FCM360W AT Commands Manual

Wi-Fi&Bluetooth Module Series

Version: 1.0.0

Date: 2023-11-14

Status: Preliminary

At Quectel, our aim is to provide timely and comprehensive services to our customers. If you require any assistance, please contact our headquarters:

Quectel Wireless Solutions Co., Ltd.
Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local offices. For more information, please visit:
http://www.quectel.com/support/sales.htm.

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/technical.htm.
Or email us at: support@quectel.com.

Legal Notices

Use and Disclosure Restrictions

License Agreements

Copyright

About the Document

Revision History

Version	Date	Author	Description
-	2023-11-14	Orange LI	Creation of the document
1.0.0	2023-11-14	Orange LI	Preliminary

Contents

Table Index

Figure Index

1 Introduction

This document outlines Wi-Fi, BLE, TCP/UDP, SSL, MQTT and HTTP(S)-related AT commands supported by Quectel FC41D module.

1.1. Definitions

● **<CR>**	Carriage return character.
● **<LF>**	Line feed character.
● **<...>**	Parameter name. Angle brackets do not appear on the command line.
● **[...]**	Optional parameter of a command or an optional part of TA information response. Square brackets do not appear on the command line. When an optional parameter is not given in a command, the new value equals its previous value or the default settings, unless otherwise specified.
● **<u>Underline</u>**	Default setting of a parameter.

1.2. AT Command Syntax

All command lines must start with **AT** or **at** and end with **<CR>**. Information responses and result codes always start and end with a carriage return character and a line feed character: **<CR><LF><response><CR><LF>**. In tables presenting commands and responses throughout this document, only the commands and responses are presented, and **<CR>** and **<LF>** are deliberately omitted.

AT commands implemented by the module can be separated into three categories syntactically: **"Basic"**, **"S Parameter"** and **"Extended"**, as listed below:

- **Basic Command**

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 (Data Circuit-terminating Equipment) whether received characters should be echoed back to the DTE (Data Terminal Equipment) according to the value of **<n>**. **<n>** is optional and a default will be used if it is omitted.

- **S Parameter Syntax**

These AT commands are in 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.

- **Extended Command**

These AT commands are in 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.

Table 1: Types of AT Commands

Command Type	Syntax	Description
Test Command	**AT+<cmd>=?**	Test the existence of the corresponding command and return information about the type, value, or range of its parameter.
Read Command	**AT+<cmd>?**	Check the current parameter value of the corresponding command.
Write Command	**AT+<cmd>=<p1>[,<p2>[,<p3>[...]]]**	Set user-definable parameter value.
Execution Command	**AT+<cmd>**	Return a specific information parameter or perform a specific action.

Multiple commands can be placed on a single line using a semi-colon (;) between commands. In such cases, only the first command should have AT prefix. Commands can be in upper or lower case.

Spaces should be ignored when you enter AT commands, except in the following cases:

- Within quoted strings, where spaces are preserved;
- Within an unquoted string or numeric parameter;
- Within an IP address;
- Within the AT command name up to and including a =, ? or =?.

On input, at least a carriage return is required. A newline character is ignored so it is permissible to use carriage return/line feed pairs on the input.

If no command is entered after the **AT** token, **OK** will be returned. If an invalid command is entered, **ERROR** will be returned.

Optional parameters, unless explicitly stated, need to be provided up to the last parameter being entered.

1.3. AT Command Responses

When the AT command processor has finished processing a line, it will output OK, ERROR or +CME ERROR: <err> to indicate that it is ready to accept a new command. Solicited information responses are sent before the final OK, ERROR or +CME ERROR: <err>.

Responses will be in the format of:

<CR><LF>+CMD1:<parameters><CR><LF>
<CR><LF>OK<CR><LF>

Or

<CR><LF><parameters><CR><LF>
<CR><LF>OK<CR><LF>

1.4. Declaration of AT Command Examples

The AT command examples in this document are provided to help you learn about the use of the AT commands introduced herein. The examples, however, should not be taken as Quectel's recommendations or suggestions about how to design a program flow or what status to set the module into. Sometimes multiple examples may be provided for one AT command. However, this does not mean that there is a correlation among these examples, or that they should be executed in a given sequence.

2 AT Commands Description

2.1. Description of Wi-Fi-Related AT Commands

2.1.1. AT+QRST Reboot Module

This command reboots the module.

AT+QRST Reboot Module	
Execution Command **AT+QRST**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

2.1.2. AT+QVERSION Get Firmware Version

This command gets firmware version of the module.

AT+QVERSION Get Firmware Version	
Execution Command **AT+QVERSION**	Response **+QVERSION: <version>** **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<version>	String type without double quotation marks. Firmware version number.

2.1.3. AT+QECHO Enable/Disable Echo Function

This command enables or disables echo function.

AT+QECHO Enable/Disable Echo Function	
Write Command **AT+QECHO=<enable>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configuration is saved automatically.

Parameter

<enable>	Integer type. Enable/disable echo function.	
	<u>0</u>	Disable
	1	Enable

2.1.4.AT+QGETIP Get IP Information

This command gets the IP information of the module.

AT+QGETIP Get IP Information	
Write Command **AT+QGETIP=<mode>**	Response **+QGETIP: <IP>,<gate>,<mask>,<DNS>** **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<mode>	String type. Wi-Fi working mode.	
	"station"	STA mode
	"ap"	AP mode
<IP>	String type. IP address of the module.	
<gate>	String type. Gateway address of the module. When no static IP is configured in AP mode, this parameter is "0.0.0.0" by default.	

<mask>	String type. Module's subnet mask.
<DNS>	String type. DNS address of the module.

2.1.5.AT+QSETBAND Configure Serial Port Baud Rate

This command configures serial port baud rate.

AT+QSETBAND Configure Serial Port Baud Rate	
Write Command **AT+QSETBAND=<baud_rate>,<save>**	Response **OK** Or **ERROR**
Read Command **AT+QSETBAND?**	Response **+QSETBAND: <baud_rate>** **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately.

Parameter

<baud_rate>	Integer type. Serial port baud rate. Range: 1200–2000000. Unit: bps.
<save>	Integer type. 0 Do not save 1 Save

2.1.6.AT+QWLANOTA Start OTA Upgrade

This command starts OTA upgrade for firmware.

AT+QWLANOTA Start OTA Upgrade	
Write Command **AT+QWLANOTA=<URL>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<URL>	String type. Address where firmware package is stored.

2.1.7. AT+QWLMAC Get MAC Address

This command gets the MAC address of the module.

AT+QWLMAC Get MAC Address	
Execution Command **AT+QWLMAC**	Response **+QWLMAC: <MAC>** **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<MAC>	String type. MAC address of the module. Hexadecimal numbers separated by colons. Default value: "88:00:33:77:69:cc".

2.1.8. AT+QSTAST Query STA Mode State

This command queries STA mode state.

AT+QSTAST Query STA Mode State	
Execution Command **AT+QSTAST**	Response **+QSTAST: <state>** **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<state>	String type. STA mode status. "STATION_DOWN" Disabled; "STATION_UP" Enabled

2.1.9. AT+QSTADHCP Enable/Disable DHCP Service in STA Mode

This command enables or disables DHCP service in STA mode.

AT+QSTADHCP Enable/Disable DHCP Service in STA Mode	
Read Command **AT+QSTADHCP?**	Response **+QSTADHCP: <enable >** **OK** If there is any error: **ERROR**
Write Command **AT+QSTADHCP=<enable>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately The configuration is not saved.

Parameter

<enable>	Integer type. Enable/Disable DHCP service in STA mode. 0 Disable; 1 Enable

2.1.10. AT+QAPSTATIC Configure Static IP of STA Mode

This command configures static IP of STA mode.

AT+QAPSTATIC Configure Static IP of STA Mode	
Write Command **AT+QAPSTATIC=<IP>,<mask>,<gate>,<DNS>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms

Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<IP>	String type. Static IP address of STA mode.
<mask>	String type. Subnet mask of the module.
<gate>	String type. Gateway address of the module.
<DNS>	String type. DNS address of the module.

2.1.11. AT+QSTASTOP Disable STA Mode

This command disables STA mode.

AT+QSTASTOP Disable STA Mode	
Execution Command **AT+QSTASTOP**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

2.1.12. AT+QSOFTAP Enable AP Mode

This command enables AP mode.

AT+QSOFTAP Enable AP Mode	
Write Command **AT+QSOFTAP=<SSID>[,<key>]**	Response **OK** Or **ERROR**
Maximum Response Time	3300 ms (Hotspot with password enabled)/ 300 ms (Hotspot without password enabled)
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<SSID>	String type. AP name. Range: 1–32. Unit: Byte.
<key>	String type. AP password. Range: 8–63. Unit: Byte.

	If this parameter is omitted, hotspot without password will be enabled.

2.1.13. AT+QAPSTATE Query AP Mode State

This command queries AP mode state.

AT+QAPSTATE Query AP Mode State	
Execution Command **AT+QAPSTATE**	Response **+QAPSTATE: <state>** **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<state>	String type. AP mode status.	
	"SOFTAP_DOWN"	Disabled
	"SOFTAP_UP"	Enabled

2.1.14. AT+QAPSTATIC Configure Static IP of AP Mode

This command configures static IP of AP mode.

AT+QAPSTATIC Configure Static IP of AP Mode	
Write Command **AT+QAPSTATIC=<IP>,<mask>,<gate>,<DNS>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	This command takes effect immediately. The configurations are not saved.

Parameter

<IP>	String type. Static IP address of STA mode.
<mask>	String type. Subnet mask of the module.

<gate>	String type. Gateway address of the module.
<DNS>	String type. DNS address of the module.

2.1.15. AT+QSOFTAPSTOP Disable AP Mode

This command disables AP mode.

AT+QSOFTAPSTOP Disable AP Mode	
Execution Command **AT+QSOFTAPSTOP**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

2.1.16. AT+QSTAAPINFO Connect an AP Hotspot

This command connects an AP hotspot to enable STA mode.

AT+QSTAAPINFO Connect an AP Hotspot	
Write Command **AT+QSTAAPINFO=<SSID>[,<pwd>]**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<SSID>	String type. Name of AP hotspot to be connected to. Range: 1–32. Unit: byte.
<pwd>	String type. Password of AP hotspot to be connected to. Range: 8–63. Unit: byte. If this parameter is omitted, the module will connect to an AP hotspot without a password.

2.1.17. AT+QSTAAPINFODEF Connect a Hotspot and Save Hotspot Information

This command connects to AP hotspot and enters STA mode, and saves the hotspot information at the same time.

AT+QSTAAPINFODEF Connect a Hotspot and Save Hotspot Information	
Write Command **AT+QSTAAPINFODEF=<SSID>[,<pwd>]**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are saved automatically.

Parameter

<SSID>	String type. Name of AP hotspot to be connected to. Range: 1–32. Unit: byte.
<pwd>	String type. Password of AP hotspot to be connected to. Range: 8–63. Unit: byte. If this parameter is omitted, the module will connect to an AP hotspot without a password.

2.1.18. AT+QGETWIFISTATE Query Connected Hotspot

This command queries the connected hotspot when the module is in STA mode.

AT+QGETWIFISTATE Query Connected Hotspot	
Execution Command **AT+QGETWIFISTATE**	Response **+QGETWIFISTATE: ssid=<SSID>,bssid=<BSSID>,rssi=<RSSI>** **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<SSID>	String type. Name of connected Wi-Fi hotspot.
<BSSID>	String type. BSSID of the Wi-Fi hotspot.

<RSSI>	Integer type. Wi-Fi signal strength.

2.1.19. AT+QWSCAN Query Scanned Hotspot Information

This command queries the scanned hotspot information.

AT+QWSCAN Query Scanned Hotspot Information	
Execution Command **AT+QWSCAN**	Response **+QWSCAN: <SSID>,<PSK_type>,<RSSI>,<BSSID>,<channel>** **[…]** **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<SSID>	String type. Name of scanned Wi-Fi hotspot.
<PSK_type>	String type. Encryption type.
<RSSI>	Integer type. Wi-Fi Signal strength.
<BSSID>	String type. BSSID of the Wi-Fi hotspot.
<channel>	Integer type. Channel used by module to connect and communicate with Wi-Fi hotspot in STA mode.

2.1.20. AT+QIDNSCFG Configure DNS Server Address

This command configures DNS server address.

AT+QIDNSCFG Configure DNS Server Address	
Read Command **AT+QIDNSCFG?**	Response **+QIDNSCFG: <pridnsaddr>,<secdnsaddr>** **OK** If there is any error: **ERROR**
Test Command	Response

AT+QIDNSCFG=?	**+QIDNSCFG: <pridnsaddr>,<secdnsaddr>** **OK**
Write Command **AT+QIDNSCFG=<pridnsaddr>[,<secdnsaddr>]**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<pridnsaddr>	String type. Primary DNS server address.
<secdnsaddr>	String type. Secondary DNS server address.

2.1.21. AT+QWEBCFG Enable/Disable Configuring Wi-Fi via Web

This function enables or disables configuring Wi-Fi via Web.

AT+QWEBCFG Enable/Disable Configuring Wi-Fi via Web	
Write Command **AT+QWEBCFG=<enable>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configuration is saved automatically.

Parameter

<enable>	Integer type. Enable/Disable configuring Wi-Fi via Web 0 Disable 1 Enable

2.1.22. AT+QSTAAPINFORMV Clear Saved Hotspot Information

This command clears saved hotspot information.

AT+QSTAAPINFORMV Clear Saved Hotspot Information	
Execution Command **AT+QSTAAPINFORMV**	Response **OK**

	Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

2.2. Description of BLE-Related AT Commands

2.2.1. AT+QBLEINIT Initialize BLE

This command initializes BLE.

AT+QBLEINIT Initialize BLE	
Write Command **AT+QBLEINIT=<role>,<auto_adv>**	Response **OK** Or **ERROR**
Read Command **AT+QBLEINIT?**	Response **+QBLEINIT: <role>** **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<role>	Integer type. Initializes BLE. 2 Initializes BLE when the module is operating as a peripheral device. The peripheral device disables advertising by default after Bluetooth is disconnected.
<auto_adv>	Integer type. After Bluetooth is disconnected, whether to enable advertising automatically when the module is operating as a peripheral device. 0 Disable 1 Enable

2.2.2. AT+QBLEADDR Query BLE Device Address

This command queries BLE device address.

AT+QBLEADDR Query BLE Device Address	
Read Command **AT+QBLEADDR?**	Response **+QBLEADDR: <BLE_addr>** **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<BLE_addr>	String type. BLE device address. A 48-bit address is represented in a string of hexadecimal numbers, such as "58:D3:91:01:02:03".

2.2.3. AT+QBLENAME Set BLE Name

This command sets a BLE name.

AT+QBLENAME Set BLE Name	
Read Command **AT+QBLENAME?**	Response **+QBLENAME: <BLE_name>** **OK** If there is any error: **ERROR**
Write Command **AT+QBLENAME=<BLE_name>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<BLE_name>	String type. BLE name. Maximum length: 17 bytes.

2.2.4. AT+QBLEADVPARAM Configure BLE Advertising Parameters

This command configures BLE advertising parameters when the module is operating as a peripheral device.

AT+QBLEADVPARAM Configure BLE Advertising Parameters	
Write Command **AT+QBLEQADVPARAM=<adv_int_min>,<adv_int_max>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<adv_int_min>	Integer type. Minimum advertising interval for non-directional advertising and low-duty cycle directional advertising. Range: 0x0020–0x4000 (corresponding time range: 20 milliseconds – 10.24 seconds). Default value: 0x40 (40ms). Unit: timeslot (1 timeslot = 0.625 milliseconds).
<adv_int_max>	Integer type. Maximum advertising interval for non-directional advertising and low-duty cycle directional advertising. Range: 0x0020–0x4000 (corresponding time range: 20 milliseconds – 10.24 seconds). Default value: 0x40 (40ms). Unit: timeslot (1 timeslot = 0.625 milliseconds).

NOTE

To configure advertising parameters, **AT+QBLEADVPARAM** should be executed before starting advertising with **AT+QBLEADVSTART**.

2.2.5. AT+QBLEADVDATA Set BLE Advertising Data

This command sets BLE advertising data when the module is operating as a peripheral device.

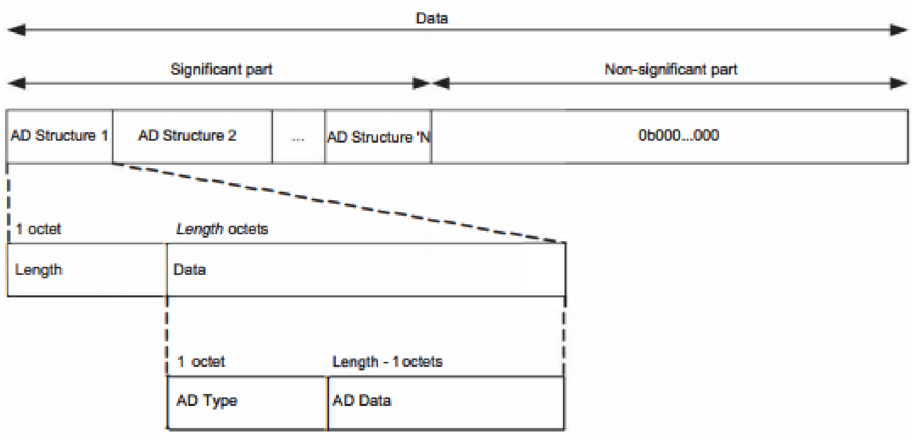


Figure 1: BLE Advertising Format

AT+QBLEADVDATA Set BLE Advertising Data	
Write Command **AT+QBLEADVDATA=<adv_data>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configuration is not saved.

Parameter

<adv_data>	String type. Advertising data (AD). It consists of three fields (i.e., multiple AD Structures). The composition conforms to the message format shown above and the content must be a hexadecimal string.	
	Length	Length of AD structure. The length includes AD type and AD data but not the length of the field which is 1 byte long. The maximum length is 0x1b, i.e., the maximum length of a data field is 27 bytes.
	AD Type	Advertising data type, such as TX Power Level (0x0A), Local Name (0x09), Le Role (0x1C) and Service UUIDs (0x16). After the peer scans the advertisement, the meaning of the advertising data can be determined from the AD Type.
	AD Data	Advertising data in big-endian byte order.

