

EC2x&EG2x&EG9x&EM05 Series

MQTT Application Note

LTE Standard Module Series

Version: 1.3

Date: 2023-11-16

Status: Released



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.

Copyright © Quectel Wireless Solutions Co., Ltd. 2023. All rights reserved.

About the Document

Revision History

Version	Date	Author	Description
1.0	2018-08-21	Chavis CHEN	Creation of the document
1.1	2018-12-12	Slark WANG	Added Write Command AT+QMTCFG="qmtping" and its related information.
1.2	2020-04-22	Chavis CHEN/ Domingo DENG/ Larson LI	- Added applicable modules in Chapter 1. - Updated MQTT data interaction diagram in Chapter 2. - Updated description of Test Command AT+QMTCFG=? and added Write Command AT+QMTCFG="willex" in Chapter 3.2.1. - Added <pkt_timeout> configuration in the parameter table in Chapter 3.2.4, 3.2.6, 3.2.7 and 3.2.8. - Updated error codes of the URC in Table 4.
1.3	2023-11-16	Mario CHEN/ Chavis CHEN/ Domingo DENG/ Larson LI	- Added the applicable modules: EG21-GL and EG25-GL and updated EC20 R2.1 to EC20-CE. (Chapter 1). - Updated the response to AT+QMTCFG=? And the parameter name of AT+QMTCFG="aliauth",<client_idx>[,<product_key>,<device_name>,<ali_device_secret>] (Chapter 3.2.1). - Added the following sub-commands in AT+QMTCFG (Chapter 3.2.1): - AT+QMTCFG="send/mode",<client_idx>[,<msg_send_mode>] - AT+QMTCFG="hwauth",<client_idx>[,<device_id>,<hw_device_secret>] - AT+QMTCFG="hwprodid",<client_idx>[,<product_id>,<product_secret>,<nodeid>]

-
- AT+QMTCFG="protocol/check",<client_idx>[,<protocol_check_enable>]
4. Added the characteristics of the following commands (Chapter 3.2):
 - AT+QMTOPEN
 - AT+QMTCLOSE
 - AT+QMTCONN
 - AT+QMTDISC
 - AT+QMTSUB
 - AT+QMTUNS
 - AT+QMTPUBEX
 - AT+QMTRECV
 5. Updated the values of <result> in AT+QMTDISC and added a note (Chapter 3.2.5).
 6. Added AT+QMTPUB (Chapter 3.2.9).
 7. Added the maximum response time of AT+QMTRECV (Chapter 3.2.10).
 8. Updated the examples of AT+QMTDISC (Chapter 5).
-

Contents

About the Document.....	3
Contents	5
Table Index.....	6
1 Introduction	7
1.1. Applicable Modules	7
2 MQTT Data Interaction.....	8
3 MQTT-Related AT Commands	9
3.1. AT Command Syntax	9
3.1.1. Definitions.....	9
3.1.2. AT Command Syntax	9
3.2. Declaration of AT Command Examples	10
3.3. Description of MQTT-Related AT Commands.....	10
3.3.1. AT+QMTCFG Configure Optional Parameters of MQTT	10
3.3.2. AT+QMTOPEN Open a Network for MQTT Client.....	17
3.3.3. AT+QMTCLOSE Close a Network for MQTT Client	18
3.3.4. AT+QMTCONN Connect a Client to MQTT Server.....	19
3.3.5. AT+QMTDISC Disconnect a Client from MQTT Server	20
3.3.6. AT+QMTSUB Subscribe to Topics	21
3.3.7. AT+QMTUNS Unsubscribe from Topics.....	23
3.3.8. AT+QMTPUBEX Publish Messages (Data Mode)	24
3.3.9. AT+QMTPUB Publish Messages (Command Mode).....	25
3.3.10. AT+QMTRECV Read Messages from Buffer	27
4 MQTT-Related URCs.....	29
4.1. +QMTSTAT Indicate State Change in MQTT Link Layer	29
4.2. +QMTRECV Notify Host to Read MQTT Packet Data.....	30
4.3. +QMTPING Indicate PING State of Keep-alive in MQTT	31
5 Examples	32
5.1. Example of MQTT Operation without SSL	32
5.2. Example of MQTT Operation with SSL.....	34
6 Appendix References	37

Table Index

Table 1: Applicable Modules.....	7
Table 2: Type of AT Commands and Responses.....	9
Table 3: MQTT-Related URCs	29
Table 4: URC Error Codes	30
Table 5: Related Documents	37
Table 6: Terms and Abbreviations	37

1 Introduction

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 Quectel EC2x family, EG2x family, EG9x family and EM05 series modules through AT commands.

