Resensys REST API v2 Documentation:

The Resensys API v1 is designed to provide a secure and RESTful approach to integrating SenSpot data into your own existing applications. To this end, the following describes how to use version one including authenticating a user, pulling data for a single or collection of devices, and getting photos from a given camera site ID to get you started with ETL processes and automations.

Please note that the following demonstrations utilize Python 3.x for its 'requests' HTTP library as it has the ability to create persistent TCP connections during a given session, which can result in increased performance. Feel free to reach out if you have any questions about how to implement for your unique use case.

User Authentication:

Connecting to Resensys servers via the API is similar to logging into SenScope and the WebPortal: you will need a username and a password.

Start by importing the relevant modules needed for the API calls:

```
#!/usr/bin/env python3.8
import requests
import sys
if ".." not in sys.path:
    sys.path.append("..")
```

Next, define both the login endpoint as `login_url = "https://resensys.net:8443/login"`. You can then create a TPC session and request the corresponding authentication token to persist the connection across consecutive calls

```
login_url = "https://resensys.net:8443/login"

csrf_client = requests.session()
csrf_client.get(url=login_url)
csrftoken = csrf_client.cookies['csrftoken']
```

You can now use this HTTP client to perform POST requests to the Resensys database.

POST Data for a Single Device:

Using the `"https://resensys.net:8443/api/v1/data_acquire"` endpoint, you can retrieve data during any time interval for any quantity for any device in any authorized site. To do this, you should specify the following parameters before sending a POST request:

a. T_start: UTC start time from which to pull data (format: **YYYY-mm-dd HH:MM:SS**)
b. T_end: UTC end time from which to pull data (format: **YYYY-mm-dd HH:MM:SS**)
c. username: username used to login to SenScope and the Resensys Web Portal
d. password: password to login
e. SID: site ID used to identify devices for a given project
f. DID: device ID used to identify as specific device on a given site
g. DF: data format prefix given to every available quantity for Resensys devices
    a. A complete list of quantity data formats can be retrieved using the following link: https://resensys.com/Software/SenScope/QuantityBank.xml
h. Unit: string-formatted unit for a given data format (e.g. Volt has unit "v" for volts)
i. csrfmiddlewaretoken: key used to authenticate user

This can be implemented in the following way:

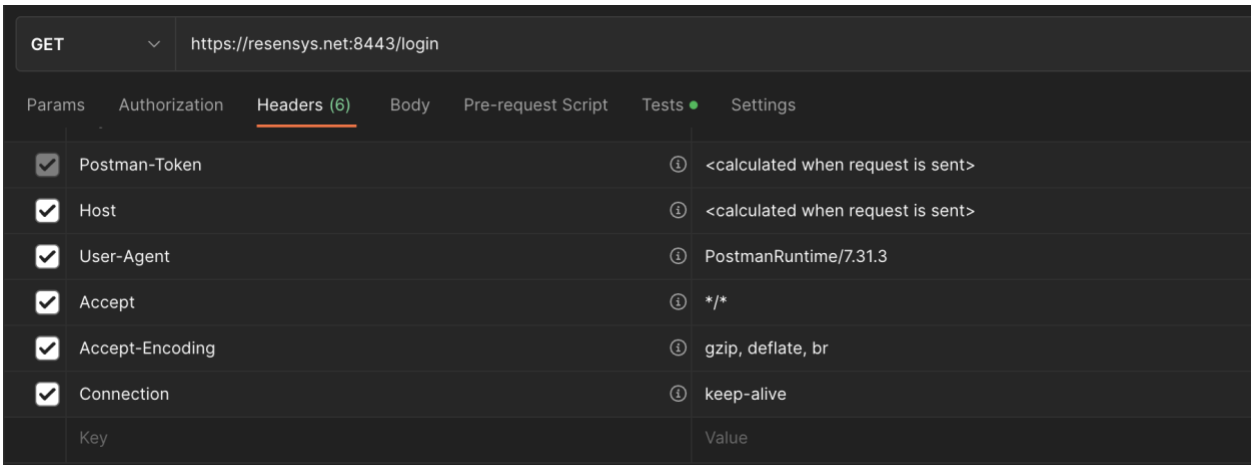
```
# define starting and ending times
t_start,t_end = '2022-04-20 00:00:00','2022-04-20 12:00:00'


# construct dictionary for POST request
sending = dict(username="username", password="password",
               SID="1F-00", DID="15-03-42-23", DF="22002", Unit="v",
               T_start=t_start, T_end=t_end,
          csrfmiddlewaretoken=csrftoken)


# retrieve server response as HTTP code
r = csrf_client.post(data_server_url, data=sending,
                     headers=dict(Referer=data_server_url))
```