NOTE

For details about types and values of AD Type, see *Core Specification 5.2* (https://www.bluetooth.com/specifications/specs/core-specification/).

2.2.6. AT+QBLEGATTSSRV Establish a BLE Service

This command establishes a BLE service when module is operating as a peripheral device.

AT+QBLEGATTSSRV Establish a BLE Service	
Write Command **AT+QBLEGATTSSRV=<srv_UUID>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configuration is not saved.

Parameter

<srv_UUID>	String type. BLE Service UUID. Length: 2 bytes or 16 bytes.

NOTE

Only one BLE service can be established at a time with this command.

2.2.7. AT+QBLEGATTSCHAR Set BLE Characteristics

This command sets BLE characteristic UUID when the module is operating as a peripheral device.

AT+QBLEGATTSCHAR Set BLE Characteristics	
Write Command **AT+QBLEGATTSCHAR=<char_UUID>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configuration is not saved.

Parameter

<char_UUID>	String type. Characteristic UUID. Length: 2 bytes or16 bytes.

NOTE

The maximum of 5 characteristics can be set with this command.

2.2.8.AT+QBLEGATTSSRVDONE Add BLE Service

This command adds the BLE service when the module is operating as a peripheral device.

AT+QBLEGATTSSRVDONE Add BLE Service	
Write Command **AT+QBLEGATTSSRVDONE**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

2.2.9.AT+QBLEADVSTART Start BLE Advertising

This command starts BLE advertising when the module is operating as a peripheral device.

AT+QBLEADVSTART Start BLE Advertising	
Execution Command **AT+QBLEADVSTART**	Response **OK** Or **ERROR**
Maximum Response Time	1000 ms
Characteristic	/

2.2.10. AT+QBLEADVSTOP Stop BLE Advertising

This command stops BLE advertising when the module is operating as a peripheral device.

AT+QBLEADVSTOP Stop BLE Advertising	
Execution Command **AT+QBLEADVSTOP**	Response

	OK Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

2.2.11. AT+QBLEGATTSNTFY Send GATT Data

This command sends GATT data when the module is operating as a peripheral device.

AT+QBLEGATTSNTFY Send GATT Data	
Write Command **AT+QBLEGATTSNTFY=<UUID>[,<hex_length>],<data>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<UUID>	String type. Characteristic UUID. Length: 2 bytes or 16 bytes.
<hex_length>	Integer type. The length of hexadecimal data. The module will convert **<data>** to hexadecimal format before sending data when this optional parameter is specified. For example, if **<data>** is 123456, the module will convert the data to 0x123456 and send it. When this parameter is omitted, the module will send **<data>** directly without conversion. For example, if **<data>** is 123456, the module will send the data as 123456.
<data>	String type. GATT data. The length range of GATT data is related to **<MTU_value>**, which is **<MTU_value>**-3. For details about **<MTU_value>**, see ***Chapter 3.2.2***.

2.3. Description of TCP/UDP Related AT commands

2.3.1. AT+QICFG Configure Parameters Related to TCP/UDP Socket Service

AT+QICFG Configure Parameters Related to TCP/UDP Socket Service	
Execution Command **AT+QICFG=?**	Response **+QICFG: "transpktsize",(**range of supported **<transpktsize>**

	s) **+QICFG: "transwaittm",(**range of supported **<transwaittm>** s) **+QICFG: "dataformat",(**list of supported **<send_data_format>**s)**,(**list of supported **<recv_data_format>**s) **+QICFG: "passiveclosed",(**list of supported **<closed>**s) **+QICFG: "tcp/accept",(**list of supported **<state>**s) **+QICFG: "qisend/timeout",(**range of supported **<timeout>**s) **+QICFG: "close/mode",(**list of supported **<close_mode>**s) **+QICFG: "tcp/tw_cycle",(**list of supported **<tw_enable>**s) **OK**
Write Command Set the maximum length of the data package to be sent. **AT+QICFG="transpktsize"[,<transpktsize>]**	Response If the optional parameter is omitted, query the current setting: **+QICFG: "transpktsize",<transpktsize>** **OK** If the optional parameter is specified, set the maximum length of the data package to be sent: **OK** If there is any error: **ERROR**
Write Command In transparent transmission mode, set the waiting time before sending data automatically. **AT+QICFG="transwaittm"[,<transwaittm>]**	Response If the optional parameter is omitted, query the current setting: **+QICFG: "transwaittm",<transwaittm>** **OK** If the optional parameter is specified, in transparent transmission mode, set the waiting time before sending data automatically: **OK** If there is any error: **ERROR**
Write Command Set the format of sending and receiving data (only in non-transparent transmission mode) **AT+QICFG="dataformat"[,<send_data_format>,<recv_data_format>]**	Response If the optional parameters are omitted, query the current setting: **+QICFG: "dataformat",<send_data_format>,<recv_data_format>** **OK** If optional parameters are specified, set the sending and

	receiving data format: **OK** If there is any error: **ERROR**
Write Command Set whether to passively close TCP connection when the server is shut down. **AT+QICFG="passiveclosed"[,<closed>]**	Response If the optional parameter is omitted, query the current setting: **+QICFG: "passiveclosed",<closed>** **OK** If optional parameter is specified, set whether to passively close TCP connection when the server is shut down. **OK** If there is any error: **ERROR**
Write Command Set whether to enable or disable automatically accepting TCP connections from the client. **AT+QICFG="tcp/accept"[,<state>]**	Response If the optional parameter is omitted, query the current setting: **+QICFG: "tcp/accept",<state>** **OK** If optional parameter is specified, set whether to automatically accept TCP connections from the client: **OK** If there is any error: **ERROR**
Write Command Set the maximum response time for sending **AT+QISEND**. **AT+QICFG="qisend/timeout"[,<timeout>]**	Response If the optional parameter is omitted, query the current setting: **+QICFG: "qisend/timeout",<timeout>** **OK** If optional parameter is specified, set the timeout after output **>**: **OK** If there is any error: **ERROR**
Write Command Set an asynchronous TCP disconnection. **AT+QICFG="close/mode"[,<close**	Response If the optional parameter is omitted, query the current setting: **+QICFG: "close/mode",<close_mode>**

_mode>]	**OK** If optional parameter is specified, set an asynchronous TCP disconnection: OK If there is any error: **ERROR**
Write Command Set whether to enable or disable a quick release of TCP connection. **AT+QICFG="tcp/tw_cycle",<tw_enable>**	Response If the optional parameter is omitted, query the current setting: **+QICFG: "tcp/tw_cycle",<tw_enable>** **OK** If optional parameter is specified, set whether to enable or disable a quick release of TCP connection: **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristic	The command takes effect immediately The configurations are not saved.

Parameter

<transpktsize>	Integer type. The length of data package to be sent in transparent transmission mode. Range: 1–1460. Default value: 1024. Unit: byte.
<transwaittm>	Integer type. Data transmission waiting time if the data to be sent is less then **<transpktsize>** in transparent transmission mode. Range: 0–20. Default value: 2. Unit: 100 ms.
<send_data_format>	Integer Type. Sent data format. When it is set to hexadecimal format, there is no need to add the prefix 0x, the module will automatically combine two bytes into one ASCII code. 0 Text format 1 Hexadecimal format
<recv_data_format>	Integer Type. Received data format. When it is set to hexadecimal format, there is no need to add the prefix 0x, the module will automatically combine two bytes into one ASCII code. 0 Text format 1 Hexadecimal format
<closed>	Integer Type. Enable or disable that TCP connection is automatically disconnected after the server is closed.

	0 Disable 1 Enable
<state>	Integer Type. Enable or disable automatic acception of TCP connection from the client. 0 Disable 1 Enable
<timeout>	Integer type. Timeout for sending data. Range: 0–120. Unit: second.
<close_mode>	Integer type. Enable or disable to disconnect TCP connection asynchronously. 0 Disable 1 Enable
<tw_enable>	Integer type. Enable or disable a quick release of TCP connection. 0 Disable 1 Enable

2.3.2.AT+QIOPEN Open Socket Service

The command opens a socket service. The service type can be specified by **<service_type>**. The data access mode (buffer access mode, direct push mode and transparent transmission mode) can be specified by **<access_mode>**. The URC **+QIOPEN** indicates whether the socket service has been opened successfully.

1. If **<service_type>** is "TCP LISTENER", the module works as TCP server. After accepting a new TCP connection, the module will automatically specify a **<connectID>** and report a URC **+QIURC**: **"incoming",<connectID>,<serverID>,<remoteIP>,<remote_port>**. The range of **<connectID>** is 0–11. The type of this new incoming connection is "TCP INCOMING" and the **<access_mode>** of "TCP INCOMING" is the same with that of "TCP LISTENER".

2. If **<service_type>** is "UDP SERVICE", UDP data can be sent to or received from the remote IP via **<local_port>**.

- Send data: execute **AT+QISEND=<connectID>,<send_length>,<remoteIP>,<remote_port>**.
- Receive data in direct push mode: the module reports the URC **+QIURC**: **"recv",<connectID>,<currentrecvlength>,<remoteIP>,<remote_port><CR><LF><data>**.
- Receive data in buffer access mode: the module reports the URC **+QIURC**: **"recv",<connectID>**, and then data can be read via **AT+QIRD=<connectID>**.

3. It is suggested to wait for 150 seconds for **+QIOPEN: <connectID>,<err>** to be outputted. If the URC cannot be received in 150 seconds, **AT+QICLOSE** should be used to close the socket.

AT+QIOPEN Open Socket Service	
Test Command **AT+QIOPEN=?**	Response **+QIOPEN: (**range of supported **<connectID>**s**),(**list of sup

	ported **<service_type>**s)**,<IP_address>/<domain_name>,(**range of supported **<remote_port>**s)**,(**range of supported **<local_port>**s)**,(**range of supported **<access_mode>**s) **OK**
Write Command **AT+QIOPEN=<connectID>,<service_type>,<IP_address>/<domain_name>,<remote_port>[,<local_port>[,<access_mode>]]**	Response If the service is in transparent transmission mode (**<access_mode>**=2) and is opened successfully: **CONNECT** If there is any error: **ERROR** Error description can be obtained via **AT+QIGETERROR**. If the service is in buffer access mode (**<access_mode>**=0) or direct push mode (**<access_mode>**=1): **OK** **+QIOPEN: <connectID>,<err>** **<err>** is 0 when the service is opened successfully. In other cases, **<err>** is not 0.
Maximum Response Time	150 seconds, determined by the network.
Characteristics	/

Parameter

<connectID>	Integer type. Socket ID. Range: 0–11.	
<service_type>	String type. Socket service type.	
	"TCP"	Establish a TCP connection. Module is a client at this time.
	"UDP"	Establish a UDP connection. Module is a client at this time.
	"TCP LISTENER"	Establish a TCP server to listen to TCP connection
	"UDP SERVICE"	Establish a UDP service.
<IP_address>	String type. If **<service_type>** is "TCP" or "UDP", this parameter is the IP address of remote server, such as "220.180.239.212". If **<service_type>** is "TCP LISTENER" or "UDP SERVICE", this parameter can be set as "127.0.0.1".	
<domain_name>	String type. The domain name address of the remote server.	
<remote_port>	Integer type. The port of the remote server. Range: 1–65535. It is valid only when **<service_type>** is "TCP" or "UDP".	
<local_port>	Integer type. The local port. Range: 1–65535.	

	If **<service_type>** is "TCP LISTENER" or "UDP SERVICE", this parameter must be specified. If **<service_type>**is "TCP" or "UDP", and **<local_port>** is not specified, then the local port will be automatically assigned, otherwise the port will be as specified.
<access_mode>	Integer type. Data access mode of Socket service. 0 Buffer access mode 1 Direct push mode 2 Transparent transmission mode
<err>	Error code. See ***Chapter*** *错误!未找到引用源。* for details.

NOTE

In this command, **<IP_address>/<domain_name>** indicates that the parameter can only be **<IP_address>** or **<domain_name>**.

2.3.3. AT+QICLOSE Close a Socket Service

The command closes a specified socket service. Depending on the network, it will take at most 10 seconds (default value, can be modified by **<timeout>**) to return **OK** or **ERROR** after executing **AT+QICLOSE**. Before the response is returned, other AT commands cannot be executed.

AT+QICLOSE Close a Socket Service	
Test Command **AT+QICLOSE=?**	Response **+QICLOSE: (**range of supported **<connectID>**s**),(**range of supported **<timeout>**s**)** **OK**
Write Command **AT+QICLOSE=<connectID>[,<timeout>]**	Response Close successfully: **OK** Failed to close: **ERROR**
Maximum Response Time	Default value: 10 seconds. Depends on **<timeout>**.
Characteristics	/

Parameter

<connectID>	Integer type. Socket ID. Range: 0–11.
<timeout>	Integer type. The timeout value for the response to be outputted. If the FIN ACK of the other peer is not received within the value of **<timeout>**, the module will be forced to

	close the socket. Range: 1–120. Default value: 10. Unit: second.

2.3.4.AT+QISTATE Query Socket Service Status

This command queries the socket service status.

AT+QISTATE Query Socket Service Status	
Test Command **AT+QISTATE=?**	Response **OK**
Read/Execution Command **AT+QISTATE?** Or **AT+QISTATE**	Response Return the status of all existing connections: **+QISTATE: <connectID>,<service_type>,<IP_address>,<remote_port>,<local_port>,<socket_state>,<serverID>,<access_mode>,<AT_port>** **[…]** **OK**
Write Command **AT+QISTATE=<connectID>**	Response **+QISTATE: <connectID>,<service_type>,<IP_address>,<remote_port>,<local_port>,<socket_state>,<serverID>,<access_mode>,<AT_port>** **OK**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<connectID>	Integer type. Socket ID. Range: 0–11.	
<service_type>	String type. The socket service type.	
	"TCP"	Establish a TCP connection. Module is a client at this time.
	"UDP"	Establish a UDP connection. Module is a client at this time.
	"TCP LISTENER"	Establish a TCP server to listen to TCP connection
	"TCP INCOMING"	Establish a TCP connection accepted by a TCP server
	"UDP SERVICE"	Establish a UDP service
<IP_address>	String type. IP address. If **<service_type>**="TCP" or "UDP", this parameter is the IP address of remote server. If **<service_type>**="TCP LISTENER" or "UDP SERVICE", this parameter is the local IP address. If **<service_type>**="TCP INCOMING", this parameter is the IP address of remote	

	client.
<remote_port>	Integer type. Remote port number. If **<service_type>**="TCP" or "UDP", this parameter is the port of remote server. If **<service_type>**="TCP LISTENER" or "UDP SERVICE", the port is invalid. If **<service_type>**="TCP INCOMING", this parameter is the port of remote client.
<local_port>	Integer type. Local Port number. If **<local_port>** is not specified, then the local port will be automatically assigned, otherwise the port will be as specified.
<socket_state>	Integer type. The socket service status. 0 "Initial": connection has not been established 1 "Opening": client is connecting or server is trying to listen 2 "Connected": client connection has been established 3 "Listening": server is listening 4 "Closing": connection is closed
<serverID>	Integer type. It is valid only when **<service_type>** is "TCP INCOMING". **<serverID>** represents which server accepts this TCP incoming connection, and the value is the same as **<connectID>** of this server's "TCP LISTENER".
<access_mode>	Integer type. Data access mode. 0 Buffer access mode 1 Direct push mode 2 Transparent transmission mode
<AT_port>	String type. COM port of socket service. "uart1" UART port 1 "uart2" UART port 2

2.3.5. AT+QISEND Send Data

If the data access mode of a specified socket service is buffer access mode (**<access_mode>**=0) or direct push mode (**<access_mode>**=1), then the data can be sent via this command. When the data is sent to the module successfully, **SEND OK** will be returned, otherwise **SEND FAIL** or **ERROR** will be returned.

- **SEND FAIL** indicates the buffer of sent data is full, and users can resend the data.
- **ERROR** indicates an error is displayed in the process of sending data. Users should wait for a while and resend the data. The maximum data length is 1460 bytes.
- **SEND OK** does not mean the data has been sent to the server successfully. Users can query the result by **AT+QISEND=<connectID>,0**.

AT+QISEND Send Data	
Test Command **AT+QISEND=?**	Response **+QISEND: (**range of supported**<connectID>**s**),(**range of supported **<send_length>**s**),<remoteIP>,(**range of supported **<remote_port>**s**)**

	OK
Write Command Send variable-length data when **<service_type>** is "TCP", "UDP" or "TCPINCOMING". **AT+QISEND=<connectID>**	Response **>** After the response **>**, input the data to be sent. Tap "**CTRL+Z**" to send, and tap **Esc** to cancel the operation. If the connection is established and the buffer of sent data is not full: **SEND OK** If the connection is established and the buffer of sent data is full: **SEND FAIL** If there is any error: **ERROR**
Write Command Send fixed-length data when **<service_type>**="TCP", "UDP" or "TCP INCOMING". **AT+QISEND=<connectID>,<send_length>**	Response **>** After the response **>**, input the data which length equals to **<send_length>**. If the connection is established and the buffer of sent data is not full: **SEND OK** If the connection is established and the buffer of sent data is full: **SEND FAIL** If there is any error: **ERROR**
Write Command **When** <service_type>**="UDP SERVICE".** **AT+QISEND=<connectID>,<send_length>,<remoteIP>,<remote_port>**	Response This command sends fixed-length data to a specified remote IP address and remote port. The **<service_type>** must be "UDP SERVICE" **>** After the response **>**, input the data which length equals to **<send_length>**. If the connection is established and the buffer of sent data is not full: **SEND OK**

	If the connection is established and the buffer of sent data is full: **SEND FAIL** If there is any error: **ERROR**
Write Command When **<send_length>** is 0, query the sent data. **AT+QISEND=<connectID>,0**	Response If the specified connection exists: **+QISEND: <total_send_length>,<ackedbytes>,<unacked bytes>** **OK** If there is any error: **ERROR**
Maximum Response Time	/
Characteristic	/

Parameter

<connectID>	Integer type. Socket ID. Range: 0–11.
<send_length>	Integer type. The length of sent data. Range: 0–1460. Unit: byte.
<remoteIP>	String type. The remote IP address (must be dot format). It is valid only when **<service_type>** is "UDP SERVICE".
<remote_port>	Integer type. Remote port. It is only valid when **<service_type>** is "UDP SERVICE".
<total_send_length>	Integer type. The total length of sent data. Unit: byte.
<ackedbytes>	Integer type. The total length of received data. Unit: byte.
<unackedbytes>	Integer type. The total length of unreceived data. Unit: byte.

