NMS 2.0 Class Cheat Sheet

Contents

[Class: IoT Box 2](#_Toc203981283)

[Class: MQTT Message 4](#_Toc203981284)

[Class: EMQX MQTT Broker 6](#_Toc203981285)

[Class: Grafana Dashboard 7](#_Toc203981286)

[Class: InfluxDB database 8](#_Toc203981287)

[Class: InfluxDB Client 9](#_Toc203981288)

[Class: Python app.py 10](#_Toc203981289)

[Class: MQTT Client 11](#_Toc203981290)

# Class: IoT Box

* MQTT Protocol: MQTT v3.1.1
* version
* session (string)
* deviceId (string)
* slot7\_state
  + spdAlert (Surge Protective Device, bool)
  + spdSurge (Number of Surges, int)
  + ardState (Automatic Reclosing Protector, bool)
  + ardAlert (bool)
  + ardVoltage (V) (int)
  + ardCurrent (mA) (int)
  + autoLight (bool)
  + doorState (bool)
  + lightState (Door Light State, bool)
  + heaterState (bool)
  + waterAlert (Submerged Alert State, bool)
  + fan
    - id
    - fanSpeed (rpm) (int)
    - fanAlert (bool)
    - fanState (bool)
* system\_state
  + dateTime (Unix time) (int)
  + runtime (Unix time) (int)
  + cpuUsage (hundredth percent, e.g. 6562 means 65.62%) (int)
  + memTotalSize (bytes) (int)
  + memFreeSize (bytes) (int)
  + memUsage (hundredth percent, e.g. 3663 means 36.63%) (int)
  + storageTotalSize (bytes) (int)
  + storageFreeSize (bytes) (int)
  + storeUsage (percent, e.g. 32 means 0.32%) (int)
  + temperature (℃)
  + humidity (percent, e.g. 35 means 35%)
  + poeRealPower (PoE Real-time Power) (units unsure)
* flow\_monitor
  + Port (1-10)
    - Id (int)
    - name (string)
    - rxByte (Received Bytes) (sting)
    - rxError (Received Error Packets) (int)
    - rxPacket (Received Packets) (int)
    - txByte (Sent Bytes) (string)
    - txError (Sent Error Packets) (int)
    - txPacket (Sent Packets) (int)
* gps\_config
  + latitude (absolute value, float)
  + longitude (absolute value, float)
  + latitude\_h (‘N’ or ‘S’) (string)
  + longitude\_h (‘E’ or ‘W’) (string)
  + altitude (int)
  + altitude\_u (unit measurement, default in ‘m’ for meters above sea level, presumably) (string)

# Class: MQTT Message

* broker: emqx
* Topics (subscribing to sys/# will let you see all incoming & outgoing messages)
  1. Sent by IoT Box
     + sys/device/{deviceId}/connect
     + sys/device/{deviceId}/command\_reply/slot7-state
     + sys/device/{deviceId}/command\_reply/system-state
     + sys/device/{deviceId}/command\_reply/switch-port-statistics
     + sys/device/{deviceId}/command\_reply/netdmate-mobileGps-get
  2. Sent by MQTT Client
     + sys/service/{deviceId}/connect\_reply
     + sys/service/{deviceId}/command
* Message type (see PRESET\_COMMANDS in app.py)
  1. connect
     + type: connect (str)
     + version
     + session (str)
     + deviceId (str)
     + time (Current time, in Unix time)
     + data
       - securityMode: [“none”, “static”, “ecdh”] (only using “none” for this project)
  2. connect\_reply
     + type: connect\_reply
     + version
     + session
     + deviceId
     + time (Current time, in Unix time)
     + response
       - code: 0
       - message: success
     + data
       - security: none
  3. command\_reply
     + type: command\_reply
     + version
     + session
     + deviceId
     + time
     + response
       - code
       - message
  4. command
     + type: command
     + version
     + session
     + deviceId
     + time
     + data
       - service
       - method
       - parms

