

16 November 2023

API Features

1. Table of Contents

Contents

1. Table of Contents	2
2. Introduction	3
Version history	3
3. API Whitelisting	4
3.1 Definition	4
3.2 How to use API whitelisting	4
3.3 General Use Case	4
4. API Sorting	7
4.1 Definition	7
4.2 How to use API sorting	7
4.3 General Use Case	7
5. API Pagination	9
5.1 Definition	9
5.2 How to use API Pagination	9
5.3 General Use Case	9
6. API Searching	12
6.1 Definition	12
6.2 Search Criteria	12
6.3 How to use API Searching	13
6.4 General Use Case	13
7. API Batch	16
7.1 Definition	16
7.2 How to use API batch call	16
7.3 General Use Case	17

2. Introduction

Version history

Ver.	Date of creation	Changes in document	Who changed
0	08/11/2023	Created initial version of API whitelisting, sorting, pagination and searching	PHCH
1	13/11/2023	Add Whitelist=all and Add Batch call feature	PHCH
2	16/11/2023	Fixed – No need “<endpoint>” in batch call	PHCH

3. API Whitelisting

3.1 Definition

API whitelisting is a parameter or query string used to indicate the fields you want to whitelist or include in response. The API, when processing this request, will use the “whitelist” parameter to filter and return only the specified fields in the response data, allowing to receive a more focused set of information containing only the attributes that you have selected. By using whitelist, you could get more fields or limit certain fields from the default results.

3.2 How to use API whitelisting

- The general concept of using a whitelist in the APIs or data retrieval is to specify a list of allowed or approved items, featured, attributes, or elements that you want to include or retrieve from a larger set of data or resources.
- Append “**whitelist=<parameters>**” in the URL
- Parameters options:
 - **All** – Return all available fields of that specific endpoints.
 - Ex. `http://<ip_address>/api/v2.0.0/<endpoint>?whitelist=all`
 - **Specified fields** – Return a set of information containing only the attributes that you have given.
 - Ex. `http://<ip_address>/api/v2.0.0/<endpoint>?whitelist=<param>`

3.3 General Use Case

- Determining the list of parameters to include in a “whitelist” or in an API request depends on your specific use case, the API you are interacting with, and your data needs. Regarding MiR API, you could simply see the parameters in a nested URL path.
- For example:
 - I would like to know robot id, robot ip and fleet state text of the robot. The normal way to do that is to go http://<ip_address>/api/v2.0.0/robots/<id>, but this way you will have a call based on the number of the robot.
 - The response data will look like Figure 1

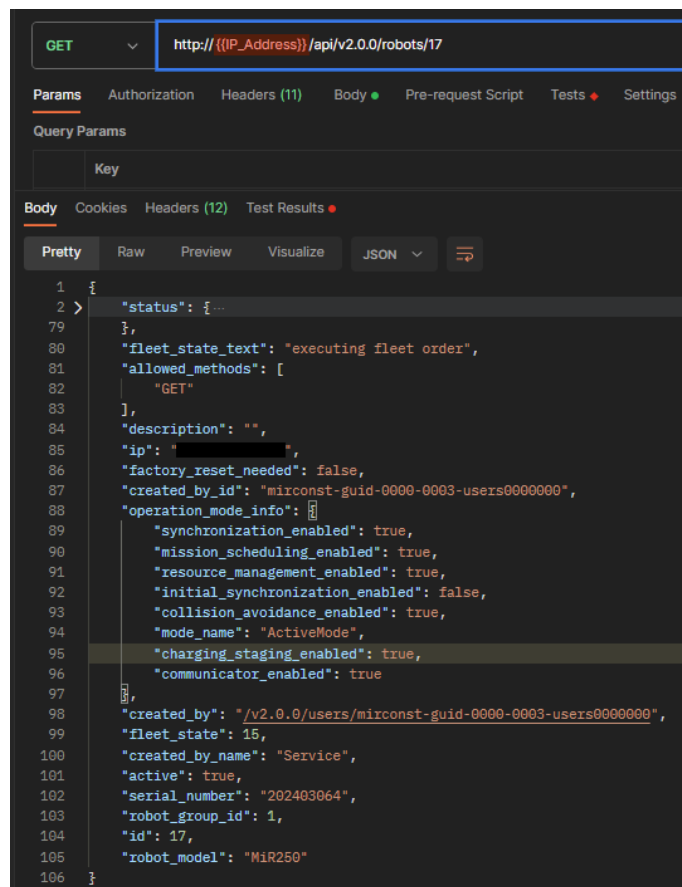


Figure 1 - /robots/<id> response data

- The flexible way to get all robots ip, id and fleet_state_text in 1 single call is whitelisting those parameters by:
 - http://<ip_address>/api/v2.0.0/robots?whitelist=ip,id,fleet_state_text
- In the provided parameters, it is important to note that the whitelisting/filtering applies only to the top-level keys of data. Any nested data within those fields is treated as a single unit, and you cannot filter or customize them separately.

