SIM7500_SIM7600_SIM7800 Series_MQTT_AT Command Manual_V1.00

LTE Module

Shanghai SIMCom Wireless Solutions Ltd.
Building A, SIM Technology Building, No.633, Jinzhong Road
Changning District 200335
Tel:86-21-31575100/31575200
support@simcom.com
www.simcom.com

Document Title:	SIM7500_SIM7600_SIM7800 Series_MQTT_AT Command Manual
Version:	1.00
Date:	2018-10-12
Status:	Release
Document ID:	SIM7500_SIM7600_SIM7800 Series_MQTT_AT Command Manual_V1.00

SIMCom Confidential

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

This document contains proprietary technical information which is the property of SIMCom Limited., copying of this document and giving it to others and the using or communication of the contents thereof, are forbidden without express authority. Offenders are liable to the payment of damages. All rights reserved in the event of grant of a patent or the registration of a utility model or design. All specification supplied herein are subject to change without notice at any time.

Copyright © Shanghai SIMCom Wireless Solutions Ltd. 2018

Version History

Version	Date	Chapter	What is new
V1.00	2018-09-28		New version

SIMCom Confidential

Contents

SIMCom Confidential

This document is a reference guide to all the AT commands defined for MQTT. Through these MQTT AT commands, you can communicate with a MQTT server.

1 Introduction

1.1 The SSL Context Management AT Commands (for SSL/TLS MQTT)

Step 1: Configure SSL version by AT+CSSLCFG="sslversion",<ssl_ctx_index>,<sslversion>.

Step 2: Configure SSL authentication mode by AT+CSSLCFG="authmode",<ssl_ctx_index>, <authmode>.

Step 3: Configure the flag of ignore local time by

AT+CSSLCFG="ignorlocaltime",<ssl_ctx_index>,<ignoreltime>.

Step 4: Configure the max time in SSL negotiation stage by

AT+CSSLCFG="negotiatetime",<ssl_ctx_index>,<negotiatetime>.

Step 5: Configure the server root CA by AT+CSSLCFG="cacert",<ssl_ctx_index>,<ca_file>.

Step 6: Configure the client certificate by AT+CSSLCFG="clientcert",<ssl_ctx_index>,<clientcert_file>.

Step 7: Configure the client key by AT+CSSLCFG="clientkey",<ssl_ctx_index>,<clientkey_file>.

Step 8: Download the certificate into the module by AT+CCERTDOWN.

Step 9: Delete the certificate from the module by AT+CCERTDELE.

Step 10: List the certificates by AT+CCERTLIST.

1.2 The process of Using MQTT AT Commands

Step 1: Ensure GPRS network is available before performing SSL related operations.

Step 2: Configure the parameter of PDP context by AT+CGDCONT.

Step 3: Activate the PDP context to start MQTT service by AT+CMQTTSTART.

Step 4: Acquire a client by AT+CMQTTACCQ.

Step 5: Configure SSL context by AT+CSSLCFG (if connect to SSL/TLS MQTT server).

Step 6: Set the SSL context used in SSL connection by AT+CMQTTSSLCFG (if connect to SSL/TLS MQTT server).

Step 7: Connect to MQTT server by AT+CMQTTCONNECT.

Step 8: Subscribe to message by AT+CMQTTSUB.

SIMCom Confidential

Step 9: Unsubscribe to message by AT+CMQTTUNSUB.

Step 10: Input the topic of a publish message by AT+CMQTTTOPIC.

Step 11: Input the payload of a publish message by AT+CMQTTPAYLOAD.

Step 12: Publish message by AT+CMQTTPUB.

Step 13: Disconnect from the server by AT+CMQTTDISC.

Step 13: Release the client by AT+CMQTTREL.

Step 14: Deactivate the PDP context to stop MQTT service by AT+CMQTTSTOP.