# Class: EMQX MQTT Broker

* image version: emqx/emqx:5.8.5
* web interface: <http://ip-address:18083>
* container\_name: emqx\_broker
* user
  + username: root
  + password: root1234
* service: emqx (used by MQTT Client to connect to broker)
* ports
  + MQTT default port: 1883
  + WebSocket port: 8083
  + Secure WebSocket: 8084
  + MQTT over SSL/TLS: 8883
  + EMQX Web Interface Dashboard: 18083

# Class: Grafana Dashboard

* Image version: grafana/grafana:11.5.2
* Container name: grafana\_dashboard
* Web Interface: <http://ip-address:3000>
* Service: grafana
* ports
  + Web Interface: 3000
* Admin user
  + Username: root
  + Password: root1234
* dashboards
  + IoT Box Dashboard
  + IoT Box Map View
* Datasources
  + Influxdb

# Class: InfluxDB database

* Image version: influxdb:2.7.11
* Container name: influxdb\_database
* Web Interface: <http://ip-address:8086>
* service: influxdb
* ports
  + Write to/Read from/Web interface: 8086
* user
  + username: root
  + password: root1234
  + All Access API Token: root1234
* organization: nms
* bucket: logs

# Class: InfluxDB Client

* database
  + url: http://influxdb:8086 (based on service)
  + org: nms
  + token: root1234
* bucket: logs
* point (data structure)
  + measurement: deviceId
    - default write precision: s (seconds)
    - time: time (from the received message)
    - field keys
      * [every key found in ‘response’ or ‘data’ of message, except ‘code’]
    - tag keys
      * [remaining keys not used in measurement, time, or fields]
        + Usually ‘session’, ‘version’ and ‘code’
    - values
      * [every value accompanying corresponding field key]
  + measurement: device\_locations
    - default write precision: s (seconds)
    - time: time (from the received message)
    - field keys
      * latitude (+/- value depending on [latitude\_h](#lat_h), float)
      * longitude (+/- value depending on [longitude\_h](#lon_h), float)
    - tag keys
      * deviceId
    - values
      * [every value accompanying field key]

# Class: Python app.py

* Image version: python:alpine
* Container name: python\_app (‘docker attach python\_app’ to view python terminal output)
* Service: python
* libraries
  + paho.mqtt==2.1.0
  + influxdb\_client==1.48.0
* global variables
  + box\_list (dict)
  + connected\_boxes (list)
  + lock (object) (For mutex threading)
  + stop\_event (object) (Signal to stop all threads and mqtt client)
  + PRESET\_COMMANDS
* threads
  + Main Thread
    - >>configure + manage MQTT client and other threads until stop\_event is set to True.
  + client.loop\_start()
    - >>handle MQTT communication and execute callbacks such as on\_connect(), on\_message(), and on\_disconnect().
  + publish\_connect\_reply()
    - >>whenever there is a IoT box that recently sent a “connect” message but is not found in box\_list, send a “connect\_reply” message to the MQTT broker. Add that box to connected\_boxes.
  + publish\_command\_loop()
    - >>for every box in connected\_boxes, send a list of “command” messages to the MQTT broker, pulling command data from PRESET\_COMMANDS.

# Class: MQTT Client

* [from paho.mqtt import client as mqtt\_client]
* Client ID: python-mqtt-tcp-client
* Username: root
* Password: root1234
* Broker: emqx
* Broker port: 1883
* MQTT Protocol: v3.1.1
* on\_connect() (Triggers when client senses emqx broker and attempts to connect to it)
  + SUB\_TOPIC: sys/device/# (Subscribed Topic(s))
  + If connection was sucessful & client is currently connected >> subscribe to Subscribed topic
* on\_disconnect() (Triggers when client disconnects from emqx broker)
  + First reconnection delay: 1s
  + Reconnection rate: Delay\*2 for every failed reconnect attempt
  + Maximum reconnection attempts before shutdown: 12
  + Maximum reconnection delay: 60s
  + >> Loop reconnection attempt with exponentially increasing delays until successful reconnection or Maximum reconnection attempts reached.
* on\_message() (Triggers when detecting message on Subscribed topic)
  + if message “type” is “connect” >> update box\_list
  + if message “type” is “command\_reply” >> flatten message payload and write data to InfluxDB in Point format to Line protocol