SIM7028 Series_MQTT(S) _Application Note

LPWA Module

SIMCom Wireless Solutions Limited
SIMCom Headquarters Building, Building 3, No. 289 Linhong Road, Changning District, Shanghai P.R.China
Tel: 86-21-31575100
support@simcom.com
www.simcom.com

Document Title:	SIM7028 Series_MQTT(S)_Application Note
Version:	1.03
Date:	2022.12.09
Status:	Released

GENERAL NOTES

SIMCOM OFFERS THIS INFORMATION AS A SERVICE TO ITS CUSTOMERS, TO SUPPORT APPLICATION AND ENGINEERING EFFORTS THAT USE THE PRODUCTS DESIGNED BY SIMCOM. THE INFORMATION PROVIDED IS BASED UPON REQUIREMENTS SPECIFICALLY PROVIDED TO SIMCOM BY THE CUSTOMERS. SIMCOM HAS NOT UNDERTAKEN ANY INDEPENDENT SEARCH FOR ADDITIONAL RELEVANT INFORMATION, INCLUDING ANY INFORMATION THAT MAY BE IN THE CUSTOMER'S POSSESSION. FURTHERMORE, SYSTEM VALIDATION OF THIS PRODUCT DESIGNED BY SIMCOM WITHIN A LARGER ELECTRONIC SYSTEM REMAINS THE RESPONSIBILITY OF THE CUSTOMER OR THE CUSTOMER'S SYSTEM INTEGRATOR. ALL SPECIFICATIONS SUPPLIED HEREIN ARE SUBJECT TO CHANGE.

COPYRIGHT

SIMCom Wireless Solutions Limited
SIMCom Headquarters Building, Building 3, No. 289 Linhong Road, Changning District, Shanghai P.R.China
Tel: +86 21 31575100
Email: simcom@simcom.com

For more information, please visit:
https://www.simcom.com/download/list-863-en.html

For technical support, or to report documentation errors, please visit:
https://www.simcom.com/ask/or email to: support@simcom.com

About Document

Version History

Revision	Date	Chapter	Description
V1.00	2022.5.12	All	New version
V1.01	2022.05.31	All	Update file
V1.02	2022.07.06	All	Update some description
V1.03	2022.10.24	All	Add details

Scope

This document could be applied to following modules.

Name	Type	Size(mm)	Comments
SIM7028	NB2	17.6*15.7	Band 1/2/3/4/5/8/12/13/14/17/18/19/20/25/26/28/66/70/85

Contents

1Introduction

1.1 Purpose of the document

Based on module AT command manual, this document will introduce MQTTS application process on SIM7028 series of module,developers could understand and develop application quickly and efficiently based on this document.

1.2 Related documents

[1] SIM7028 Series_AT Command Manual

1.3 Conventions and abbreviations

In this document, the GSM engines are referred to as following term:

- ME (Mobile Equipment);
- MS (Mobile Station);
- TA (Terminal Adapter);
- DCE (Data Communication Equipment) or facsimile DCE (FAX modem, FAX board);

In application, controlling device controls the GSM engine by sending AT Command via its serial interface. The controlling device at the other end of the serial line is referred to as following term:

- TE (Terminal Equipment);
- DTE (Data Terminal Equipment) or plainly "the application" which is running on an embedded system;

Other Conventions:

- MQTT(Message Queuing Telemetry Transport);
- SSL(Secure Sockets Layer);
- PDP(Packet Data Protocol);

1.4 AT Command syntax

The "AT" or "at" or "aT" or "At" prefix must be set at the beginning of each Command line. To terminate a Command line enter **<CR>**.

Commands are usually followed by a response that includes. **"<CR><LF><response><CR><LF>"**

Throughout this document, only the responses are presented,<**CR><LF>** are omitted intentionally.

1.4.1 Basic syntax

These AT commands have the format of "**AT<*x*><*n*>**", or "**AT&<x><n>**", where "**<*x*>**"is the Command, and "**<*n*>**"is/are the argument(s) for that Command. An example of this is "**ATE<*n*>**", which tells the DCE whether received characters should be echoed back to the DTE according to the value of "**<*n*>**". "**<*n*>**" is optional and a default will be used if missing.

1.4.2 S Parameter syntax

These AT commands have the format of "**ATS<*n*>=<*m*>**", where "**<*n*>**" is the index of the **S** register to set, and *"***<*m*>***"*is the value to assign to it. "**<*m*>**" is optional; if it is missing, then a default value is assigned.

1.4.3 Extended Syntax

These commands can operate in several modes, as in the following table:

Table 1: Types of AT commands and responses

Test Command **AT+<*x*>=?**	The mobile equipment returns the list of parameters and value ranges set with the corresponding Write Command or by internal processes.
Read Command **AT+<*x*>?**	This command returns the currently set value of the parameter or parameters.
Write Command **AT+<x>=<…>**	This command sets the user-definable parameter values.
Execution Command	The execution command reads non-variable parameters affected by

AT+<x>	internal processes in the GSM engine.

1.4.4 Combining AT commands on the same Command line

You can enter several AT commands on the same line. In this case, you do not need to type the "**AT**" or "**at**" prefix before every command. Instead, you only need type "**AT**" or "**at**" the beginning of the command line. Please note to use a semicolon as the command delimiter after an extended command; in basic syntax or S parameter syntax, the semicolon need not enter, for example:
ATE1Q0S0=1S3=13V1X4;+IFC=0,0;+IPR=115200.

The Command line buffer can accept a maximum of 559 characters(counted from the first command without "AT" or "at" prefix) or 39 AT commands. If the characters entered exceeded this number then none of the Command will executed and TA will return "**ERROR**".

1.4.5 Entering successive AT commands on separate lines

When you need to enter a series of AT commands on separate lines, please Note that you need to wait the final response (for example OK, CME error, CMS error) of last AT Command you entered before you enter the next AT Command.

1.5 AT Command definitions

- <CR>Carriage return character
- <LF>Line feed character
- <..> Parameter name. Angle brackets do not appear on command line
- [..] Option parameter. Square brackets do not appear on the command line.

