

## Part 1. Consider the GET request: /api/v0/aircrafts/export

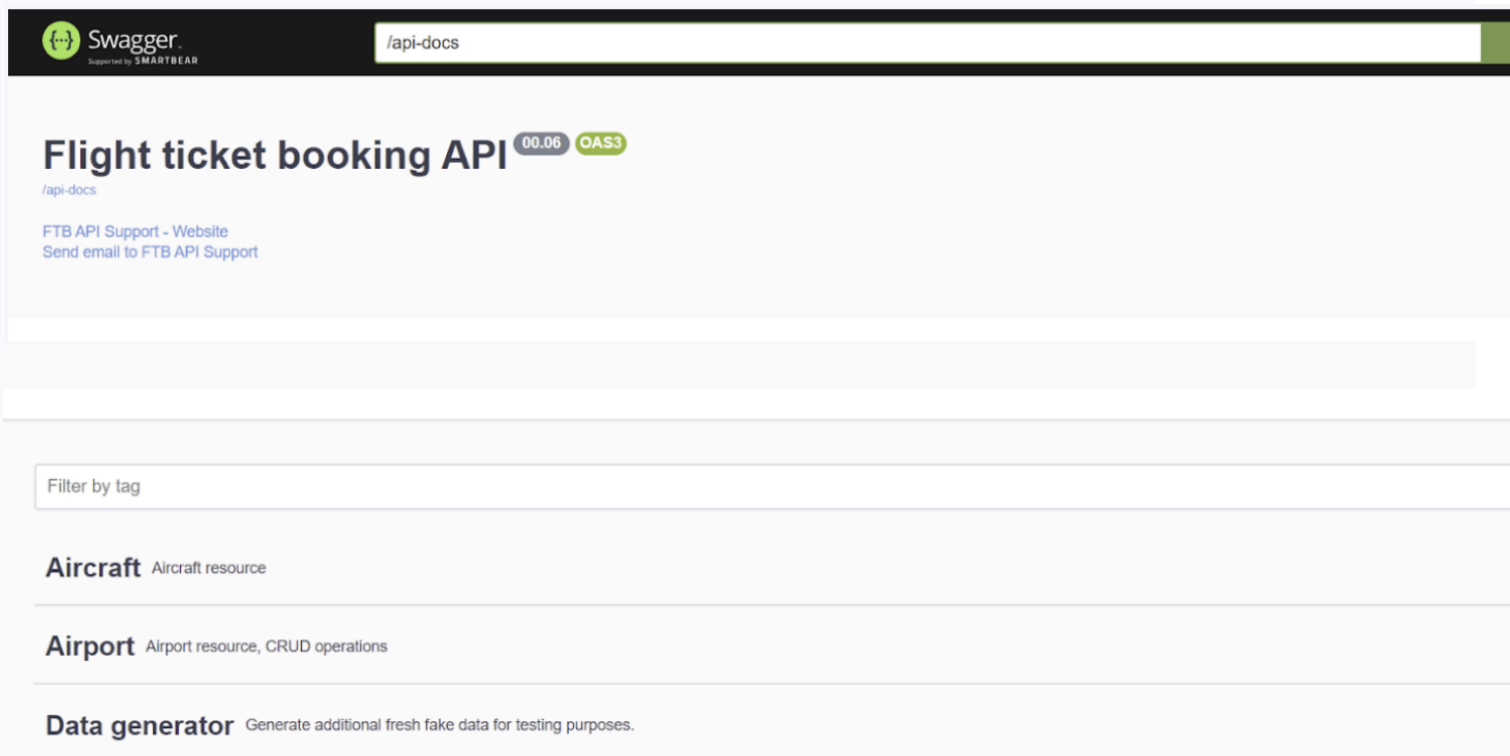
- Read the documentation in Swagger

<http://server:8000/swagger-ui/index.html#/Aircraft/exportAllAsCsvFile> and describe what this method does.

Login with user: john and password: john123

<http://server:8000/swagger-ui/index.html#/Aircraft/exportAllAsCsvFile> and we can see

Documentation about Flight ticket booking API



Swagger  
Supported by SMARTBEAR

/api-docs

# Flight ticket booking API

00.06 OAS3

/api-docs

[FTB API Support - Website](#)  
[Send email to FTB API Support](#)

Filter by tag

**Aircraft** Aircraft resource

**Airport** Airport resource, CRUD operations

**Data generator** Generate additional fresh fake data for testing purposes.

We can see such different buttons:

Filter by tag

**Aircraft** Aircraft resource

**Airport** Airport resource, CRUD operations

**Data generator** Generate additional fresh fake data for testing purposes.

**Debug** Debug assistance

**Flight** Flight resource. Flight booking is possible as well!

**Passenger** Passenger resource.

**User** User resource. CRUD+ operations.

**Version** This is the way to get app/API version info.

When we click on each button we can see all commands that we can do with this api Flight ticket booking API and all responses are written here. We can delete, get, patch, post, put the information with this API. By clicking on all buttons we can see detailed menu :

## Aircraft

Aircraft resource

### DELETE

</api/v0/aircrafts/{id}>

Attempt to delete an entity by its id.

