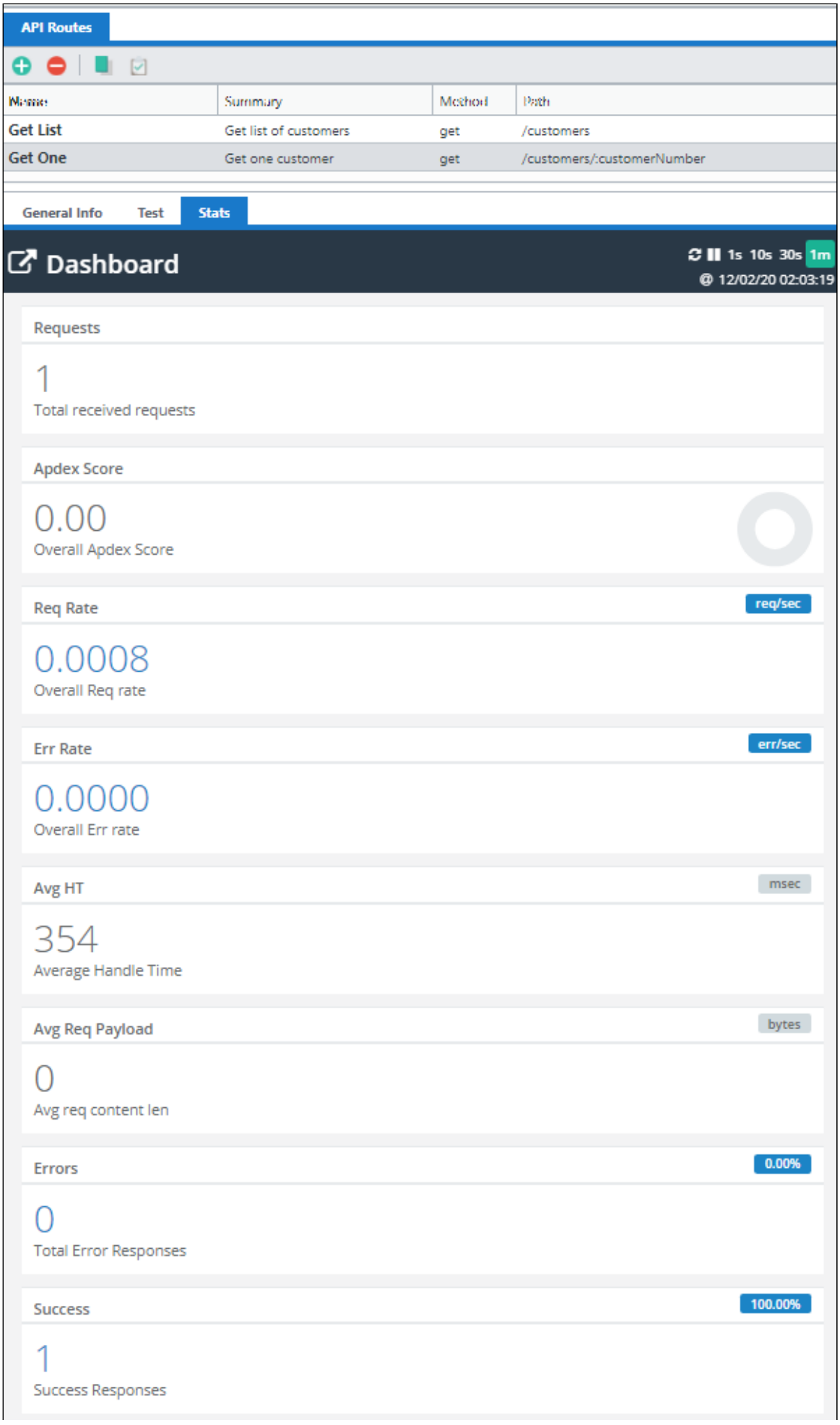
1. Navigate to the Profound.js IDE
2. Open the API file
3. At the bottom right, change to the **Stats** tab
   To view all of your API globally, you can use the API Dashboard, it includes all of this plus much more.

> At the top-right corner of this tab, there are refresh buttons. They allow you to refresh now, pause auto-refresh, auto-refresh every 1 second, 10 seconds, 30 seconds and every minute.
>
> These statistics are collected automatically through the Profound.js service.

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

From this test view, you can see a lot of information. Some of the interesting statistics include:

- **Requests** - Total number of requests
- **Errors** - Total number of errors
- **Average Handle Time** - Average time the server took to respond to each request over the last hour
- **Handle Time Histogram** - A graph showing how this API has been responding over the last hour
- **Overall Req rate** - Requests per second since the start of the Profound.js service
- **Overall Err rate** - Errors per second since the start of the Profound.js service


API Routes
Name
Summary
Method
Path
Get List
Get list of customers
get
/customers
Get One
Get one customer
get
/customers/:customerNumber
General Info
Test
Stats
Dashboard
1s 10s 30s 1m
@ 12/02/20 02:03:19
Requests
1
Total received requests
Apdex Score
0.00
Overall Apdex Score
Req Rate
req/sec
0.0008
Overall Req rate
Err Rate
err/sec
0.0000
Overall Err rate
Avg HT
msec
354
Average Handle Time
Avg Req Payload
bytes
0
Avg req content len
Errors
0.00%
0
Total Error Responses
Success
100.00%
1
Success Responses