2 Description of AT Command

2.1 SSL Context Management AT (only for SSL/TLS MQTT)

2.1.1 AT+CSSLCFG Configure the SSL Context

AT+CSSLCFG Configure the SSL Context	
Test Command **AT+CSSLCFG=?**	Response **+CSSLCFG: "sslversion",(0-9),(0-4)** **+CSSLCFG: "authmode",(0-9),(0-3)** **+CSSLCFG: "ignorelocaltime",(0-9),(0,1)** **+CSSLCFG: "negotiatetime",(0-9),(10-300)** **+CSSLCFG: "cacert",(0-9),(5-128)** **+CSSLCFG: "clientcert",(0-9),(5-128)** **+CSSLCFG: "clientkey",(0-9),(5-128)** **OK**
Read Command **AT+CSSLCFG?**	Response **+CSSLCFG: 0,<sslversion>,<authmode>,<ignoreltime>,<negotiatetime>,<ca_file>,<clientcert_file>,<clientkey_file>** **+CSSLCFG: 1,<sslversion>,<authmode>,<ignoreltime>,<negotiatetime>,<ca_file>,<clientcert_file>,<clientkey_file>** **+CSSLCFG: 2,<sslversion>,<authmode>,<ignoreltime>,<negotiatetime>,<ca_file>,<clientcert_file>,<clientkey_file>** **+CSSLCFG: 3,<sslversion>,<authmode>,<ignoreltime>,<negotiatetime>,<**

Confidential

	ca_file>,<clientcert_file>,<clientkey_file> **+CSSLCFG:** **4,<sslversion>,<authmode>,<ignoreltime>,<negotiatetime>,<ca_file>,<clientcert_file>,<clientkey_file>** **+CSSLCFG:** **5,<sslversion>,<authmode>,<ignoreltime>,<negotiatetime>,<ca_file>,<clientcert_file>,<clientkey_file>** **+CSSLCFG:** **6,<sslversion>,<authmode>,<ignoreltime>,<negotiatetime>,<ca_file>,<clientcert_file>,<clientkey_file>** **+CSSLCFG:** **7,<sslversion>,<authmode>,<ignoreltime>,<negotiatetime>,<ca_file>,<clientcert_file>,<clientkey_file>** **+CSSLCFG:** **8,<sslversion>,<authmode>,<ignoreltime>,<negotiatetime>,<ca_file>,<clientcert_file>,<clientkey_file>** **+CSSLCFG:** **9,<sslversion>,<authmode>,<ignoreltime>,<negotiatetime>,<ca_file>,<clientcert_file>,<clientkey_file>** **OK**
Write Command /*Query the configuration of the specified SSL context*/ **AT+CSSLCFG=<ssl_ctx_index>**	Response **+CSSLCFG:** **<ssl_ctxindex>,<sslversion>,<authmode>,<ignoreltime>,<negotiatetime>,<ca_file>,<clientcert_file>,<clientkey_file>** **OK**
Write Command /*Configure the version of the specified SSL context*/ **AT+CSSLCFG="sslversion",<ssl_ctx_index>,<sslversion>**	Response a)If successfully: **OK** b)If failed: **ERROR**
Write Command /*Configure the authentication mode of the specified SSL context*/ **AT+CSSLCFG="authmode",<ssl_ctx_index>,<authmode>**	Response a)If successfully: **OK** b)If failed: **ERROR**
Write Command /*Configure the ignore local time flag of the specified SSL context*/ **AT+CSSLCFG="ignorelocaltime",<ssl_ctx_index>,<ignoreltime>**	Response a)If successfully: **OK** b)If failed: **ERROR**
Write Command	Response

Confidential

/*Configure the negotiate timeout value of the specified SSL context*/ **AT+CSSLCFG="negotiatetime",<ssl_ctx_index>,<negotiatetime>**	a)If successfully: **OK** b)If failed: **ERROR**
Write Command /*Configure the server root CA of the specified SSL context*/ **AT+CSSLCFG="cacert",<ssl_ctx_index>,<ca_file>**	Response a)If successfully: **OK** b)If failed: **ERROR**
Write Command /*Configure the client certificate of the specified SSL context*/ **AT+CSSLCFG="clientcert",<ssl_ctx_index>,<clientcert_file>**	Response a)If successfully: **OK** b)If failed: **ERROR**
Write Command /*Configure the client key of the specified SSL context*/ **AT+CSSLCFG="clientkey",<ssl_ctx_index>,<clientkey_file>**	Response a)If successfully: **OK** b)If failed: **ERROR**