### GET

</api/v0/aircrafts/{id}>

Get an entity by its id.

### GET

</api/v0/aircrafts>

Get all entities available.

### GET

</api/v0/aircrafts/paged>

Get entities as a paged list.

### GET

</api/v0/aircrafts/page/{number}>

Get entities available on the page.

GET

</api/v0/aircrafts/model/{modelName}>

Attempt to get an aircraft by its model name.

GET

</api/v0/aircrafts/manufacturer/{manufacturerName}>

Attempt to get an aircraft by its manufacturer name.

GET

</api/v0/aircrafts/export>

Attempt to export all aircraft records to CSV file.

Parameters

Try it out

No parameters

Responses

C o d e	Description
------------------	-------------

2 0 0	<div>OK</div> <div>Media type</div> <div><b><i>*/*</i></b></div> <div>Controls Accept header.</div> <div>Example Value</div> <div>Schema</div> <div><code>{ }</code></div>
-------------	--

4  
0  
0

## Bad Request

Media type

\*/\*

### Example Value

Schema

```
{  
  "timestamp": "2023-12-12T10:33:58.672Z",  
  "status": "string",  
  "message": "string",  
  "details": [  
    "string"  
  ]  
}
```

## PATCH

</api/v0/aircrafts/{id}>

Partial update using JSON-Patch operations.

## POST

</api/v0/aircrafts>

Attempt to create an entity by using its DTO.

## POST

</api/v0/aircrafts/import>

Attempt to import aircraft data from CSV file.

## POST

</api/v0/aircrafts/import/async>

Attempt to import aircraft data from CSV file asynchronously.

## PUT

</api/v0/aircrafts/{id}>

Attempt to update an entity by using its DTO.

For example we can see information from our own app with port 8108 about flights . Let us take flightid 56

[http://server\\_ip:8108/swagger-ui/index.html#/Flight/findById\\_2](http://server_ip:8108/swagger-ui/index.html#/Flight/findById_2)

Server response	
Code	Details
200	<div>Response body</div> <pre>{   "flightId": 56,   "flightNumber": "IA-TEX101",   "departureAirportCode": "CID",   "destinationAirportCode": "TEX",   "departureDate": [     2023,     10,     7   ],   "arrivalDate": [     2023,     10,     7   ],   "departureTime": "09:00",   "arrivalTime": "15:00",   "gate": "12",   "status": "12",   "flightCharge": 100,   "aircraftId": 36,   "passengerIds": [     36   ],   "id": 56 }</pre> <div>Response headers</div>

- How to run this method from curl? What happened?

We can see the info about flight with flightid= 56 with command curl

curl -v -X "GET" http://server\_ip:8108/api/v0/flights/56 --user "john:john123"

```
> Authorization: Basic am9objpb2huMTIz
> User-Agent: curl/8.4.0
> Accept: */*
>
< HTTP/1.1 200
< Set-Cookie: JSESSIONID=786A8F88CE13C99394B3AF4647624292; Path=/; HttpOnly
< X-Content-Type-Options: nosniff
< X-XSS-Protection: 1; mode=block
< Cache-Control: no-cache, no-store, max-age=0, must-revalidate
< Pragma: no-cache
< Expires: 0
< X-Frame-Options: SAMEORIGIN
< Content-Type: application/json
< Transfer-Encoding: chunked
< Date: Tue, 12 Dec 2023 15:04:56 GMT
<
{"flightId":56,"flightNumber":"IA-TEX101","departureAirportCode":"CID","destinationAirportCode":"TEX","departureDate":[2023,10,7],"arrivalDate":[2023,10,7],"departureTime":"09:00","arrivalTime":"15:00","gate":"12","status":"12","flightCharge":100.0,"aircraftId":36,"passengerIds":[36],"id":56}* Connection closed by client.
```

- How to redirect the result to a file instead of a terminal?

curl -v -X "GET" http://server\_ip:8108/api/v0/flights/56 --user "john:john123" | cat > flights

All info about this flight will be in file flights

```
ipapara@c7-sandbox ~]$ cat flights
{"flightId":56,"flightNumber":"IA-TEX101","departureAirportCode":"CID","destinationAirportCode":"TEX","departureDate":[2023,10,7],"arrivalDate":[2023,10,7],"departureTime":"09:00","arrivalTime":"15:00","gate":"12","status":"12","flightCharge":100.0,"aircraftId":36,"passengerIds":[36],"id":56}
ipapara@c7-sandbox ~]$
```

- Say, we need to see an empty result to check how the method works in this case. What preparation needs to be done in the system?

