SIM7672X & SIM7652X Series_MQTT(S)_ Application Note

LTE 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:	SIM7672X & SIM7652X Series_MQTT(S)_Application Note
Version:	1.00
Date:	2023.05.22
Status:	Released

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/technical_files.html

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

About Document

Version History

Revision	Date	Owner	Description
V1.00	2023.5.22		New version

Scope

Based on module AT command manual, this document will introduce MQTT(S) application process. Developers could understand and develop application quickly and efficiently based on this document. This document applies to SIM7672X Series, SIM7652X Series.

Contents

1 Introduction

1.1 Purpose of the document

Based on module AT command manual, this document will introduce MQTT(S) application process. Developers could understand and develop application quickly and efficiently based on this document.

1.2 Related documents

[1] SIM7672X & SIM7652X Series_AT Command Manual

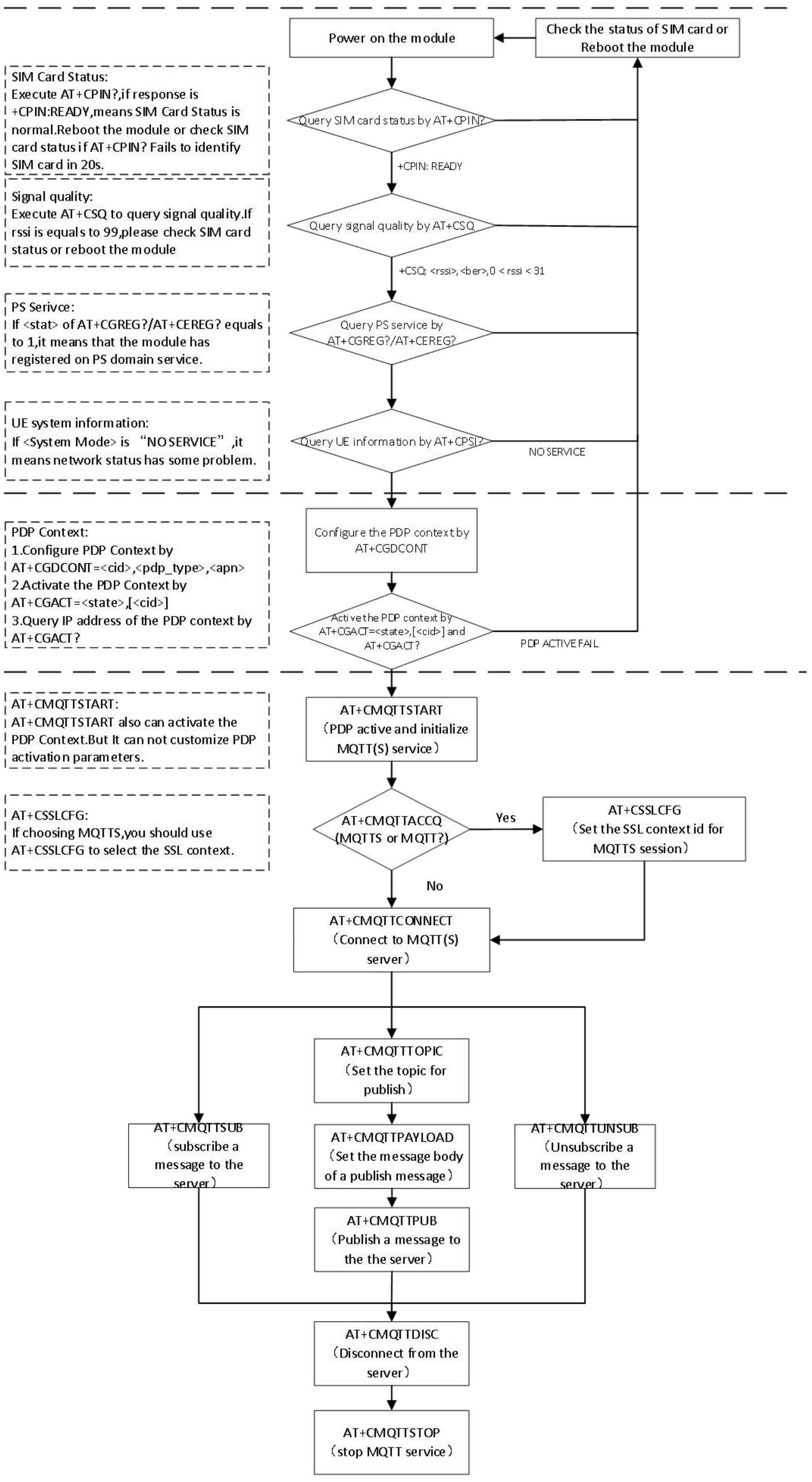
1.3 Conventions and abbreviations

In this document, the engines are referred as following term:
ME (Mobile Equipment);
MS (Mobile Station);
TA (Terminal Adapter);
DCE (Data Communication Equipment);

In application, controlling device controls the engine by sending AT Command via its serial interface. The controlling device at the other end of the serial line is referred 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 The process of Using MQTT(S) AT Command


Power on the module
Check the status of SIM card or Reboot the module
SIM Card Status: Execute AT+CPIN?,if response is +CPIN:READY,means SIM Card Status is normal.Reboot the module or check SIM card status if AT+CPIN? Fails to identify SIM card in 20s.
Query SIM card status by AT+CPIN?
+CPIN: READY
Signal quality: Execute AT+CSQ to query signal quality.If rssi is equals to 99,please check SIM card status or reboot the module
Query signal quality by AT+CSQ
+CSQ: <rssi>,<ber>,0 < rssi < 31
PS Serivce: If <stat> of AT+CGREG?/AT+CEREG? equals to 1,it means that the module has registered on PS domain service.
