**PDMG SSP**

*Programmatic Digital Gateway*

*Supply-Side Platform*

PDMG\_SSP

# Documentation

Version 1.1

08/21/2024

**Table of Contents**

**1.** [**Introduction…………………………………………………………………………………3**](#_Introduction)

**2.** [**Structure…….……………………………………………………………………………….4**](#_Structure)

**3.** [**Setup Environments……………………………………………………………………….5**](#_Setup_Environments)

**4.** [**Best Practices…...………………………………………………………………………….6**](#_Best_Practices)

**5.** [**Running Unit Test………………………………………………………………………….7**](#_Running_Tests)

**6. [Test RESTful APIs using Postman…..………………………………………………..16](#_Test_RESTful_APIs)**

**7.** [**APIs..………………………………………………………………………………………..17**](#_APIs)

**8.** [**High Traffic Test using Apache Bench………………………………….……………29**](#_High_traffic_testing)

## Introduction

*PDMG\_SSP SUMMARY.*

*PDMG\_SSP (Programmatic Digital Media Gateway – Supply Side Platform) is a comprehensive Supply Side Platform designed to manage and optimize ad inventory for digital publishers. It supports the OpenRTB (Real\_Time Bidding) protocol, enabling publishers to connect with multiple demand channels (DSPs – Demand Side Platform), and maximize their ad revenue through programmatic advertising. The SSP handles bid requests, processes bids, selects the winning bid based on predetermined criteria, records the results for future daily and monthly reporting and optimization.*

This documentation provides comprehensive technical documentation for the restify-based Supply-Side Platform project and provides guidelines and instructions for running and maintaining tests including unit tests, tests using postman file and high traffic testing using Apache Benche for the ‘pdmg\_ssp’ project. This project uses Mocha as the test framework, Chai for assertions, and Sinon for mocking and stubbing.

### Key Features

* **Auction Handling:** The cord functionality of the SSP is to manage real-time bidding (RTB) auction based on OpenRTB 2.6 protocol. The platform efficiently handles auction requests, processes bid responses, and determines the winning bid based on the highest or second highest bid price.
* **Private Marketplace (PMP):** The SSP supports Private Marketplace deals, enabling publishers to offer premium inventory to select buyers at negotiated rates. These deals ensure that the inventory is only accessible to specific DSPs, maintaining exclusivity and allowing for higher CPMs (Cost Per Mile).
* **Extensible API Endpoints**: The platform offers well-structured API endpoints for handling different aspects of the auction process, including managing publishers, devices, endpoints, deals, reporting and initiating auctions.
* **Daily and Monthly Reporting**: The SSP provides daily and monthly reporting to publishers on auction performance, including metrics such as total bids, total wins, win prices, and more along with csv file. This information is critical for publishers to optimize their inventory and maximize revenue.
* **Error Handling and Logging**: Integrated logging using `morgain` and `winston` ensures that all server activities are logged, and any errors encountered are properly handled and reported.
* **Scalable Architecture**: The SSP is designed with scalability in mind, allowing it to handle a large number of concurrent auction requests with minimal latency.

## Structure

***Project structure is as follows:***

pdmg\_ssp/

├── config/

├── controllers/

├── crons/

├── engine/

├── libs/

├── middlewares/

├── model/

├── openrtb/

├── reports/

├── services/

├── tempates/

**├── test/**

**│ ├── apis/**

**│ │ └──deal.router.test.js**

**│ │ └──device.router.test.js**

**│ │ └──endpoint.router.test.js**

**│ │ └──publisher.router.test.js**

**│ ├── controllers/**

**│ ├── middlewares/**

**│ │ └──validate.middleware.test.js**

**│ ├── libs/**

**│ │ └──methods.test.js**

**│ ├── openrtb/**

**│ │ └──bid\_request.test.js**

**│ ├── db.test.js**

**│ ├── server.test.js**

├── validators/

├── index.js

└── server.js

## Setup Environments

### Install Node.js and npm:

Ensure Node.js and npm are installed on your system. You can download them from nodejs.org. Node version *16.15.1* required.

### Move to the project directory

cd pdmg\_ssp

### Install Dependecies

npm install

### Set up MySQL Database:

Ensure you have MySQL installed and running. Create a MySQL database and update the ‘config’ file with your database credentials.

#### Example MySQL Setup

sudo apt-get update

sudo apt-get install mysql-server

sudo mysql\_secure\_installation

mysql -u root -p

CREATE DATABASE pdmg\_ssp;

## Best Practices

### Code Quality:

* **Modular Design**: Structure your code into modular components, each responsible for a specific functionality. This enhances code readability and maintainability.
* **Error Handling**: Implement consistent error handling across the application using middleware. Ensure that all errors are logged and that appropriate responses are sent to clients.
* **Documentation**: Write clear and concise comments in the code, and maintain up-to-date API documentation.

### Performance Optimization:

* **Database Optimization**: Use indexing and query optimization techniques to enhance database performance. Avoid unnecessary joins and complex queries.
* **Caching**: Implement caching for frequently accessed data to reduce database load and improve response times.
* **Asynchronous Processing**: Use asynchronous programming to handle I/O-bound operations, such as database queries and API calls, to prevent blocking the event loop.

### Security:

* **Input Validation**: Validate all incoming requests to prevent SQL injection, XSS, and other common attacks.
* **Authentication and Authorization**: Implement strong authentication and authorization mechanisms, especially for sensitive endpoints by using Admin access key and publisher security key.

### Scalability:

* **Horizontal Scaling**: Design the application to scale horizontally by adding more instances to handle increased load.
* **Stateless Services**: Keep services stateless where possible, allowing for easier scaling and load balancing.
* **Microservices Architecture**: Consider breaking down the SSP into microservices for better scalability and maintainability.

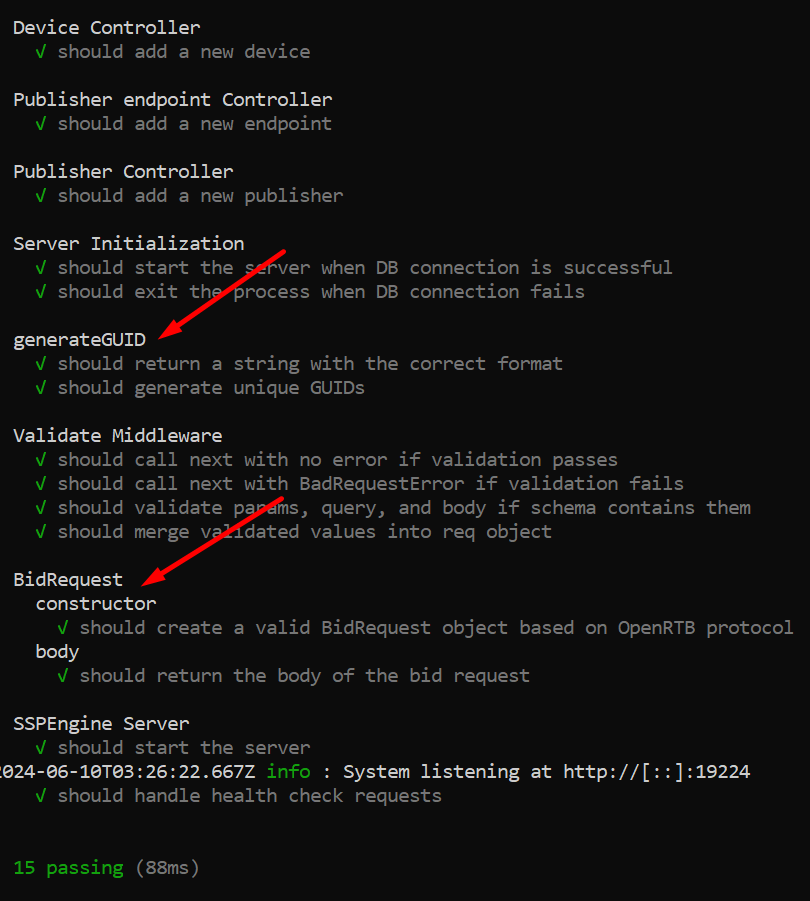
## Running Tests

### Run all tests

* **npm test**

This tests all test files in test folder.