We can do empty info in all fields in this flight with flightid 56 manually in DB

25	56	[NULL]	[NULL]	[NULL]	[NULL]	0	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]

And then use get with Flight

Flight

Flight resource. Flight booking is possible as well!

## Server response

Code      Details

200

### Response body

```
{
  "flightId": 56,
  "flightNumber": null,
  "departureAirportCode": null,
  "destinationAirportCode": null,
  "departureDate": null,
  "arrivalDate": null,
  "departureTime": null,
  "arrivalTime": null,
  "gate": null,
  "status": null,
  "flightCharge": 0,
  "aircraftId": 0,
  "passengerIds": [],
  "id": 56
}
```

### Response headers

```
cache-control: no-cache,no-store,max-age=0,must-revalidate
connection: keep-alive
content-type: application/json
date: Tue,12 Dec 2023 15:20:10 GMT
expires: 0
keep-alive: timeout=60
pragma: no-cache
transfer-encoding: chunked
x-content-type-options: nosniff
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
```

## With curl from cmd

```
Authorization: Basic am9objpqb2huMTIz
User-Agent: curl/8.4.0
Accept: */*

HTTP/1.1 200
Set-Cookie: JSESSIONID=81B0DF936708E3282627C11ED736D0B5; Path=/; HttpOnly
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Transfer-Encoding: chunked
Date: Tue, 12 Dec 2023 16:34:38 GMT

{"flightId":56,"flightNumber":null,"departureAirportCode":null,"destinationAirportCode":null,"departureDate":null,"arrivalDate":null,"departureTime":null,"arrivalTime":null,"gate":null,"status":null,"flightCharge":0.0,"aircraftId":0,"passengerIds":[],"id":56}* Connection #0 to host 192.168.40.100 left intact
```

## Part 2. Explain the components of URI:

- **https://john.doe@www.example.com:8080/forum/questions/?tag=networking&order=newest#top**

Https - protocol component specifies which protocol being used to access the resource

john.doe@[www.example.com](http://www.example.com) - host

8080 - port component, which specifies the port number where the server is listening

/forum/questions/ - path component specifies the location of the resource on the server

?tag=networking&order=newest#top - parameters component contains additional information that the server can use to process the request.

- **telnet://192.0.5.105:23/**

Telnet - here we use telnet protocol

192.0.5.105 - it is an IP address

23 - port

**Part 3\* (optional for 100 points): Explain each component of the following command:**

```
curl -i -H 'X-USER-IDENTITY-DOMAIN-NAME: OAuthTestTenant125' -H 'Authorization: Basic MzAzYTI0OTItZDY0Zi00ZTA0LWI3OGYtYjQzMzAwNDczMTJiOll5Sk5NSkdFc0ZqUkxWZVZsdVMz' -H 'Content-Type: application/x-www-form-urlencoded;charset=UTF-8' --request POST https://<idm-domain>.identity.<data-center>.oraclecloud.com/oauth/tokens -d 'grant_type=password &username=tenantAdminUser &password=Fusionapps1&scope=http://www.example.com'
```

i (or --include): Includes the HTTP response headers in the output along with the response body.

H (or --header): Sets a specific HTTP request header. It can be used to pass various parameters, such as setting an authorization token or other custom headers.

'X-USER-IDENTITY-DOMAIN-NAME: OAuthTestTenant125' This header specifies the name of the identity domain

-H 'Authorization: Basic MzAzYTI0OTItZDY0Zi00ZTA0LWI3OGYtYjQzMzAwNDczMTJiOll5Sk5NSkdFc0ZqUkxWZVZsdVMz' - header

This header specifies the authorization credentials.

'Content-Type: application/x-www-form-urlencoded;charset=UTF-8' -header. This header specifies the type of content that is being sent in the request

--request option in curl is used to specify the HTTP method to be used in the request. This option allows you to explicitly set the type of HTTP request you want to perform.

POST - it is our HTTP method

https://<idm-domain>.identity.<data-center>.oraclecloud.com/oauth/tokens - This is the URL that is being used to obtain the access token.

-d 'grant\_type=password &username=tenantAdminUser &password=Fusionapps1&scope=http://www.example.com'

This option is used to specify the data that is being sent in the request body. In this case, it is grant\_type=password &username=tenantAdminUser&password=Fusionapps1&scope=http://www.example.com.