

EG800Q&EG91xQ Series

MQTT Application Note

LTE Standard Module Series

Version: 1.3

Date: 2024-12-25

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. 2024. All rights reserved.

About the Document

Revision History

| Version | Date | Author | Description |
|---------|------------|--------------|--|
| - | 2022-11-15 | Greyson DONG | Creation of the document |
| 1.0 | 2023-01-31 | Greyson DONG | First official release |
| 1.1 | 2023-09-05 | Greyson DONG | Updated the applicable modules: <ul style="list-style-type: none"> Added EG916Q-GL. Updated EG800Q-EU to EG800Q series. |
| 1.2 | 2024-05-22 | Greyson DONG | Updated EG915Q-NA to EG915Q series. |
| 1.3 | 2024-12-25 | Fawei ZHOU | <ol style="list-style-type: none"> Updated the declaration of AT command examples (Chapter 3.2). Deleted AT+QMTCFG="hwauth" and AT+QMTCFG="hwprodid" (Chapter 3.3.1). Updated the server address in the examples (Chapter 5). |

Contents

| | |
|--|-----------|
| About the Document..... | 3 |
| Contents | 4 |
| Table Index..... | 5 |
| 1 Introduction | 6 |
| 2 MQTT Data Interaction..... | 7 |
| 3 MQTT-Related AT Commands | 8 |
| 3.1. AT Command Introduction | 8 |
| 3.1.1. Definitions..... | 8 |
| 3.1.2. AT Command Syntax | 8 |
| 3.2. Declaration of AT Command Examples | 9 |
| 3.3. Description of MQTT Related AT Commands..... | 9 |
| 3.3.1. AT+QMTCFG Configure Optional Parameters of MQTT | 9 |
| 3.3.2. AT+QMTOPEN Open a Network Connection for MQTT Client | 17 |
| 3.3.3. AT+QMTCLOSE Close a Network Connection 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..... | 22 |
| 3.3.8. AT+QMTPUBEX Publish Messages | 23 |
| 3.3.9. AT+QMTRECV Read Messages from Buffer | 25 |
| 4 MQTT URCs | 27 |
| 4.1. +QMTSTAT URC to Indicate the Reason for State Change in MQTT Link Layer | 27 |
| 4.2. +QMTRECV URC to Notify Host to Read MQTT Packet Data | 28 |
| 4.3. +QMTPING URC to Indicate Keep-Alive Timeout in MQTT | 29 |
| 5 Examples | 30 |
| 5.1. Example of MQTT Operation Without SSL..... | 30 |
| 5.2. Example of MQTT Operation with SSL..... | 32 |
| 6 Appendix References | 34 |

Table Index

| | |
|--|----|
| Table 1: Types of AT Commands | 8 |
| Table 2: MQTT URCs | 27 |
| Table 3: Error Codes of the URC | 28 |
| Table 4: Related Documents | 34 |
| Table 5: Terms and Abbreviations | 34 |

1 Introduction

Quectel LTE Standard EG800Q series and EG91xQ family (EG915Q series and EG916Q-GL) modules support MQTT. MQTT is a broker-based publish/subscribe messaging protocol designed to be open, simple, light-weight and user-friendly. It is designed for connections with remote locations requiring a “small code footprint” or limited network bandwidth.

This document outlines how to use the MQTT function of EG800Q series and EG91xQ family modules through AT commands.

2 MQTT Data Interaction

This chapter gives the data interaction mechanism of MQTT function.