* **Expected result**



### Use npx mocha

You can test just a file or a folder you want to test using `npx mocha` command.

#### Test database connection

* **npx mocha test/db.test.js**

This tests whether your system is connected with MySQL database with your credentials or not.

db.test.js

 ... ... ...

  it('should start the server when DB connection is successful', (done) => {

    connectStub.callsFake((callback) => callback(null));

    DB\_HANDLE.connect(err => {

      assert.strictEqual(err, null)

      done()

    })

  });

  it('should exit the process when DB connection fails', (done) => {

    const error = new Error('DB connection failed');

    connectStub.callsFake((callback) => callback(error));

    DB\_HANDLE.connect(err => {

      assert.strictEqual(err, error)

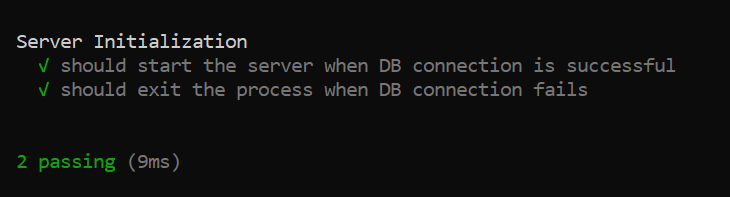
      done()

    })

  });

});

* **Expected result**



#### Test server functionality

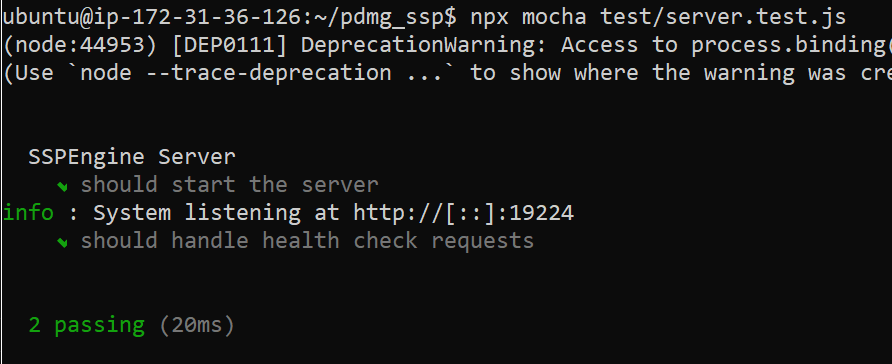
* **npx mocha test/server.test.js**

This tests whether your system is running correctly

server.test.js



* **Expected result**



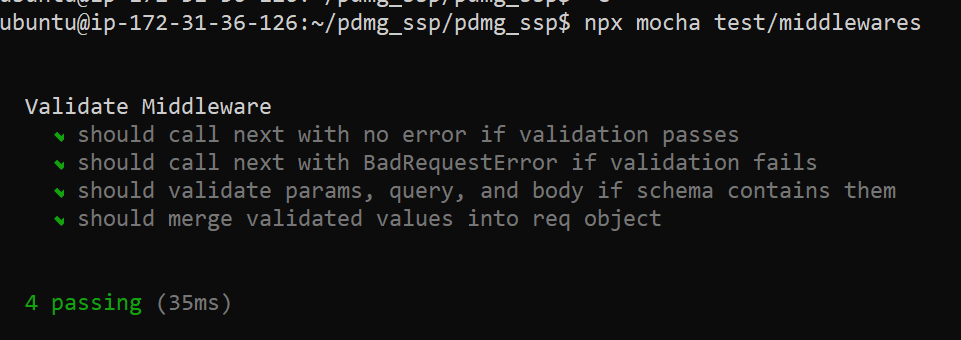
#### Test middlewares

* **npx mocha test/middlewares**

This tests all test files in middlewares’ folder.

(you can test each file: npx mocha test/middlewares/file\_name e.g. validate.middleware.test.js)

**Expected results**

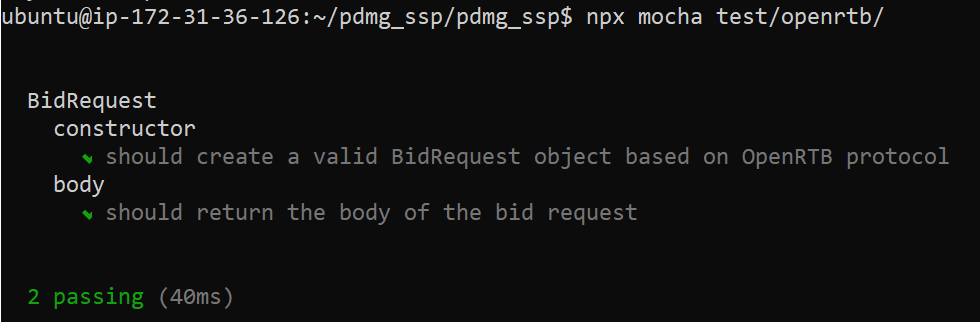


#### Test openrtb support

* **npx mocha test/openrtb/**

This tests whether this system generates correct bid request based on OpenRTB support.

**Expected results**



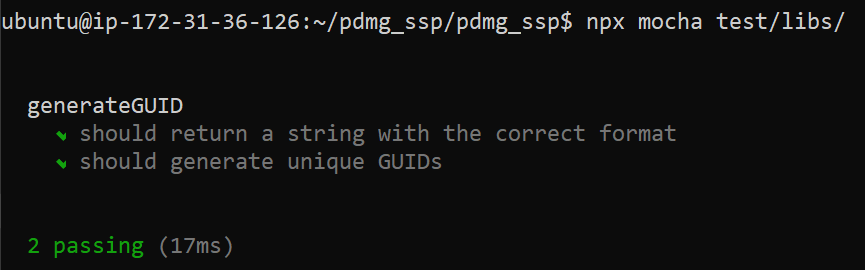
#### Test libs

* **npx mocha test/libs/**

This tests all files in libs folder (which is used as library in this system).