2.3.6.AT+QIRD Read the Received TCP/IP Data

This command reads the received TCP/IP data. In buffer access mode, the data is buffered and **+QIURC: "recv",<connectID>** is reported first when the module receives the data. And then you can execute **AT+QIRD** to read the data.

When there is the data in the buffer, **+QIURC: "recv",<connectID>** is not reported if the module receives the data again. And **+QIURC: "recv",<connectID>** is reported until all the data in the buffer is read.

AT+QIRD Read the Received TCP/IP Data	
Test Command **AT+QIRD=?**	Response **+QIRD: (**range of supported **<connectID>**s**),(**range of

	supported **<read_length>**s) **OK**
Write Command When **<service_type>** is "TCP"/"UDP"/"TCP INCOMING" **AT+QIRD=<connectID>[,<read_length>]**	Response Socket connection is established and the module receives the data: **+QIRD: <read_actual_length><CR><LF><data>** **OK** The module does not receive the data: **+QIRD: 0** **OK** If there is any error: **ERROR**
Write Command When **<service_type>** is "UDP SERVICE" **AT+QIRD=<connectID>**	Response The module receives the data: **+QIRD: <read_actual_length>,<remoteIP>,<remote_port>,<CR><LF><data>** **OK** The module does not receive the data: **+QIRD: 0** **OK** If there is any error: **ERROR**
Write Command When **<read_length>** is 0, query the length of the read data **AT+QIRD=<connectID>,0**	Response **+QIRD: <total_receive_length>,<have_read_length>,<unread_length>** **OK** If there is any error: **ERROR**
Maximum Response Time	/
Characteristics	/

Parameter

<connectID>	Integer type. Socket ID. Range: 0–11.
<read_length>	Integer type. Length of data to be read. Range: 0–1500. Unit: byte.
<read_actual_length>	Integer type. Length of actually read data. Unit: byte.
<remoteIP>	String type. Remote IP address. It is valid only when **<service_type>** is "UDP SERVICE".
<remote_port>	Integer type. Remote port. It is valid only when **<service_type>** is "UDP SERVICE".
<data>	Integer type. Actually read data.
<total_receive_length>	Integer type. Total length of received data. Unit: byte.
<have_read_length>	Integer type. Length of read data. Unit: byte.
<unread_length>	Integer type. Length of unread data. Unit: byte.

2.3.7. AT+QIACCEPT Accept/Reject Remote Incoming Connection Request from TCP/UDP Socket Service

This command accepts/rejects remote incoming connection request from TCP/UDP socket service

AT+QIACCEPT Accept/Reject Remote Incoming Connection Request from TCP/UDP Socket Service	
Test Command **AT+QIACCEPT=?**	Response **+QIACCEPT: (**range of supported **<listener_socketID>**s**),(**list of supported **<accept>**s**),(**range of supported **<incoming_socketID>**s**)** **OK**
Write Command Accept/Reject incoming connection request **AT+QIACCEPT=<listener_socketID>,<accept>[,<incoming_socketID>]**	Response **[+QIACCEPT: <incoming_socketID>,<remote_addr>,<remote_port>]** **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<listener_socketID>	Integer type. Socket ID of TCP server. Range: 0–11.

<accept>	Integer type. Accept or reject remote incoming connection request of TCP/UDP socket service. 0 Reject 1 Accept
<incoming_socketID>	Integer type. Socket ID for incoming connection. It is valid only when **<accept>** is 1. Range: 0–11.
<remote_addr>	String type. Source address of incoming connection.
<remote_port>	Integer type. Source port of incoming connection.

2.3.8. AT+QISWTMD Switch Data Access Mode

This command switches data access mode.

AT+QISWTMD Switch Data Access Mode	
Test Command **AT+QISWTMD=?**	Response **+QISWTMD: (**range of supported **<connectID>**s**),(**range of supported **<access_mode>**s**)** **OK**
Write Command **AT+QISWTMD=<connectID>,<access_mode>**	Response If **<access_mode>**=0 or 1 and data access mode has been switched successfully: **OK** If **<access_mode>**=2 and the data access mode has been switched successfully, the module will enter the data mode: **CONNECT** If there is any error: **ERROR**
Maximum Response Time	/
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<connectID>	Integer type. Socket ID. Range: 0–11.
<access_mode>	Integer type. Data access mode of SSL connection. 0 Buffer access mode. Socket sends and receives data with AT commands. 1 Direct push mode. Socket sends data in AT command and receives data in URC format.

	2	Transparent transmission mode. Serial port is exclusively used for sending/receiving data directly to/from the Internet.

2.3.9. AT+QIGETERROR Query the Last Error Code

If **ERROR** is returned after executing TCP/IP AT commands, the details of error can be queried via **AT+QIGETERROR**. Please note that **AT+QIGETERROR** just returns error code of the last TCP/IP AT command.

AT+QIGETERROR Query Error Code Related to Last TCP/UDP Socket Service AT Command	
Test Command **AT+QIGETERROR=?**	Response **OK**
Execution Command **AT+QIGETERROR**	Response **+QIGETERROR: <err>,<errcode_description>** **OK**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<err>	Integer type. Error code. See ***Chapter 5*** for details.
<errcode_description>	String type. Error code description. See ***Chapter 5*** for details.

2.3.10. ATO Enter Transparent Transmission Mode

This command enables the module to enter transparent transmission mode.

ATO Enter Transparent Transmission Mode	
Execution Command **ATO**	Response **CONNECT** Or **NO CARRIER**
Maximum Response Time	300 ms
Characteristic	/

NOTE

If the socket connection has not been established before, **ATO** returns **NO CARRIER**.

2.3.11. +++ Exit Transparent Transmission Mode

This command enables the module to exit transparent transmission mode.

+++ Exit Transparent Transmission Mode	
Execution Command **+++**	Response **OK**
Maximum Response Time	300 ms
Characteristics	/

NOTE

After exiting transparent transmission mode with **+++**, if the Socket connection is active, the connection enters transparent transmission mode again with **ATO**.

2.4. Description of SSL-Related AT Commands

2.4.1. AT+QSSLCFG Set SSL Context Parameters

This command sets SSL context parameters.

AT+QSSLCFG Set SSL Context Parameters	
Test Command **AT+QSSLCFG=?**	Response **+QSSLCFG: "sslversion",(**range of supported **<SSL_ctxID>**s**),(**range of supported **<SSL_version>**s**)** **+QSSLCFG: "ciphersuite",(**range of supported **<SSL_ctxID>**s**),(** range of supported **<cipher_suites>**s**)** **+QSSLCFG: "seclevel",(**range of supported **<SSL_ctxID>**s**),(**range of supported **<seclevel>**s**)** **+QSSLCFG: "ignorelocaltime",(**range of supported **<SSL_ctxID>**s**),(**list of supported **<ignore_ltime>**s**)** **+QSSLCFG: "negotiatetime",(**range of supported **<SSL_ctxID>**s**),(**range of supported **<negotiate_time>**s**)** **+QSSLCFG: "sni",(**range of supported **<SSL_ctxID>**s**),(**list of supported **<SNI>**s**)**

	+QSSLCFG: "session_cache",(range of supported **<SSL_ctxID>**s**),(**list of supported **<session_cache_enable>**s**)** **OK**
Write Command Set SSL version for the specific SSL context: **AT+QSSLCFG="sslversion",<SSL_ctxID>[,<SSL_version>]**	Response If the optional parameter is omitted, query the current setting: **+QSSLCFG: "sslversion",<SSL_ctxID>,<SSL_version>** **OK** If the optional parameter is specified, set the SSL version for the specific SSL context: **OK** If there is any error: **ERROR**
Write Command Set the cipher suite for the specified SSL context: **AT+QSSLCFG="ciphersuite",<SSL_ctxID>[,<cipher_suites>]**	Response If the optional parameters is omitted, query the current setting: **+QSSLCFG: "ciphersuite",<SSL_ctxID>,<cipher_suites>** **OK** If the optional parameter is specified, set the cipher suite for the specified SSL context: **OK** If there is any error: **ERROR**
Write Command Set the authentication mode for the specified SSL context: **AT+QSSLCFG="seclevel",<SSL_ctxID>[,<seclevel>]**	Response If the optional parameter is omitted, query the current setting: **+QSSLCFG: "seclevel",<SSL_ctxID>,<seclevel>** **OK** If the optional parameter is specified, set the authentication mode for the specified SSL context: **OK** If there is any error: **ERROR**
Write Command Set whether the specified SSL context ignores certificate validity verification: **AT+QSSLCFG="ignorelocaltime",<**	Response If the optional parameter is omitted, query the current setting: **+QSSLCFG: "ignorelocaltime",<SSL_ctxID>,<ignore_ltime>**

SSL_ctxID>[,<ignore_ltime>]	 **OK** If optional parameter is specified, set whether the specified SSL context ignores certificate validity verification: **OK** If there is any error: **ERROR**
Write Command Set specified SSL context negotiation timeout: **AT+QSSLCFG="negotiatetime",<SSL_ctxID>[,<negotiate_time>]**	Response If the optional parameter is omitted, query the current setting: **+QSSLCFG: "negotiatetime",<SSL_ctxID>,<negotiate_time>** **OK** If the optional parameter is specified, set specified SSL context negotiation timeout: **OK** If there is any error: **ERROR**
Write Command Enable or disable server name indication of specified SSL context: **AT+QSSLCFG="sni",<SSL_ctxID>[,<SNI>]**	Response If the optional parameter is omitted, query the current setting: **+QSSLCFG: "sni",<SSL_ctxID>,<SNI>** **OK** If the optional parameter is specified, enable or disable server name indication of specified SSL context: **OK** If there is any error: **ERROR**
Write Command Enable or disable SSL session buffer: **AT+QSSLCFG="session_cache",<SSL_ctxID>[,<session_cache_enable>]**	Response If the optional parameter is omitted, query the current setting: **+QSSLCFG: "session_cache",<SSL_ctxID>,<session_cache_enable>** **OK** If the optional parameter is specified, enable or disable SSL session buffer: **OK**

	If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<SSL_ctxID>	Integer type. SSL context ID. Range: 0–5.	
<SSL_version>	Integer type. SSL version.	
	0	SSL3.0
	1	TLS1.0
	2	TLS1.1
	3	TLS1.2
	<u>4</u>	All
<cipher_suites>	Integer type. Hexadecimal value. SSL cipher suite.	
	0X0035	TLS_RSA_WITH_AES_256_CBC_SHA
	0X002F	TLS_RSA_WITH_AES_128_CBC_SHA
	0X0005	TLS_RSA_WITH_RC4_128_SHA
	0X000A	TLS_RSA_WITH_3DES_EDE_CBC_SHA
	0X003D	TLS_RSA_WITH_AES_256_CBC_SHA256
	0XC003	TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA
	0XC004	TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
	0XC005	TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA
	0XC008	TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
	0XC009	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
	0XC00A	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
	0XC011	TLS_ECDHE_RSA_WITH_RC4_128_SHA
	0XC012	TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
	0XC013	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
	0XC014	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
	0XC00D	TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA
	0XC00E	TLS_ECDH_RSA_WITH_AES_128_CBC_SHA
	0XC00F	TLS_ECDH_RSA_WITH_AES_256_CBC_SHA
	0XC023	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
	0XC024	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
	0XC025	TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256
	0XC026	TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
	0XC027	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
	0XC028	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
	0xC029	TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256
	0XC02A	TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384

	0XC02F TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 0XC030 TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 0XFFFF Support all SSL cipher suites.
<ignore_ltime>	Integer type. Whether to ignore certificate validity verification. 0 Do not ignore 1 ignored
<seclevel>	String type. Authentication mode. 0 No authentication mode 1 One way- Perform server authentication 2 Two way- Server and client authentication
<negotiate_time>	Integer type. Negotiation timeout. Range: 10–300. Default value: 300. Unit: second.
<SNI>	Integer type. Enable/disable server name indication. 0 Disable 1 Enable
<session_cache_enable>	Integer type. Enable/disable SSL session buffer. 0 Disable 1 Enable

2.4.2. AT+QSSLOPEN Open SSL Socket to Connect Remote Server

This command establishes an SSL connection, that is, open an SSL socket to connect to the remote server.

You need to execute **AT+QSTAAPINFO** first to connect to the Wi-Fi network, and then execute **AT+QSSLOPEN**. After waiting for the specified time (refer to the maximum response time below), URC **+QSSLOPEN: <clientID>,<err>** will be output. If no URC response is received within this period, you can use **AT+QSSLCLOSE** to close the SSL connection.

AT+QSSLOPEN Open SSL Socket to Connect Remote Server	
Test Command **AT+QSSLOPEN=?**	Response **+QSSLOPEN: (**range of supported **<SSL_ctxID>**s**),(**range of supported **<clientID>**s**),<server_address>,<server_port>,(**range of supported **<access_mode>**s**)** **OK**
Write Command **AT+QSSLOPEN=<SSL_ctxID>,<connectID>,<server_address>,<server_port>[,<access_mode>]**	Response If **<access_mode>**=2 and SSL connection is established: **CONNECT** If there is any error: **ERROR** If **<access_mode>**=0/1:

	OK **+QSSLOPEN: <connectID>,<err>** If there is any error: **ERROR**
Maximum Response Time	Maximum network response time is 150 seconds, and it needs to add the time configured in **<negotiate_time>**
Characteristic	This command takes effect immediately; The configurations are not saved.

Parameter

<SSL_ctxID>	Integer type. SSL context ID. range: 0–5.
<connectID>	Integer type. Socket ID. Range: 0–11.
<server_address>	String type. Remote server address.
<server_port>	Integer type. Remote server listening port.
<access_mode>	Integer type. Access mode for SSL connection. 0 Buffer access mode. Socket sends and receives data with AT commands. 1 Direct push mode. Socket sends data in AT command and receives data in URC format. 2 Transparent transmission mode. Serial port is exclusively used for sending/receiving data directly to/from the Internet.
<err>	Integer type. Error code. See ***Chapter 5*** for details.
<negotiate_time>	Integer type. Negotiation timeout. Range: 10–300; Default Value: 300; Unit: second

2.4.3.AT+QSSLSEND Send Data Through SSL Connection

This command sends data through SSL Socket connection.

AT+QSSLSEND Send Data Through SSL Connection	
Test Command **AT+QSSLSEND=?**	Response **+QSSLSEND: (**range of supported **<connectID>**s**),(**range of supported **<send_length>**s**)** **OK**
Write Command Send the data with variable length. **AT+QSSLSEND=<connectID>**	Response **>** After responded with **>**, enter the data needed to be sent. Use **CTRL+Z** to send and use **ESC** to exit. SSL Socket already opened and data sent successfully: **SEND OK**

	SSL Socket opened successful but buffer is full already: **SEND FAIL** If there is any error: **ERROR**
Write Command Send the data with fixed length. **AT+QSSLSEND=<connectID>,<send_length>**	Response **>** After responded with **>**, enter data to be sent with data length up to **<send_length>** SSL Socket is already opened and data sent successfully: **SEND OK** SSL Socket is opened successfully but buffer is full already: **SEND FAIL** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristic	The command takes effect immediately. The configurations are not saved.

Parameter

<connectID>	Integer type. Socket ID. Range: 0–11.
<send_length>	Integer type. Length of data to be sent. Range:1–1460; Unit: byte.

NOTE

Maximum length for the sent data with fixed or non-fixed length is 1460 bytes.

2.4.4. AT+QSSLRECV Read the Received Data via SSL Connection

When the data access mode of the SSL connection is buffer access mode, the module will report URC **+QSSLURC: "recv",<clientID>** when it receives the data sent by the network. Buffered data can be read through **AT+QSSLRECV**.

AT+QSSLRECV Read the Received Data via SSL Connection	
Test Command	Response

AT+QSSLRECV=?	**+QSSLRECV: (**range of supported **<connectID>**s**), (**range of supported **<read_length>**s**)** **OK**
Write Command **AT+QSSLRECV=<connectID>,<read_length>**	Response If the specified Socket connection receives the data: **+QSSLRECV: <have_readlen><CR><LF><data>** **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristic	The command takes effect immediately. The configurations are not saved.

Parameter

<connectID>	Integer type. Socket ID. Range: 0–11.
<read_length>	Integer type. The maximum length of the read data. Range:1–1500; Unit: byte.
<have_readlen>	Integer type. Length of the actually read data. Unit: byte.
<data>	String type. The actually read data.

2.4.5. AT+QSSLCLOSE Close SSL Connectio

This command closes SSL connection.

AT+QSSLCLOSE Close SSL Connection	
Test Command **AT+QSSLCLOSE=?**	Response **+QSSLCLOSE: (**range of supported **<connectID>**s**),(**range of supported **<close_timeout>**s**)** **OK**
Write Command **AT+QSSLCLOSE=<connectID>[,<close_timeout>]**	Response **OK** Or **ERROR**
Maximum Response Time	Depends on the time configured in **<close_timeout>**
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<connectID>	Integer type. Socket ID. Range: 0–11.
<close_timeout>	Integer type. Timeout of closing SSL connection. Range: 0–65535; Default value: 10. Unit: second.

2.4.6. AT+QSSLSTATE Query Socket Connection Status

This command queries the Socket connection status.