2MQTT(S) Introduction

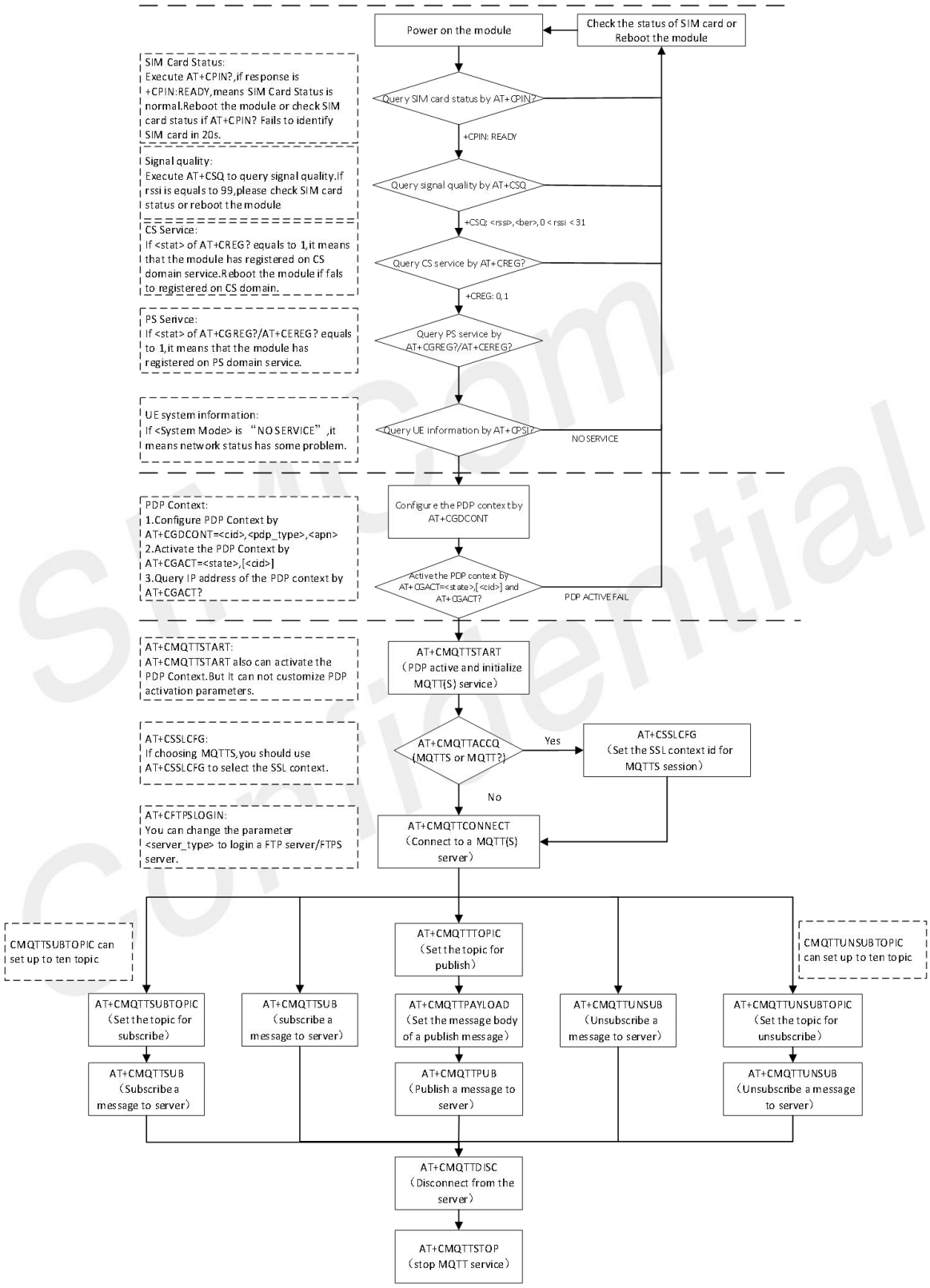
2.1 MQTT(S) Introduction

MQTT (Message Queue Telemetry Transport) is a messaging protocol based on the publish/subscribe paradigm under the ISO standard (ISO/IEC PRF 20922). It works on the TCP/IP protocol suite and is a publish/subscribe messaging protocol designed for remote devices with poor hardware performance and poor network conditions.

The MQTT protocol is a protocol designed for the communication of remote sensors and control devices with limited computing power and working on low-bandwidth, unreliable networks. It has the following main features:

- Use the publish/subscribe message mode to provide one-to-many message publishing and uncouple the application;
- Message transmission for shielding the payload content;
- Provide network connection using TCP/IP;
- There are three types of message publishing service quality:
  - "At most once," message publishing relies entirely on the underlying TCP/IP network. Message loss or duplication can occur. This level can be used in the following situations, environmental sensor data, loss of a read record does not matter, because there will be a second transmission in the near future.
  - "At least once" to ensure that the message arrives, but message duplication may occur.
  - "Only once" to ensure that the message arrives once. This level can be used in situations where repeated or missing messages can result in incorrect results.
- small transmission, low overhead (fixed length of the head is 2 bytes), protocol exchange is minimized to reduce network traffic;
- Use the Last Will and Testament features to notify the parties about the mechanism of client abort.

2.2 The process of Using MQTT(S) AT Command



3AT Commands for MQTT(S)

3.1 Overview

Command	Description
AT+CMQTTSTART	Start MQTT service
AT+CMQTTSTOP	Stop MQTT service
AT+CMQTTACCQ	Acquire a client
AT+CMQTTREL	Release a client
AT+CMQTTSSLCFG	Set the SSL context (only for SSL/TLS MQTT)
AT+CMQTTWILLTOPIC	Input the topic of will message
AT+CMQTTWILLMSG	Input the will message
AT+CMQTTCONNECT	Connect to MQTT server
AT+CMQTTDISC	Disconnect from server
AT+CMQTTTOPIC	Input the topic of publish message
AT+CMQTTPAYLOAD	Input the publish message
AT+CMQTTPUB	Publish a message to server
AT+CMQTTSUB	Subscribe a message to server
AT+CMQTTUNSUBTOPIC	Input the topic of unsubscribe message
AT+CMQTTUNSUB	Unsubscribe a message to server
AT+CMQTTCFG	Configure the MQTT Context

3.2 Detailed Description of Commands

3.2.1 AT+CMQTTSTART Start MQTT service

AT+CMQTTSTART is used to start MQTT service by activating PDP context. You must execute this command before any other MQTT related operations.