Redirect 0.00%

0

Redirect Responses

Client Error 0.00%

0

Client Error Responses

Server Error 0.00%

0

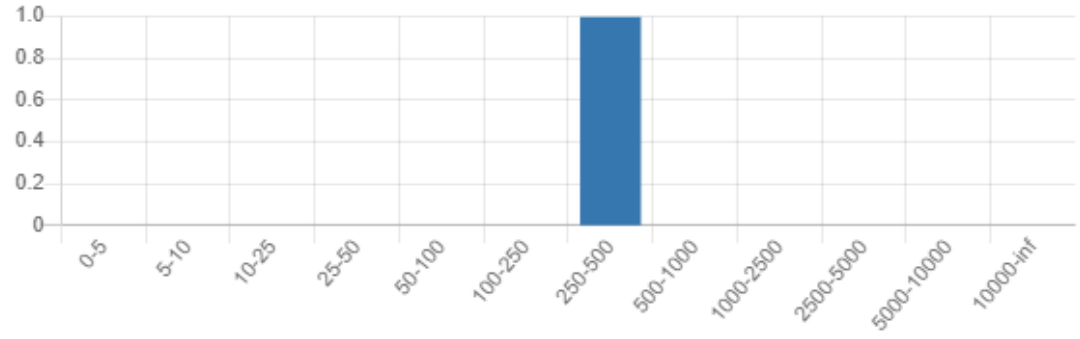
Server Error Responses

Avg Res Payload bytes

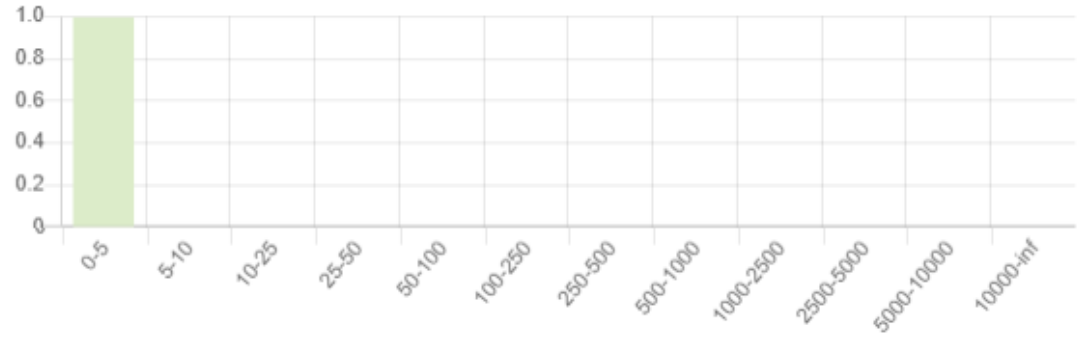
316

Avg res content len

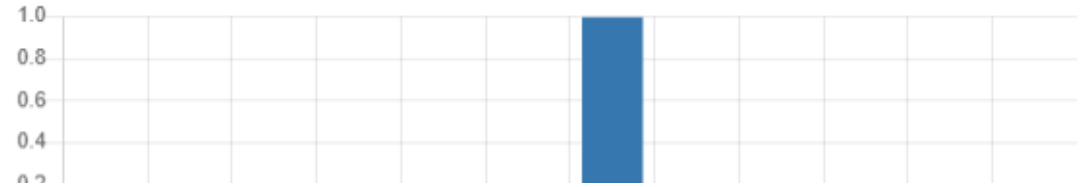
Handle Time Histogram (msec)

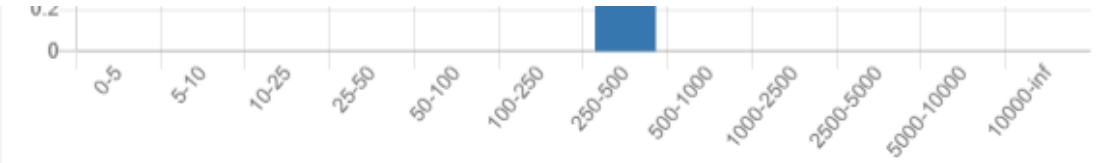


Request Size Histogram (bytes)

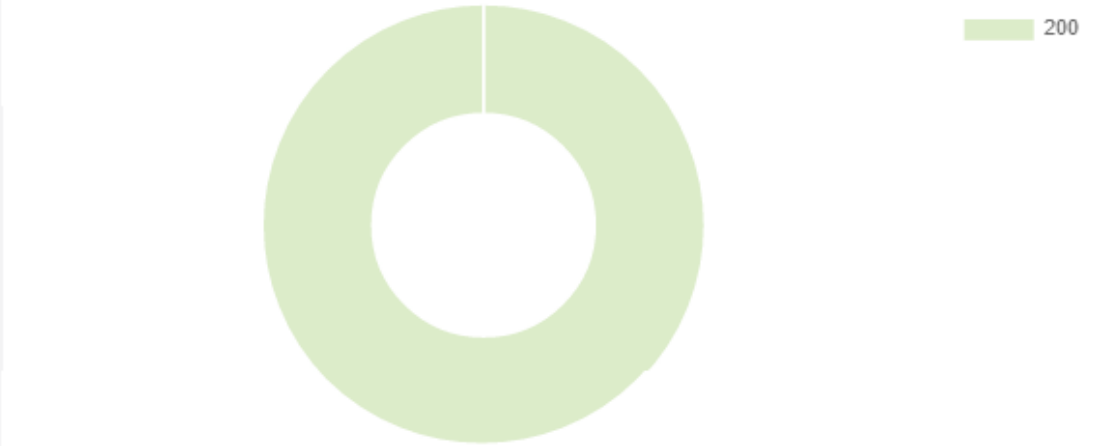


Response Size Histogram (bytes)





Response Codes



Parameters

Copy CSV Column visibility

Show 25 entries

Search:

Showing 1 to 1 of 1 entries

	Name	In	Hits	Misses	Type	Format	Required	Descrip
▸	customerNumber	path	1	0			true	

Previous 1 Next