AT+QSSLSTATE Query Socket Connection Status	
Test Command **AT+QSSLSTATE=?**	Response **OK**
Query Command **AT+QSSLSTATE?**	Response Returns the connection status of all existing SSL. **[+QSSLSTATE: <connectID>,"SSLClient",<IP_address>,<remote_port>,<local_port>,<socket_state>,<serverID>,<access_mode>,<devname>,<SSL_ctxID>]** **[…]** **OK**
Write Command If **<query_type>**is 0, query the connection status of a specific context. **AT+QSSLSTATE=<query_type>,<connectID>**	Response Returns the status of all existing SSL connections for a specific context. **[+QSSLSTATE: <connectID>,"SSLClient",<IP_address>,<remote_port>,<local_port>,<socket_state>,<serverID>,<access_mode>,<devname>,<SSL_ctxID>]** **[…]** **OK**
Write Command If **<query_type>** is 1, query the connection status of a specific Socket service. **AT+QSSLSTATE=<query_type>,<connectID>**	Response **+QSSLSTATE: <connectID>,"SSLClient",<IP_address>,<remote_port>,<local_port>,<socket_state>,<serverID>,<access_mode>,<devname>,<SSL_ctxID>** **OK**
Execution Command **AT+QSSLSTATE**	Response Returns the connection status of all existing SSL. **[+QSSLSTATE: <connectID>>,"SSLClient",<IP_address>,<remote_port>,<local_port>,<socket_state>,<serverID>,<access_mode>,<devname>,<SSL_ctxID>]** **[…]**

	OK
Maximum Response Time	300 ms
Characteristic	/

Parameter

<connectID>	Integer type. Socket ID. Range: 0–11.
<IP_address>	String type. Remote server address.
<remote_port>	Integer type. Remote server socket ID. Range:0–65535.
<local_port>	Integer type. Local Socket ID. Range:0–65535.
<socket_state>	Integer type. SSL connection status. 0 "Initial" Connection is not established 1 "Opening" Client is connecting 2 "Connected" Client connection is established. 4 "Closing" Connection is closing.
<serverID>	Reserved.
<access_mode>	Integer type. Data access mode of SSL connection. 0 Buffer access mode. Socket sends and receives data with AT commands. 1 Direct push mode. Socket sends data in AT command and receives data in URC format. 2 Transparent access mode. Serial port is exclusively used for sending/receiving data directly to/from the Internet.
<devname>	String type. Socket COM port of the service "uart1" UART port 1 "uart2" UART port 2
<SSL_ctxID>	Integer type. SSL context ID. Range:0–5.
<query_type>	Integer type. Query type 0 Query the connection status of a specific context. 1 Query the connection status of a specific Socket service.

2.5. Description of MQTT-Related AT Commands

2.5.1. AT+QMTCFG Set Optional Parameters of MQTT

This command sets the optional parameters of MQTT.

AT+QMTCFG Set Optional Parameters of MQTT	
Test Command **AT+QMTCFG=?**	Response **+QMTCFG:"version",(**range of supported

	<client_idx>s)**,(**list of supported **<vsn>**s**)** **+QMTCFG: "ssl",(**range of supported **<client_idx>**s)**,(**list of supported **<SSL_enable>**s)**,(**range of supported **<SSL_ctx_idx>**s**)** **+QMTCFG:"keepalive",(**range of supported **<client_idx>**a)**,(**range of supported **<keep_alive_time>**s**)** **+QMTCFG:"session",(**range of supported **<client_idx>**s)**,(**list of supported **<clean_session>**s**)** **+QMTCFG:"timeout",(**range of supported **<client_idx>s),(**range of supported **<pkt_timeout>s),(**range of supported **<retry_times>),(**list of supported **<timeout_notice>**s**)** **+QMTCFG: "will",(**range of supported **<client_idx>**s)**,(**list of supported **<will_flag>**s)**,(**range of supported **<will_qos>s),(**list of supported **<will_retain>**s)**,<will_topic>,<will_message>** **+QMTCFG: "recv/mode",(**range of supported **<client_idx>s),(**list of supported **<msg_recv_mode>**s**)** **+QMTCFG: "dataformat",(**range of supported **<client_idx>s),(**list of supported **<send_mode>**s)**,(**list of supported **<recvmode>**s**)** **OK**
Write Command Set MQTT protocol version **AT+QMTCFG="version",<client_idx>[,<vsn>]**	Response If the optional parameter is omitted, query the current setting: **+QMTCFG: "version",<vsn>** **OK** If the optional parameter is specified, set the MQTT protocol version: **OK** If there is any error: **ERROR**
Write Command Set MQTT SSL mode and SSL context index. **AT+QMTCFG="ssl",<client_idx>[,<SSL_enable>[,<SSL_ctx_idx>]]**	Response If the optional parameters are omitted, query the current setting: **+QMTCFG: "ssl",<SSL_enable>[,<SSL_ctxID>]** **OK** If optional parameters are specified, set the MQTT SSL mode and SSL context index:

	OK If there is any error: **ERROR**
Write Command Set keep-alive time. **AT+QMTCFG="keepalive",<client_idx>[,<keep_alive_time>]**	Response If the optional parameter is omitted, query the current setting: **+QMTCFG: "keepalive",<keep_alive_time>** **OK** If optional parameter is specified, set the keep-alive time: **OK** If there is any error: **ERROR**
Write Command Set session type. **AT+QMTCFG="session",<client_idx>[,<clean_session>]**	Response If the optional parameter is omitted, query the current setting: **+QMTCFG: "session",<clean_session>** **OK** If optional parameter is specified, set the session type: **OK** If there is any error: **ERROR**
Write Command Set message transmission timeout. **AT+QMTCFG="timeout",<client_idx>[,<pkt_timeout>,<retry_times>,<timeout_notice>]**	Response If the optional parameters are omitted, query the current setting: **+QMTCFG: "timeout",<pkt_timeout>,<retry_times>,<timeout_notice>** **OK** If optional parameters are specified, set the message transmission timeout: **OK** If there is any error: **ERROR**
Write Command Set Will Information. **AT+QMTCFG="will",<client_idx>[,<will_flag>[,<will_qos>,<will_retain>,<wi**	Response If the optional parameters are omitted, query the current setting: **+QMTCFG: "will",<will_flag>[,<will_qos>,<will_retain>,<**

ll_topic>,<will_message>]]	**will_topic>,<will_message>]** **OK** If optional parameters are specified, set Will information: **OK** If there is any error: **ERROR**
Write Command Set the reception mode of MQTT client. **AT+QMTCFG="recv/mode",<client_idx>[,<msg_recv_mode>]**	Response If the optional parameter is omitted, query the current setting: **+QMTCFG: "recv/mode",<msg_recv_mode>** **OK** If optional parameter is specified, set the reception mode of MQTT client: **OK** If there is any error: **ERROR**
Write Command Set MQTT data format. **AT+QMTCFG="dataformat",<client_idx>[,<send_mode>,<recv_mode>]**	Response If the optional parameters are omitted, query the current setting: **+QMTCFG: "dataformat",<send_mode>,<recv_mode>** **OK** If optional parameters are specified, set the MQTT data format: **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<client_idx>	Integer type. MQTT client identifier. Range :0–5.
<vsn>	Integer type. MQTT Protocol version 3 MQTT protocol v3.1

	4 MQTT protocol v3.1.1
<SSL_enable>	Integer type. Set MQTT SSL type.
	0 Enable normal SSL connection
	1 Enable SSL TCP safety connection
<SSL_ctx_idx>	Integer type. SSL context ID. Range: 0–5.
<keep_alive_time>	Integer type. Keep alive time. Range: 0–3600; Default value: 120; Unit: seconds. This parameter defines the maximum interval between receiving messages from the client. Within 1.5 times the set time, if the server does not receive a message from the client, the client sends a DISCONNECT message by default, so the server will disconnect the client.
<clean_session>	Integer type. Session type.
	0 After a client disconnects, the server saves the client's subscription messages.
	1 After the client disconnects, the server must delete any previously retained messages from the client and the connection status is "Clean".
<pkt_timeout>	Integer type. Packet transmission timeout. Range: 1–1200; Default value: 5; Unit: seconds.
<retry_times>	Integer type. Number of retransmissions after data packet transmission times out. Range: 0–10; Default value: 3.
<timeout_notice>	Integer type. Whether to report a timeout message when transmitting data packets
	0 No reporting
	1 Reporting enabled
<will_topic>	String type. Will topic name. Range of Will topic name length: 1–256. Unit: byte.
<will_message>	String type. Message content of Will topic. Range of Will topic message content length. Unit: byte.
<will_flag>	Integer type. Whether to configure Will Flag.
	0 No Will Flag configuration
	1 Configure Will Flag
<will_qos>	Integer type. QoS level when sending messages.
	0 Send at most once
	1 Send at least once
	2 Send just once
<will_retain>	Integer type. Will Retain tag only applies to PUBLISH messages.
	0 When the client publishes a PUBLISH message to the server and the message is successfully sent to the current subscriber, the server does not retain the message.
	1 When the client publishes a PUBLISH message to the server, after the message is successfully sent to the current subscriber, the server retains the message
<will_length>	Integer type. The length of Will message. Range: 0–256; Unit: bytes.
<msg_recv_mode>	Integer type. MQTT message reception mode.

	0 MQTT messages received from the server are reported in the form of URC. 1 MQTT messages received from the server are not reported in the form of URC.
<send_mode>	Integer type. MQTT message sending format. 0 String type 1 Hexadecimal
<recvmode>	Integer type. MQTT message receiving format. 0 String type. 1 Hexadecimal

NOTE

1. If **<will_fg>**=1, Then **<will_qos>**, **<will_retain>**, **<will_topic>** and **<will_message>** must be specified; if **<will_flag>** is not 1, the above parameters will be omitted.
2. <**clean_session>**=0 is only valid when the server supports the operation of storing session information.
3. If the MQTT connection is configured in SSL mode, **<SSL_ctxID>** must be specified. In addition, during the MQTT SSL handshake process, you need to configure the SSL context parameters through **AT+QSSLCFG**.

2.5.2. AT+QMTOPEN Open MQTT Client Network

This command opens the MQTT client network.

AT+QMTOPEN Open MQTT Client Network	
Test Command **AT+QMTOPEN=?**	Response **+QMTOPEN: (**range of supported **<client_idx>**s**),"hostname",(**range of supported **<port>**s**)** **OK**
Read Command **AT+QMTOPEN?**	Response **[+QMTOPEN: <client_idx>,<host_name>,<port>]** **[…]** **OK** If there is any error: **ERROR**
Write Command **AT+QMTOPEN=<client_idx>,<host_name>,<port>**	Response **OK**

	+QMTOPEN: <client_idx>,<result> If there is any error: **ERROR**
Maximum Response Time	120 seconds. Affected by network status.
Characteristic	/

Parameter

<client_idx>	Integer type. MQTT Client identifier. Range: 0–5.
<host_name>	String type. Server address, which can be an IP address or domain name. Range of server address length: 0–100. Unit: byte.
<port>	Integer type. Server port. Range: 1–65535.
<result>	Integer type. Command execution result -1 Failed execution 0 Successful execution 1 Parameter error 2 MQTT identifier is occupied 3 Network error 4 Failed to parse domain name 5 Network disconnection causing error 6 Insufficient memory

2.5.3. AT+QMTCLOSE Close MQTT Client Network

This command closes the MQTT client network.

AT+QMTCLOSE Close MQTT Client Network	
Test Command **AT+QMTCLOSE=?**	Response **+QMTCLOSE: (**range of supported **<client_idx>**s**)** **OK**
Write Command **AT+QMTCLOSE=<client_idx>**	Response **OK** **+QMTCLOSE: <client_idx>,<result>** If there is any error: **ERROR**
Maximum Response Time	30 s

Characteristic	/

Parameter

<client_idx>	Integer type. MQTT Client identifier. Range:0–5.
<result>	Integer type. Command execution result. -1 Failed execution 0 Successful execution

2.5.4. AT+QMTCONN Connect a Client to MQTT Server

This command connects a client to MQTT server.

AT+QMTCONN Connect a Client to MQTT Server	
Test Command **AT+QMTCONN=?**	Response **+QMTCONN: (**range of supported **<client_idx>**s)**,<clientID>,<username>,<password>** **OK**
Query Command **AT+QMTCONN?**	Response **[+QMTCONN: <client_idx>,<state>]** **[…]** **OK** If there is any error: **ERROR**
Write Command **AT+QMTCONN=<client_idx>,<clientID>[,<username>,<password>]**	Response **OK** **+QMTCONN: <client_idx>,<result>[,<retcode>]** If there is any error: **ERROR**
Maximum Response Time	**<pkt_timeout>** value (default value: 5 seconds). Affected by network status
Characteristic	/

Parameter

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

<state>	Integer type. MQTT connection status. 1 MQTT initialization 2 MQTT connecting 3 MQTT already connected successfully 4 MQTT disconnecting
<clientID>	String type. client identifier.
<username>	String type. Client username, which can be used for authorization.
<password>	String type. The password corresponding to the client username, which can be used for authorization.
<result>	Integer type. Command execution result. 0 The data packet is sent successfully and the ACK from the server is received 1 Packet retransmission 2 Packet sending failed
<ret_code>	Integer type. Connection status return code. 0 Accept connection 1 Connection refused: Unaccepted protocol version 2 Connection refused: Identifier refused 3 Connection refused: Server is unavailable 4 Connection refused: Wrong username or password 5 Connection refused: Unauthorized
<pkt_timeout>	Integer type. Packet transmission timeout. Range: 1–60. Default value: 5. Unit: second. The timeout can be configured through **AT+QMTCFG="timeout",<client_idx>[,<pkt_timeout>,<retry_times>,<timeout_notice>]**.

NOTE

If a client ID is already connected to the server and another client reconnects using the same ID, the server must disconnect the original client before making a TCP connection with the new client.

2.5.5. AT+QMTDISC Disconnect MQTT Client from Server

This command disconnects MQTT client from server.

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

	+QMTDISC: <client_idx>,<result> If there is any error: **ERROR**
Maximum Response Time	30 s
Characteristic	/

Parameter

<client_idx>	Integer type. MQTT Client identifier. Range: 0–5.
<result>	Integer type. Command execution result -1 Failed execution 0 Successful execution

2.5.6.AT+QMTSUB Subscribe to Topics

This command subscribes to one or more topics. The client sends a SUBSCRIBE message to subscribe to one or more topics from the server. When the subscribed topic publishes messages, the server will transmit these messages to the client as PUBLISH messages.

AT+QMTSUB Subscribe to Topics	
Test Commands **AT+QMTSUB=?**	Response **+QMTSUB: (**range of supported **<client_idx>s),(**range of supported **<msgID>s),<topic>,(**range of supported **<qos>**s**)** **OK**
Write Command **AT+QMTSUB=<client_idx>,<msgID>,<topic1>,<qos1>[,<topic2>,<qos2>[,..]]**	Response **OK** **+QMTSUB: <client_idx>,<msgID>,<result>[,<value>]** If there is any error: **ERROR**
Maximum Response Time	**<pkt_timeout>** × **<retry_times>** (default value: 15 seconds), Affected by network status
Characteristic	/

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<msgID>	Integer type. SUBSCRIBE message identifier. Range: 0–65535.
<topic>	String type. Topics that the client subscribes to.
<qos>	Integer type. QoS level of messages published by the client 0 Send once at most 1 Send once at least 2 Send just once
<result>	Integer type. Command execution result 0 SUBSCRIBE message was sent successfully and SUBACK message was received. 1 SUBSCRIBE message was sent successfully but SUBACK message was not received within the specified response time. Retransmission was executed. 2 Failed to send SUBSCRIBE message.
<value>	Integer type. If **<result>**=0, it indicates QoS of SUBSCRIBE message reply. If **<result>**=1, it indicates the number of SUBSCRIBE message retransmissions. If **<result>**=2, it indicates this parameter is meaningless and the filed is empty.
<pkt_timeout>	Integer type. Packet transmission timeout. Range: 1–60. Default value: 5. Unit: s. The timeout can be configured through **AT+QMTCFG="timeout",<client_idx>[,<pkt_timeout>,<retry_times>,<timeout_notice>]**.
<retry_times>	Integer type. Number of retries after failed packet transmission. Range: 0–10; Default value: 3

NOTE

<msgID> is only displayed in messages where the QoS bits in the fixed header indicate a QoS level of 1 or 2. **<msgID>** must be unique within a set of in-flight messages in a specific communication direction. Generally speaking, the parameter value increases one by one according to the number of messages, but the actual situation does not require this.

2.5.7.AT+QMTUNS Unsubscribe from Topics

This command unsubscribes from one or more topics. The client sends an UNSUBSCRIBE message to the server to unsubscribe from the specific topic.

AT+QMTUNS Unsubscribe from Topic	
Test Command **AT+QMTUNS=?**	Response **+QMTUNS: (**range of supported **<client_idx>**s**),(**range of supported **<msgID>**s**),<topic>**

	OK
Write Command **AT+QMTUNS=<client_idx>,<msgID>,<topic1>[,<topic2>[,..]]**	Response **OK** **+QMTUNS: <client_idx>,<msgID>,<result>[,<value>]** If there is any error: **ERROR**
Maximum Response Time	**<pkt_timeout>** × **<retry_times>** (default value: 15 s). Affected by network status
Characteristics	/

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<msgID>	Integer type. UNSUBSCRIBE message identifier. Range: 0–65535.
<topic>	String type. The topic that the client unsubscribes from.
<result>	Integer type. Command execution result 0 UNSUBSCRIBE message was sent successfully and UNSUBACK message was received. 1 UNSUBSCRIBE message was sent successfully but UNSUBACK message was not received within the specified response time. Retransmission was executed. 2 Failed to send UNSUBSCRIBE message.
<value>	Integer type. If **<result>**=0, it indicates QoS of SUBSCRIBE message reply. If **<result>**=1, it indicates the number of SUBSCRIBE message retransmissions If **<result>**=2, it indicates this parameter is meaningless and the filed is empty.
<pkt_timeout>	Integer type. Packet transmission timeout. Range: 1–60; Default value: 5; Unit: seconds. The timeout can be configured through **AT+QMTCFG="timeout",<client_idx>[,<pkt_timeout>,<retry_times>,<timeout_notice>]**.
<retry_times>	Integer type. Number of retries after failed packet transmission. Range: 0–10; Default value: 3

2.5.8. AT+QMTPUB Publish a Message

The client can publish fixed-length messages to the server through this command, and the server will then distribute them to interested subscribers. Each PUBLISH message is associated with a topic name. If a client subscribes to one or more topics, when the subscribed topic publishes messages, the server will transmit these messages to the client as PUBLISH messages.

