BG95&BG77&BG600L Series MQTT Application Note

LPWA Module Series

Version: 1.2

Date: 2023-03-14

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.

About the Document

Revision History

Version	Date	Author	Description
1.0	2019-08-08	Lane HAO	First official release
1.1	2020-05-28	Jaryoung LI	1. Added an applicable module BG600L-M3. 2. Updated the maximum length of messages to be published into 4096 bytes (<msglen> of AT+QMTPUB). 3. Deleted AT+QMTCFG="prefix". 4. Updated the error response of the AT commands (Chapter 3.2). 5. Deleted the summary of error codes.
1.2	2023-03-14	Mario CHEN	1. Updated the description of <clientID>, <username> and <password> in AT+QMTCONN (Chapter 3.2.4). 2. Updated the description of <result> in AT+QMTDISC (Chapter 3.2.5). 3. Updated the examples of MQTT operation without or with SSL (Chapter 5).

Contents

Table Index

1 Introduction

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 introduces how to use the MQTT function of Quectel BG95 series, BG77 and BG600L-M3 modules through AT commands.

2 MQTT Data Interaction

This chapter illustrates the data interaction mechanism of MQTT function.

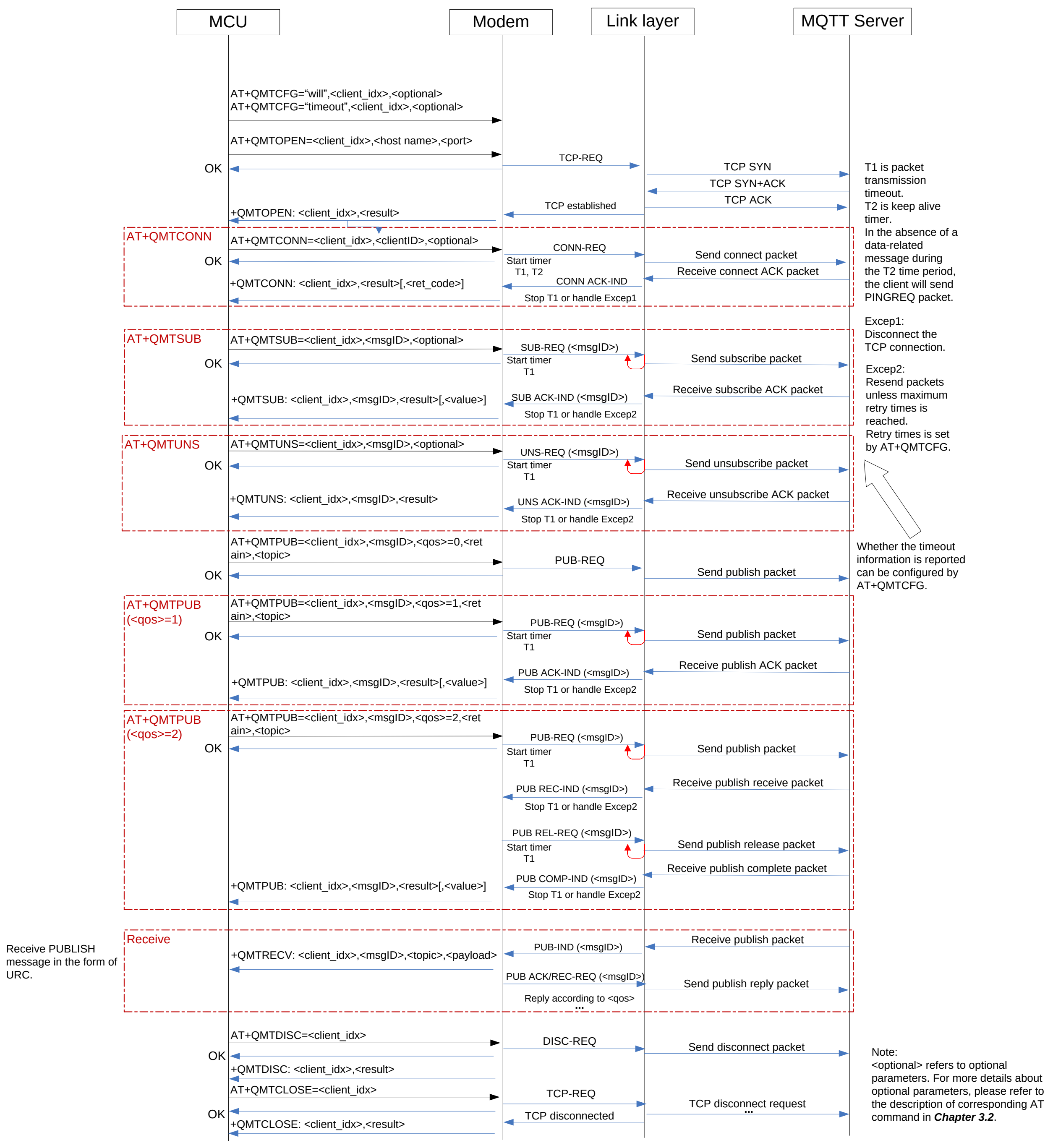
MCU
Modem
Link layer
MQTT Server
AT+QMTCFG="will",<client_idx>,<optional>
AT+QMTCFG="timeout",<client_idx>,<optional>
AT+QMTOPEN=<client_idx>,<host name>,<port>
OK
TCP-REQ
TCP SYN
TCP SYN+ACK
TCP ACK
TCP established
+QMTOPEN: <client_idx>,<result>
T1 is packet transmission timeout.
T2 is keep alive timer.
In the absence of a data-related message during the T2 time period, the client will send PINGREQ packet.
AT+QMTCONN
AT+QMTCONN=<client_idx>,<clientID>,<optional>
OK
CONN-REQ
Start timer T1, T2
Send connect packet
Receive connect ACK packet
CONN ACK-IND
+QMTCONN: <client_idx>,<result>[,<ret_code>]
Stop T1 or handle Excep1
Excep1: Disconnect the TCP connection.
AT+QMTSUB
AT+QMTSUB=<client_idx>,<msgID>,<optional>
OK
