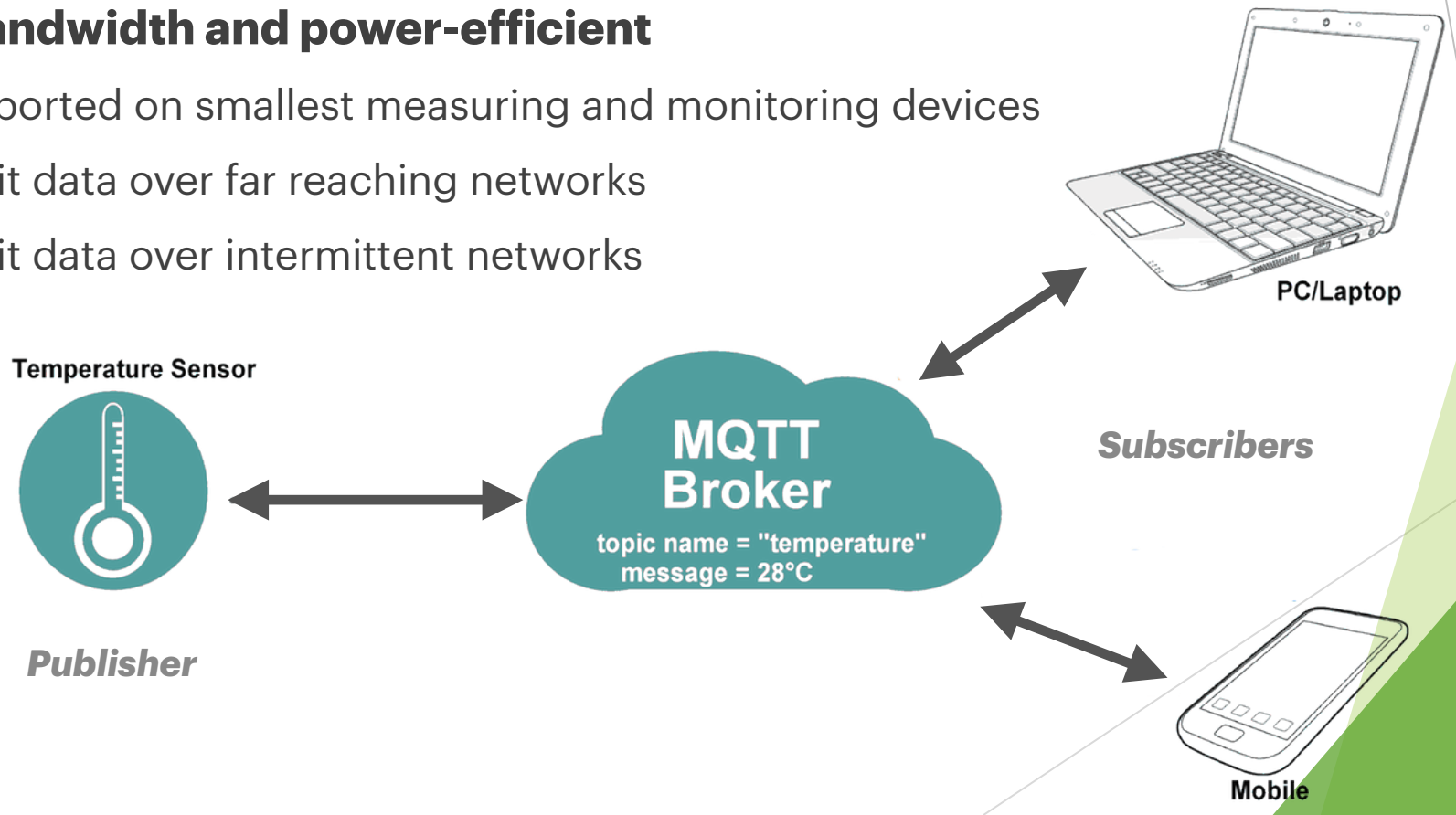
Message Queue Telemetry Transport

http://mqtt.org/

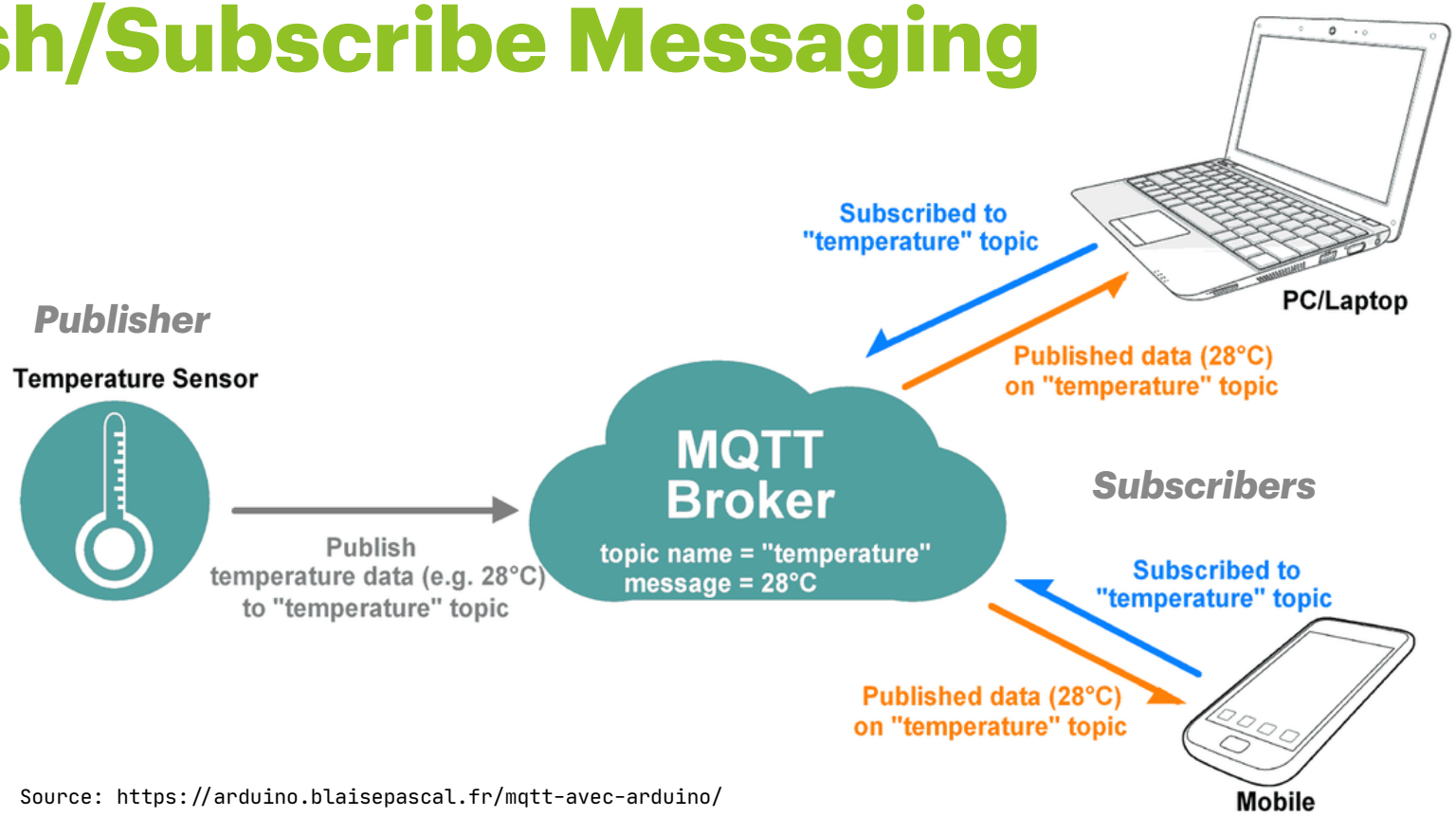
MQTT - Highlights

used whe connection come an go

- **A machine-to-machine (M2M) / IoT connectivity protocol**
- MQTT is an Event-based IoT middleware (one to many)
  - Publish/subscribe messaging transport protocol
  - MQTT (TCP/IP, ...), MQTT-SN (UDP or bluetooth)
- **Lightweight, bandwidth and power-efficient**
  - It can be supported on smallest measuring and monitoring devices
  - It can transmit data over far reaching networks
  - It can transmit data over intermittent networks