**Expected results**

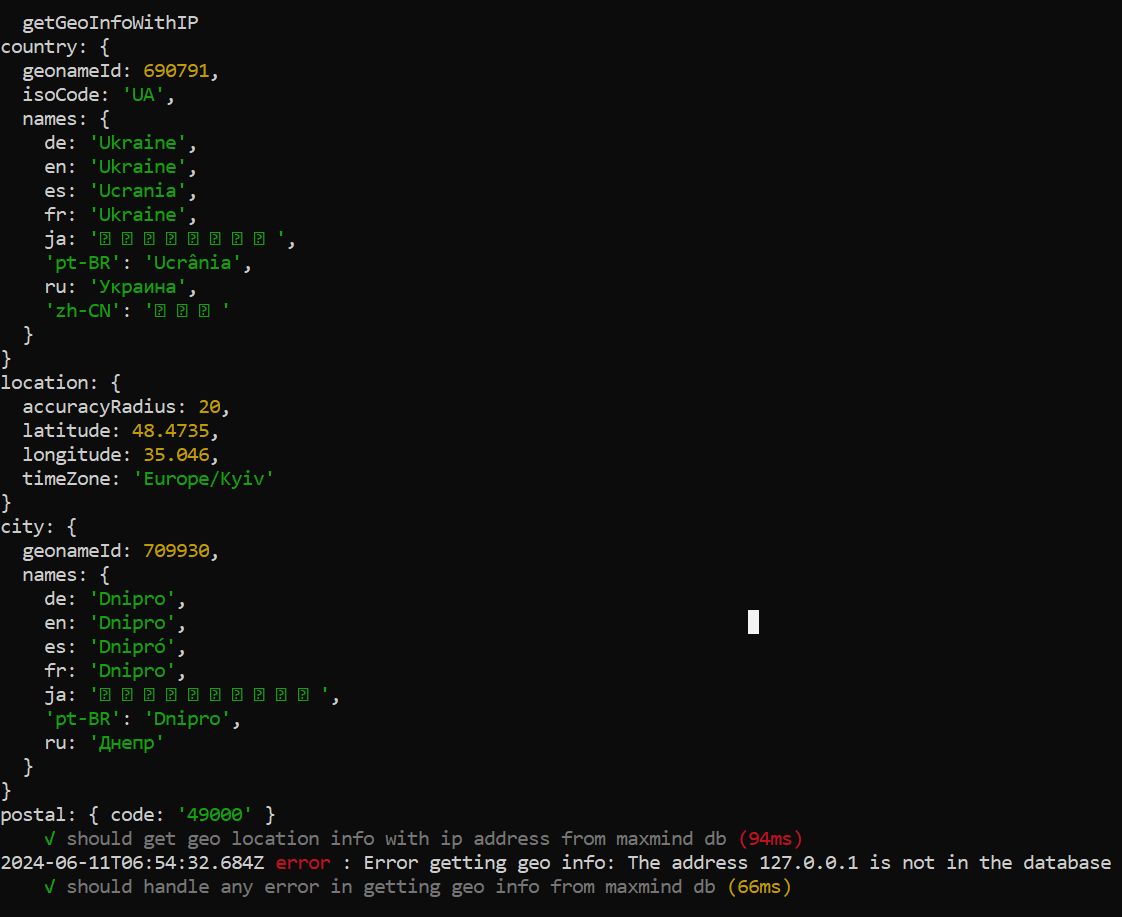
*generateGUID function*



This is a test for the generateGUID function, which generates unique GUIDs that are used as GUIDs such as impression\_id, publisher\_id, device\_id, etc.

This test ensures that the function generates a GUID in the correct format for the GUID and that the generated GUID is unique.

