EC200x-CN&EC800x-CN
MQTT Application Note

LTE Standard Module Series

Version: 1.0.0

Date: 2023-02-01

Status: Preliminary

At Quectel, our aim is to provide timely and comprehensive services to our customers. If you require any assistance, please contact our headquarters:

Quectel Wireless Solutions Co., Ltd.
Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local offices. For more information, please visit:
http://www.quectel.com/support/sales.htm.

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/technical.htm.
Or email us at: support@quectel.com.

Legal Notices

We offer information as a service to you. The provided information is based on your requirements and we make every effort to ensure its quality. You agree that you are responsible for using independent analysis and evaluation in designing intended products, and we provide reference designs for illustrative purposes only. Before using any hardware, software or service guided by this document, please read this notice carefully. Even though we employ commercially reasonable efforts to provide the best possible experience, you hereby acknowledge and agree that this document and related services hereunder are provided to you on an “as available” basis. We may revise or restate this document from time to time at our sole discretion without any prior notice to you.

Use and Disclosure Restrictions

License Agreements

Documents and information provided by us shall be kept confidential, unless specific permission is granted. They shall not be accessed or used for any purpose except as expressly provided herein.

Copyright

Our and third-party products hereunder may contain copyrighted material. Such copyrighted material shall not be copied, reproduced, distributed, merged, published, translated, or modified without prior written consent. We and the third party have exclusive rights over copyrighted material. No license shall be granted or conveyed under any patents, copyrights, trademarks, or service mark rights. To avoid ambiguities, purchasing in any form cannot be deemed as granting a license other than the normal non-exclusive, royalty-free license to use the material. We reserve the right to take legal action for noncompliance with abovementioned requirements, unauthorized use, or other illegal or malicious use of the material.

Trademarks

Except as otherwise set forth herein, nothing in this document shall be construed as conferring any rights to use any trademark, trade name or name, abbreviation, or counterfeit product thereof owned by Quectel or any third party in advertising, publicity, or other aspects.

Third-Party Rights

This document may refer to hardware, software and/or documentation owned by one or more third parties ("third-party materials"). Use of such third-party materials shall be governed by all restrictions and obligations applicable thereto.

We make no warranty or representation, either express or implied, regarding the third-party materials, including but not limited to any implied or statutory, warranties of merchantability or fitness for a particular purpose, quiet enjoyment, system integration, information accuracy, and non-infringement of any third-party intellectual property rights with regard to the licensed technology or use thereof. Nothing herein constitutes a representation or warranty by us to either develop, enhance, modify, distribute, market, sell, offer for sale, or otherwise maintain production of any our products or any other hardware, software, device, tool, information, or product. We moreover disclaim any and all warranties arising from the course of dealing or usage of trade.

Privacy Policy

To implement module functionality, certain device data are uploaded to Quectel's or third-party's servers, including carriers, chipset suppliers or customer-designated servers. Quectel, strictly abiding by the relevant laws and regulations, shall retain, use, disclose or otherwise process relevant data for the purpose of performing the service only or as permitted by applicable laws. Before data interaction with third parties, please be informed of their privacy and data security policy.

Disclaimer

a) We acknowledge no liability for any injury or damage arising from the reliance upon the information.
b) We shall bear no liability resulting from any inaccuracies or omissions, or from the use of the information contained herein.
c) While we have made every effort to ensure that the functions and features under development are free from errors, it is possible that they could contain errors, inaccuracies, and omissions. Unless otherwise provided by valid agreement, we make no warranties of any kind, either implied or express, and exclude all liability for any loss or damage suffered in connection with the use of features and functions under development, to the maximum extent permitted by law, regardless of whether such loss or damage may have been foreseeable.
d) We are not responsible for the accessibility, safety, accuracy, availability, legality, or completeness of information, advertising, commercial offers, products, services, and materials on third-party websites and third-party resources.

About the Document

Revision History

Version	Date	Author	Description
-	2023-02-01	Luffy LIU/ Larson LI	Creation of the document
1.0.0	2023-02-01	Luffy LIU/ Larson LI	Preliminary

Contents

Table Index

1 Introduction

Quectel LTE Standard EC200M-CN, EC200N-CN, EC800M-CN and EC800N-CN modules support MQTT function. Message Queuing Telemetry Transport (MQTT) is a broker-based publish/subscribe messaging protocol designed to be open, simple, lightweight, and easy to implement. It is designed for connections with remote locations where a "small code footprint" is required or the network bandwidth is limited.

This document explains how to use the MQTT function of the following Quectel modules through AT commands.

1.1. Applicable Modules

Table 1: Applicable Modules

Module Family	Module
EC200x	EC200M-CN
	EC200N-CN
EC800x	EC800M-CN
	EC800N-CN

2 MQTT Data Interaction

This chapter illustrates the data interaction mechanism of the MQTT function.