Publish/Subscribe Messaging



Source: https://arduino.blaisepascal.fr/mqtt-avec-arduino/

- MQTT protocol has three important characteristics:
  1. **Publishers/subscribers loosing coupling**, allowing for more flexible applications,
  2. **One to many broadcasting**
  3. **The number of subscribers can change over time**

MQTT Topics

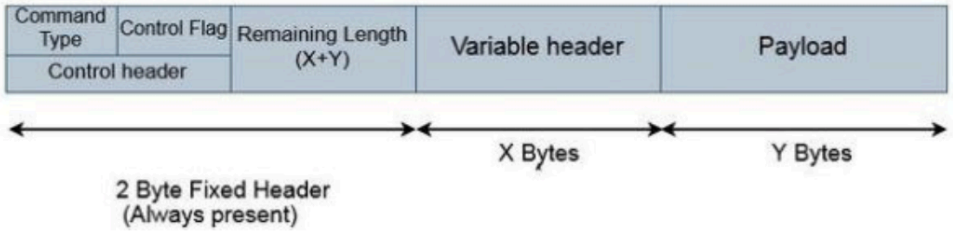
- **A topic forms a namespace**
- **Hierarchically organized (sub-topics separated by "/")**
  - `<country>/<region>/<town>/<postcode>/<house#>/powerConsumption`
  - `<country>/<region>/<town>/<postcode>/<house#>/alarmState`
- **Subscribers can use wildcards (not the publishers)**
  - "+" : single-level; can appear anywere in the namespace
  - "#" : multi-level; must appear at the end of the namespace
  - Wildcards must appear next to a separator

- **Examples**
  - `UK/Hants/Hursley/SO212JN/1/powerConsumption`
    - ➡ Power consumption for house#1 in Hursley
  - `UK/Hants/Hursley/+/+/powerConsumption`
    - ➡ Power consumption for all houses in Hursley
  - `UK/Hants/Hursley/SO212JN/1/#`
    - ➡ Power consumption and alarm state for house#1 in Hursley

Ideal for constrained networks

(low bandwidth, high latency and intermittent connections)

- **MQTT messages are delivered asynchronously**
- **MQTT control packet headers are kept as small as possible.**
- Each control packet consists a **fixed header**, a **variable header** and a **payload**.

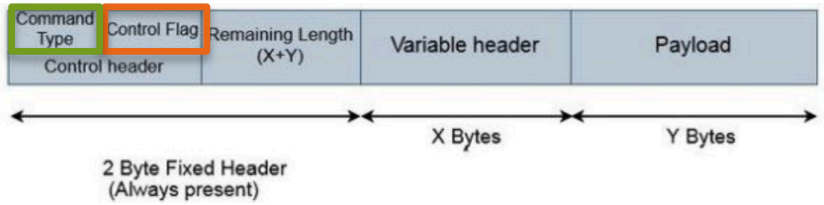


- Not all the control packets have the variable header and payload headers (depends on the command type).
- **Up to 256 MB payload can be attached to the packets.**

Small header overhead makes this protocol appropriate for IoT with constrained networks.