SUB-REQ (<msgID>)
Start timer T1
Send subscribe packet
Excep2: Resend packets unless maximum retry times is reached.
Retry times is set by AT+QMTCFG.
Receive subscribe ACK packet
SUB ACK-IND (<msgID>)
+QMTSUB: <client_idx>,<msgID>,<result>[,<value>]
Stop T1 or handle Excep2
AT+QMTUNS
AT+QMTUNS=<client_idx>,<msgID>,<optional>
OK
UNS-REQ (<msgID>)
Start timer T1
Send unsubscribe packet
Receive unsubscribe ACK packet
UNS ACK-IND (<msgID>)
+QMTUNS: <client_idx>,<msgID>,<result>
Stop T1 or handle Excep2
AT+QMTPUB=<client_idx>,<msgID>,<qos>=0,<retain>,<topic>
PUB-REQ
OK
Send publish packet
Whether the timeout information is reported can be configured by AT+QMTCFG.
AT+QMTPUB (<qos>=1)
AT+QMTPUB=<client_idx>,<msgID>,<qos>=1,<retain>,<topic>
OK
PUB-REQ (<msgID>)
Start timer T1
Send publish packet
Receive publish ACK packet
PUB ACK-IND (<msgID>)
+QMTPUB: <client_idx>,<msgID>,<result>[,<value>]
Stop T1 or handle Excep2
AT+QMTPUB (<qos>=2)
AT+QMTPUB=<client_idx>,<msgID>,<qos>=2,<retain>,<topic>
OK
PUB-REQ (<msgID>)
Start timer T1
Send publish packet
Receive publish receive packet
PUB REC-IND (<msgID>)
Stop T1 or handle Excep2
PUB REL-REQ (<msgID>)
Start timer T1
Send publish release packet
Receive publish complete packet
PUB COMP-IND (<msgID>)
+QMTPUB: <client_idx>,<msgID>,<result>[,<value>]
Stop T1 or handle Excep2
Receive
Receive PUBLISH message in the form of URC.
Receive publish packet
PUB-IND (<msgID>)
+QMTRECV: <client_idx>,<msgID>,<topic>,<payload>
PUB ACK/REC-REQ (<msgID>)
Send publish reply packet
Reply according to <qos>
...
AT+QMTDISC=<client_idx>
DISC-REQ
Send disconnect packet
OK
+QMTDISC: <client_idx>,<result>
AT+QMTCLOSE=<client_idx>
TCP-REQ
TCP disconnect request
...
OK
TCP disconnected
+QMTCLOSE: <client_idx>,<result>
Note:
<optional> refers to optional parameters. For more details about optional parameters, please refer to the description of corresponding AT command in *Chapter 3.2*.