AT+CMQTTSTART Start MQTT service	
Test Command	Response

AT+CMQTTSTART=?	**OK**
Execute Command **AT+CMQTTSTART**	Response 1)If start MQTT service successfully: **OK** **+CMQTTSTART: 0** 2)If failed: **OK** **+CMQTTSTART: <errcode>** 3)If MQTT service have started successfully and you executed AT+CMQTTSTART again: **ERROR**
Max Response Time	12000ms
Parameter Saving Mode	-
Reference	

Defined Values

<errcode>	The result code, please refer to Chapter 3.3

Examples

```
AT+CMQTTSTART
OK

+CMQTTSTART: 0
```

NOTE

AT+CMQTTSTART is used to start MQTT service. You must execute this command before any other MQTT related operations.
If you don't execute AT+CMQTTSTART, the Write/Read Command of any other MQTT will return ERROR immediately.

3.2.2 AT+CMQTTSTOP Stop MQTT service

AT+CMQTTSTOP is used to stop MQTT service.

AT+CMQTTSTOP Stop MQTT service	
Test Command **AT+CMQTTSTOP=?**	Response **OK**
Execute Command **AT+CMQTTSTOP**	Response 1)If stop MQTT service successfully: **OK** **+CMQTTSTOP: 0** 2)If failed: **+CMQTTSTOP: <errcode>** **ERROR** 3)If MQTT service have stopped successfully and you executed AT+CMQTTSTOP again: **ERROR**
Max Response Time	12000ms
Parameter Saving Mode	-
Reference	

Defined Values

<errcode>	The result code, please refer to chapter 3.3

Examples

```
AT+CMQTTSTOP
OK

+CMQTTSTOP: 0
```

NOTE

AT+CMQTTSTOP is used to stop MQTT service. You can execute this command after AT+CMQTTDISC and AT+CMQTTREL.

3.2.3 AT+CMQTTACCQ Acquire a client

AT+CMQTTACCQ is used to acquire a MQTT client. It must be called before all commands about MQTT

connect and after AT+CMQTTSTART.

AT+CMQTTACCQ Acquire a client	
Test Command **AT+CMQTTACCQ=?**	Response **+CMQTTACCQ: (0-0),(1-256)[,(0-1)]** **OK**
Read Command **AT+CMQTTACCQ?**	Response **[+CMQTTACCQ: <client_index>,<clientID>,<server_type>** **[……]]** **OK**
Write Command **AT+CMQTTACCQ=<client_index>,<clientID>[<server_type>]**	Response 1)If successfully: **OK** 2)If failed: **+CMQTTACCQ: <client_index>,<err>** **ERROR** 3)If failed: **ERROR**
Parameter Saving Mode	-
Max Response Time	-
Reference	

Defined Values

<client_index>	A numeric parameter that identifies a client. The range of permitted values is 0 to 0.
<clientID>	The UTF-encoded string. It specifies a unique identifier for the client. The string length is from 1 to 256 bytes.
<server_type>	A numeric parameter that identifies the server type. The default value is 0. 0 MQTT server with TCP 1 MQTT server with SSL/TLS
<errcode>	The result code, please refer to chapter 3.3

Examples

```
AT+CMQTTACCQ=0,"a12mmmm",0
OK
AT+CMQTTACCQ?
+CMQTTACCQ: 0,"a12mmmm",0
```

OK **AT+CMQTTACCQ=?**
+CMQTTACCQ: (0-0),(1-256)[,(0-1)] **OK**

3.2.4 AT+CMQTTREL Release a client

AT+CMQTTREL is used to release a MQTT client. It must be called after AT+CMQTTDISC and before AT+CMQTTSTOP.

AT+CMQTTREL Release a client	
Test Command **AT+CMQTTREL=?**	Response **+CMQTTREL: (0-1)** **OK**
Read Command **AT+CMQTTREL?**	Response 1)If successfully: **OK** 2)if MQTT not start **ERROR**
Write Command **AT+CMQTTREL=<client_index>**	Response 1)If successfully: **OK** 2)If failed: **+CMQTTREL: <client_index>,<err>** **ERROR** 3)If failed: **ERROR**
Parameter Saving Mode	-
Max Response Time	-
Reference	

Defined Values

<client_index>	A numeric parameter that identifies a client. The range of permitted values is 0 to 0.
<errcode>	The result code, please refer to chapter 3.3

Examples

```
AT+CMQTTREL=?
+CMQTTREL: (0-1)

OK
AT+CMQTTREL=0
OK
AT+CMQTTREL?
OK
```

3.2.5 AT+CMQTTSSLCFG Set the SSL context (only for SSL/TLS MQTT)

AT+CMQTTSSLCFG is used to set the SSL context which to be used in the SSL connection when it will connect to a SSL/TLS MQTT server. It must be called before AT+CMQTTCONNECT and after AT+CMQTTSTART. The setting will be cleared after AT+CMQTTCONNECT failed or AT+CMQTTDISC.

AT+CMQTTSSLCFG Set the SSL context (only for SSL/TLS MQTT)	
Test Command **AT+CMQTTSSLCFG=?**	Response **+CMQTTSSLCFG: (0,0),(0-1)** **OK**
Read Command **AT+CMQTTSSLCFG?**	Response **[+CMQTTSSLCFG: <session_id>,[<ssl_ctx_index>]** **[……]]** **OK**
Write Command **AT+CMQTTSSLCFG=<session_id>,<ssl_ctx_index>**	Response 1)If successfully: **OK** 2)If failed: **ERROR**
Parameter Saving Mode	-
Max Response Time	-
Reference	-

Defined Values

<session_id>	The session_id to operate. It's from 0 to 0.
<ssl_ctx_index>	The SSL context ID which will be used in the SSL connection. It's from 0 to 1.