Query PS service by AT+CGREG?/AT+CEREG?
UE system information: If <System Mode> is "NO SERVICE",it means network status has some problem.
Query UE information by AT+CPSI?
NO SERVICE
PDP Context: 1.Configure PDP Context by AT+CGDCONT=<cid>,<pdp_type>,<apn> 2.Activate the PDP Context by AT+CGACT=<state>,[<cid>] 3.Query IP address of the PDP context by AT+CGACT?
Configure the PDP context by AT+CGDCONT
Active the PDP context by AT+CGACT=<state>,[<cid>] and AT+CGACT?
PDP ACTIVE FAIL
AT+CMQTTSTART: AT+CMQTTSTART also can activate the PDP Context.But It can not customize PDP activation parameters.
AT+CMQTTSTART（PDP active and initialize MQTT(S) service）
AT+CSSLCFG: If choosing MQTTS,you should use AT+CSSLCFG to select the SSL context.
AT+CMQTTACCQ (MQTTS or MQTT?)
Yes
AT+CSSLCFG（Set the SSL context id for MQTTS session）
No
AT+CMQTTCONNECT（Connect to MQTT(S) server）
AT+CMQTTSUB（subscribe a message to the server）
AT+CMQTTTOPIC（Set the topic for publish）
AT+CMQTTPAYLOAD（Set the message body of a publish message）
AT+CMQTTPUB（Publish a message to the the server）
AT+CMQTTUNSUB（Unsubscribe a message to the server）
AT+CMQTTDISC（Disconnect from the server）
AT+CMQTTSTOP（stop MQTT service）

1.5 Error Handling

For more details, please refer to SIM7672X & SIM7652X Series_AT Command Manual.

2 AT Commands for MQTT(S)

2.1 Overview of AT Commands for MQTT(S)

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+CMQTTUNSUB	Unsubscribe a message to server
AT+CMQTTCFG	Configure the MQTT Context

2.2 Detailed Description of AT Commands for MQTT(S)

2.2.1 AT+CMQTTSTART Start MQTT service

AT+CMQTTSTART is used to start MQTT service by activating PDP context. This command must be executed before any other MQTT related operations.

AT+CMQTTSTART Start MQTT service	
Execute Command AT+CMQTTSTART	Response 1)If start MQTT service successfully: **OK** **+CMQTTSTART: 0** 2)If failed:

	OK **+CMQTTSTART: <err>** 3)If MQTT service have started successfully and you executed AT+CMQTTSTART again: **ERROR**
Max Response Time	12000ms
Parameter Saving Mode	-
Reference	

Defined Values

<err>	The result code, please refer to <err> list.

