PlantIQ

REST API Documentation

This document outlines the current REST API endpoints available and any requirements for their use.

Created by Jack Harris & Alex Vorraro for PP2307_T2_RS (PlantIQ - Team 2)

Table Of Contents

Section 1(Introduction) 2

Unauthenticated respoonses 3

Authenticating with Insomnia 3

Section 2 (Authentication) 4

Loggin In 5

Logging Out 7

Validating Session Tokens 9

Registering a new account 10

Activating a new account 12

Generating a password reset token 12

Validating a password reset token 14

Consuming a password reset token 15

Section 3 (Users) 17

Retrieving users information 18

Updating users information 19

Section 4 (Smart Home Hubs) 21

List all smart home hubs 22

Registering a new smart home hub 24

Updating a smart home hub 26

Deleting a smart home hub 28

Get post frequnecy 29

Section 5 (Plants) 30

Retrieving plant data 31

Submitting plant data 33

Listing all a users plants 35

Getting a plants name 36

Setting a plants name 37

Section 6 (Critical Ranges) 39

Getting a smart home hubs critical range 40

Setting a smart home hubs critical range 41

Section 7 (Notifications) 43

Getting all notifications 44

Deleting all notifications 46

Deleting a specific notification 47

Section 8 (Admin Functions) 48

Activating a users account 49

Listing all users 50

Deleting a user 52

Listing all a users smart home hubs 53

Deleting a users smart home hub including all data 55

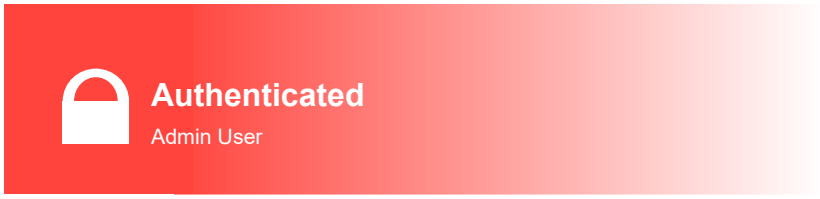
Deleting a users smart home hub data 56

Promoting a user to admin 57

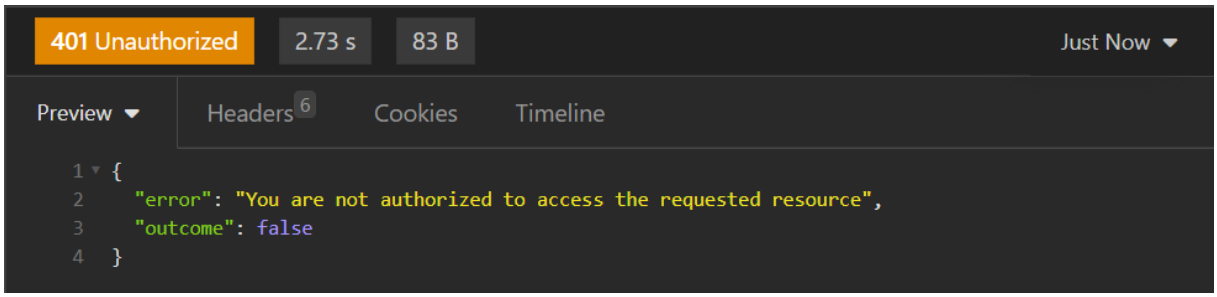
Demoting a admin to user 58

Unauthenticated Response's

The majority of the endpoints in the PlantIQ REST API will require the user to be authenticated or "Logged In" before processing the request, these will be indicated by the following indicators for standard and admin users.



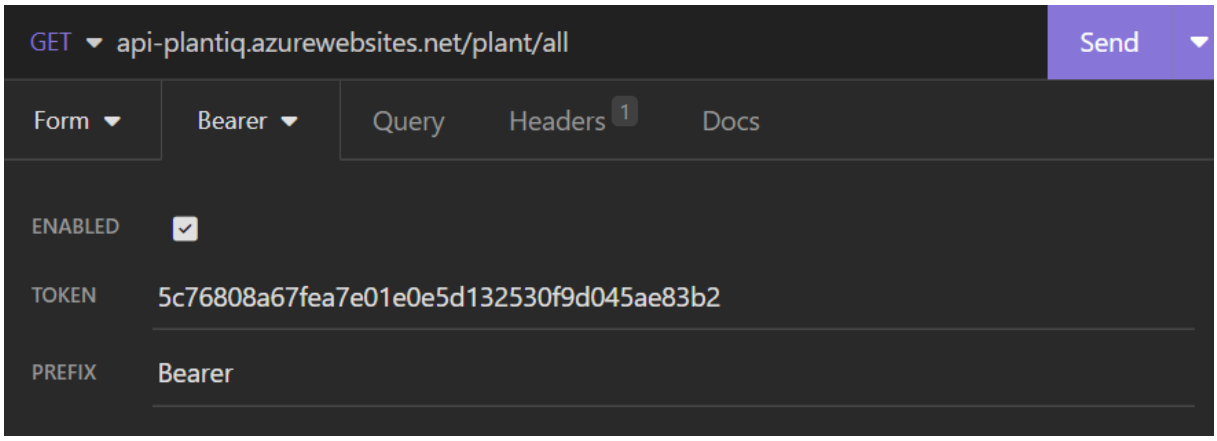
When an endpoint is hit and proper authentication is not provided the API will return the following response.



Authenticating With Insomnia

To authenticate your request first login to the /auth/login endpoint as outlined in this document, then for all following requests simply select "Auth" from the request drop down menu on insomnia and select "Bearer Token", next past in the token that was return from the login.

Congratulations your now authenticated!



Authentication

Loggin In 5
Logging Out 7
Validating Session Tokens 9
Registering a new account 10
Activating a new account 12
Generating a password reset token 12
Validating a password reset token 14
Consuming a password reset token 15

Image Credt: https://www.pexels.com/photo/photography-of-green-leaves-797793/

Loggin In

To login to PlantIQ you will need to provide a valid email address and password as parameters, these should be passed as FORM URL Encoded, the only method support by the /auth/login endpoint is the POST. On a successful login the users session token will be returned.

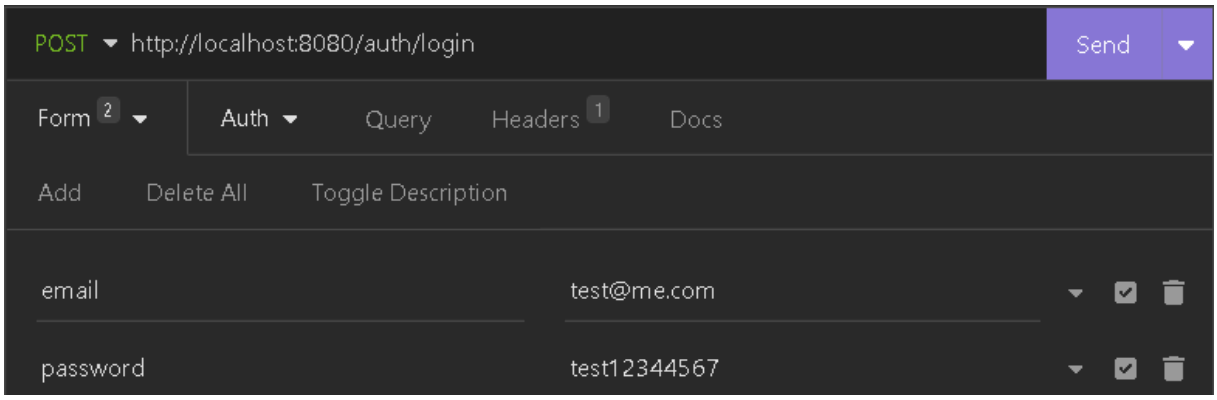
URI:

https://api-plantiq.azurewebsites.net/auth/login POST

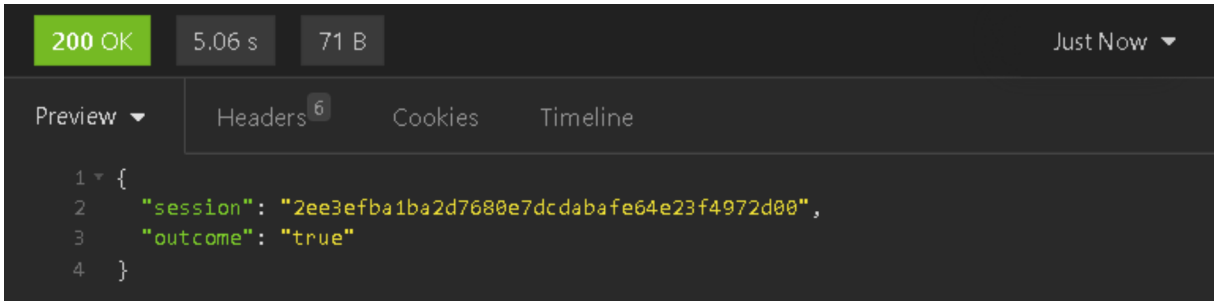
Parameters:

Key	Criteria
email	Required, Regex Email
password	Required, Min 8 chars, Max 25 chars

Example Request:



Example Response (Success):



Example Response (Failed):

401 Unauthorized | 512 ms | 81 B | Just Now

Preview | Headers 6 | Cookies | Timeline

```
{
  "outcome": "false",
  "errors": "Login failed, please check your email and password"
}
```

Example Response (Criteria Failed):

400 Bad Request | 4.42 ms | 153 B | Just Now

Preview | Headers 7 | Cookies | Timeline

```
{
  "outcome": false,
  "errors": {
    "password": {
      "min": "Provided value should be at least 8 characters long"
    },
    "email": {
      "regex": "Validation failed for key email"
    }
  }
}
```

Logging Out

Whilst a user will be logged out when the max session time expires on a token there may be a number of cases where they wish to logout sooner! To perform a logout all the user has to do is hit the /auth/logout endpoint via the delete method with a valid token provided as a parameter.

Note: As users cannot login to multiple devices at once this logout is sufficient to be considered a logout of all.

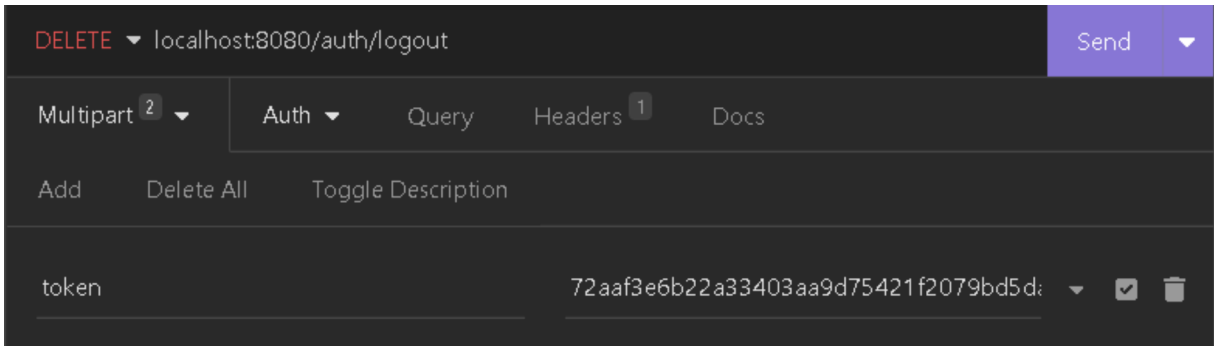
URI:

https://api-plantiq.azurewebsites.net/auth/logout DELETE

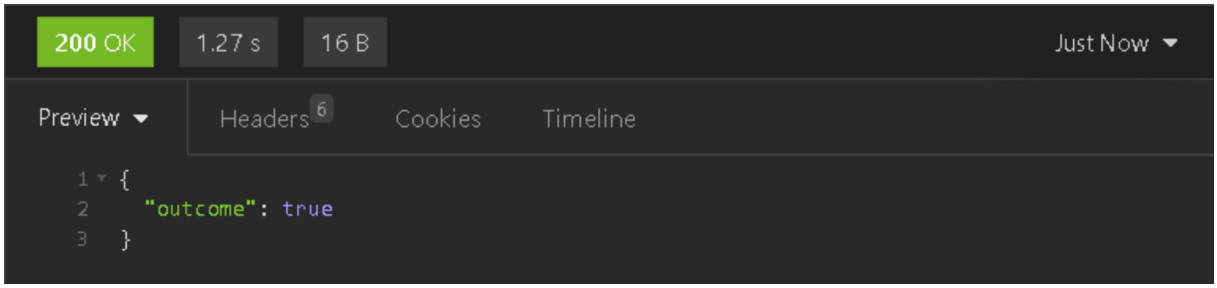
Parameters:

Key	Criteria
token	Required, Min 40 chars, Max 40 chars, Not Unique in Sessions table

Example Request:



Example Response (Success):



Example Response (Failed):

400 Bad Request | 949 ms | 70 B | Just Now

Preview | Headers 7 | Cookies | Timeline

```
{
  "outcome": false,
  "errors": {
    "token": {
      "unique": "value does not exist"
    }
  }
}
```

Validating Session Tokens

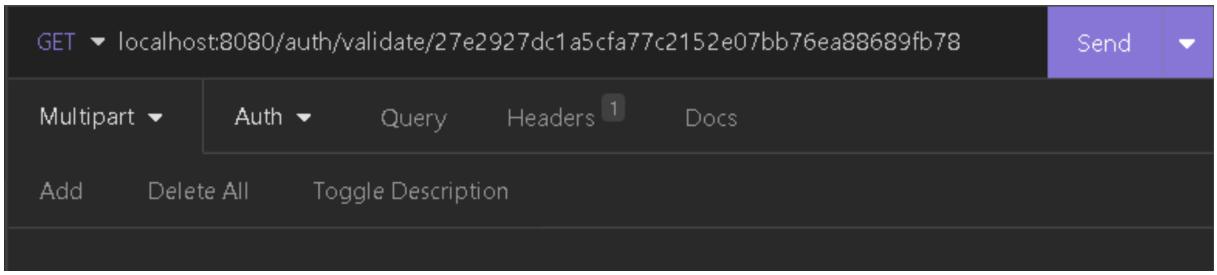
Session tokens can be valided by the front end or any other app via the /auth/validate/{token}.