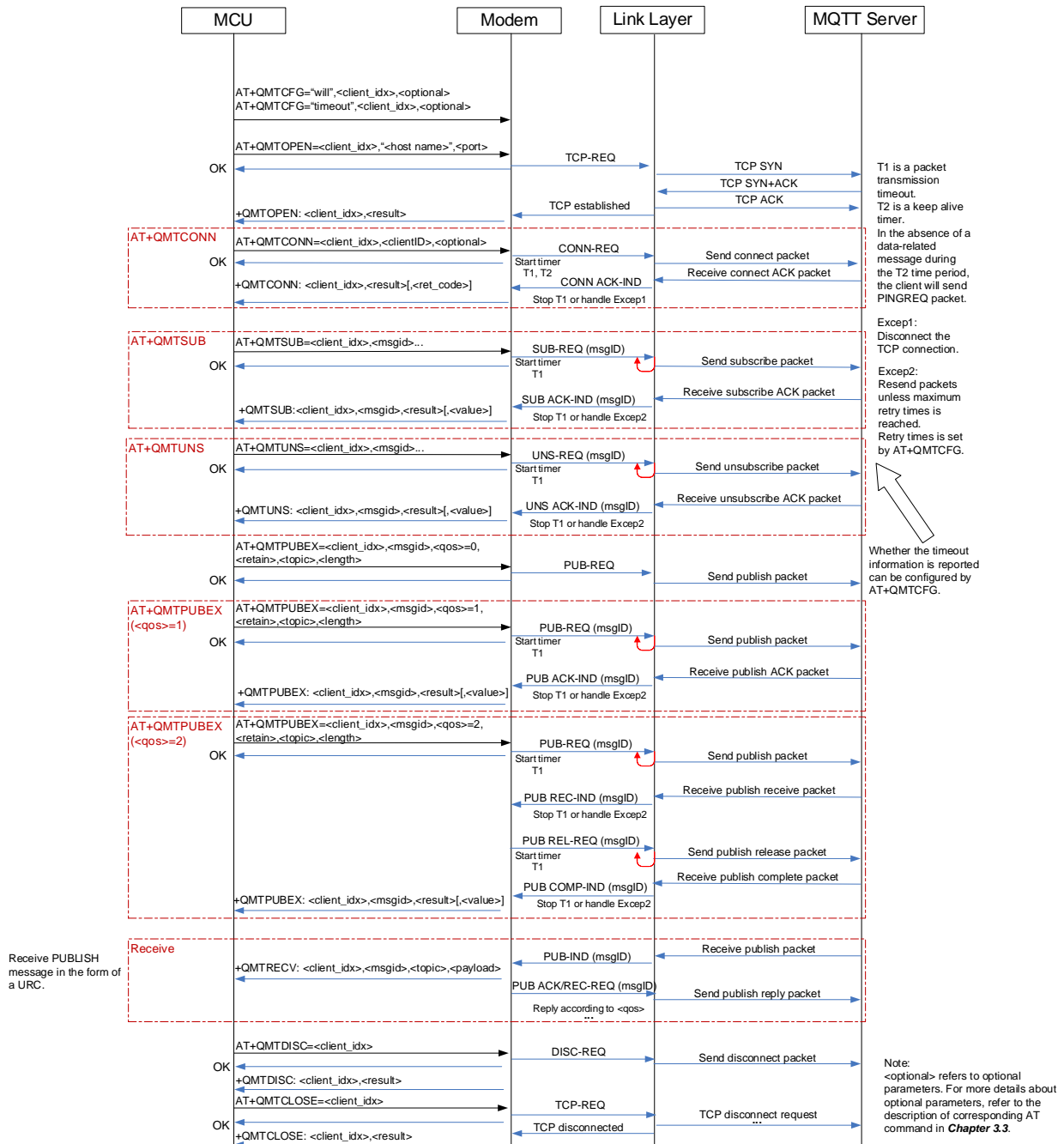


Figure 1: MQTT Data Interaction Diagram

3 MQTT-Related AT Commands

This chapter presents the AT commands for implementing the 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 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 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 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. The URLs, domain names, IP addresses, usernames/accounts, and passwords (if any) in the AT command examples are provided for illustrative and explanatory purposes only, and they should be modified to reflect your actual usage and specific needs.