Figure 1: MQTT Data Interaction Diagram

3 MQTT Related AT Commands

The AT commands for using the MQTT function are outlined in this chapter.

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 1: 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.

3.1.3. 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.

3.2. Description of MQTT-Related AT Commands

3.2.1. 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 **<ctx_index>**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**),<will_topic>,<will_message>** **+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>**

	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, 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, 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>[,<ctx_index>]]**	Response If the optional parameters are omitted, query the MQTT SSL mode and SSL context index: **+QMTCFG: "ssl",<SSL_enable>[,<ctx_index>]** **OK** If any of the optional parameters is specified, 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="keepa**	Response If the optional parameter is omitted, query the keep-alive time: **+QMTCFG: "keepalive",<keep_alive_time>**

live",<client_idx>[,<keep_alive_time>]	**OK** If the optional parameter is specified, 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, set the session type: **OK** If there is any error: **ERROR**
Write Command Query/Set the message delivery timeout **AT+QMTCFG="timeout",<client_idx>[,<pkt_timeout>[,<retry_times>][,<timeout_notice>]]**	Response If the optional parameters are omitted, query the message delivery timeout: **+QMTCFG: "timeout",<pkt_timeout>,<retry_times>,<timeout_notice>** **OK** If any of the optional parameters is specified, set the timeout of message delivery: **OK** If there is any error: **ERROR**
Write Command Query/Set the Will information **AT+QMTCFG="will",<client_idx>[,<will_fg>[,<will_qos>,<will_retain>,<will_topic>,<will_message>]]**	Response If the optional parameters are omitted, query the Will information: **+QMTCFG: "will",<will_fg>[,<will_qos>,<will_retain>,<will_topic>,<will_message>]** **OK** If any of the optional parameters is specified, set the Will information: **OK**

	If there is any error: **ERROR**
Write Command Query/Set the MQTT message receiving mode when the 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 any of the optional parameters is specified, set the MQTT message receiving mode: **OK** If there is any error: **ERROR**
Write Command Query/Set Alibaba device information for AliCloud **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, set the device information: **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes 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 v3.1 4 MQTT v3.1.1
<cid>	Integer type. The PDP to be used by the MQTT client. Range: 1–16. Default value: 1.
<will_fg>	Integer type. Whether to configure the Will flag.

	0 Ignore the Will flag configuration 1 Require the Will flag configuration
<will_qos>	Integer type. QoS level of 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 retain the message after it has been delivered to the current subscribers 1 When a client sends a PUBLISH message to a server, the server will retain the message after it has been delivered to the current subscribers
<will_topic>	String type. Will topic. Maximum length: 255 bytes.
<will_message>	String type. The Will message defines the content of the message that is published on the Will topic if the client is unexpectedly disconnected. It can be a zero-length message. Maximum length: 255 bytes.
<pkt_timeout>	Integer type. Packet delivery timeout. Range: 1–60. Default value: 5. Unit: second.
<retry_times>	Integer type. Retry times when packet delivery times out. Range: 0–10. Default value: 3.
<timeout_notice>	Integer type. Whether the module need to report timeout message when it is transmitting a packet. 0 Do not report 1 Report
<clean_session>	Integer type. Configures the session type. 0 The server must store the subscriptions of the client after it is disconnected. 1 The server must discard any previously maintained information about the client after it disconnects and treat the connection as “clean”.
<keep_alive_time>	Integer type. Keep-alive time. Range: 0–3600. Default value: 120. Unit: second. 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 value, it disconnects the client as if the client has sent a DISCONNECT message. If the keep-alive time value is 0, this means the server is not required to disconnect the client on the grounds of inactivity.
<SSL_enable>	Integer type. MQTT SSL mode. 0 Use normal TCP connection for MQTT 1 Use SSL TCP secure connection for MQTT
<ctx_index>	Integer type. SSL context index. Range: 0–5.
<msg_recv_mode>	Integer type. 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. 0 Length of MQTT message received from server will not be contained in URC 1 Length of MQTT message received from server will be contained in URC
<product_key>	String type. Product key issued by AliCloud.