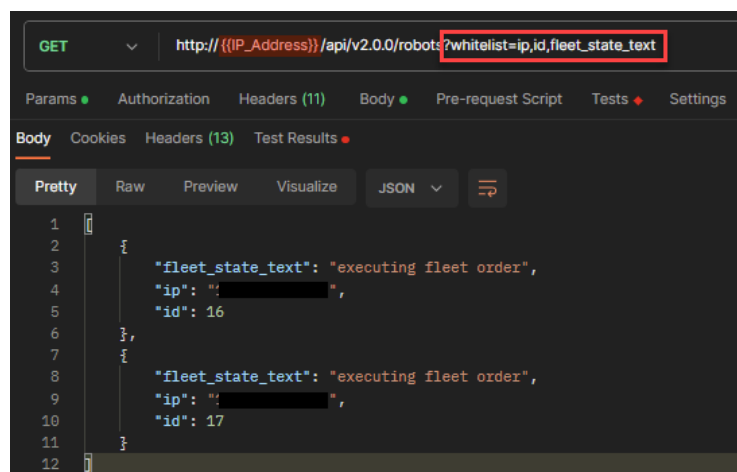


Figure 2 - /robots?whitelist=ip,id,fleet_state_text response data

- In addition to that, if you would like to know all robot's status in 1 single call, you could simply do:
 - http://<ip_address>/api/v2.0.0/robots?whitelist=all
 - This way you will no need to list all parameters as shown as Figure 3.

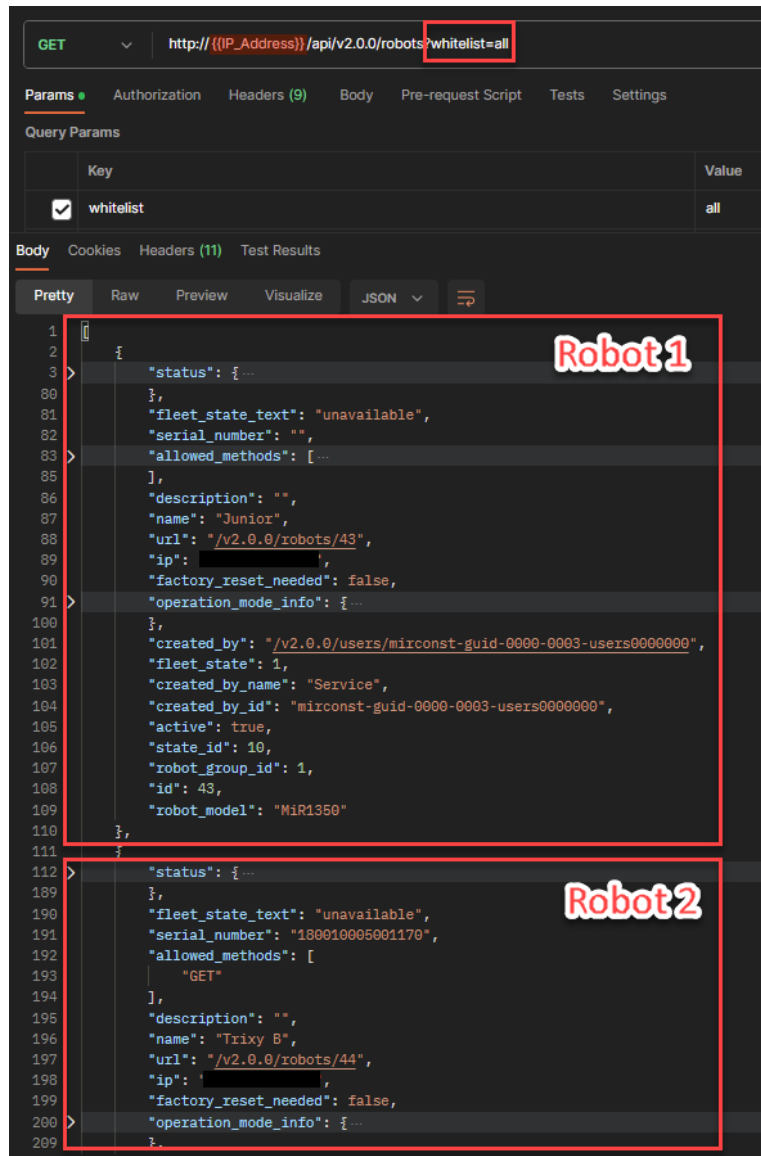


Figure 3 - /robots?whitelist=all response data

4. API Sorting

4.1 Definition

API sorting is the practice of arranging data returned by an API in a specific order based on one or more criteria, such as alphabetical order or numerical order. It allows you to retrieve data in a structured and organized manner, making it easier to find, analyze, or display the information according to your needs. Sorting is commonly used to order lists of items, such as ID, in ascending or descending order.

4.2 How to use API sorting

- The general concept of API sorting involves arranging data from an API response in a specific order based on defined criteria.
- Append **"sort_by=<parameters>,<asc or desc>"** in the URL
 - o http://<ip_address>/api/v2.0.0/<endpoint>?sort_by=<param>,<asc or dsc>
 - o ASC = Ascending order
 - o DESC = Descending order

4.3 General Use Case

- For example:
 - o I would like to know the latest mission queue of the robots. In general, you could simply request `"/api/v2.0.0/mission_queue"`, but it will respond as an ascending order.