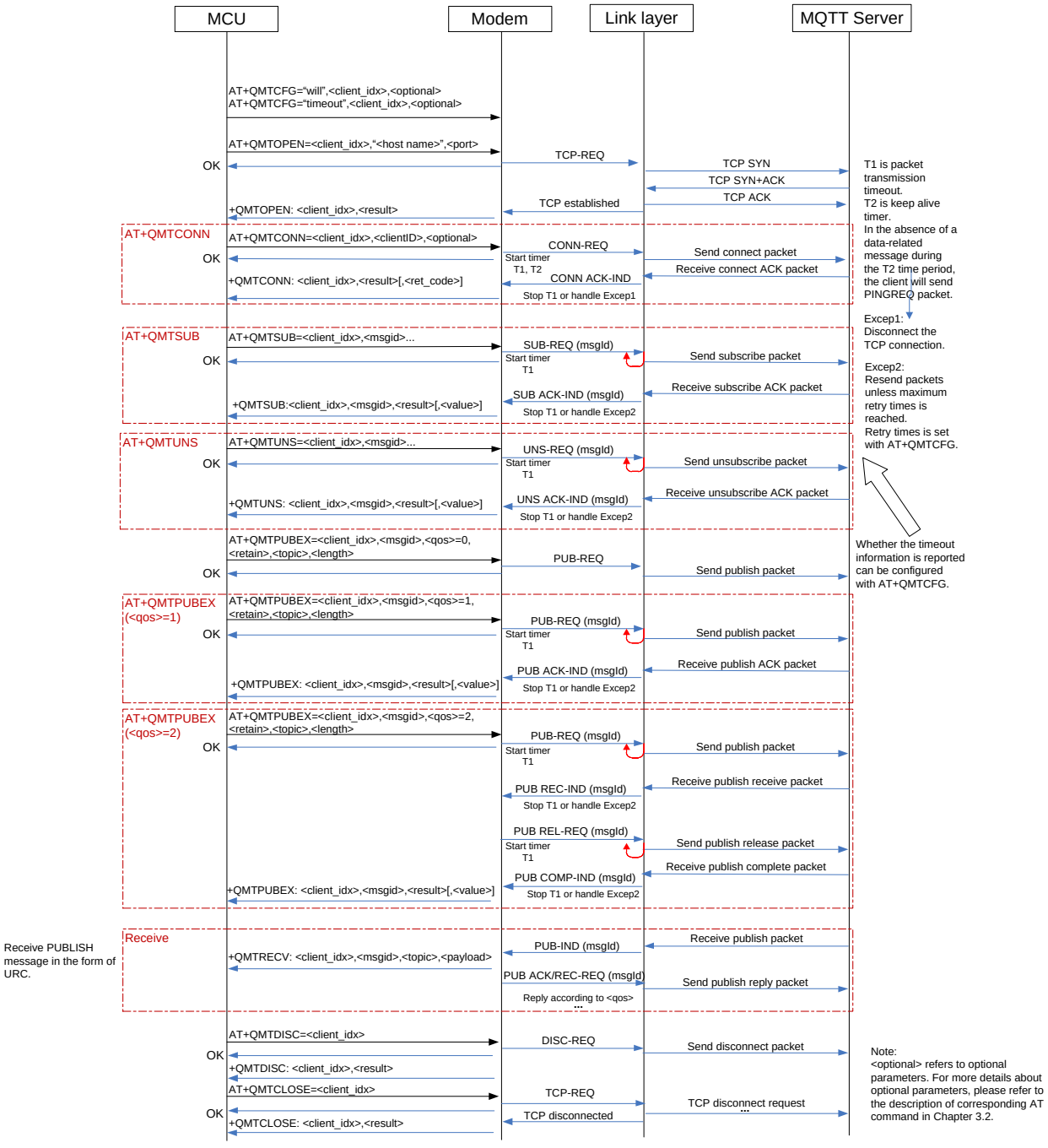

Figure 1: MQTT Data Interaction Diagram

3 MQTT Related AT Commands

3.1. AT Command Introduction

3.1.1. Definitions

- **<CR>** Carriage return character.
- **<LF>** Line feed character.
- **<...>** Parameter name. Angle brackets do not appear on the command line.
- **[...]** Optional parameter of a command or an optional part of TA information response. Square brackets do not appear on the command line. When an optional parameter is not given in a command, the new value equals its previous value or the default settings, unless otherwise specified.
- **Underline** Default setting of a parameter.

3.1.2. AT Command Syntax

All command lines must start with **AT** or **at** and end with **<CR>**. Information responses and result codes always start and end with a carriage return character and a line feed character: **<CR><LF><response><CR><LF>**. In tables presenting commands and responses throughout this document, only the commands and responses are presented, and **<CR>** and **<LF>** are deliberately omitted.

Table 2: Types of AT Commands

Command Type	Syntax	Description
Test Command	**AT+<cmd>=?**	Test the existence of the corresponding command and return information about the type, value, or range of its parameter.
Read Command	**AT+<cmd>?**	Check the current parameter value of the corresponding command.
Write Command	**AT+<cmd>=<p1>[,<p2>[,<p3>[...]]]**	Set user-definable parameter value.
Execution Command	**AT+<cmd>**	Return a specific information parameter or perform a specific action.

3.2. Declaration of AT Command Examples

The AT command examples in this document are provided to help you familiarize with AT commands and learn how to use them. The examples, however, should not be taken as Quectel's recommendation or suggestions about how you should design a program flow or what status you should set the module into. Sometimes multiple examples may be provided for one AT command. However, this does not mean that there exists a correlation among these examples and that they should be executed in a given sequence.

3.3. Description of MQTT Related AT Commands

3.3.1. AT+QMTCFG Configure Optional Parameters of MQTT

This command configures optional parameters of MQTT.

AT+QMTCFG Configure Optional Parameters of MQTT	
Test Command **AT+QMTCFG=?**	Response **+QMTCFG: "version",(**range of supported **<client_idx>**s**),(**list of supported **<vsn>**s**)** **+QMTCFG: "pdpcid",(**range of supported **<client_idx>**s**),(**range of supported **<cid>**s**)** **+QMTCFG: "ssl",(**range of supported **<client_idx>**s**),(**list of supported **<SSL_enable>**s**),(**range of supported **<SSL_ctx_idx>**s**)** **+QMTCFG: "keepalive",(**range of supported **<client_idx>**s**),(**range of supported **<keep_alive_time>**s**)** **+QMTCFG: "session",(**range of supported **<client_idx>**s**),(**list of supported **<clean_session>**s**)** **+QMTCFG: "timeout",(**range of supported **<client_idx>**s**),(**range of supported **<pkt_timeout>**s**),(**range of supported **<retry_times>**s**),(**list of supported **<timeout_notice>**s**)** **+QMTCFG: "will",(**range of supported **<client_idx>**s**),(**list of supported **<will_fg>**s**),(**range of supported **<will_qos>**s**),(**list of supported **<will_retain>**s**),"willtopic","willmessage"** **+QMTCFG: "willex",(**range of supported **<client_idx>**s**),(**list of supported **<will_fg>**s**),(**range of supported **<will_qos>**s**),(**list of supported **<will_retain>**s**),"willtopic",(**range of supported **<will_len>**s**)** **+QMTCFG: "recv/mode",(**range of supported **<client_idx>**s**),(**list of supported **<msg_recv_mode>**s**),(**list of supported **<msg_len_enable>**s**)** **+QMTCFG: "aliauth",(**range of supported **<client_idx>**s**),"product key","device name","device secret"** **+QMTCFG: "qmtping",(**range of supported **<client_idx>**s**),(**r

