**Design document for accommodating the new requirement for New topic format in UWC:**

**Page 1:** TOC

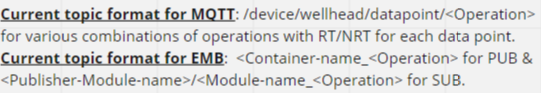
**Page 2**: Context setting, Requirement description in detail.

Colour coding & other conventions used in the architecture diagram.

**Page 3**: Detailed Architecture design diagram.

**Page 4, 5, 6**: Verbal tech explanation of the above architecture diagram. Step by step Logical flow end to end.

**Context:**



Current nature of MQTT topics in UWC:

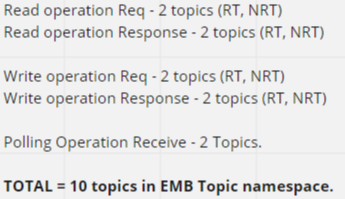
In UWC, Currently, there can be multiple & variable no: of MQTT topics in MQTT topic namespace.

Total no: of MQTT topics = (no: of datapoints to be queried)\*(no: of operations to be performed on that data point).

So, this means, the no: of MQTT topics are variable & not fixed.

Current nature of EMB (EIS MSG BUS) topics i.e. ZMQ topics in UWC:

Currently the no: of topics in EMB (ZMQ) in UWC is FIXED & GENERIC. i.e. the ZMQ EMB topics have been categorized based on the operation type & RT/Non-RT, like below.



Meaning, the messages on numerous & variable no: of MQTT topics gets converged & bucketized into one of the 8 (on-demand) EMB ZMQ topics.

Currently there are 1 on 1 threads & Queues for these ZMQ topics in UWC, which form the basis of the prioritization a per RT/NRT & operation Type.

**Requirement in detail**: Now the Requirement is to not have these EMB topic formats (as above) going forward. Instead the MQTT topics should be used for EMB too. Meaning, there won’t be any “GENERIC & FIXED” topics in EMB topic namespace and it will be replaced by the numerous & variable no: MQTT topic names in ZMQ EMB too.

**PROPOSED ARCHITECTURE FOR ACCOMODATING THIS REQUIREMENT**:

Lot of architectural changes are needed as described below: (Verbal explanation & Logic Flow explained below the architecture diagram).



**WRT above architecture diagram, below is the flow**:

**TOP TO DOWN FLOW I.E. MQTT CLIENT BEING AT TOP TILL MODBUS SLAVE DEVICE AT BOTTOM:**

1. **MQTT-client**: MQTT client (at the top), sends(publishes) the JSON payload on topic “/device/wellhead/datapoint/operation”. This topic format is used by the existing MQTT clients.
2. **Within MQTT-Export module**: MQTT-Subscriber/Listener in the MQTT-export module subscribes to all messages (\*) from the MQTT broker endpoint. The subscription happens on the old MQTT topic format (/device/wellhead/datapoint/operation) to maintain backward compatibility for the mqtt-client apps.
3. The listener module also bucketizes all the incoming messages into 4 queues in MQTT-Export module. (4 queues being – RT-writeReq, NRT-writeReq, RT-readReq, NRT-readReq queues).
4. There are 4 threads operating on these 4 queues on 1 on 1 basis.
5. These 4 threads do a **topic mapping from MQTT topic format (device/wellhead/datapoint/operation) to EMB ZMQ topic format <RT|NRT>/<operation>/device/wellhead/datapoint.**

Example: **/flowmeter/PL0/DP1/write** gets mapped to **/RT/write/flowmeter/PL0/DP1.**

Information regarding RT/NRT is added to topic name itself. And this information regarding RT/NRT is added to EMB ZMQ topic so that the Modbus containers spend less time in parsing the JSON payload to know the RT/NRT nature of the message.

1. These 4 threads do publish on the Mapped topic format **<RT|NRT>/<operation>/device/wellhead/datapoint/operation (example: RT/write/flowmeter/PL0/DP1).**
2. The EMB (ZMQ) publish happens on 2 brokers. 1 broker is meant for RT operation (which is give high priority). Another broker is given normal priority for handling NRT operations. So the MQTT-Export or other apps (like ControlLoop) which sit on EMB do publish/subscribe on the RT-broker, NRT-broker based on the message being RT/NRT.
3. There are 4 threads (attached to 4 SUB sockets) on the Modbus-master container (RT-read, NRT-read, RT-write,NRT-write), which subscribe from **RT-Broker** endpoint to all topics of format **RT/<operation>/\* (**as shown in the diagram) for RT operations**. And NRT-Broker** endpoint to all topics of format **NRT/<operation>/\*** for NRT operations**.**
4. The 4 subscribers on modbus-master containers do work on the ZMQ internal queues (High watermark in ZMQ terminology). The default value of HWM to be set to 10000 in UWC global configuration.yml file.

Since these4 threads do work on the ZMQ internal queues directly while subscribing the messages from ZMQ, no explicit queue needs to be maintained on Modbus-master (in top to bottom flow).