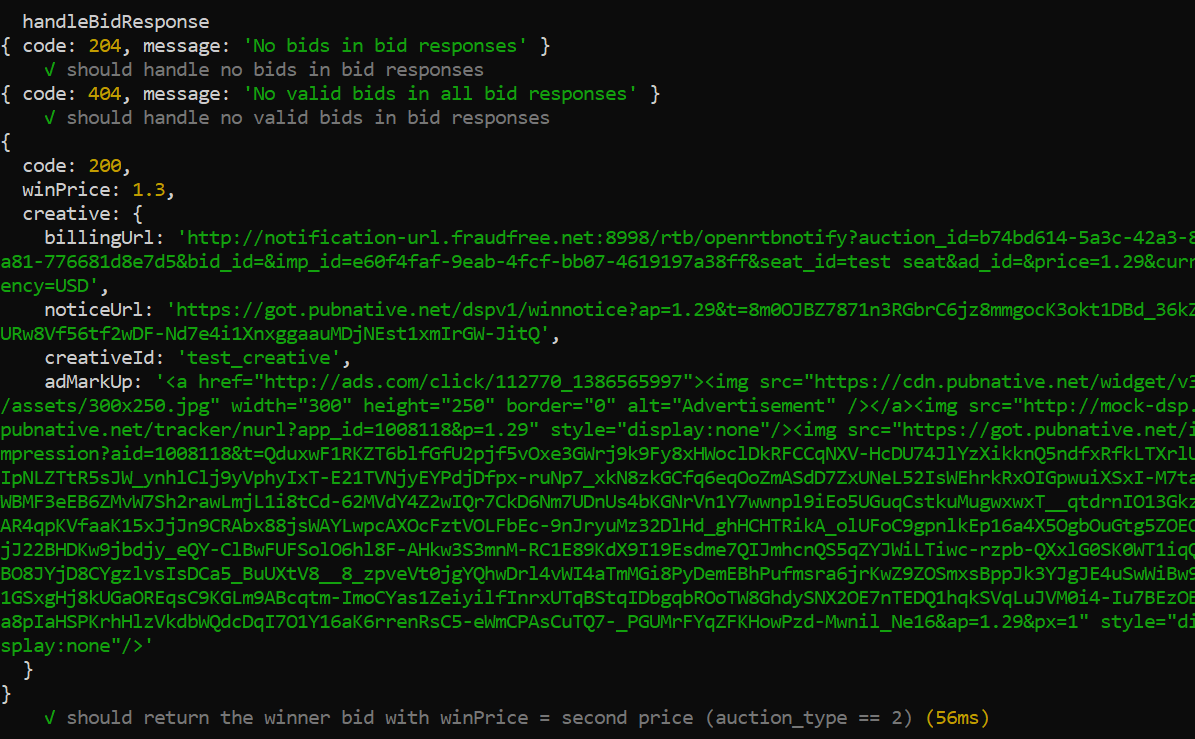
*getGeoInfoWithIp function*



#### Test auction processing

* **npx mocha test/controllers/auction.controller.test.js**

**Expected Results**



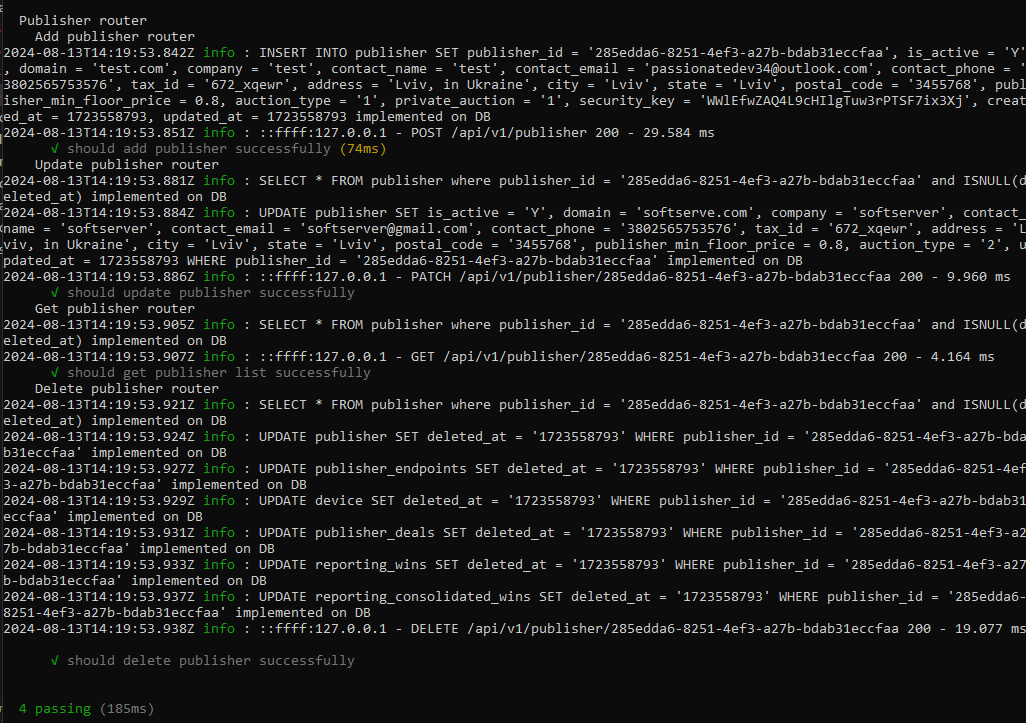


#### Test RESTful APIs

##### Publisher APIs

* **npx mocha test/apis/publisher.router.test.js**

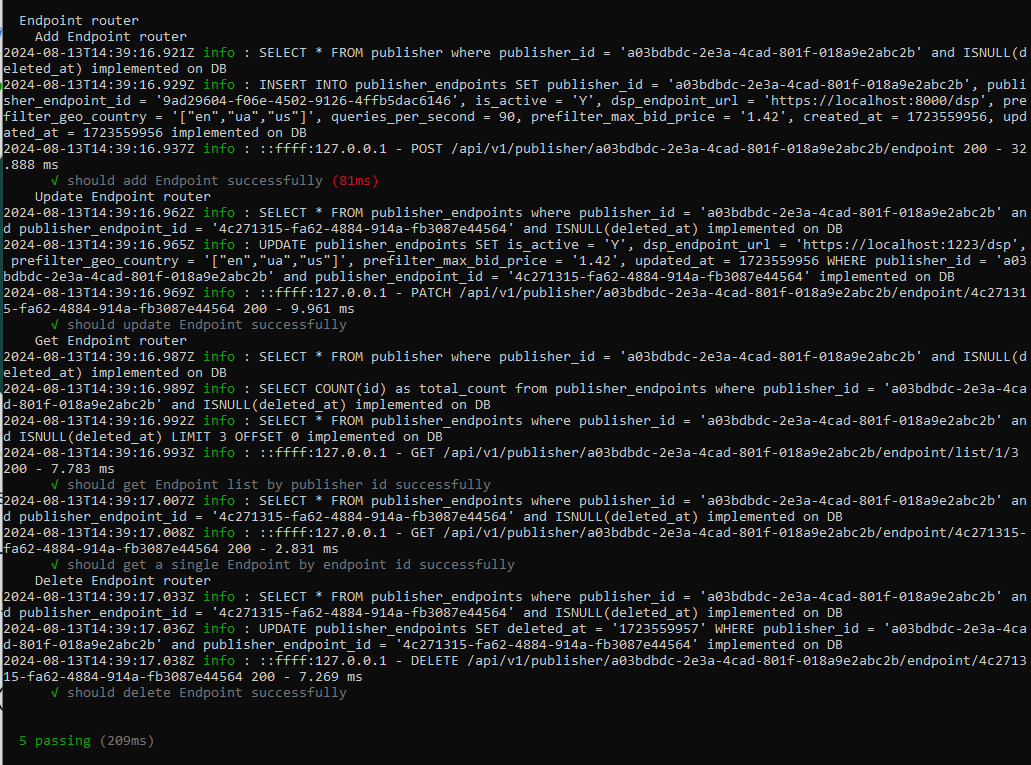
**Expected Result**



##### Publisher Endpoint Apis

* **npx mocha test/apis/endpoint.router.test.js**

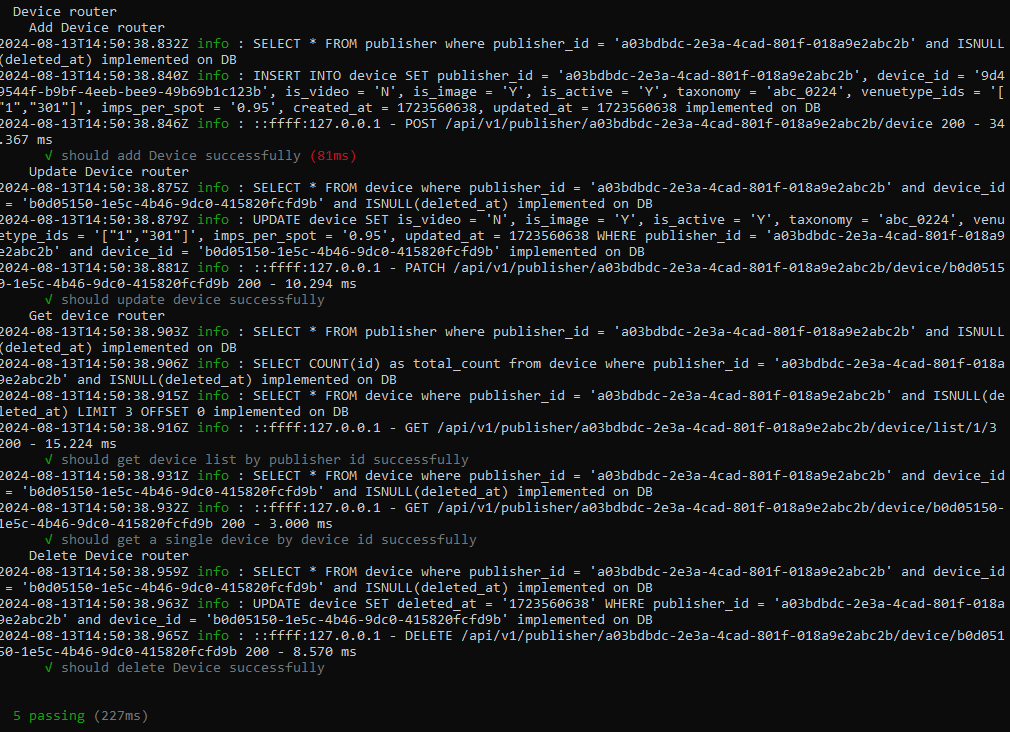
**Expected Result**



##### Device Apis

* **npx mocha test/apis/device.router.test.js**

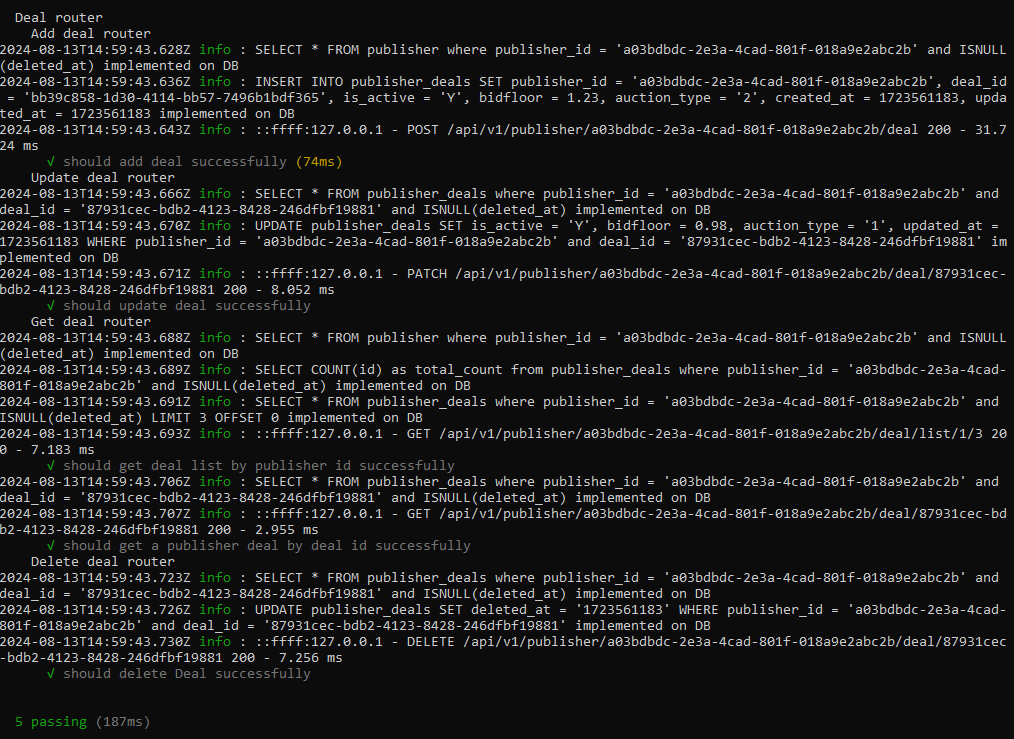
**Expected Result**



##### Publisher deal Apis

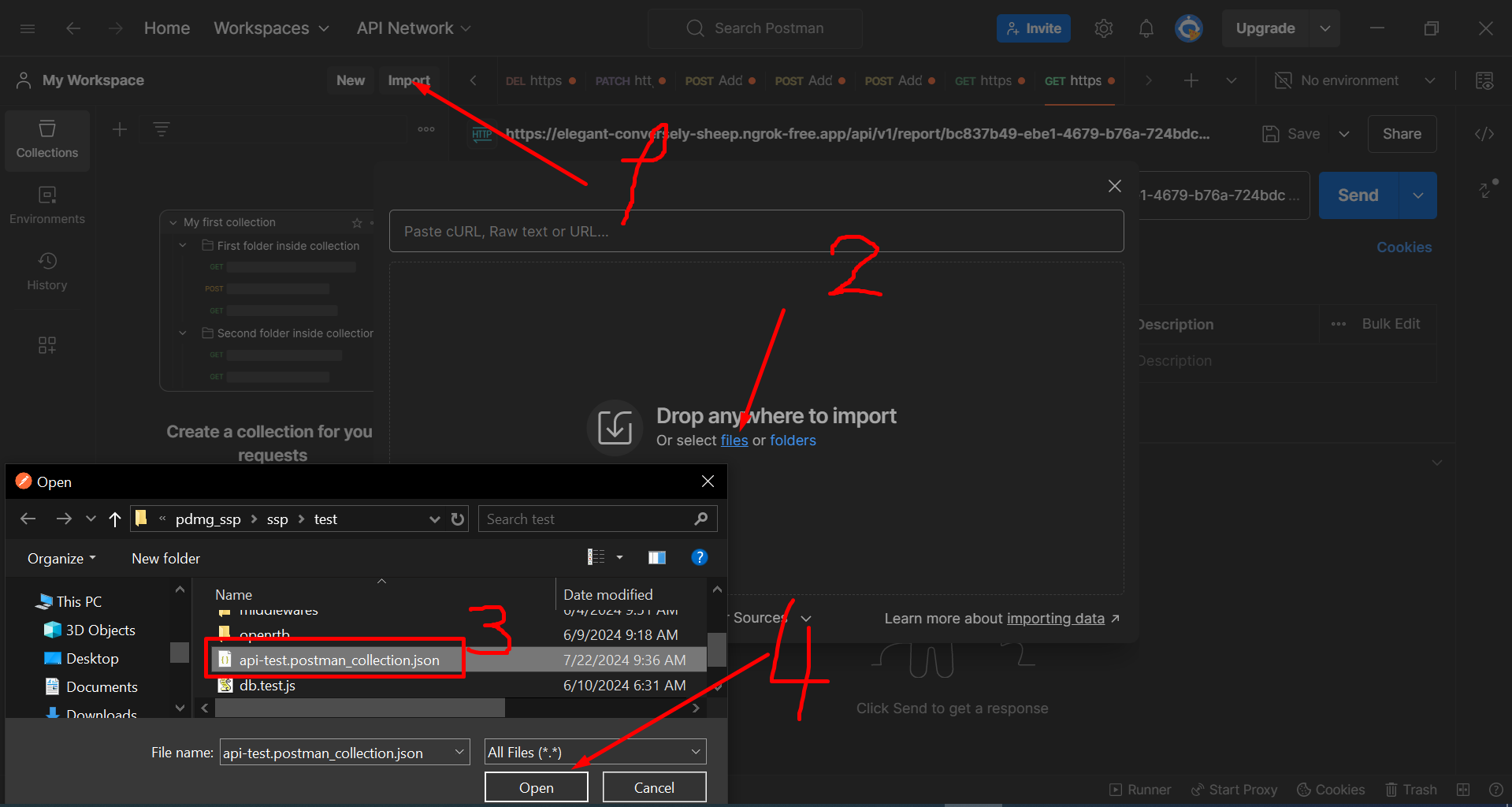
* **npx mocha test/apis/deal.router.test.js**

**Expected Result**



## Test RESTful APIs using Postman

* Open Postman
* Import postman file



* Run the collection

**NOTE:** All APIs MUST include access key (which can be either Admin key or API key for a specific publisher) at headers info.

*Explanation for each API is as next Chapter.*

## APIs

### Publisher APIs

#### Add publisher – add a new publisher with admin access and retrieve publisher id and access key of the publisher.

* router: [/api/v1/publisher](https://aardvark-tight-quietly.ngrok-free.app/api/v1/publisher)
* Method: POST
* Headers: [{ “x-api-key”: “Admin access key” }]
* body: {

"isActive": "Y",

"domain": "softserve.com",

"company": "softserver",

"contactName": "softserver",

