strating PURELOCATE

User Document

Pure Locate API for Version 1.0

Overview

The REST API for PureLocate offers access to the data for integration into other systems and interfaces. This is what is used for the web app, mobile and desktop apps, as well as server services. This is for version 1.x of the API.

High Level Structure:

- Users
  - Login and JWT Token Generation
- Organization
  - Information about the organization
- Locations
  - Name, information, GPS position
- Floors
  - Floorplans, geoJSON, GPS position
- Devices
  - Name, service (BASE_STATION, TAG, PHONE), GPS position, short history, raw data

Resources

Postman Generated Documentation & Web Version of Postman Endpoints (easy copy/paste commands in any language:
https://documenter.getpostman.com/view/3189822/SzmY81JN

Postman Collection :
Holds all endpoint routes and configurations
https://drive.google.com/file/d/1WiE3_0FzttwgYz0yfkBPtHp5oGEVHknk/view?usp=sharing

Postman Environment :
Handles variables such as URL (app.purelocate.com), User, and JWT token used in transactions after login
https://drive.google.com/file/d/1d6XPrZwu77iNMIPCidTJZNIgEPK5e1wK/view?usp=sharing

Note that postman IS NOT a solution to using the API, it's just a convenient way to exercise the API endpoints, get the necessary requests structured, and see example responses. In this way, you can confidently “wire up” your client to use the API knowing what kind of requests and responses to expect.

Any client library in any language supporting HTTPS requests can be used to get information from the API.

Sections

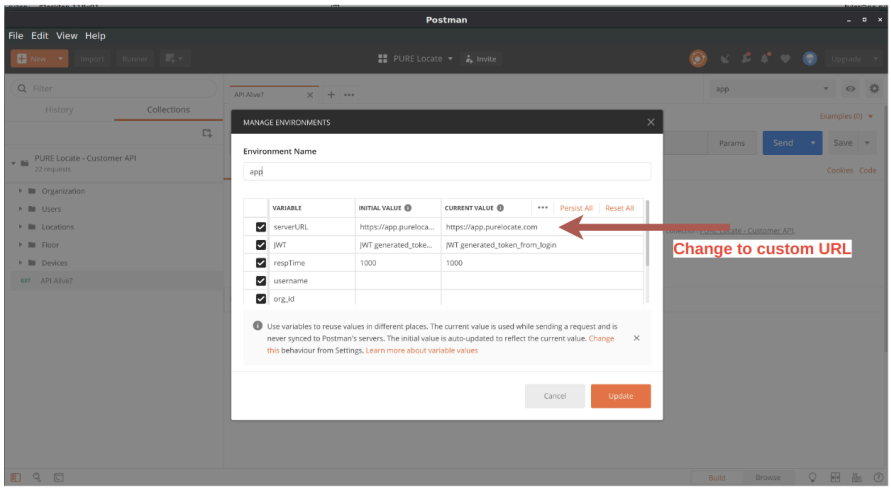
Setting up Postman

Postman makes the process of testing the API much easier to have a known path of communication, before integrating into your custom environment. This allows you as a developer to see all transactions, requirements for authentication, generated code output for several languages, as well as a quick tool to view the structure of response data.

