



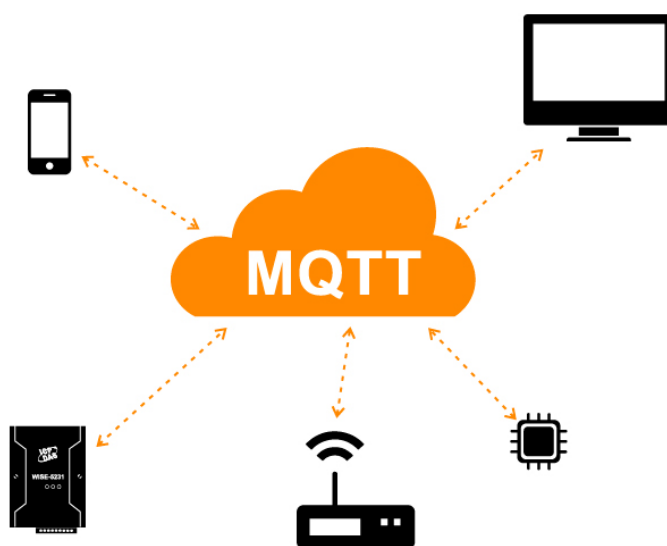
---

# Client MQTT - Documentație

---

**Baltariu Ionuț-Alexandru**  
Grupa 1305A

**Bejenariu Răzvan-Marius**  
Grupa 1306A



## 1 Scurtă descriere:

MQTT (**M**essage **Q**ueue **T**elemetry **T**ransport) este un standard de comunicație prin internet, inventat și dezvoltat de IBM în 1999, care definește un protocol de transport al mesajelor prin internet (prin TCP/IP în general), între un server ("message broker") și mai mulți clienți, pe modelul "publicare și abonare" ("**publish and subscribe**").

Este un protocol de conectivitate "mașină - mașină" (M2M) / "**I**nternet of **T**hings", conceput a fi simplu și flexibil, cu cerințe minime de resurse, util pentru conexiunile cu locații la distanță și perfect pentru toate tipurile de aplicații IoT, fie că sunt automatizări industriale (**I**IoT) sau automatizări casnice/rezidențiale/de consum (**C**onsumer **I**ot).

Este, de asemenea ideal și pentru **aplicații mobile**, datorită dimensiunilor mici, consumului redus de energie, pachetelor de date minimizate și distribuției eficiente a informației către unul sau mai mulți receptori.

## 2 Cerințe

### 1. 2 view-uri: unul pentru publicare și altul pentru abonare

În momentul în care un client se conectează, acesta se poate abona la topicurile de care este interesat, pentru a primi informații de la toate dispozitivele care furnizează acea informație sau poate publica diferite topicuri brokerului pentru ca acesta să transmită informația la rândul său.

### 2. Autentificare cu utilizator și parolă

Acest feature va fi posibil prin trimiterea informațiilor respective într-un pachet folosit la conectare. Brokerul va gestiona întregul proces de autentificare la acesta de către publisheri.

### 3. Listă de abonare configurabilă:

Vom implementa un mecanism prin care clientul să poată să se aboneze/dezaboneze de la anumite topicuri. Acest lucru va fi posibil prin modificarea unui fișier de configurări sau direct prin interfața grafică a aplicației.

### 4. Publicare manuală sau automată:

Publisherul va avea posibilitatea de a posta informații atât la cerere, cât și automat la un interval de timp (configurabil), pentru a menține alți utilizatori informați.

### 5. Mecanism KeepAlive:

Mecanismul KeepAlive se utilizează pentru a verifica dacă un client este conectat la broker, ambele părți știind de existența celeilalte. În momentul în care clientul se conectează la broker, aceștia pot sta maxim o anumită perioadă fără a interacționa unul cu celălalt.

Dacă această perioadă este depășită clientul trebuie să trimită un pachet **PINGREQ** brokerului, la care brokerul va trimite la rândul său un pachet **PINGRESP**, iar dacă aceste pachete nu ajung într-un anumit interval de timp, clientul va fi deconectat de la broker.

### 6. Implementare QoS 0,1,2:

**QoS**(Quality of Service) se referă la conexiunea dintre client și broker, și reprezintă o modalitate de garantare a livrării mesajului.

**QoS 0:** rapid, dar nesigur, mesajul nu este salvat nicăieri, deci, mesajul este pierdut în cazul în care informația nu ajunge la broker.

**QoS 1:** mesaje se pot trimite de mai multe ori, chiar dacă nu este nevoie, după trimiterea informației către broker, se așteaptă o confirmare de la acesta(**PUBACK**), în cazul în care clientul primește confirmarea, informația este ștearsă din coadă, în caz contrar, mesajul se trimite până se primește o confirmare de primire de la broker.