These 4 ZMQ internal queues along with the priority brokers (RT and NRT broker) play a main role in maintaining the prioritization logic for all operations.

1. These 4 threads do make calls to appropriate Modbus-Stack APIs to perform the required action on modbus slave devices.

**BOTTOM TO TOP FLOW I.E. MODBUS-SLAVE DEVICE BEING AT THE BOTTOM TO MQTT CLIENT BEING AT TOP:**

1. The responses from Modbus-stack is received on 6 callback functions. (RT-writeResponse, RT-readResponse, RT-pollResponse, NRT-writeResponse, NRT-readResponse, NRT-PollResponse).
2. There are 6 queues maintained in modbus-master (1 on 1 for each callback). These 6 callback functions do push the incoming response messages into these 6 queues (custom app queues maintained inside the modbus-master process space).
3. Each of these 6 app-queues in modbus-master have a corresponding thread (1 on 1 basis) , which pops the messages from it’s corresponding queue & publishes on the same topic on which it was received. This publish happens on 2 broker endpoints (priority RT broker & normal NRT broker).

**<RT|NRT>/<writeResponse|readResponse|pollResponse>/<Absolute-path-to-datapoint>**

for example:

1. Response topic is **RT/writeResponse/flowmeter/PL0/DP1** for the request topic **RT/write/flowmeter/PL0/DP1.**
2. Response topic is **NRT/writeResponse/flowmeter/PL0/DP1** for the request topic **NRT/write/flowmeter/PL0/DP1.**
3. Response topic is **RT/readResponse/flowmeter/PL0/DP1** for the request topic **RT/read/flowmeter/PL0/DP1.**
4. Response topic is **NRT/readResponse/flowmeter/PL0/DP1** for the request topic **NRT/read/flowmeter/PL0/DP1.**
5. Response topic is **RT/pollResponse/flowmeter/PL0/DP1** for the RT polling operation. There is no request topic for polling as it is timer triggered. (Timer is inside modbus=master, configured inside UWc config YML files).
6. Response topic is **NRT/pollResponse/flowmeter/PL0/DP1** for NRT polling operation. There is no request topic for polling as it is timer triggered. (Timer is inside modbus=master, configured inside UWc config YML files).

**MQTT-Export (in bottom to top flow)**:

1. MQTT-Export has 6 subscribers (SUB sockets) which read from 6 ZMQ internal queues. (Again the HWm for these 6 queues is maintained at 10000 on receiver SUB socket side).
2. There are 6 subscriber/listener threads on MQTT-export conatiner which subscribe on the topic “**RT/writeResponse/\*, RT/readResponse/\*, RT/PollResponse/\***” on RT broker for receiving all RT response messages.

Similiarly the SUB topics used by these subscriber threads for NRT response messages are “**NRT/writeResponse/\*, NRT/readResponse/\*, NRT/pollResponse/\***”.

Note: the “\*” (wildcard in topic name for ZMQ would subscribeusing prefix matching on ZMQ topic format mentioned).

1. **(OPTIONAL STEP exclusively For Control-loop or any other apps sitting directly on EMB):**

No back mapping as below is needed. Instead these apps just consume the subscribed in step 5 and no need of any of the below steps.

THE BELOW STEPS ARE NEEDED ONLY FOR MQTT CLIENTS & NOT NEEDED FOR APPS SITTING ON EMB (ZMQ):

1. To maintain backward compatibility with existing MQTT topic formats, these 6 threads do map back the topic **“RT/Operation/<Absolute-path-to-datapaoint>”** to **“<Absolute-path-to-datapaoint>/<operation>”.**
2. These 6 threads do push the JSON response payloads from corresponding ZMQ internal queue (after subscribing) into 6 app queues (for the 6 operations RT-ReasResponse, RT-writeResponse, RT-pollResponse, NRT-readResponse, NRT-writeResponse, NRT-pollResponse). The back-mapped topics are also stored in these 6 queues (as these back mapped topics are needed for publishing to MQTT-client).
3. Another set of 6 threads operate on these 6 queues (1 on 1 basis), which pop the messagesd 1 by 1 and publish onto the MQTT-client via MQTT-broker.

**Conclusion**:

1. All the updates on the design, needs to be captured in the UWC user guide & HLD appropriately.
2. Thread priorities & the ZMQ broker priorities needs to be captured in UWC global config file.
3. The MQTT topic format is **device/wellhead/datapoint/operation** & new proposed (in this design) EMB topic format is **<RT|NRT>/<operation>/device/wellhead/datapoint.**
4. The old MQTT topic format is still maintained to have backward compatibility with existing MQTT clients. MQTT-Export is responsible for doing the mapping from MQTT topic format to EMB topic format (in top to bottom flow) & back-map from EMB topic format to MQTT topic format, for maintaining this backward compatibility.

However, going forward for new apps (like Control apps) which sit directly on EMB needs to adhere to the EMB topic format as defined above.