AT+QMTPUB Publish a Message	
Test Command	Response

AT+QMTPUB=?	**+QMTPUB: (**range of supported **<client_idx>**s**),(**range of supported **<msgID>**s**),(**range of supported **<qos>**s**),(**list of supported **<retain>**s**),<topic>,(**range of supported **<length>**s**)** **OK**
Write Command **AT+QMTPUB=<client_idx>,<msgID>,<qos>,<retain>,<topic>[,<length>]**	Response **>** After **>** is responded, input the data to be sent. If **<length>** is specified, the actual length of the data is greater than **<length>**, and the excess bytes will be deleted. **OK** If **<length>** is not specified, then enter **CTRL+Z** (**CTRL+Z** will not be included in sent data). **OK** **+QMTPUB: <client_idx>,<msgID>,<result>[,<value>]** If there is any error: **ERROR**
Maximum Response Time	**<pkt_timeout>** × **<retry_times>** (default value: 15 s). Affected by network status.
Characteristic	/

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<msgID>	Integer type. PUBLISH message identifier. Range :0–65535.
<qos>	Integer type. QoS level of messages published by the client. <u>0</u> Send once at most 1 Send once at least 2 Send just once
<retain>	Integer type. After the message is sent to the current subscriber, whether the server saves the message <u>0</u> Do not save the message 1 Save the message
<topic>	String type. The topic to be published
<length>	Integer type. The data length of the message to be published. Range: 1–2048. Unit: byte.
<result>	Integer type. Command execution result. 0 PUBLISH message was sent successfully and ACK message was received. 1 PUBLISH message was sent successfully but ACK message was not received

	within delivery time. Retransmission was executed. 2 Failed to send PUBLISH message.
<value>	Integer type If **<result>**=1, it indicates the number of PUBLISH message retransmissions. If **<result>**=0 or 2, it indicates this parameter is meaningless and the filed is empty.
<pkt_timeout>	Integer type. Data transfer timeout. Range: 1–60; Default value: 5; Unit: s. The timeout can be configured through **AT+QMTCFG="timeout",<client_idx>[,<pkt_timeout>,<retry_times>,<timeout_notice>]**.
<retry_times>	Integer type. Number of retries after failed packet transmission. Range: 0–10. Default value: 3.

NOTE

1. If the command is executed successfully and the response is **OK**, the client can continue to publish new data packets. The maximum number of transmitted data packets cannot be greater than the size of the sliding window (the size of the sliding window is 5), otherwise **ERROR** will be returned.
2. After the command is executed, the client can send data, that is, payload information. The maximum length of data sent each time is 2048 bytes. If it exceeds, please send it in segments.
3. The publisher can publish PUBLISH messages to the server, and the server can also publish PUBLISH messages to subscribers. When the server publishes a message to the subscriber, it will return a URC to notify the Host to read the data sent by the MQTT server: **+QMTRECV: <client_idx>,<msgID>,<topic>[,<payload_length>],<payload>**. For details on URCs please refer to ***Chapter*** *错误!未找到引用源。* **.**

2.5.9. AT+QMTRECV Read Messages from Buffer

This command reads messages in the storage buffer. When the server reports a message, it will store the message in the buffer.

AT+QMTRECV Read Messages from Buffer	
Test Command AT+QMTRECV=?	Response **OK**
Query Command **AT+QMTRECV?**	Response **[+QMTRECV: <client_idx>,<store_status0>,<store_status1>,<store_status2>,<store_status3>,<store_status4>]** **[…]** **OK** If there is any error: **OK**

Write Command **AT+QMTRECV=<client_idx>[,<receiveID>]**	Response If the optional parameter is omitted, read all buffered messages of the specified client: **[+QMTRECV: <client_idx>,<msgID>,<topic>,<payload_len>,<payload>]** **[…]** **OK** If the optional parameter is specified, read the messages specified by **<receiveID>** of the specified client: **+QMTRECV: <client_idx>,<msgID>,<topic>,<payload_len>,<payload>** **OK** If there is any error: **ERROR**
Maximum Response Time	/
Characteristic	/

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<store_status0>	Integer type. Indicates whether a message stored in buffer corresponds to storeID0. 0 No message in buffer 1 One or more messages are stored in buffer
<store_status_>	Integer type. Indicates whether a message stored in buffer corresponds to storeID1. 0 No message in buffer 1 One or more messages are stored in the buffer
<store_status2>	Integer type. Indicates whether a message stored in buffer corresponds to storeID2. 0 No message in buffer 1 One or more messages are stored in buffer
<store_status3>	Integer type. Indicates whether a message stored in buffer corresponds to storeID3. 0 No message in buffer 1 One or more messages are stored in buffer
<store_status4>	Integer type. Indicates whether a message stored in buffer corresponds to storeID4. 0 No message in buffer 1 One or more messages stored in buffer
<receiveID>	Integer type. ID of messages stored in buffer. Range: 0–4.
<msgID>	Integer type. PUBLISH message identifier. Range: 0–65535. Only when **<qos>**=0, **<msgID>**=0.

<topic>	String type. Topic to be published.
<payload_len>	Integer type. Length of received message. Range: 0–10240. Unit: byte.
<payload>	String type. Received message.

2.6. Description of HTTP Related AT Commands

2.6.1. AT+QHTTPCFG Configure HTTP(S) Parameters

This command configures HTTP(S) server parameters, including configuring PDP context ID, customizing HTTP(S) request header information, outputting HTTP(S) response header information, and configuring SSL context ID. If optional parameters are omitted when executing the Write Command, it means querying the current configuration.

AT+QHTTPCFG Configure HTTP(S) Parameters	
Test Command **AT+QHTTPCFG=?**	Response **+QHTTPCFG: "url",<URL_string>** **+QHTTPCFG: "header",<header_value>** **+QHTTPCFG: "auth",<username_password>** **+QHTTPCFG: "sslctxid",(**range of supported **<sslctxID>**s**)** **+QHTTPCFG: "rsp/header",(**list of supported **<response_header>**s**)** **+QHTTPCFG: "rspout/auto",(**list of supported **<auto_outrsp>**s**)** **+QHTTPCFG: "closed/ind",(**list of supported **<closedind>**s**)** **+QHTTPCFG: "form/option",<name>,<file_name>,<content_type>** **+QHTTPCFG: "reset"** **OK**
Write Command **AT+QHTTPCFG="url"[,<URL_string>]**	Response If optional parameters are omitted, query the current configuration: **+QHTTPCFG: "url",<URL_string>** **OK** If optional parameters are specified, configure the HTTP(S) URL: **OK** If there is any error:

	+CME ERROR: <err>
Write Command **AT+QHTTPCFG="header"[,<header_value>]**	Responses If optional parameters are omitted, query the current configuration: **+QHTTPCFG: "header",<header_value>** **[…]** **OK** If optional parameters are specified, configure the HTTP(S) request header line/header field: **OK** If there is any error: **+CME ERROR: <err>**
Write Commands **AT+QHTTPCFG="auth"[,<username_password>]**	Response If optional parameters are omitted, query the current configuration: **+QHTTPCFG: "auth",<username_password>** **OK** If optional parameters are specified, configure the username and password: **OK** If there is any error: **+CME ERROR: <err>**
Write Command **AT+QHTTPCFG="sslctxid"[,<sslctxID>]**	Response If optional parameters are omitted, query the current configuration: **+QHTTPCFG: "sslctxid",<sslctxID>** **OK** If optional parameters are specified, configure the SSL context ID used for HTTP(S): **OK** If there is any error: **+CME ERROR: <err>**
Write Command **AT+QHTTPCFG="rsp/header"[,<resp**	Response If optional parameters are omitted, query the current

onse_header>]	configuration: **+QHTTPCFG: "rsp/header",<response_header>** **OK** If the optional parameter is specified, disables or enables the output of HTTP(S) response header information. **OK** If there is any error: **+CME ERROR: <err>**
Write Command **AT+QHTTPCFG="rspout/auto"[,<auto_outrsp>]**	Response If optional parameters are omitted, query the current configuration: **+QHTTPCFG: "rspout/auto",<auto_outrsp>** **OK** If optional parameters are specified, disables or enables automatic output of HTTP(S) response header information: **OK** If there is any error: **+CME ERROR: <err>**
Write Command **AT+QHTTPCFG="closed/ind"[,<closedind>]**	Response If optional parameters are omitted, query the current configuration: **+QHTTPCFG: "closed/ind",<closedind>** **OK** If optional parameters are specified, disable or enable reporting of HTTP(S) session close URC **+QHTTPURC: "closed"**: **OK** If there is any error: **+CME ERROR: <err>**
Write Command **AT+QHTTPCFG="form/option"[,<name>[,<file_name>[,<content_type>]]]**	Response If optional parameters are omitted, query the current configuration: **+QHTTPCFG: "form/option",<name>,<file_name>,<content_type>**

	[…] **OK** If any optional parameters are specified, configure the parameter values of form/option: **OK** If there is any error: **+CME ERROR: <err>**
Write Command **AT+QHTTPCFG="reset"**	Response **OK** Or **+CME ERROR: <err>**
Maximum Response Time	/
Characteristic	The command takes effect immediately; The configurations are not saved.

Parameter

<response_header>	Integer type. Disable or enable output of HTTP(S) response headers. 0 Disable 1 Enable
<sslctxID>	Integer type. SSL context ID for HTTP(S). Range: 0–5; Default value: 1. Configure SSL parameters through **AT+QSSLCFG**. See ***Chapter 2.4.1*** for details.
<auto_outrsp>	Integer type. Disable or enable automatic output of HTTP(S) response header information. If automatic output of HTTP(S) response header information is enabled, **AT+QHTTPREAD** and **AT+QHTTPREADFILE** will fail to execute. 0 Disable 1 Enable
<closedind>	Integer type. Disable or enable reporting of HTTP(S) session closed URC **+QHTTPURC: "closed"**. 0 Disable 1 Enable
<URL_string>	String type. HTTP(S) URL.
<header_value>	String type. HTTP(S) request header line/header field name, such as: "Content-type: text/plain" or "Content-type".
<username_password>	String type. Username and password. The format is "username:password".
<name>	String type. The name value in form/option.
<file_name>	String type. The file name value in form/option.

<content_type>	String type. The content-type value in form/option.
<err>	Error codes. See ***Chapter 1*** for details.

2.6.2. AT+QHTTPGET Send GET Request to HTTP(S) Server

This command sends GET request to HTTP(S) server. After executing AT+QHTTPGET Write Command and OK is returned, it is recommended to wait for a specific period of time (determined by the maximum response time) for URC +QHTTPGET: <err>[,<httprspcode>[,<content_length>]] to be outputted. <httprspcode> can only be reported when <err> is 0. If HTTP(S) response header contains Content-Length, then <content_length> information will be reported.

AT+QHTTPGET Send GET Request to HTTP(S) Server	
Test Command **AT+QHTTPGET=?**	Response **+QHTTPGET: (**range of supported **<rsptime>**s**)** **OK**
Write Command **AT+QHTTPGET[=<rsptime>]**	Response If the parameter format is correct and no other errors occur: **OK** After the module receives the response from the HTTP(S) server, it will report the following URC: **+QHTTPGET: <err>[,<httprspcode>[,<content_length>]]** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	depends on **<rsptime>**
Characteristic	The command takes effect immediately; The configuration is not saved.

Parameter

<rsptime>	Integer type. Timeout value for URC **+QHTTPGET: <err>[,<httprspcode>[,<content_length>]]** to be outputted. Range: 1–65535; Default: 60; Unit: second.
<httprspcode>	HTTP(S) server response code. See ***Chapter 1*** for details.
<content_length>	Integer type. HTTP(S)Response length. Unit: byte.
<err>	Error codes. See *Chapter 1* for details.

2.6.3.AT+QHTTPPOST Send POST Request to HTTP(S) Server

This command sends a POST request to the HTTP(S) server. After sending AT+QHTTPPOST, if CONNECT is returned within 125 seconds, it means that the HTTP(S) server connection is successful; if CONNECT is not returned within 125 seconds, +CME ERROR: <err> will be output.

After executing **AT+QHTTPPOST** Write Command and **OK** is returned, it is recommended to wait for a specific period of time (determined by the maximum response time) for **+QHTTPPOST: <err>[,<httprspcode>[,<content_length>]]** to be outputted.

AT+QHTTPPOST Send POST Request to HTTP(S) Server	
Test Command **AT+QHTTPPOST=?**	Response **+QHTTPPOST: (**range of supported **<data_length>**s**),(**range of supported **<input_time>**s**),(**range of supported **<rsptime>**s**)** **OK**
Write Command **AT+QHTTPPOST=<data_length>[,<input_time>,<rsptime>]**	Response If the parameter format is correct, the HTTP(S) server connection is successful and the HTTP(S) request header information is sent: **CONNECT** TA switches to transparent transmission mode (that is, data mode) and can enter the HTTP(S) POST request body. When the total size of input data reaches **<data_length>**, TA will switch back to command mode and report the following results: **OK** After the module receives the response from the HTTP(S) server, it will report the following URC: **+QHTTPPOST: <err>[,<httprspcode>[,<content_length>]]** If the input time reaches **<input_time>**, but the received data length is less than **<data_length>**, TA will switch back to command mode and report the following results: **+QHTTPPOST: <err>** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	Depends on network and **<rsptime>**
Characteristic	The command takes effect immediately;

	The configurations are not saved.

Parameter

<data_length>	Integer type. If **<request_header>** is 0, Indicates the length of the POST request body; if **<request_header>** is 1, it indicates the length of the HTTP(S) request information, including the HTTP(S) POST request header information and the HTTP(S) POST request body. Range: 1–1024000; Unit: Bytes.
<input_time>	Integer type. The maximum input time for POST request body or HTTP(S) request information. Range: 1–65535; Default value: 60; Unit: seconds
<rsptime>	Integer type. After **OK** is returned, this parameter is the maximum output time of HTTP(S) POST response **+QHTTPPOST: <err>[,<httprspcode>[,<content_length>]]**. Range :1–65535, default value:60. Unit: second.
<httprspcode>	HTTP(S) server response code. See ***Chapter 1*** for details.
<content_length>	Integer type. HTTP(S) response length. Unit: byte.
<err>	Error codes. See *Chapter 1* for details.

2.6.4.AT+QHTTPPUT Send PUT Request to HTTP(S) Server

This command sends a PUT request to the HTTP(S) server. After sending AT+QHTTPPUT, if CONNECT is returned within 125 seconds, it means that the HTTP(S) server connection is successful; if CONNECT is not returned within 125 seconds, +CME ERROR: <err> will be output.

After executing AT+QHTTPPUT and OK is returned, it is recommended to wait for a specific period of time (determined by the maximum response time) for +QHTTPPUT: <err>[,<httprspcode>[,<content_length>]] to be outputted.

AT+QHTTPPUT Send PUT Request to HTTP(S) Server	
Test Command **AT+QHTTPPUT=?**	Response **+QHTTPPUT: (**range of supported **<data_length>**s**),(**range of supported **<input_time>**s**),(**range of supported **<rsptim e>**s**)** **OK**
Write Command **AT+QHTTPPUT=<data_length>[,<input_time>,<rsptime>]**	Response If the parameter format is correct, the HTTP(S) server connection is successful and the HTTP(S) request header information is sent. **CONNECT** TA switches to transparent transmission mode (that is, data mode) and can enter the HTTP(S) PUT request body. When

	the total size of input data reaches **<data_length>**, TA will return to command mode and report the following results: **OK** After the module receives the response from the HTTP(S) server, it will report the following URC: **+QHTTPPUT: <err>[,<httprspcode>[,<content_length>]]** If the input time reaches **<input_time>**, but the received data length is less than **<data_length>**, TA will return to command mode and report the following results: **+QHTTPPUT: <err>** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	Depends on network and **<rsptime>**
Characteristic	The command takes effect immediately; The configurations are not saved.

Parameter

<data_length>	Integer type. If **<request_header>** is 0, it indicates the length of the PUT request body; if **<request_header>** is 1, it indicates the length of the HTTP(S) request information, including the HTTP(S) PUT request header information and the HTTP(S) PUT request body. Range: 1–1024000; Unit: Bytes
<input_time>	Integer type. The maximum input time for PUT request body or HTTP(S) request information. Range: 1–65535; Default value: 60; Unit: seconds
rsptime>	Integer type. After **OK** is returned, this parameter is the maximum output time of HTTP(S) PUT response **+QHTTPPUT: <err>[,<httprspcode>[,<content_length>]]**. Range: 1–65535; Default value: 60; Unit: second.
<httprspcode>	HTTP(S) server response code. See ***Chapter 1*** for details.
<content_length>	Integer type. HTTP(S) response body length. Unit: Bytes.
<err>	Error code. See ***Chapter 1*** for details.

2.6.5.AT+QHTTPREAD Read HTTP(S) Server Response Information

After sending an HTTP(S) GET/POST/PUT request, you can use AT+QHTTPREAD to read the HTTP(S) response information returned by the server. It must be executed after any one of the following URCs is received:

- **+QHTTPGET: <err>[,<httprspcode>[,<content_length>]]**
- **+QHTTPPOST: <err>[,<httprspcode>[,<content_length>]]**
- **+QHTTPPUT: <err>[,<httprspcode>[,<content_length>]]**

AT+QHTTPREAD Read HTTP(S) Server Response Information	
Test Command **AT+QHTTPREAD=?**	Response **+QHTTPREAD: (**range of supported **<wait_time>**s**)** **OK**
Write Command **AT+QHTTPREAD[=<wait_time>]**	Response If the parameter format is correct and read successfully: **CONNECT** <Output HTTP(S) response information> **OK** When the response information reading is completed or the interval between receiving two data packets reaches **<wait_time>**: **+QHTTPREAD: <err>** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	depends on **<wait_time>**
Characteristic	The command takes effect immediately; The configuration is not saved.

Parameter

<wait_time>	Integer type. Maximum time between receiving two packets. Range :1–65535. Default value:60; Unit: second
<err>	Error code. See ***Chapter 1*** for details.

2.6.6.AT+QHTTPSTOP Stop HTTP(S) Request

MCU can use this command to cancel HTTP(S) GET/POST/PUT requests and disconnect the session with HTTP(S).