This endpoint will consume the token and return a true or false back to the caller matching the tokens status, the token will be checked for validity and to ensure its with in the expiry window of two week.

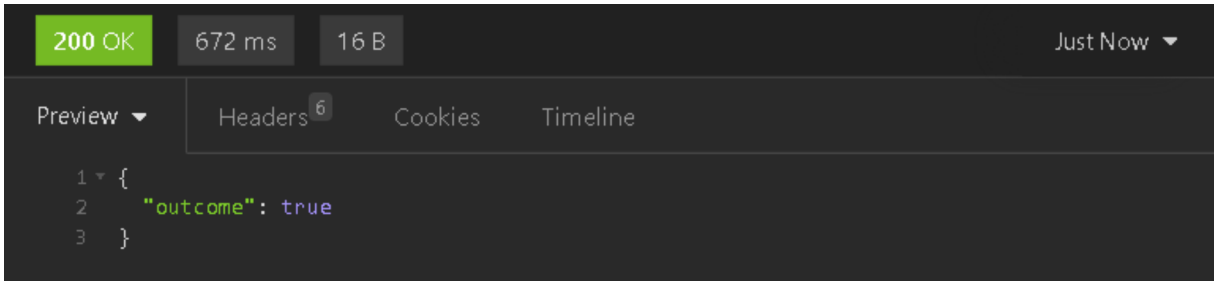
URI:

https://api-plantiq.azurewebsites.net/auth/validate/{token} GET

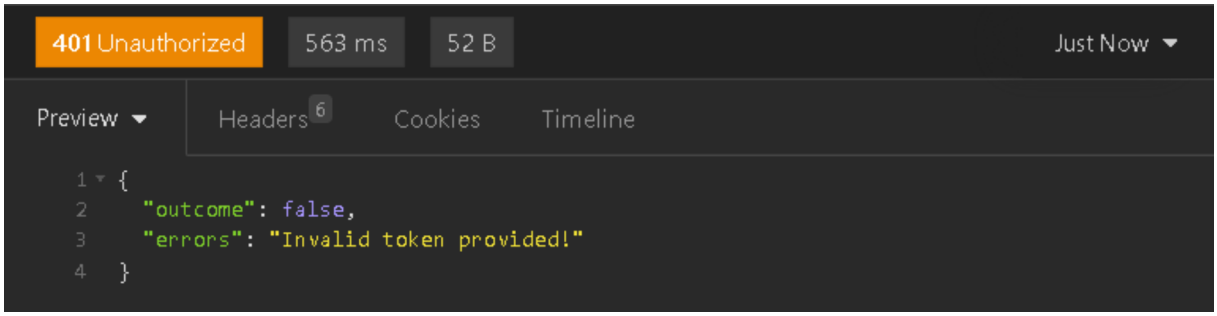
Example Request:



Example Response (Success):



Example Response (Failed):



Register New User

User registration is handled by the /auth/register endpoint, this will accept all the details of a user account and if valid return a outcome of true. User accounts must ensure that they all have a unique email address, this endpoint will prevent user registration if the email address is already taken.

Note: User accounts must be activated after sign up to login and start using the program.

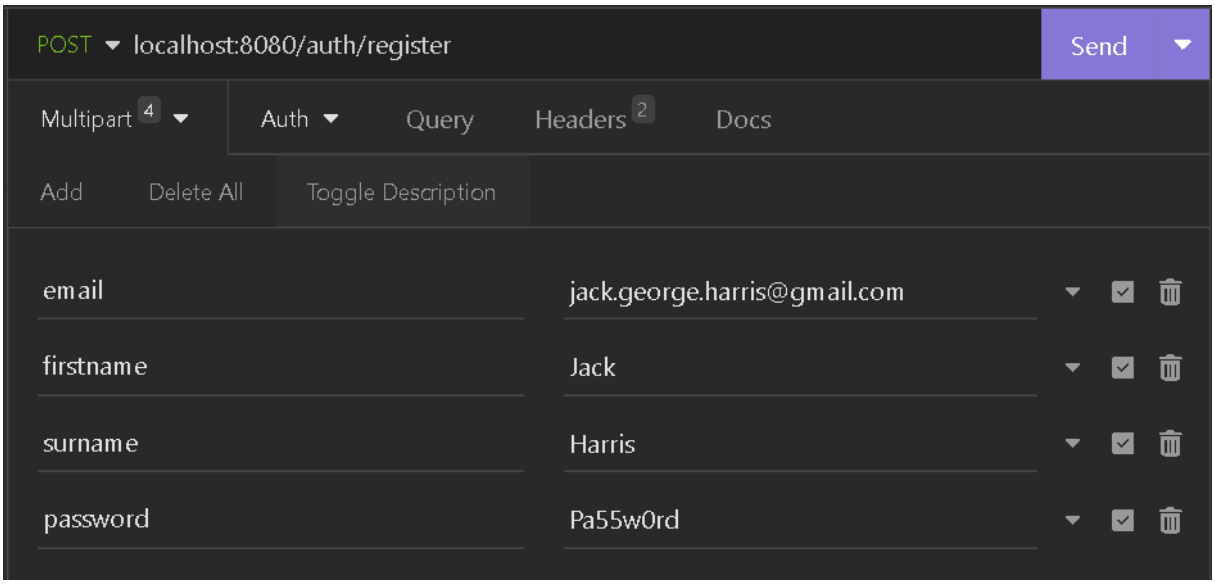
URI:

https://api-plantiq.azurewebsites.net/auth/register	POST

Parameters:

Key	Criteria
email	Required, Regex Email
password	Required, Min 8 chars, Max 25 chars
firstname	Required, Min 3 chars, Max 25 chars
surname	Required, min 3 chars, Max 25 chars

Example Request:



Example Response (Success):

200 OK | 2.4 s | 81 B | 2 Minutes Ago

Preview | Headers 6 | Cookies | Timeline

```
1 {
2   "message": "User successfully registered and awaiting activation",
3   "outcome": true
4 }
```

Example Response (Failed):

400 Bad Request | 748 ms | 90 B | Just Now

Preview | Headers 7 | Cookies | Timeline

```
1 {
2   "outcome": false,
3   "errors": {
4     "email": {
5       "unique": "value is already taken and is not unique"
6     }
7   }
8 }
```

Activating An Account

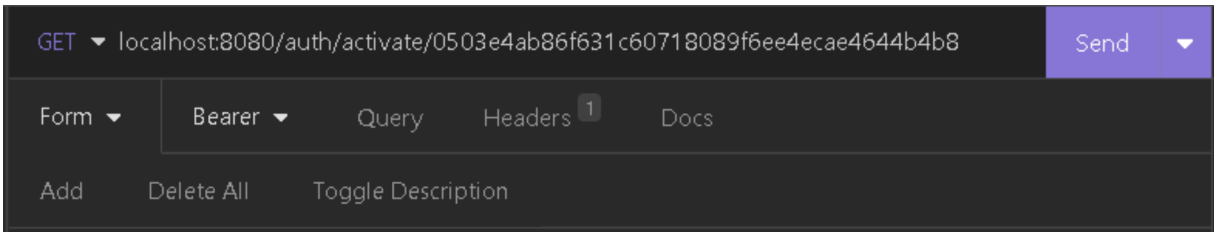
User registration is handled by the /auth/register endpoint, this will accept all the details of a user account and if valid return a outcome of true. User accounts must ensure that they all have a unique email address, this endpoint will prevent user registration if the email address is already taken.

Note: User accounts must be activated after sign up to login and start using the program.

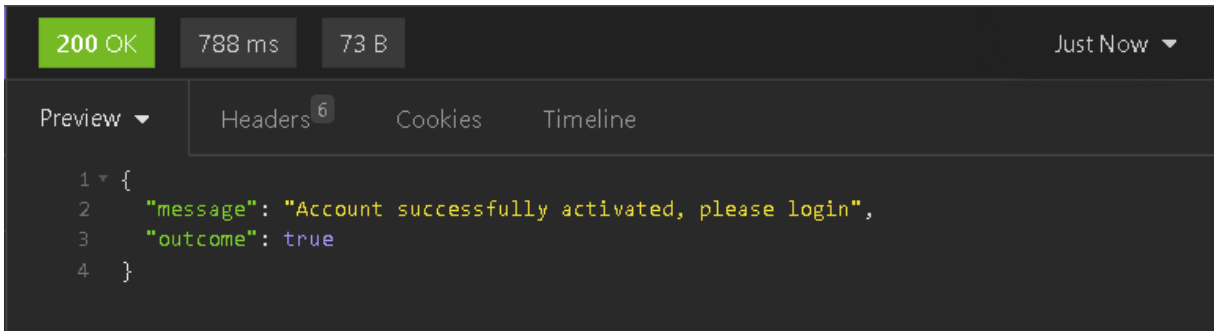
URI:

https://api-plantiq.azurewebsites.net/auth/activate/{token} POST

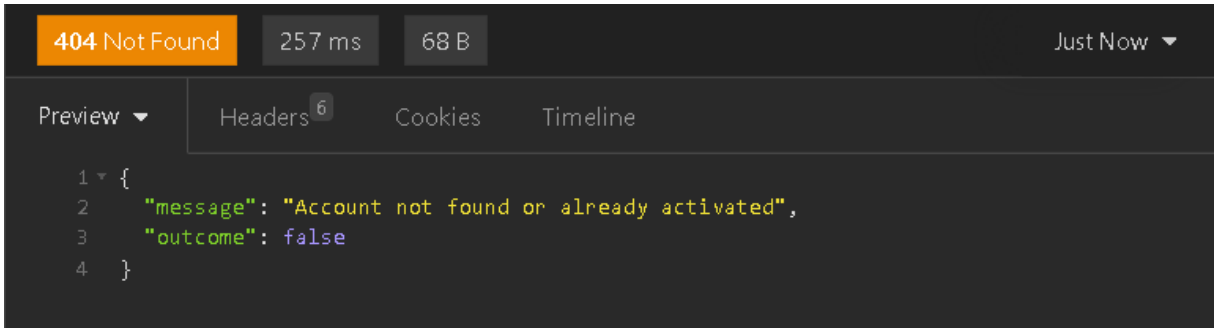
Example Request:



Example Response (Success):



Example Response (Failed):



Generating A Password Reset Token

To reset a users password they must generate a password reset link, to do this they can hit the endpoint /auth/reset using the POST method. This will simply require a valid email address. If a invalid email is provided the system will not tell you, this is important from a secuirty perspective.

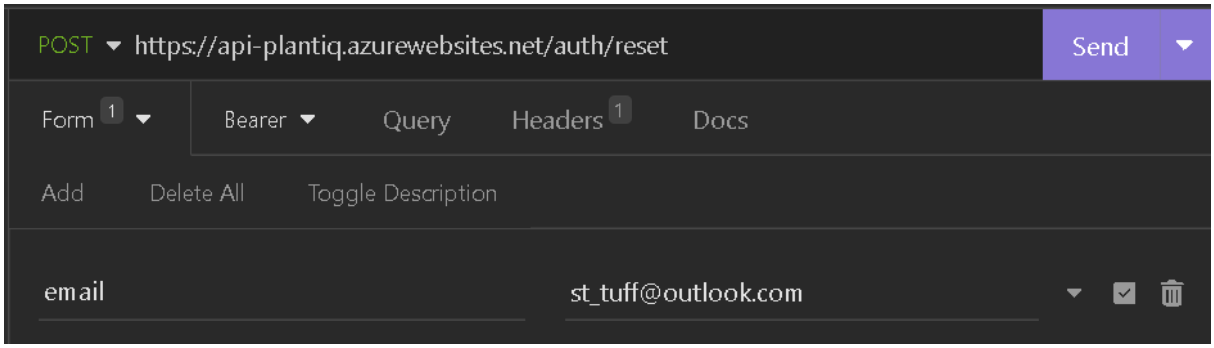
URI:

https://api-plantiq.azurewebsites.net/auth/reset POST

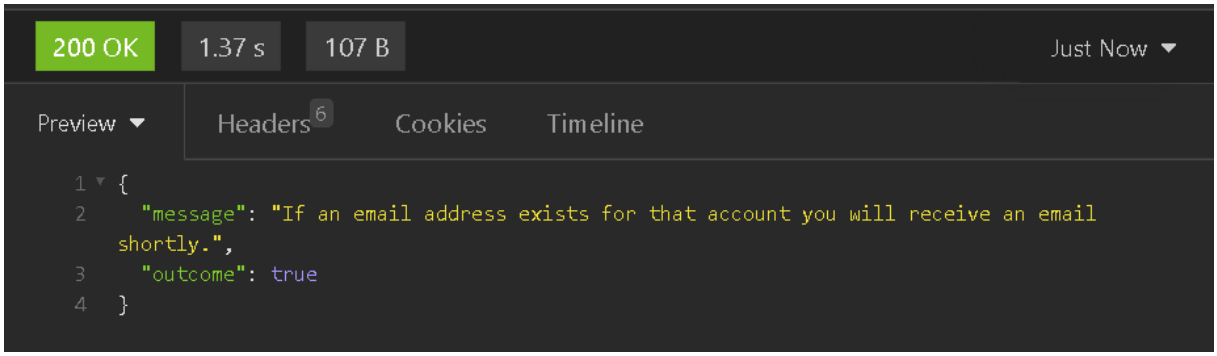
Parameters:

Key	Criteria
email	Required, Regex Email

Example Request:



Example Response (Success & Failed):



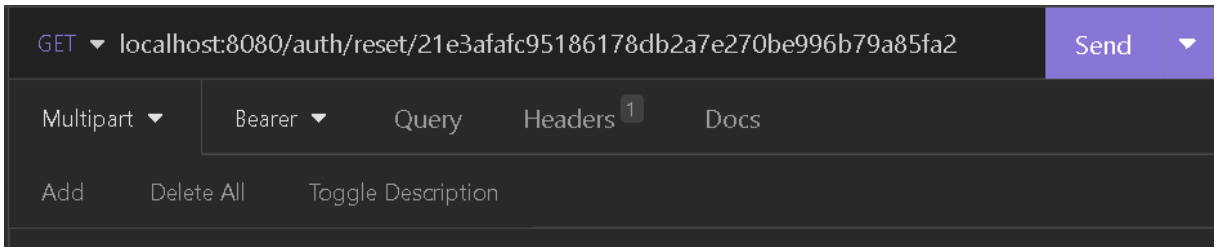
Validating A Password Reset Token

To make front end validation easier password reset links can be validated prior to be used or “consumed”, to validate a link simply hit the /auth/reset/{token} endpoint, this will return true or false representing the tokens validity.