"contactEmail": "passionatedev34@outlook.com",

"contactPhone": "3802565753576",

"taxId": "672\_xqewr",

"address": "Lviv, in Ukraine",

"city": "Lviv",

"state": "Lviv",

"postalCode": "3455768",

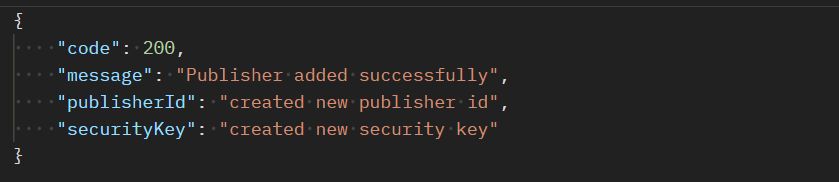
"publisherMinFloorPrice": "0.8",

"auctionType": "1",

"privateAuction": 1

}

* Successful Response



***\* The security key can be used in all apis just for the publisher on our system.***

#### Update publisher – update a publisher by publisher id

* router: [/api/v1/publisher/:publisherId](https://aardvark-tight-quietly.ngrok-free.app/api/v1/publisher/7dec9fef-18a2-443b-98de-4f44ea73cb6a)
* method: PATCH
* Headers: [{ “x-api-key”: “Security key of the publisher or Admin access key”}]
* body: {

"isActive": "Y",

"domain": "softserve.com",

"company": "softserver",

"contactName": "softserver",

"contactEmail": "softserver@gmail.com",

"contactPhone": "3802565753576",

"taxId": "672\_xqewr",

"address": "Lviv, in Ukraine",

"city": "Lviv",

"state": "Lviv",

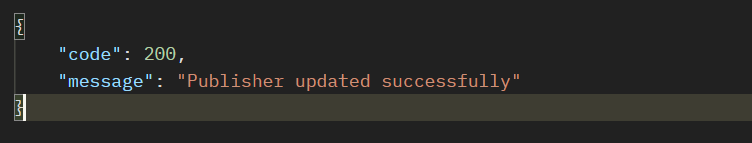
"postalCode": "3455768",

"publisherMinFloorPrice": "0.8",

"auctionType": "2"

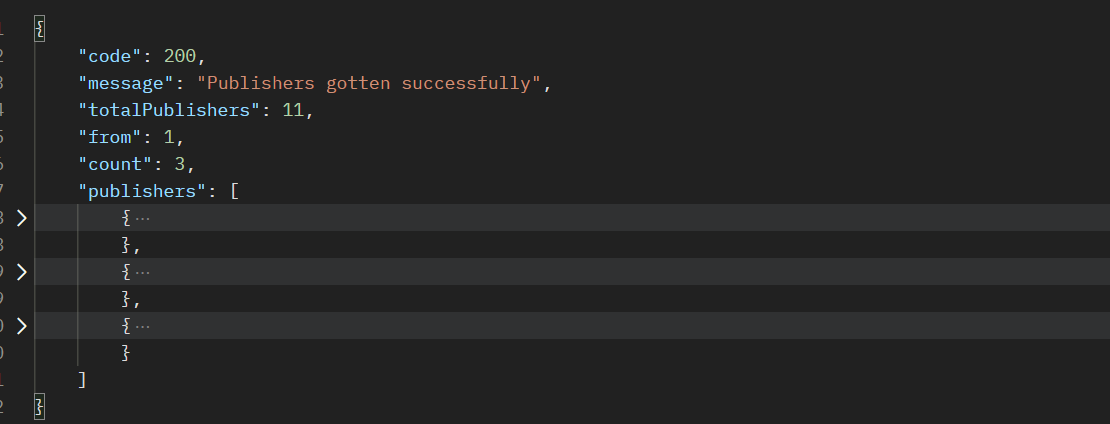
}

* Successful Response



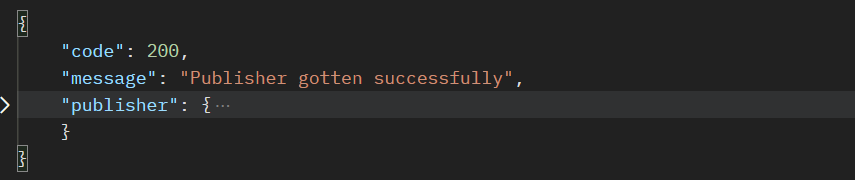
#### Get publisher list – retrieve desired number of publishers from specific position in our database

* router: /api/v1/publisher/list/:start/:limit
* method: GET
* Headers: [{ “x-api-key”: “Security key of the publisher or Admin access key”}]
* Successful Response