AT+QHTTPSTOP Stop HTTP(S) Request	
Test Command **AT+QHTTPSTOP=?**	Response **OK**
Execution Command **AT+QHTTPSTOP**	Response If the parameter format is correct and no other errors occur: **OK**

	If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	10 seconds
Characteristic	This command takes effect immediately; The configuration is not saved.

2.7. Description of AT+QPING AT Command

2.7.1. AT+QPING PING Remote Host IP

This command detects the reachability of the host network protocol. Before using AT+QPING, the host should first connect to the Wi-Fi network.

AT+QPING PING Remote Host IP	
Test Command **AT+QPING=?**	Response **+QPING: <host>,(**range of supported **<timeout>**s**),(**range of supported **<pingnum>**s**)** **OK**
Write Command **AT+QPING=<host>[,<timeout>[,<pingnum>]]**	Response If the remote host Ping is successful: **OK** **[+QPING: <result>[,<IP_address>,<bytes>,<time>,<ttl>]<CR><LF>…]** **[…]** **+QPING: <finresult>[,<sent>,<rcvd>,<lost>,<min>,<max>,<avg>]** If there is any error: **ERROR**
Maximum Response Time	/
Characteristic	/

Parameter

<host>	Integer type. Host address. The format is domain name or IP address.	
<timeout>	Integer type. The maximum waiting time for each Ping request response. Range: 1–255. Default value: 4; Unit: second.	
<pingnum>	Integer type. The maximum numbers of ping requests. Range: 1–10. Default value: 4.	
<ping_result>	Integer type. Result for every ping request	
	0	A Ping response is received from the remote host, followed by **,<IP_address>,<bytes>,<time>,<ttl>**.
	other	Error codes, see ***Chapter 1*** for details.
<IP_address>	String type. IP address of remote host. The format is dotted decimal IP.	
<bytes>	Integer type. The length of the response of the ping request. Unit: byte.	
<time>	Integer type. The time wait for response after sending a Ping request. Unit: ms	
<ttl>	Integer type. TTL value of the response packet for Ping request	
<finresult>	Integer type. The final result after executing this command	
	0	Works normally and the host is found. In this case, followed by **,<sent>,<rcvd>,<lost>,<min>,<max>,<avg>**
	Other	Error code, please refer to Chapter5 for details.
<sent>	Integer type. Number of Ping requests sent.	
<rcvd>	Integer type. Number of Ping requests that received responses	
<lost>	Integer type. Number of timed out Ping requests	
<min>	Integer type. Minimum response time. Unit: ms	
<max>	Integer type. Maximum response time. Unit: ms	
<avg>	Integer type. Average response time. Unit: ms	

2.8. Desscription of DNS related AT Command

2.8.1.AT+QIDNSGIP Get IP Address By Domain Name

Before querying DNS, the Host should first connect to the Wi-Fi network.

AT+QIDNSGIP Get IP Address By Domain Name	
Test Command **AT+QIDNSGIP=?**	Response **+QIDNSGIP: <hostname>** **OK**
Write Command **AT+QIDNSGIP=<hostname>**	Response **OK** If there is any error: **ERROR**

	Return results in URC format: **+QIURC: "dnsgip",<err>,<IP_count>,<DNS_ttl>** **[……** **+QIURC: "dnsgip",<hostIPaddr>]**
Maximum Response Time	Affected by network status, the maximum response time is 60 seconds.
Characteristic	/

Parameter

<hostname>	String type. Domain name
<err>	Error code. See ***Chapter 1*** for details.
<IP_count>	Integer type. The number of IP addresses corresponds to **<hostname>**.
<DNS_ttl>	Integer type. TTL value of DNS. Unit: second.
<hostIPaddr>	String type. IP address of <hostname>.

2.9. Description of FILE Related AT Commands

2.9.1.AT+QFOPEN Open a File

AT+QFOPEN Open a File	
Test Command **AT+QFOPEN=?**	Response **+QFOPEN: <filename>,(**range of supported **<mode>**s**)** **OK**
Query Command **AT+QFOPEN?**	Response **+QFOPEN: <filename>,<filehandle>,<mode>** **[+QFOPEN: <filename>,<filehandle>,<mode>** **[…]]** **OK**
Write Command **AT+QFOPEN=<filename>[,<mode>]**	Response **+QFOPEN: <filehandle>** **OK** If there is any error: **ERROR**

Maximum Response Time	300ms
Characteristic	The command takes effect immediately; The configuration is not saved.

Parameter

<filename>	String type. The name of the file to opened. Maximum length is 80 bytes "**<filename>**" File name in UFS memory
<filehandle>	Integer type. File handle. The data type is 4 bytes
<mode>	Integer type. File opening mode 0 If the file does not exist, a new file will be created; if the file exists, the file will be opened directly. The files in both cases are readable and writable. 1 If the file does not exist, a new file will be created; if the file exists, the file will be cleared and overwritten. The files in both cases are readable and writable. 2 If the file exists, it will be opened directly as a read-only file; if the file does not exist; an error will be responded.

2.9.2.AT+QFREAD   Read File

This command reads file data specified by the file handle. Data is read starting from the current position of the file pointer, which belongs to the file handle.

AT+QFREAD Read File	
Test Command **AT+QFREAD=?**	Response **+QFREAD: <filehandle>,<length>** **OK**
Write Command **AT+QFREAD=<filehandle>[,<length>]**	Response **CONNECT <read_length>** TA Switch to data mode. When the total data size exceeds **<length>** (Unit: bytes), TA will switch back to command mode: **OK** If there is any error: **ERROR**
Maximum Response Time	5 seconds
Characteristic	The command takes effect immediately; The configuration is not saved

Parameter

<filehandle>	Integer type. The file handle to be operated on.
<length>	Integer type. The expected length of the file to read, defaults to the file length
<read_length>	Integer type. Actual read file length

2.9.3. AT+QFWRITE Write Data to File

This command writes data to a file. Data is written starting from the current position of the file pointer, which belongs to the file handle.

AT+QFWRITE Write Data to File	
Test Command **AT+QFWRITE=?**	Response **+QFWRITE: <filehandle>,<length>,<timeout>** **OK**
Write Command **AT+QFWRITE=<filehandle>[,<length>[,<timeout>]]**	Response **CONNECT** TA switches to data mode. When the total size of written data exceeds **<length>** (Unit: bytes) or the writing time exceeds **<timeout>**, TA will switch back to command mode: **+QFWRITE: <written_length>,<total_length>** **OK** If there is any error: **ERROR**
Maximum Response Time	5 seconds
Characteristic	The command takes effect immediately; The configurations are not saved.

Parameter

<filehandle>	Integer type. The file handle to be operated on.
<length>	Integer type. The expected length of the file to be written, default length is 10 KB
<timeout>	Integer type. Waiting time for data input via USB/UART. Default value: 5; Unit: s.
<written_length>	Integer type. The length of the actual written data.
<total_length>	Integer type. Total file length.

2.9.4. AT+QFCLOSE Close the File

This command is used to close the file or end the operation on the file. When the file is closed, the file handle is released. The file handle cannot be used again unless the file is opened again via **AT+QFOPEN**.

AT+QFCLOSE Close the File	
Test Command **AT+QFCLOSE=?**	Response **+QFCLOSE: <filehandle>** **OK**
Write Command **AT+QFCLOSE=<filehandle>**	Response **OK** If there is any error: **ERROR**
Maximum Response Time	300ms
Characteristic	The command takes effect immediately; The configuration is not saved.

Parameter

<filehandle>	Integer type. The file handle to be closed.

2.9.5. AT+QFLST List File Information on Storage Medium

This command lists all file information in the specified storage medium.

AT+QFLST List File Information On Storage Medium	
Test Command **AT+QFLST=?**	Response **OK**
Write Command **AT+QFLST=<name_pattern>**	Response **+QFLST: <filename>,<file_size>** **[+QFLST: <filename>,<file_size>** **[…]]** **OK** If there is any error: **ERROR**

Execution Command **AT+QFLST**	Response Returns space information for all files stored in UFS: **+QFLST: <filename>,<file_size>** **[+QFLST: <filename>,<file_size>** **[…]]** **OK** If there is any error: **ERROR**
Maximum Response Time	300ms
Characteristic	The command takes effect immediately; The configuration is not saved.

Parameter

<name_pattern>	String type. Listed file types	
	"UFS:"	List all files in UFS
<filename>	String type. file name	
<file_size>	Integer type. File size. Unit: byte	

3 Description of URCs

3.1. Wi-Fi Related URCs

3.1.1.+QSTASTAT URC Indicating STA State Change

+QSTASTAT URC Indicating STA State Change	
+QSTASTAT: <event>	This URC indicates STA state changes.

Parameter

<event>	String type. Event reported when STA state changes. "WLAN DISCONNECTED" STA disconnected "WLAN_CONNECTED" STA connected "GOT_IP" STA got IP "SCAN_NO_AP" STA scanned no AP

3.1.2.+QAPSTAT URC Indicating AP State Change

+QAPSTAT URC Indicating AP State Change	
+QAPSTAT: <event>	This URC indicates AP status changes.

Parameter

<event>	String type. Event reported when AP state changes. "AP_UP" AP turned ON successfully "AP_DOWN" Close AP successfully "AP_CONNECT" STA connects to AP "AP_DISCONNECT" STA disconnects from AP

3.1.3.+QOTASTAT URC Indicating OTA State Change

+QOTASTAT URC Indicating OTA State Change	
+QOTASTAT: <event>	This URC indicates OTA upgrade status changes.

Parameter

<event>	String type. URC reported when OTA upgrade status changes.	
	"START_DOWNLOAD"	OTA download started
	"DOWNLOAD_SUCCEED"	OTA download succeed
	"DOWNLOAD_FAILED"	OTA download failed
	"START_UPGRADE"	OTA downloading

3.2. BLE Related URCs

3.2.1.+QBLESTAT URC Indicating BLE State Change

+QBLESTAT URC Indicating BLE State Change	
+QBLESTAT: <event>	This URC indicates BLE state changes.

Parameter

<event>	String type. URC reported when BLE status changes.	
	"CONNECTED"	BLE connected
	"DISCONNECTED"	BLE disconnected

3.2.2.+QBLEMTU URC Indicating BLE MTU Value Change

+QBLEMTU URC Indicating BLE MTU Value Change	
+QBLEMTU: <MTU_value>	This URC indicates BLE MTU value changes.

Parameter

<MTU_value>	Integer type. Maximum transmission unit value. Range: 23–512. Unit: byte

3.2.3.+QBLERECV URC Indicating BLE Device Receives Data

+QBLERECV URC Indicating BLE Device Receives Data	
+QBLERECV: <peer_role>,<UUID>,<Length>\r\n<data>\r\n	This URC indicates BLE device receives data.

Parameter

<peer_role>	Integer type. BLE connection peer role 0 The role of peer device is host 1 The role of the peer device is peripheral
<char_handle>	Integer type. Character handle.
<Length>	Integer type. Data length (The maximum data length does not exceed **<MTU_value>**-3, the length of **<MTU_value>** can be got through **+QBLEMTU**)
<data>	Received data.

3.2.4.+QBLEPEERROLE URC Indicating BLE Peer Role Received After Establishing Connection

+QBLEPEERROLE URC Indicating BLE Peer Role Received After Establishing Connection	
+QBLEPEERROLE: <peer_role>	This URC indicates peer role information received after BLE establishes connection.

Parameter

<peer_role>	Integer type. BLE connection peer role 0 The role of peer device is host 1 The role of the peer device is peripheral

3.3. TCP/UDP Related URCs

3.3.1.+QIURC: "closed" URC Indicating Connection Closed

When the TCP socket service is closed by the remote client or due to network error, the URC will be outputted, and the status of socket service will be "Closing" (**<socket_state>**=4). **AT+QICLOSE=<connectID>** can be used to change the **<socket_state>** back to "Initial".

+QIURC: "closed" URC Indicating Connection Closed	
+QIURC: "closed",<connectID>	This URC indicates socket service connection is closed.

Parameter

<connectID>	Integer type. Socket ID. Range: 0–11.

3.3.2.+QIURC: "recv" URC Indicating Incoming Data

In buffer mode or direct access mode, after receiving the data, the module will report the URC to notify the host.

- In buffer mode: The URC format is +QIURC: "recv",<connectID>; after the URC is reported, the host can retrieve the data through AT+QIRD. Please note that if the buffer is not empty, when the module receives data again, it will not report a new URC until all the received data has been retrieved from buffer through AT+QIRD.
- In direct mode: the received data will be outputted directly from the COM port.

+QIURC: "recv" URC Indicating Incoming Data	
+QIURC: "recv",<connectID>	This URC indicates the incoming data in buffer access mode. The host can receive data through **AT+QIRD**.
+QIURC: "recv",<connectID>,<currentrecvlength><CR><LF><data>	This URC indicates the incoming data in direct push mode when the **<service_type>** is "TCP", "UDP", or "TCP INCOMING".
+QIURC: "recv",<connectID>,<currentrecvlength>,<remoteIP>,<remote_port><CR><LF><data>	This URC indicates data incoming in direct push mode when **<service_type>** is "UDP SERVICE".

Parameter

<connectID>	Integer type. Socket ID. Range: 0–11.
<currentrecvlength>	Integer type. The length of the actual received data
<remoteIP>	String type. Remote IP address.
<remote_port>	Integer type. Remote port
<data>	String type. The received data.

3.3.3.+QIURC: "incoming full" URC Indicating Incoming Connection Full

If the incoming connection reaches the limit, or no socket system resources can be allocated, then the module will report the URC **+QIURC: "incoming full"** for the new incoming connection request.

+QIURC: "incoming full" URC Indicating Incoming Connection Full	
+QIURC: "incoming full"	This URC indicates the incoming connection is full.

3.3.4.+QIURC: "incoming" URC Indicating Incoming Client Connection

If **<service_type>** is "TCP LISTENER", when a remote client connects to this server, the host will automatically assign a free **<connectID>** for the new connection, where the range of **<connectID>** is 0–11. At this time, the module will report the URC. The **<service_type>** of the new connection is "TCP INCOMING", and the **<access_mode>** will be buffer access mode.

+QIURC: "incoming" URC Indicating Incoming Client Connection	
+QIURC: "incoming",<connectID>,<serverID>,<remoteIP>,<remote_port>	When the new incoming connection is accepted by **<serverID>**, the allocated **<connectID>**, **<remoteIP>** and **<remote_port>** will be informed by this URC.

Parameter

<connectID>	Integer type. The module automatically specifies the assigned socket service for the incoming client connections. Range: 0–11.
<serverID>	Integer type. The incoming **<connectID>** accepted by the server whose **<service_type>** is "TCP LISTENER", and the listening socket ID is **<serverID>**.
<remoteIP>	String type. The remote IP address of the client connection **<connectID>**
<remote_port>	Integer type. Remote port of the client connection **<connectID>**

3.4. SSL Related URCs

3.4.1.+QSSLURC: "recv" URC Indicating Incoming Data

The URC is reported when the SSL client receives data.

+QSSLURC: "recv" URC Indicating Incoming Data	
+QSSLURC: "recv",<clientID>	This URC reported when receiving data from the cache. SSL data can be received through **AT+QSSLRECV**.
+QSSLURC: "recv",<clientID>,<current_recvlength><CR><LF><data>	The URC reported when receiving data in direct access mode

Parameter

<clientID>	Integer type. Socket ID. Range: 0–11.
<current_recvlength>	Integer type. Actual received data length. Unit: byte.
<data>	The data to be read. Unit: byte

3.4.2. +QSSLURC: "closed" URC Indicating SSL Connection Closed

This URC indicates the host that the connection has been disconnected. Disconnection may be caused by a variety of reasons, such as the network closing the connection or GPRS PDP being deactivated. The SSL connection status of the specified socket may be "closing". In this case, **AT+QSSLCLOSE=<clientID>** needs to be executed to change the SSL connection status to "initial".

+QSSLURC: "closed" URC Indicating SSL Connection Closed	
+QSSLURC: "closed",<clientID>	This URC reported when the SSL connection based on the specified socket has been closed.

Parameter

<clientID>	Integer type. Socket ID. Range: 0–11.

3.5. MQTT Related URCs

3.5.1. +QMTSTAT URC Indicating MQTT Link Layer Status Change

The URC indicates MQTT link layer status changes. When the MQTT link layer status changes, the client will disconnect the MQTT connection and report this URC.

+QMTSTAT URC Indicating MQTT Link Layer Status Change	
+QMTSTAT: <client_idx>,<err_code>	When the MQTT link layer status changes, the client will disconnect the MQTT connection and report this URC.

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<err_code>	Integer type. Error codes. Please refer to the table below for details.

Table 2: Error Codes in URC

<err_code>	Description	Resolution
1	The connection was disconnected or reset by the server.	Execute **AT+QMTOPEN** to reconnect the MQTT connection.
2	Sending PINGREQ packet timed out or failed.	First deactivate the PDP, then activate the PDP and re-establish the MQTT connection.
3	Sending CONNECT packet timed out or failed.	1. Check whether the entered username and password are correct. Make sure the client ID was not already in use. 2. Re-establish the MQTT connection and try sending CONNECT packet again to the server.
4	Receiving CONNACK packet timed out or failed.	1. Check whether the entered username and password are correct. Make sure the client ID was not already in use. 2. Re-establish the MQTT connection and try sending CONNECT packet again to the server.
5	The client sends a DISCONNECT packet to the server, but the server actively disconnects the MQTT connection.	This is a normal process.
6	Because sending data packets always fails, the client actively disconnects MQTT connection.	1. Check whether the data sending through AT commands is consistent with that from the module. Make sure the data is correct. 2. Possibly due to network congestion or other errors, try to re-establish the MQTT connection.
7	The link is not working or the server is unavailable.	Make sure the current link or the server is available.
8	The client actively disconnects the MQTT connection.	Try to re-establish the MQTT connection.
9~255	Reserved value.	-

3.5.2.+QMTRECV URC Indicating Host to Read MQTT Packet Data

This URC indicates the host to read the data packet sent by the MQTT server.