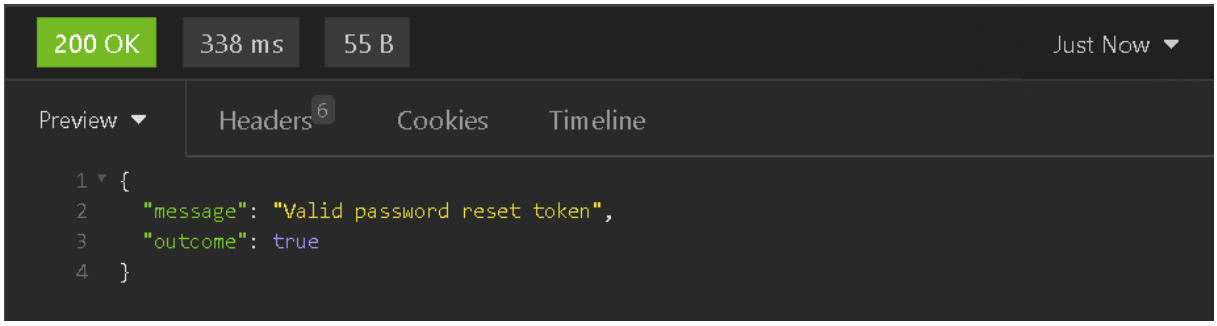
URI:

https://api-plantiq.azurewebsites.net/auth/reset/{token} GET

Example Request:



Example Response (Success):



Example Response (Failed):

Consuming A Password Reset Token

To reset a users password they must generate a password reset link, to do this they can hit the endpoint /auth/reset using the POST method. This will simply require a valid email address. If a invalid email is provided the system will not tell you, this is important from a secuirty perspective.

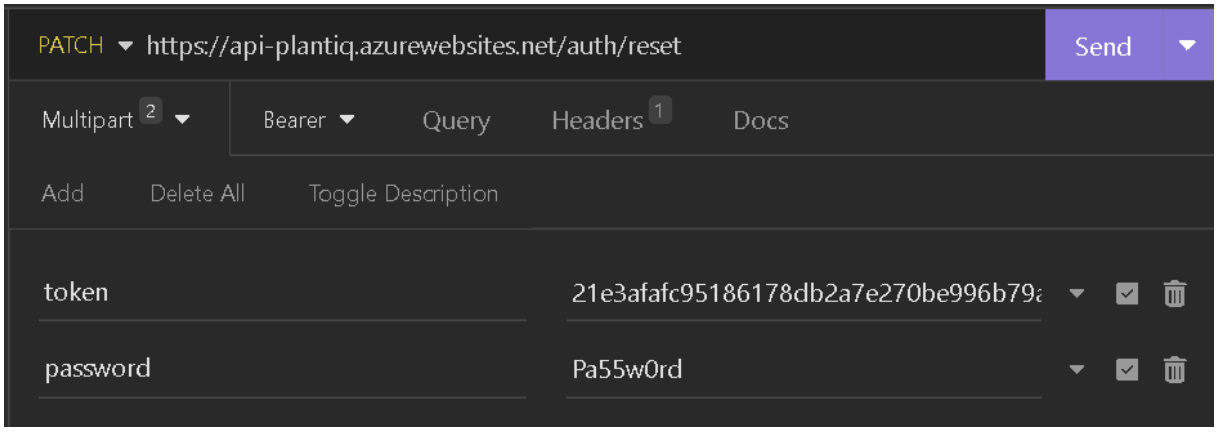
URI:

https://api-plantiq.azurewebsites.net/auth/reset PATCH

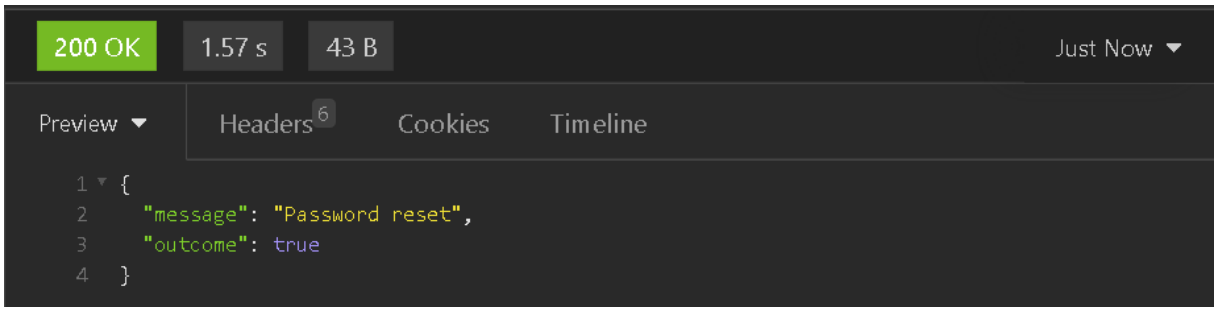
Parameters:

Key	Criteria
password	Required, Regex Email
token	Required, Min 8 chars, Max 50 chars

Example Request:



Example Response (Success):



Example Response (Failed):

400 Bad Request | 505 ms | 70 B | Just Now

Preview | Headers 6 | Cookies | Timeline

```
1 {
2   "outcome": false,
3   "errors": {
4     "token": {
5       "unique": "value does not exist"
6     }
7   }
8 }
```

Users

Retrieving users information 18
Updating users information 19

Image Credt: https://www.pexels.com/photo/green-and-white-leaf-plant-2316571/

Getting A Users Information

Once a user has successfully logged-in, it will be possible for them to visualise the data of their account by hitting /user/info via get method.

Note: This endpoint will not return the users password.

URI:

https://api-plantiq.azurewebsites.net/user/info GET

Example Request:

GET https://api-plantiq.azurewebsites.net/user/info Send

Example Response (Success):

200 OK 7.31 s 185 B Just Now

Preview Headers 6 Cookies Timeline


```
{
  "data": {
    "id": "a19bf9ac2f08c748b030dcaeabb2817d1cf255c0",
    "firstname": "Alex",
    "surname": "Vorraro",
    "activated": true,
    "administrator": false,
    "email": "alexvorraro21@gmail.com"
  },
  "outcome": true
}
```


Example Response (Failed):

401 Unauthorized 489 ms 83 B Just Now

Preview Headers 6 Cookies Timeline


```
{
  "error": "You are not authorized to access the requested resource",
  "outcome": false
}
```

Updating A Users Information

If the user has inserted wrong details or if these have changed, for a logged-in user it will be possible to update the data of their account.

Note: The user will only be able to change their first name, surname, email and password.

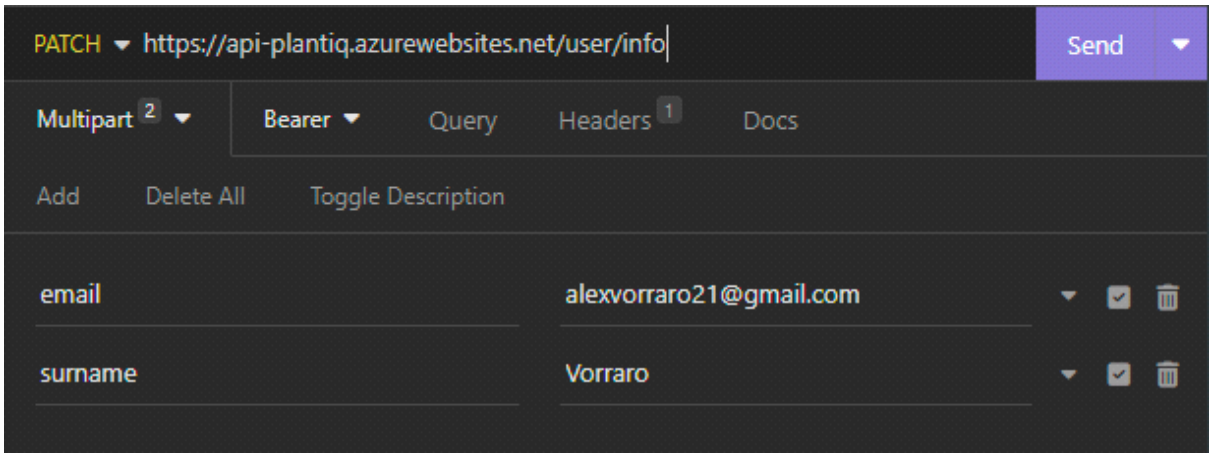
URI:

https://api-plantiq.azurewebsites.net/user/info	PATCH

Parameters:

Key	Criteria
email	Optional, Regex Email, Unique to user
firstname	Optional, Min 3 chars, Max 25 chars
surname	Optional, Min 2 chars, Max 25 chars
password	Optional, Min 8 chars, Max 50 chars

Example Request:



Example Response (Success):

200 OK | 9.62 s | 49 B | Just Now

Preview | Headers 6 | Cookies | Timeline

```
{
  "message": "User details updated",
  "outcome": true
}
```

Example Response (Failed):

400 Bad Request | 799 ms | 304 B | Just Now

Preview | Headers 6 | Cookies | Timeline

```
{
  "outcome": false,
  "errors": {
    "firstname": {
      "max": "Provided value should be no more than 25 characters long"
    },
    "password": {
      "min": "Provided value should be at least 8 characters long"
    },
    "surname": {
      "min": "Provided value should be at least 2 characters long"
    },
    "email": {
      "regex": "Validation failed for key email"
    }
  }
}
```

Smart Home Hub

List all smart home hubs 22
Registering a new smart home hub 24
Updating a smart home hub 26
Deleting a smart home hub 28
Get post frequnecy 29

Image Credt: https://www.pexels.com/photo/green-and-white-leaf-plant-2316571/

Get All Smart Home Hubs

A user may have multiple SmartHomeHubs within their environment. In order to retrieve them all, the logged-in user must hit /all via get method. The returned list will be ordered based on the input the user gives.

The limit parameter will set the maximum of results, the offset parameter will allow the user to skip a certain number of records, the sort by parameter will allow them to order the SmartHomeHubs based on one of their attributes, while sort parameter will indicate if the results will have an ascending or descending sorting.

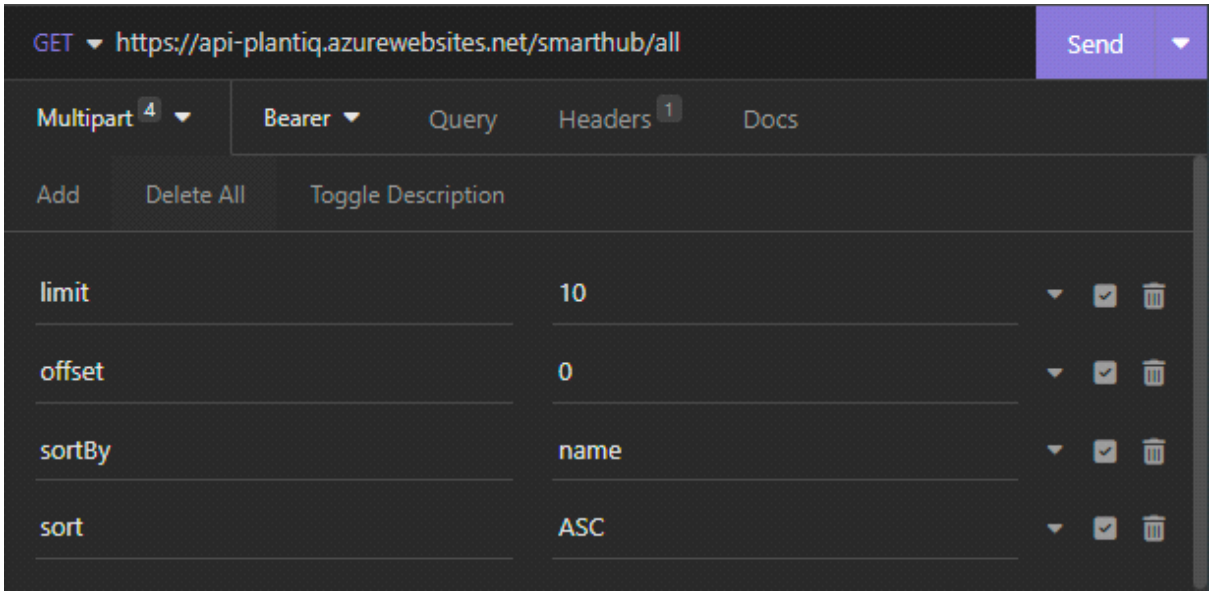
URI:

https://api-plantiq.azurewebsites.net/smarthub/all GET

Parameters:

Key	Criteria
limit	Required, Integer, Min 10, Max 100
offset	Required, Integer, Min 0, Max 100
sortBy	Required, Min 3 chars , Max 25 chars
sort	Required, Sort (must be value of enum either ASC or Desc)

Example Request:



Example Response (Success):

200 OK | 1.02 s | 235 B | 1 Minute Ago

Preview | Headers 6 | Cookies | Timeline


```
{
  "list": [
    {
      "name": "123456789126",
      "id": "123456789126",
      "postFrequency": 3600,
      "lastPosted": 1683938619,
      "virtual": false
    },
    {
      "name": "123456789128",
      "id": "123456789128",
      "postFrequency": 3600,
      "lastPosted": 1683941012,
      "virtual": false
    }
  ],
  "outcome": true
}
```


Example Response (Failed):

400 Bad Request | 1.27 s | 302 B | Just Now

Preview | Headers 6 | Cookies | Timeline


```
{
  "outcome": false,
  "errors": {
    "offset": {
      "integer": "Invalid type provided, input must be a valid integer"
    },
    "limit": {
      "min": "Provided value should be at least 10"
    },
    "sortBy": {
      "min": "Provided value should be at least 3 characters long"
    },
    "sort": {
      "enum": "Invalid sort type provided, sort must match ASC, DESC"
    }
  }
}
```

Registering A Smart Home Hub

Each user should have a SmartHomeHub per area. In order to get it fully functioning, this must be registered. Once the user logged-in and the SmartHomeHub has been added to the awaiting list, hitting the endpoint /register via post method will send the confirmation email to the user containing the link to finalize the registration.