Examples

```
AT+CMQTTSTART
OK

+CMQTTSTART: 0
```

NOTE

AT+CMQTTSTART is used to start MQTT service by activating PDP context. This command must be executed before any other MQTT related operations.
If ***AT+CMQTTSTART*** is not executed, the Write/Read Command of any other MQTT will return ERROR immediately.

2.2.2 AT+CMQTTSTOP Stop MQTT service

AT+CMQTTSTOP is used to stop MQTT service.

AT+CMQTTSTOP Stop MQTT service	
Execute Command AT+CMQTTSTOP	Response 1)If stop MQTT service successfully: **OK** **+CMQTTSTOP: 0** 2)If failed:

	+CMQTTSTOP: <err> **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

<err>	The result code, please refer to <err> list.

Examples

```
AT+CMQTTSTOP
OK

+CMQTTSTOP: 0
```

NOTE

AT+CMQTTSTOP is used to stop MQTT service. This command can be executed after ***AT+CMQTTDISC*** and ***AT+CMQTTREL***.

2.2.3 AT+CMQTTACCQ Acquire a client

AT+CMQTTACCQ is used to acquire 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: (0-1),(1-128)[,(0-1)]** **OK**
Read Command AT+CMQTTACCQ?	Response **+CMQTTACCQ: <client_index>,<clientID>,<server_type>** **+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 1.
<clientID>	The UTF-encoded string. It specifies a unique identifier for the client. The string length is from 1 to 128 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
<err>	The result code, please refer to <err> list.

Examples

```
AT+CMQTTACCQ=0,"a12mmmm",0
OK
AT+CMQTTACCQ?
+CMQTTACCQ: 0,"a12mmmm",0
+CMQTTACCQ: 1,"",0

OK
AT+CMQTTACCQ=?
+CMQTTACCQ: (0-1),(1-128)[,(0-1)]

OK
```

2.2.4 AT+CMQTTREL Release a client

AT+CMQTTREL is used to release 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 1.
<err>	The result code, please refer to <err> list.

Examples

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

OK
AT+CMQTTREL=0
```

```
OK
AT+CMQTTREL?
OK
```

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

AT+CMQTTSSLCFG is used to set the SSL context which will be used in the SSL connection when it connects 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: (0,1),(0-9)** **OK**
Read Command **AT+CMQTTSSLCFG?**	Response **+CMQTTSSLCFG: <session_id>,[<ssl_ctx_index>]** **+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 of the operate. The range of permitted values is 0 to 1.
<ssl_ctx_index>	The SSL context ID will be used in the SSL connection. The range is 0-9. Refer to the <ssl_ctx_index> of ***AT+CSSLCFG***.

Examples

```
AT+CMQTTSSLCFG?
+CMQTTSSLCFG: 0,0
+CMQTTSSLCFG: 1,0
```

```
OK
AT+CMQTTSSLCFG=?
+CMQTTSSLCFG: (0,1),(0-9)

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

2.2.6 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: (0-1),(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 1.
<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 <err> list.

Examples

```
AT+CMQTTWILLTOPIC=0,10
>

OK
```

2.2.7 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: (0-1),(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 1.
<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 of permitted values is 0 to 2.

Examples

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

2.2.8 AT+CMQTTCONNECT Connect to MQTT server

AT+CMQTTCONNECT is used to connect to MQTT server.

AT+CMQTTCONNECT Connect to MQTT server	
Test Command **AT+CMQTTCONNECT=?**	Response **+CMQTTCONNECT:** **(0-1),(9-256),(1-64800),(0-1)[,<user_name>,<pass_word>]** **OK**
Read Command **AT+CMQTTCONNECT?**	Response **+CMQTTCONNECT:** **0[,<server_addr>,<keepalive_time>,<clean_session>[,<user_name>[,<pass_word>]]]** **+CMQTTCONNECT:** **1[,<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>** 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 1.
<server_addr>	The string that described the server address and port. The range of the string length is 9 to 256 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 the server for a long time. The range is from 1s to 64800s (18 hours).
<clean_session>	The clean session flag. The range of permitted values is 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 status 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 the 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 the server. The string length is from 1 to 256 bytes.
<err>	The result code: 0 is success. Other values are failure. Please refer to <err> list.