+QMTRECV URC Indicating Host to Read MQTT Packet Data	
+QMTRECV:<client_idx>,<msgID>,<topic>[,<payload_lenth>],<payload>	Notify the host to read the data packet sent by the MQTT Server.
+QMTRECV: <client_idx>,<recv_id>	This URC is reported when the message received from the MQTT server is stored in buffer.

Parameter

<client_idx>	Integer type. MQTT client identifier. Range: 0–5.
<msgID>	integer type. PUBLISH message identifier. Range: 0–65535.
<topic>	String type. Topic received from MQTT server.
<payload_lenth>	Integer type. The length of the payload message. Range: 0–10240. Unit: byte.
<payload>	String type. Topic name related payload.
<recv_id>	Integer type. Indicates the sequence number of each piece of received data. Range: 0–4.

4 Examples

4.1. Examples of Wi-Fi Related AT Commands

```
AT+QSTAAPINFO="testssid","123456789"      //Set the module to STA mode and connect to AP
                                            hotspot.
OK
AT+QSOFTAP="testap","12345678",0          //Set the module to AP mode.
OK
```

4.2. Examples of BLE Related AT Commands

4.2.1.Peripheral Role

The BLE device, which accepts the request to establish an active physical connection, is a peripheral device. When the connection is established, the peripheral device operates as a slave in the link layer.

4.2.1.1. Set Module to a Peripheral Device

```
AT+QBLEINIT=2,0                    //Set the module as a peripheral device for initializing BLE.
OK
AT+QBLENAME="QuecFCM360K"          //Set BLE name.
OK
AT+QBLEADDR?                       //Query and obtain BLE address.
+QBLEADDR: "c8:e4:a1:b1:c1:f9"

OK
AT+QBLEGATTSSRV="fff1"             //Establish a BLE service and set the service UUID to 0xFFF1.
OK
AT+QBLEGATTSCHAR="fff2"            //Set GATT characteristic UUID to 0xFFF2.
OK
AT+QBLEGATTSSRVDONE                //Peripheral device completes creating BLE service.
OK
AT+QBLEADVPARAM=150,150            //Set BLE advertising parameters.
OK
```

```
AT+QBLEADVSTART                              //Start BLE advertising.
OK
```

4.2.1.2. nRF Connect Connectivity

1. Click "SCAN" to scan for peripheral devices, select "QuecFCM360W" in the scan results and then click "CONNECT", as shown in the figure below:

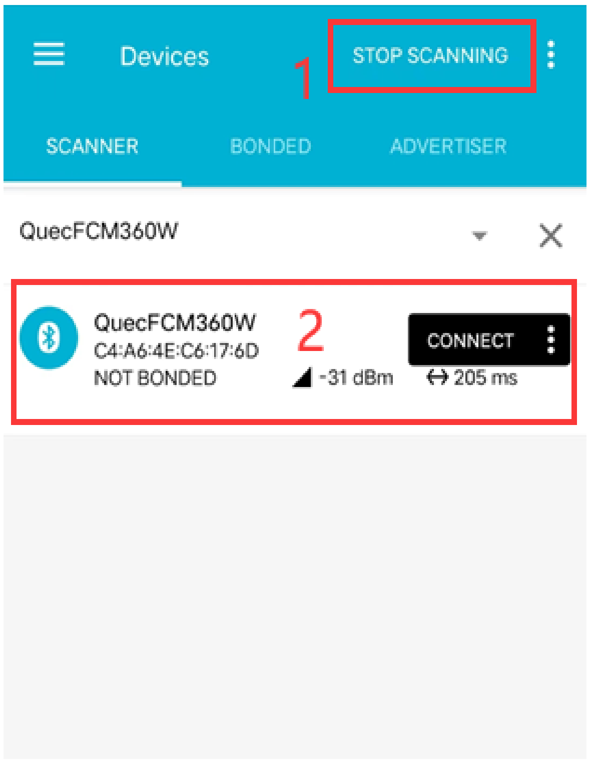
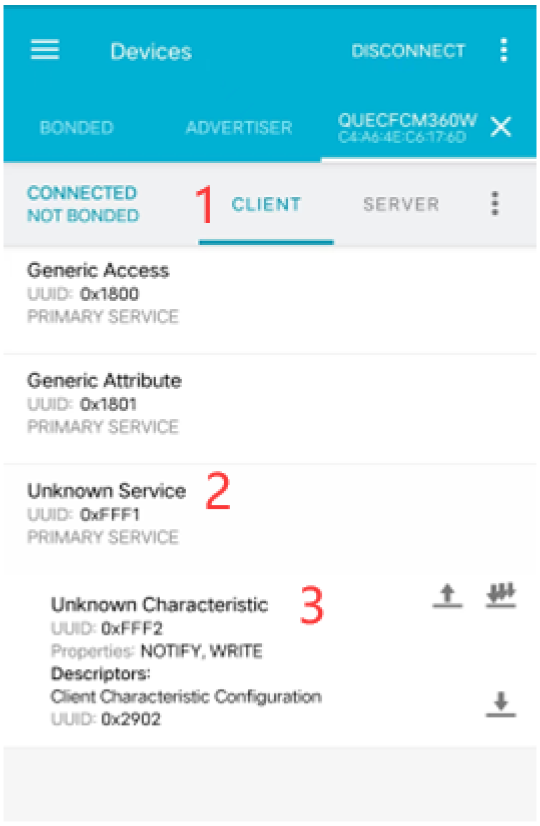


Figure 2: Scanning Result

2. After the module is connected successfully, "**CONNECTED**" is displayed in the interface, and the added UUID is displayed in "**CLIENT**":



Generic Access
UUID: 0x1800
PRIMARY SERVICE

Generic Attribute
UUID: 0x1801
PRIMARY SERVICE

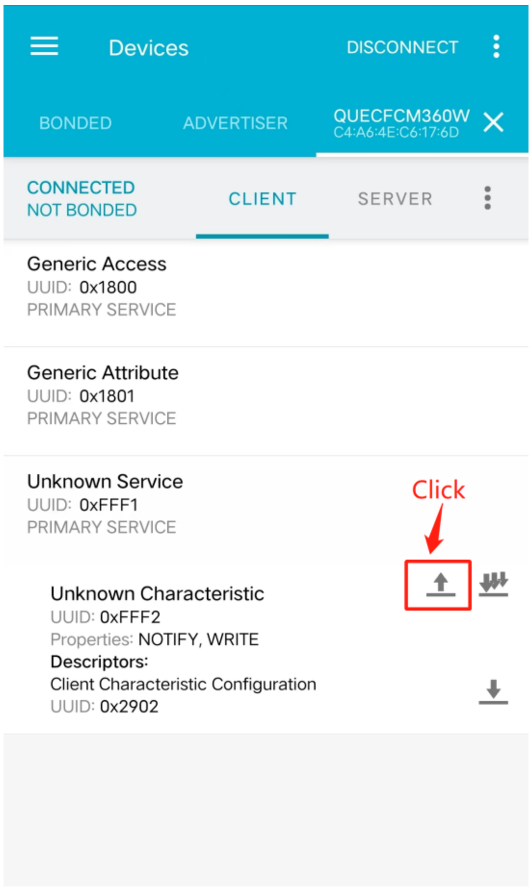
Unknown Service
UUID: 0xFFF1
PRIMARY SERVICE

Unknown Characteristic
UUID: 0xFFF2
Properties: NOTIFY, WRITE
Descriptors:
Client Characteristic Configuration
UUID: 0x2902

Figure 3: Connected Successfully

4.2.1.3. Sending Data to the Module

1. Select an editable characteristic, and click the up arrow:



Generic Access
UUID: 0x1800
PRIMARY SERVICE

Generic Attribute
UUID: 0x1801
PRIMARY SERVICE

Unknown Service
UUID: 0xFFF1
PRIMARY SERVICE

Unknown Characteristic
UUID: 0xFFF2
Properties: NOTIFY, WRITE
Descriptors:
Client Characteristic Configuration
UUID: 0x2902

Figure 4: Editing Characteristic

2. Input the data to be sent in "TEXT" format. Then click "**SEND**":

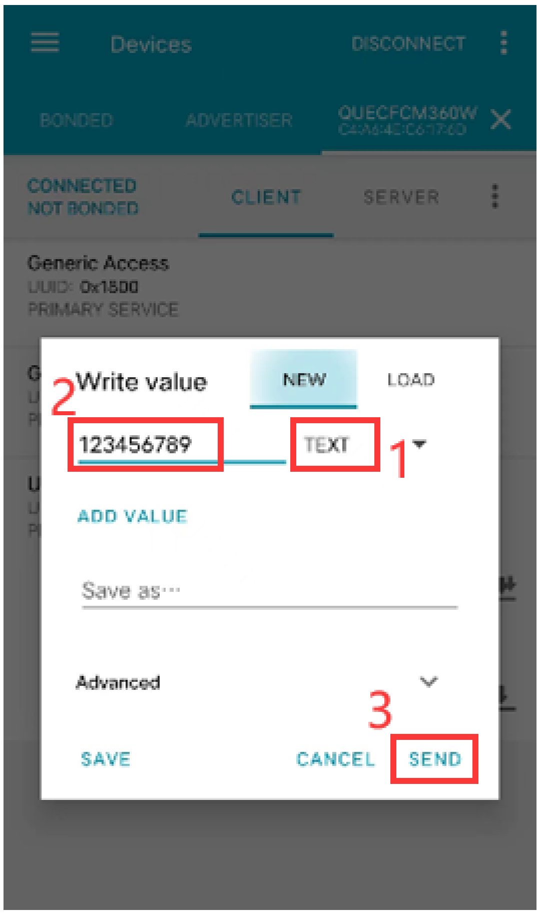


Figure 5: Sending Data

3. Once the data is sent successfully, QCOM tool receives the data:

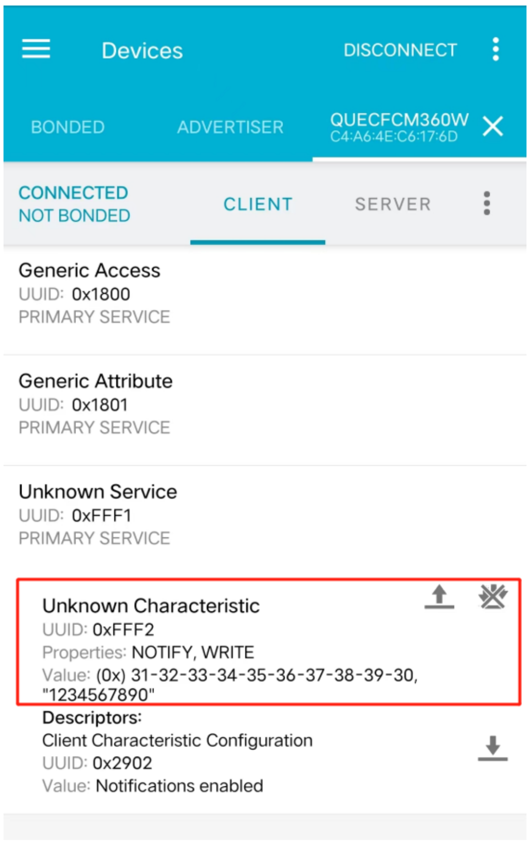
```
+QBLERECV: 0,"10",9
123456789
```

4.2.1.4. Sending data to nRF Connect

1. Enable the notification function of UUID 0xFFF2 and use QCOM tool to send data:

```
AT+QBLEGATTSNTFY="fff2","1234567890"
OK
```

2. The received data is displayed in nRF Connect UUID 0xFFF2.



Generic Access
UUID: 0x1800
PRIMARY SERVICE

Generic Attribute
UUID: 0x1801
PRIMARY SERVICE

Unknown Service
UUID: 0xFFF1
PRIMARY SERVICE

Unknown Characteristic
UUID: 0xFFF2
Properties: NOTIFY, WRITE
Value: (0x) 31-32-33-34-35-36-37-38-39-30, "1234567890"
Descriptors:
Client Characteristic Configuration
UUID: 0x2902
Value: Notifications enabled

Figure 6: Received Data in 0xFFF2

4.3. Examples of TCP/UDP Related AT Commands

4.3.1. Transparent Transmission Mode

Create a TCP client connection and enter transparent transmission mode.

AT+QIOPEN=0,"TCP","220.180.239.212",8009,0,2	//**<connectID>** is 0. Before **AT+QIOPEN** is executed, the host needs to use **AT+QIACT** to activate the scene.
CONNECT	//The connection is successful. It is recommended to wait 150 seconds for the URC response result **CONNECT**. If there is no URC response within 150 seconds, the host can disconnect the socket through **AT+QICLOSE**.

Send data in transparent transmission mode.

<All data got from COM port will be sent to internet directly>

Receive data from remote server in transparent transmission mode.

```
Test 1          //All data received from the network is output directly through the COM port.
```

Disconnect TCP client connection.

```
AT+QICLOSE=0                //After exiting transparent transmission mode through +++, the host can
                             disconnect the TCP connection through AT+QICLOSE. Affected by
                             network status, the maximum response time is 10 seconds.
OK
```

4.4. Examples of SSL Related AT Commands

4.4.1.Configure SSL Context ID

```
AT+QSSLCFG="sslversion",1,1             //Set both the SSL context ID and SSL version to 1.
OK
AT+QSSLCFG="ciphersuite",1,0X0035       //Set the SSL context ID to 1 and the SSL cipher
                                          suite to TLS_RSA_WITH_AES_256_CBC_SHA.
OK
AT+QSSLCFG="seclevel",1,1               //Set the SSL context ID to 1 and the
                                          authentication mode to server authentication.
OK
AT+QFOPEN="ca.pem"
+QFOPEN: 1

OK
AT+QFWRITE=1,1757
CONNECT

+QFWRITE: 1757,1757
//Configure the trusted CA certificate path for the SSL context
AT+QSSLCFG="cacert",1,"ca.pem"
OK
```

4.4.2.SSL Client in Buffer Mode

```
//Establish an SSL connection and enter buffer mode
AT+QSSLOPEN=1,4,"220.180.239.212",8010,0
OK
```

```
+QSSLOPEN: 4,0                          //SSL connection is successfully established.
AT+QSSLSTATE                            //Query the status of all SSL connections.
+QSSLSTATE: 4,"SSLClient","220.180.239.212",8010,65344,2,4,0,"uart1",1

OK
//Send data in buffer mode
AT+QSSLSEND=4                            //Send data with variable length.
>
Test data from SSL
<CTRL+Z>
SEND OK
AT+QSSLSEND=4,18                         //Send fixed-length data and the data length is 18 bytes.
>
Test data from SSL
SEND OK
//Receive data in buffer mode
+QSSLURC: "recv",4                  //Socket 4 (<clientID>=4) receives data.

AT+QSSLRECV=4,1500                      //Read data. The length of data to be read is 1500 bytes.
+QSSLRECV: 18                       //The length of the data actually received is 18 bytes.
Test data from SSL

OK
AT+QSSLRECV=4,1500
+QSSLRECV: 0                        //No data in buffer.
OK
//Close SSL connection
AT+QSSLCLOSE=4                      //Close the SSL connection (<clientID>=4). Depends on network,
                                    and the maximum response time is 10 seconds.
OK
```

4.4.3.SSL Client in Direct Access Mode

```
//Establish an SSL connection and enter direct access mode.
AT+QSSLOPEN=1,4,"220.180.239.212",8011,1
OK

+QSSLOPEN: 4,0                          //SSL connection is successfully established.
AT+QSSLSTATE                            //Query the status of all SSL connections.
+QSSLSTATE: 4,"SSLClient","220.180.239.212",8011,65047,2,4,1,"uart1",1

OK
```

```
//Send data in direct access mode
AT+QSSLSEND=4                   //Send data with variable length.
>
Test data from SSL
<CTRL+Z>
SEND OK
AT+QSSLSEND=4,18                //Send fixed-length data and the data length is 18 bytes.
>
Test data from SSL
SEND OK
//Receive data in direct access mode
+QSSLURC: "recv",4,18
Test data from SSL
//Close SSL connection
AT+QSSLCLOSE=4                  //Close the SSL connection (<clientID>=4). Depends on
                                network, and the maximum response time is 10 seconds.
OK
```

4.4.4.SSL Client in Transparent Transmission Mode

```
Establish an SSL connection and send data in transparent transmission mode
AT+QSSLOPEN=1,4,"220.180.239.212",8011,2     //Establish SSL connection
CONNECT                                      //Enter transparent transmission mode.
//The client sends data directly to the network through the COM port.
OK                                           //+++ can be used to exit transparent transmission
                                               mode. If the server disconnects the SSL
                                               connection abnormally, a result code NO
                                               CARRIER will be reported.
//Establish an SSL connection and receive data in transparent transmission mode
AT+QSSLOPEN=1,4,"220.180.239.212",8011,2     //Establish SSL connection
CONNECT
<Received data>                              //Client is reading data.
OK                                           //Use +++ to exit transparent transmission mode. If
                                               the server disconnects the SSL connection, a
                                               result code NO CARRIER will be reported.
//Close SSL connection
AT+QSSLCLOSE=4                               //Close the SSL connection (<clientID>=4).
                                               Depending on the network, the maximum response
                                               time is 10 seconds.
OK
```