Example : PUBLISH command

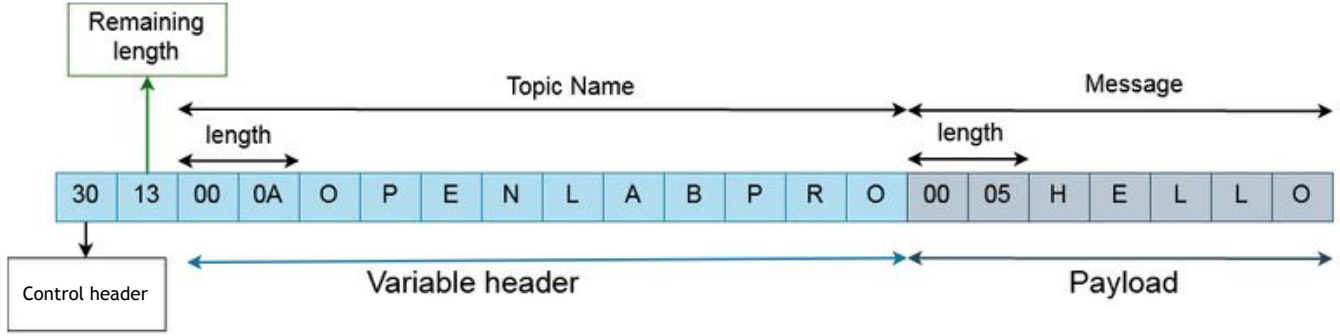
- The **control flag bits** define :
  - Whether the message is **DUP**licated/resent (one bit)
  - Quality-of-Service (**QOS**) level (two bits)
  - Message retention flag (one bit).



Command Type

Name	Value	Direction of flow	Description
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server	Client request to connect to Server
CONNACK	2	Server to Client	Connect acknowledgment
PUBLISH	3	Client to Server or Server to Client	Publish message
PUBACK	4	Client to Server or Server to Client	Publish acknowledgment
PUBREC	5	Client to Server or Server to Client	Publish received (assured delivery part 1)
PUBREL	6	Client to Server or Server to Client	Publish release (assured delivery part 2)
PUBCOMP	7	Client to Server or Server to Client	Publish complete (assured delivery part 3)
SUBSCRIBE	8	Client to Server	Client subscribe request
SUBACK	9	Server to Client	Subscribe acknowledgment
UNSUBSCRIBE	10	Client to Server	Unsubscribe request
UNSUBACK	11	Server to Client	Unsubscribe acknowledgment
PINGREQ	12	Client to Server	PING request
PINGRESP	13	Server to Client	PING response
DISCONNECT	14	Client to Server	Client is disconnecting
Reserved	15	Forbidden	Reserved

Example (sending “HELLO” on topic “OPENLABPRO”)



With QoS = 0, RETAIN=0 and DUP=0

source : https://openlabpro.com/guide/mqtt-packet-format/

Control Flag bits

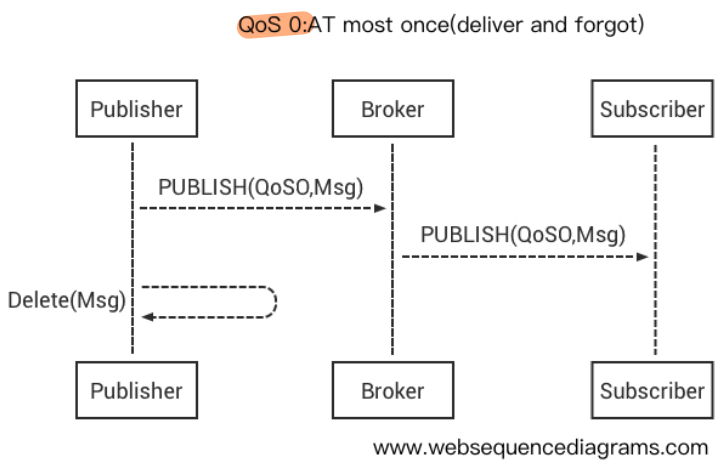
Control Packet	Fixed header flags	Bit 3	Bit 2	Bit 1	Bit 0
CONNECT	Reserved	0	0	0	0
CONNACK	Reserved	0	0	0	0
PUBLISH	Used in MQTT 3.1.1	DUP[1]	QoS[2]	QoS[2]	RETAIN[3]
PUBACK	Reserved	0	0	0	0
PUBREC	Reserved	0	0	0	0
PUBREL	Reserved	0	0	1	0
PUBCOMP	Reserved	0	0	0	0
SUBSCRIBE	Reserved	0	0	1	0
SUBACK	Reserved	0	0	0	0
UNSUBSCRIBE	Reserved	0	0	1	0
UNSUBACK	Reserved	0	0	0	0
PINGREQ	Reserved	0	0	0	0
PINGRESP	Reserved	0	0	0	0
DISCONNECT	Reserved	0	0	0	0