Now you can retrieve JSON-formatted data using

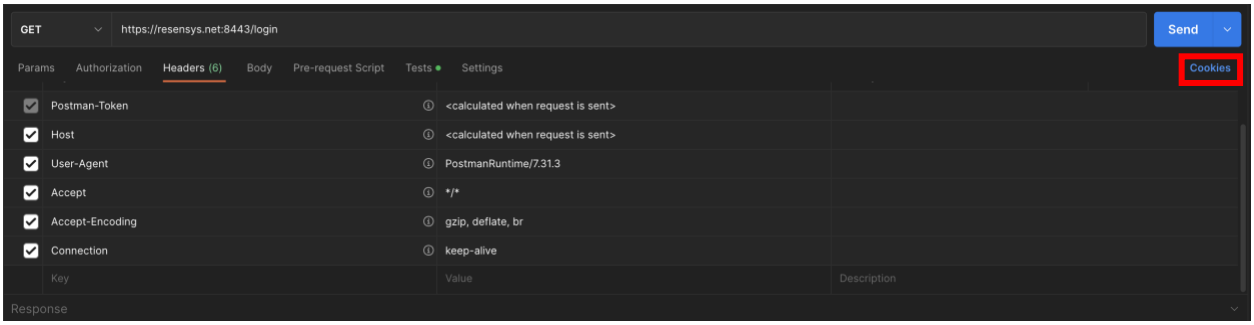
```
data_json = r.json() # returns JSON data of device for given quantity
```

Data Acquire via Postman:

To start, we need to make a GET request to retrieve cookie needed for handling the session transactions for user. Open a new tab and enter the following URL with method GET selected (see below):
https://resensys.net:8443/login

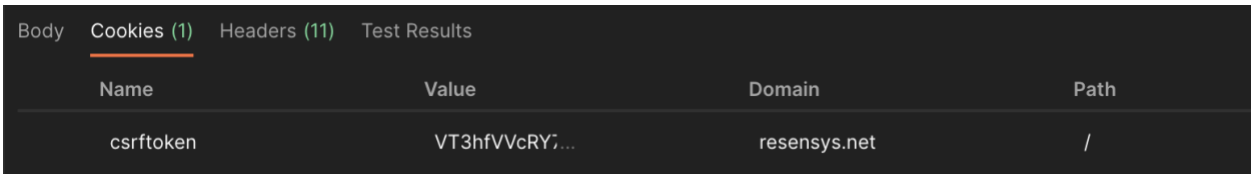


Before sending this request, be sure to clear any cookies that may exist for host resensys.net by selecting “Cookies” in the top right of the screen.