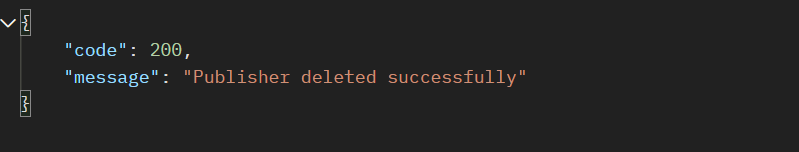
#### Get a publisher – retrieve a publisher by publisher id

* router: /api/v1/publisher/:publisherId
* method: GET
* Headers: [{ “x-api-key”: “Security key of the publisher or Admin access key”}]
* Successful Response



#### Delete a publisher – delete a publisher by publisher id

* router: [/api/v1/publisher/:publisherId](https://aardvark-tight-quietly.ngrok-free.app/api/v1/publisher/7dec9fef-18a2-443b-98de-4f44ea73cb6a)
* method: DELETE
* Headers: [{ “x-api-key”: “Security key of the publisher or Admin access key”}]
* Successful Response



### Publisher Endpoint APIs

#### Add endpoint – add a publisher endpoint for a specific publisher and retrieve the endpoint id

* router: [/api/v1/publisher/:publisherId/endpoint](https://aardvark-tight-quietly.ngrok-free.app/api/v1/publisher/719d3dcd-e6ee-45d9-a8ec-7fc5f40a73c0/endpoint)
* Method: POST
* Headers: [{ “x-api-key”: “Security key of the publisher or Admin access key”}]
* body: {

"dspEndpointUrl": "http://localhost:8000 /dsp",

"queriesPerSecond": 90,

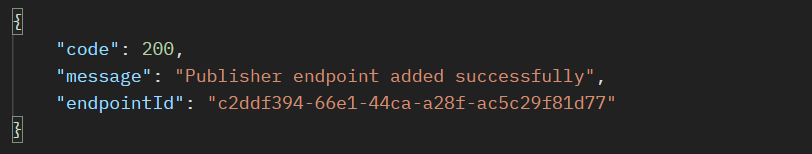
"prefilterGeoCountry": ["en","ua", "us"],

"prefilterMaxBidPrice": 1.42,

"isActive": "Y"

}

* Successful Response



#### Update endpoint – update a publisher endpoint by endpoint id for a specific publisher

* router: /api/v1/publisher/:publisherId/endpoint/:endpointId
* method: PATCH
* Headers: [{ “x-api-key”: “Security key of the publisher or Admin access key”}]
* body: {

"dspEndpointUrl": "http://localhost:8001 /dsp",

"queriesPerSecond": 100,

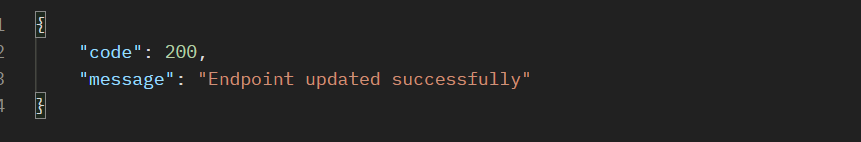
"prefilterGeoCountry": ["en","ua", "us"],

"prefilterMaxBidPrice": 1.42,

"isActive": "Y"

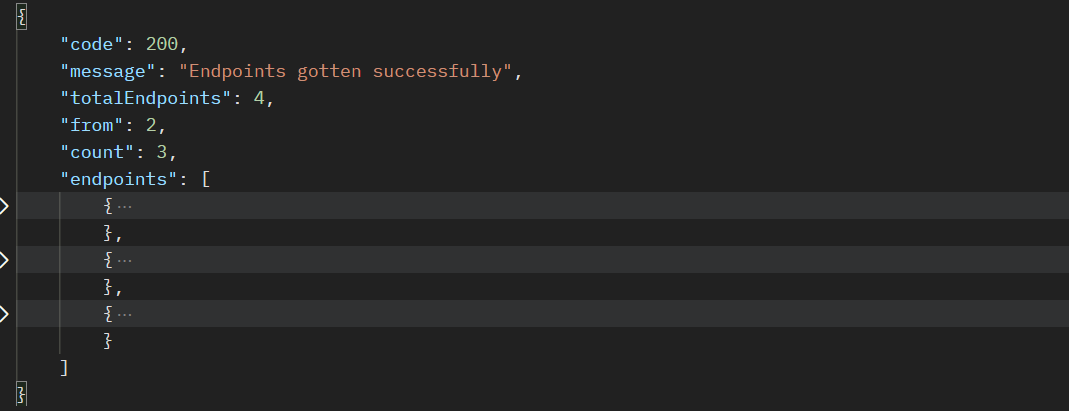
}

* Successful Response



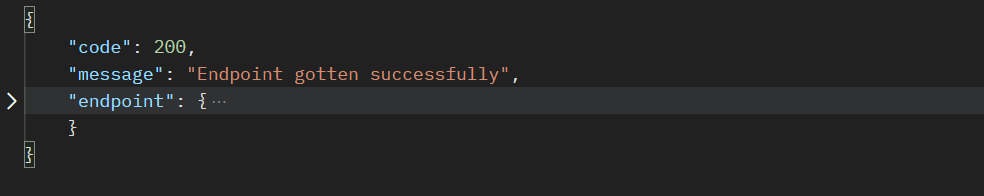
#### Get endpoints list by publisher id – retrieve desired number of endpoints specific position for a specific publisher

* router: /api/v1/publisher/:publisherId/endpoint/list/:start/:limit
* method: GET
* Headers: [{ “x-api-key”: “Security key of the publisher or Admin access key”}]
* Successful Response



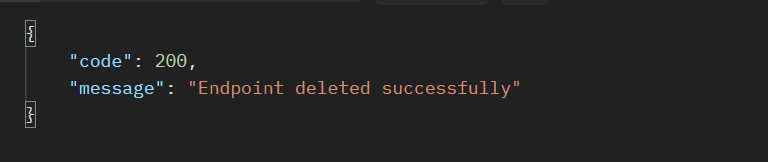
#### Get a publisher endpoint – retrieve a publisher endpoint by endpoint id for a specific publisher

* router: /api/v1/publisher/:publisherId/endpoint/:endpointId
* method: GET
* Headers: [{ “x-api-key”: “Security key of the publisher or Admin access key”}]
* Successful Response



#### Delete a publisher endpoint – delete a publisher endpoint by endpoint id for a specific publisher

* router: /api/v1/publisher/:publisherId/endpoint/:endpointId
* method: DELETE
* Headers: [{ “x-api-key”: “Security key of the publisher or Admin access key”}]
* Successful Response



### Device APIs

#### Add device – add a new device and retrieve the device id

* router: /api/v1/publisher/:publisherId/device
* Method: POST
* Headers: [{ “x-api-key”: “Security key of the publisher or Admin access key”}]
* body: {

"isVideo": "N",

"isImage": "Y",

"isActive": "Y",

"taxonomy": "abc\_0224",

"venuetypeIds": [

"1",

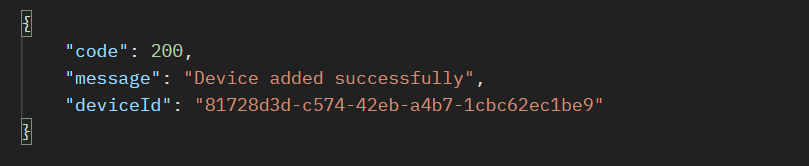
"301"

],

"impsPerSpot": 0.95

}

* Successful Response



#### Update device – update a device by device id for a specific publisher

* router: /api/v1/publisher/:publisherId/device/:deviceId
* method: PATCH
* Headers: [{ “x-api-key”: “Security key of the publisher or Admin access key”}]
* body: {

"isVideo": "N",

"isImage": "Y",

"isActive": "Y",

"taxonomy": "abc\_0224",

"venuetypeIds": [

"1",

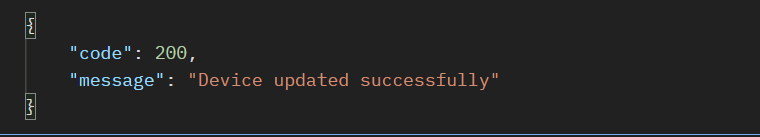
"301"