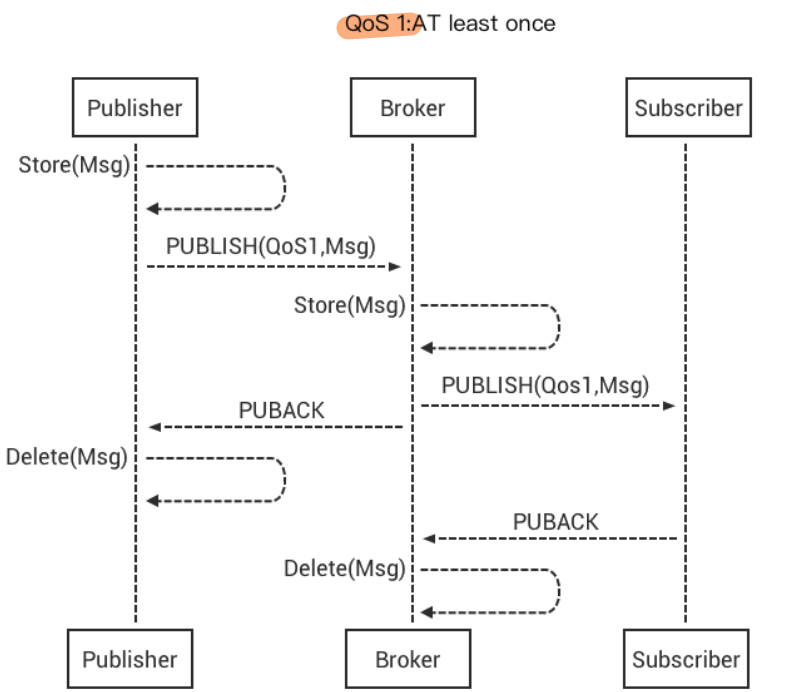
Quality of Service (QoS)

- **Quality of service (QoS) levels determine how each MQTT message is delivered**