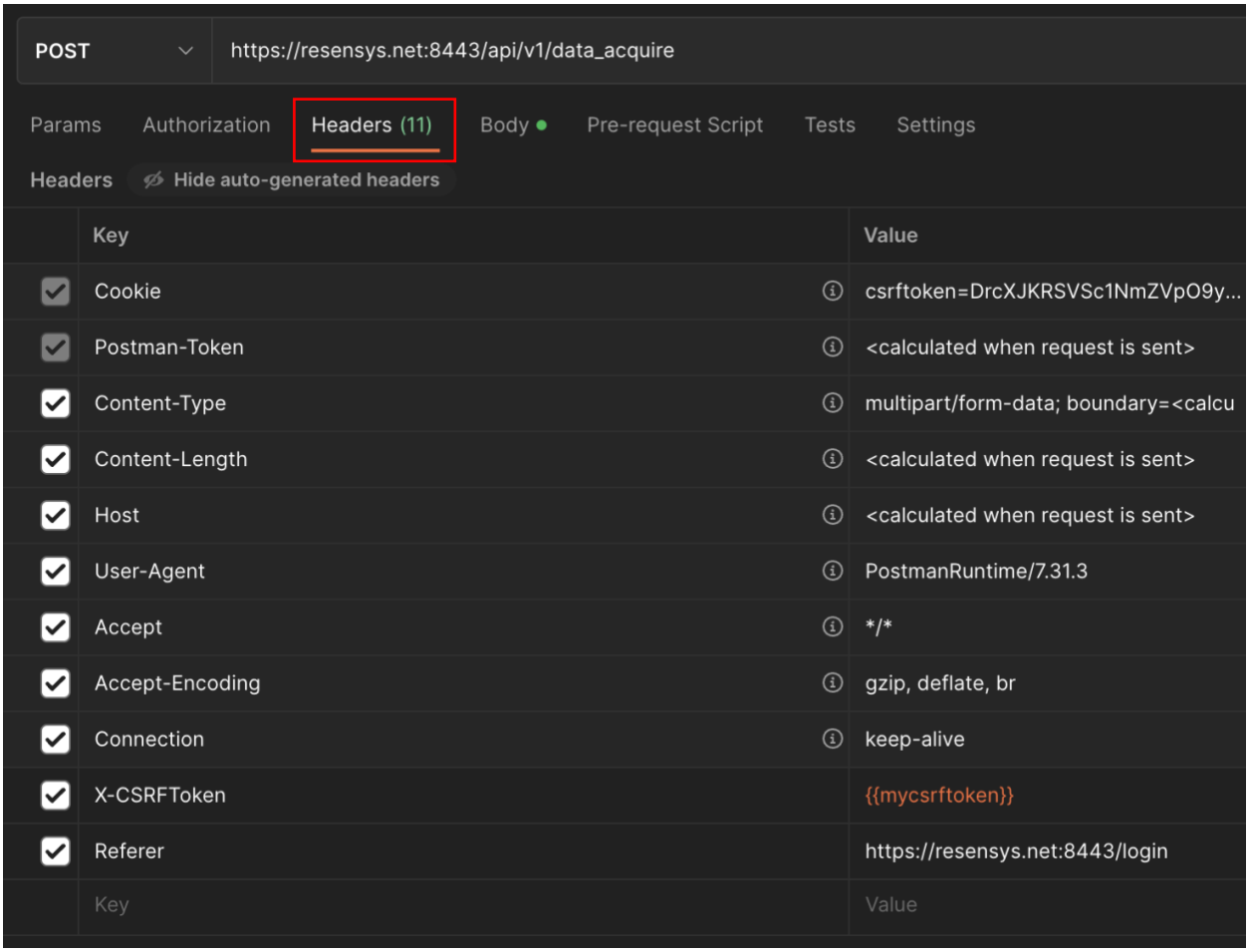


Additionally, you can store the value of the resulting cookie in a Postman global variable to be reused later for data retrieval by toggling the “Tests” tab below the URL and entering the following javascript:

Send the request and toggle the “Cookies” tab in the response, you should see one cookie with name **csrftoken** with its value attached to domain **resensys.net**. The value of this cookie should also be stored in the global variable “mycsrftoken.”