1.1. Applicable Modules

Table 1: Applicable Modules

Module Family	Module
EC2x	EC20-CE
	EC21 Series
	EC25 Series
EG2x	EG21-G
	EG25-G
	EG21-GL
	EG25-GL
EG9x	EG91 Series
	EG95 Series
-	EM05 Series

2 MQTT Data Interaction

This chapter illustrates the data interaction mechanism of MQTT function.

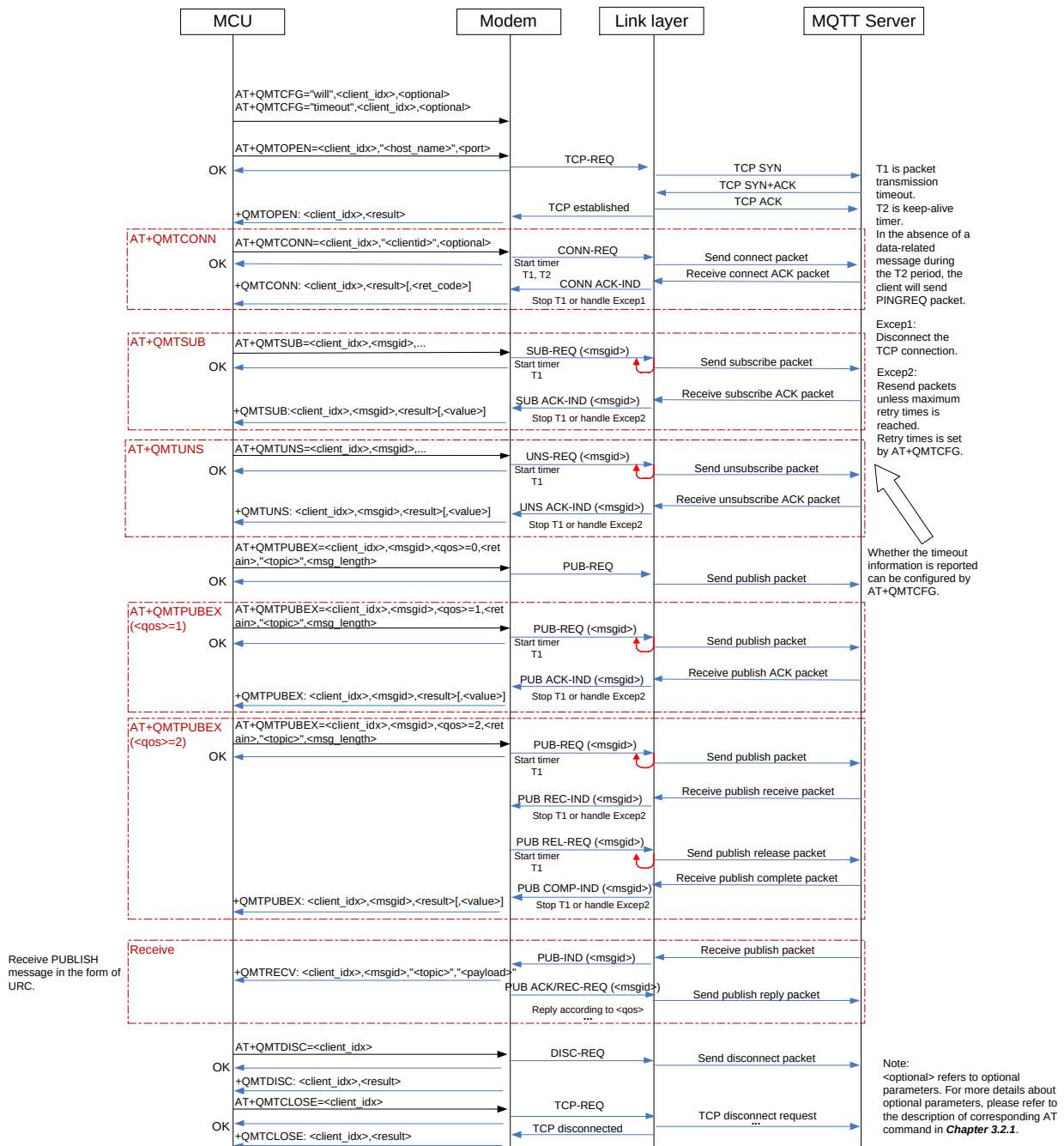


Figure 1: MQTT Data Interaction Diagram

3 MQTT-Related AT Commands

This chapter presents the AT commands for using the MQTT function.

3.1. AT Command Syntax

3.1.1. Definitions

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

3.1.2. AT Command Syntax

The **AT** or **at** prefix must be added at the beginning of each command line. Entering **<CR>** will terminate a command line. Commands are usually followed by a response that includes **<CR><LF><response><CR><LF>**. Throughout this document, only the response **<response>** will be presented, **<CR><LF>** are omitted intentionally.

Table 2: Type of AT Commands and Responses

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 learn about the use of the AT commands introduced herein. The examples, however, should not be taken as Quectel's recommendations or suggestions about how to design a program flow or what status to set the module into. Sometimes multiple examples may be provided for one AT command. However, this does not mean that there is a correlation among these examples, or 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 <vsns>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: "send/mode",(range of supported <client_idx>s)

	s),(list of supported <msg_send_mode>s) +QMTCFG: "aliauth",(range of supported <client_idx>s),"product key","device name","device secret" +QMTCFG: "hwauth",(range of supported <client_idx>s),"device id","device secret" +QMTCFG: "hwprodid",(range of supported <client_idx>s),"product id","product secret","nodeid" +QMTCFG: "qmtping",(range of supported <client_idx>s),(range of supported <qmtping_interval>s) +QMTCFG: "protocol/check",(range of supported <client_idx>s),(list of supported <protocol_check_enable>s) OK
Write Command Query/set the MQTT protocol version AT+QMTCFG="version",<client_idx>[,<vsn>]	Response If <vsn> is omitted, query the MQTT protocol version: +QMTCFG: "version",<vsn> OK If <vsn> is specified and the MQTT connection is not established, set the MQTT protocol version: OK Or ERROR
