EC200U&EG915U Series
MQTT Application Note

LTE Standard Module Series

Version: 1.1

Date: 2021-08-20

Status: Released

Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company 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 office. 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 to support@quectel.com.

General Notes

Quectel offers the information as a service to its customers. The information provided is based upon customers' requirements. Quectel makes every effort to ensure the quality of the information it makes available. Quectel does not make any warranty as to the information contained herein, and does not accept any liability for any injury, loss or damage of any kind incurred by use of or reliance upon the information. All information supplied herein is subject to change without prior notice.

Disclaimer

While Quectel has made efforts to ensure that the functions and features under development are free from errors, it is possible that these functions and features could contain errors, inaccuracies and omissions. Unless otherwise provided by valid agreement, Quectel makes no warranties of any kind, implied or express, with respect to the use of features and functions under development. To the maximum extent permitted by law, Quectel excludes all liability for any loss or damage suffered in connection with the use of the functions and features under development, regardless of whether such loss or damage may have been foreseeable.

Duty of Confidentiality

The Receiving Party shall keep confidential all documentation and information provided by Quectel, except when the specific permission has been granted by Quectel. The Receiving Party shall not access or use Quectel's documentation and information for any purpose except as expressly provided herein. Furthermore, the Receiving Party shall not disclose any of the Quectel's documentation and information to any third party without the prior written consent by Quectel. For any noncompliance to the above requirements, unauthorized use, or other illegal or malicious use of the documentation and information, Quectel will reserve the right to take legal action.

Copyright

About the Document

Revision History

Version	Date	Author	Description
-	2021-04-09	Fei XUE	Creation of the document
1.0	2021-05-24	Fei XUE	First official release
1.1	2021-08-20	Fei XUE	Added an applicable module series EG915U.

Contents

Table Index

1 Introduction

Quectel LTE Standard EC200U and EG915U series modules support MQTT. 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 introduces how to use the MQTT function of the following Quectel modules through AT commands.

2 MQTT Data Interaction

This chapter gives the data interaction mechanism of MQTT function.