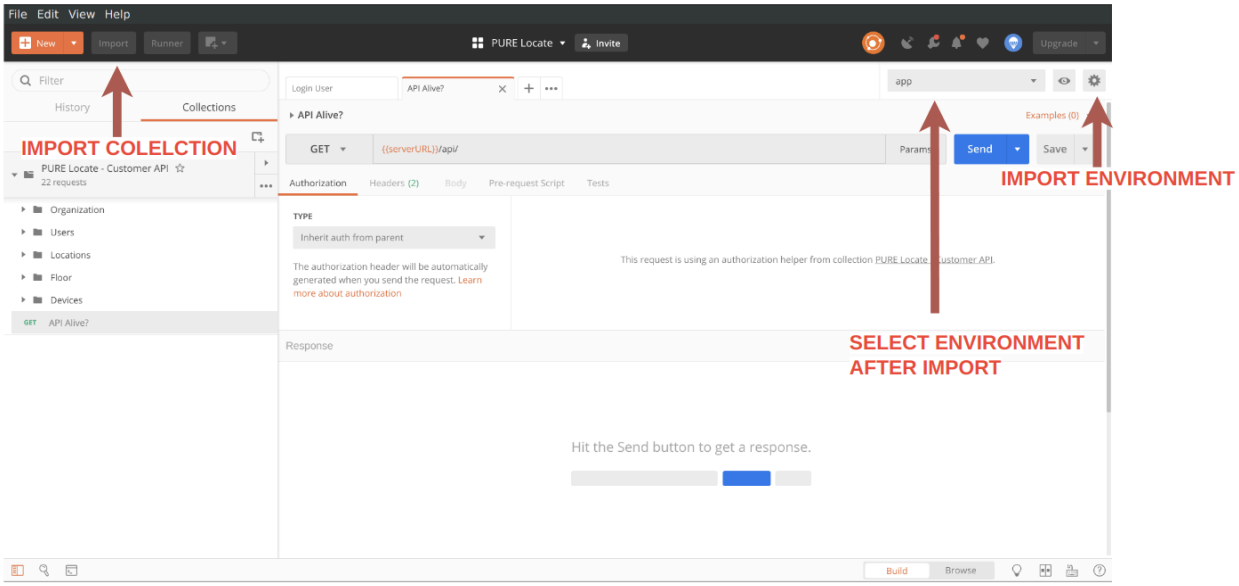
As a quick look around the API, you can see the web version : https://documenter.getpostman.com/view/3189822/S1EJWLBA?version=latest#81ec5e67-942d-41f8-8040-30faf8883dd0

To get the full desktop tool and configuration:

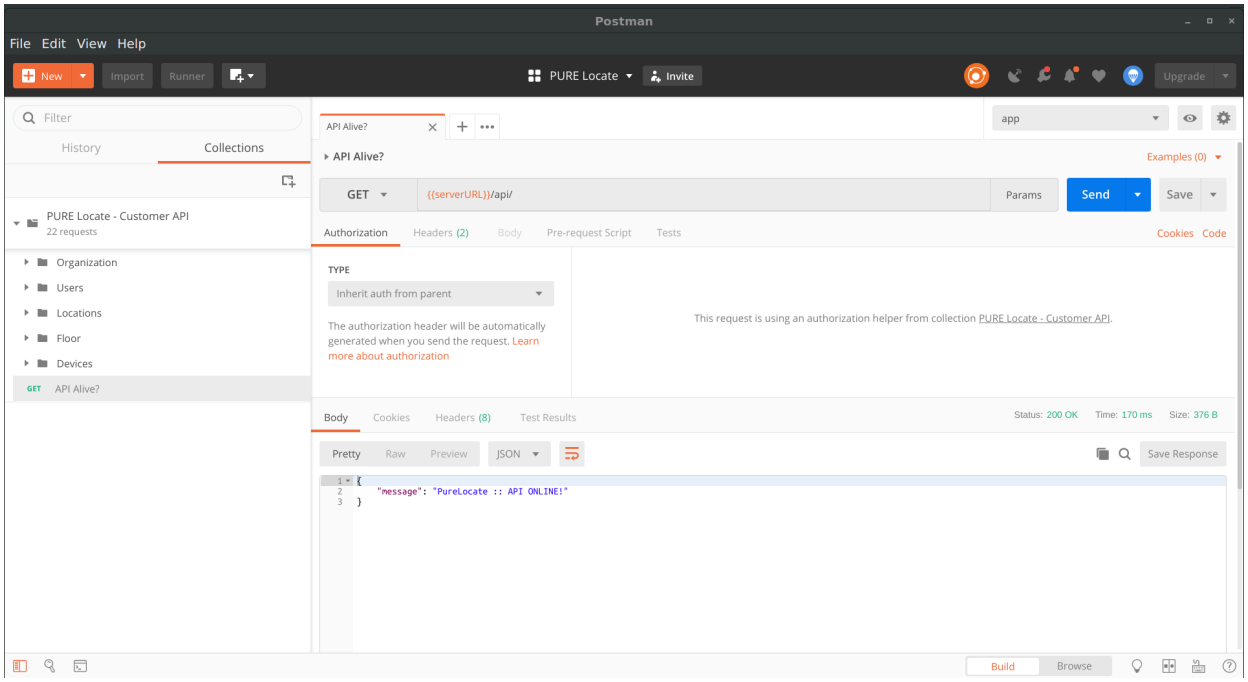
1. Download Postman : https://www.getpostman.com/downloads/
2. Download the collection : https://drive.google.com/file/d/1WiE3_0FzttwgYz0yfkBPtHp5oGEVHknk/view?usp=sharing
3. Download the environment : https://drive.google.com/file/d/1d6XPrZwu77iNMIPCidTJZNIgEPK5e1wK/view?usp=sharing
4. Import the collection : Top left corner, import, select the downloaded PureLocate collection.
5. Import the environment : Top right gear, Import, select the downloaded PureLocate environment.
    a. NOTE : If you have an enterprise release of PureLocate with a Custom URL, you will have to change the URL in the environment variable with your URL and clicking update.