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
+CMQTTCONNECT: 1
```

```
OK
```

NOTE

AT+CMQTTCONNECT is used to connect to 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+CMQTTCONNECT***)SSL context when connecting to the server.

2.2.9 AT+CMQTTDISC Disconnect from the server

AT+CMQTTDISC is used to disconnect from the server.

AT+CMQTTDISC Disconnect from server	
Test Command **AT+CMQTTDISC=?**	Response: **+CMQTTDISC: (0-1),(0, 60-180)** **OK**
Read Command **AT+CMQTTDISC?**	Response: **+CMQTTDISC: 0,<disc_state>** **+CMQTTDISC: 1,<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 1.
<timeout>	The timeout value for disconnection. The unit is second. The range is 60s to 180s. The default value is 60s (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 <err> list.

Examples

```
AT+CMQTTDISC=0,120
OK

+CMQTTDISC: 0,0
```

2.2.10 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: (0-1),(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 1.
<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 <err> list.

Examples

```
AT+CMQTTTOPIC=0,9
>

OK
```

NOTE

The topic will be cleaned after executing ***AT+CMQTTPUB***.

2.2.11 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: (0-1),(1-4096)** **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 1.
<req_length>	The length of input message data. The publish message should be UTF-encoded string. The range is from 1 to 4096 bytes.
<err>	The result code: 0 is success. Other values are failure. Please refer to <err> list.

Examples

```
AT+CMQTTPAYLOAD=0,6
>

OK
```

NOTE

The topic will be cleaned after executing ***AT+CMQTTPUB***.

2.2.12 AT+CMQTTPUB Publish a message to the 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: (0-1),(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 1.
<qos>	The publish message's <qos>. The range of permitted values is 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 the server, it will report failed if the client receive no response from the 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 of 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 <err> list.

Examples

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

+CMQTTPUB: 0,0
```

NOTE

The topic and payload will be cleaned after executing ***AT+CMQTTPUB***.

2.2.13 AT+CMQTTSUB Subscribe a message to the 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: (0-1),(1-1024),(0-2),(0-1)** **OK**
Read Command **AT+CMQTTSUB?**	Response **+CMQTTSUB:** **[<topic>]** **OK**
Write Command /* subscribe one topics */ **AT+CMQTTSUB=<client_index>[,<dup>]**	Response 1)If successfully: **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**
Write Command /* subscribe 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 1.
<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 of permitted values is 0 to 2. 0 at most once 1 at least once 2 exactly once
<dup>	The <dup> flag of 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 <err> list.
<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 cleaned after executing ***AT+CMQTTSUB***.

2.2.14 AT+CMQTTUNSUB Unsubscribe a message to the 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: (0-1),(1-1024),(0-1)** **OK**
Write Command /*unsubscribe one topics*/ **AT+CMQTTUNSUB=<client_index>,<dup>**	Response 1)If successfully: **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**
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 1.
<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 of 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 <err> list.

Examples

```
AT+CMQTTUNSUB=0,1
OK

+CMQTTUNSUB: 0,0
```

NOTE

The topic will be cleaned after executing ***AT+CMQTTUNSUB***.

2.2.15 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",(0-1),(0-1)** **+CMQTTCFG: "optimeout ",(0-1),(20-120)**

	+CMQTTCFG: "version",(0-1),(3-4) **OK**
Read Command **AT+CMQTTCFG?**	Response **+CMQTTCFG: 0,<checkUTF8_flag>,<optimeout_val>** **+CMQTTCFG: 1,<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.
+CMQTTCFG: "version",(0-1),(3-4)	(0-1): A numeric parameter that identifies a client. The range of permitted values is 0 to 1. (3-4): Version of MQTT. 3: MQTT 3.1. The default value is 3. 4: MQTT 3.1.1.