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=? | <p>Response</p> <p>+QMTCFG: "version",(range of supported <client_idx>s),(list of supported <vsn>s)</p> <p>+QMTCFG: "pdpcid",(range of supported <client_idx>s),(range of supported <cid>s)</p> <p>+QMTCFG: "ssl",(range of supported <client_idx>s),(list of supported <SSL_enable>s),(range of supported <SSL_ctx_idx>s)</p> <p>+QMTCFG: "keepalive",(range of supported <client_idx>s),(range of supported <keep_alive_time>s)</p> <p>+QMTCFG: "session",(range of supported <client_idx>s),(list of supported <clean_session>s)</p> <p>+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)</p> <p>+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"</p> <p>+QMTCFG: "willex",(range of supported <client_idx>s),(list of supported <will_fg>s),(range of supported <will_qos>s),(list of</p> |

| | |
|---|--|
| | <p>supported <will_retain>s),"willtopic",(range of supported <will_len>s)</p> <p>+QMTCFG: "recv/mode",(range of supported <client_idx>s),(list of supported <msg_recv_mode>s),(list of supported <msg_len_enable>s)</p> <p>+QMTCFG: "qmtping",(range of supported <client_idx>s),(range of supported <qmtping_interval>s)</p> <p>+QMTCFG: "send/mode",(range of supported <client_idx>s),(list of supported <send_mode>s)</p> <p>+QMTCFG: "aliauth",(range of supported <client_idx>s),"product_key","device_name","device_secret"</p> <p>+QMTCFG: "onenet",(range of supported <client_idx>s),"product_id","access_key"</p> <p>+QMTCFG: "dataformat",(range of supported <client_idx>s),(list of supported <send_mode>s),(list of supported <recv_mode>s)</p> <p>+QMTCFG: "view/mode",(range of supported <client_idx>s),(list of supported <view_mode>s)</p> <p>+QMTCFG: "edit/timeout",(range of supported <client_idx>s),(list of supported <enable_timeout>s),(range of supported <edit_timeout>s)</p> <p>OK</p> |
| <p>Write Command</p> <p>Query/set the MQTT protocol version</p> <p>AT+QMTCFG="version",<client_idx>[,<vsn>]</p> | <p>Response</p> <p>If the optional parameter is omitted, query the MQTT protocol version:</p> <p>+QMTCFG: "version",<vsn></p> <p>OK</p> <p>If the optional parameter is specified and the MQTT connection is not established, set the MQTT protocol version:</p> <p>OK</p> <p>If there is any error:</p> <p>ERROR</p> |
| <p>Write Command</p> <p>Query/set the PDP to be used by the MQTT client</p> <p>AT+QMTCFG="pdpcid",<client_idx>[,<cid>]</p> | <p>Response</p> <p>If the optional parameter is omitted, query the PDP used by the MQTT client:</p> <p>+QMTCFG: "pdpcid",<cid></p> <p>OK</p> <p>If the optional parameter is specified and the MQTT connection is not established, set the PDP to be used by the MQTT client:</p> <p>OK</p> |

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

| | |
|---|--|
| <pre>try_times>,<timeout_notice>]</pre> | <p>OK</p> <p>If the optional parameters are specified and the MQTT connection is not established, set the message delivery timeout:</p> <p>OK</p> <p>If there is any error:</p> <p>ERROR</p> |
| <p>Write Command</p> <p>Query/set Will information</p> <p>AT+QMTCFG="will",<client_idx>[,<will_fg>[,<will_qos>,<will_retain>,<willtopic>,<willmessage>]]</p> | <p>Response</p> <p>If the optional parameters are omitted, query the current Will information:</p> <p>+QMTCFG: "will",<will_fg>[,<will_qos>,<will_retain>,<willtopic>,<willmessage>]</p> <p>OK</p> <p>If the optional parameters are specified and the MQTT connection is not established, set the Will information:</p> <p>OK</p> <p>If there is any error:</p> <p>ERROR</p> |
| <p>Write Command</p> <p>Query/set Will information</p> <p>AT+QMTCFG="willex",<client_idx>[,<will_fg>[,<will_qos>,<will_retain>,<willtopic>,<will_len>]]</p> | <p>Response</p> <p>If the optional parameters are omitted, query the current setting:</p> <p>+QMTCFG: "willex",<will_fg>[,<will_qos>,<will_retain>,<willtopic>,<will_len>]</p> <p>OK</p> <p>If the optional parameters are specified, set the Will information:</p> <p>></p> <p>Input Will messages. When the actual size of data is greater than <will_len>, the data length exceeding <will_len> will be discarded.</p> <p>OK</p> <p>If there is any error:</p> <p>ERROR</p> |
| <p>Write Command</p> <p>Query/set receiving mode when data is received from server</p> <p>AT+QMTCFG="recv/mode",<client_idx>[,<msg_rcv_mo</p> | <p>Response</p> <p>If the optional parameters are omitted, query the MQTT message receiving mode:</p> <p>+QMTCFG: "recv/mode",<msg_rcv_mode>,<msg_len_enable></p> <p>OK</p> |