<device_name>	String type. Device name issued by AliCloud.
<device_secret>	String type. Device secret key issued by AliCloud.

NOTE

1. If **<will_fg>**=1, then **<will_qos>**, **<will_retain>**, **<will_topic>** and **<will_message>** 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, **<ctx_index>** must be specified, and **AT+QSSLCFG** must be used 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. For more details of **AT+QSSLCFG**, please see ***document [1]***.
4. Ensure that message delivery does not time out while the sending process is in progress.
5. **AT+QMTCFG="aliauth"** is only used for AliCloud. If it is configured, **<username>** and **<password>** in **AT+QMTCONN** can be omitted. Once **AT+QMTCFG="aliauth"** is set, a specified session can only be used for AliCloud until the module reboots.

3.2.2. AT+QMTOPEN Open a Network Connection for MQTT Client

This command opens a network connection for an MQTT client.

AT+QMTOPEN Open a Network Connection 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**
Write Command **AT+QMTOPEN=<client_idx>,<host_name>,<port>**	Response **OK** **+QMTOPEN: <client_idx>,<result>** If there is any error: **ERROR**
Maximum Response Time	Determined by network
Characteristics	/

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 number. Range: 0–65535.
<result>	Integer type. Command execution result. -1 Failed to open network 0 Network opened successfully 1 Wrong parameter 2 MQTT client identifier is occupied 3 Failed to activate PDP 4 Failed to parse domain name 5 Network connection error

3.2.3. AT+QMTCLOSE Close a Network Connection for MQTT Client

This command closes a network connection for an MQTT client.

AT+QMTCLOSE Close a Network Connection 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	300 ms
Characteristics	/

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

3.2.4. AT+QMTCONN Connect a Client to MQTT Server

This command is used when a client requests a connection to the 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**
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
Characteristics	/

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<clientID>	String type. Client identifier string. Maximum size: 256 bytes.
<username>	String type. Client username. It can be used for authentication. Maximum size: 512 bytes.
<password>	String type. Password corresponding to the client username. It can be used for authentication. Maximum size: 512 bytes.
<result>	Integer type. Command execution result. 0 Packet sent successfully and ACK received from server 1 Packet retransmission 2 Failed to send a 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.

NOTE

1. If a client with the same client identifier is already connected to the server, the "older" client must be disconnected by the server before completing the CONNECT flow of the new client.
2. **AT+QMTCFG="aliauth"** is only used for AliCloud. If it is configured, then **<username>** and **<password>** in **AT+QMTCONN** can be omitted.

3.2.5. AT+QMTDISC Disconnect a Client from MQTT Server

This command requests a disconnection of a client from MQTT server. 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	**<pkt_timeout>** (default 5 s), determined by network
Characteristics	/

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.

<result>	Integer type. Command execution result 0 Send **DISCONNECT** message successfully 2 Failed to send **DISCONNECT** message

NOTE

When **<result>** is 0, it means that the MQTT **DISCONNECT** message has been delivered to network side and this MQTT session is expected to be closed. There are two scenarios here:
1. If server closes the MQTT session actively after receiving MQTT **DISCONNECT** message, a URC **+QMTSTAT** with **<err_code>** of 5 will be reported.
2. If the server does not respond, customer calls **AT+QMTCLOSE** to close the MQTT session after receiving **+QMTDISC: <client_idx>,0** URC, and the URC **+QMTCLOSE** will be reported.

