FCM362K

AT Commands Manual

Short-Range Module Series

Version: 1.0.0

Date: 2024-11-05

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

We offer information as a service to you. The provided information is based on your requirements and we make every effort to ensure its quality. You agree that you are responsible for using independent analysis and evaluation in designing intended products, and we provide reference designs for illustrative purposes only. Before using any hardware, software or service guided by this document, please read this notice carefully. Even though we employ commercially reasonable efforts to provide the best possible experience, you hereby acknowledge and agree that this document and related services hereunder are provided to you on an “as available” basis. We may revise or restate this document from time to time at our sole discretion without any prior notice to you.

Use and Disclosure Restrictions

License Agreements

Documents and information provided by us shall be kept confidential, unless specific permission is granted. They shall not be accessed or used for any purpose except as expressly provided herein.

Copyright

Our and third-party products hereunder may contain copyrighted material. Such copyrighted material shall not be copied, reproduced, distributed, merged, published, translated, or modified without prior written consent. We and the third party have exclusive rights over copyrighted material. No license shall be granted or conveyed under any patents, copyrights, trademarks, or service mark rights. To avoid ambiguities, purchasing in any form cannot be deemed as granting a license other than the normal non-exclusive, royalty-free license to use the material. We reserve the right to take legal action for noncompliance with abovementioned requirements, unauthorized use, or other illegal or malicious use of the material.

Trademarks

Except as otherwise set forth herein, nothing in this document shall be construed as conferring any rights to use any trademark, trade name or name, abbreviation, or counterfeit product thereof owned by Quectel or any third party in advertising, publicity, or other aspects.

Third-Party Rights

This document may refer to hardware, software and/or documentation owned by one or more third parties ("third-party materials"). Use of such third-party materials shall be governed by all restrictions and obligations applicable thereto.

We make no warranty or representation, either express or implied, regarding the third-party materials, including but not limited to any implied or statutory, warranties of merchantability or fitness for a particular purpose, quiet enjoyment, system integration, information accuracy, and non-infringement of any third-party intellectual property rights with regard to the licensed technology or use thereof. Nothing herein constitutes a representation or warranty by us to either develop, enhance, modify, distribute, market, sell, offer for sale, or otherwise maintain production of any our products or any other hardware, software, device, tool, information, or product. We moreover disclaim any and all warranties arising from the course of dealing or usage of trade.

Privacy Policy

To implement module functionality, certain device data are uploaded to Quectel's or third-party's servers, including carriers, chipset suppliers or customer-designated servers. Quectel, strictly abiding by the relevant laws and regulations, shall retain, use, disclose or otherwise process relevant data for the purpose of performing the service only or as permitted by applicable laws. Before data interaction with third parties, please be informed of their privacy and data security policy.

Disclaimer

a) We acknowledge no liability for any injury or damage arising from the reliance upon the information.
b) We shall bear no liability resulting from any inaccuracies or omissions, or from the use of the information contained herein.
c) While we have made every effort to ensure that the functions and features under development are free from errors, it is possible that they could contain errors, inaccuracies, and omissions. Unless otherwise provided by valid agreement, we make no warranties of any kind, either implied or express, and exclude all liability for any loss or damage suffered in connection with the use of features and functions under development, to the maximum extent permitted by law, regardless of whether such loss or damage may have been foreseeable.
d) We are not responsible for the accessibility, safety, accuracy, availability, legality, or completeness of information, advertising, commercial offers, products, services, and materials on third-party websites and third-party resources.

About the Document

Revision History

Version	Date	Author	Description
-	2024-11-05	Tyler XIONG/ Vera SHI/ Orange LI	Creation of the document
1.0.0	2024-11-05	Tyler XIONG/ Vera SHI/ Orange LI	Preliminary

Contents

Table Index

Figure Index

1 Introduction

This document describes Wi-Fi, BLE, TCP/UDP, SSL, MQTT, HTTP(S), PING, DNS, NTP, and FILE-related AT commands supported by Quectel FCM362K modules. The maximum length of AT command can be set to 1024 bytes (including the trailing \r\n). If exceeded, an error will be returned. The module supports STA+BLE concurrent mode.

1.1. Definitions

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

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 commands can be operated in several modes, as following table:

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

2 AT Commands Description

2.1. Description of Wi-Fi-Related AT Commands

2.1.1. AT+QRST Reboot Module

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

Parameter

<enable>	Integer Type. Enable/Disable echo function. 0 Disable 1 Enable

2.1.4. AT+QGETIP Get IP Information

This command gets 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. Subnet mask of the module.
<DNS>	String type. DNS address of the module.

2.1.5. AT+QSETBAUD Set Baud Rate

This command sets and queries serial port baud rate.