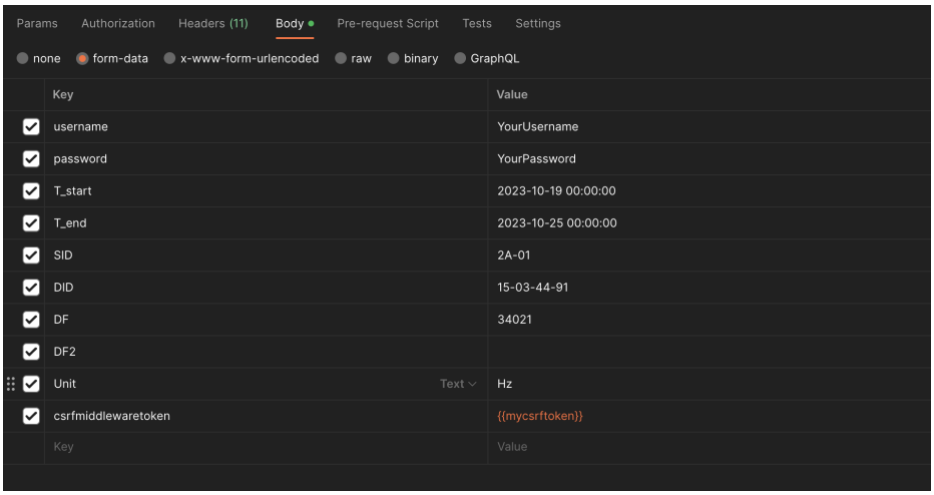


With the session token stored, open a new tab to send the POST request for data and add the following URL with method changed to POST: https://resensys.net:8443/api/v1/data_acquire



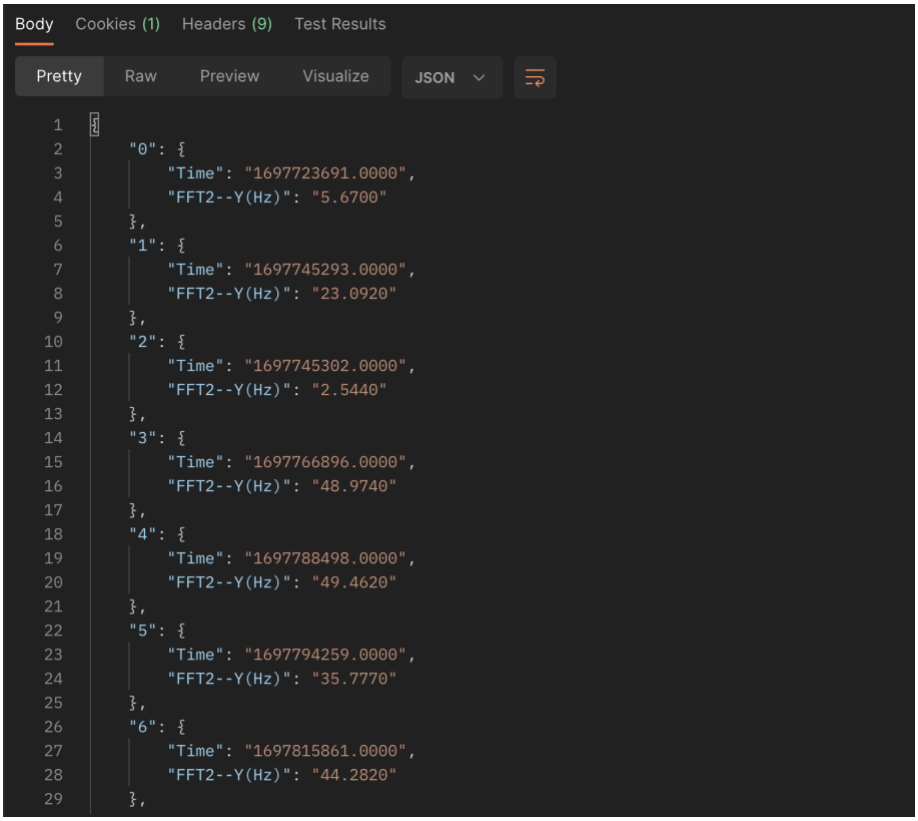
Note: you should add two items to the header information before sending: “X-CSRFToken” and “Referer” (see above).

Each call to the data_acquire endpoint requires 10 parameters be sent in its body: username, password, T_start (start time UTC), T_end (end time UTC), SID (site ID), DID (device ID), DF (dataformat integer), DF2, Unit, and csrfmiddlewaretoken



Note: You can set the csrfmiddleware token automatically with the stored value of “mycsrftoken” (see above).