],

"impsPerSpot": 0.98

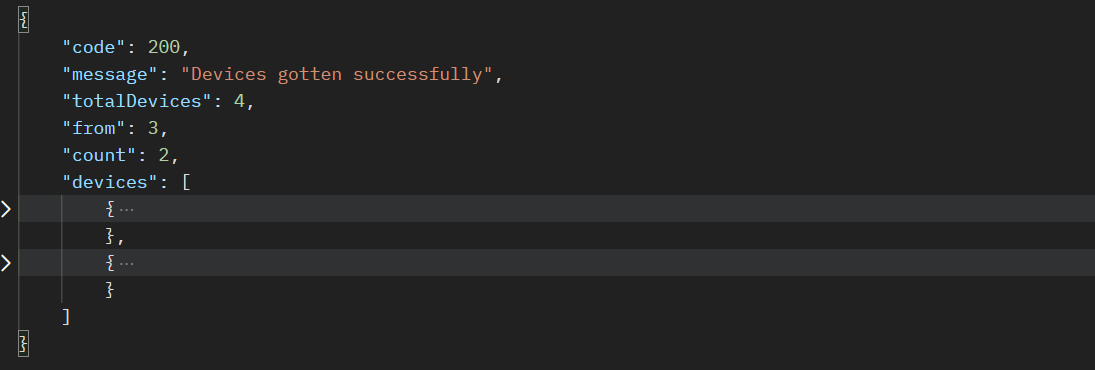
}

* Successful Response



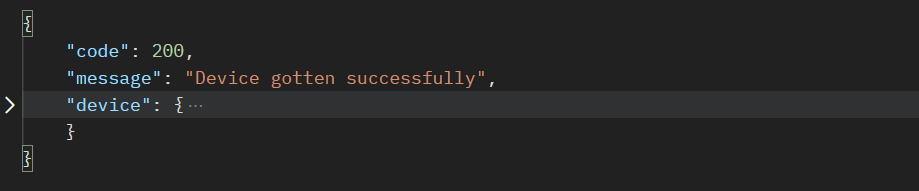
#### Get device list by publisher id – retrieve desired number of devices from specific position for a specific publisher

* router: /api/v1/publisher/:publisherId/device/list/:start/:limit
* method: GET
* Headers: [{ “x-api-key”: “Security key of the publisher or Admin access key”}]
* Successful Response



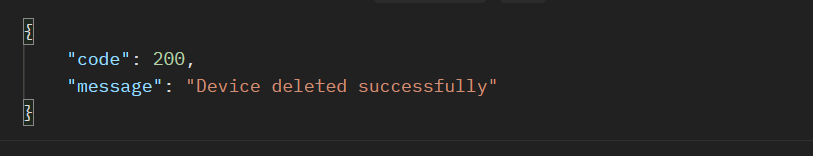
#### Get a device – retrieve a device by device id for a specific publisher

* router: /api/v1/publisher/:publisherId/device/:deviceId
* method: GET
* Headers: [{ “x-api-key”: “Security key of the publisher or Admin access key”}]
* Successful Response



#### Delete a device – delete a device by device id for a specific publisher

* router: /api/v1/publisher/:publisherId/device/:deviceId
* method: DELETE
* Headers: [{ “x-api-key”: “Security key of the publisher or Admin access key”}]
* Successful Response



### Publisher Deal APIs

#### Add a publisher deal – add a new publisher deal and retrieve the deal id

* router: /api/v1/publisher/:publisherId/deal
* Method: POST
* Headers: [{ “x-api-key”: “Security key of the publisher or Admin access key”}]
* body: {

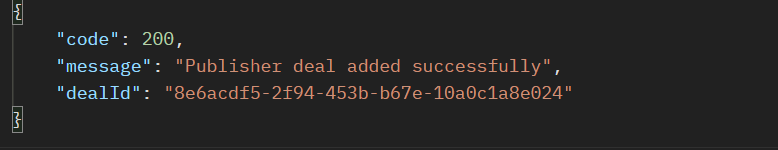
"isActive": "Y",

"bidfloor": 1.23,

"auctionType": "2"

}

* Successful Response



#### Update publisher deal – update a publisher deal by deal id for a specific publisher

* router: /api/v1/publisher/:publisherId/deal/:dealId
* method: PATCH
* Headers: [{ “x-api-key”: “Security key of the publisher or Admin access key”}]
* body: {

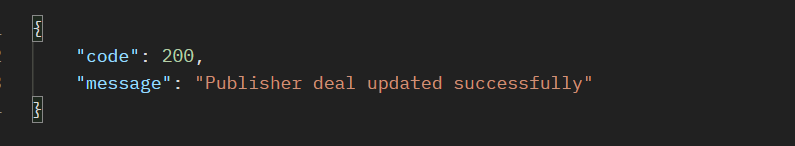
"isActive": "Y",

"bidfloor": 0.98,

"auctionType": "2"

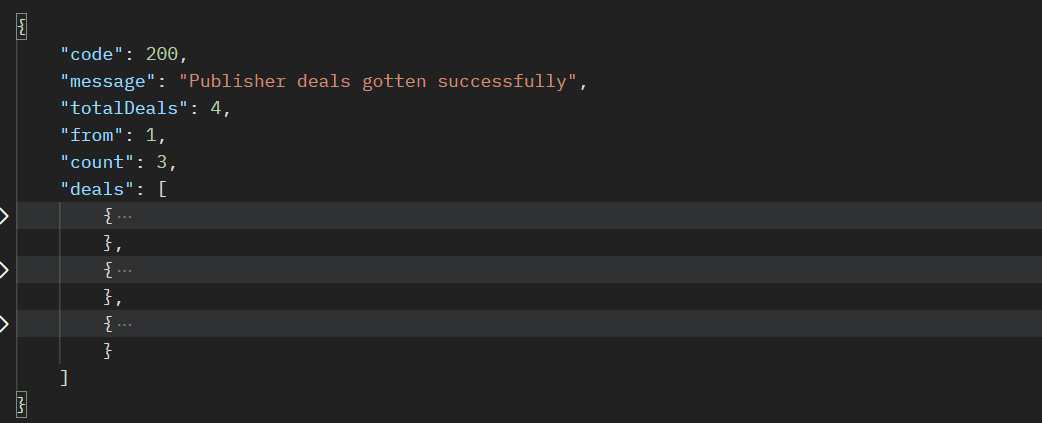
}

* Successful Response



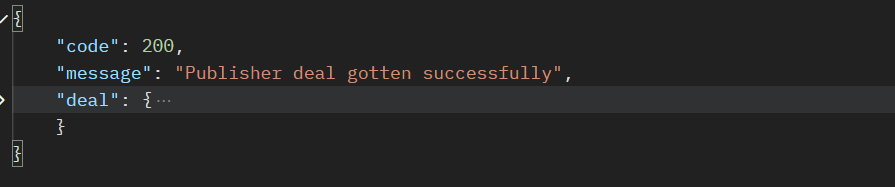
#### Get publisher deal list by publisher id – retrieve desired number of publisher deals from specific position in our database

* router: /api/v1/publisher/:publisherId/deal/list/:start/:limit
* method: GET
* Headers: [{ “x-api-key”: “Security key of the publisher or Admin access key”}]
* Successful Response



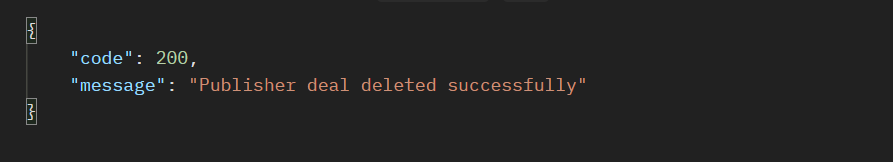
#### Get a publisher deal – retrieve a publisher deal by deal id for a specific publisher

* router: /api/v1/publisher/:publisherId/deal/:dealId
* method: GET
* Headers: [{ “x-api-key”: “Security key of the publisher or Admin access key”}]
* Successful Response