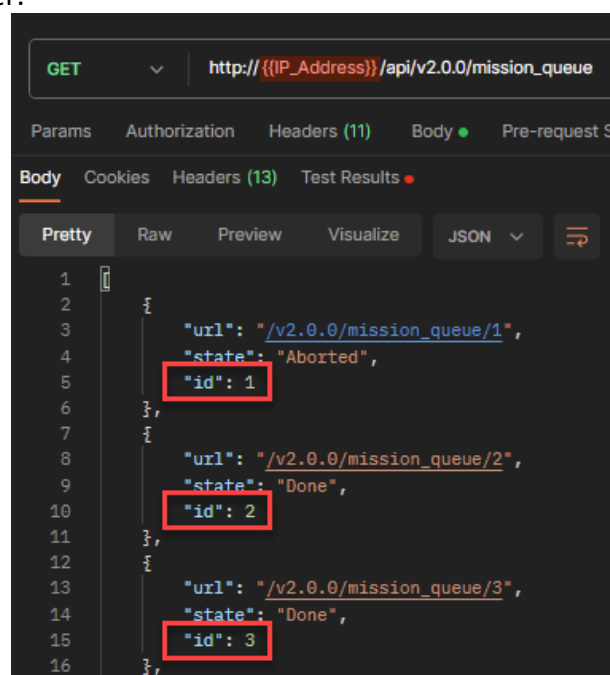


Figure 4 - /mission_queue response data

- To sort the latest data, you could simply do:
 - http://<ip>/api/v2.0.0/mission_queue?sort_by=id,desc
 - It means that the response data will return descending order of id as shown in Figure 5

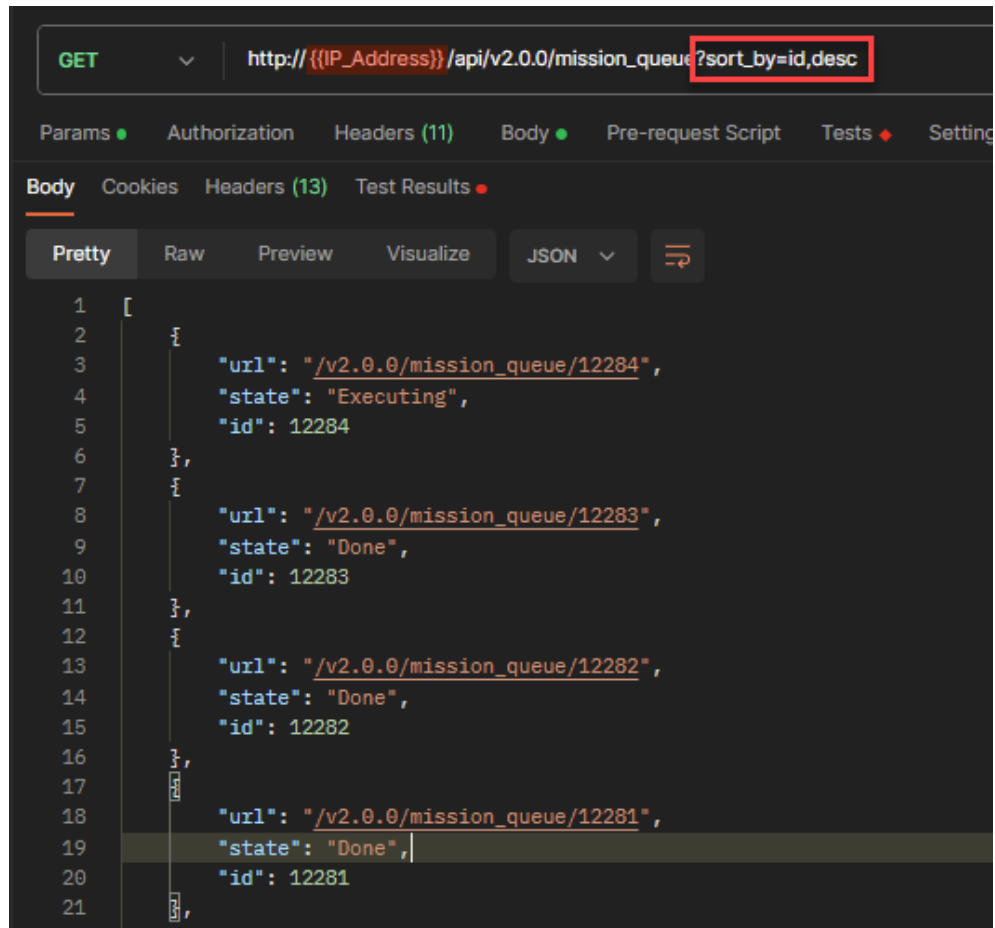


Figure 5 - /mission_queue?sort_by=id,desc

5. API Pagination

5.1 Definition

API pagination is a technique used to break down large sets of data into a smaller, manageable subsets to improve the efficiency and performance of data retrieval. It's particularly useful when working with APIs that provide extensive datasets. It will be very important to save system resources and limiting the size of the return message from REST API. It involves 2 methods, **limit** and **offset**.

Limit: The 'limit' parameter specifies the maximum number of records or items to retrieve in a single API request. It restricts the quantity of data returned.