Write Command Query/set the PDP to be used by the MQTT client AT+QMTCFG="pdpcid",<client_idx>[,<cid>]	Response If <cid> is omitted, query the PDP to be used by the MQTT client: +QMTCFG: "pdpcid",<cid> OK If <cid> is specified and the MQTT connection is not established, set the PDP to be used by the MQTT client: OK Or ERROR
Write Command Query/set Will information AT+QMTCFG="will",<client_idx>[,<will_fg>[,<will_qos>,<will_retain>,<will_topic>,<will_msg>]]	Response: If <will_fg>, <will_qos>, <will_retain>, <will_topic> and <will_msg> are omitted, query the Will information: +QMTCFG: "will",<will_fg>[,<will_qos>,<will_retain>,<will_topic>,<will_msg>] OK

	If <will_fg>, <will_qos>, <will_retain>, <will_topic> and <will_msg> are specified and the MQTT connection is not established, set the Will information: OK Or ERROR
Write Command Query/set Will information AT+QMTCFG="willex",<client_idx>[,<will_fg>[,<will_qos>,<will_retain>,<will_topic>,<will_len>]]	Response If <will_fg>, <will_qos>, <will_retain>, <will_topic>, <will_len> are omitted, query the current configuration: +QMTCFG: "willex",<will_fg>[,<will_qos>,<will_retain>,<will_topic>,<will_len>] OK If <will_fg>, <will_qos>, <will_retain>, <will_topic>, <will_len> are specified, set the Will information: > Input the Will message. When the actual size of data is greater than <will_len>, the first <will_len> byte(s) data will be sent out. OK Or ERROR
Write Command Query/set message delivery timeout AT+QMTCFG="timeout",<client_idx>[,<pkt_timeout>,<retry_times>,<timeout_notice>]	Response If <pkt_timeout>, <retry_times> and <timeout_notice> are omitted, query message delivery timeout: +QMTCFG: "timeout",<pkt_timeout>,<retry_times>,<timeout_notice> OK If <pkt_timeout>, <retry_times> and <timeout_notice> are specified and the MQTT connection is not established, set message delivery timeout: OK Or ERROR
Write Command Query/set session type AT+QMTCFG="session",<client_idx>[,<clean_session>]	Response If <clean_session> is omitted, query session type: +QMTCFG: "session",<clean_session> OK If <clean_session> is specified and the MQTT connection is not established, set the session type:

	OK Or ERROR
Write Command Query/set keep-alive time AT+QMTCFG="keepalive",<client_idx>[,<keep_alive_time>]	Response If <keep_alive_time> is omitted, query the keep-alive time: +QMTCFG: "keepalive",<keep_alive_time> OK If <keep_alive_time> is specified and the MQTT connection is not established, set the keep-alive time: OK Or 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 <SSL_enable> and <SSL_ctx_idx> are omitted, query the MQTT SSL mode and SSL context index: +QMTCFG: "ssl",<SSL_enable>[,<SSL_ctx_idx>] OK If <SSL_enable> and <SSL_ctx_idx> are specified and the MQTT connection is not established, set the MQTT SSL mode and SSL context index: OK Or ERROR
Write Command Query/set receiving mode when data is received from server AT+QMTCFG="recv/mode",<client_idx>[,<msg_rcv_mode>[,<msg_len_enable>]]	Response If <msg_rcv_mode> and <msg_len_enable> are omitted, query the MQTT message receiving mode: +QMTCFG: "recv/mode",<msg_rcv_mode>,<msg_len_enable> OK If <msg_rcv_mode> and <msg_len_enable> are specified and the MQTT connection is not established, set the receiving mode when data is received from server: OK Or ERROR
Write Command Query/set client data sending mode AT+QMTCFG="send/mode",<client_idx>[,<msg_send_mode>]	Response If <msg_send_mode> is omitted, query the MQTT data sending mode:

t_idx>[,<msg_send_mode>]	+QMTCFG: "send/mode",<msg_send_mode> OK If <msg_send_mode> is specified and the MQTT connection is not established, set the client data sending mode: OK Or ERROR
Write Command Query/set Alibaba device information for Alibaba Cloud AT+QMTCFG="aliauth",<client_id x>[,<product_key>,<device_name>,<ali_device_secret>]	Response If <product_key>, <device_name> and <ali_device_secret> are omitted, query the device information: +QMTCFG: "aliauth",<product_key>,<device_name>,<ali_device_secret> OK If <product_key>, <device_name> and <ali_device_secret> are specified and the MQTT connection is not established, set Alibaba device information for Alibaba Cloud: OK Or ERROR
Write Command Query/set device information for HUAWEI Cloud AT+QMTCFG="hwauth",<client_id x>[,<device_id>,<hw_device_secret>]	Response If <device_id> and <hw_device_secret> are omitted, query the device information: +QMTCFG: "hwauth",<device_id>,<hw_device_secret> OK If <device_id> and <hw_device_secret> are specified and the MQTT connection is not established, set device information for HUAWEI Cloud: OK Or ERROR
Write Command Query/set product information for HUAWEI Cloud AT+QMTCFG="hwprodid",<client_id x>[,<product_id>,<product_secret>,<nodeid>]	Response If <product_id>, <product_secret> and <nodeid> are omitted, query the product information: +QMTCFG: "hwprodid",<product_id>,<product_secret>,<nodeid> OK

	If <product_id>, <product_secret> and <nodeid> are specified and the MQTT connection is not established, set product information for HUAWEI Could: OK Or ERROR
Write Command Query/set the maximum retention time of MQTT connection before disconnection due to heartbeat timeout AT+QMTCFG="qmtping",<client_idx>[,<qmtping_interval>]	Response If <qmtping_interval> is omitted, query the current configuration: +QMTCFG: "qmtping",<qmtping_interval> OK If <qmtping_interval> is specified, and the MQTT connection is not established, set the maximum retention time of MQTT connection before disconnection due to heartbeat timeout: OK Or ERROR
Write Command Query/set MQTT protocol check AT+QMTCFG="protocol/check",<client_idx>[,<protocol_check_enable>]	Response If <protocol_check_enable> is omitted, query the current configuration: +QMTCFG: "protocol/check",<client_idx>,<protocol_check_enable> OK If <protocol_check_enable> is specified, and the MQTT connection is not established, set MQTT protocol check: OK Or ERROR
Maximum Response Time	300 ms
Characteristics	These commands take effect immediately. The configurations are not 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: 1–16. Default: 1.