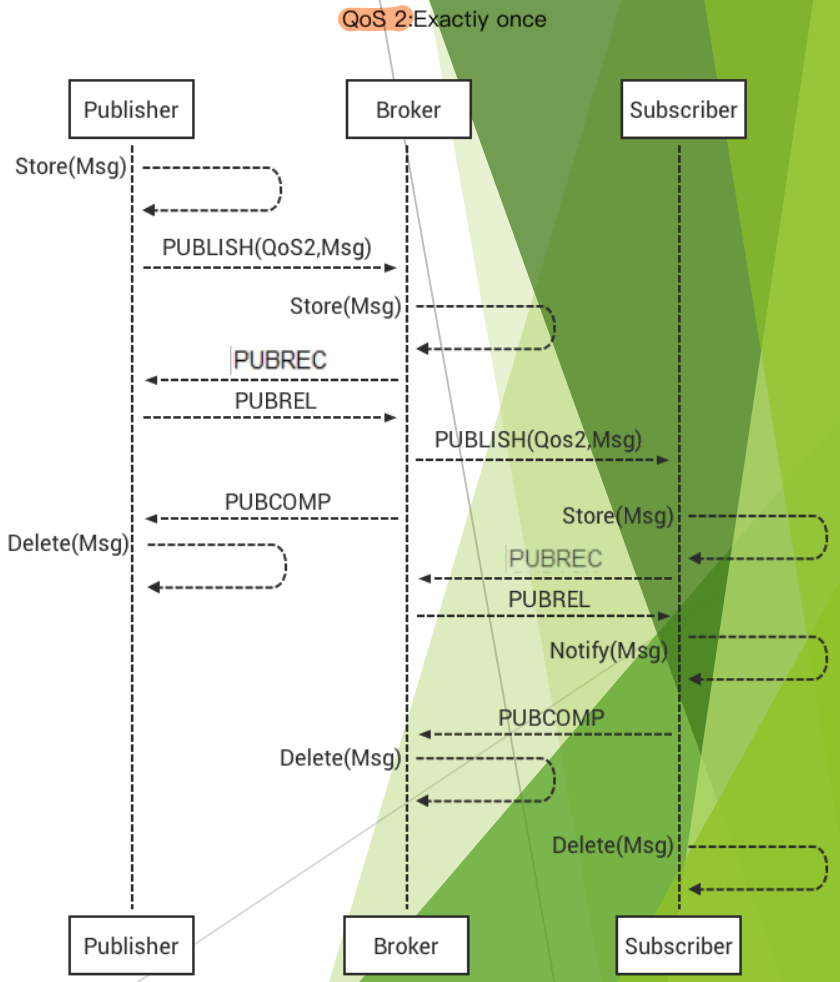
- Three QoS levels for message delivery could be achieved using MQTT:
  - **QoS 0 (At most once)**
    - Messages are delivered according to the best efforts of the operating environment. **Message loss can occur, NO GUARANTEES**.
  - **QoS 1 (At least once)**
    - Messages are **GUARANTEED** to arrive but **DUPLICATES** can occur (retransmission if ack not received).
  - **QoS 2 (Exactly once)**
    - Message are **GUARANTEED** to arrive exactly once (**NO DUPLICATES**).

Quality of Service (QoS)





// DA DELIVERED AT LEASTE ONCE
(CAUSE REDOUNDANCY)