	ange of supported **<qmtping_interval>**s) **+QMTCFG: "send/mode",(**range of supported **<client_idx>**s**),(**list of supported **<send_mode>s)** **+QMTCFG: "onenet",(**range of supported **<client_idx>**s**),"product id","access key"** **+QMTCFG: "hwauth",(**range of supported **<client_idx>s),"product id","device secret"** **+QMTCFG: "hwprodid",(**range of supported **<client_idx>s), "product id","product secret","nodeid"** **+QMTCFG: "dataformat",(**range of supported **<client_idx>s),(**list of supported **<send_mode>**s**),(**list of supported **<recv_mode>**s**)** **+QMTCFG: "view/mode",(**range of supported **<client_idx>**s**),(**list of supported **<view_mode>**s**)** **+QMTCFG: "edit/timeout",(**range of supported **<client_idx>**s**),(**list of supported **<edit_mode>**s**),(**range of supported **<edit_time>**s**)** **OK**
Write Command Query/set the MQTT protocol version **AT+QMTCFG="version",<client_idx>[,<vsn>]**	Response If the optional parameter is omitted, query the MQTT protocol version: **+QMTCFG: "version",<vsn>** **OK** If the optional parameter is specified and the MQTT connection is not established, set the MQTT protocol version: **OK** If there is any error: **ERROR**
Write Command Query/set the PDP to be used by the MQTT client **AT+QMTCFG="pdpcid",<client_idx>[,<cid>]**	Response If the optional parameter is omitted, query the PDP used by the MQTT client: **+QMTCFG: "pdpcid",<cid>** **OK** If the optional parameter is specified and the MQTT connection is not established, set the PDP to be used by the MQTT client: **OK** If there is any error:

	ERROR
Write Command Query/set the MQTT SSL mode and SSL context index **AT+QMTCFG="ssl",<client_idx>[,<SSL_enable>,<SSL_ctx_idx>]**	Response If the optional parameters are omitted, query the MQTT SSL mode and SSL context index: **+QMTCFG: "ssl",<SSL_enable>[,<SSL_ctx_idx>]** **OK** If the optional parameters are specified and the MQTT connection is not established, set the MQTT SSL mode and SSL context index: **OK** If there is any error: **ERROR**
Write Command Query/set the keep-alive time **AT+QMTCFG="keepalive",<client_idx>[,<keep_alive_time>]**	Response If the optional parameter is omitted, query the keep-alive time: **+QMTCFG: "keepalive",<keep_alive_time>** **OK** If the optional parameter is specified and the MQTT connection is not established, set the keep-alive time: **OK** If there is any error: **ERROR**
Write Command Query/set the session type **AT+QMTCFG="session",<client_idx>[,<clean_session>]**	Response If the optional parameter is omitted, query the session type: **+QMTCFG: "session",<clean_session>** **OK** If the optional parameter is specified and the MQTT connection is not established, set the session type: **OK** If there is any error: **ERROR**
Write Command Query/set timeout of message delivery **AT+QMTCFG="timeout",<client_id**	Response If the optional parameters are omitted, query the timeout of message delivery: **+QMTCFG: "timeout",<pkt_timeout>,<retry_times>,<timeo**

x>[,<pkt_timeout>,<retry_times>,<timeout_notice>]	**ut_notice>** **OK** If the optional parameters are specified and the MQTT connection is not established, set timeout of message delivery: **OK** If there is any error: **ERROR**
Write Command Query/set Will information **AT+QMTCFG="will",<client_idx>[,<will_fg>[,<will_qos>,<will_retain>,<will_topic>,<willmessage>]]**	Response If the optional parameters are omitted, query the current configuration: **+QMTCFG: "will",<will_fg>[,<will_qos>,<will_retain>,<will_topic>,<willmessage>]** **OK** If the optional parameters are specified and the MQTT connection is not established, set the Will information: **OK** If there is any error: **ERROR**
Write Command Query/set Will information **AT+QMTCFG="willex",<client_idx>[,<will_fg>,<will_qos>,<will_retain>,<willtopic>,<will_len>]**	Response If the optional parameters are omitted, query the current configuration: **+QMTCFG: "willex",<will_fg>[,<will_qos>,<will_retain>,<willtopic>,<will_len>]** If the optional parameters are specified, set the Will information: **>** Input the Will messages. When the actual size of data is greater than **<will_len>**, the first **<will_len>** byte(s) data is sent out. **OK** If there is any error: **ERROR**
Write Command Query/set receiving mode when data are received from server **AT+QMTCFG="recv/mode",<client_idx>[,<msg_recv_mode>[,<msg_len_enable>]]**	Response If the optional parameters are omitted, query the MQTT message receiving mode: **+QMTCFG: "recv/mode",<msg_recv_mode>,<msg_len_enable>**

	OK If the optional parameters are specified and the MQTT connection is not established, set the receiving mode when data are received from server: **OK** If there is any error: **ERROR**
Write Command Query/set Alibaba device information for Alibaba Cloud **AT+QMTCFG="aliauth",<client_idx>[,<product key>,<device name>,<device secret>]**	Response If the optional parameters are omitted, query the device information: **+QMTCFG: "aliauth",<product key>,<device name>,<device secret>** **OK** If the optional parameters are specified and the MQTT connection is not established, set Alibaba device information for Alibaba Could: **OK** If there is any error: **ERROR**
Write Command Query/set the MQTT heartbeat interval **AT+QMTCFG="qmtping",<client_idx>[,<qmtping_interval>]**	Response If the optional parameter is omitted, query the current configuration: **+QMTCFG: "qmtping",<qmtping_interval>** **OK** If the optional parameter is specified, and the MQTT connection is not established, set the MQTT heartbeat interval: **OK** If there is any error: **ERROR**
Write Command Query/set the MQTT message sending mode **AT+QMTCFG="send/mode",<client_idx>[,<send_mode>]**	Response If the optional parameter is omitted, query the current configuration: **+QMTCFG: "send/mode",<send_mode>** **OK**