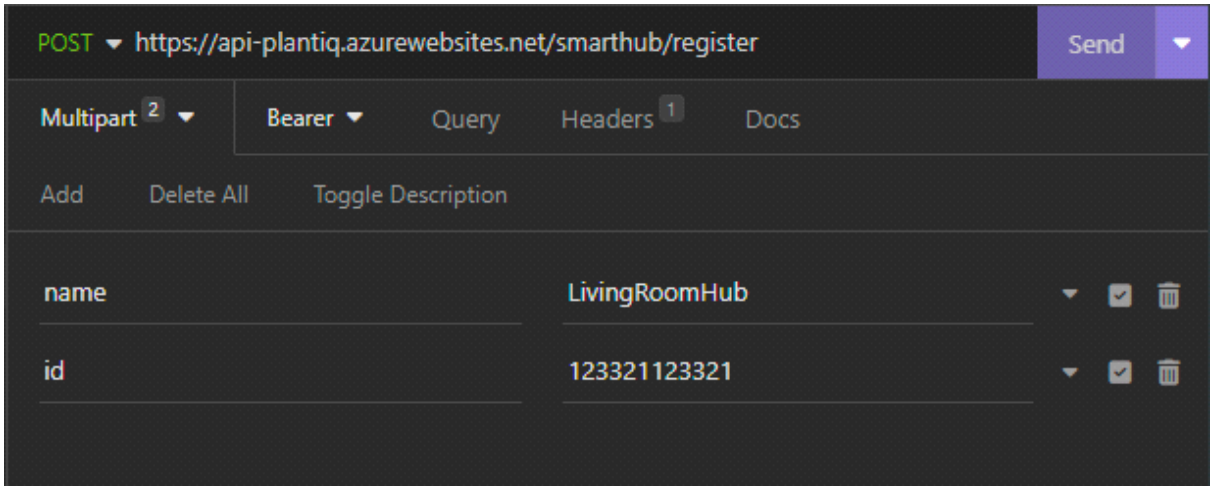
URI:

https://api-plantiq.azurewebsites.net/smarthub/register	POST

Parameters:

Key	Criteria
id	Required, Exactly 12 chars, Currently Awaiting Registration
name	Required, Min 5 chars , Max 50 chars

Example Request:



Example Response (Success):

200 OK | 5.4 s | 60 B | Just Now

Preview | Headers 6 | Cookies | Timeline

```
{
  "message": "Smart Hub registered to account",
  "outcome": true
}
```

Example Response (Failed):

400 Bad Request | 1.64 s | 136 B | Just Now

Preview | Headers 6 | Cookies | Timeline

```
{
  "outcome": false,
  "errors": {
    "name": {
      "min": "Provided value should be at least 5 characters long"
    },
    "id": {
      "unique": "value does not exist"
    }
  }
}
```

Updating A Smart Home Hub

Once a SmartHomeHub has been registered it will still be possible to modify some of its data. By hitting /{id} via patch method, where id is the SmartHomeHub's id, it will be possible to update name and post frequency for a SmartHomeHub.

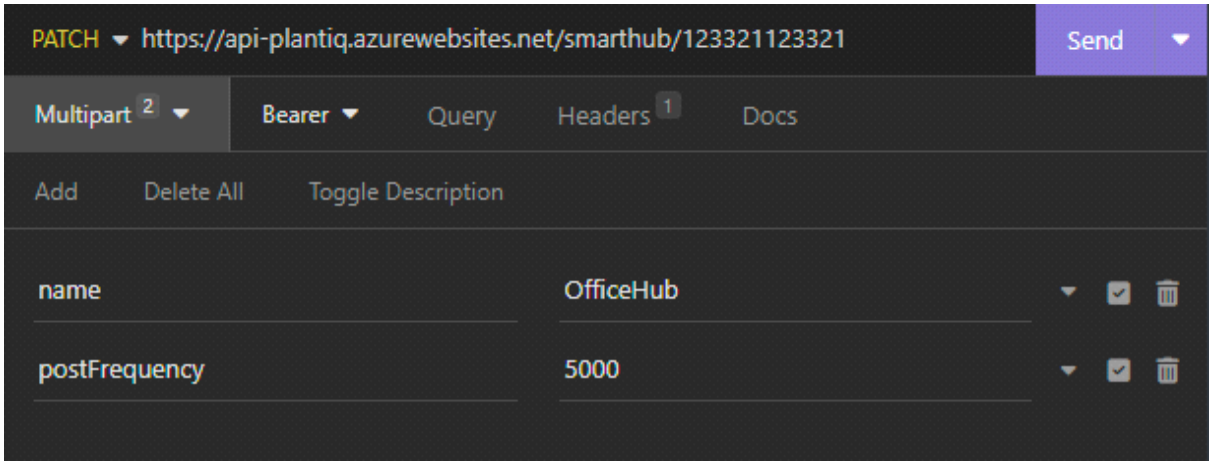
URI:

https://api-plantiq.azurewebsites.net/smarthub/{id} PATCH

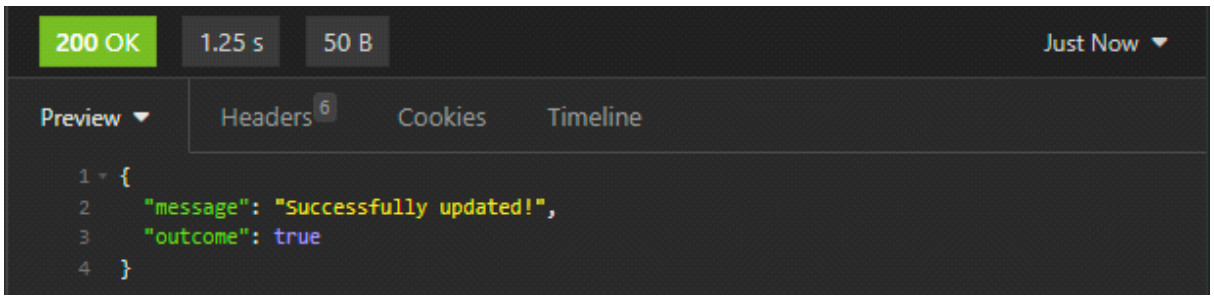
Parameters:

Key	Criteria
name	Required, Min 5 chars , Max 50 chars
postFrequency	Required, Integer

Example Request:



Example Response (Success):



Example Response (Failed):

400 Bad Request | 1.01 s | 180 B | Just Now

Preview | Headers 6 | Cookies | Timeline


```
1  {
2    "outcome": false,
3    "errors": {
4      "name": {
5        "min": "Provided value should be at least 3 characters long"
6      },
7      "postFrequency": {
8        "integer": "Invalid type provided, input must be a valid integer"
9      }
10   }
11 }
```

Deleting A Smart Home Hub

If a SmartHomeHub is not used anymore by a user, they can delete and remove it from the dashboard. By hitting /{id} via delete method, where id is the SmartHomeHub's id, it will be possible to remove the SmartHomeHub and all the plants relating to that hub.

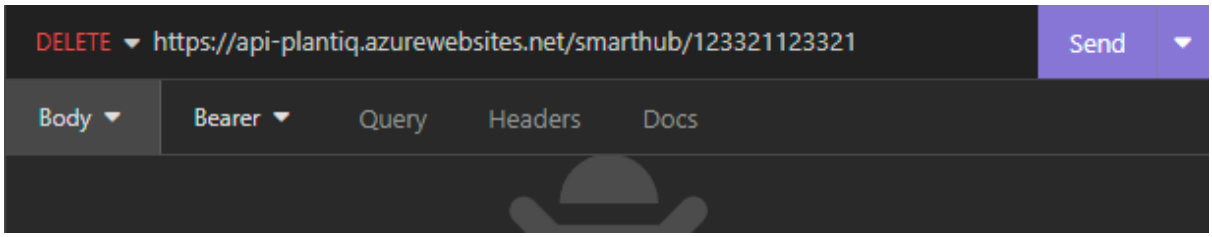
URI:

https://api-plantiq.azurewebsites.net/smarthub/{id} DELETE

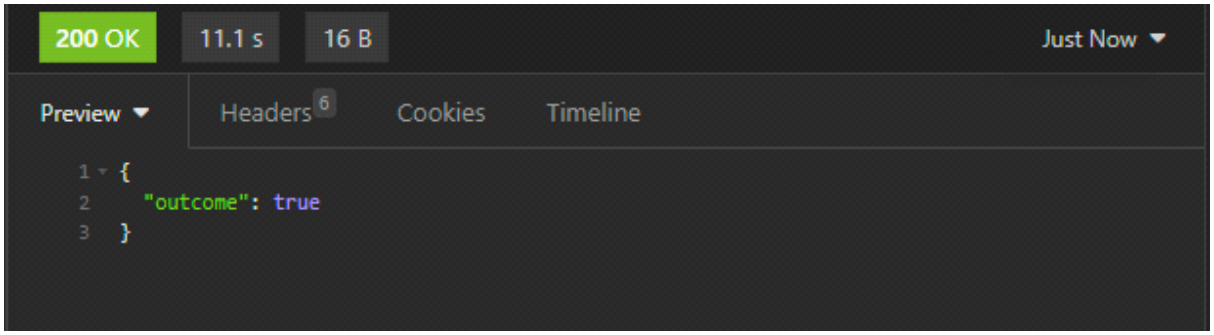
Parameters:

Key	Criteria
name	Required, Min 5 chars , Max 50 chars
postFrequency	Required, Integer

Example Request:



Example Response (Success):



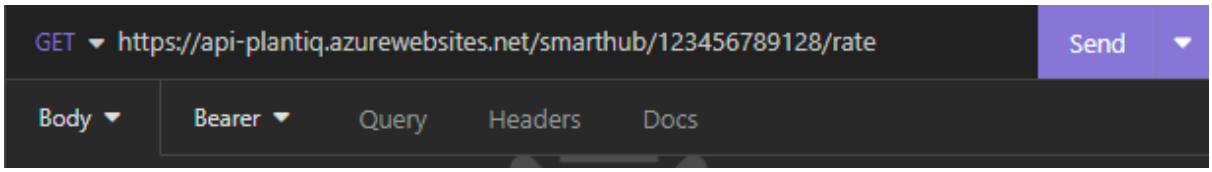
Getting Post Frequency

How often a SmartHomeHub will send data regarding the plants will depend on the set rate. In order to retrieve the rate of a SmartHomeHub the user must hit /{id}/rate via get method, where id is the id of the SmartHomeHub.