Examples

```
AT+CMQTTSSLCFG?
+CMQTTSSLCFG: 0,0

OK
AT+CMQTTSSLCFG=?
+CMQTTSSLCFG: (0,0),(0-1)

OK
AT+CMQTTSSLCFG=0,1
OK
```

3.2.6 AT+CMQTTWILLTOPIC Input the topic of will message

AT+CMQTTWILLTOPIC is used to input the topic of will message.

AT+CMQTTWILLTOPIC Input the topic of will message	
Test Command **AT+CMQTTWILLTOPIC=?**	Response **+CMQTTWILLTOPIC: (0-0),(1-1024)** **OK**
Write Command **AT+CMQTTWILLTOPIC=<client_index>,<req_length>**	Response 1)If successfully: **> <input data here>** **OK** 2)If failed: **+CMQTTWILLTOPIC: <client_index>,<err>** **ERROR** 3)If failed: **ERROR**
Parameter Saving Mode	-
Max Response Time	-
Reference	

Defined Values

<client_index>	A numeric parameter that identifies a client. The range of permitted values is 0 to 0.
<req_length>	The length of input topic. The will topic should be UTF-encoded string. The range is from 1 to 1024 bytes.
<err>	The result code, please refer to chapter 3.3

Examples

```
AT+CMQTTWILLTOPIC=0,10
>

OK
```

3.2.7 AT+CMQTTWILLMSG Input the will message

AT+CMQTTWILLMSG is used to input the message body of will message.

AT+CMQTTWILLMSG Input the will message	
Test Command **AT+CMQTTWILLMSG=?**	Response **+CMQTTWILLMSG: (0-0),(1-1024),(0-2)** **OK**
Write Command **AT+CMQTTWILLMSG=<client_index>,<req_length>,<qos>**	Response 1)If successfully: **>** **<input data here>** **OK** 2)If failed: **+CMQTTWILLMSG: <client_index>,<err>** **ERROR** 3)If failed: **ERROR**
Parameter Saving Mode	-
Max Response Time	-
Reference	

Defined Values

<client_index>	A numeric parameter that identifies a client. The range of permitted values is 0 to 0.
<req_length>	The length of input data. The will message should be UTF-encoded string. The range is from 1 to 1024 bytes.
<qos>	The qos value of the will message. The range is from 0 to 2.

Examples

```
AT+CMQTTWILLMSG=0,6,1
>
OK
```

3.2.8 AT+CMQTTCONNECT Connect to MQTT server

AT+CMQTTCONNECT is used to connect to a MQTT server.

AT+CMQTTCONNECT Connect to MQTT server	
Test Command **AT+CMQTTCONNECT=?**	Response **+CMQTTCONNECT:** **(0-0),(1-128),(1-64800),(0-1)[,<user_name>,<pass_word>]** **OK**
Read Command **AT+CMQTTCONNECT?**	Response **+CMQTTCONNECT:** **[client_index,[,<server_addr>,<keepalive_time>,<clean_session>[,<user_name>[,<pass_word>]]]** **[……]]** **OK**
Write Command **AT+CMQTTCONNECT=<client_index>,<server_addr>,<keepalive_time>,<clean_session>[,<user_name>[,<pass_word>]]**	Response 1)If successfully: **OK** **+CMQTTCONNECT: <client_index>,0** 2)If failed: **OK** **+CMQTTCONNECT: <client_index>,<err>** 3)If failed: **ERROR** **+CMQTTCONNECT: <client_index>,<err>**

	3)If failed: **ERROR** **+CMQTTCONNECT: <client_index>,<err>** 4) If failed: **+CMQTTCONNECT: <client_index>,<err>** **ERROR** 5)If failed: **ERROR**
Parameter Saving Mode	-
Max Response Time	-
Reference	

Defined Values

<client_index>	A numeric parameter that identifies a client. The range of permitted values is 0 to 0.
<server_addr>	The string that described the server address and port. The range of the string length is 1 to 128 bytes. The string should be like this "tcp://116.247.119.165:5141", must begin with "tcp://". If the <server_addr> not include the port, the default port is 1883.
<keepalive_time>	The time interval between two messages received from a client. The client will send a keep-alive packet when there is no message sent to server after song long time. The range is from 1s to 64800s (18 hours).
<clean_session>	The clean session flag. The value range is from 0 to 1, and default value is 0. 0 the server must store the subscriptions of the client after it disconnected. This includes continuing to store QoS 1 and QoS 2 messages for the subscribed topics so that they can be delivered when the client reconnects. The server must also maintain the state of in-flight messages being delivered at the point the connection is lost. This information must be kept until the client reconnects. 1 the server must discard any previously maintained information about the client and treat the connection as "clean". The server must also discard any state when the client disconnects.
<user_name>	The user name identifies the name of the user which can be used for authentication when connecting to server. The string length is from 1 to 256 bytes.
<pass_word>	The password corresponding to the user which can be used for authentication when connecting to server. The string length is from 1 to 256 bytes.
<err>	The result code: 0 is success. Other values are failure. Please refer to chapter 3.3.

Examples

```
AT+CMQTTCONNECT=0,"tcp://120.27.2.154:1883",20,1
OK

+CMQTTCONNECT: 0,0
AT+CMQTTCONNECT?
+CMQTTCONNECT: 0,"tcp://120.27.2.154:1883",20,1

OK
```

NOTE

AT+CMQTTCONNECT is used to connect to a MQTT server.
If you don't set the SSL context by AT+CMQTTSSLCFG before connecting a SSL/TLS MQTT server by AT+CMQTTCONNECT, it will use the <client_index> (the 1st parameter of AT+CMQTTCONNNECT)SSL context when connecting to the server.