Defined Values

<ssl_ctx_index>	The SSL context ID. The range is 0-9.
<sslversion>	The SSL version, the default value is 4. 0 – SSL3.0 1 – TLS1.0 2 – TLS1.1 3 – TLS1.2 4 – All The configured version should be support by server. So you should use the default value if you can't confirm the version which the server supported.
<authmode>	The authentication mode, the default value is 0. 0 – no authentication. 1 –server authentication. It needs the root CA of the server. 2 –server and client authentication. It needs the root CA of the server, the cert and key of the client. 3–client authentication and no server authentication. It needs the cert and key of the client.
<ignoreltime>	The flag to indicate how to deal with expired certificate, the default value is 1. 0 – care about time check for certification. 1 – ignore time check for certification

SIMCom Confidential

	When set the value to 0, it need to set the right current date and time by AT+CCLK when need SSL certification.
<negotiatetime>	The timeout value used in SSL negotiate stage. The range is 10-300 seconds. The default value is 300.
<ca_file>	The root CA file name of SSL context. The file name must have type like ".pem" or ".der".The length of filename is from 5 to 128 bytes. If the filename contains non-ASCII characters, the file path parameter should contain a prefix of {non-ascii} and the quotation mark (The string in the quotation mark should be hexadecimal of the filename's UTF8 code). There are two ways to download certificate files to module: 1. By AT+CCERTDOWN. 2. By FTPS or HTTPS commands. Please refer to: SIM7500_SIM7600_SIM7800 Series_FTPS_AT Command Manual and SIM7500_SIM7600_SIM7800 Series_HTTP_AT Command Manual
<clientcert_file>	The client cert file name of SSL context. The file name must have type like ".pem" or ".der".The length of filename is from 5 to 128 bytes. If the filename contains non-ASCII characters, the file path parameter should contain a prefix of {non-ascii} and the quotation mark (The string in the quotation mark should be hexadecimal of the filename's UTF8 code). There are two ways to download certificate files to module: 1. By AT+CCERTDOWN. 2. By FTPS or HTTPS commands. Please refer to: SIM7500_SIM7600_SIM7800 Series_FTPS_AT Command Manual and SIM7500_SIM7600_SIM7800 Series_HTTP_AT Command Manual
<clientkey_file>	The client key file name of SSL context. The file name must have type like ".pem" or ".der".The length of filename is from 5 to 128 bytes. If the filename contains non-ASCII characters, the file path parameter should contain a prefix of {non-ascii} and the quotation mark (The string in the quotation mark should be hexadecimal of the filename's UTF8 code). There are two ways to download certificate files to module: 1. By AT+CCERTDOWN.

SIMCom Confidential

	2. By FTPS or HTTPS commands. Please refer to: SIM7500_SIM7600_SIM7800 Series_FTPS_AT Command Manual and SIM7500_SIM7600_SIM7800 Series_HTTP_AT Command Manual

2.1.2 AT+CCERTDOWN Download certificate into the module

AT+CCERTDOWN Download certificate into the module	
Test Command **AT+CCERTDOWN=?**	Response **+CCERTDOWN: (5-128),(1-10240)** **OK**
Write Command **AT+CCERTDOWN=<filename>,<len>**	Response a)If it can be download: > **<input data here>** b)If failed: **ERROR**

Defined Values

<filename>	The name of the certificate/key file. The file name must have type like ".pem" or ".der". The length of filename is from 5 to 128 bytes. If the filename contains non-ASCII characters, the file path parameter should contain a prefix of {non-ascii} and the quotation mark (The string in the quotation mark should be hexadecimal of the filename's UTF8 code). For example: If you want to download a file with name "中华.pem", you should convert the "中华.pem" to UTF8 coding (中华.pem), then input the hexadecimal (262378344532443B262378353334453B2E70656D) of UTF8 coding.
<len>	The length of the file data to send. The range is from 1 to 10240 bytes.

2.1.3 AT+CCERTLIST List certificates

AT+CCERTLIST List certificates	
Execute Command	Response

SIMCom Confidential

AT+CCERTLIST	**[+CCERTLIST:<file_name>** **[+CCERTLIST:<file_name>]** **…** **<CR><LF>]** **OK**

Defined Values

<filename>	The certificate/key files which has been downloaded to the module. If the filename contains non-ASCII characters, it will show the non-ASCII characters as UTF8 code.