6. Select the environment from the drop down in top right.



7. We can now click “API Alive?” and select Send, The body, we can see the response from the server.



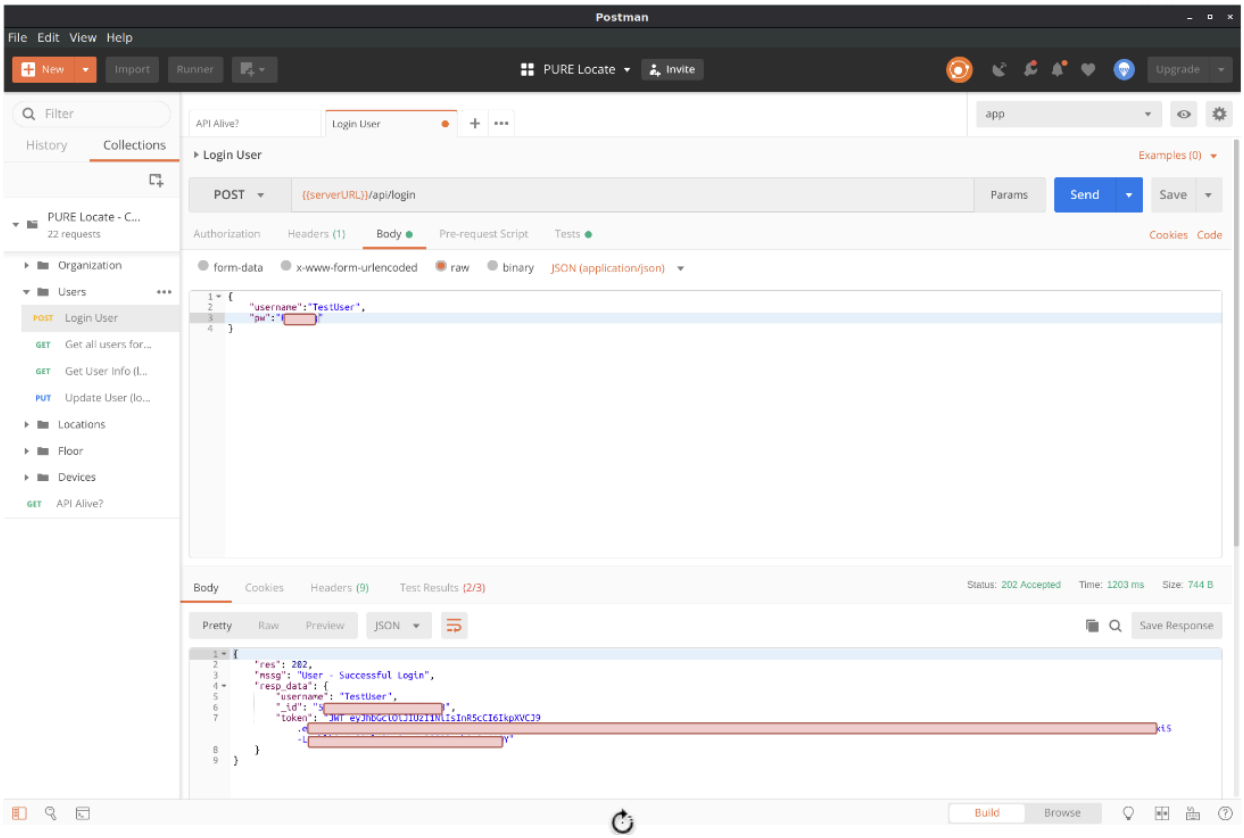
Pure Locate - Logging In

Authentication for PureLocate API endpoints is done using JSON Web Tokens (JWT). In order to request a JWT, a user login is required.