3.2.9 AT+CMQTTDISC Disconnect from server

AT+CMQTTDISC is used to disconnect from the server.

AT+CMQTTDISC Disconnect from server	
Test Command **AT+CMQTTDISC=?**	Response: **+CMQTTDISC: (0-0),(0, 60-180)** **OK**
Read Command **AT+CMQTTDISC?**	Response: **[+CMQTTDISC: 0,<disc_state>** **[……]]** **OK**
Write Command **AT+CMQTTDISC=<client_index>,<timeout>**	Response 1)If disconnect successfully: **+CMQTTDISC: <client_index>,0** **OK** 2)If disconnect successfully: **OK**

	+CMQTTDISC: <client_index>,0 3)If failed: **OK** **+CMQTTDISC: <client_index>,<err>** 4)If failed: **ERROR** 5)If failed: **+CMQTTDISC: <client_index>,<err>** **ERROR**
Parameter Saving Mode	-
Max Response Time	-
Reference	

Defined Values

<client_index>	A numeric parameter that identifies a client. The range of permitted values is 0 to 0.
<timeout>	The timeout value for disconnection. The unit is second. The range is 60s to 180s. The default value is 0s (not set the timeout value).
<disc_state>	1 disconnection 0 connection
<err>	The result code: 0 is success. Other values are failure. Please refer to chapter 3.3.

Examples

```
AT+CMQTTDISC=0,120
OK

+CMQTTDISC: 0,0
```

3.2.10 AT+CMQTTTOPIC Input the topic of publish message

AT+CMQTTTOPIC is used to input the topic of a publish message.

AT+CMQTTTOPIC Input the topic of publish message	
Test Command **AT+CMQTTTOPIC=?**	Response **+CMQTTTOPIC: (0-0),(1-1024)**

	OK
Write Command **AT+CMQTTTOPIC=<client_index>,<req_length>**	Response 1)If successfully: **> <input data here> OK** 2)If failed: **+CMQTTTOPIC: <client_index>,<err>** **ERROR** 3)If failed: **ERROR**
Parameter Saving Mode	-
Max Response Time	-
Reference	

Defined Values

<client_index>	A numeric parameter that identifies a client. The range of permitted values is 0 to 0.
<req_length>	The length of input topic data. The publish message topic should be UTF-encoded string. The range is from 1 to 1024 bytes.
<err>	The result code: 0 is success. Other values are failure. Please refer to chapter 3.3.

Examples

```
AT+CMQTTTOPIC=0,9
>

OK
```

NOTE

The topic will be clean after execute AT+CMQTTPUB.

3.2.11 AT+CMQTTPAYLOAD Input the publish message

AT+CMQTTPAYLOAD is used to input the message body of a publish message.

AT+CMQTTPAYLOAD	Input the publish message
Test Command **AT+CMQTTPAYLOAD=?**	Response **+CMQTTPAYLOAD: (0-0),(1-10240)** **OK**
Write Command **AT+CMQTTPAYLOAD=<client_index>,<req_length>**	Response 1)If successfully: **>** **<input data here>** **OK** 2)If failed: **+CMQTTPAYLOAD: <client_index>,<err>** **ERROR** 3)If failed: **ERROR**
Parameter Saving Mode	-
Max Response Time	-
Reference	

Defined Values

<client_index>	A numeric parameter that identifies a client. The range of permitted values is 0 to 0.
<req_length>	The length of input message data. The publish message should be UTF-encoded string. The range is from 1 to 10240 bytes.
<err>	The result code: 0 is success. Other values are failure. Please refer to chapter 3.3.

Examples

```
AT+CMQTTPAYLOAD=0,6
>

OK
```

NOTE

The topic will be clean after execute AT+CMQTTPUB.

3.2.12 AT+CMQTTPUB Publish a message to server

AT+CMQTTPUB is used to publish a message to MQTT server.

AT+CMQTTPUB Publish a message to server	
Test Command **AT+CMQTTPUB=?**	Response **+CMQTTPUB: (0-0),(0-2),(60-180),(0-1),(0-1)** **OK**
Write Command **AT+CMQTTPUB=<client_index>,<qos>,<pub_timeout>[,<ratained>[,<dup>]]**	Response 1)If successfully: **OK** **+CMQTTPUB: <client_index>,0** 2)If failed: **OK** **+CMQTTPUB: <client_index>,<err>** 3)If failed: **+CMQTTPUB: <client_index>,<err>** **ERROR** 4)If failed: **ERROR**
Parameter Saving Mode	-
Max Response Time	-
Reference	

Defined Values

<client_index>	A numeric parameter that identifies a client. The range of permitted values is 0 to 0.
<qos>	The publish message's qos. The range is from 0 to 2. 0 at most once 1 at least once 2 exactly once
<pub_timeout>	The publishing timeout interval value. Since the client publish a message to server, it will report failed if the client receive no response from server after the timeout value seconds. The range is from 60s to 180s.
<ratained>	The retain flag of the publish message. The value is 0 or 1. The default value is 0.

	When a client sends a PUBLISH to a server, if the retain flag is set to 1, the server should hold on to the message after it has been delivered to the current subscribers.
<dup>	The dup flag to the message. The value is 0 or 1. The default value is 0. The flag is set when the client or server attempts to re-deliver a message.
<err>	The result code: 0 is success. Other values are failure. Please refer to chapter 3.3.

Examples

```
AT+CMQTTPUB=0,1,60
OK

+CMQTTPUB: 0,0
```

NOTE

The topic and payload will be clean after execute AT+CMQTTPUB.

3.2.13 AT+CMQTTSUB Subscribe a message to server

AT+CMQTTSUB is used to subscribe a message to MQTT server.