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

Parameter

<baud_rate>	Integer type. Serial port baud rate. Range: 1200–2000000. Unit: bps.
<save>	Integer type. Whether to save the baud rate. 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. The address where firmware package is stored on the server.

2.1.7. AT+QWLMAC Get MAC address

This command gets MAC address of the module.

AT+QWLMAC Get MAC address	
Execution Command **AT+QWLMAC**	Response **+QWLMAC: <MAC>** **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	-

Parameter

<MAC>	String type in hexadecimal. MAC address of the module. Default value: "8800337769cc".

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 state. "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**
Write Command **AT+QSTADHCP=<enable>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	This 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+QSTASTATIC Set Static IP in STA Mode

This command sets static IP in STA mode

AT+QSTASTATIC Set Static IP in STA Mode	
Write Command **AT+QSTASTATIC=<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 in 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>],<band>**	Response **OK** Or **ERROR**
Maximum Response Time	3300 ms (Hotspot with password enabled)/300 ms (Hotspot without password enabled)
Characteristics	This command takes effect immediately. The configurations are not saved.

Parameter

<SSID>	String type. AP name. Range: 1–31. Unit: byte.
<key>	String type. AP password. Range: 8–63. Unit: byte. If this parameter is omitted, hotspot without password will be enabled.
<band>	Integer type. Wi-Fi band. 0 2.4 GHz 1 5 GHz

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 state.	
	"SOFTAP_DOWN"	Disabled
	"SOFTAP_UP"	Enabled

2.1.14. AT+QAPSTATIC Set Static IP in AP Mode

This command sets static IP in AP mode.

AT+QAPSTATIC Set Static IP in 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 in AP 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 to an AP Hotspot

This command connects to an AP hotspot and enters STA mode.

AT+QSTAAPINFO Connect to an AP Hotspot	
Write Command **AT+QSTAAPINFO=<SSID>[,<pwd>,<BSSID>,<DTIM>]**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	This 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.

<BSSID> String type in hexadecimal. BSSID of AP hotspot to be connected to. The length of this parameter should be 12 bytes. If this parameter is omitted, the module will not connect to an AP hotspot with a specified BSSID.

<DTIM> Integer type. The DTIM value in low-power keep-alive mode. The module will be periodically awakened according to this value in low-power keep-alive mode.
Range: 1–30. Default value: 10. Range: 100 ms.

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

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

AT+QSTAAPINFODEF Connect to a Hotspot and Save Hotspot Information	
Write Command **AT+QSTAAPINFODEF=<SSID>[,<pwd>,<BSSID>,<DTIM>]**	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.
<BSSID>	String type in hexadecimal. BSSID of AP hotspot to be connected to. The length of this parameter should be 12 bytes. If this parameter is omitted, the module will not connect to an AP hotspot with a specified BSSID.
<DTIM>	Integer type. The DTIM value in low-power keep-alive mode. The module will be periodically awakened according to this value in low-power keep-alive mode. Range: 1–30. Default value: 10. Range: 100 ms.

2.1.18. AT+QGETWIFISTATE Query Connected Hotspot Information

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

AT+QGETWIFISTATE Query Connected Hotspot Information	
Execution Command **AT+QGETWIFISTATE**	Response **+QGETWIFISTATE: ssid=<SSID>,bssid=<BSSID>,rssi=<RSSI>,pwd=<password>,freq=<frequency>,protocol=<protocol>,max_rate=<max_rate>** **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 Wi-Fi hotspot.
<RSSI>	Integer type. Wi-Fi signal strength.
<password>	String type. Wi-Fi password.
<frequency>	Integer type. Wi-Fi frequency.
<protocol>	Integer type. Wi-Fi protocol. 0 802.11b/g 1 802.11n/ac 2 802.11ax
<max_rate>	Integer type. Maximum transmission rate of Wi-Fi.

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 Wi-Fi hotspot.
<channel>	Integer type. In STA mode, Wi-Fi channel to be used to connect with hotspot.

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

This command 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	This command takes effect immediately. The configuration is not saved.

Parameter

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

2.1.21. 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	-

2.1.22. AT+QDEEPSLEEP Enter Deep Sleep Mode

This command sets the module to enter deep sleep mode.

AT+QDEEPSLEEP Enter Deep Sleep Mode	
Execution Command **AT+QDEEPSLEEP**	Response **OK** Or **ERROR**
Maximum Response Time	300 毫秒

Characteristics	-

NOTE

After the module enters deep sleep mode, it can be awakened through the rising edge interrupt of GPIO B4 pin.

2.1.23. AT+QCFG="urc/pin" Set URC Reporting Delay Time

This command sets the module to start a timer for delayed reporting of URC. After the RI signal wakes up the host, the URC is reported based on the delay time.