#### Delete a publisher deal – delete a publisher deal by deal id for a specific publisher

* router: /api/v1/publisher/:publisherId/deal/:dealId
* method: DELETE
* Headers: [{ “x-api-key”: “Security key of the publisher or Admin access key”}]
* Successful Response



### Report APIs

### These APIs send an email to a specific publisher by publisher id along with a csv file included analytic data until current date

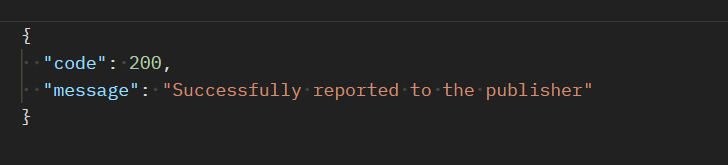
##### Router1: /api/v1/report/:publisherId

This sends a report to a publisher for all publisher endpoints

##### Router2: /api/v1/report/:publisherId/:endpointId

This sends a report to a publisher for a publisher endpoint

* method: GET
* Headers: [{ “x-api-key”: “Security key of the publisher or Admin access key”}]
* Successful Response



### Auction API – initiate an auction and retrieve auction result

* router: /api/v1/auction
* method: GET
* Headers: [{ “x-api-key”: “Security key of the publisher or Admin access key”}]
* query: {

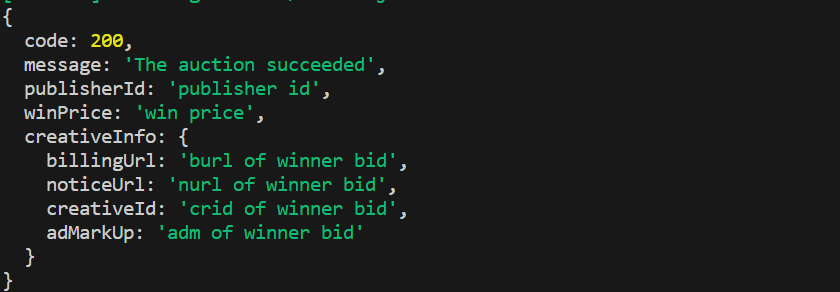
“publisher\_id”: publisherId,

“device\_id”: deviceId,

“ad\_unit\_id”: 1

}

* Successful Response



## High Traffic Testing using Apache bench

### Environment Setup

#### On Ubuntu

1. Update package database

# apt-get update

1. Install apache2 utils package to get access to Apache Bench

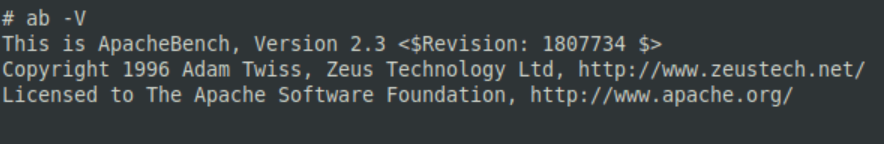
# apt-get install apache2-utils

Source: https://bobcares.com/blog/apache-benchmark-install-ubuntu

1. Verifying Apache Bench Installation

# ab –V

The command above produced on output as shown below:



#### On Windows

* Install using choco command

chocho install apache-httpd

Source: <https://community.chocolatey.org/packages/apache-httpd>

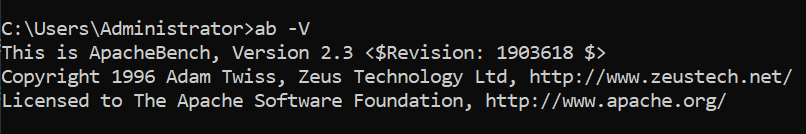
*\* how can use choco command on windows*

Run the following command

@"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -ExecutionPolicy Bypass -Command "[System.Net.ServicePointManager]::SecurityProtocol = 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))" && SET "PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin"

* Verifying Apache Bench Installation

ab –V



### Test using ab command

ab -n 1 500 000 -c 5 -H "x-api-key: Admin access key or security key of the publisher" "auction url"

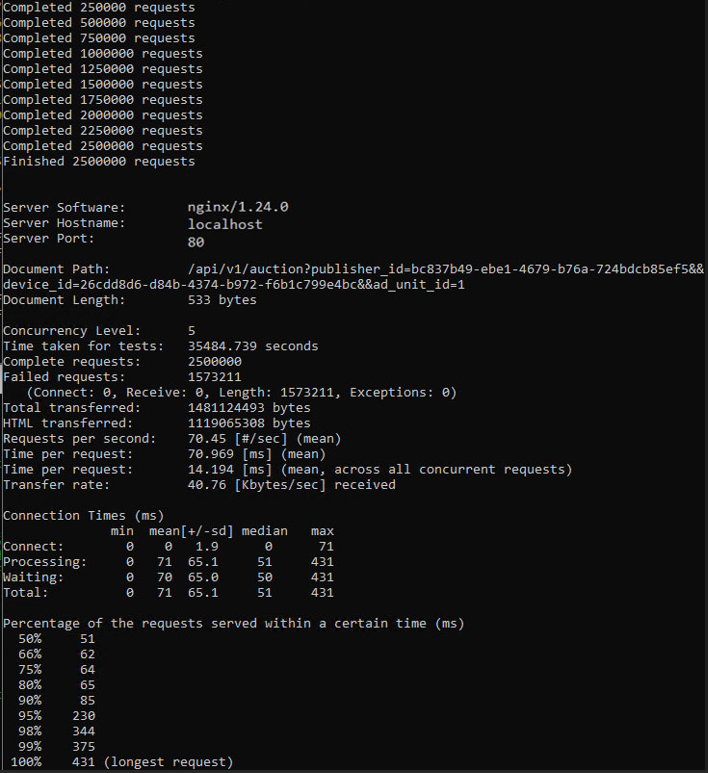
***eg* :** ab –n 1 500 000 -c 5 -H "x-api-key: 3yT7jN9sBdR5fQgP2mW6uE4zA8cV1xX0" "http://localhost:80/api/v1/auction?publisher\_id=bc837b49-ebe1-4679-b76a-724bdcb85ef5&&device\_id=26cdd8d6-d84b-4374-b972-f6b1c799e4bc&&ad\_unit\_id=1"

**This is used to perform a load test on our system with high traffic condition.**

* ab: This is the Apache Bechmark tool used to perform the load testing
* -n 3000: This option specifies the total number of requests to perform.
* -c 10: This option specifies the number of multiple requests to perform at a time (i.e. currently level). Here, it is set to 10, meaning 10 requests will be sent simultaneously.
* -H “x-api-key: 3yT7jN9sBdR5fQgP2mW6uE4zA8cV1xX0”: This option adds a custom header to authenticate the request.
* http://localhost:80/api/v1/auction?publisher\_id=bc837b49-ebe1-4679-b76a-724bdcb85ef5&&device\_id=26cdd8d6-d84b-4374-b972-f6b1c799e4bc&&ad\_unit\_id=1

This is the URL of the endpoint being tested.

### Testing result



**Summary of Results**

* Total Requests: 2 500 000
* **Concurrency Level**: 5 which sends 5 requests to the server concurrently.
* Time Taken for Tests: 35484.739 seconds equals around 10hrs
* Complete Requests: 2 500 000
* Failed Requests: 1 573 211

Length Errors: 1 573 211 (This doesn’t mean server error, just length error related to responses' length. don’t need to worry about this)

* Total Transferred: 1481124493 bytes
* HTML Transferred: 1119065308 bytes
* Requests per Second: 70.45 [#/sec] (mean)
* Time per Request: 70.969[ms] (mean)
* Time per Request (across all concurrent requests): 14.194[ms]
* Transfer Rate: 40.76 [Kbytes/sec] received

**Percentage of Requests Served within a Certain Time (ms):**

* 50%: 51ms
* 66%: 62ms
* 75%: 64ms
* 80%: 65ms
* 90%: 85ms
* 95%: 230ms
* 99%: 375ms
* 100%: 431ms (longest request)