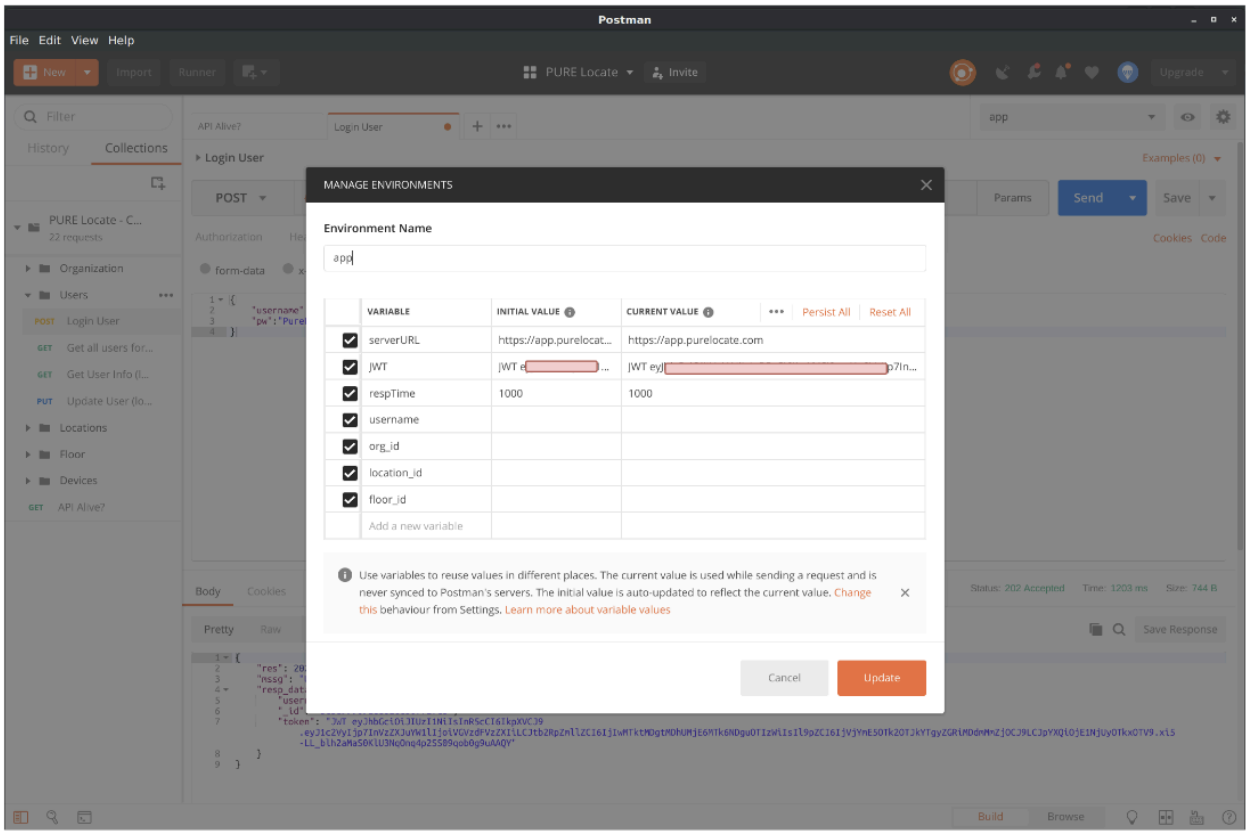
1. Click on User, Login User
2. Click Body
3. Change the JSON to match with your user's credentials.

```
{
      "username":"EnterUsername",
      "pw":"PasswordEntered"
}
```