**QoS 2:** ne garantează livrarea mesajului o singură dată, chiar dacă operațiunea este mai lentă. Se petrece același lucru ca și la QoS 1 cu diferența că se mai întâmplă câteva verificări, înainte ca brokerul să trimită informația mai departe.

### 7. Mecanismul Last Will:

Este o modalitate de a-i înștiința pe ceilalți clienți în momentul în care cineva se deconectează într-un mod neașteptat. Fiecare client își poate specifica mesajul ce va fi trimis în momentul deconectării de către broker, mesaj, ce va fi trimis tuturor clienților abonați la acel topic. În cazul deconectării intenționate apare **MQTT Disconnect**.

## 3 Implementare

### 3.1 Reprezentarea datelor

Tip de dată	Reprezentarea în cadrul MQTT
Întreg - 2 Octeți ( <b>short int</b> )	Big Endian: Octetul 1 MSB; Octetul 2 LSB
Întreg - 4 Octeți ( <b>int</b> )	Big Endian: $3 \times MSB$ ; $1 \times LSB$
String	Lungimea pe 2 Octeți(MSB & LSB) + String codat UTF-8
Date binare	Lungimea pe 2 Octeți(MSB & LSB) + Octeții corespunzători datelor

## 3.2 Structura pachetelor de control

### 3.2.1 Fixed Header

Fiecare pachet are un **Fixed Header**, format din:

- **Byte 1:** primii 4 biți pentru **tipul pachetului**
- **Byte 1:** ultimii 4 biți pentru **flaguri** specifice pachetelor
- **Byte 2 -> Byte N:** Remaining Length - **dimensiunea** (în Octeți) câmpurilor următoare

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type				Flags specific to each MQTT Control Packet type			
byte 2...	Remaining Length							

Figure 1: Structură Fixed Header

### 3.2.2 Variable Header

Majoritatea pachetelor au un **Variable Header**, mai puțin PINGREQ și PINGRESP, situat între Fixed Header și Payload, acesta fiind diferit de la pachet la pachet.

	Description	7	6	5	4	3	2	1	0
Topic Name									
byte 1	Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	Length LSB (3)	0	0	0	0	0	0	1	1
byte 3	'a' (0x61)	0	1	1	0	0	0	0	1
byte 4	'/' (0x2F)	0	0	1	0	1	1	1	1
byte 5	'b' (0x62)	0	1	1	0	0	0	1	0
Packet Identifier									
byte 6	Packet Identifier MSB (0)	0	0	0	0	0	0	0	0
byte 7	Packet Identifier LSB (10)	0	0	0	0	1	0	1	0
Property Length									
byte 8	No Properties	0	0	0	0	0	0	0	0

Figure 2: Exemplu Variable Header (PUBLISH)

### 3.2.3 Payload

În acest câmp sunt introduse datele care trebuie să fie trimise efectiv către server. Un exemplu ar fi, pentru pachetul CONNECT, username-ul (atât timp cât flag-ul pentru username este pus pe 1).

Precum Variable Header, câmpul Payload nu este prezent la toate pachetele, ci doar la 6 dintre ele (în cadrul unui pachet fiind chiar opțional).

Description	7	6	5	4	3	2	1	0
Topic Filter								
byte 1	Length MSB							
byte 2	Length LSB							
bytes 3..N	Topic Filter							
Subscription Options								
	Reserved		Retain Handling		RAP	NL	QoS	
byte N+1	0	0	X	X	X	X	X	X

Figure 3: Exemplu Structură Payload (SUBSCRIBE)

### 3.2.4 Reason Code

Este un octet care indică rezultatul operației.

În cazul pachetelor SUBACK sau UNSUBACK pot exista mai mulți octeți care să conțină Reason Codes.

### 3.3 Pachetele de control

În MQTT, versiunea 5.0, există 15 pachete de control. Acestea sunt ilustrate în **Figure 4**.

Name	Value	Direction of flow	Description
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server	Connection request
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 (QoS 1)
PUBREC	5	Client to Server or Server to Client	Publish received (QoS 2 delivery part 1)
PUBREL	6	Client to Server or Server to Client	Publish release (QoS 2 delivery part 2)
PUBCOMP	7	Client to Server or Server to Client	Publish complete (QoS 2 delivery part 3)
SUBSCRIBE	8	Client to Server	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 or Server to Client	Disconnect notification
AUTH	15	Client to Server or Server to Client	Authentication exchange

Figure 4: Pachetele MQTT

În cadrul aplicației, acestea sunt reprezentate de către o **clasă**, în care găsim metoda de ”creare” efectivă a pachetului (serializare) - pack.

Datele din Variable Header și/sau Payload vor fi introduse în pachet prin intermediul constructorului (*\_\_init\_\_*) sau al altor metode auxiliare, iar conținutul pachetului va fi serializat într-un *bytearray* pentru a putea fi trimis către un server.