| | |
|--|---|
| <p>de>[,<msg_len_enable>]]</p> | <p>If the optional parameters are specified and the MQTT connection is not established, set the receiving mode when data is received from server:</p> <p>OK</p> <p>If there is any error:</p> <p>ERROR</p> |
| <p>Write Command</p> <p>Query/set the MQTT heartbeat interval</p> <p>AT+QMTCFG="qmtping",<client_idx>[,<qmtping_interval>]</p> | <p>Response</p> <p>If the optional parameter is omitted, query the MQTT heartbeat interval:</p> <p>+QMTCFG: "qmtping",<qmtping_interval></p> <p>OK</p> <p>If the optional parameter is specified, and the MQTT connection is not established, configure the MQTT heartbeat interval:</p> <p>OK</p> <p>If there is any error:</p> <p>ERROR</p> |
| <p>Write Command</p> <p>Query/set the MQTT message sending mode</p> <p>AT+QMTCFG="send/mode",<client_idx>[,<send_mode>]</p> | <p>Response</p> <p>If the optional parameter is omitted, query the MQTT message sending mode:</p> <p>+QMTCFG: "send/mode",<send_mode></p> <p>OK</p> <p>If the optional parameter is specified, and the MQTT connection is not established, set the MQTT message sending mode:</p> <p>OK</p> <p>If there is any error:</p> <p>ERROR</p> |
| <p>Write Command</p> <p>Query/set Alibaba device information for AliCloud</p> <p>AT+QMTCFG="aliauth",<client_idx>[,<product_key>,<device_name>,<device_secret>]</p> | <p>Response</p> <p>If the optional parameters are omitted, query the device information:</p> <p>+QMTCFG: "aliauth",<product_key>,<device_name>,<device_secret></p> <p>OK</p> <p>If the optional parameters are specified and the MQTT connection is not established, set Alibaba device information for AliCloud:</p> <p>OK</p> |

| | |
|--|--|
| | <p>If there is any error:</p> <p>ERROR</p> |
| <p>Write Command</p> <p>Query/set the MQTT device for China Mobile OneNET IoT platform</p> <p>AT+QMTCFG="onenet",<client_idx>[,<product_id>,<access_key>]</p> | <p>Response</p> <p>If the optional parameters are omitted, query the MQTT device:</p> <p>+QMTCFG: "onenet",<product_id>,<access_key></p> <p>OK</p> <p>If the optional parameters are specified, and the MQTT connection is not established, set the MQTT device for OneNET IoT platform:</p> <p>OK</p> <p>If there is any error:</p> <p>ERROR</p> |
| <p>Write Command</p> <p>Query/set the MQTT data format</p> <p>AT+QMTCFG="dataformat",<client_idx>[,<send_mode>,<recv_mode>]</p> | <p>Response</p> <p>If the optional parameters are omitted, query the MQTT data format:</p> <p>+QMTCFG: "dataformat",<send_mode>,<recv_mode></p> <p>OK</p> <p>If the optional parameters are specified, and the MQTT connection is not established, set the MQTT data format:</p> <p>OK</p> <p>If there is any error:</p> <p>ERROR</p> |
| <p>Write Command</p> <p>Query/set the MQTT data view mode in transparent mode</p> <p>AT+QMTCFG="view/mode",<client_idx>[,<view_mode>]</p> | <p>Response</p> <p>If the optional parameter is omitted, query the MQTT data view mode in transparent mode:</p> <p>+QMTCFG: "view/mode",<view_mode></p> <p>OK</p> <p>If the optional parameter is specified, and the MQTT connection is not established, set the MQTT data view mode in transparent mode:</p> <p>OK</p> <p>If there is any error:</p> <p>ERROR</p> |
| <p>Write Command</p> <p>Query/set the timeout of message editing</p> <p>AT+QMTCFG="edit/timeout",<client_idx>[,<enable_timeout>]</p> | <p>Response</p> <p>If the optional parameters are omitted, query the timeout of message editing:</p> <p>+QMTCFG: "edit/timeout",<enable_timeout>[,<edit_timeout>]</p> |