<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.
<will_topic>	String type. Will topic string. Maximum size: 256 bytes.
<will_msg>	String type. Will message defines the content of the message published to the Will topic when the client is unexpectedly disconnected. Maximum size: 256 bytes.
<will_len>	Integer type. Will message length. Range: 0–256.
<pkt_timeout>	Integer type. Package delivery timeout. Range: 1–60. Default: 5. Unit: s.
<retry_times>	Integer type. Retry times when packet delivery times out. Range: 0–10. Default: 3.
<timeout_notice>	Integer type. Whether to report timeout message when transmitting a packet. 0 Do not report 1 Report
<clean_session>	Integer type. Configure session type. 0 The server must store the client subscriptions after it disconnects. 1 The server must discard any previously maintained information about the client and treat the connection as "clean".
<keep_alive_time>	Integer type. Keep-alive time. Range: 0–3600. Default: 120. Unit: s. It defines the maximum time interval between messages received from a client. If the server does not receive a message from the client within 1.5 times of the keep-alive time period, it disconnects the client as if the client has sent a DISCONNECT message. 0 The client will not be disconnected
<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.
<msg_rcv_mode>	Integer type. Configure the MQTT message receiving mode. 0 MQTT message received from server will be contained in URC. 1 MQTT message received from server will not be contained in URC.
<msg_len_enable>	Integer type. Whether length of MQTT message received from server will be contained in URC. 0 Not contained 1 Contained

<msg_send_mode>	Integer type. Configure the MQTT message sending mode. 0 MQTT message sent to server will be contained in URC. 1 MQTT message sent to server will not be contained in URC.
<product_key>	String type. Product key issued by Alibaba Cloud.
<device_name>	String type. Device name issued by Alibaba Cloud.
<ali_device_secret>	String type. Device secret key issued by Alibaba Cloud.
<device_id>	String type. Device name issued by HUAWEI Cloud.
<hw_device_secret>	String type. Device secret key issued by HUAWEI Cloud.
<product_id>	String type. Product name issued by HUAWEI Cloud.
<product_secret>	String type. Product secret key issued by HUAWEI Cloud.
<nodeid>	String type. Device identification code of the product issued by HUAWEI Cloud.
<qmtping_interval>	Integer type. Maximum retention time of MQTT connection before disconnection due to heartbeat timeout. Range: 5–60. Default: 5. Unit: s.
<protocol_check_enable>	Integer type. Whether to enable MQTT protocol checking. 0 Disable 1 Enable

NOTE

1. If **<will_fg>**=1, then **<will_qos>**, **<will_retain>**, **<will_topic>** and **<will_msg>** must be specified. Otherwise they will be 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, customers need to use **AT+QSSLCFG** to configure the SSL version, cipher suite, secure level, CA certificate, client certificate, client key and ignorance of RTC time, which will be used in MQTT SSL handshake procedure.
4. Please ensure that message delivery does not time out while it is still being sent.
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),"host_name",(range of supported <port> s) OK
Read Command AT+QMTOPEN?	Response [+QMTOPEN: <client_idx> , <host_name> , <port>]

	OK Or ERROR
Write Command AT+QMTOPEN=<client_idx>,<host_name>,<port>	Response OK +QMTOPEN: <client_idx>,<result> Or ERROR
Maximum Response Time	120 s, determined by network
Characteristics	This command takes effect immediately. The configurations are not saved.

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 an 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> Or

	ERROR
Maximum Response Time	30 s
Characteristics	This command takes effect immediately. The configurations are not saved.

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 and 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),"clientid","username","password" OK
Read Command AT+QMTCONN?	Response [+QMTCONN: <client_idx>,<state>] OK Or ERROR
Write Command AT+QMTCONN=<client_idx>,<clientid>[,<username>,<password>]	Response OK +QMTCONN: <client_idx>,<result>[,<ret_code>] Or ERROR
Maximum Response Time	<pkt_timeout> (default 5 s), determined by network
Characteristics	This command takes effect immediately. The configurations are not saved.