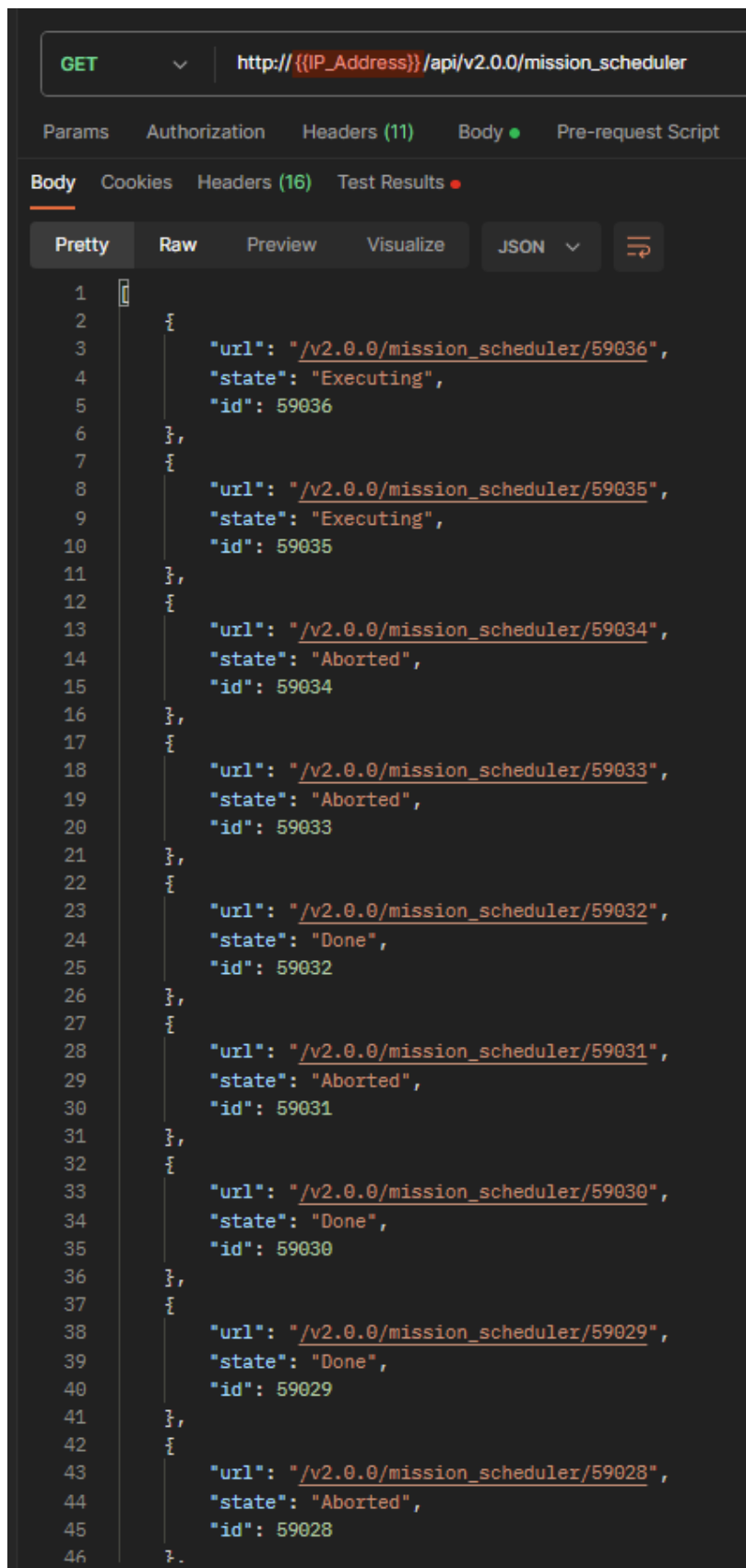
Offset: The 'offset' parameter indicates where in the dataset to begin retrieving records. It's often used for navigating to the next set of results.

5.2 How to use API Pagination

- The general concept of API pagination is a strategy for efficiently retrieving and presenting data in smaller, manageable portions, make it easier to work with extensive datasets.
- Append "**limit=<size of data>**" in the URL
 - <http://<ip>/api/v2.0.0/<endpoint>?limit=<SizeOfData>>
- Append "**offset=<number of beginnings retrieving records>**"
 - <http://<ip>/api/v2.0.0/<endpoint>?offset=<StartingPoint>>
- It is able to combine both limit and offset in the URL.
 - <http://<ip>/api/v2.0.0/<endpoint>?limit=<SizeOfData>&offset=<StartingPoint>>

5.3 General Use Case

- For example:
 - I would like to know only last 5 missions and want to start from second data set in the Fleet mission scheduler. In general, you could simply request '/api/v2.0.0/mission_scheduler', but it will return all the datasets available in the APIs, as shown in Figure 6



```

GET http://{{IP_Address}}/api/v2.0.0/mission_scheduler

Params Authorization Headers (11) Body ● Pre-request Script

Body Cookies Headers (16) Test Results ●

Pretty Raw Preview Visualize JSON ↕

1 [
2   {
3     "url": "/v2.0.0/mission_scheduler/59036",
4     "state": "Executing",
5     "id": 59036
6   },
7   {
8     "url": "/v2.0.0/mission_scheduler/59035",
9     "state": "Executing",
10    "id": 59035
11  },
12  {
13    "url": "/v2.0.0/mission_scheduler/59034",
14    "state": "Aborted",
15    "id": 59034
16  },
17  {
18    "url": "/v2.0.0/mission_scheduler/59033",
19    "state": "Aborted",
20    "id": 59033
21  },
22  {
23    "url": "/v2.0.0/mission_scheduler/59032",
24    "state": "Done",
25    "id": 59032
26  },
27  {
28    "url": "/v2.0.0/mission_scheduler/59031",
29    "state": "Aborted",
30    "id": 59031
31  },
32  {
33    "url": "/v2.0.0/mission_scheduler/59030",
34    "state": "Done",
35    "id": 59030
36  },
37  {
38    "url": "/v2.0.0/mission_scheduler/59029",
39    "state": "Done",
40    "id": 59029
41  },
42  {
43    "url": "/v2.0.0/mission_scheduler/59028",
44    "state": "Aborted",
45    "id": 59028
46  }

```

Figure 6 - /mission_scheduler endpoint response data

- To paginate only latest 5 missions and start from the 2nd data, you could simply do:
 - http://<ip>/api/v2.0.0/mission_scheduler?limit=5&offset=1
 - Limit=5 means limit the response data to only 5 datasets
 - Offset=1 means start retrieving data from the first item, as array starts with 0. In other words, it starts 2nd item of the dataset.