3.2.6. 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**),(**range of supported **<msgID>**s**),<topic>,(**range of supported **<qos>**s**)** **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
Characteristics	The 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: 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 the server 1 Packet retransmission 2 Failed to send a packet
<value>	Integer type. If **<result>** is 0, it is a vector of granted QoS levels. If **<result>** is 1, is the number of times the packet has been retransmitted. If **<result>** is 2, it will not be presented.
<pkt_timeout>	Integer type. Packet delivery timeout. Range: 1–60. Default value: 5. Unit: second.
<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 among the set of "inflight" messages in a particular communication direction. It typically increases by 1 from one message to the next.

3.2.7. AT+QMTUNS Unsubscribe from Topics

This command unsubscribes a client from one or more topics. The client sends an UNSUBSCRIBE message is sent by the client 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**),(**range of supported **<msgID>**s**),<topic>** **OK**
Write Command **AT+QMTUNS=<client_idx>,<msgID>,<topic1>[,<topic2>…]**	Response **OK** **+QMTUNS: <client_idx>,<msgID>,<result>** If there is any error: **ERROR**
Maximum Response Time	**<pkt_timeout>** × **<retry_times>** (default 15 s), determined by network

Characteristics	The 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: 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 the server 1 Packet retransmission 2 Failed to send a packet
<pkt_timeout>	Integer type. Packet delivery timeout. Range: 1–60. Default value: 5. Unit: second.
<retry_times>	Integer type. Retry times when packet delivery times out. Range: 0–10. Default value: 3.

3.2.8. AT+QMTPUB Publish Messages (Data Mode)

This command publishes 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+QMTPUB Publish Messages (Data Mode)	
Test Command **AT+QMTPUB=?**	Response **+QMTPUB: (**range of supported **<client_idx>**s**),(**range of supported **<msgID>**s**),(**range of supported **<qos>**s**),(**list of supported **<retain>**s**),<topic>,(**range of supported **<msglen>**s**)** **OK**
Write Command Publish variable-length messages **AT+QMTPUB=<client_idx>,<msgID>,<qos>,<retain>,<topic>**	Response **>** After **>** is reported, input the data to be sent. Press **Ctrl+Z** to send the data, or tap **Esc** to cancel the operation. **OK** **+QMTPUB: <client_idx>,<msgID>,<result>[,<value>]** If there is any error: **ERROR**
Write Command Publish fixed-length messages	Response **>**

AT+QMTPUB=<client_idx>,<msgID>,<qos>,<retain>,<topic>,<msglen>	After **>** is responded, input the data to be sent. The number of bytes of input data must equal **<msglen>**. **OK** **+QMTPUB: <client_idx>,<msgID>,<result>[,<value>]** If there is any error: **ERROR**
Maximum Response Time	**<pkt_timeout>** × **<retry_times>** (default 15 s), determined by network
Characteristics	/

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. 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. Determines whether the server will retain the message after it has been delivered to the current subscribers. 0 The server will not retain the message after it has been delivered to the current subscribers 1 The server will retain the message after it has been delivered to the current subscribers
<topic>	String type. Topic that needs to be published.
<msglen>	Integer type. Length of the message to be published. Range: 1–4096. Unit: byte.
<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 a packet
<value>	Integer type. If **<result>** is 1, it means number of times a packet has been retransmitted. If **<result>** is 0 or 2, it will not be presented.
<pkt_timeout>	Integer type. Packet delivery timeout. Range: 1–60. Default value: 5 Unit: second.
<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 reported, the client can continue to publish new packets. The maximum quantity of packets to be transmitted should not be greater than that of inflight windows (5).
2. After executing this command, the client will be ready to send data as a payload. The maximum length of data input at a time is 4096 bytes. To send the data, remember to press tap **Ctrl+Z**.
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 are reported by MQTT server:
   +QMTRECV: <client_idx>,<msgID>,<topic>[,<payload_len>],<payload>.
   For more details about the URC description, Refer to ***Chapter 4.2***.