2.1.4 AT+CCERTDELE Delete certificates

AT+CCERTDELE Delete certificate from the module	
Write Command **AT+CCERTDELE=<filename>**	Response a)If delete successfully: **OK** b)If failed: **ERROR**

Defined Values

<filename>	The name of the certificate/key file. The file name must have type like ".pem" or ".der". The length of filename is from 5 to 128 bytes. If the filename contains non-ASCII characters, the file path parameter should contain a prefix of {non-ascii} and the quotation mark (The string in the quotation mark should be hexadecimal of the filename's UTF8 code). For example: If you want to download a file with name "中华.pem", you should convert the "中华.pem" to UTF8 coding (中华.pem), then input the hexadecimal (262378344532443B262378353334453B2E70656D) of UTF8 coding.

SIMCom Confidential

2.2 MQTT Services AT

2.2.1 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	
Execute Command **AT+CMQTTSTART**	Response a)If start MQTT service successfully: **OK** **+CMQTTSTART: <err>** b)If start MQTTservice successfully: **+CMQTTSTART: <err>** **OK** c)If failed: **ERROR** d)If failed: **ERROR** **+CMQTTSTART: <err>**
Maximum Response Time	120000ms

Defined Values

<err>	The result code, please refer to chapter 2.3.1

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 a)If stop MQTT service successfully: **+CMQTTSTOP: <err>** **OK** b)If stop MQTT service successfully:

SIMCom Confidential

	OK **+CMQTTSTOP: <err>** c)If failed: **ERROR**

Defined Values

<err>	The result code, please refer to chapter 2.3.1

2.2.3 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: (0-1),(1-23),(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 a)If successfully: **OK** b)If failed: **+CMQTTACCQ: <client_index>,<err>** **ERROR** c) If failed: **ERROR**

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 23 bytes.
<server_type>	A numeric parameter that identifies the server type. The default value is 0.

Confidential

	0 - MQTT server with TCP 1-MQTT server with SSL/TLS
<err>	The result code, please refer to chapter 2.3.1

2.2.4 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 **OK**
Write Command **AT+CMQTTREL=<client_index>**	Response a)If successfully: **OK** b)If failed: **+CMQTTREL: <client_index>,<err>** **ERROR** c) If failed: **ERROR**

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 chapter 2.3.1.

2.2.5 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 the CMQTTCONNECT operation is finished.

NOTE: If you don't set the SSL context by this command before connecting to server by AT+CMQTTCONNECT, the CMQTTCONNECT operation will use the SSL context as same as index <session_id> (the 1st parameter of AT+ CMQTTCONNECT) when connecting to the server.

SIMCom Confidential

AT+CCHSSLCFG Set the SSL context	
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 a)If successfully: **OK** b)If failed: **ERROR**

Defined Values

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

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 will topic	
Test Command **AT+CMQTTWILLTOPIC=?**	Response **+CMQTTWILLTOPIC: (0-1),(1-1024)** **OK**
Write Command **AT+CMQTTWILLTOPIC=<client_index>,<req_length>**	Response a)If successfully: **>** **<input data here>** **OK** b)If failed: **+CMQTTWILLTOPIC: <client_index>,<err>** **ERROR** c)If failed:

m Confidential

	ERROR

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 chapter 2.3.1

2.2.7 AT+CMQTTWILLMSG Input the will message

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

AT+CMQTTWILLTOPIC Input the will message	
Test Command **AT+CMQTTWILLMSG=?**	Response **+CMQTTWILLMSG: (0-1),(1-10240),(0-2)** **OK**
Write Command **AT+CMQTTWILLMSG=<client_index>,<req_length>,<qos>**	Response a)If successfully: **> <input data here>** **OK** b)If failed: **+CMQTTWILLMSG: <client_index>,<err>** **ERROR** c)If failed: **ERROR**

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 10240 bytes.
<qos>	The qos value of the will message. The range is from 0 to 2.
<err>	The result code: 0 is success. Other values are failure. Please refer to chapter 2.3.1

SIMCom Confidential

2.2.8 AT+CMQTTCONNECT Connect to MQTT server

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

AT+CMQTTCONNECT Connect to MQTT server	
Test Command **AT+CMQTTCONNECT=?**	Response **+CMQTTCONNECT: (0-1),(9-256),(60-64800),(0-1)** **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 a)If successfully: **OK** **+CMQTTCONNECT: <client_index>,0** b)If failed: **OK** **+CMQTTCONNECT: <client_index>,<err>** c)If failed: **+CMQTTCONNECT: <client_index>,<err>** **ERROR** d)If failed: **ERROR**

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

Confidential

	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 60s 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.
<password>	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 2.3.1.

