This script is an **MQTT data logger** that subscribes to multiple topics, logs received messages to a CSV file, and can be customized for database logging if needed. Let's break it down step by step:

### ****1. Importing Required Modules****

python

CopyEdit

import os

import sys

import time

import argparse

import logging

import csv

import paho.mqtt.client as mqtt

* os & sys: Used for file path handling and script termination.
* time: Used for timestamping received messages.
* argparse: Enables command-line argument parsing.
* logging: Manages logging of messages to a CSV file.
* csv: Handles writing data to CSV.
* paho.mqtt.client: The MQTT client library for subscribing to topics and receiving messages.

### ****2. Handling MQTT Connection****

python

CopyEdit

def on\_connect(client, userdata, flags, rc):

if rc == 0:

print("Connected to MQTT broker")

for topic in args.topic:

client.subscribe(topic, qos=0) # Set QoS as needed (0, 1, or 2)

else:

print("Failed to connect to MQTT broker")

* When the MQTT client successfully connects, this function subscribes to multiple topics.
* rc == 0 means the connection was successful.
* The script subscribes to each topic in the list with a Quality of Service (QoS) level of 0 (no acknowledgment from the broker).

### ****3. Handling Incoming MQTT Messages****

python

CopyEdit

def on\_message(client, userdata, msg):

timestamp = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime())

message = msg.payload.decode()

log\_data(timestamp, message)

print("ready")

* When an MQTT message is received, this function:
  + Decodes the message from bytes to a string.
  + Gets the current timestamp.
  + Calls log\_data() to store the message.

### ****4. Logging Data to CSV****

python

CopyEdit

def log\_data(timestamp, message):

log\_file = os.path.expanduser("path address/data\_log.csv")

with open(log\_file, mode='a') as file:

writer = csv.writer(file, delimiter=',', quotechar='"', quoting=csv.QUOTE\_MINIMAL)

writer.writerow([timestamp, message])

print("write")

* **Stores received messages in a CSV file** (data\_log.csv).
* Uses csv.writer() to write data in a comma-separated format.
* os.path.expanduser() ensures that the file path is expanded correctly.

### ****5. Parsing Command-Line Arguments****

python

CopyEdit

if \_\_name\_\_ == "\_\_main\_\_":

parser = argparse.ArgumentParser(description="MQTT data logger")

parser.add\_argument("-H", "--host", default="ip address", help="MQTT broker host address")

parser.add\_argument("-p", "--port", default=1883, type=int, help="MQTT broker port number")

parser.add\_argument("-t", "--topic", nargs='+', default=[...], help="List of MQTT topics to subscribe to")

parser.add\_argument("-u", "--username", default="", help="MQTT username")

parser.add\_argument("-P", "--password", default="", help="MQTT password")

args = parser.parse\_args()

print("Process")

* Allows the script to **accept user inputs** for:
  + MQTT broker IP address
  + Port number (default 1883)
  + List of topics (default includes 23 topics)
  + Username & Password for authentication

### ****6. Setting Up Logging****

python

CopyEdit

log\_file = os.path.expanduser("path address /data\_log.csv")

logging.basicConfig(filename=log\_file, level=logging.INFO, format="%(message)s")

print("set")

* Configures **logging to the CSV file**.
* Uses logging.INFO to log messages in a simple format.

### ****7. Initializing MQTT Client****

python

CopyEdit

client = mqtt.Client()

client.on\_connect = on\_connect

client.on\_message = on\_message

if args.username:

client.username\_pw\_set(args.username, args.password)

client.connect(args.host, args.port, 60)

* Creates an **MQTT client**.
* Assigns on\_connect and on\_message functions.
* If authentication credentials are provided, they are set using client.username\_pw\_set().

### ****8. Running the MQTT Loop****

python

CopyEdit

try:

client.loop\_forever()

except KeyboardInterrupt:

client.disconnect()

sys.exit(0)

* The script continuously listens for MQTT messages.
* If interrupted (Ctrl+C), it **disconnects gracefully**.

## ****How the Script Works****

1. **Start the script with command-line arguments**:

bash

CopyEdit

python script.py -u username -P password

1. It **connects to the MQTT broker** and subscribes to topics.
2. When messages are received, they are:
   * Printed to the console.
   * Logged into data\_log.csv.
3. The script **runs indefinitely** until manually stopped.

## ****Customizations You Can Make****

* **Change MQTT QoS level** in client.subscribe(topic, qos=0).
* **Modify CSV storage path** in log\_data().
* **Store messages in a database** instead of a CSV file.