4.5. Examples of MQTT Related AT Commands

4.5.1.Example of MQTT Operation Without SSL

```
//Configure receive mode
AT+QMTCFG="recv/mode",0,0
OK
AT+QMTOPEN=?
+QMTOPEN: (0-5),"hostname",(1-65535)

OK
//Open MQTT client network
AT+QMTOPEN=0,"112.31.84.164",8306
OK

+QMTOPEN: 0,0                                      //MQTT client has successfully opened the
                                                   network.
AT+QMTOPEN?
+QMTOPEN: 0,"112.31.84.164",8306

OK
AT+QMTCONN=?
+QMTCONN: (0-5),<clientID>,<username>,<password>

OK
//MQTT client connects to MQTT server
//If MQTT client was connected to Alibaba Cloud, you can use AT+QMTCFG="aliauth" to configure the
device information in advance, and then you can omit <username> and <password>.
AT+QMTCONN=0,"test","quectel","quectel"
OK

+QMTCONN: 0,0,0                                    //The client is successfully connected to the
                                                   MQTT server.
AT+QMTSUB=?
+QMTSUB: (0-5),(0-65535),<topic>,(0-2)

OK
//Subscribe to topic
AT+QMTSUB=0,1,"topic/example",2
OK

+QMTSUB: 0,1,0,2
AT+QMTSUB=0,1,"topic/pub",0
OK
```

```
+QMTSUB: 0,1,0,0
//If the client subscribes to a topic and other devices publish the same topic to the server, the module will
report the following information:
+QMTRECV: 0,0,"topic/example",36,"This is the payload related to topic"
//Unsubscribe from topic
AT+QMTUNS=0,2,"topic/example"
OK

+QMTUNS: 0,2,0
AT+QMTPUB=?
+QMTPUB: (0-5),(0-65535),(0-2),(0,1),<topic>,(1-2048)

OK
//After > is responded, enter This is test data, hello MQTT. and then send the data. The maximum length
of data is 1500 bytes, and excess bytes will be deleted.
AT+QMTPUB=0,0,0,0,"topic/pub",30
>This is test data, hello MQTT.
OK

+QMTPUB: 0,0,0

//If the client subscribes to a topic named "topic/pub" and other devices publish the same topic to the server,
the module will report the following information:
+QMTRECV: 0,0,"topic/pub",30,This is test data, hello MQTT.

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

+QMTDISC: 0,0                                   //Connection is disconnected successfully.
```

4.5.2.Example of MQTT Operation With SSL

```
//Configure receive mode
AT+QMTCFG="recv/mode",0,0
OK
//Configure the MQTT session in SSL mode
AT+QMTCFG="ssl",0,1,2
OK
//If the SSL authorization method is server authentication, store the CA certificate in the file system of the
module.
AT+QFOPEN="ca.pem"
+QFOPEN: 1
```

```
OK
AT+QFWRITE=1,1757
CONNECT

+QFWRITE: 1757,1757

OK
AT+QFCLOSE=1
OK
//If the SSL authorization method is server authentication, store the CC certificate in the file system of the
module.
AT+QFOPEN="user.pem"
+QFOPEN: 1

OK
AT+QFWRITE=1,1639
CONNECT

+QFWRITE: 1639,1639

OK
AT+QFCLOSE=1
OK
//If the SSL authorization method is server authentication, store the CK certificate in the file system of the
module.
AT+QFOPEN="user_key.pem"
+QFOPEN: 1

OK
AT+QFWRITE=1,2455
CONNECT

+QFWRITE: 2455,2455

OK
AT+QFCLOSE=1
OK
//Configure SSL parameters
AT+QSSLCFG="seclevel",2,2            //Configure SSL authorization method as server authentication.
OK
AT+QSSLCFG="sslversion",2,4          //Configure SSL authorization version.
OK
AT+QSSLCFG="ciphersuite",2,0xFFFF    //Configure cipher suite.
OK
```

```
AT+QSSLCFG="ignorelocaltime",2,1        //Ignore the authorization time.
OK
//Start MQTT SSL connection
AT+QMTOPEN=0,"112.31.84.164",8307
OK

+QMTOPEN: 0,0
//Connect to MQTT server
AT+QMTCONN=0,"test","quectel","quectel"
OK

+QMTCONN: 0,0,0
//Subscribe to topic
AT+QMTSUB=0,1,"$aws/things/M26_0206/shadow/update/accepted",1
OK

+QMTSUB: 0,1,0,1
//Publish message
AT+QMTPUB=0,1,1,0,"$aws/things/M26_0206/shadow/update/accepted",32
>This is publish data from client
OK

+QMTPUB: 0,1,0

//If the client subscribes to a topic named "$aws/things/M26_0206/shadow/update/accepted" and other
devices publish messages on the same topic to the server, the module will report the following information:
+QMTRECV: 0,1,"$aws/things/M26_0206/shadow/update/accepted",32,"This is publish data from
client"
//Client disconnects from MQTT server
AT+QMTDISC=0
OK

+QMTDISC: 0,0
```

4.6. Examples of HTTP(S) Related AT Commands

```
Send an HTTP GET request and read the response information
The following examples illustrate how to send an HTTP GET request, enable the output of HTTP response
header information, and read the HTTP GET response.
//Example of sending HTTP GET request.
AT+QHTTPCFG="rsp/header",1              //Enable the output of HTTP response header information.
OK
AT+QHTTPCFG="url","http://www.sina.com.cn"          //Set the URL to access.
```

```
OK
AT+QHTTPGET=80                          //Send an HTTP GET request with a maximum response time of
                                          80 seconds.
OK

+QHTTPGET: 0,200,601710             //If the HTTP response header information includes Content-Length,
                                    <content_length> information will be reported.

//Example of reading HTTP GET response information.
//Method 1: Read the HTTP response information and output it through the UART port.
AT+QHTTPREAD=80                         //Read the HTTP response information and output it through
                                        the UART port. Maximum wait time for HTTP session to be
                                        closed is 80 seconds.
CONNECT
HTTP/1.1 200 OK <CR><LF>                  //HTTP response header information and response body.
Server: nginx<CR><LF>
Date: Tue, 12 Sep 2017 05:57:29 GMT<CR><LF>
Content-Type: text/html<CR><LF>
Content-Length: 601710<CR><LF>
Connection: close<CR><LF>
Last-Modified: Tue, 12 Sep 2017 05:54:48 GMT<CR><LF>
Vary: Accept-Encoding<CR><LF>
Expires: Tue, 12 Sep 2017 05:58:28 GMT<CR><LF>
Cache-Control: max-age=60<CR><LF>
X-Powered-By: shci_v1.03<CR><LF>
Age: 1<CR><LF>
……<CR><LF>                                   //Response information is omitted here.
<CR><LF>
<body>
OK

+QHTTPREAD: 0                                //Successfully read HTTP response header information and
                                               response body.

//Send an HTTP POST request and read the response information
//The following examples illustrate how to send an HTTP POST request and how to read the HTTP POST
response information.
AT+QHTTPCFG="url","http://api.efxnow.com/DEMOWebServices2.8/Service.asmx/Echo?"
//Set the URL to access
OK
AT+QHTTPPOST=20,80,80                        //Send an HTTP POST request and obtain the POST request
                                             body through UART. The maximum input time and response
                                             time of the POST request body are both 80 seconds.
CONNECT
```

```
Message=HelloQuectel                    //Enter a POST request body of 20 bytes in length. (This
                                        POST request body is only an example. Please enter the
                                        correct POST request body according to the actual situation.)
OK

+QHTTPPOST: 0,200,177                   //If the HTTP response header information contains Content-
                                        Length, <content_length> information will be reported.
AT+QHTTPREAD=80                         //Read the HTTP response information and output it through the
                                        UART port. Maximum wait time for HTTP session to be closed is
                                        80 seconds.
CONNECT
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="httpHTTPs://api.efxnow.com/webservices2.3">Message='HelloQuectel' ASCII:72
101 108 108 111 81 117 101 99 116 101 108 </string>     //Output HTTP response information。
OK

+QHTTPREAD: 0                           //Successfully output HTTP response information.

//Send an HTTP PUT request and read the response information.
//The following examples illustrate how to send an HTTP PUT request and how to read the HTTP PUT
response information.
AT+QHTTPCFG="url","http://api.efxnow.com/DEMOWebServices2.8/Service.asmx/Echo?"
//Set the URL to access.
OK
AT+QHTTPPUT=20,80,80                    //Send an HTTP PUT request and obtain the PUT request body
                                        through UART. The maximum input time and response time of
                                        the PUT request body are both 80 seconds.
CONNECT
Message=HelloQuectel                    //Input the PUT request body with a length of 20 bytes. (This PUT
                                        request body is only an example, please enter the correct PUT
                                        request body according to the actual situation.)
OK

+QHTTPPUT: 0,200,177                    //If the HTTP response header information contains Content-
                                        Length, <content_length> information will be reported.
AT+QHTTPREAD=80                         //Read the HTTP response information and output it through the
                                        UART port. Maximum wait time for HTTP session to be closed is
                                        80 seconds.
CONNECT
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="httpHTTPs://api.efxnow.com/webservices2.3">Message='HelloQuectel' ASCII:72
101 108 108 111 81 117 101 99 116 101 108 </string>     //Output HTTP response information.
OK
```

```
+QHTTPREAD: 0                          // Successfully output HTTP response information.

//Access to HTTPS server.
//Send an HTTPS GET request and read the response information.
//The following examples illustrate how to send an HTTPS GET request, enable the output of HTTPS
response header information, and how to read the HTTPS GET response information.
//Example of sending an HTTPS GET request
AT+QHTTPCFG="rsp/header",1               //Enable output of HTTPS response header information.
OK
AT+QHTTPCFG="sslctxid",1                 //Set SSL context ID to 1.
OK
AT+QSSLCFG="sslversion",1,1              //Set the SSL version to 1, indicating TLSV1.0.
OK
AT+QSSLCFG="ciphersuite",1,0x0005       //Set the SSL cipher suite to 0x0005, which means RC4-SHA.
OK
AT+QSSLCFG="seclevel",1,0                //Set the SSL verification level to 0, indicating no
                                           authentication mode.
OK
AT+QHTTPCFG="url","https://www.alipay.com"     //Set the URL to access.
OK
AT+QHTTPGET=80                           //Send HTTPS GET request with a maximum response time
                                           of 80 seconds.
OK

+QHTTPGET: 0,200,21472               //If the HTTPS response header information contains Content-
                                       Length, <content_length> information will be reported.

//Example of reading HTTPS response information.
//Method 1: Read HTTPS response information and output it through UART.
AT+QHTTPREAD=80                         //Read the HTTPS response information and output it through
                                          UART. The maximum wait time for HTTP session to be
                                          closed is 80 seconds.
CONNECT
//HTTPS response header information and response body.
HTTP/1.1 200 OK<CR><LF>
Server: Tengine/2.1.0<CR><LF>
Date: Tue, 12 Sep 2017 05:54:34 GMT <CR><LF>
Content-Type: text/html; charset=utf-8<CR><LF>
Content-Length: 21451<CR><LF>
Connection: keep-alive <CR><LF>
……<CR><LF>                             //Response information is omitted here
<CR><LF>
<body>
OK
```

```
+QHTTPREAD: 0                    //HTTPS response header information and response body
                                   are read successfully.

//Send an HTTPS POST request and read the response information.
//The following examples illustrate how to send an HTTPS POST request and how to read the HTTPS
POST response information.
AT+QHTTPCFG="sslctxid",1           //Set SSL context ID to 1.
OK
AT+QSSLCFG="sslversion",1,1        //Set the SSL version to 1, indicating TLSV1.0.
OK
AT+QSSLCFG="ciphersuite",1,0x0005  //Set the SSL cipher suite to 0x0005, indicating RC4-SHA.
OK
AT+QSSLCFG="seclevel",1,2          //Set SSL verification level to 2.
OK
AT+QFOPEN="ca.pem"
+QFOPEN: 1

OK
AT+QFWRITE=1,1757
CONNECT

+QFWRITE: 1757,1757

OK
AT+QFCLOSE=1
OK
AT+QFOPEN="user.pem"
+QFOPEN: 1

OK
AT+QFWRITE=1,1684
CONNECT

+QFWRITE: 1684,1684

OK
AT+QFCLOSE=1
OK
AT+QFOPEN="user_key.pem"
+QFOPEN: 1

OK
AT+QFWRITE=1,2455
```

```
CONNECT

+QFWRITE: 2455,2455

OK
AT+QFCLOSE=1
OK
AT+QSSLCFG="cacert",1,"ca.pem"              //Configure the trusted CA certificate path for the
                                              SSL context.
OK
AT+QSSLCFG="clientcert",1,"user.pem"        //Configure the client certificate path for the SSL context.
OK
AT+QSSLCFG="clientkey",1,"user_key.pem"     //Configure the client key for the SSL context.
OK
AT+QHTTPCFG="url","https://220.180.239.212:8011/processorder.php"     //Set the URL to access.
OK
AT+QHTTPPOST=48,80,80                  //Send an HTTPS POST request and obtain the POST request
                                         body through UART. The maximum input time and response
                                         time of the POST request body are both 80 seconds.
CONNECT
Message=1111&Appleqty=2222&Orangeqty=3333&find=1    //Input the POST request body with a
                                                      length of 48 bytes. (This POST request
                                                      body is only an example, please enter the
                                                      correct POST request body according to
                                                      the actual situation.)
OK

+QHTTPPOST: 0,200,285                  //If the HTTPS response header information contains Content-
                                         Length, <content_length> information will be reported.
AT+QHTTPREAD=80                        // Read the HTTPS response information and output it through
                                         UART. The maximum wait time for HTTP session to be closed is
                                         80 seconds.
CONNECT                                //HTTPS response information is read successfully.
<html>
<head>
<title>Quectel's Auto Parts - Order Results</title>
</head>
<body>
<h1>Quectel's Auto Parts</h1>
<h2>Order Results</h2>

<p>Order processed at 02:49, 27th December</p><p>Your order is as follows: </p>1111
message<br />2222  apple<br />3333 orange<br /></body>
</html>
```

```
OK

+QHTTPREAD: 0                          //HTTPS response information is output successfully.

//Send an HTTPS PUT request and read the response information
//The following examples illustrate how to send an HTTPS PUT request and how to read the HTTPS PUT
response information.
AT+QHTTPCFG="sslctxid",1                      // Set SSL context ID to 1.
OK
AT+QSSLCFG="sslversion",1,1              // Set the SSL version to 1, indicating TLSV1.0.
OK
AT+QSSLCFG="ciphersuite",1,0x0005        // Set the SSL cipher suite to 0x0005, which means RC4-
                                         SHA.
OK
AT+QSSLCFG="seclevel",1,2                // Set SSL verification level to 2

AT+QFOPEN="ca.pem"
+QFOPEN: 1

OK
AT+QFWRITE=1,1757
CONNECT

+QFWRITE: 1757,1757

OK
AT+QFCLOSE=1
OK
AT+QFOPEN="user.pem"
+QFOPEN: 1

OK
AT+QFWRITE=1,1684
CONNECT

+QFWRITE: 1684,1684

OK
AT+QFCLOSE=1
OK
AT+QFOPEN="user_key.pem"
+QFOPEN: 1

OK
```

```
AT+QFWRITE=1,2455
CONNECT

+QFWRITE: 2455,2455

OK
AT+QFCLOSE=1
OK
AT+QSSLCFG="cacert",1,"ca.pem"          //Configure the trusted CA certificate path for the SSL context.
OK
AT+QSSLCFG="clientcert",1,"user.pem"   //Configure the client certificate path for the SSL context.
OK
AT+QSSLCFG="clientkey",1,"user_key.pem"   // Configure the client key for the SSL context.
OK
AT+QHTTPCFG="url","https://220.180.239.212:8011/processorder.php"       // Set the URL to access.
OK
AT+QHTTPPUT=48,80,80                  // Send an HTTPS PUT request and obtain the PUT request
                                     body through UART. The maximum input time and response
                                     time of the PUT request body are both 80 seconds.
CONNECT
Message=1111&Appleqty=2222&Orangeqty=3333&find=1   //Input the PUT request body with a length
                                                   of 48 bytes. (This PUT request
                                                   body is only an example, please enter the
                                                   correct PUT request body according to
                                                   the actual situation.)
OK

+QHTTPPUT: 0,200,285                  // If the HTTPS response header information contains Content-
                                        Length, <content_length> information will be reported.
AT+QHTTPREAD=80                         // Read the HTTPS response information and output it via
                                        UART. The maximum wait time for HTTP session closure is
                                        80 seconds.
CONNECT                                  //HTTPS response information is read successfully.
<html>
<head>
<title>Quectel's Auto Parts - Order Results</title>
</head>
<body>
<h1>Quectel's Auto Parts</h1>
<h2>Order Results</h2>

<p>Order processed at 02:49, 27th December</p><p>Your order is as follows: </p>1111
message<br />2222   apple<br />3333 orange<br /></body>
</html>
```

```
OK

+QHTTPREAD: 0                              //HTTPS response information is output successfully.
```

5 Result Code

Table 3: TCP/UDP/SSL Result Code

Result Code	English Description
0	Operation success
550	Unknown error
551	Operation blocked
552	Invalid parameters
553	Memory allocation failed
554	Create socket failed
555	Operation not support
556	Socket bind failed
557	Socket listen failed
558	Socket write failed
559	Socket read failed
560	Socket accept failed
561	Socket Identity has been used
562	Dns busy
563	Dns failed
564	Socket connect failed
565	Socket has been closed
566	Operation busy
567	operation timeout

568	Cancel sending
569	Operation not allowed
570	Port busy

Table 4: HTTP(S) Result Code

Result Code	English Description
0	Operation success
701	HTTP(S) unknown error
702	HTTP(S) timeout
703	HTTP(S) busy
704	HTTP(S) UART busy
705	HTTP(S) no request
706	HTTP(S) network error
707	HTTP(S) URL error
708	HTTP(S) empty URL
709	HTTP(S) IP address error
710	HTTP(S) DNS error
711	HTTP(S) socket create error
712	HTTP(S) socket connect error
713	HTTP(S) socket read error
714	HTTP(S) socket write error
715	HTTP(S) socket closed
716	HTTP(S) data encode error
717	HTTP(S) data decode error
718	HTTP(S) read timeout
719	HTTP(S) response failed

720	Input timeout
721	Wait data timeout
722	HTTP(S) response timeout
723	Memory not enough
724	Invalid parameter

6 Appendix Abbreviations

Table 5: Term Abbreviations

Abbreviation	Description
ACK	Acknowledgement
AP	Access Point
BLE	Bluetooth Low Energy
BSSID	Basic Service Set Identifier
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Server
GATT	Generic Attribute Profile
MAC	Medium Access Control
MQTT	Message Queuing Telemetry Transport
MTU	Maximum Transmission Unit
HTTP	Hyper Text Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ID	Mostly refers to Identifier in terms of software
IP	Internet Protocol
OTA	Over-the-Air Techology
PSK	Pre-Shared Key
QoS	Quality of Service
SNI	Server Name Indication
STA	Station

SSID	Service Set Identifier
SSL	Secure Sockets Layer
TA	Terminal Adapter
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
URC	Unsolicited Result Code
UUID	Universally Unique Identifier