2.2.9 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: (0-1),(0, 60-180)** **OK**
Read Command **AT+CMQTTDISC?**	Response: **+CMQTTDISC: 0,<disc_state>** **+CMQTTDISC: 1,<disc_state>**

SIMCom Confidential

	OK
Write Command **AT+CMQTTDISC=<client_index>,<timeout>**	Response a)If connect successfully： **+CMQTTDISC: <client_index>,0** **OK** b)If connect successfully： **OK** **+CMQTTDISC: <client_index>,0** c) If failed： **OK** **+CMQTTDISC: <client_index>,<err>** d)If failed: **ERROR** e)If failed: **+CMQTTDISC: <client_index>,<err>** **ERROR**

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 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 2.3.1.

2.2.10 AT+CMQTTTOPIC Input the topic of publish message

AT+CMQTTTOPIC is used to input the topic of a publish message.
NOTE: The topic will be clean after execute AT+CMQTTPUB.

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

SIMCom Confidential

AT+CMQTTTOPIC=?	**OK**
Write Command **AT+CMQTTTOPIC=<client_index>,<req_length>**	Response a)If successfully: **> <input data here> OK** b)If failed: **+CMQTTTOPIC: <client_index>,<err>** **ERROR** c)If failed: **ERROR**

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 chapter 2.3.1.

2.2.11 AT+CMQTTPAYLOAD Input the publish message

AT+CMQTTPAYLOAD is used to input the message body of a publish message.
NOTE: The payload will be clean after execute AT+CMQTTPUB.

AT+CMQTTPAYLOAD Input the publish message body	
Test Command **AT+CMQTTPAYLOAD=?**	Response **+CMQTTPAYLOAD: (0-1),(1-10240)** **OK**
Write Command **AT+CMQTTPAYLOAD=<client_index>,<req_length>**	Response a)If successfully: **> <input data here> OK** b)If failed: **+CMQTTPAYLOAD: <client_index>,<err>** **ERROR**

Confidential

	c)If failed: **ERROR**

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 10240 bytes.
<err>	The result code: 0 is success. Other values are failure. Please refer to chapter 2.3.1.

2.2.12 AT+CMQTTPUB Publish a message to server

AT+CMQTTPUB is used to publish a message to MQTT server.
NOTE: The topic and payload will be clean after execute AT+CMQTTPUB.

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 a)If successfully: **OK** **+CMQTTPUB: <client_index>,0** b)If failed: **OK** **+CMQTTPUB: <client_index>,<err>** c)If failed: **+CMQTTPUB: <client_index>,<err>** **ERROR** d)If failed: **ERROR**

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 is from 0 to 2.

Confidential

	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 2.3.1.

2.2.13 AT+CMQTTSUBTOPIC Input the topic of subscribe message

AT+CMQTTSUBTOPIC is used to input the topic of a subscribe message.
NOTE: The topic will be clean after execute AT+CMQTTSUB.

AT+CMQTTSUBTOPIC Input a subscribe message topic	
Test Command **AT+CMQTTSUBTOPIC=?**	Response **+CMQTTSUBTOPIC: (0-1),(1-1024)** **OK**
Write Command **AT+CMQTTSUBTOPIC=<client_index>,<req_length>,<qos>**	Response a)If successfully: > **<input data here>** **OK** b)If failed: **+CMQTTSUBTOPIC: <client_index>,<err>** **ERROR** c)If failed: **ERROR**

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.
<qos>	The publish message's qos. The range is from 0 to 2. 0 – at most once 1 – at least once 2 – exactly once
<err>	The result code: 0 is success. Other values are failure. Please refer to chapter 2.3.1.