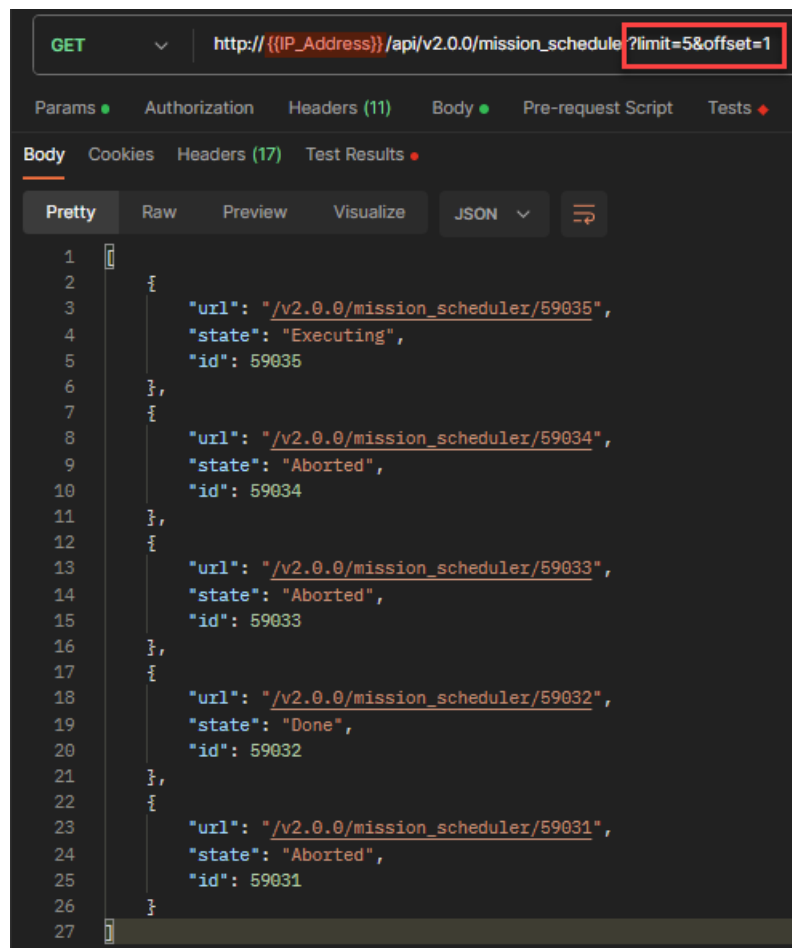


Figure 7 - /mission_scheduler?limit=5&offset=1 endpoint response data

6. API Searching

6.1 Definition

API Searching is a feature that allows user to query an API to find a specific data or information within a dataset. It enables users to retrieve relevant data by specifying search criteria, such as 'Fieldnames', 'Operators' and 'Value' and the API returns results that match those criteria.

6.2 Search Criteria

Search criteria consists of 3 keys component used to construct the search queries:

- 1) **Fieldname:** The fieldname refers to the specific attribute or property of data that you want to search within. It identifies which part of dataset you want to target.
- 2) **Operators:** Operators are symbols or keywords used to define the type of comparison or condition you want to apply to fieldname and its associated value.
 - Allow operators:
 - "="
 - "<=>"
 - "<>"
 - "!="
 - ">"
 - ">="
 - "<"
 - "<="
 - "NOT LIKE"
 - "LIKE"
 - "IN"
 - "NOT IN"
 - "IS NOT"
 - "IS"
- 3) **Value:** The value is the actual content you are looking for or comparing with in the fieldname. It can be a specific string, number, date, or any other data type that matches the field you are searching within.

6.3 How to use API Searching

- You will need to construct the searching body:

```
{
  "filters": [{
    "fieldname": "<fieldname>",
    "operator": "<operators>",
    "value": "<value(s)>"
  }]
}
```

- Append **"/search"** in the URL
 - o <http://<ip>/api/v2.0.0/<endpoint>/search>
- It is important to note that API Searching will work only POST method since we need to give API searching body to make a request.

6.4 General Use Case

- For example:
 - o I would like to know only the mission that is executing in Fleet mission scheduler currently. In general, you could simply request `"/api/v2.0.0/mission_scheduler"`, but it will return all datasets, requiring you to check the string afterwards to determine which mission is currently executing, as shown in Figure 8.