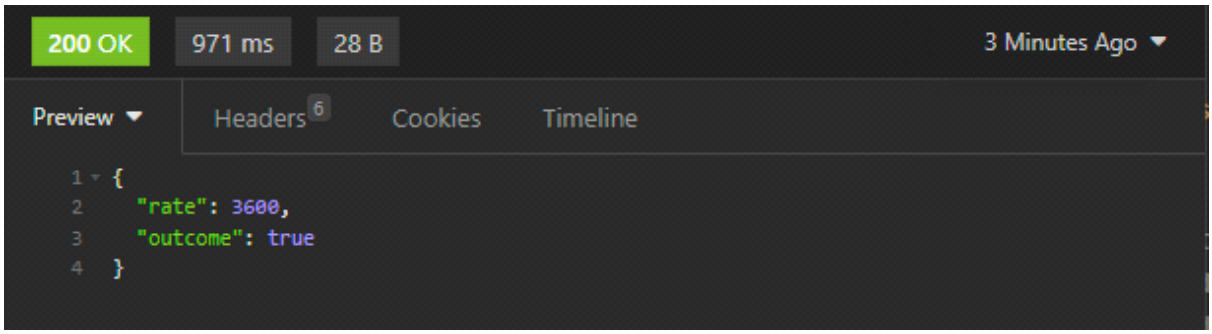
URI:

https://api-plantiq.azurewebsites.net/smarthub/{id}/rate | GET

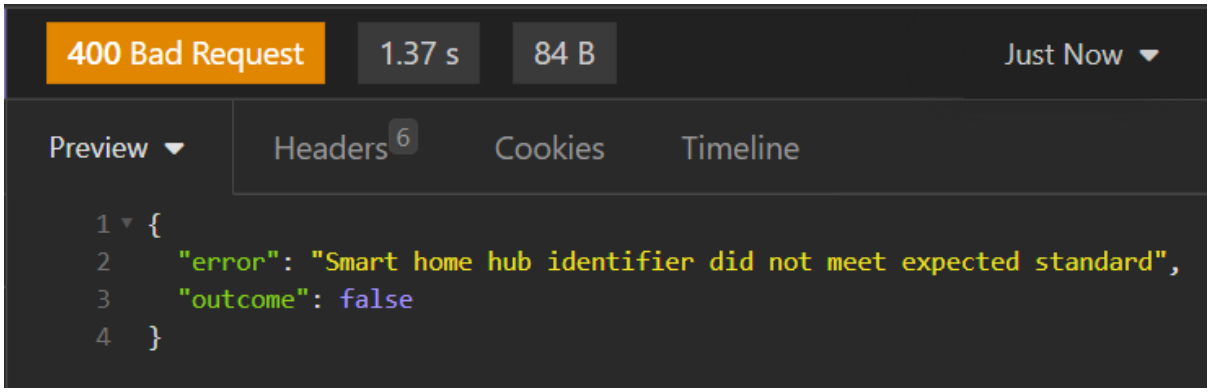
Example Request:



Example Response (Success):



Example Response (Failed):



Plants

Retrieving plant data 31
Submitting plant data 33
Listing all a users plants 35
Getting a plants name 36
Setting a plants name 37

Image Credt: https://www.pexels.com/photo/wooden-bench-and-stools-in-a-cozy-room-3049121/

Retrieving Plant Data

Each SmartHomeHub may have one or more plants related to them. The user will be able to retrieve the data of all the plants relating to their SmartHomeHub by hitting /{id}/data via get method, where the id is the SmartHomeHub id.

URI:

https://api-plantiq.azurewebsites.net/smarthub/{id}/data | GET

Example Request:

GET https://api-plantiq.azurewebsites.net/smarthub/123456789128/data Send

Example Response (Success):

200 OK 1.61 s 663 B Just Now

Preview Headers 6 Cookies Timeline


```
1  {
2    "data": [
3      {
4        "id": 1554,
5        "timestamp": 1683942526,
6        "temperature": 10.0,
7        "humidity": 10.0,
8        "light": 0.0,
9        "moisture": 0.1,
10       "smartHomeHubId": "123456789128",
11       "sensorId": "1111111111111112"
12     },
13     {
14       "id": 3541,
15       "timestamp": 1684915518,
16       "temperature": 10.0,
17       "humidity": 10.0,
18       "light": 0.0,
19       "moisture": 0.1,
20       "smartHomeHubId": "123456789128",
21       "sensorId": "1111111111111112"
22     },
```

Example Response (Failed):

400 Bad Request | 1.37 s | 84 B | Just Now

Preview | Headers 6 | Cookies | Timeline

```
1 {
2   "error": "Smart home hub identifier did not meet expected standard",
3   "outcome": false
4 }
```

Submitting New Plant Data

The user will be able to perform two actions by hitting /{id}/data via post method. If the SmartHomeHub has already been register, hitting this endpoint will add data for one of the plants assigned to the SmartHomeHub and send an email notification to the user if this plant has a value out of the healthy range.

If the SmartHomeHub is not registered, hitting this endpoint will add the SmartHomeHub to the waiting list which is a necessary step prior successful registration.

URI:

https://api-plantiq.azurewebsites.net/smarthub/{id}/data	POST

Parameters:

Key	Criteria
sensor_id	Required, Min 16 chars, Max 16 chars
light	Required, Float
temperature	Required, Float
humidity	Required, Float
moisture	Required, Float

Example Request:

GET https://api-plantiq.azurewebsites.net/smarthub/123456789128/data Send

Example Response (Success):

200 OK | 15.7 s | 16 B | Just Now

Preview | Headers 6 | Cookies | Timeline

```
1 {
2   "outcome": true
3 }
```

Example Response (Failed):

425 Too Early | 7.26 s | 66 B | Just Now

Preview | Headers 6 | Cookies | Timeline

```
1 {
2   "message": "Smart Home Hub Awaiting Registration",
3   "outcome": false
4 }
```

400 Bad Request | 4.04 s | 640 B | 6 Minutes Ago

Preview | Headers 7 | Cookies | Timeline

```
1 {
2   "outcome": false,
3   "errors": {
4     "sensor_id": {
5       "min": "Provided value should be at least 16 characters long",
6       "required": "Validation failed for key sensor_id"
7     },
8     "light": {
9       "float": "Invalid type provided, input must be a valid float",
10      "required": "Validation failed for key light"
11    },
12    "temperature": {
13      "float": "Invalid type provided, input must be a valid float",
14      "required": "Validation failed for key temperature"
15    },
16    "humidity": {
17      "float": "Invalid type provided, input must be a valid float",
18      "required": "Validation failed for key humidity"
19    },
20    "moisture": {
21      "float": "Invalid type provided, input must be a valid float",
22      "required": "Validation failed for key moisture"
23    }
24  }
25 }
```

Listing All User Plants

As the user you may wish to show all your plants across all your smart home hubs, to perform this action you can hit the /plants/all endpoint via the HTTP GET method, this will return the name, sensor_id, smarthomehub_id and post frequency.

URI:

https://api-plantiq.azurewebsites.net/plant/all | GET

Example Request:

GET https://api-plantiq.azurewebsites.net/plant/all Send

Example Response (Success):

200 OK 6.83 s 138 B 2 Minutes Ago

Preview Headers 6 Cookies Timeline


```
{
  "data": [
    {
      "sensor_id": "1234567812345678",
      "smarthomehub_id": "123456789123",
      "name": "1234567812345678",
      "postFrequency": 3600
    }
  ],
  "outcome": true
}
```


Example Response (Failed):

This endpoint does not have a fail condition, in the event no data is present it will simply return data as an empty array with an outcome of true and a status of 200.

Getting A Plants Name

Plants or “Sensors” can have a name assigned to them to make it easier for users to interact with, to get the name of a specific plant simply use the /plant/{id}/name endpoint via the HTTP GET method.

URI:

https://api-plantiq.azurewebsites.net/plant/{id}/name GET

Example Request:

GET https://api-plantiq.azurewebsites.net/plant/123/name Send

Example Response (Success):

200 OK 1.42 s 105 B 2 Minutes Ago

Preview Headers 6 Cookies Timeline


```
{
  "name": "1234567812345678",
  "message": "No display name set, defaulting to sensor_id value",
  "outcome": true
}
```


Example Response (Failed):

404 Not Found 3.07 s 74 B Just Now

Preview Headers 6 Cookies Timeline


```
{
  "message": "No record found of plant for the provided id",
  "outcome": false
}
```

Setting A Plants Name

Plants or “Sensors” can have a name assigned to them, this name can be set by hitting the /plant/{id}/name endpoint via the HTTP PATCH method.

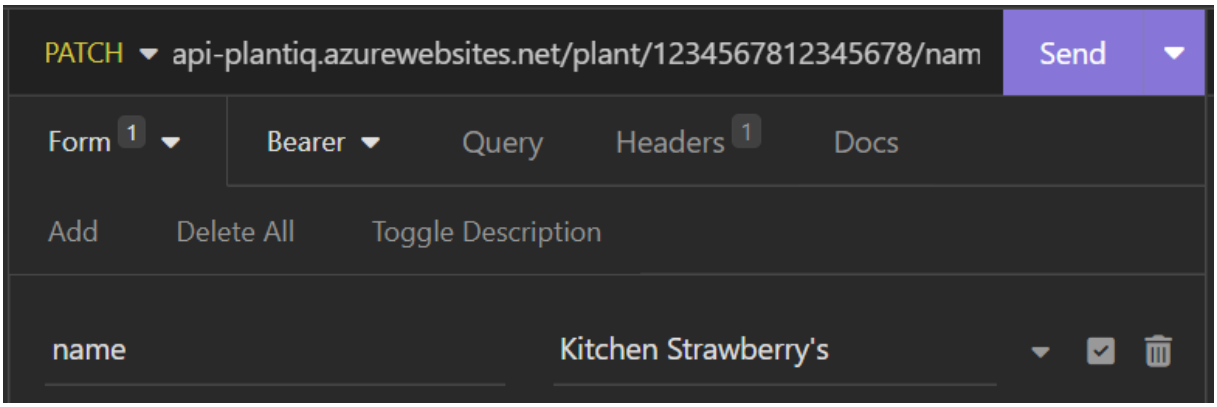
URI:

https://api-plantiq.azurewebsites.net/plant/{id}/name PATCH

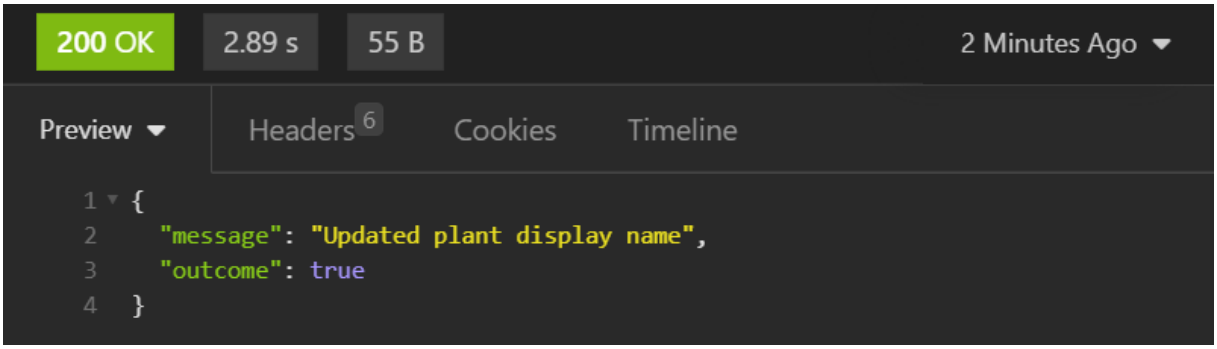
Parameters:

Key	Criteria
name	Required, Min 3 chars, Max 25 chars

Example Request:



Example Response (Success):



Example Response (Failed):

404 Not Found | 2.54 s | 74 B | Just Now

Preview | Headers 6 | Cookies | Timeline

```
1 {
2   "message": "No record found of plant for the provided id",
3   "outcome": false
4 }
```

Critical Ranges

Getting a smart home hubs critical range 40
Setting a smart home hubs critical range 41

Image Credt: https://www.pexels.com/photo/person-watering-the-hanging-plant-9414266/

Getting A Smart Hubs Critical Range

Smart Home Hubs have a range that will be used to validate incoming data, if the data is not within the range then an email and notification will be published to the user. To get the range for a specific smart home hub you can use the /range/{id} endpoint via HTTP GET.

Note: When a smart home hub does not have a user defined range it will simply return the applications default range in its place.

URI:

https://api-plantiq.azurewebsites.net/range/{id} GET

Example Request:

GET https://api-plantiq.azurewebsites.net/range/12345678asdf Send

Example Response (Success):

200 OK 1.44 s 99 B 1 Minute Ago

Preview Headers 6 Cookies Timeline


```
{
  "range": {
    "temperature": "10,90",
    "moisture": "5,25",
    "humidity": "5,10",
    "light": "1,16"
  },
  "outcome": true
}
```


Example Response (Failed):

401 Unauthorized 2.77 s 65 B Just Now

Preview Headers 6 Cookies Timeline


```
{
  "message": "Invalid smart home hub id provided!",
  "outcome": false
}
```

Setting Smart Hubs Critical Range

To set the critical ranges of a smart home hub you can simply post the data to the /range/{id} endpoint via the HTTP POST method.

Note: This endpoint is also used to update a smart hubs critical limits as it will clear any existing ranges and save the new ranges in there its place.