2.2.14 AT+CMQTTSUB Subscribe a message to server

AT+CMQTTSUB is used to subscribe a message to MQTT server.
NOTE: The topic will be clean after execute AT+CMQTTSUB.

AT+CMQTTSUB Subscribe a message to server	
Test Command **AT+CMQTTSUB=?**	Response **+CMQTTSUB (0-1),(0-1024),(0-2),(0-1)** **OK**
Write Command /* subscribe one or more topics which input by AT+CMQTTSUBTOPIC*/ **AT+CMQTTSUB=<client_index>[,<dup>]**	Response a)If successfully: **OK** **+CMQTTSUB: <client_index>,0** b)If failed: **OK** **+CMQTTSUB: <client_index>,<err>** c)If failed: **+CMQTTSUB: <client_index>,<err>** **ERROR** d)If failed: **ERROR**
Write Command /* subcribe one topic*/ **AT+CMQTTSUB=<client_index>,<reqLength>,<qos>[,<dup>]**	Response a)If successfully: > **<input data here>** **OK**

Confidential

	+CMQTTSUB: <client_index>,<err> b)If failed: **OK** **+CMQTTSUB: <client_index>,<err>** c)If failed: **+CMQTTSUB: <client_index>,<err>** **ERROR** d)If failed: **ERROR**

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 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 2.3.1.

2.2.15 AT+CMQTTUNSUBTOPIC Input the topic of unsubscribe message

AT+CMQTTUNSUBTOPIC is used to input the topic of a unsubscribe message.
NOTE: The topic will be clean after execute AT+CMQTTUNSUB.

AT+CMQTTUNSUBTOPIC Input a unsubscribe message topic	
Test Command **AT+CMQTTUNSUBTOPIC=?**	Response **+CMQTTUNSUBTOPIC: (0-1),(1-1024)** **OK**
Write Command **AT+CMQTTUNSUBTOPIC=<client_ind**	Response a)If successfully:

SIMCom Confidential

ex>,<req_length>	**>** **<input data here>** **OK** b)If failed: **+CMQTTUNSUBTOPIC: <client_index>,<err>** **ERROR** c)If failed: **ERROR**

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 chapter 2.3.1.

2.2.16 AT+CMQTTUNSUB Unsubscribe a message to server

AT+CMQTTUNSUB is used to unsubscribe a message to MQTT server.
NOTE: The topic will be clean after execute AT+CMQTTUNSUB.

AT+CMQTTSUB Subscribe a message to server	
Test Command **AT+CMQTTUNSUB=?**	Response **+CMQTTUNSUB (0-1),(0-1024),(0-1)** **OK**
Write Command /* unsubscribe one or more topics which input by AT+CMQTTUNSUBTOPIC*/ **AT+CMQTTUNSUB=<client_index>,<dup>**	Response a)If successfully: **OK** **+CMQTTUNSUB: <client_index>,0** **OK** **+CMQTTUNSUB: <client_index>,<err>** b)If failed: **OK** **+CMQTTUNSUB: <client_index>,<err>** c)If failed:

Confidential

	+CMQTTUNSUB: <client_index>,<err> **ERROR** d)If failed: **ERROR**
Write Command /* unsubscribe one topic*/ **AT+CMQTTUNSUB=<client_index>,<reqLength>,<dup>**	Response a)If successfully: > **<input data here>** b)If failed: **OK** **+CMQTTUNSUB: <client_index>,<err>** c)If failed: **+CMQTTUNSUB: <client_index>,<err>** **ERROR** d)If failed: **ERROR**

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 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 2.3.1.

2.2.17 AT+CMQTTCFG Configure the MQTT Context

AT+CSSLCFG Configure the SSL Context	
Test Command **AT+CMQTTCFG=?**	Response **+CMQTTCFG: "checkUTF8",(0-1),(0-1)** **OK**

SIMCom Confidential

Read Command **AT+CMQTTCFG?**	Response **+CMQTTCFG: 0,<checkUTF8_flag>** **+CMQTTCFG: 1, <checkUTF8_flag>** **OK**
Write Command /*Configure the check UTF8 flag of the specified MQTT client context*/ **AT+CMQTTCFG="checkUTF8",<index>,<checkUTF8_flag>**	Response a)If successfully: **OK** b)If failed: **ERROR**

Defined Values

<index>	The MQTT client index. The range is 0-1.
<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.