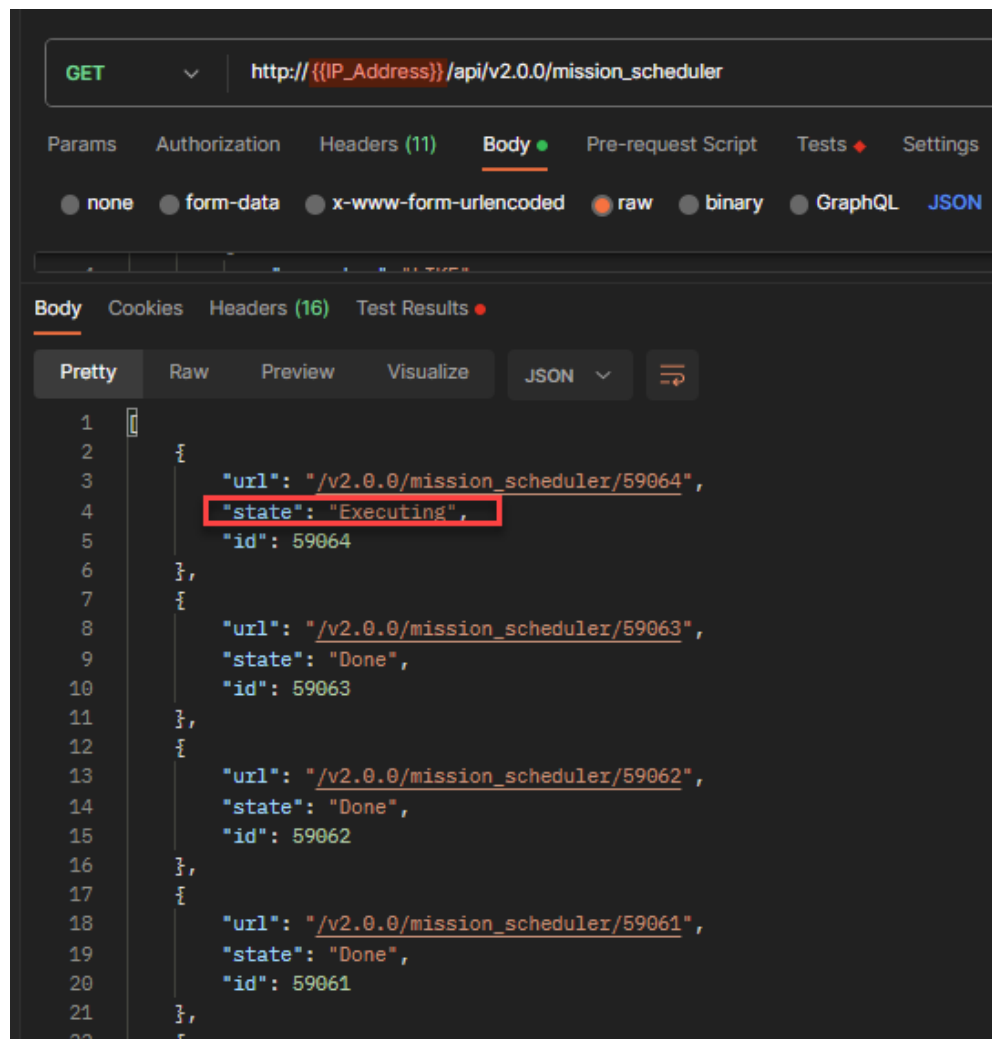


Figure 8 - /mission_scheduler find Executing state

- To search only mission that executing currently by using API searching, you could simply do:
 - POST http://<ip>/api/v2.0.0/mission_scheduler/search
 - Body:

```

{
  "filters": [{
    "fieldname": "state",
    "operator": "=",
    "value": "Executing"
  }]
}