| | |
|-----------------------|--|
| ut>,[<edit_timeout>]] | <p>OK</p> <p>If the optional parameters are specified, and the MQTT connection is not established, set the timeout of message editing:</p> <p>OK</p> <p>If there is any error:</p> <p>ERROR</p> |
| Maximum Response Time | 300 ms |
| Characteristic | <p>The command takes effect immediately.</p> <p>The configurations will not be saved.</p> |

Parameter

| | |
|------------------|---|
| <client_idx> | Integer type. MQTT client identifier. Range: 0–5. |
| <vsn> | <p>Integer type. MQTT protocol version.</p> <p>3 MQTT protocol V3.1</p> <p>4 MQTT protocol V3.1.1</p> |
| <cid> | <p>Integer type. PDP to be used by the MQTT client. Range: 1–15.</p> <p>Default value: 1.</p> |
| <will_fg> | <p>Integer type. Query/set the Will flag.</p> <p>0 Ignore the Will flag configuration</p> <p>1 Require the Will flag configuration</p> |
| <will_qos> | <p>Integer type. QoS level of message delivery.</p> <p>0 At most once</p> <p>1 At least once</p> <p>2 Exactly once</p> |
| <will_retain> | <p>Integer type. Will retain flag is only used on PUBLISH messages.</p> <p>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</p> <p>1 When a client sends a PUBLISH message to a server, the server should retain the message after it has been delivered to the current subscribers</p> |
| <willtopic> | String type. Will topic. Length: 1–256 bytes. |
| <willmessage> | String type. Will message defines the content of the message published on the will topic if the client is unexpectedly disconnected. Length: 0–256 bytes. |
| <will_len> | Integer type. Will message length. Range: 1–256. Unit: byte. |
| <pkt_timeout> | Integer type. Timeout of packet delivery. Range: 1–60. Default value: 5. Unit: s. |
| <retry_times> | <p>Integer type. Retry times when packet delivery times out. Range: 0–10.</p> <p>Default value: 3.</p> |
| <timeout_notice> | <p>Integer type. Whether to report timeout message when transmitting packets.</p> <p>0 Do not report</p> <p>1 Report</p> |

| | |
|---------------------------------|---|
| <clean_session> | Integer type. Query/set the session type. 0 The server must store the subscriptions of the client after it disconnects. (Effective only when the server supports 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: 1–3600. Default value: 120. Unit: s. It defines the maximum time interval between messages received from a client. If it's not zero, and 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, the server will disconnect the client as if the client has sent a DISCONNECT message. Otherwise, the server will not disconnect the client. |
| <SSL_enable> | Integer type. Query/set 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. 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 AliCloud. |
| <device_name> | String type. Device name issued by AliCloud. |
| <device_secret> | String type. Device verification certificate issued by AliCloud. |
| <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. |
| <rcv_mode> | Integer type. MQTT message receiving mode. 0 String 1 Hex |
| <view_mode> | Integer type. MQTT data view mode in transparent mode. 0 Data is not displayed in transparent mode 1 Data are displayed in transparent mode. |
| <enable_timeout> | Integer type. When publishing messages, whether to exit the editing mode automatically when editing message times out. 0 Disable 1 Enable |
| <edit_timeout> | Integer type. Timeout for editing published messages. 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 exceeding **<edit_timeout>**, and **ERROR** will be reported.

NOTE

1. If **<will_fg>=1**, then **<will_qos>**, **<will_retain>**, **<willtopic>** and **<willmessage>** must be specified. Otherwise, they are omitted.
2. If MQTT connection is configured to SSL mode, **<SSL_ctx_idx>** 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. For more details, refer to **document [1]**.
3. It is crucial to ensure that message delivery does exceed the maximum response time while sending process is in progress.
4. **AT+QMTCFG="aliauth"** is only used for AliCloud. If it is configured, **<username>** and **<password>** can be omitted in **AT+QMTCONN**.
5. **AT+QMTCFG="onenet"** is only used for OneNET Cloud. If it is configured, **<username>** and **<password>** can be omitted in **AT+QMTCONN**.