	If the optional parameter is specified, and the MQTT connection is not established, configure the MQTT message sending mode: **OK** If there is any error: **ERROR**
Write Command Query/set the MQTT device for China Mobile OneNET IoT platform **AT+QMTCFG="onenet",<client_idx>[,<product id>,<access key>]**	Response If the optional parameters are omitted, query the current configuration: **+QMTCFG: "onenet",<product id>,<access key>** **OK** If the optional parameters are specified, and the MQTT connection is not established, set the MQTT device for OneNET IoT platform: **OK** If there is any error: **ERROR**
Write Command Query/set the MQTT device for Huawei IoT platform **AT+QMTCFG="hwauth",<client_idx>[,<product id>,<device secret>]**	Response If the optional parameters are omitted, query the current configuration: **+QMTCFG: "hwauth",<product id>,<device secret>[,<hw_time_enable>]** **OK** If the optional parameters are specified, and the MQTT connection is not established, set the MQTT device for Huawei IoT platform: **OK** If there is any error: **ERROR**
Write Command Query/set the MQTT device for Huawei IoT platform **AT+QMTCFG="hwprodid",<client_idx>[,<product id>,<product secret>,<nodeid>]**	Response If the optional parameters are omitted, query the current configuration: **+QMTCFG: "hwprodid",<product id>,<product secret>,<nodeid>[,<hw_time_enable>]** **OK** If the optional parameters are specified, and the MQTT

	connection is not established, set the MQTT device for Huawei IoT platform: **OK** If there is any error: **ERROR**
Write Command Query/set the MQTT data format **AT+QMTCFG="dataformat",<client_idx>[,<send_mode>,<recv_mode>]**	Response If the optional parameters are omitted, query the current configuration: **+QMTCFG: "dataformat",<send_mode>,<recv_mode>** **OK** If the optional parameters are specified, and the MQTT connection is not established, set the MQTT data format: **OK** If there is any error: **ERROR**
Write Command Query/set the MQTT data view mode in transparent mode **AT+QMTCFG="view/mode",<client_idx>[,<view_mode>]**	Response If the optional parameter is omitted, query the current configuration: **+QMTCFG: "view/mode",<view_mode>** **OK** If the optional parameter is specified, and the MQTT connection is not established, set the MQTT data view mode in transparent mode: **OK** If there is any error: **ERROR**
Write Command Query/set the MQTT input data timeout **AT+QMTCFG="edit/timeout",<client_idx>[,<edit_mode>,<edit_time>]**	Response If the optional parameters are omitted, query the current configuration: **+QMTCFG: "edit/timeout",<edit_mode>[,<edit_time>]** **OK** If the optional parameters are specified, and the MQTT connection is not established, set the MQTT input data timeout: **OK**

	If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristic	The command takes effect immediately. The configurations will not be saved.

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<vsn>	Integer type. MQTT protocol version. 3 MQTT protocol v3.1 4 MQTT protocol v3.1.1
<cid>	Integer type. PDP to be used by the MQTT client. Range: 1–15. Default value: 1.
<SSL_enable>	Integer type. Configure the MQTT SSL mode. 0 Use normal TCP connection for MQTT 1 Use SSL TCP secure connection for MQTT
<SSL_ctx_idx>	Integer type. SSL context index. Range: 0–5.
<keep_alive_time>	Integer type. Maximum interval between messages received from a client. If the server does not receive a message (interactive message and keep-alive packet) from the client within 1.5 times the keep-alive time, it disconnects the client as if the client has sent a DISCONNECT message. Range: 0–3600. Default value: 120. Unit: s. 0 Client is not disconnected
<clean_session>	Integer type. Configure session type. 0 The server must store the subscriptions of the client after it disconnects 1 The server must discard any previously maintained information about the client and treat the connection as "clean"
<pkt_timeout>	Integer type. Timeout of packet delivery. Range: 1–60. Default value: 5. Unit: s.
<retry_times>	Integer type. Retry times when packet delivery times out. Range: 0–10. Default value: 3.
<timeout_notice>	Integer type. Whether to report timeout message when transmitting packets. 0 Do not report 1 Report
<will_fg>	Integer type. Configure the Will flag. 0 Ignore the Will flag configuration 1 Require the Will flag configuration
<will_qos>	Integer type. Quality of service for message delivery. 0 At most once 1 At least once 2 Exactly once
<will_retain>	Integer type. Will retain flag is only used on **PUBLISH** messages. 0 When a client sends a **PUBLISH** message to a server, the server will not hold on to the message after it has been delivered to the current subscribers

	1 When a client sends a **PUBLISH** message to a server, the server should hold on to the message after it has been delivered to the current subscribers
<willtopic>	String type. Will topic string. Length: 1–256 bytes.
<willmessage>	String type. Will message defines the content of the message that is published to the will topic when the client is unexpectedly disconnected. Range: 0–256.
<will_len>	Integer type. Length of Will message. Range: 0–256. Unit: byte.
<msg_recv_mode>	Integer type. MQTT message receiving mode. 0 MQTT message received from server is contained in URC 1 MQTT message received from server is not contained in URC
<msg_len_enable>	Integer type. Whether the length of MQTT message received from server is contained in URC. 0 Not contained 1 Contained
<product key>	String type. Product key issued by Alibaba Cloud.
<device name>	String type. Device name issued by Alibaba Cloud.
<device secret>	String type. Device secret key issued by Alibaba Cloud/Huawei IoT platform.
<qmtping_interval>	Integer type. Heartbeat interval. Range: 5–60. Default value: 5. Unit: s.
<send_mode>	Integer type. Configure MQTT message sending mode. 0 String type 1 Hex type
<product id>	String type. Product ID issued by OneNET/Huawei IoT platform.
<access key>	String type. Product access key issued by OneNET/Huawei IoT platform.
<hw_time_enable>	Integer type. Enable/disable timestamp check when the device is connected to the Huawei IoT platform. 0 Disable 1 Enable
<product secret>	String type. Product verification certificate issued by Huawei IoT platform.
<nodeid>	String type. Device identification code, identifying the only device.
<recv_mode>	Integer type. MQTT message receiving mode. 0 String 1 Hex
<view_mode>	Integer type. MQTT data view mode in transparent mode. 0 Data are not displayed in transparent mode 1 Data can be displayed in transparent mode
<edit_mode>	Integer type. Configure whether to exit when the MQTT input data time out. 0 Disable 1 Enable
<edit_time>	Integer type. MQTT input data timeout. Range: 1–120. Unit: s.