Examples

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

```
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 MQTT(S)Examples

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

```
AT+CSQ
+CSQ: 23,0

OK
AT+CPSI?
+CPSI:
LTE,Online,460-00,0x333C,39589680,308,EUT
RAN-BAND3,1350,5,0,0,54,0,22

OK
AT+CGACT?
+CGACT: 1,1

OK
```

3.1 Access to MQTT server without SSL/TLS

Following commands show how to communicate with MQTT server.

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

+CMQTTSTART: 0
AT+CMQTTACCQ=0,"client test0"              // Acquire one client which will connect to MQTT
OK                                         server without SSL/TLS
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 MQTT server
```

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

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

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 the server
+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 server
>

OK

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

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

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

+CMQTTSTOP: 0
```

3.2 Access to SSL/TLS MQTT server (not verify server)

Following commands show how to access to MQTT server without verifying the server. It needs to configure the authentication mode to 0(not verify server), 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 SSL/TLS
OK                                              MQTT server
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.o       // Connect to MQTT server
rg: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 server
>

OK

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

OK

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

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

+CMQTTSTOP: 0
```

3.3 Access to SSL/TLS MQTT server (verify server only)

Following commands shows how to access to SSL/TLS MQTT server with verifying the server. It needs to configure the authentication mode to 1(verify server only) and the root CA of the server, and then it will connect to the server successfully.

```
AT+CSSLCFG="sslversion",0,4                // Set SSL version for the first SSL context
OK
AT+CSSLCFG="authmode",0,1                  // Set the authentication mode(verify server) for
OK                                         the first SSL context
AT+CSSLCFG="cacert",0,"server_ca.pem"      // Set the server root CA for 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 SSL/TLS
OK                                           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 MQTT server, input the right server
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 server
>

OK

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

OK

+CMQTTUNSUB: 0,0
AT+CMQTTDISC=0,120                           // Disconnect from server
```

```
OK

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

+CMQTTSTOP: 0
```

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

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

```
AT+CSSLCFG="sslversion",0,4            // Set the SSL version for the first SSL context
OK
AT+CSSLCFG="authmode",0,2              // Set the authentication mode(verify server and
OK                                     client) for the first SSL context
AT+CSSLCFG="cacert",0,"ca_cert.pem"    // Set the server root CA for the first SSL context
OK
AT+CSSLCFG="clientcert",0,"cert.pem"   // Set the client certificate for the first SSL context
OK
AT+CSSLCFG="clientkey",0,"key_cert.pem"  // Set the client key for 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 SSL/TLS
OK                                     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://hooleeping.co       // Connect to MQTT server
m: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 server
>

OK

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

OK

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

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

+CMQTTSTOP: 0
```

3.5 Access to MQTT server without checking UTF8 coding

Following commands shows how to communicate with MQTT server without checking UTF8 coding.

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

+CMQTTSTART: 0
AT+CMQTTACCQ=0,"client test0"              // Acquire one client which will connect to MQTT
OK                                         server without SSL/TLS
AT+CMQTTCFG="checkUTF8",0,0                // Configure not checking UTF8 coding
OK
AT+CMQTTCONNECT=0,"tcp://198.41.30.241:1   // Connect to MQTT server
883",60,1
OK

+CMQTTCONNECT: 0,0
AT+CMQTTSUB=0,9,1                          // Subscribe one topic which is not UTF8 coding
>                                          string.

OK                                         // The data can be input in hexadecimal format.

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

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

+CMQTTPUB: 0,0
+CMQTTRXSTART: 0,9,0                       // Receive publish message from the server
+CMQTTRXTOPIC: 0,9
鼢鼢鼢鼢?

+CMQTTRXEND: 0
AT+CMQTTDISC=0,120                         // Disconnect from the server
OK

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

```
+CMQTTSTOP: 0
```

4 Appendix

4.1 Summary of <err>

<err>	Meaning
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

4.2 Unsolicited Result Codes

URC	Description
+CMQTTCONNLOST: <client_index>,<cause>	When the client disconnect passively, URC "+CMQTTCONNLOST" will be reported, then user need to connect to MQTT server again.
+CMQTTNONET	When the network becomes no network, the module will report this URC. If received this message, please restart the MQTT service by AT+CMQTTSTART.
+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 the server to the 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.