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

This command opens a network connection for MQTT client.

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

| | |
|----------------|---|
| Characteristic | / |
|----------------|---|

Parameter

| | |
|--------------|--|
| <client_idx> | Integer type. MQTT client identifier. Range: 0–5. |
| <host_name> | String type. Server address. It could be an IP address or a domain name. Maximum size: 100 bytes. |
| <port> | Integer type. Server port number. 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+QMTCCLOSE Close a Network Connection for MQTT Client

This command closes a network connection for MQTT client.

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

Parameter

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

3.3.4. AT+QMTCONN Connect a Client to MQTT Server

This command connects a client to MQTT server. When a TCP/IP socket connection is established between a client 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 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> × <retry_times> (default 15 s), determined by network |
| Characteristic | / |

Parameter

| | |
|---------------------------|--|
| <client_idx> | Integer type. MQTT client identifier. Range: 0–5. |
| <clientID> | String type. Client identifier string. |
| <username> | String type. Client username. It can be used for authentication. |
| <password> | String type. Password corresponding to the client username. It can be used for authentication. |

| | |
|----------------------------|---|
| <result> | Integer type. Command execution result. 0 Packet sent successfully and ACK received from server (message is published if <qos> =0 does not require ACK) 1 Packet retransmission 2 Failed to send packet |
| <state> | Integer type. MQTT connection state. 1 MQTT is initializing 2 MQTT is connecting 3 MQTT is connected 4 MQTT is disconnecting |
| <ret_code> | Integer type. Connection status return code. 0 Connection Accepted 1 Connection Refused: Unacceptable Protocol Version 2 Connection Refused: Identifier Rejected 3 Connection Refused: Server Unavailable 4 Connection Refused: Bad Username or Password 5 Connection Refused: Not Authorized |
| <pkt_timeout> | Integer type. Packet delivery timeout. Range: 1–60. Default value: 5. Unit: s. The value can be configured by AT+QMTCFG="timeout",<client_id x>[,<pkt_timeout>,<retry_times>,<timeout_notice>] . |
| <retry_times> | Integer type. Retry times when packet delivery times out. Range: 0–10. Default value: 3. |

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. The client sends a **DISCONNECT** message to the server to indicate that it is about to close its TCP/IP connection.

| AT+QMTDISC Disconnect a Client from MQTT Server | |
|--|---|
| Test Command AT+QMTDISC=? | Response +QMTDISC: (range of supported <client_idx> s) OK |
| Write Command AT+QMTDISC=<client_idx> | Response OK +QMTDISC: <client_idx> , <result> |

| | |
|-----------------------|--|
| | If there is any error: ERROR |
| Maximum Response Time | 30 s |
| Characteristic | / |

Parameter

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

NOTE

Disconnecting a client closes the socket.

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 interest in one or more topics. The server delivers messages published to 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> ," topic ",(list of supported <qos>s)[," topic ",(list 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 |
| Characteristic | The command takes effect immediately. The configurations will not be saved. |

Parameter

| | |
|----------------------------|---|
| <client_idx> | Integer type. MQTT client identifier. Range: 0–5. |
| <msgid> | Integer type. Message identifier. Range: 0–65535. 0 indicates no message identifier. It is 0 only when <qos>=0 . |
| <topic> | String type. Topic that the client wants to subscribe to. |
| <qos> | Integer type. 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 Packet sent successfully and ACK received from the server (message is published if <qos>=0 does not require ACK) 1 Packet retransmission 2 Failed to send packet |
| <value> | Integer type. If <result> is 0, it is the level of granted QoS levels. If <result> is 1, it 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: 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 message identifier only exists in messages where the QoS bits in the fixed header indicate QoS level 1 or 2. It must be unique amongst the set of messages in a particular communication direction. It typically increases by 1 from one message to the next.