NOTE

1. If **<will_fg>**=1, then **<will_qos>**, **<will_retain>**, **<willtopic>** and **<willmessage>** must be set. Otherwise, they are omitted.
2. **<clean_session>**=0 is only effective when the server supports the operation.

3. If MQTT connection is configured to SSL mode, **<SSL_ctx_idx>** must be specified. In addition, you can use **AT+QSSLCFG** to configure the SSL version, cipher suite, secure level, CA certificate, client certificate, client key and ignorance of RTC time, which are used in MQTT SSL handshake procedure.
4. It is crucial to ensure that message delivery does not time out while the message sending is ongoing.
5. **AT+QMTCFG="aliauth"** is only used for Alibaba Cloud. If it is configured, **<username>** and **<password>** in **AT+QMTCONN** can be omitted.

3.3.2. AT+QMTOPEN Open a Network for MQTT Client

This command opens a network for MQTT client.

AT+QMTOPEN Open a Network for MQTT Client	
Test Command **AT+QMTOPEN=?**	Response **+QMTOPEN: (**range of supported **<client_idx>**s**),"hostname",(**range of supported **<port>**s**)** **OK**
Read Command **AT+QMTOPEN?**	Response **[+QMTOPEN: <client_idx>,<host_name>,<port>]** **OK** If there is any error: **ERROR**
Write Command **AT+QMTOPEN=<client_idx>,<host_name>,<port>**	Response **OK** **+QMTOPEN: <client_idx>,<result>** If there is any error: **ERROR**
Maximum Response Time	120 s, determined by network
Characteristic	/

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<host_name>	String type. Server address. It could be an IP address or a domain name. Maximum size: 100 bytes.
<port>	Integer type. Server port. Range: 1–65535.
<result>	Integer type. Command execution result.

-1	Failed to open network
0	Network opened successfully
1	Wrong parameter
2	MQTT identifier is occupied
3	Failed to activate PDP
4	Failed to parse domain name
5	Network connection error

3.3.3. AT+QMTCLOSE Close a Network for MQTT Client

This command closes a network for MQTT client.

AT+QMTCLOSE Close a Network for MQTT Client	
Test Command **AT+QMTCLOSE=?**	Response **+QMTCLOSE: (**range of supported **<client_idx>**s**)** **OK**
Write Command **AT+QMTCLOSE=<client_idx>**	Response **OK** **+QMTCLOSE: <client_idx>,<result>** If there is any error: **ERROR**
Maximum Response Time	30 s
Characteristic	/

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<result>	Integer type. Command execution result. -1 Failed to close network 0 Network closed successfully

3.3.4. AT+QMTCONN Connect a Client to MQTT Server

This command connects a client to MQTT server. When a TCP/IP socket connection is established between a client to a server, a protocol level session must be created using a CONNECT flow.

AT+QMTCONN Connect a Client to MQTT Server	
Test Command **AT+QMTCONN=?**	Response **+QMTCONN: (**range of supported **<client_idx>**s**),"clienti**

	d","username","password" **OK**
Read Command **AT+QMTCONN?**	Response **[+QMTCONN: <client_idx>,<state>]** **OK** If there is any error: **ERROR**
Write Command **AT+QMTCONN=<client_idx>,<clientid>[,<username>,<password>]**	Response **OK** **+QMTCONN: <client_idx>,<result>[,<ret_code>]** If there is any error: **ERROR**
Maximum Response Time	**<pkt_timeout>** (default 5 s), determined by network
Characteristic	/

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<clientid>	String type. Client identifier string.
<username>	String type. Client username. It can be used for authentication.
<password>	String type. Password corresponding to the client username. It can be used for authentication.
<result>	Integer type. Command execution result. 0 Packet sent successfully and ACK received from server 1 Packet retransmission 2 Failed to send packet
<state>	Integer type. MQTT connection state. 1 MQTT is initializing 2 MQTT is connecting 3 MQTT is connected 4 MQTT is disconnecting
<ret_code>	Integer type. Connection status return code. 0 Connection Accepted 1 Connection Refused: Unacceptable Protocol Version 2 Connection Refused: Identifier Rejected 3 Connection Refused: Server Unavailable 4 Connection Refused: Bad Username or Password

	5 Connection Refused: Not Authorized
<pkt_timeout>	Integer type. Packet delivery timeout. Range: 1–60. Default value: 5. Unit: second. The value can be configured with **AT+QMTCFG="timeout",<client_idx>[,<pkt_timeout>,<retry_times>,<timeout_notice>]**.

NOTE

If a client with the same client ID is already connected to the server, the "older" client must be disconnected by the server before completing the CONNECT flow of the new client.

3.3.5. AT+QMTDISC Disconnect a Client from MQTT Server

This command requests a disconnection from MQTT server by a client. The client sends a **DISCONNECT** message to the server to indicate that it is about to close its TCP/IP connection.

AT+QMTDISC Disconnect a Client from MQTT Server	
Test Command **AT+QMTDISC=?**	Response **+QMTDISC: (**range of supported **<client_idx>**s**)** **OK**
Write Command **AT+QMTDISC=<client_idx>**	Response **OK** **+QMTDISC: <client_idx>,<result>** If there is any error: **ERROR**
Maximum Response Time	30 s
Characteristic	/

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<result>	Integer type. Command execution result.
	-1 Failed to close connection
	0 Connection closed successfully

3.3.6. AT+QMTSUB Subscribe to Topics

This command subscribes to one or more topics. A **SUBSCRIBE** message is sent to the server by a client to register an interest in one or more topics. Messages published to these topics are delivered from the server to the client as **PUBLISH** messages.