4. Hit Send
5. The response body includes the JWT and other user information



6. This token is included with all other requests, and in Postman, is added to the environment variables


Postman
File Edit View Help
New
Import
Runner
PURE Locate
Invite
Upgrade
Filter
History
Collections
API Alive?
Login User
app
Examples (0)
POST
Params
Send
Save
Cookies
Code
PURE Locate - C...
22 requests
Organization
Users
POST Login User
GET Get all users for...
GET Get User Info (l...
PUT Update User (lo...
Locations
Floor
Devices
GET API Alive?
MANAGE ENVIRONMENTS
Environment Name
app
VARIABLE
INITIAL VALUE
CURRENT VALUE
Persist All
Reset All
serverURL
https://app.purelocat...
https://app.purelocate.com
JWT
respTime
1000
1000
username
org_id
location_id
floor_id
Add a new variable
Use variables to reuse values in different places. The current value is used while sending a request and is never synced to Postman's servers. The initial value is auto-updated to reflect the current value. Change this behaviour from Settings. Learn more about variable values
Cancel
Update
Body
Cookies
Status: 202 Accepted
Time: 1203 ms
Size: 744 B
Pretty
Raw
Save Response
Build
Browse

PureLocate Organization, Locations, and Floors

PureLocate has a hierarchy concept for organization of assets. This includes your organization, such as your company, Locations such as a building, and floors within this building. All of this can be managed via the website.

GET : Organization

- Returns the organizations details for the logged-in user

GET : Locations

- Returns the organizations locations for the logged-in user
- By ID
  - Returns the location for the given Location ID

GET : Floors

- This includes the floor plan and its GPS coordinates for the image given as a base64 string. These can be uploaded via the website. Useful if building a map of a given floor, the device positions (see PureLocte Devices) can be “overlaid” on the floor plan image.

  - Returns the locations floors for the given Location ID
  - By ID
    - Returns the floor for the given ID

PureLocate Devices

Each block of the PureLocate system can have various operations done. For most customers use of the API, getting device information is the most important. Each device is given a unique ID, as well as using the unique MAC address.

This data includes the name and GPS coordinates of the device. For stations, it gives additional data such as raw scan results, and all payload delivered (has only the last delivered scan). For asset tags, a short history of stations that have seen the tag, as well as calculated position and Last Station that has seen the asset (not necessarily the same).

This GPS information can be used to place assets on a map, import into your existing asset management system, or perhaps use the raw extracted data for custom BLE advertisement collection.

If more is necessary from the device data, please contact PureLocate directly to discuss how we can meet your data needs.

Devices can be asked by:

- From Organization (returns all devices)
- By a location (by location id)
- By a floor (by location floor)
- Single device (by device id)
- Single device (by MAC address)

You can also create a device with a post, you can update with Put (by id or separately by MAC Address):

```
{
      "name":"test",
      "macAddress":"112233445501",
      "service":"TAG",
      "location":"5ca6473fb3246fe7c5b0ff6b",
      "floor":"5ca6476ab3246f013bb0ff6c",
      "data": {"SETUP":"POSTMAN"}
}
```