3.3.7. AT+QMTUNS Unsubscribe from Topics

This command unsubscribes 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>,"topic"["topic" ...] OK |
| Write Command | Response |

| | |
|---|--|
| AT+QMTUNS=<client_idx>,<msgID>,<topic1>[,<topic2>...] | <p>OK</p> <p>+QMTUNS: <client_idx>,<msgID>,<result>[,<value>]</p> <p>If there is any error:</p> <p>ERROR</p> |
| 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. 0 indicates no message identifier. It is 0 only when <qos>=0. |
| <topic> | String type. Topic that the client wants to unsubscribe from. |
| <result> | Integer type. Command execution result. |
| | <p>0 Packet sent successfully and ACK received from the server (message is published if <qos>=0 does not require ACK)</p> <p>1 Packet retransmission</p> <p>2 Failed to send packet</p> |
| <value> | <p>Integer type.</p> <p>If <result> is 0, it is the level of granted QoS levels.</p> <p>If <result> is 1, it is the number of times the packet has been retransmitted.</p> <p>If <result> is 2, it will not be presented.</p> |
| <pkt_timeout> | Integer type. Packet delivery timeout. 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+QMT PUBEX Publish Messages

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

| AT+QMT PUBEX Publish Messages | |
|--------------------------------|---|
| Test Command AT+QMT PUBEX=? | <p>Response</p> <p>+QMT PUBEX: (range of supported <client_idx>s), <msgID>,(range of supported <qos>s),(list of supported <retain>s),"topic","length"</p> |

| | |
|--|---|
| | OK |
| Write Command AT+QMTPUBEX=<client_idx>,<msgID>,<qos>,<retain>,<topic>,<length> | Response > After receiving > , input the data to be sent. If the actual size of data is greater than <length> , extra byte(s) will be deleted. 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. 0 indicates no message identifier. 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. Command execution result. 0 Packet sent successfully and ACK received from server (message is published if <qos>=0 does not require ACK) 1 Packet retransmission 2 Failed to send packet |
| <value> | Integer type. If <result> is 0, it is the level of granted QoS levels. If <result> is 1, it 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. 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. |
|----------------------------|---|

NOTE

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 the server publishes messages to the subscriber, the following URC is returned to notify the host to read the received data reported by the MQTT server:
+QMTRECV: <client_idx>,<msgID>,<topic>[,<payload_length>],<payload>.
For more details about the URC, see **Chapter 4.2**.

3.3.9. AT+QMTRECV Read Messages from Buffer

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

| AT+QMTRECV Read Messages from Buffer | |
|---|--|
| Test Command AT+QMTRECV=? | Response OK |
| Read Command AT+QMTRECV? | Response +QMTRECV: <client_idx>,<store_status> OK If there is no MQTT connection: OK |
| Write Command AT+QMTRECV=<client_idx>[,<recv_id>] | Response [+QMTRECV: <client_idx>,<msgID>,<topic>[,<payload_length>],<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 if a message is stored in the buffer. Since maximum 5 messages can be stored in the buffer, URC can report maximum 5 messages simultaneously. 0 No messages stored in the buffer 1 One or more messages is stored in the buffer |
| <recv_id> | Integer type. Serial number of every received message. Range: 0–4. All buffer data will be read if this parameter is not specified. |
| <msgID> | Integer type. Message identifier of packet. Range: 0–65535. 0 indicates no message identifier. It is 0 only when <qos> =0. |
| <topic> | String type. Topic of messages from the buffer. |
| <payload_len> | Integer type. Payload length. |
| <payload> | String type. Payload relating to topic name. |

4 MQTT URCs

This chapter describes MQTT URCs. Ensure that `<msg_rcv_mode>` takes the default value, and then the following URC [2] and [3] will be reported.

Table 2: MQTT URCs

| SN | URC Format | Description |
|-----|--|--|
| [1] | +QMTSTAT: <code><client_idx>,<err_code></code> | URC to Indicate the reason for state change in MQTT link layer. |
| [2] | +QMTRECV: <code><client_idx>,<msgID>,<topic>[,<payload_len>],<payload></code> | Reported when the client has received packet data from MQTT server. |
| [3] | +QMTRECV: <code><client_idx>,<recv_id></code> | Reported when the message received from MQTT server has been stored in buffer. |
| [4] | +QMTPING: <code><client_idx>,<result></code> | URC to Indicate Keep-Alive timeout in MQTT |

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

The URC that begins with **+QMTSTAT:** is reported when the state of an MQTT link layer changes.