2.3 Command result codes and unsolicited codes

2.3.1 Command result <err> codes

Result codes	
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

SIMCom Confidential

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

2.3.2 Unsolicited result codes

2.3.2.1 Disconnected by server

Unsolicited codes	
+CMQTTCONNLOST: <client_index>,<cause>	When client disconnect passively, URC "+CMQTTCONNLOST" will be reported, then user need to connect MQTT server again.

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.

SIMCom Confidential

	1 – Socket is closed passively. 2 – Socket is reset. 3 – Network is closed.

2.3.2.2 Receive publish message from MQTT server

If a client subscribes to one or more topics, any message published to those topics are sent by the server to the client.

Unsolicited codes	
+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>** 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>. 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

Confidential

	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.
<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.

3 Example

Before all MQTT related operations, we should ensure the following:

a) ensure GPRS network is available:

```
AT+CSQ

+CSQ: 23,0


OK

AT+CREG?

+CREG: 0,1


OK

AT+CGREG?

+CGREG: 0,1


OK
```

b) PDP context Enable:

Confidential

// Specify the parameter value of the PDP context corresponding to cid
AT+CGSOCKCONT=1,"IP","CMNET"

OK

AT+CGPADDR

+CGPADDR: 1,10.49.14.68 //ensure the first PDP context get a IP address

+CGPADDR: 4,0.0.0.0

OK

Note: usually CSOCKAUTH and CSOCKSETPN parameter are kept default if not care about.

3.1 Access to MQTT server not SSL/TLS

Following commands shows how to communicate with a MQTT server.

// start MQTT service, activate PDP context

AT+CMQTTSTART

OK

+CMQTTSTART: 0
// Acquire one client which will connect to a MQTT server not SSL/TLS
AT+CMQTTACCQ=0,"client test0"
OK

// Set the will topic for the CONNECT message
AT+CMQTTWILLTOPIC=0,10
>0123456789

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

OK

// Connect to a MQTT server
AT+CMQTTCONNECT=0,"tcp://test.mosquitto.org:1883",60,1
OK

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

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

OK
// Set the payload for the PUBLISH message
AT+CMQTTPAYLOAD=0,60
> 012345678901234567890123456789012345678901234567890123456789

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

+CMQTTPUB: 0,0
//receive publish message from server
+CMQTTRXSTART: 0,9,60
+CMQTTRXTOPIC: 0,9
simcommsg
+CMQTTRXPAYLOAD: 0,60
012345678901234567890123456789012345678901234567890123456789
+CMQTTRXEND: 0
// Set one topic for the SUBSCRIBE message
AT+CMQTTSUBTOPIC=0,9,1
>123456789

OK
// Subscribe a message
AT+CMQTTSUB=0
OK

+CMQTTSUB: 0,0

```
// Unsubscribe one topic from the server
AT+CMQTTUNSUB=0,9,0
>simcommsg
OK

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

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

+CMQTTSTOP: 0
```

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

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

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

+CMQTTSTART: 0
// Acquire one client which will connect to a SSL/TLS MQTT server
AT+CMQTTACCQ=0,"client test0",1
OK
// Set the will topic for the CONNECT message
AT+CMQTTWILLTOPIC=0,10
>0123456789

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

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

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

OK
// Set the payload for the PUBLISH message
AT+CMQTTPAYLOAD=0,60
> 012345678901234567890123456789012345678901234567890123456789

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

+CMQTTPUB: 0,0
// Set one topic for the SUBSCRIBE message
AT+CMQTTSUBTOPIC=0,9,1
>123456789

OK
// Subscribe a message
AT+CMQTTSUB=0
OK

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

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

+CMQTTUNSUB: 0,0

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

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

+CMQTTSTOP: 0
```

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

Following commands shows how to access to a SSL/TLS MQTT server 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.