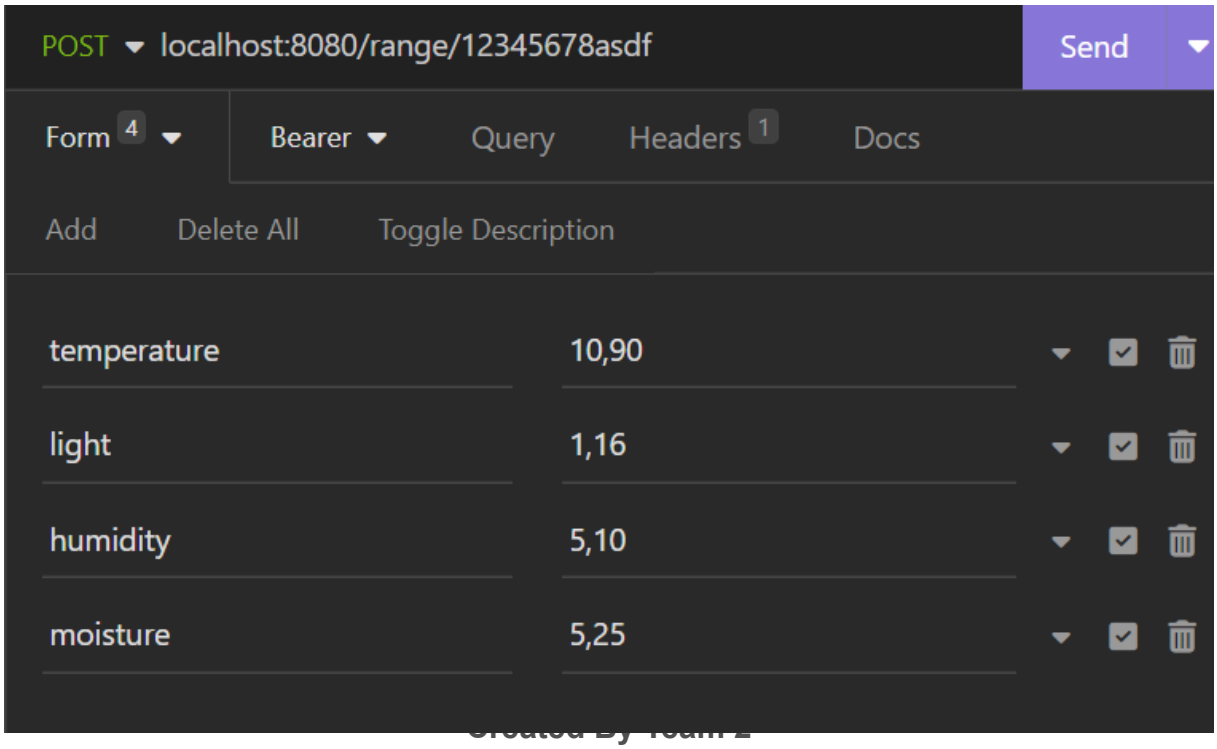
URI:

https://api-plantiq.azurewebsites.net/range/{id} POST

Parameters:

Key	Criteria
temperature	Required, Range, Float[1,100]
light	Required, Range, Float[1,24]
humidity	Required, Range, Float[1,100]
moisture	Required, Range, Float[1,100]

Example Request:



Example Response (Success):

200 OK | 10.5 s | 64 B | 4 Hours Ago

Preview | Headers 6 | Cookies | Timeline

```
1 {
2   "message": "Updated range values for smart hub!",
3   "outcome": true
4 }
```

Example Response (Failed):

401 Unauthorized | 2.77 s | 65 B | Just Now

Preview | Headers 6 | Cookies | Timeline

```
1 {
2   "message": "Invalid smart home hub id provided!",
3   "outcome": false
4 }
```

Notifications

Getting all notifications 44
Deleting all notifications 46
Deleting a specific notification 47

Image Credt: https://www.pexels.com/photo/pink-petaled-flowers-with-leaves-1494067/

Getting All A Users Notifications

User notifications can be created from a number of sources like plant data, account registration, password resetting and smart home hub registration. This endpoint allows for the requester to get a list of all the notifications of the current user.

URI:

https://api-plantiq.azurewebsites.net/notification/all | GET

Example Request:

GET https://api-plantiq.azurewebsites.net/notification/all Send

Form | Bearer | Query | Headers 1 | Docs

Add | Delete All | Toggle Description

Name | Value

Example Response (Success):

200 OK | 1.41 s | 789 B | Just Now

Preview | Headers 6 | Cookies | Timeline


```
1  {
2    "outcome": true,
3    "notifications": [
4      {
5        "message": "Kitchen Smart Plants reports one of its plants is
   out of range!, please review your recent data.",
6        "id": "12",
7        "timestamp": 1685366811,
8        "userID": "1"
9      },
10     {
11       "message": "New smart home hub registered to account!",
12       "id": "13",
```

Example Response (Failed):

200 OK | 1.1 s | 35 B | Just Now

Preview | Headers 6 | Cookies | Timeline

```
1 {
2   "outcome": true,
3   "notifications": []
4 }
```

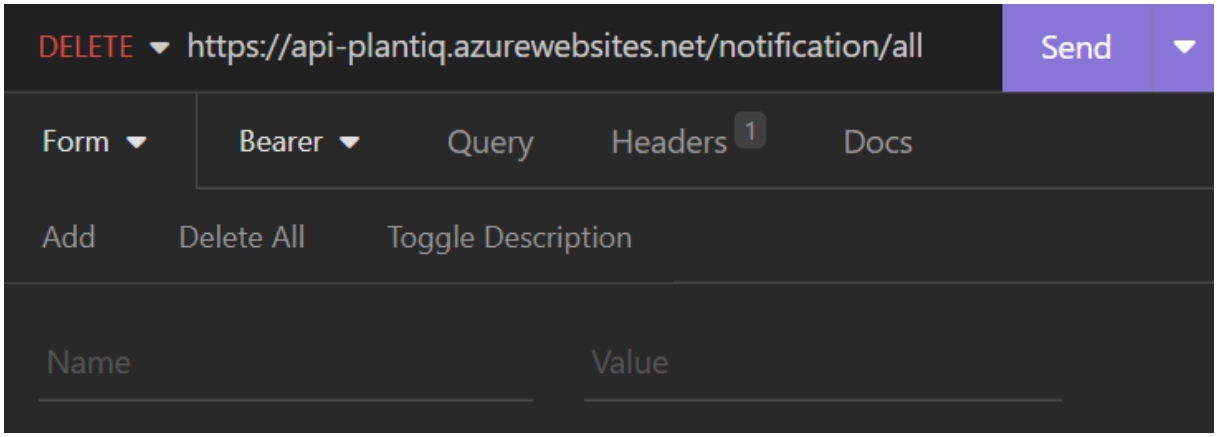
Deleting All A Users Notifications

The user may wish to delete or “Clear” all their current notifications, to do this they can simply hit the /notifications/all endpoint via the HTTP DELETE method, this will trigger all the notifications for that current user to be deleted from the application.

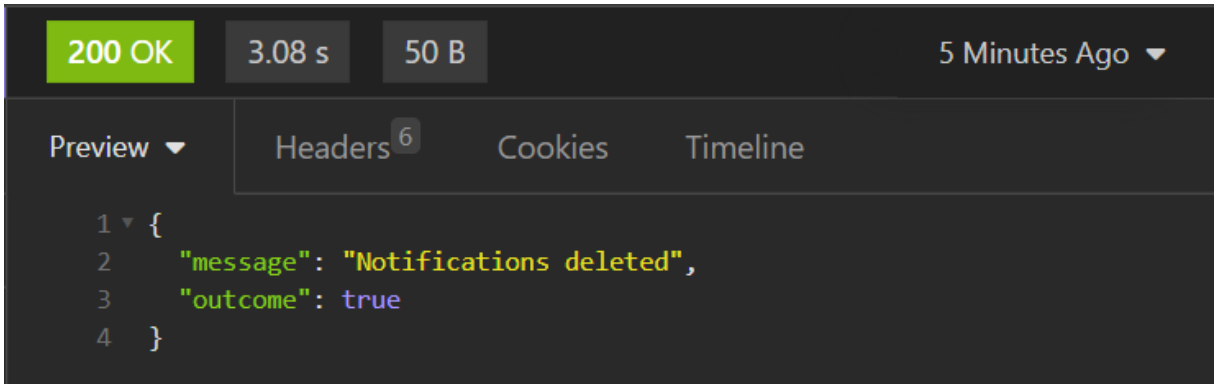
URI:

https://api-plantiq.azurewebsites.net/notifications/all DELETE

Example Request:



Example Response (Success):



Example Response (Failed):

No fail condition return for this endpoint, “Notifications deleted” with a outcome of true and a status of 200 should always be returned.

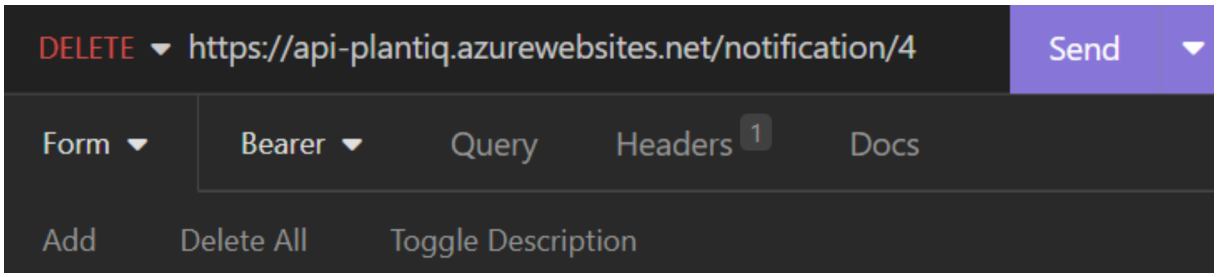
Deleting A Specific Notification

The user may sometimes wish to delete or “Clear” a specific notification instead of all of them, this can be done via hitting the /notification/{id} endpoint with the HTTP DELETE method, ensuring that a valid notification id is provided in the URI.

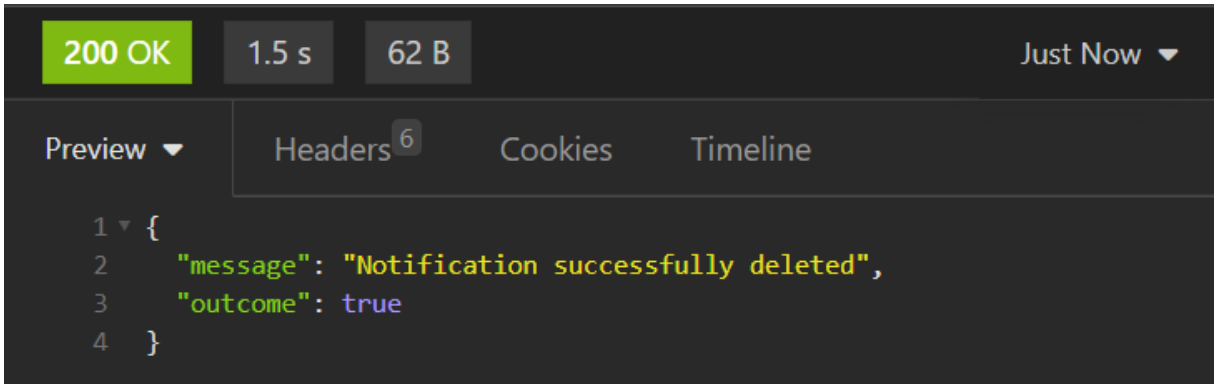
URI:

https://api-plantiq.azurewebsites.net/notifications/{id} DELETE

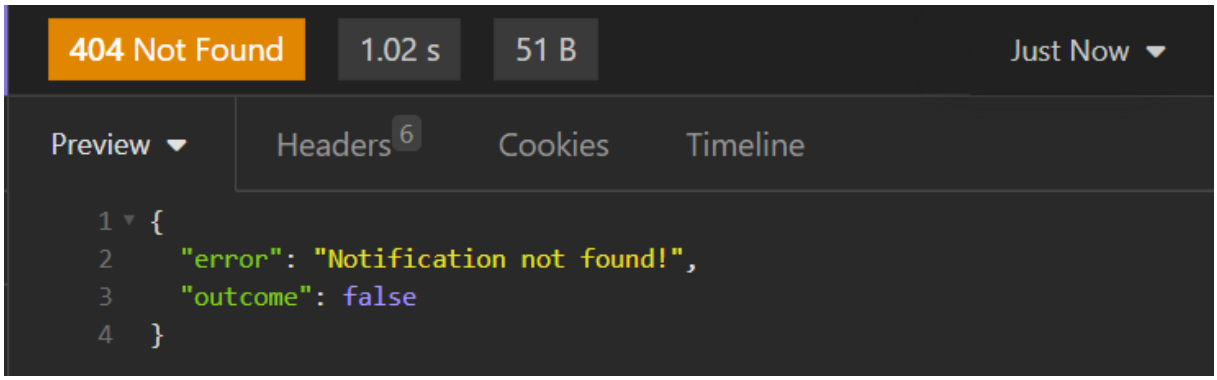
Example Request:



Example Response (Success):



Example Response (Failed):



Admin Functions

Activating a users account 49
Listing all users 50
Deleting a user 52
Listing all a users smart home hubs 53
Deleting a users smart home hub including all data 55
Deleting a users smart home hub data 56
Promoting a user to admin 57
Demoting a admin to user 58

Image Credt: https://www.pexels.com/photo/green-plant-on-white-ceramic-pot-4503744/

Activating A Users Account

Administrators of the PlantIQ website can activate user accounts for users in the event they run into email difficulties or are unable to recieve the activation email from the PlantIQ email.

URI:

https://api-plantiq.azurewebsites.net/admin/user/activate/{id}	POST

Example Request:

POST localhost:8080/admin/user/activate/e63b51fa67e32be5f6466b5f1c3ac3c3156b0e95 Send

Example Response (Success):

200 OK 2.69 s 64 B Just Now

Preview Headers 6 Cookies Timeline

```
1 {
2   "message": "Successfully activated user account",
3   "outcome": true
4 }
```

Example Response (Failed):

404 Not Found 1.47 s 52 B Just Now

Preview Headers 6 Cookies Timeline

```
1 {
2   "message": "User account not found",
3   "outcome": false
4 }
```

200 OK 5.86 s 62 B 12 Days Ago

Preview Headers 6 Cookies Timeline

```
1 {
2   "message": "User account is already activated",
3   "outcome": true
4 }
```

Listing All Users

The admin can visualise all the users registered with PlantIQ and modify the list by limiting, offsetting and ordering by an attribute in an ascending or descending manner.

To extrapolate the list the admin must hit the endpoint /user/all via get method.