AT+QMTSUB Subscribe to Topics	
Test Command **AT+QMTSUB=?**	Response **+QMTSUB: (**range of supported **<client_idx>**s**),<msgid>**,**list of ["topic",qos]** **OK**
Write Command **AT+QMTSUB=<client_idx>,<msgid>,<topic1>,<qos1>[,<topic2>,<qos2>…]**	Response **OK** **+QMTSUB: <client_idx>,<msgid>,<result>[,<value>]** If there is any error: **ERROR**
Maximum Response Time	**<pkt_timeout>** × **<retry_times>** (default 15 s), determined by network
Characteristic	/

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<msgid>	Integer type. Message identifier of packet. Range: 1–65535.
<topic>	String type. Topic that the client wants to subscribe to or unsubscribe from.
<qos>	Integer type. QoS level at which the client wants to publish messages. 0 At most once 1 At least once 2 Exactly once
<result>	Integer type. Command execution result. 0 Packet sent successfully and ACK received from server 1 Packet retransmission 2 Failed to send packet
<value>	Integer type. If **<result>** is 0, it is a vector of granted QoS levels. If **<result>** is 1, it is the number of times the packet has been retransmitted. If **<result>** is 2, it is not presented.
<pkt_timeout>	Integer type. Packet delivery timeout. Range: 1–60. Default value: 5. Unit: s. The value can be configured with **AT+QMTCFG="timeout",<client_idx>[,<pkt_timeout>,<retry_times>,<timeout_notice>]**.
<retry_times>	Integer type. Retry times when packet delivery times out. Range: 0–10. Default value: 3.

NOTE

The **<msgid>** is only present in messages where the QoS bits in the fixed header indicate QoS levels 1 or 2. It must be unique amongst the set of "inflight" messages in a particular communication direction. It typically increases by 1 from one message to the next, but it is not compulsory in practical applications.

3.3.7. AT+QMTUNS Unsubscribe from Topics

This command unsubscribes from one or more topics. An **UNSUBSCRIBE** message is sent by the client to the server to unsubscribe from named topics.

AT+QMTUNS Unsubscribe from Topics	
Test Command **AT+QMTUNS=?**	Response **+QMTUNS: (**range of supported **<client_idx>**s**),<msgid>,list of ["topic"]** **OK**
Write Command **AT+QMTUNS=<client_idx>,<msgid>,<topic1>[,<topic2>…]**	Response **OK** **+QMTUNS: <client_idx>,<msgid>,<result>[,<value>]** If there is any error: **ERROR**
Maximum Response Time	**<pkt_timeout>** × **<retry_times>** (default 15 s), determined by network
Characteristic	/

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<msgid>	Integer type. Message identifier of packet. Range: 1–65535.
<topic>	String type. Topic that the client wants to subscribe to or unsubscribe from.
<result>	Integer type. Command execution result. 0 Packet sent successfully and ACK received from server 1 Packet retransmission 2 Failed to send packet
<value>	Integer type. If **<result>** is 0, it is a vector of granted QoS levels. If **<result>** is 1, it the number of times the packet has been retransmitted. If **<result>** is 2, it is not presented.
<pkt_timeout>	Integer type. Packet delivery timeout. Range: 1–60. Default value: 5. Unit: s. The value can be configured with **AT+QMTCFG="timeout",<client_idx>[,<pkt**

	_timeout>,<retry_times>,<timeout_notice>].
<retry_times>	Integer type. Retry times when packet delivery times out. Range: 0–10. Default value: 3.

3.3.8. AT+QMTPUBEX Publish Messages

This command publishes fixed-length messages by a client to a server for distribution to interested subscribers. Each **PUBLISH** message is associated with a topic name. If a client subscribes to one or more topics, any message published on those topics is sent by the server to the client as a **PUBLISH** message.

AT+QMTPUBEX Publish Messages	
Test Command **AT+QMTPUBEX=?**	Response **+QMTPUBEX: (**range of supported **<client_idx>**s**),<msgid>,(**range of supported **<qos>**s**),(**list of supported **<retain>**s**),"topic","length"** **OK**
Write Command **AT+QMTPUBEX=<client_idx>,<msgid>,<qos>,<retain>,<topic>,<length>**	Response **>** After **>** is reported, input the data to be sent. When the actual size of data is greater than **<length>**, the first **<length>** byte(s) data is sent out. **OK** **+QMTPUBEX: <client_idx>,<msgid>,<result>[,<value>]** If there is any error: **ERROR**
Maximum Response Time	**<pkt_timeout>** × **<retry_times>** (default 15 s), determined by network
Characteristic	/

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<msgid>	Integer type. Message identifier of packet. Range: 0–65535. It is 0 only when **<qos>**=0.
<qos>	Integer type. QoS level at which the client wants to publish messages. 0 At most once 1 At least once 2 Exactly once
<retain>	Integer type. Whether or not the server will retain the message after it has been

	delivered to the current subscribers. 0 Do not retain 1 Retain
<topic>	String type. Topic that needs to be published.
<length>	Integer type. Length of message to be published.
<result>	Integer type. Command execution result. 0 Packet sent successfully and ACK received from server (message that is published when **<qos>**=0 does not require ACK) 1 Packet retransmission 2 Failed to send packet
<value>	Integer type. If **<result>** is 1, it means the number of times a packet has been retransmitted . If **<result>** is 0 or 2, it is not presented.
<pkt_timeout>	Integer type. Packet delivery timeout. Range: 1–60. Default value: 5. Unit: s. **AT+QMTCFG="timeout",<client_idx>[,<pkt_timeout>,<retry_times>,<timeout_notice>]**.
<retry_times>	Integer type, Retry times when packet delivery times out. Range: 0–10. Default value: 3.