AT+CMQTTSUB Subscribe a message to server	
Test Command **AT+CMQTTSUB=?**	Response **+CMQTTSUB: (0-0),(1-1024),(0-2),(0-1)** **OK**
Read Command **AT+CMQTTSUB?**	Response **+CMQTTSUB:** **[<topic>]** **OK**
Write Command /* subcribe one topic*/ **AT+CMQTTSUB=<client_index>,<reqLength>,<qos>[,<dup>]**	Response 1)If successfully: **>** **<input data here>** **OK** **+CMQTTSUB: <client_index>,0** 2)If failed:

	OK **+CMQTTSUB: <client_index>,<err>** 3)If failed: **+CMQTTSUB: <client_index>,<err>** **ERROR** 4)If failed: **ERROR**
Parameter Saving Mode	-
Max Response Time	-
Reference	-

Defined Values

<client_index>	A numeric parameter that identifies a client. The range of permitted values is 0 to 0.
<req_length>	The length of input topic data. The message topic should be UTF-encoded string. The range is from 1 to 1024 bytes.
<qos>	The publish message's qos. The range is from 0 to 2. 0 at most once 1 at least once 2 exactly once
<dup>	The dup flag to the message. The value is 0 or 1. The default value is 0. The flag is set when the client or server attempts to re-deliver a message.
<err>	The result code: 0 is success. Other values are failure. Please refer to chapter 3.3.
<topic>	Topics to which you have subscribed

Examples

```
AT+CMQTTSUB=0,9,1
>

OK

+CMQTTSUB: 0,0
AT+CMQTTSUB=0,1
OK

+CMQTTSUB: 0,0
```

NOTE

The topic will be clean after execute AT+CMQTTSUB.

3.2.14 AT+CMQTTUNSUB Unsubscribe a message to server

AT+CMQTTUNSUB is used to unsubscribe a message to MQTT server.

AT+CMQTTUNSUB Unsubscribe a message to server	
Test Command **AT+CMQTTUNSUB=?**	Response **+CMQTTUNSUB: (0-0),(1-1024),(0-2),(0-1)** **OK**
Write Command /* unsubscribe one topic*/ **AT+CMQTTUNSUB=<client_index>,<reqLength>,<dup>**	Response 1)If successfully: **> <input data here> OK** **+CMQTTUNSUB: <client_index>,0** 2)If failed: **OK** **+CMQTTUNSUB: <client_index>,<err>** 3)If failed: **+CMQTTUNSUB: <client_index>,<err>** **ERROR** 4)If failed: **ERROR**
Parameter Saving Mode	-
Max Response Time	-
Reference	-

Defined Values

<client_index>	A numeric parameter that identifies a client. The range of permitted values is 0 to 0.
<req_length>	The length of input topic data. The message topic should be UTF-encoded string. The range is from 1 to 1024 bytes.
<dup>	The dup flag to the message. The value is 0 or 1. The default value is

	0. The flag is set when the client or server attempts to re-deliver a message.
<err>	The result code: 0 is success. Other values are failure. Please refer to chapter 3.3.

Examples

```
AT+CMQTTUNSUBTOPIC=0,9
>

OK
AT+CMQTTUNSUB=0,1
OK

+CMQTTUNSUB: 0,0
```

NOTE

The topic will be clean after execute AT+CMQTTUNSUB.

3.2.15 AT+CMQTTCFG Configure the MQTT Context

AT+CMQTTCFG is used to configure the MQTT context. It must be called before AT+CMQTTCONNECT and after AT+CMQTTACCQ. The setting will be cleared after AT+CMQTTREL.

AT+CMQTTCFG Configure the MQTT Context	
Test Command **AT+CMQTTCFG=?**	Response **+CMQTTCFG: "checkUTF8",(0-0),(0-1)** **+CMQTTCFG: "optimeout ",(0-0)** **OK**
Read Command **AT+CMQTTCFG?**	Response **[+CMQTTCFG: 0,<checkUTF8_flag>,<optimeout_val>** **[……]]** **OK**

Write Command /*Configure the check UTF8 flag of the specified MQTT client context*/ **AT+CMQTTCFG="checkUTF8",<index>,<checkUTF8_flag>**	Response 1)If successfully: **OK** 2)If failed: **ERROR**
Write Command /*Configure the max timeout interval of the send or receive data operation */ **AT+CMQTTCFG="optimeout",<index>,<optimeout_val>**	Response 1)If successfully: **OK** 2)If failed: **ERROR**
Parameter Saving Mode	-
Max Response Time	-
Reference	-

Defined Values

<checkUTF8_flag>	The flag to indicate whether to check the string is UTF8 coding or not, the default value is 1. 0 Not check UTF8 coding. 1 Check UTF8 coding.
<optimeout_val>	The max timeout interval of sending or receiving data operation. The range is from 20 seconds to 120 seconds, the default value is 120 seconds.

Examples

```
AT+CMQTTCFG?
+CMQTTCFG: 0,1,120

OK
AT+CMQTTCFG="optimeout",0,24
OK
AT+CMQTTCFG="checkUTF8",0,0
OK
AT+CMQTTCFG?
+CMQTTCFG: 0,0,24
+CMQTTCFG: 1,1,120

OK
```

NOTE

The setting will be cleared after AT+CMQTTREL.

3.3 Command Result Codes

3.3.1 Description of <err>

<err>	Description
0	operation succeeded
1	failed
2	bad UTF-8 string
3	sock connect fail
4	sock create fail
5	sock close fail
6	message receive fail
7	network open fail
8	network close fail
9	network not opened
10	client index error
11	no connection
12	invalid parameter
13	not supported operation
14	client is busy
15	require connection fail
16	sock sending fail
17	timeout
18	topic is empty
19	client is used
20	client not acquired
21	client not released
22	length out of range
23	network is opened
24	packet fail
25	DNS error
26	socket is closed by server

27	connection refused: unaccepted protocol version
28	connection refused: identifier rejected
29	connection refused: server unavailable
30	connection refused: bad user name or password
31	connection refused: not authorized
32	handshake fail
33	not set certificate
34	Open session failed
35	Disconnect from server failed