+QMTSTAT URC to Indicate the reason for State Change in MQTT Link Layer

+QMTSTAT: `<client_idx>,<err_code>` When the state of an MQTT link layer is changed, the client will close the MQTT connection and report the URC.

Parameter

| | |
|---------------------------------|---|
| <code><client_idx></code> | Integer type. MQTT client identifier. Range: 0–5. |
| <code><err_code></code> | Integer type. Error code. Refer to the table below for details. |

Table 3: Error Codes of the URC

| <err_code> | Description | Solution |
|------------|--|--|
| 1 | Connection is closed or reset by a peer. | Execute AT+QMTOPEN and reopen MQTT connection. |
| 2 | Sending PINGREQ packet timed out or failed. | First, deactivate and activate the PDP. Then, reopen MQTT connection. |
| 3 | Sending CONNECT packet timed out or failed. | <ol style="list-style-type: none"> 1. Check whether the inputted username and password are correct. 2. Make sure the client ID is not used. 3. Reopen MQTT connection and try to send CONNECT packet to server again. |
| 4 | Receiving CONNACK packet timed out or failed. | <ol style="list-style-type: none"> 1. Check whether the inputted username and password are correct. 2. Make sure the client ID is not used. 3. Reopen MQTT connection and try to send CONNECT packet to server again. |
| 5 | The client sends DISCONNECT packet to sever and the server closes MQTT connection. | This is a normal process. |
| 6 | The client closes MQTT connection due to packet sending failure all the time. | <ol style="list-style-type: none"> 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 closed the MQTT connection. | Try to reconnect. |
| 9-255 | 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 notify the host to read the received MQTT packet data reported by MQTT server.

| +QMTRECV URC to Notify Host to Read MQTT Packet Data | |
|---|--|
| +QMTRECV: <client_idx>,<msgid>,<topic>[,<payload_len>],<payload> | Notify the host to read the received data reported by MQTT server. |
| +QMTRECV: <client_idx>,<recv_id> | 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. 0 indicates no message identifier. It is 0 only when <qos>=0. |
| <topic> | String type. Topic received from MQTT server. |
| <payload_len> | Integer type. Payload length. |
| <payload> | String type. Payload relating to topic name. |
| <recv_id> | Integer type. Serial number of every received message. Range: 0–4. |

4.3. +QMTPING URC to Indicate Keep-Alive Timeout in MQTT

The URC that begins with **+QMTPING:** is reported when server does not receive any messages 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 Keep-Alive Timeout 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 some examples that explain how to use MQTT AT commands.

5.1. Example of MQTT Operation Without SSL

```
//Configure receiving mode.
AT+QMTCFG="recv/mode",0,0,1
OK
//Configure Alibaba device information for Alicloud.
AT+QMTCFG="aliauth",0,"oyjtmPI5a5j","MQTT_TEST","wN9Y6pZSIly7Exa5qVzcmigEGO4kAazZ"
OK
AT+QMTOPEN=?
+QMTOPEN: (0-5),"hostname",(1-65535)

OK
//Open a network for MQTT client.
AT+QMTOPEN=0,"example.com",1883
OK

+QMTOPEN: 0,0 //Opened the MQTT client network successfully.
AT+QMTOPEN?
+QMTOPEN: 0,"example.com",1883

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

OK
//Connect a client to MQTT server.
//If AliCloud is connected, customers can use AT+QMTCFG="aliauth" command to configure the device
information in advance, and <username> and <password> can be omitted.
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 **"topic/example"** 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: 0,2,0,1
```

```
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 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 //For AT+QFUPL, refer to document [2].
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="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,"example.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

```

6 Appendix References

Table 4: Related Documents

| Document Name |
|--|
| [1] Quectel_EG800Q&EG91xQ_Series_SSL_Application_Note |
| [2] Quectel_EG800Q&EG91xQ_Series_FILE_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 |