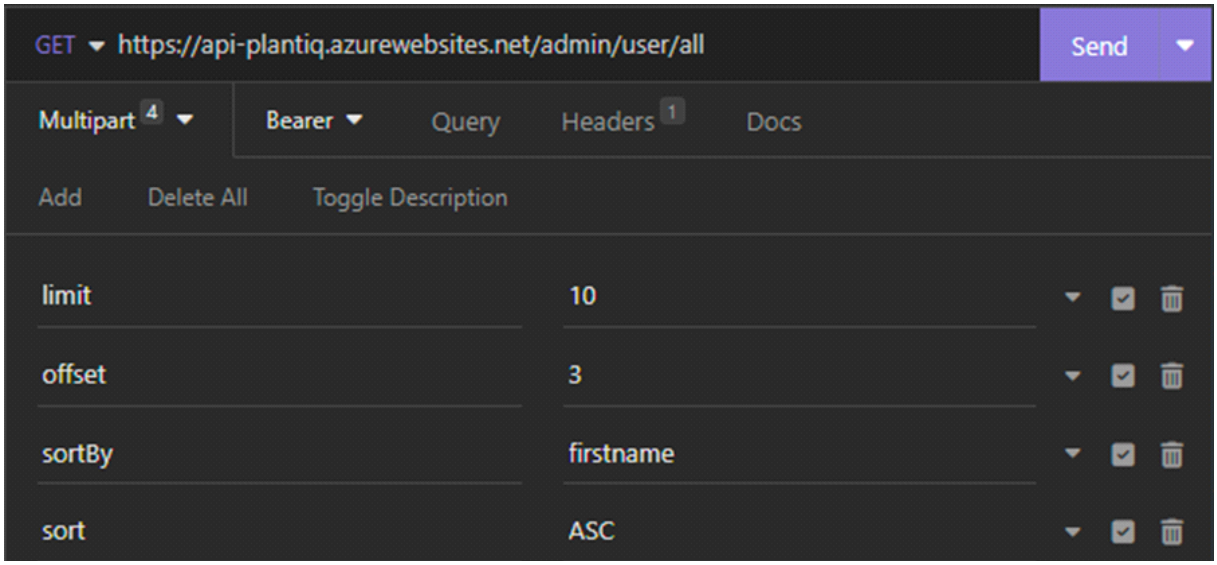
URI:

https://api-plantiq.azurewebsites.net/admin/user/all	GET

Parameters:

Key	Criteria
limit	Required, Integer, Min 10, Max 100
offset	Required, Integer, Min 0, Max 100
sortBy	Required, Min 3 chars , Max 25 chars
sort	Required, Sort (must be value of enum either ASC or Desc)

Example Request:



Example Response (Success):

200 OK | 628 ms | 1628 B | Just Now

Preview | Headers 6 | Cookies | Timeline


```
1  {
2    "list": [
3      {
4        "id": "68684cc142d1aeca5a8a1ab192520a6bbce9a166",
5        "firstname": "Bill",
6        "surname": "Gates",
7        "activated": true,
8        "administrator": false,
9        "email": "profess78@hotmail.com"
10     },
11     {
12       "id": "b5f191f9f314629df65c39f1cad7707efc4f663a",
13       "firstname": "Blake",
14       "surname": "Bortles",
15       "activated": true,
16       "administrator": false,
17       "email": "blake@gmail.com"
18     },
```


Example Response (Failed):

400 Bad Request | 1.65 s | 302 B | Just Now

Preview | Headers 6 | Cookies | Timeline


```
1  {
2    "outcome": false,
3    "errors": {
4      "offset": {
5        "integer": "Invalid type provided, input must be a valid integer"
6      },
7      "limit": {
8        "min": "Provided value should be at least 10"
9      },
10     "sortBy": {
11       "min": "Provided value should be at least 3 characters long"
12     },
13     "sort": {
14       "enum": "Invalid sort type provided, sort must match ASC, DESC"
15     }
16   }
17 }
```

Deleting A User

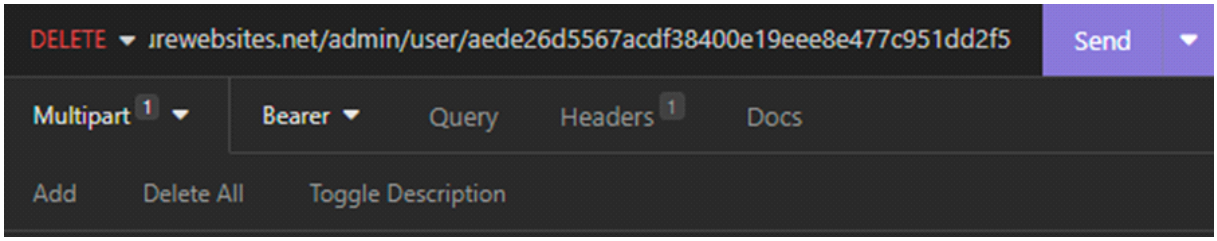
The admin will have the permission to delete the account of a user. To accomplish the task, the admin must hit the endpoint /user/{id} via delete method, where the id is the id of the account that is going to be deleted.

Note: Deleting an account will also delete SmartHomeHubs and Plants relating the deleted user.

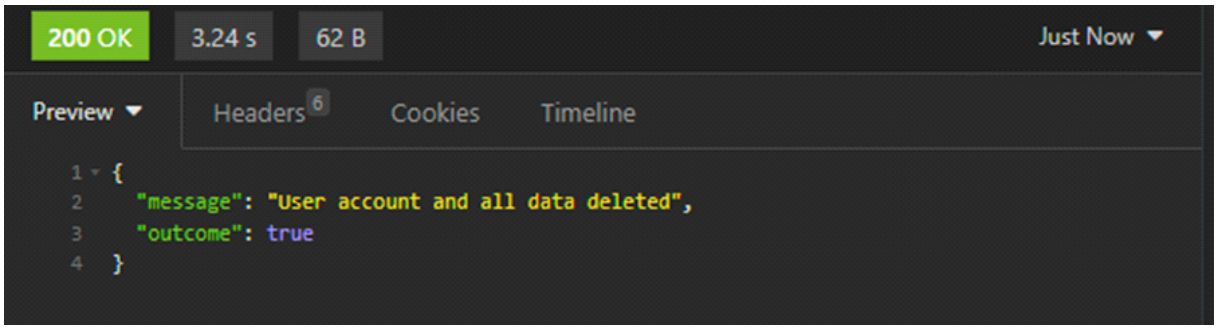
URI:

https://api-plantiq.azurewebsites.net/admin/user/{id} DELETE

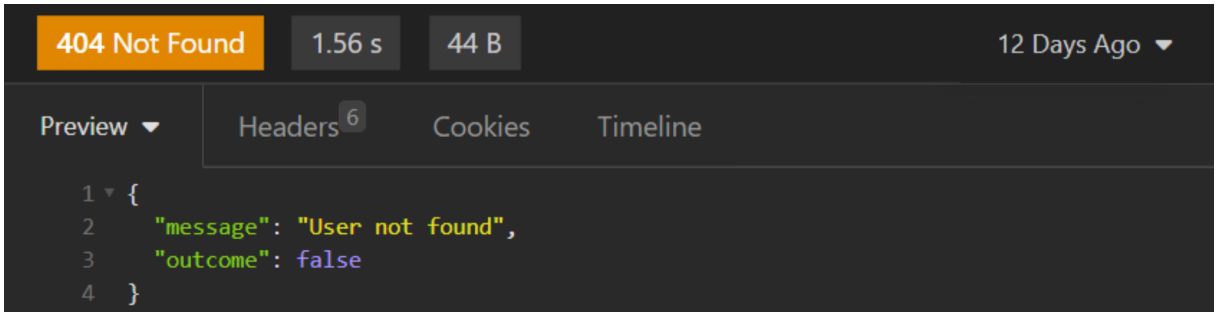
Example Request:



Example Response (Success):



Example Response (Failed):



Listing A User Smart Home Hubs

Admins will be able to visualise all the SmartHomeHubs relating to an account. To retrieve the list of hubs, the admin will need to hit the endpoint /user/{id}/smarthomehub/all via get method, where id is the user's id.

URI:

https://api-plantiq.azurewebsites.net/admin/user/{id}/smarthomehub/all	GET

Example Request:

GET /admin/user/a19bf9ac2f08c748b030dcaeabb2817d1cf255c0/smarthomehub/all Send

Body Bearer Query Headers Docs

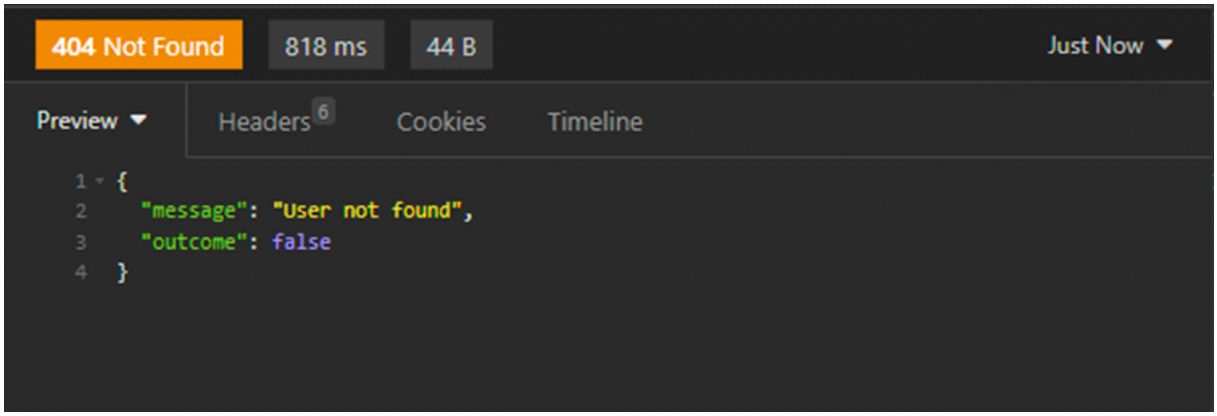
Example Response (Success):

200 OK 2.05 s 341 B Just Now

Preview Headers 6 Cookies Timeline


```
{
  "list": [
    {
      "name": "DiningRoomHub",
      "id": "123321123322",
      "postFrequency": 3600,
      "lastPosted": 1684923072,
      "virtual": false
    },
    {
      "name": "123456789126",
      "id": "123456789126",
      "postFrequency": 3600,
      "lastPosted": 1683938619,
      "virtual": false
    },
    {
      "name": "123456789128",
      "id": "123456789128",
      "postFrequency": 3600,
      "lastPosted": 1683941012,
      "virtual": false
    }
  ],
  "outcome": true
}
```

Example Response (Failed):



Deleting A Users Smart Home Hub Including All Data

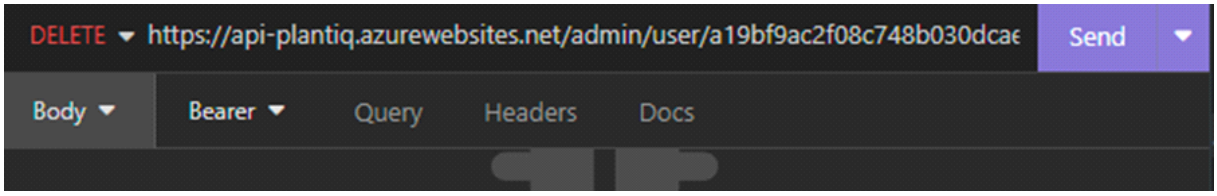
Admins have the permission to delete a SmartHomeHub of other users. This action can be accomplished by hitting the endpoint /user/{id}/smarthomehub/{home_id} via delete method, where id is the user's id while home_id is the SmartHomeHub's id.

Note: Deleting a SmartHomeHub will delete all the plants data relating that hub.

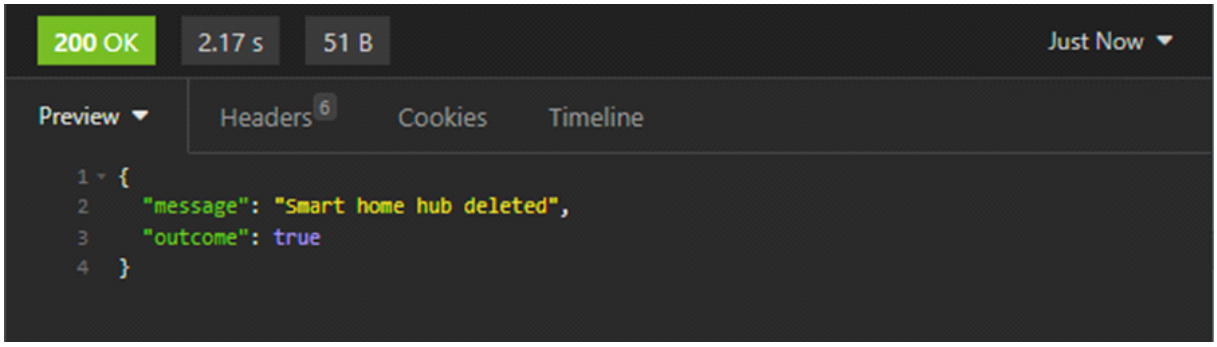
URI:

https://api-plantiq.azurewebsites.net/admin/user/{user_id}/smarthomehub/{hub_id}	DELETE

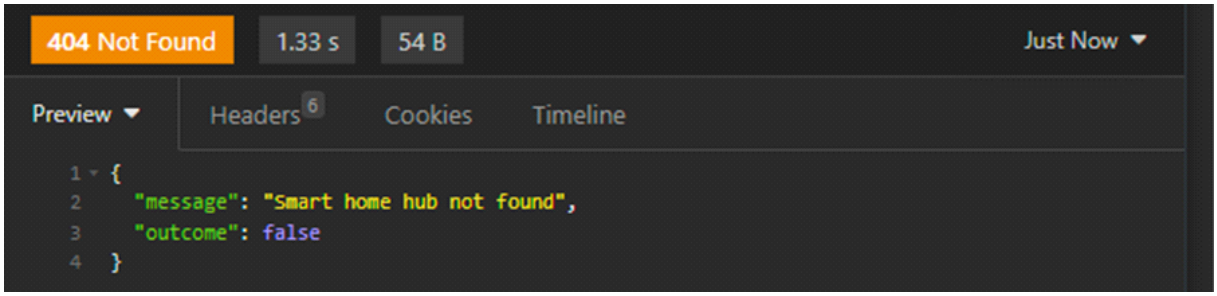
Example Request:



Example Response (Success):



Example Response (Failed):



Deleting A Users Smart Home Hubs Data Only

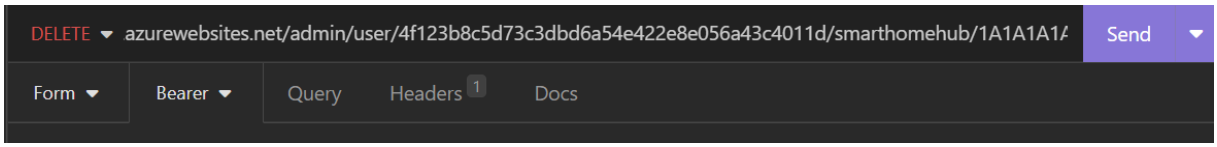
Admins can delete all plant data belonging to other user's SmartHomeHubs. This action can be accomplished by hitting /user/{id}/smarthomehub/{hub_id}/data via delete method, where id is the user's id and the hub_id is the SmartHomeHub's id.