3.3.2 Unsolicited Result Codes

URC	Description
+CMQTTCONNLOST: <client_index>,<cause>	When client disconnect passively, URC "+CMQTTCONNLOST" will be reported, then user need to connect MQTT server again.
+CMQTTRXSTART: <client_index>,<topic_total_len>,<payload_total_len> **+CMQTTRXTOPIC: <client_index>,<sub_topic_len>** **<sub_topic>** /*for long topic, split to multiple packets to report*/ **[<CR><LF>+CMQTTRXTOPIC: <client_index>,<sub_topic_len>** **<sub_topic>]** **+CMQTTRXPAYLOAD: <client_index>,<sub_payload_len>** **<sub_payload>** /*for long payload, split to multiple packets to report*/ **[+CMQTTRXPAYLOAD: <client_index>,<sub_payload_len>** **<sub_payload>]** **+CMQTTRXEND: <client_index>**	If a client subscribes to one or more topics, any message published to those topics are sent by the server to the client. The following URC is used for transmitting the message published from server to client. 1)+CMQTTRXSTART: <client_index>,<topic_total_len>,<payload_total_len>\r\n At the beginning of receiving published message, the module will report this to user, and indicate client index with <client_index>, the topic total length with <topic_total_len> and the payload total length with <payload_total_len> after "\r\n". 2)+CMQTTRXTOPIC: <client_index>,<sub_topic_len>\r\n <sub_topic> After the command "+CMQTTRXSTART" received, the module will report the second message to user, and indicate client index with <client_index>, the topic packet length with <sub_topic_len>

	and the topic content with <sub_topic> after "\r\n". For long topic, it will be split to multiple packets to report and the command "+CMQTTRXTOPIC" will be send more than once with the rest of topic content. The sum of <sub_topic_len> is equal to <topic_total_len>. 3)+CMQTTRXPAYLOAD: <client_index>,<sub_payload_len>\r\n<sub_payload> After the command "+CMQTTRXTOPIC" received, the module will send third message to user, and indicate client index with <client_index>, the payload packet length with <sub_payload_len> and the payload content with <sub_payload> after "\r\n". For long payload, the same as "+CMQTTRXTOPIC". 4)+CMQTTRXEND: <client_index> At last, the module will send fourth message to user and indicate the topic and payload have been transmitted completely.

Defined Values

<client_index>	A numeric parameter that identifies a client. The range of permitted values is 0 to 1.
<cause>	The cause of disconnection. 1 Socket is closed passively. 2 Socket is reset. 3 Network is closed.
<topic_total_len>	The length of message topic received from MQTT server. The range is from 1 to 1024 bytes.
<payload_total_len>	The length of message body received from MQTT server. The range is from 1 to 10240 bytes.
<sub_topic_len>	The sub topic packet length, The sum of <sub_topic_len> is equal to <topic_total_len>.
<sub_topic>	The sub topic content.
<sub_payload_len>	The sub message body packet length, The sum of <sub_payload_len> is equal to <payload_total_len>.
<sub_payload>	The sub message body content.

4MQTT(S) Examples

Before all MQTT(S) related operations, we should ensure the following:
Ensure network is available:

AT+CSQ	
+CSQ: 23,0	
OK	
AT+CGREG?	Need to check network registeration state until get 1(home register) or 5(roaming register)
+CGREG: 0,1	
OK	
AT+CGDCONT=1,"IP","apn"	Customer need to set IP type(IP or IPV6) and correct apn name
OK	

4.1 Connect to MQTT broker without SSL/TLS

Following commands shows how to communicate with an MQTT broker.

AT+CMQTTSTART	//Start MQTT service, activate PDP context
OK	
+CMQTTSTART: 0	
AT+CMQTTACCQ=0,"client test0"	//Acquire one client which will connect to a MQTT server without SSL/TLS
OK	
AT+CMQTTWILLTOPIC=0,10	//Set the will topic for the CONNECT message
>	
OK	
AT+CMQTTWILLMSG=0,6,1	//Set the will message for the CONNECT message
>	
OK	
AT+CMQTTCONNECT=0,"tcp://test.mosquitto.	//Connect to an MQTT broker

```
org:1883",60,1
OK

+CMQTTCONNECT: 0,0
AT+CMQTTSUB=0,9,1                    //Subscribe one topic from the broker
>

OK

+CMQTTSUB: 0,0
AT+CMQTTTOPIC=0,9                    //Set the topic for the PUBLISH message
>

OK
AT+CMQTTPAYLOAD=0,60                 //Set the payload for the PUBLISH message
>

OK
AT+CMQTTPUB=0,1,60                   //Publish a message
OK

+CMQTTPUB: 0,0
+CMQTTRXSTART: 0,9,60                //Receive publish message from broker
+CMQTTRXTOPIC: 0,9
simcommsg
+CMQTTRXPAYLOAD: 0,60
012345678901234567890123456789012345678
901234567890123456789
+CMQTTRXEND: 0
AT+CMQTTSUB=0                        //Subscribe a message
OK

+CMQTTSUB: 0,0
AT+CMQTTUNSUB=0,9,0                  //Unsubscribe one topic from the broker
>

OK

+CMQTTUNSUB: 0,0
AT+CMQTTDISC=0,120                   //Disconnect from broker
OK

+CMQTTDISC: 0,0
AT+CMQTTREL=0                        //Release the client
OK
```

```
AT+CMQTTSTOP                                    //Stop MQTT Service
OK

+CMQTTSTOP: 0
```