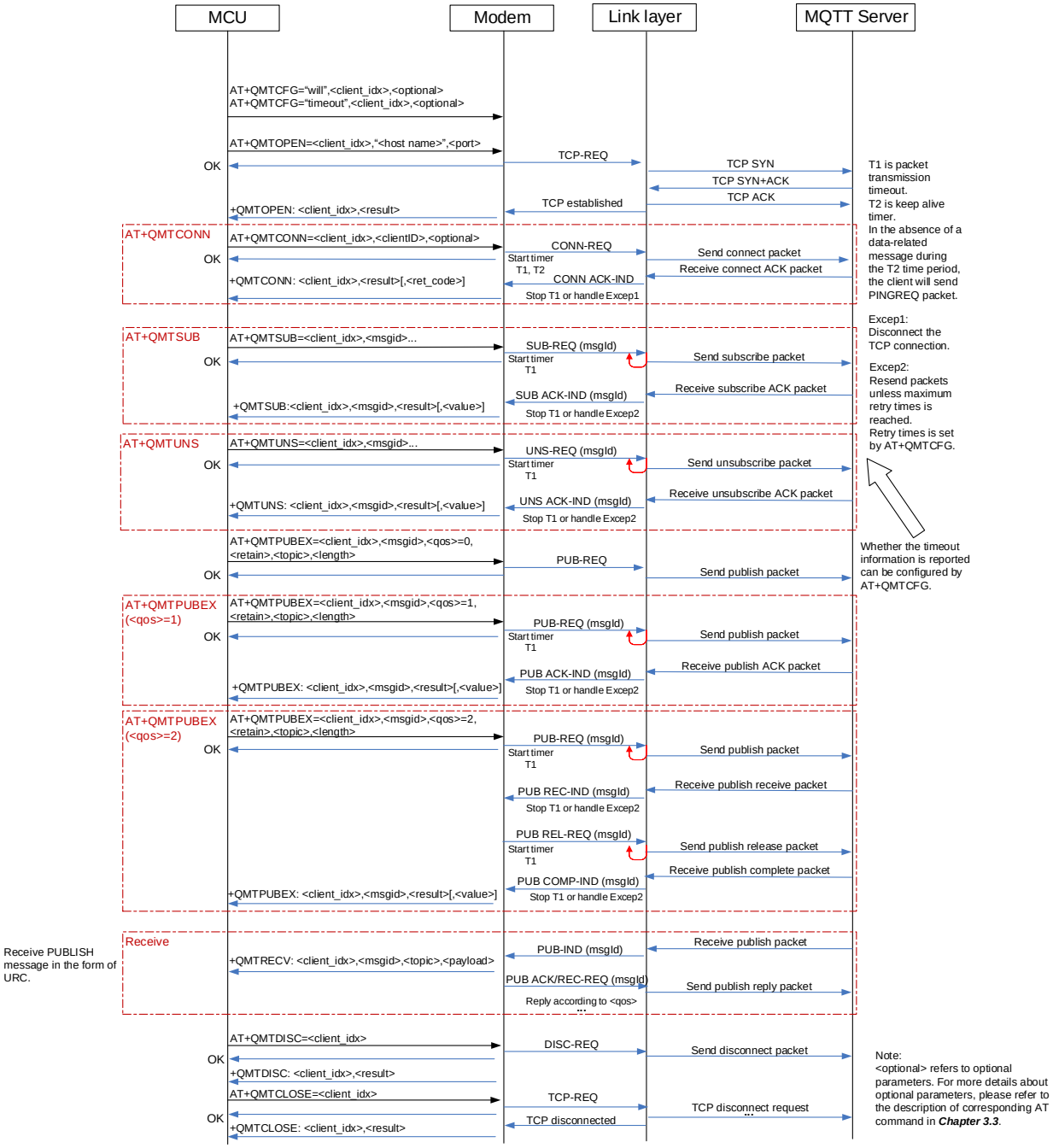

Figure 1: MQTT Data Interaction Diagram

3 MQTT Related AT Commands

This chapter presents the AT commands for operating MQTT function.

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 to 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 1: Types of AT Commands

Command Type	Syntax	Description
Test Command	**AT+<cmd>=?**	Test the existence of corresponding Write Command and return information about the type, value, or range of its parameter.
Read Command	**AT+<cmd>?**	Check the current parameter value of a corresponding Write 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 how to use the AT commands introduced herein. 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: "qmtping",(**range of supported **<client_idx>**s**),(**range of supported **<qmtping_interval>**s**)** **+QMTCFG: "send/mode",(**range of supported **<client_idx>**s**),(**list of supported **<send_mode>**s**)** **+QMTCFG: "aliauth",(**range of supported **<client_idx>**s**),"product key","device name","device secret"** **+QMTCFG: "onenet",(**range of supported **<client_idx>**s**),"product id","access key"** **+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 **<enable_timeout>**s**),(**range of supported **<edit_timeout>**s**)** **OK**
Write Command Configure 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, configure the MQTT protocol version: **OK** If there is any error: **ERROR**
Write Command Configure 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 to be used by the MQTT client: **+QMTCFG: "pdpcid",<cid>** **OK** If the optional parameter is specified and the MQTT connection is not established, configure the PDP to be used by the MQTT client: **OK** If there is any error: **ERROR**

Write Command Configure 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, configure the MQTT SSL mode and SSL context index: **OK** If there is any error: **ERROR**
Write Command Configure 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, configure the keep-alive time: **OK** If there is any error: **ERROR**
Write Command Configure 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, configure the session type: **OK** If there is any error: **ERROR**
Write Command Configure timeout of message delivery **AT+QMTCFG="timeout",<client_idx>[,<pkt_timeout>,<retry_times>,<timeout_notice>**	Response If the optional parameters are omitted, query the timeout of message delivery: **+QMTCFG: "timeout",<pkt_timeout>,<retry_times>,<timeout_notice>**

]	**OK** If the optional parameters are specified and the MQTT connection is not established, configure timeout of message delivery: **OK** If there is any error: **ERROR**
Write Command Configure Will information **AT+QMTCFG="will",<client_idx>[,<will_fg>[,<will_qos>,<will_retain>,<willtopic>,<willmessage>]]**	Response If the optional parameters are omitted, query the current configuration: **+QMTCFG: "will",<will_fg>[,<will_qos>,<will_retain>,<willtopic>,<willmessage>]** **OK** If the optional parameters are specified and the MQTT connection is not established, configure the Will information: **OK** If there is any error: **ERROR**
Write Command Configure 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, configure 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 Configure receiving mode when data is 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, configure the receiving mode when data is received from server:

	OK If there is any error: **ERROR**
Write Command Configure 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, configure the MQTT heartbeat interval: **OK** If there is any error: **ERROR**
Write Command Configure 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 Configure 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, configure Alibaba device information for Alibaba Could: **OK** If there is any error: **ERROR**
Write Command Configure the MQTT device for	Response If the optional parameters are omitted, query the current configuration:

China Mobile OneNET IoT platform **AT+QMTCFG="onenet",<client_idx>[,<product id>,<access key>]**	**+QMTCFG: "onenet",<product id>,<access key>** **OK** If the optional parameters are specified, and the MQTT connection is not established, configure the MQTT device for OneNET IoT platform: **OK** If there is any error: **ERROR**
Write Command Configure 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, configure the MQTT data format: **OK** If there is any error: **ERROR**
Write Command Configure 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, configure the MQTT data view mode in transparent mode: **OK** If there is any error: **ERROR**
Write Command Configure the timeout of message editing **AT+QMTCFG="edit/timeout",<client_idx>[,<enable_timeout>,[<edit_timeout>]]**	Response If the optional parameters are omitted, query the current configuration: **+QMTCFG: "edit/timeout",<enable_timeout>[,<edit_timeout>]** **OK** If the optional parameters are specified, and the MQTT connection is not established, configure the timeout of message editing: **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. The PDP to be used by the MQTT client. Range: 1–7. Default value: 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. The 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. The 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. The length of Will message. Range: 1–256. Unit: byte.
<pkt_timeout>	Integer type. Timeout of the 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 Not report 1 Report
<clean_session>	Integer type. Configure the session type. 0 The server must store the subscriptions of the client after it disconnects. (effective only when the server supports the operation of storing session information) 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 value: 120. Unit: s. It defines the maximum time interval between messages received from a client. If the server does not receive a message (Interactive data or keep-alive package) from the client within 1.5 times the keep-alive time period, it disconnects the client as if the client has sent a DISCONNECT message. 0 The client is not 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_recv_mode>	Integer type. Whether the 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 verification certificate issued by Alibaba Cloud.
<qmtping_interval>	Integer type. The heartbeat interval. Range: 5–60. Default value: 5. Unit: s.
<send_mode>	Integer type. MQTT message sending format. 0 String 1 Hex
<product id>	String type. Product ID in the product overview page of OneNET platform.
<access key>	String type. Key in the device details of OneNET platform.
<recv_mode>	Integer type. MQTT message receiving mode. 0 String 1 Hex
<view_mode>	Integer type. MQTT data view mode in transparent mode. 0 The data is not displayed in transparent mode 1 The data can be displayed in transparent mode.
<enable_timeout>	Integer type. When publishing messages, whether to exit the editing mode automatically when editing messages times out. 0 Disable 1 Enable
<edit_timeout>	Integer type. Timeout for editing the message when the message is published. Range: 1–180. Unit: s. In the editing mode, if no data is inputted or the length of the inputted data does not reach the setting length, the module will exit editing mode automatically after timeout, and **Error** will be reported.