3.2.9. AT+QMTPUBEX Publish Messages (Command Mode)

This command publishes messages. It provides the same functions as **AT+QMTPUB**, except that the format is different.

AT+QMTPUBEX Publish Messages (Command Mode)	
Test Command **AT+QMTPUBEX=?**	Response **+QMTPUBEX: (**range of supported **<client_idx>**s**),(** range of supported **<msgID>**s**),(**range of supported **<qos>**s**),(**list of supported **<retain>**s**),<topic>,<msg>** **OK**
Write Command **AT+QMTPUBEX=<client_idx>,<msgID>,<qos>,<retain>,<topic>,<msg>**	Response **OK** **+QMTPUB: <client_idx>,<msgID>,<result>[,<value>]** If there is any error: **ERROR**
Maximum Response Time	**<pkt_timeout>** × **<retry_times>** (default 15 s), determined by network
Characteristics	/

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. 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. Determines whether the server will retain the message after it has been delivered to the current subscribers. 0 The server will not retain the message after it has been delivered to the current subscribers 1 The server will retain the message after it has been delivered to the current subscribers
<topic>	String type. Topic that needs to be published.
<msg>	String type. Message to be published. Maximum length is 560 bytes.
<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 a packet
<value>	Integer type. If **<result>** is 1, it means it means the number of times a packet has been retransmitted . If **<result>** is 0 or 2, it will not be presented.
<pkt_timeout>	Integer type. Packet delivery timeout. Range: 1–60. Default value: 5. Unit: second.
<retry_times>	Integer type. Retry times when packet delivery times out. Range: 0–10. Default value: 3.

NOTE

The response flow of **AT+QMTPUBEX** will be the same as the response flow **AT+QMTPUB**.

3.2.10. AT+QMTRECV Read Messages from Buffer

This command reads messages from the storage buffer where the messages are stored after they are 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 received message: **OK** If there is no MQTT connection: **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<store_status>	Integer type. Indicates if a message is stored in the buffer. As there can be maximum 5 messages 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. All buffer data will be read if this parameter is not specified.
<msgID>	Integer type. Message identifier of a 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 related topic name.

4 MQTT Related URCs

This chapter introduces MQTT-related URCs.

Table 1: 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>**	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.

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

The URC that begins with **+QMTSTAT:** will be reported when the state of MQTT link layer is changed.

+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. Range: 0–5.
<err_code>	Integer type. Error code. Refer to the table below for details.

Table 2: Error Codes of +QMTSTAT: URC

<err_code>	Description	How to do
1	The connection is closed or reset by a peer.	Execute **AT+QMTOPEN** to reopen the MQTT connection.
2	Sending PINGREQ packet timed out	First deactivate PDP, and then activate PDP and

	or failed.	reopen 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 identifier is not used. 3. Reopen MQTT connection and try to send CONNECT packet to the 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 identifier is not used. 3. Reopen MQTT connection and try to send CONNECT packet to the server again.
5	The client sends DISCONNECT packet to sever and the server is initiative to close MQTT connection.	This is a normal process.
6	The client takes the initiative to close the 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 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.
8–255	Reserved for future use.	

4.2. +QMTRECV URC to Notify Host to Read MQTT Packet Data

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

+QMTRECV URC to Inform 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>	Notify that 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 that received from MQTT server.
<payload_len>	Integer type. Payload length.
<payload>	String type. Payload that relates to the topic name.
<recv_id>	Integer type. serial number of every single message received. Range: 0–4.