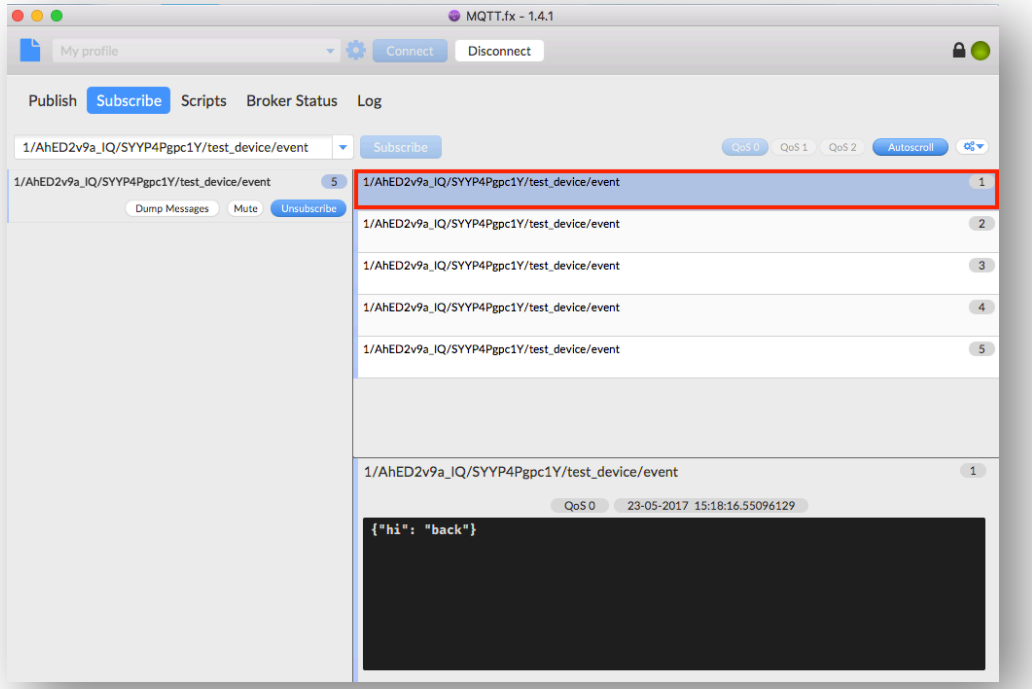
//DATA DELIVERED EXACTLY ONCE

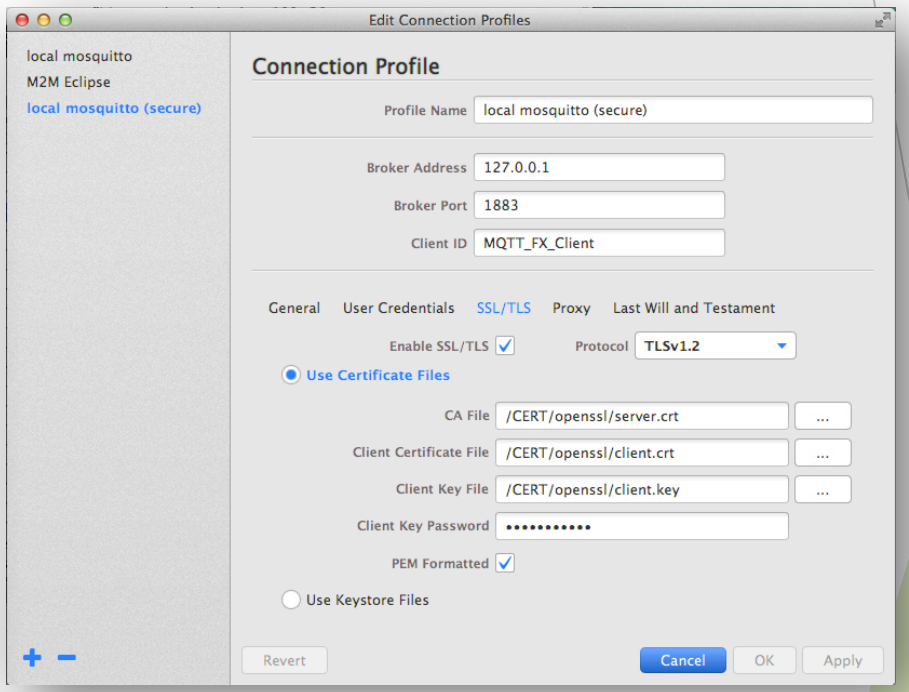
Additional concepts - USEFUL FEATURES

- **Clean session (default true)**
  - At **connection** to the broker, indicate if subscriptions have to be removed or kept on disconnection.
  - If true, subsequent messages with Qos=1/2 are stored for delivery after reconnection.
- **Retained messages**
  - **Published messages are kept on broker side**. Usefull if data is rarely updated and new clients connect to the topic.
- **Last will & Testament**
  - In the event of an ungracefull disconnection of a client, the broker sends a last will message to the subscribed clients.

MQTT Test/Debug tool

- https://mqttfx.jensd.de/





MQTT Brokers in the Cloud

- **EMQ X MQTT (https://www.emqx.com/en/mqtt/public-mqtt5-broker)**
  broker.emqx.io
- **HiveMQ (https://www.hivemq.com/mqtt-cloud-broker/)**
  broker.hivemq.com
- **MQTTHQ (https://mqtthq.com)**
  public.mqtthq.com
- **Mosquitto (https://test.mosquitto.org)**
  test.mosquitto.org

MQTT Clients and APIs

- **Open Source clients available in Eclipse Paho project**
  - C, C++, Java, JavaScript, Lua, Python and Go
- **Clients for other languages are available, see mqtt.org/ software**
  - E.g. Delphi, Erlang, .Net, Objective-C, PERL, PHP, Ruby
  - Not all the client libraries listed on **mqtt.org/** are stable/mature. Some are at an early or experimental stage of development, whilst others are stable and mature.