```

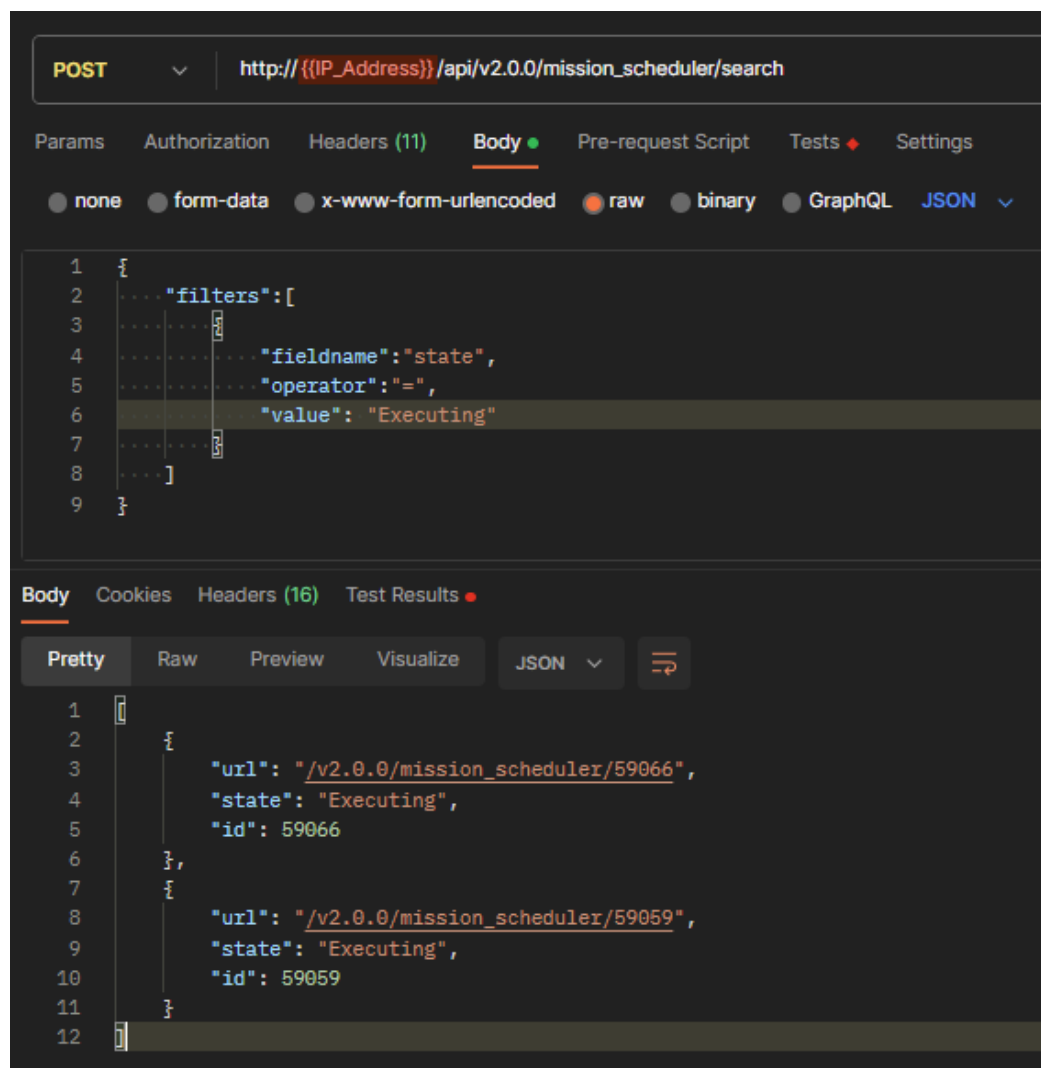


Figure 9 - /mission_scheduler/search find Executing state

7. API Batch

7.1 Definition

- API batch endpoint is a specific route or URL path within an API that is designed to handle batch processing. Batch processing involves the execution of a series of tasks or operations in a single batch, as opposed to processing them individually. This could be more efficient in certain scenarios, especially when dealing with multiple requests. It used to send multiple API requests in a single HTTP request. This can be useful for reducing the number of networks round-trips and improving overall performance.

7.2 How to use API batch call

- Append “/batch” in the URL:
 - Ex. **POST** http://<ip_address>/api/v2.0.0/batch
- You will need to construct the batch body:

```
{
  "requests": [
    {
      "url": "<API endpoint 1>",
      "method": "<Method 1>"
    },
    {
      "url": "<API endpoint 2>",
      "method": "<Method 2>"
    },
    ....
    ....
    ....
  ]
}
```

- It is important to note that API Batch call will work only POST method since we need to give individual API body to make a request.

7.3 General Use Case

- For example:
 - I would like to know the robot's **register 69**, **status** (robot_name, state_text and velocity) and search current executing mission in **mission_queue** (state_id, id and fleet_schedule_guid) within 1 call.
 - Create a body of batch call (as shown in Figure 10) which consists of:
 - /registers/69
 - /status?whitelist=robot_name,state_text,velocity
 - /mission_queue/search?whitelist=state_id,id,fleet_schedule_guid