NOTES

1. If **<will_fg>**=1, then **<will_qos>**, **<will_retain>**, **<willtopic>** and **<willmessage>** must be present. Otherwise, they are omitted.
2. If MQTT connection is configured to SSL mode, **<SSL_ctx_idx>** must be specified. Also, you can use **AT+QSSLCFG** to configure the SSL version, cipher suite, secure level, CA certificate, client certificate, client secret key and ignorance of RTC time, which need to be used in MQTT SSL handshake. For more details, please refer to ***document [1]***.
3. It is crucial to ensure message delivery does not timeout while the message is still being sent.
4. **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. The address of the server. It could be an IP address or a domain name. The maximum size is 100 bytes.
<port>	Integer type. The port of the server. Range: 1–65535.
<result>	Integer type. Result of the command execution. -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. Result of the command execution. -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 from 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**),"clientid","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. The 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. Result of the command execution. 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. Timeout of the packet delivery. Range: 1–60. Default value: 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>** 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. Result of the command execution. -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 by a client to register an interest in one or more topics with the server. 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: 0–65535.
<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. Result of the command execution. 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 means the times of packet retransmission. If **<result>** is 2, it is not presented.
<pkt_timeout>	Integer type. Timeout of the packet delivery. Range: 1–60. Default value: 5. Unit: s. 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 value: 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 exactly one 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: 0–65535.
<topic>	String type. Topic that the client wants to subscribe to or unsubscribe from.
<result>	Integer type. Result of the command execution. 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 means the times of packet retransmission. If **<result>** is 2, it is not presented.
<pkt_timeout>	Integer type. Timeout of the packet delivery. Range: 1–60. Default value: 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 value: 3.

3.3.8. AT+QMTPUBEX Publish Messages

This command publishes messages with fixed 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 to those topics are 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 receiving **>**, 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. The QoS level at which the client wants to publish the 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 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. Result of the command execution. 0 Packet sent successfully and ACK received from server (message that 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 times of packet retransmission. If **<result>** is 0 or 2, it is not presented.
<pkt_timeout>	Integer type. Timeout of the packet delivery. Range: 1–60. Default value: 5. Unit: second. It 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 value: 3.

NOTES

1. If this command is executed successfully and **OK** is returned, the client can continue to publish new packet. The maximum quantity of to-be-transmitted packets 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 is 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 is 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 information, please refer to ***Chapter 4.2***.

3.3.9. AT+QMTRECV Read Messages from Buffers

This command reads the messages from the storage buffer where the messages are stored when they are reported by the server.

AT+QMTRECV Read Messages from Buffers	
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 **[+QMTRECV: <client_idx>,<msgid>,<topic>[,<payload_len>],<payload>** **[…]]** **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 whether there is a message stored in the buffer. A maximum of 5 messages can be stored in the buffer. Therefore, URC reports maximally 5 messages simultaneously.

	0 No message in the buffer 1 The buffer stores one or more messages
<recv_id>	Integer type. Indicate the serial number of every message received. 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. The length of payload.
<payload>	String type. The payload that relates to the topic name.

4 MQTT Related URCs

This chapter describes MQTT related URCs.

Table 2: MQTT Related URCs

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

Parameter

<client_idx>	Integer type. MQTT client identifier.
<err_code>	Integer type. Error code. Refer to the table below for details.

Table 3: Error Codes of the URC

<err_code>	Description	How to do
1	Connection is closed or reset by a peer end.	Execute **AT+QMTOPEN** command and reopen MQTT connection.
2	Sending PINGREQ packet timed out or failed.	Deactivate PDP first, and then active 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 packet sending failure all the time.	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 currently.
8	The client closes the MQTT connection.	Try to reconnect.
9–255	Reserved for future use.	

4.2. +QMTRECV URC to Notify the 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 URC to Notify the Host to Read MQTT Packet Data	
+QMTRECV: <client_idx>,<msgid>,<topic>[,<payload_len>],<payload>	Notify the host to read the received data that is 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. Range: 0–65535.
<topic>	String type. The topic received from MQTT server.
<payload_len>	Integer type. The length of payload.
<payload>	String type. The payload that relates to the topic name.
<recv_id>	Integer type. Indicate the serial number of every message received. Range: 0–4.

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

The URC begins with **+QMTPING:**. It is reported when server receives no message from the client within 1.5 times the keep-alive time period 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>	When the state of MQTT link layer is 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 gives the examples to 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" command to configure the device information in advance, and do not need to provide username/password here anymore.
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 that beyond 1500 bytes is 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 reports 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 //Connection closed successfully.

5.2. Example of MQTT Operation with 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 SSL parameters.
AT+QSSLCFG="seclevel",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 publish 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 publish data from client"

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

+QMTDISC: 0,0
```

6 Appendix References

Table 4: Related Documents

Document Name
[1] Quectel_EC200U&EG915U_Series_SSL_Application_Note

Table 5: Terms and Abbreviations

Abbreviation	Description
ACK	Acknowledgement
CA	Certificate Authority
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