This will only delete or "Clear" the data relating to that hub and not remove the hub itself.

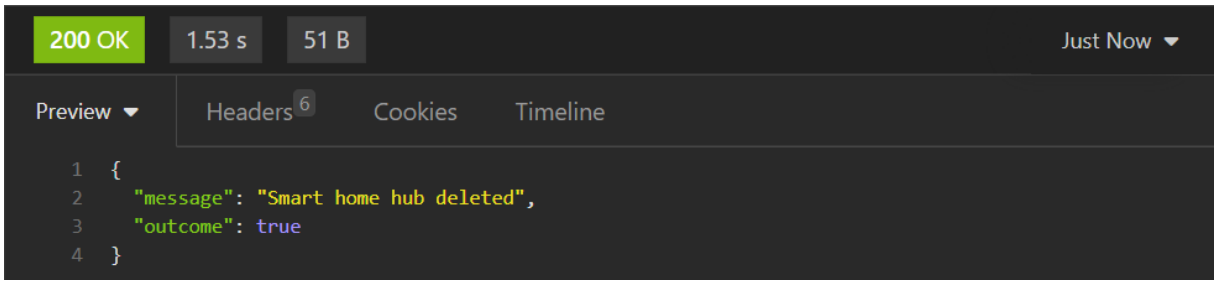
URI:

https://api-plantiq.azurewebsites.net/admin/user/{user_id}/smarthomehub/{hub_id}/data	DELETE

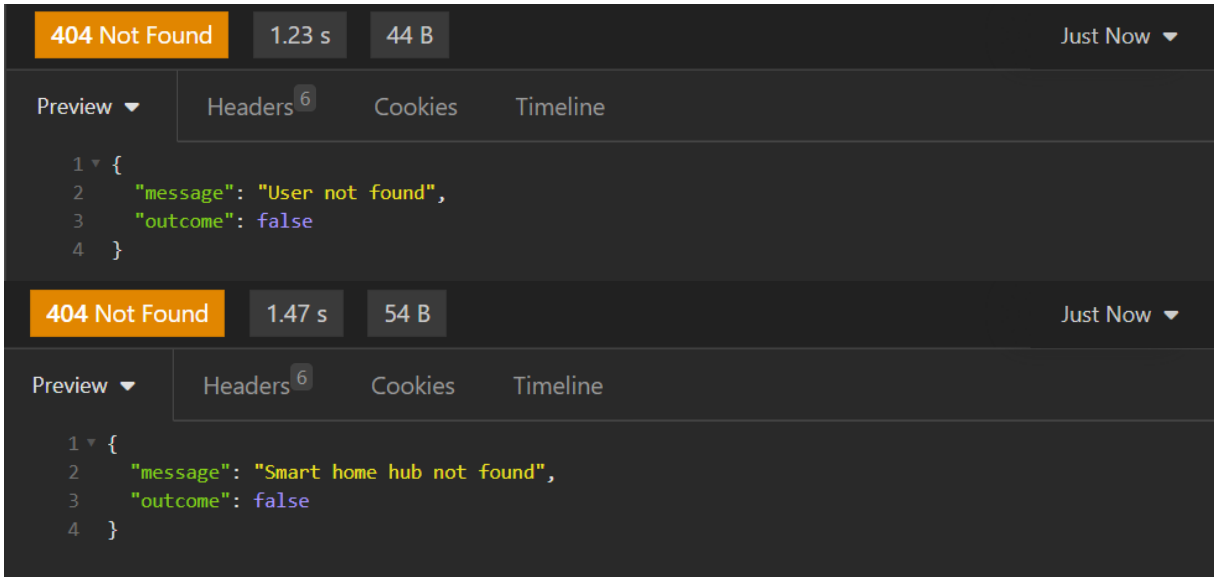
Example Request:



Example Response (Success):



Example Response (Failed):



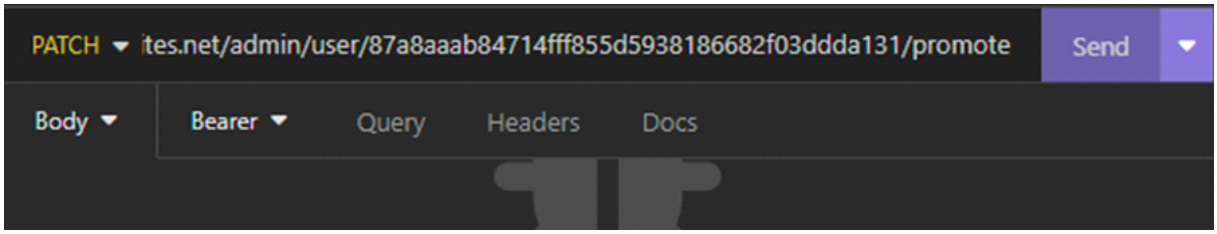
Promoting A User Account To Admin

Admins have the permission to promote other users to the role of admin. To accomplish the task, an admin must hit the endpoint /user/{id}/promote via patch method, where id is the id of the user getting promoted.

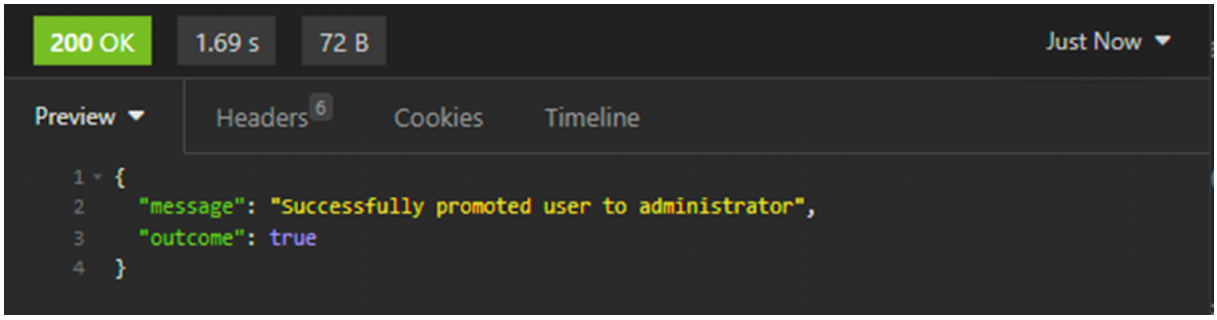
URI:

https://api-plantiq.azurewebsites.net/admin/user/{user_id}/promote	PATCH

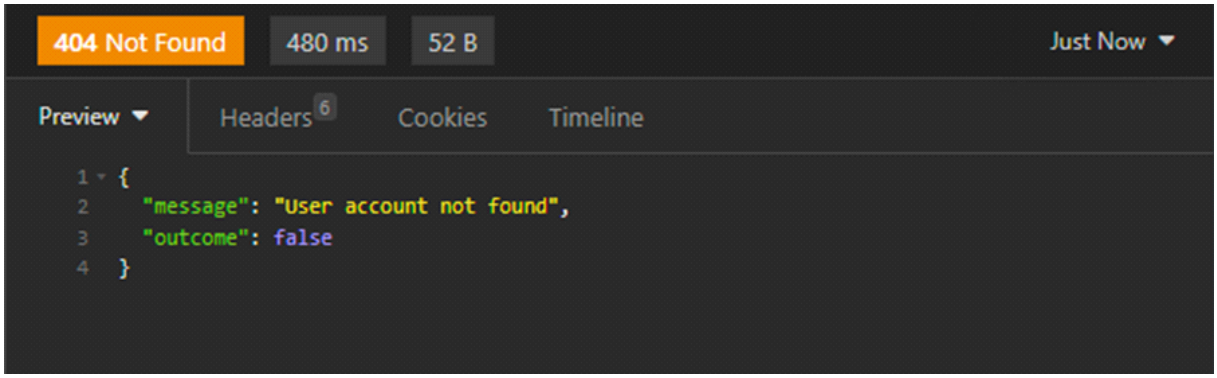
Example Request:



Example Response (Success):



Example Response (Failed):



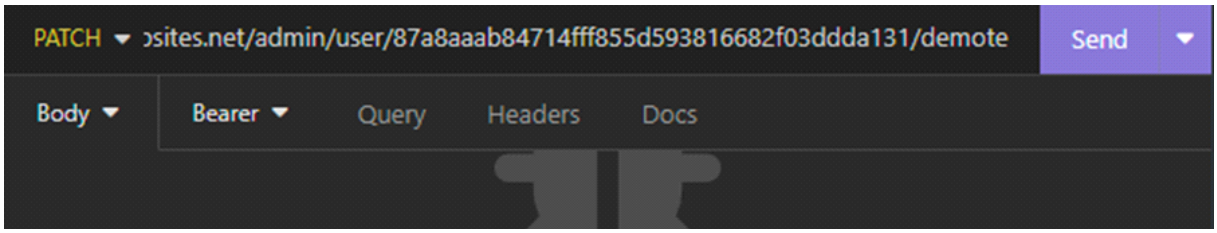
Demoting A Admin Account To User

Admins don't only have the permission to promote a user, but they have also the ability to demote them. To make an admin a normal user the endpoint /user/{id}/demote must be hit via patch method, where id is the id of the user getting demoted.

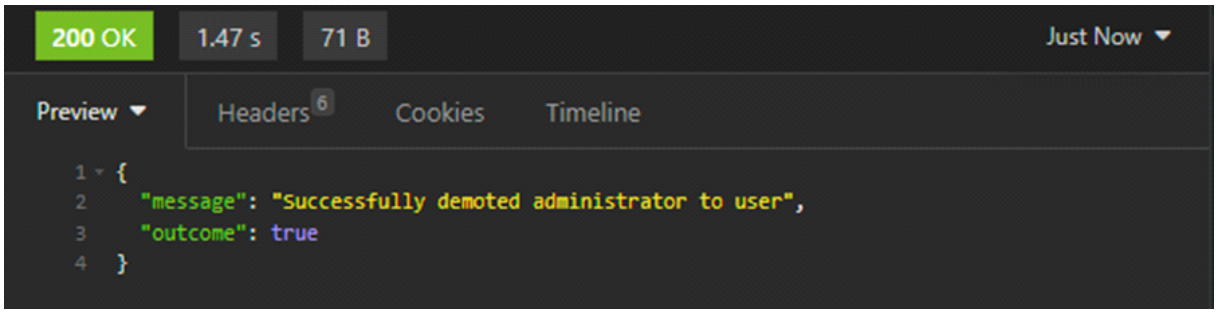
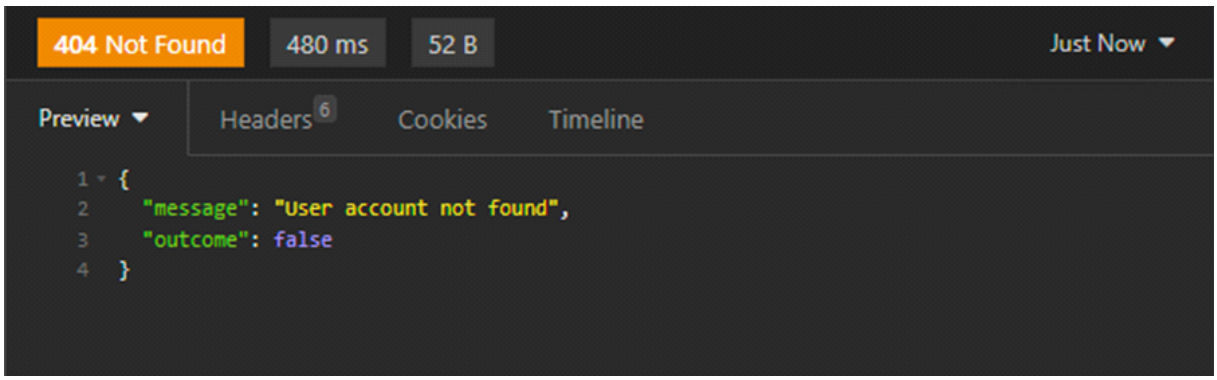
URI:

https://api-plantiq.azurewebsites.net/admin/user/{user_id}/demote	PATCH

Example Request:



Example Response (Success):



Example Response (Failed):