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<clientid>	String type. Client identifier string.
<username>	String type. User name of the client. It can be used for authentication.
<password>	String type. Password corresponding to the user name of the client. 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 User Name or Password 5 Connection Refused: Not Authorized
<pkt_timeout>	Integer type. Package delivery timeout. Range: 1–60. Default: 5. Unit: s. The value can be configured by 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 disconnects a client from MQTT server. A DISCONNECT message is sent from the client 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> Or ERROR
Maximum Response Time	30 s
Characteristics	This command takes effect immediately. The configurations are not saved.

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<result>	Integer type. Command execution result.
0	Send the request for disconnection successfully
2	Failed to send the request for disconnection

NOTE

When <result> is 0, it indicates that a DISCONNECT message has been successfully sent to the network, and the MQTT connection is ready to be closed. There are two scenarios:

1. If the server actively disconnects the MQTT connection and the connection is successfully disconnected, report URC **+QMTSTAT: <client_idx>,<err_code>**, and <err_code> is 5.
2. If the server fails to respond to the disconnection request, execute **AT+QMCLOSE** to disconnect the MQTT connection.

3.3.6. AT+QMTSUB Subscribe to Topics

This command subscribes a client to one or more topics of interest. The client sends a **SUBSCRIBE** message to the server to register an interest in one or more topics. The server delivers messages published on these topics 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>] Or ERROR
Maximum Response Time	<pkt_timeout> × <retry_times> (default 15 s), determined by network
Characteristics	This command takes effect immediately. The configurations are not saved.

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. The QoS level at which the client wants to publish the messages. 0 At most once 1 At least once 2 Exactly once
<result>	Integer type. Command execution result. 0 Sent packet successfully and received ACK 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 packet retransmission count. If <result> is 2, it will not be presented.
<pkt_timeout>	Integer type. Package delivery timeout. Range: 1–60. Default: 5. Unit: second. The value can be configured by 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: 3.

NOTE

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

3.3.7. AT+QMTUNS Unsubscribe from Topics

This command unsubscribes a client from one or more topics. The client sends an UNSUBSCRIBE message to the server to unsubscribe from the 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>] Or ERROR
Maximum Response Time	<pkt_timeout> × <retry_times> (default 15 s), determined by network
Characteristics	This command takes effect immediately. The configurations are not saved.

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 Sent packet successfully and received ACK 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 packet retransmission count. If <result> is 2, it will not be presented.
<pkt_timeout>	Integer type. Package delivery timeout. Range: 1–60. Default: 5. Unit: second. The

	value	can	be	configured	by
	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: 3.				

3.3.8. AT+QMTPUBEX Publish Messages (Data Mode)

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 to those topics will be sent by the server to the client as a **PUBLISH** message.

AT+QMTPUBEX Publish Messages (Data Mode)	
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>,<msg_length>	Response > Input the data to be sent. When the actual size of data exceeds <msg_length>, the first <msg_length> byte(s) data will be sent out. OK +QMTPUBEX: <client_idx>,<msgid>,<result>[,<value>] Or ERROR
Maximum Response Time	<pkt_timeout> × <retry_times> (default 15 s), determined by network
Characteristics	This command takes effect immediately. The configurations are not saved.

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<msgid>	Integer type. Message identifier of packet. Range: 0–65535. It will be 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.
<msg_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 published when <qos>=0 does not require ACK) 1 Packet retransmission 2 Failed to send packet
<value>	Integer type. If <result> is 1, it is the packet retransmission count. If <result> is 0 or 2, it will not be presented.
<pkt_timeout>	Integer type. Package delivery timeout. Range: 1–60. Default: 5. Unit: second. The value can be configured by 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: 3.

NOTE

1. If this command is executed successfully and **OK** is returned, the client can continue to publish new packet. The maximum quantity of transmitted packets should not exceed inflight window size: 5. Otherwise, **ERROR** will be returned.
2. After executing this command, the client will be ready to send data as payload. The maximum length of input data is 1500 bytes at a time.
3. PUBLISH messages can be sent either from a publisher to the server, or from the server to a subscriber. When a server publishes messages to a subscriber, the following URC will be returned to notify the host to read the received data that is reported from MQTT server: **+QMTRECV:<client_idx>,<msgid>,<topic>,<payload_length>,<payload>**. For more details about the URC description, please refer to **Chapter 4.2**.