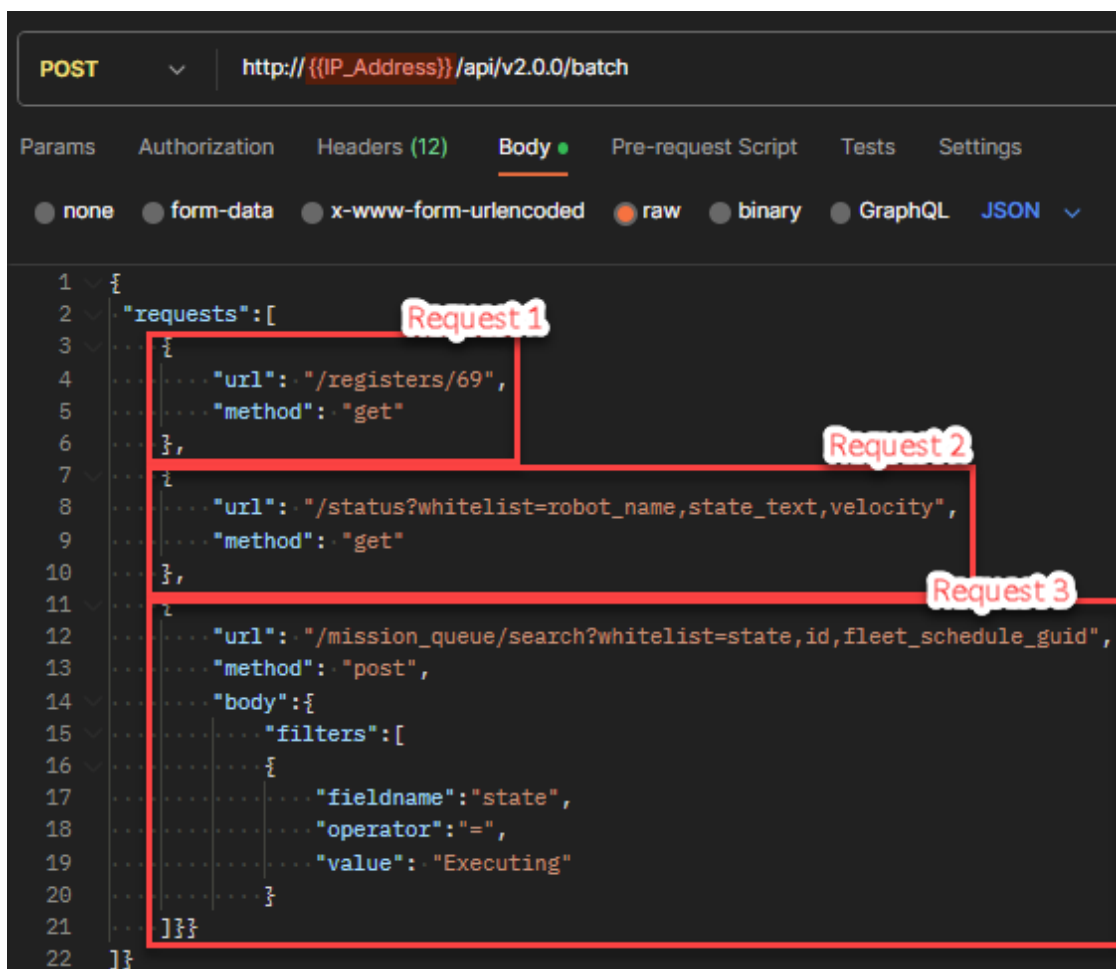


Figure 10 - /batch calls – registers, status and mission_queue

- The response data will consist of 3 datasets according to your API requests respectively as shown in Figure 11.

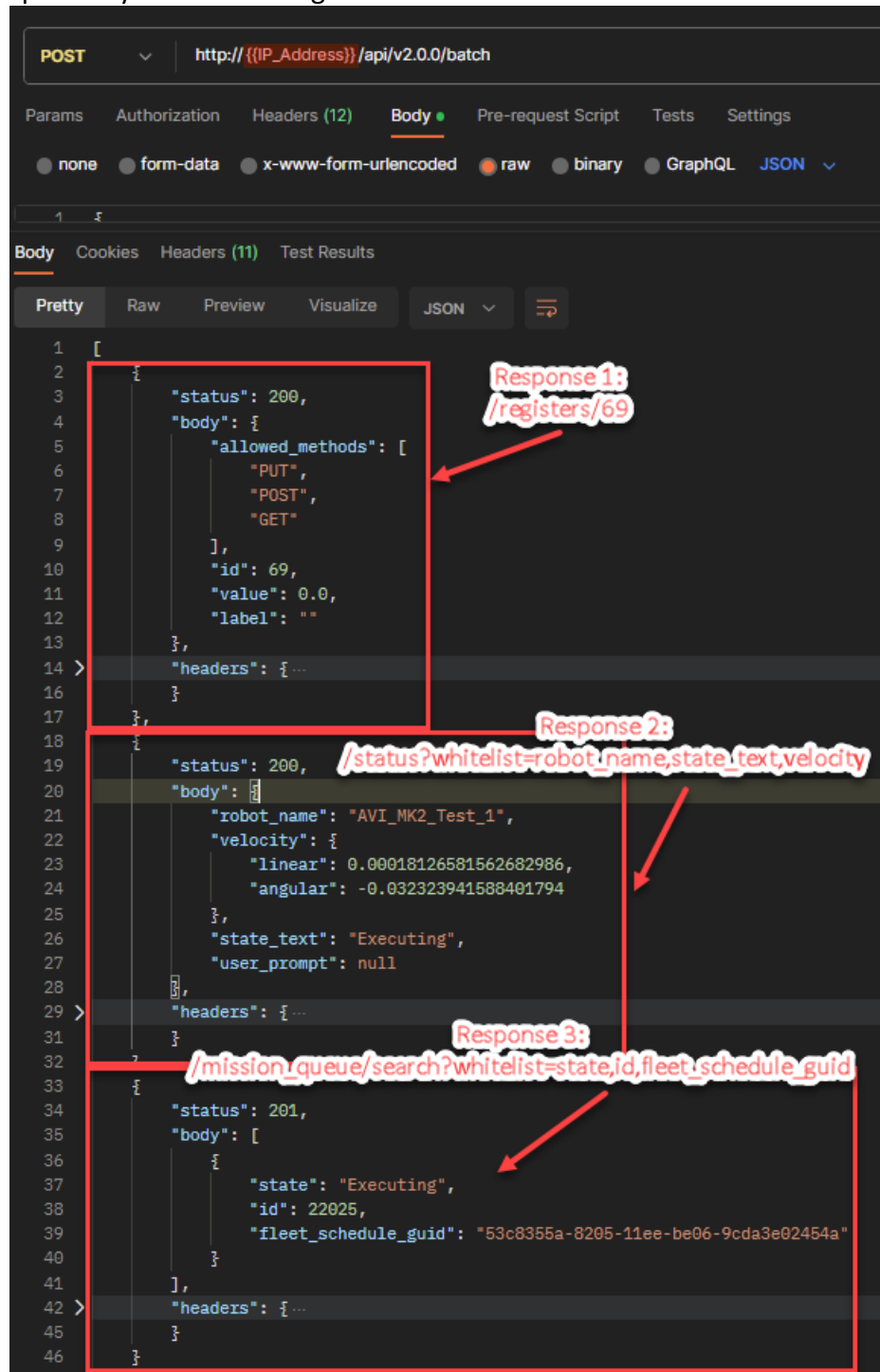


Figure 11 - /batch call response data