Send the POST request to the endpoint, and you should receive a JSON response (see below for sample response).




```
{
    "0": {
        "Time": "1697723691.0000",
        "FFT2--Y(Hz)": "5.6700"
    },
    "1": {
        "Time": "1697745293.0000",
        "FFT2--Y(Hz)": "23.0920"
    },
    "2": {
        "Time": "1697745302.0000",
        "FFT2--Y(Hz)": "2.5440"
    },
    "3": {
        "Time": "1697766896.0000",
        "FFT2--Y(Hz)": "48.9740"
    },
    "4": {
        "Time": "1697788498.0000",
        "FFT2--Y(Hz)": "49.4620"
    },
    "5": {
        "Time": "1697794259.0000",
        "FFT2--Y(Hz)": "35.7770"
    },
    "6": {
        "Time": "1697815861.0000",
        "FFT2--Y(Hz)": "44.2820"
    },
```

POST to Retrieve Device Registrations:

Resensys SenSpot devices are organized into sites with a unique site ID. Each device also has a list of associated quantities/datastreams that can be extracted. For each datastream, there is a unique integer value, called a dataformat, which is needed in the payload for POST requests made to retrieve raw data. Use the following endpoint in a POST request to list all the available quantities, dataformats, device IDs, device types, and site ID assignments for each device associated to your account: https://resensys.net:8443/api/v2/registration

payload = { 'username' : <string:username> , 'password' : <string:password> }

The response returns JSON with records for each unique device-quantity pair (example shown below):


```
    "16": {
        "Name": "Device 25-14",
        "DID": "15-03-25-14",
        "SID": "18-64",
        "QuantityName": "Volt",
        "DataFormat": 3003,
        "Type": "2DHRT"
    },
    "17": {
        "Name": "Device 25-14",
        "DID": "15-03-25-14",
        "SID": "18-64",
        "QuantityName": "Internal Temperature",
        "DataFormat": 3002,
        "Type": "2DHRT"
    },
    "18": {
        "Name": "Device 25-14",
        "DID": "15-03-25-14",
        "SID": "18-64",
        "QuantityName": "RSSI",
        "DataFormat": 3001,
        "Type": "2DHRT"
    },
```


GET Photos from Camera Site:

Photos stored as .jpg files on Resensys servers can be retrieved by encoding the image in base64 at the server side. These encoded photos are sent in JSON formatting to the client using a POST request and can be decoded and stored locally on the client side using the following implementation:

```
# import necessary libraries
import base64
import requests
import sys
import os
if ".." not in sys.path:
    sys.path.append("..")



# define the HTTP endpoints
photo_server_url = "https://resensys.net:8443/api/v1/photos"
login_url = "https://resensys.net:8443/login"


# create persistent HTTP client and obtain token
csrf_client = requests.session()
csrf_client.get(url=login_url)
csrftoken = csrf_client.cookies['csrftoken']
```

You can then obtain pictures from any time period stored on the Resensys cloud by first specifying the start and end time in UTC and constructing the dictionary for the POST request:

```
# define starting and ending times
t_start,t_end = '2022-04-15 07:53:06','2022-04-15 14:29:12'  # note format

# construct dictionary for photo POST request
sending = dict(username="username",password="password",SID="1F-00",
T_start=t_start,T_end=t_end,csrfmiddlewaretoken=csrftoken)

#retrieve the server response at HTTP code
r = csrf_client.post(photo_server_url, data=sending,
                     headers=dict(Referer=photo_server_url))
```

By default, photos will be saved to the current working directory unless otherwise specified. You can specify the path and then decode each record in the encoded photo JSON object into that path with a few lines of code:

```
path = "./SenSpot_Photos/"  # Specify the name of path you want to store photos
if not os.path.isdir(path):
    os.mkdir(path)

photo_json = r.json() # convert the HTTP response to JSON for photos
# decode every photo in from base64 and push to directory set in 'path' variable
for key, value in photo_json.items():
```

```
encoded_string = str.encode(value["photoString"])
decodeit = open(path+value["title"]+'.png', 'wb')
decodeit.write(base64.b64decode((encoded_string)))
decodeit.close()
```