3.3.9. AT+QMTPUB Publish Messages (Command Mode)

This command publishes specified messages or messages with any length 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+QMTPUB Publish Messages (Command Mode)	
Test Command AT+QMTPUB=?	Response +QMTPUB: (range of supported <client_idx>s), <msgid>,(range of supported <qos>s),(list of supported <retain>s),"topic" OK
Write Command AT+QMTPUB=<client_idx>,<msgid>,<qos>,<retain>,<topic>[,<message>]	Response If <message> is omitted: > Input the data to be sent. OK +QMTPUB: <client_idx>,<msgid>,<result>[,<value>] Or ERROR If <message> is specified: OK +QMTPUB: <client_idx>,<msgid>,<result>[,<value>] Or ERROR
Maximum Response Time	<pkt_timeout> × <retry_times> (default 15 s), determined by network
Characteristics	This command takes effect immediately. The configurations are not saved.

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<msgid>	Integer type. Message identifier of a packet. Range: 0–65535. It will be 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.
<message>	Integer type. Message that needs to be published. Maximum length: 1500 bytes.

<result>	Integer type. Command execution result. 0 Packet sent successfully and ACK received from server (message published when <qos> =0 does not require ACK) 1 Packet retransmission 2 Failed to send a packet
<value>	Integer type. If <result> is 1, it is the packet retransmission count. If <result> is 0 or 2, it will not be presented.
<pkt_timeout>	Integer type. Packet delivery timeout, which can be configured by AT+QMTCFG="timeout",<client_idx>[,<pkt_timeout>,<retry_times>,<timeout_notice>] . Range: 1–60. Default value: 5. Unit: second.
<retry_times>	Integer type. Retry times after packet delivery failure. Range: 0–10. Default value: 3.

NOTE

1. If this command (without **<message>**) is executed successfully and **OK** is returned, the client can continue to publish a new packet. The maximum quantity of transmitting packets should not exceed the inflight window size: 5. Otherwise, **ERROR** will be returned.
2. After executing this command, the client will be ready to send data, which will be sent as payload. The maximum length of the input data is 1500 bytes at a time.
3. **PUBLISH** messages can be sent either from a publisher to the server, or from the server to a subscriber. When a server publishes messages to a subscriber, the following URC will be returned to notify the host to read the received data that is reported from MQTT server: **+QMTRECV: <client_idx>,<msgid>,<topic>[,<payload_len>],<payload>**. For more details about the URC description, please refer to **Chapter 4.2**.

3.3.10. AT+QMTRECV Read Messages from Buffer

This command reads messages from 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>)] OK If there is no MQTT connection: ERROR
Maximum Response Time	300 ms
Characteristics	This command takes effect immediately. The configurations are not saved.

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<store_status>	Integer type. Indicates if a message stored in the buffer. As maximum 5 messages can be stored in a buffer, the URC can report at most 5 messages simultaneously. 0 No stored messages 1 There are stored messages (up to 5)
<recv_id>	Integer type. Serial number of a received message. Range: 0–4.
<msgid>	Integer type. Message identifier of packet. Range: 0–65535. It will be 0 only when <qos>=0 .
<topic>	String type. Topic that needs to be published.
<payload_len>	Integer type. Payload length.
<payload>	String type. Payload relating to topic name.

4 MQTT-Related URCs

This chapter describes the MQTT-related URCs.

Table 3: MQTT-Related URCs

SN	URC Format	Description
[1]	+QMTSTAT: <client_idx>,<err_code>	When the state of MQTT link layer changes, 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>	When the state of MQTT link layer changes, the client will close the MQTT connection and report the URC.

4.1. +QMTSTAT Indicate State Change in MQTT Link Layer

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

+QMTSTAT Indicate State Change in MQTT Link Layer

+QMTSTAT: <client_idx>,<err_code>	When the state of MQTT link layer changes, 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. Please refer to Table 4 for details.