NOTE

1. If this command is executed successfully and **OK** is returned, the client can publish a new packet. The maximum quantity of packets to be transmitted should not be greater than the inflight window size (5); otherwise, **ERROR** is returned.
2. After executing this command, the client is ready to send data, which are sent as payload. Maximum length of data input at a time: 1500 bytes.
3. **PUBLISH** messages can be sent either by a publisher to the server, or by the server to a subscriber. When a server publishes messages to a subscriber, the following URC is returned to notify the host to read the received data reported by MQTT server:
   +QMTRECV: <client_idx>,<msgid>,<topic>[,<payload_length>],<payload>.
   For more information about the URC, see ***Chapter 4.2***.

3.3.9. AT+QMTRECV Read Messages from Buffer

This command reads the messages from the storage buffer where the messages are stored after being reported by the server.

AT+QMTRECV Read Messages from Buffer	
Test Command **AT+QMTRECV=?**	Response **OK**
Read Command **AT+QMTRECV?**	Response **+QMTRECV: <client_idx>,<store_status_0>,<store_status_1>,<store_status_2>,<store_status_3>,<store_status_4>** **OK** If there is no MQTT connection: **OK**
Write Command **AT+QMTRECV=<client_idx>[,<recv_id>]**	Response **[**List of (**+QMTRECV: <client_idx>,<msgid>,<topic>[,<payload_len>],<payload>)**s**]** **OK** If there is no MQTT connection: **ERROR**
Maximum Response Time	/
Characteristic	/

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<store_status>	Integer type. Indicate if a message is stored in the buffer. Maximum 5 messages can be stored in the buffer. Therefore, URC reports maximally 5 messages simultaneously. 0 No message in the buffer 1 There are stored messages in the buffer
<recv_id>	Integer type. Serial number of a received message. Range: 0–4.
<msgid>	Integer type. Message identifier of packet. Range: 0–65535. It is equal to 0 only when **<qos>**=0.
<topic>	String type. Topic that needs to be published.
<payload_len>	Integer type. Payload length. Unit: Byte.
<payload>	String type. Payload relating to the topic name.

4 MQTT URCs

This chapter describes MQTT URCs.

Table 3: MQTT URCs

SN	URC Format	Description
[1]	**+QMTSTAT: <client_idx>,<err_code>**	Reported when the state of MQTT link layer has changed, the client will close the MQTT connection and report the URC.
[2]	**+QMTRECV: <client_idx>,<msgid>,<topic>[,<payload_len>],<payload>**	Reported when the client has received the packet data from MQTT server.
[3]	**+QMTRECV: <client_idx>,<recv_id>**	Reported when the message received from MQTT server has been stored in buffer.
[4]	**+QMTPING: <client_idx>,<result>**	Reported when the state of MQTT link layer has changed, the client will close the MQTT connection and report the URC.

4.1. +QMTSTAT URC to Indicate State Change in MQTT Link Layer

The URC beginning with **+QMTSTAT:** is reported when there is a change in the state of MQTT link layer.

+QMTSTAT URC to Indicate State Change in MQTT Link Layer	
+QMTSTAT: <client_idx>,<err_code>	When the state of MQTT link layer has changed, the client will close the MQTT connection and report the URC.

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<err_code>	Integer type. Error code. Refer to the table below for more information.

Table 4: Error Codes of +QMTSTAT: URC

<err_code>	Description	Solution
1	Connection is closed or reset by a peer end.	Execute **AT+QMTOPEN** and reopen MQTT connection.
2	Sending PINGREQ packet timed out or failed.	First, deactivate and activate the PDP. Then, reopen the MQTT connection.
3	Sending CONNECT packet timed out or failed.	1. Check whether the inputted username and password are correct. 2. Make sure the client ID is not used. 3. Reopen MQTT connection and try to send CONNECT packet to server again.
4	Receiving CONNACK packet timed out or failed.	1. Check whether the inputted username and password are correct. 2. Make sure the client ID is not used. 3. Reopen MQTT connection and try to send CONNECT packet to server again.
5	The client sends DISCONNECT packet to sever and the server starts closing the MQTT connection.	This is a normal process.
6	The client closes MQTT connection due to packet sending failure all the time.	1. Make sure the data are correct. 2. Try to reopen MQTT connection since there may be a network congestion or an error.
7	The link is not alive or the server is unavailable.	Make sure the link is alive or the server is currently available.
8	The client closes the MQTT connection.	Try to reconnect.
9–255	Reserved for future use.	

4.2. +QMTRECV URC to Inform the Host to Read MQTT Packet Data

The URC beginning with **+QMTRECV:**. is mainly used to notify the host to read the received MQTT packet data reported by MQTT server.

+QMTRECV URC to Inform the Host to Read MQTT Packet Data	
+QMTRECV: <client_idx>,<msgid>,<topic>[,<payload_len>],<payload>	Notify the host to read the received data reported by MQTT server.
+QMTRECV: <client_idx>,<recv_id>[,<payload_len>]	Reported when the message received from MQTT server has been stored in buffer.

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<msgid>	Integer type. Message identifier of packet.
<topic>	String type. Topic received from MQTT server.
<payload_len>	Integer type. Payload length.
<payload>	String type. Payload relating to the topic name.
<recv_id>	Integer type. Serial number of a received message. Range: 0–4.

4.3. +QMTPING URC to Indicate PING State of Keep-alive in MQTT

The URC beginning with **+QMTPING:**. is reported when server receives no message from the client within 1.5 times the keep-alive interval and it will disconnect the client as if the client has sent a **DISCONNECT** message.

+ QMTPING URC to Indicate PING State of Keep-alive in MQTT	
+QMTPING: <client_idx>,<result>	Received when the state of MQTT link layer has changed. The client will close the MQTT connection and report the URC.

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<result>	Integer type. Result of PING state.
.	1 Failed

5 Examples

This chapter provides some examples to explain how to use MQTT AT commands.