```
AT+CSSLCFG="sslversion",0,4// Set the SSL version of the first SSL context
OK

// Set the authentication mode(verify server) of the first SSL context
AT+CSSLCFG="authmode",0,1
OK

// Set the server root CA of the first SSL context
AT+CSSLCFG="cacert",0,"server_ca.pem"
OK
AT+CMQTTSTART // start MQTT service, activate PDP context
OK

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

// Set the will topic for the CONNECT message
AT+CMQTTWILLTOPIC=0,10
>0123456789

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

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

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

OK
// Set the payload for the PUBLISH message
AT+CMQTTPAYLOAD=0,60
> 012345678901234567890123456789012345678901234567890123456789

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

+CMQTTPUB: 0,0
// Set one topic for the SUBSCRIBE message
AT+CMQTTSUBTOPIC=0,9,1
>123456789

OK
// Subscribe a message
AT+CMQTTSUB=0
OK

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

```
OK

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

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

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


+CMQTTSTOP: 0
```

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

Following commands shows how to access to a SSL/TLS MQTT server 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.

```
// Set the SSL version of the first SSL context

AT+CSSLCFG="sslversion",0,4

OK

// Set the authentication mode(verify server and client) of the first SSL context

AT+CSSLCFG="authmode",0,2

OK

// Set the server root CA of the first SSL context

AT+CSSLCFG="cacert",0,"ca_cert.pem"
```

OK

// Set the client certificate of the first SSL context

AT+CSSLCFG=”clientcert”,0,”cert.pem”

OK

// Set the client key of the first SSL context

AT+CSSLCFG=”clientkey”,0,”key_cert.pem”
OK
// start MQTT service, activate PDP context
AT+CMQTTSTART
OK

+CMQTTSTART: 0
// Acquire one client which will connect to a SSL/TLS MQTT server
AT+CMQTTACCQ=0,”client test0”,1
OK
// Set the first SSL context to be used in the SSL connection
AT+CMQTTSSLCFG=0,0
OK
// Set the will topic for the CONNECT message
AT+CMQTTWILLTOPIC=0,10
>0123456789

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

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

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

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

> 01234567890123456789012345678901234567890123456789012345678901234567890123456789

OK

// Publish a message

AT+CMQTTPUB=0,1,60

OK

+CMQTTPUB: 0,0

// Set one topic for the SUBSCRIBE message

AT+CMQTTSUBTOPIC=0,9,1

>123456789

OK

// Subscribe a message

AT+CMQTTSUB=0

OK

+CMQTTSUB: 0,0

// Subscribe one topic from the server

AT+CMQTTSUB=0,9,1

>simcommsg

OK

+CMQTTSUB: 0,0

// Unsubscribe one topic from the server

AT+CMQTTUNSUB=0,9,0

>simcommsg

OK

+CMQTTUNSUB: 0,0

// Disconnect from server

AT+CMQTTDISC=0,120

OK

+CMQTTDISC: 0,0

//Release the client

AT+CMQTTREL=0

OK

//stop MQTT Service

AT+CMQTTSTOP

OK

+CMQTTSTOP: 0

3.5 Access to MQTT server without checking UTF8 coding

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

// start MQTT service, activate PDP context

AT+CMQTTSTART

OK

+CMQTTSTART: 0

// Acquire one client which will connect to a MQTT server not SSL/TLS

AT+CMQTTACCQ=0,”client test0”

OK

// Configure not checking UTF8 coding

AT+CMQTTCFG=”checkUTF8”,0,0

OK

// Connect to a MQTT server

AT+CMQTTCONNECT=0,"tcp://test.mosquitto.org:1883",60,1

OK

+CMQTTCONNECT: 0,0

// Subscribe one topic which is not UTF8 coding string.

//The data can input by hexadecimal format.

AT+CMQTTSUB=0,9,1

>鼢鼢鼢鼢?

OK

+CMQTTSUB: 0,0

// Set the topic for the PUBLISH message

AT+CMQTTTOPIC=0,9

>鄖鄖鄖鄖?

OK

// Publish a message

AT+CMQTTPUB=0,1,60

OK

+CMQTTPUB: 0,0

//receive publish message from server

+CMQTTRXSTART: 0,9,0

+CMQTTRXTOPIC: 0,9

鄖鄖鄖鄖?

+CMQTTRXEND: 0

// Disconnect from server

AT+CMQTTDISC=0,120

OK

+CMQTTDISC: 0,0

//Release the client

AT+CMQTTREL=0

OK

//stop MQTT Service

AT+CMQTTSTOP

OK

+CMQTTSTOP: 0