Table 4: URC Error Codes

<err_code>	Description	How to Do
1	Connection is closed or reset by peer.	Execute AT+QMTOPEN and reopen MQTT connection.
2	Sending PINGREQ packet timed out or failed.	Deactivate PDP first, and then activate PDP and reopen MQTT connection.
3	Sending CONNECT packet timed out or failed.	1. Check whether the inputted user name 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 user name 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 closes MQTT connection.	This is a normal process.
6	The client closes MQTT connection due to repetitive packet sending failure.	1. Make sure the data is correct. 2. Try to reopen MQTT connection since there may be 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 available.
8	The client closes the MQTT connection.	Try to reconnect.
9–255	Reserved for future use.	

4.2. +QMTRECV Notify Host to Read MQTT Packet Data

The URC begins with **+QMTRECV**:. It is mainly used to notify the host to read the received MQTT packet data that is reported from MQTT server.

+QMTRECV Notify Host to Read MQTT Packet Data	
+QMTRECV: <client_idx>,<msgid>,<topic>[,<payload_len>],<payload>	Notify the host to read the received data reported from MQTT server.
+QMTRECV: <client_idx>,<recv_id>	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 received message. Range: 0–4.

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

The URC begins with **+QMTPING:**. It will be reported when server does not receive a message from the client within 1.5 times of the keep-alive time period and it will disconnect the client as if the client has sent a **DISCONNECT** message.

+QMTPING Indicate PING State of Keep-alive in MQTT

+QMTPING: <client_idx>,<result>	When the state of MQTT link layer changes, 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 examples that explain how to use MQTT-related 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, customers can use **AT+QMTCFG="aliauth"** to configure the device information in advance, and do not 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 "This is test data, hello MQTT." and then send it. The maximum length of the data is 1500 bytes and the data exceeding 1500 bytes will be omitted.

AT+QMTPUBEX=0,0,0,0,"topic/pub",30

>This is 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 will report the following information.

+QMTRECV: 0,0,"topic/pub",30,"This is test data, hello MQTT."

//Disconnect a client from MQTT server.

```
AT+QMTDISC=0
```

```
OK
```

```
+QMTDISC: 0,0 //Send the request for disconnection successfully.
```

```
//There are two possible scenarios.
```

```
//The server responds to disconnection request.
```

```
+QMTSTAT: 0,5
```

```
//The server did not respond to the disconnection request, and the module executed the command.
```

```
AT+QMTCLOSE=0
```

```
OK
```

```
+QMTCLOSE: 0,0
```

5.2. Example of MQTT Operation with SSL

See *documents [1]* and *[2]* for **AT+QFUPL** and 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 RAM.
```

```
AT+QFUPL="RAM: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 RAM.
```

```
AT+QFUPL="RAM: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 RAM.

AT+QFUPL="RAM: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,"RAM:cacert.pem"

OK

//Configure CC certificate.

AT+QSSLCFG="clientcert",2,"RAM:client.pem"

OK

//Configure CK certificate.

AT+QSSLCFG="clientkey",2,"RAM:user_key.pem"

OK

//Configure SSL parameters.

AT+QSSLCFG="secllevel",2,2 //SSL authentication mode: server authentication

OK

AT+QSSLCFG="sslversion",2,4 //SSL authentication version

OK

AT+QSSLCFG="ciphersuite",2,"0xFFFF" //Cipher suite

OK

AT+QSSLCFG="ignorelocaltime",2,1 //Ignore the time of authentication.

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

>This is 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,"This is published data from client"

//Disconnect a client from MQTT server.

AT+QMTDISC=0

OK

+QMTDISC: 0,0

//There are two possible scenarios.

//The server responds to disconnection request.

+QMTSTAT: 0,5

//The server did not respond to the disconnection request, and the module executed the command.

AT+QMTCLOSE=0

OK

+QMTCLOSE: 0,0

6 Appendix References

Table 5: Related Documents

Document Name
[1] Quectel_EC2x&EG2x&EG9x&EM05_Series_FILE_Application_Note
[2] Quectel_EC2x&EG2x&EG9x&EM05_Series_SSL_Application_Note

Table 6: Terms and Abbreviations

Abbreviation	Description
ACK	Acknowledgement
MQTT	Message Queuing Telemetry Transport
QoS	Quality of Service
RAM	Random Access Memory
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
URC	Unsolicited Result Code