Example Response : Devices

GET Request to Devices From Organization. This particular organization has one **Station** and one **Asset Tag**:


```
{
    "res": 200,
    "mssg": "Devices Found",
    "resp_data": [
        {
            "last_location": {
                "type": "Point",
                "coordinates": [
                    00.00,
                    -00.00
                ]
            },
            "keyValueList": [],
            "_id": "0000000",
            "macAddress": "000000000000",
            "__v": 1,
            "created": "2019-05-29T04:26:34.003Z",
            "floor": "0000",
            "location": "0000",
            "modified": "2019-05-29T04:26:34.003Z",
            "name": "Test Station",
            "onboarded": true,
            "organization": "00000",
            "service": "BASE_STATION",
            "data": {
                "macAddress": "b4e62dc1ef75",
                "message": {
                    "wifiRecord": [],
                    "bleRecord": [
                        {
                            "macAddress": {
                                "0": 0,
                                "1": 0,
                                "2": 0,
                                "3": 0,
                                "4": 0,
                                "5": 0
```

```
                        },
                        "rssi": -83,
                        "advPacket": {
                            "0": 2,
                            ....
                        },
                        "rspPacket": ""
                    },
                ],
                "macAddress": {
                    "0": 00,
                    "1": 00,
                    "2": 00,
                    "3": 00,
                    "4": 00,
                    "5": 00
                },
                "systemRecord": {
                    "silicon": 1,
                    "version": {
                        "0": 0,
                        "1": 1,
                        "2": 30
                    },
                    "nextOta": "0.0.0.bin",
                    "uptime": 0,
                    "rssi": 0
                },
                "timestamp": 1565299820904
            }
        },
        "last_seen": "2019-08-08T21:30:20.904Z",
        "last_altitude": -00.00
    },
    {
        "last_location": {
            "type": "Point",
            "coordinates": [
                00.00,
                -00.00
            ]
        },
```

```
            "keyValueList": [],
            "_id": "000000000000",
            "macAddress": "000000000000",
            "__v": 0,
            "created": "2019-05-04T17:43:20.481Z",
            "floor": "000000000000",
            "location": "000000000000",
            "modified": "2019-05-04T17:43:20.481Z",
            "name": "Test Bluetooth Asset",
            "organization": "000000000000",
            "service": "TAG",
            "data": {
                "last_station": {
                    "name": "Test Station",
                    "_id": "000000000000",
                    "macAddress": "000000000000",
                    "timestamp": 1556502732672,
                    "rssi": -50,
                    "latitude": 000000000000,
                    "longitude": -000000000000,
                    "altitude": 0,
                    "raw_report":
"{\"macAddress\":\"mQBw6nM4\",\"rssi\":-92,\"rspPacket\":\"BAlwbmc=\"}",
                    "accuracy": 0,
                    "location_id": "000000000000",
                    "location_name": "Test Location",
                    "location_entered_timestamp": 000000000000
                },
                "calculated_position": {
                    "name": "Test Station",
                    "_id": "000000000000",
                    "macAddress": "000000000000",
                    "timestamp": 1556502732672,
                    "rssi": -50,
                    "latitude": 000000000000,
                    "longitude": -000000000000,
                    "altitude": 0,
                    "raw_report":
"{\"macAddress\":\"mQBw6nM4\",\"rssi\":-92,\"rspPacket\":\"BAlwbmc=\"}",
                    "accuracy": 0,
                    "location_id": "000000000000",
                    "location_name": "Test Location",
```

```
                    "location_entered_timestamp": 000000000000
                },
                "history": [
                    {
                        "name": "Test Station",
                        "_id": "000000000000",
                        "macAddress": "000000000000",
                        "timestamp": 1556502732672,
                        "rssi": -50,
                        "latitude": 000000000000,
                        "longitude": -000000000000,
                        "altitude": 0,
                        "raw_report":
"{\"macAddress\":\"mQBw6nM4\",\"rssi\":-92,\"rspPacket\":\"BAlwbmc=\"}",
                        "accuracy": 0,
                        "location_id": "000000000000",
                        "location_name": "Test Location",
                        "location_entered_timestamp": 000000000000
                    }
                ]
            },
            "last_seen": "2019-04-29T01:52:12.672Z",
            "details": ""
        }
    ]
}
```

```
{
    "res": 200,
    "mssg": "Found",
    "resp_data": {
        "level": 0,
        "floor_plan_img_b64": [],
        "floor_plan_img_url": [],
        "unity": [],
        "_id": "5d25203ea12da30022d4ef20",
        "location": "5d25200fa12da30022d4ef1f",
        "name": "KF04F1",
        "__v": 0,
        "created": "2019-07-09T23:16:14.354Z",
        "modified": "2019-08-08T00:23:48.575Z",
        "organization": "5d251fbaa12da30022d4ef1e",
        "floor_plan_imgJson": {
            "cornerOne": [
                37.37417094406119,
                -121.99905395507814
            ],
            "cornerTwo": [
                37.375289983842904,
                -121.99793413281442
            ],
            "image": "......<base64 imaage>...........AP+gv"
        },
        "id": "5d25203ea12da30022d4ef20"
    }
}
```