5.1. Example of MQTT Operation Without SSL

```
//Configure receiving mode.
AT+QMTCFG="recv/mode",0,0,1
OK
//Configure Alibaba device information for Alibaba cloud.
AT+QMTCFG="aliauth",0,"oyjtmPl5a5j","MQTT_TEST","wN9Y6pZSIly7Exa5qVzcmigEGO4kAazZ"
OK
AT+QMTOPEN=?
+QMTOPEN: (0-5),"hostname",(1-65535)

OK

//Open a network for MQTT client.
AT+QMTOPEN=0,"iot-as-mqtt.cn-shanghai.aliyuncs.com",1883
OK

+QMTOPEN: 0,0                    //Opened the MQTT client network successfully.
AT+QMTOPEN?
+QMTOPEN: 0,"iot-as-mqtt.cn-shanghai.aliyuncs.com",1883

OK
AT+QMTCONN=?
+QMTCONN: (0-5),"clientid","username","password"

OK

//Connect a client to MQTT server.
//If Alibaba Cloud is connected, AT+QMTCFG="aliauth" can be used to configure the device information in advance. There is no need to provide username/password from now on.
AT+QMTCONN=0,"clientExample"
OK
```

```
+QMTCONN: 0,0,0                //Connected the client to MQTT server successfully.
AT+QMTSUB=?
+QMTSUB: (0-5),<msgid>,list of ["topic",qos]

OK

//Subscribe to topics.
AT+QMTSUB=0,1,"topic/example",2
OK

+QMTSUB: 0,1,0,2
AT+QMTSUB=0,1,"topic/pub",0
OK

+QMTSUB: 0,1,0,0

//If a client subscribes to a topic and other devices publish the same topic to the server, the module will report the following information.
+QMTRECV: 0,0,"topic/example",36,"This is the payload related to topic"

//Unsubscribe from topics.
AT+QMTUNS=0,2,"topic/example"
OK

+QMTUNS: 0,2,0
AT+QMTPUBEX=?
+QMTPUBEX: (0-5),<msgid>,(0-2),(0,1),"topic","length"
OK

//Publish messages. After receiving >, input data "These are test data, hello MQTT." and then send it. The maximum length of the data to be sent is 1500 bytes. All data beyond 1500 bytes will be omitted.
AT+QMTPUBEX=0,0,0,0,"topic/pub",30
> These are test data, hello MQTT.
OK

+QMTPUBEX: 0,0,0

//If a client subscribes to a topic named "topic/pub" and other devices publish the same topic to the server, the module reports the following information.
+QMTRECV: 0,0,"topic/pub",30,"These are test data, hello MQTT."

//Disconnect a client from MQTT server.
AT+QMTDISC=0
OK
```

```
+QMTDISC: 0,0                    //Connection closed successfully.
```

5.2. Example of MQTT Operation with SSL

See ***document [1]*** for more information on AT commands related to SSL.

```
//Configure receiving mode.
AT+QMTCFG="recv/mode",0,0,1
OK

//Configure MQTT session into SSL mode.
AT+QMTCFG="SSL",0,1,2
OK

//If SSL authentication mode is "server authentication", store CA certificate to UFS.
AT+QFUPL="UFS:cacert.pem",1758,100
CONNECT
<Input the cacert.pem data, the size is 1758 bytes>
+QFUPL: 1758,384a

OK

//If SSL authentication mode is "server authentication", store CC certificate to UFS.
AT+QFUPL="UFS:client.pem",1220,100
CONNECT
<Input the client.pem data, the size is 1220 bytes>
+QFUPL: 1220,2d53

OK

//If SSL authentication mode is "server authentication", store CK certificate to UFS.
AT+QFUPL="UFS:user_key.pem",1679,100
CONNECT
<Input the user_key.pem data, the size is 1679 bytes>
+QFUPL: 1679,335f

OK

//Configure CA certificate.
AT+QSSLCFG="cacert",2,"UFS:cacert.pem"
OK
```

```
//Configure CC certificate.
AT+QSSLCFG="clientcert",2,"UFS:client.pem"
OK
//Configure CK certificate.
AT+QSSLCFG="clientkey",2,"UFS:user_key.pem"
OK

//Configure the authentication mode for SSL context 2.
AT+QSSLCFG="seclevel",2,2                  //SSL authentication mode: server and client authentication
                                           if requested by the remote server.
OK
AT+QSSLCFG="sslversion",2,4                //SSL authentication version.
OK
AT+QSSLCFG="ciphersuite",2,"0xFFFF"        //Cipher suite.
OK
AT+QSSLCFG="ignorelocaltime",2,1           //Ignore authentication time.
OK

//Start MQTT SSL connection.
AT+QMTOPEN=0,"a1zgnxur10j8ux.iot.us-east-1.amazonaws.com",8883
OK

+QMTOPEN: 0,0

//Connect to MQTT server.
AT+QMTCONN=0,"M26_0206"
OK

+QMTCONN: 0,0,0

//Subscribe to topics.
AT+QMTSUB=0,1,"$aws/things/M26_0206/shadow/update/accepted",1
OK

+QMTSUB: 0,1,0,1

//Publish messages.
AT+QMTPUBEX=0,1,1,0,"$aws/things/M26_0206/shadow/update/accepted",32
>These are published data from client
OK

+QMTPUBEX: 0,1,0

//If a client subscribes to a topic named "$aws/things/M26_0206/shadow/update/accepted" and other
```

devices publish the same topic to the server, the module will report the following information.

+QMTRECV: 0,1,"$aws/things/M26_0206/shadow/update/accepted",32,"These are published data from client"

//Disconnect a client from MQTT server.

AT+QMTDISC=0

OK

+QMTDISC: 0,0

6 Appendix References

Table 5: Related Document

Document Name
[1] Quectel_EC200x-CN&EC800x-CN_SSL_Application_Note

Table 6: Terms and Abbreviations

Abbreviation	Description
ACK	Acknowledgement
CA	Certificate Authority
IP	Internet Protocol
MQTT	Message Queuing Telemetry Transport
PDP	Packet Data Protocol
QoS	Quality of Service
RTC	Real-Time Clock
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
UFS	User File System
URC	Unsolicited Result Code