4.2 Connect to SSL/TLS MQTT broker(not verify server)

Following commands shows how to access to an MQTT broker without verifying the server. It needs to configure the authentication mode to 0, and then it will connect to the server successfully.

```
AT+CMQTTSTART                                   //Start MQTT service, activate PDP context
OK

+CMQTTSTART: 0
AT+CMQTTACCQ=0,"client test0",1                 //Acquire one client which will connect to a
OK                                              SSL/TLS MQTT broker
AT+CMQTTWILLTOPIC=0,10                          //Set the will topic for the CONNECT message
>

OK
AT+CMQTTWILLMSG=0,6,1                           //Set the will message for the CONNECT message
>

OK
AT+CMQTTCONNECT=0,"tcp://test.mosquitto.        //Connect to a MQTT broker
org:8883",60,1
OK

+CMQTTCONNECT: 0,0
AT+CMQTTTOPIC=0,13                              //Set the topic for the PUBLISH message
>

OK
AT+CMQTTPAYLOAD=0,60                            //Set the payload for the PUBLISH message
>

OK
AT+CMQTTPUB=0,1,60                              //Publish a message
OK

+CMQTTPUB: 0,0
AT+CMQTTSUB=0                                   //Subscribe a message
```

```
OK

+CMQTTSUB: 0,0
AT+CMQTTSUB=0,9,1                    //Subscribe one topic from the broker
>

OK

+CMQTTSUB: 0,0
AT+CMQTTUNSUB=0,9,0                  //Unsubscribe one topic from the broker
>

OK

+CMQTTUNSUB: 0,0
AT+CMQTTDISC=0,120                   //Disconnect from broker
OK

+CMQTTDISC: 0,0
AT+CMQTTREL=0                        //Release the client
OK
AT+CMQTTSTOP                         //Stop MQTT Service
OK

+CMQTTSTOP: 0
```

4.3 Access to SSL/TLS MQTT broker(only verify the server)

Following commands shows how to access to a SSL/TLS MQTT broker with verifying the server. It needs to configure the authentication mode to 1 and the right server root CA, and then it will connect to the server successfully.

```
                                     //Set the SSL version of the first SSL context
AT+CSSLCFG="sslversion",0,4
OK
AT+CSSLCFG="authmode",0,1            //Set the authentication mode(verify server) of the
OK                                   first SSL context
AT+CSSLCFG="cacert",0,"server_ca.pem" //Set the server root CA of the first SSL context
OK
AT+CMQTTSTART                        //Start MQTT service, activate PDP context
OK
```

```
+CMQTTSTART: 0
AT+CMQTTACCQ=0,"client test0",1          //Acquire one client which will connect to a
OK                                       SSL/TLS MQTT server
AT+CMQTTSSLCFG=0,0                       //Set the first SSL context to be used in the SSL
OK                                       connection
AT+CMQTTWILLTOPIC=0,10                   //Set the will topic for the CONNECT message
>

OK
AT+CMQTTWILLMSG=0,6,1                    //Set the will message for the CONNECT message
>

OK
AT+CMQTTCONNECT=0,"tcp://mqtts_server:p  //Connect to a MQTT broker, input the right broker
ort",60,1                                and port
OK

+CMQTTCONNECT: 0,0
AT+CMQTTTOPIC=0,13                       //Set the topic for the PUBLISH message
>

OK
AT+CMQTTPAYLOAD=0,60                     //Set the payload for the PUBLISH message
>

OK
AT+CMQTTPUB=0,1,60                       //Publish a message
OK

+CMQTTPUB: 0,0
AT+CMQTTSUB=0                            //Subscribe a message
OK

+CMQTTSUB: 0,0
AT+CMQTTSUB=0,9,1                        //Subscribe one topic from the broker
>

OK

+CMQTTSUB: 0,0
AT+CMQTTUNSUB=0,9,0                      //Unsubscribe one topic from the broker
>

OK

+CMQTTUNSUB: 0,0
```

```
AT+CMQTTDISC=0,120                    //Disconnect from broker
OK

+CMQTTDISC: 0,0
AT+CMQTTREL=0                         //Release the client
OK
AT+CMQTTSTOP                          //Stop MQTT Service
OK

+CMQTTSTOP: 0
```

4.4 Access to SSL/TLS MQTT broker(verify server and client)

Following commands shows how to access to a SSL/TLS MQTT broker with verifying the server and client. It needs to configure the authentication mode to 2, the right server root CA, the right client certificate and key, and then it will connect to the server successfully.

```
AT+CSSLCFG="sslversion",0,4           //Set the SSL version of the first SSL context
OK
AT+CSSLCFG="authmode",0,2             //Set the authentication mode(verify server and
OK                                    client) of the first SSL context
AT+CSSLCFG="cacert",0,"ca_cert.pem"   //Set the server root CA of the first SSL context
OK
AT+CSSLCFG="clientcert",0,"cert.pem"  //Set the client certificate of the first SSL context
OK
AT+CSSLCFG="clientkey",0,"key_cert.pem"  //Set the client key of the first SSL context
OK
AT+CMQTTSTART                         //Start MQTT service, activate PDP context
OK

+CMQTTSTART: 0
AT+CMQTTACCQ=0,"client test0",1       //Acquire one client which will connect to a
OK                                    SSL/TLS MQTT broker
AT+CMQTTSSLCFG=0,0                    //Set the first SSL context to be used in the SSL
OK                                    connection
AT+CMQTTWILLTOPIC=0,10                //Set the will topic for the CONNECT message
>

OK
AT+CMQTTWILLMSG=0,6,1                 //Set the will message for the CONNECT message
>
```

```
OK
AT+CMQTTCONNECT=0,"tcp://hooleeping.com:8883",60,1          //Connect to a MQTT broker
OK

+CMQTTCONNECT: 0,0
AT+CMQTTTOPIC=0,13                                          //Set the topic for the PUBLISH message
>

OK
AT+CMQTTPAYLOAD=0,60                                        //Set the payload for the PUBLISH message
>

OK
AT+CMQTTPUB=0,1,60                                          //Publish a message
OK

+CMQTTPUB: 0,0
AT+CMQTTSUB=0                                               //Subscribe a message
OK

+CMQTTSUB: 0,0
AT+CMQTTSUB=0,9,1                                           //Subscribe one topic from the broker
>

OK

+CMQTTSUB: 0,0
AT+CMQTTUNSUB=0,9,0                                         //Unsubscribe one topic from the broker
>

OK

+CMQTTUNSUB: 0,0
AT+CMQTTDISC=0,120                                          //Disconnect from broker
OK

+CMQTTDISC: 0,0
AT+CMQTTREL=0                                               //Release the client
OK
AT+CMQTTSTOP                                                //Stop MQTT Service
OK

+CMQTTSTOP: 0
```