5 Examples

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

5.1. Example of MQTT Operation Without SSL

```
AT+QMTCFG="aliauth",0,"oyjtmPl5a5j","MQTT_TEST","wN9Y6pZSIly7Exa5qVzcmigEGO4kAazZ"
                                        //Configure Alibaba device information for AliCloud.
OK
AT+QMTOPEN=?
+QMTOPEN: (0-5),<host_name>,(0-65535)

OK
AT+QMTOPEN=0,"iot-as-mqtt.cn-shanghai.aliyuncs.com",1883   //Open a network connection for
MQTT client.
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 AliCloud is connected, AT+QMTCFG="aliauth" can be used to configure the device information in
advance, and there is no 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),(1-65535),<topic>,(0-2)
```

```
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","This is the payload related to topic"

//Unsubscribe from topics.
AT+QMTUNS=0,1,"topic/example"
OK

+QMTUNS: 0,2,0
AT+QMTPUB=?
+QMTPUB : (0-5),(0-65535),(0-2),(0,1),<topic>,(1-4096)

OK

//Publish messages.
AT+QMTPUB=0,0,0,0,"topic/pub"
>This is test data, hello MQTT.     //After receiving >, input data This is test data, hello MQTT. and
                                    then send it. The maximum length of the data to be sent is 4096
                                    bytes. All the data beyond 4096 bytes will be omitted. After
                                    inputting data, press Ctrl+Z to send them.
OK

+QMTPUB: 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","This is test data, hello MQTT."
AT+QMTDISC=0                              //Disconnect a client from MQTT server.
OK

+QMTDISC: 0,0                             //Client sends DISCONNECT message successfully.

+QMTSTAT: 0,5                             //Connection is closed by server successfully. If the server
```

```
                                       doesn't close the TCP connection, this URC would not be
                                       reported.
//If +QMTSTAT: 0,5 is not reported, please use AT+QMTCLOSE to close the TCP connection.
AT+QMTCLOSE=0                          //Close the TCP connection.
OK

+QMTCLOSE: 0,0                         //The TCP connection is closed successfully.
```

5.2. Example of MQTT Operation with SSL

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

//If SSL authentication mode is intended to be set as "manage server and client authentication if
requested by the remote server" (<seclevel>=2 in AT+QSSLCFG), upload server root CA certificate,
client certificate and client private key to UFS.
AT+QFUPL="cacert.pem",1758,100             //Upload CA certificate to UFS.
CONNECT
<Input the cacert.pem data, and the size is 1758 bytes>
+QFUPL: 1758,384a

OK

AT+QFUPL="client.pem",1220,100             //Upload client certificate to UFS.
CONNECT
<Input the client.pem data, and the size is 1220 bytes>
+QFUPL: 1220,2d53

OK
AT+QFUPL="user_key.pem",1679,100           //Upload client private key to UFS.
CONNECT
<Input the user_key.pem data, and the size is 1679 bytes>
+QFUPL: 1679,335f

OK

//Configure the path of CA certificate for SSL context 2.
AT+QSSLCFG="cacert",2,"cacert.pem"
OK

//Configure the path of client certificate for SSL context 2.
```

```
AT+QSSLCFG="clientcert",2,"client.pem"
OK

//Configure the path of client private key for SSL context 2.
AT+QSSLCFG="clientkey",2,"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 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,"MQTT-1"
OK

+QMTCONN: 0,0,0

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

+QMTSUB: 0,1,0,1

//Publish messages.
AT+QMTPUB=0,1,1,0,"$aws/things/MQTT-1/shadow/update/accepted"
>This is publish data from client
OK

+QMTPUB: 0,1,0

//If a client subscribes to a topic named "$aws/things/MQTT-1/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/MQTT-1/shadow/update/accepted","This is publish data from client"

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

+QMTDISC: 0,0                    //Client sends DISCONNECT message successfully.

+QMTSTAT: 0,5                    //Connection is closed by server successfully. If the server
                                 doesn't close the TCP connection, this URC would not be
                                 reported.
//If +QMTSTAT: 0,5 is not reported, please use AT+QMTCLOSE to close the TCP connection.
AT+QMTCLOSE=0                    //Close the TCP connection.
OK

+QMTCLOSE: 0,0                   //The TCP connection is closed successfully.
```

6 Appendix References

Table 3: Related Documents

Document Name
[1] Quectel_BG95&BG77&BG600L_Series_SSL_Application_Note

Table 4: Terms and Abbreviations

Abbreviation	Description
ACK	Acknowledgement
CA	Certificate Authority
LPWA	Low Power Wide Area
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