AT+QCFG Set URC Reporting Delay Time	
Test Command **AT+QCFG=?**	Response **+QCFG: "urc/pin",(**list of supported **<enable>**s**),(**list of supported **<pulse>**s**)** **OK** If there is any error: **ERROR**
Write Command **AT+QCFG="urc/pin"[,<enable>[,<pulse>]]**	Response If the optional parameters are omitted, query the current setting. **+QCFG: "urc/pin",<enable>** **OK** If the optional parameters are specified, set URC reporting delay time. **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<enable>	Integer type. Enable/Disable delayed reporting of URC. 0 Disable

	1 Enable
<pulse>	Integer type. URC reporting delay time. Range: 60–300. Default value: 120. Unit: ms.

2.1.24. AT+QIFC Enable/Disable Flow Control for Serial Communication

This command enables or disables the flow control for serial communication.

AT+QIFC Enable/Disable Flow Control for Serial Communication	
Test Command **AT+QIFC=?**	Response **+QIFC: (**list of supported **<enable>**s**)** **OK** If there is any error: **ERROR**
Read Command **AT+QIFC?**	Response **+QIFC: <enable>** **OK** If there is any error: **ERROR**
Write Command **AT+QIFC=<enable>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configuration is not saved.

Parameter

<enable>	Inter type. Enables or disable flow control for serial communication. 0 Disable 1 Enable

2.1.25. AT+QSCLK Enter Low Power Mode

This command controls the module to enter low power mode. Before executing this command to enter low power mode, you must first execute **AT+QSTAAPINFO** or **AT+QSTAAPINFODEF** to connect to Wi-Fi. After entering low power mode, the module can be awakened by pulling up the GPIO B3 pin.

AT+QSCLK Enter Low Power Mode	
Test Command **AT+QSCLK=?**	Response **+QSCLK: (**list of supported **<mode>**s**)** **OK**
Write Command **AT+QSCLK=<mode>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configuration is not saved.

Parameter

<mode>	Integer type. Low power mode.	
	1	Light sleep mode. In this mode, peripherals and RAM remain powered, while the system clock periodically powers down. If Wi-Fi is connected, the module will automatically enter Wi-Fi keep-alive mode, and it will be awaken periodically based on DTIM value. The DTIM value can be set in **AT+QSTAAPINFO** or **AT+QSTAAPINFODEF**. If it is not set, the default DTIM value is 10 (unit: 100 ms).
	2	Deep sleep mode. In this mode, the system clock and peripherals periodically power down, while RAM remains powered. If Wi-Fi is connected, the module will automatically enter Wi-Fi keep-alive mode, and it will be awaken periodically based on DTIM value. The DTIM value can be set in **AT+QSTAAPINFO** or **AT+QSTAAPINFODEF**. If it is not set, the default DTIM value is 10 (unit: 100 ms).
	3	Hibernate sleep mode. In this mode, the system clock, peripherals and RAM directly power down, and waking up the module is equivalent to rebooting it.

2.2. Description of BLE-Related AT Commands

2.2.1. AT+QBLEINIT Initialize BLE Service

This command initializes BLE service

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

Parameter

<role>	Integer type. Initializes BLE service. 1 Module is operating as a central device for initializing BLE. 2 Module is operating as a peripheral device for initializing BLE. The advertisement is turned on automatically by default after the BLE disconnection. If you do not need this feature, you can execute **AT+QBLEINIT=2,0** during BLE initialization to turn it off. 3 Module is operating as a peripheral device for configuring Wi-Fi via BLE.
<auto_adv>	Integer type. Whether the advertisement is turned on automatically after the BLE disconnection 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 "90395E8A70D1".

2.2.3. AT+QBLENAME Set BLE Name

This command sets the 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	The command takes effect immediately. The configuration is saved automatically.

Parameter

<BLE_name>	String type. BLE name. Max length: 25 bytes. Default name: "FCM362K".

2.2.4. AT+QBLEADVPARAM Set BLE Advertising Parameters

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

AT+QBLEADVPARAM Set BLE Advertising Parameters	
Read Command **AT+QBLEADVPARAM?**	Response **+QBLEADVPARAM: <adv_int_min>,<adv_int_max>** **OK** If there is any error: **ERROR**
Write Command **AT+QBLEADVPARAM=<adv_int_min>,<adv_int_max>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	This 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: 32–16384 (corresponding time range: 20 ms – 10240 ms). Default value: 160 (100 ms). Unit: timeslot (1 timeslot = 0.625 ms).
<adv_int_max>	Integer type. Maximum advertising interval for non-directional advertising and low-duty cycle directional advertising. Range: 32–16384 (corresponding time range: 20 ms – 10240 ms). Default value: 160 (100 ms). Unit: timeslot (1 timeslot = 0.625 ms).

NOTE

To configure advertising parameters, **AT+QBLEADVPARAM** should be executed before initiating 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.

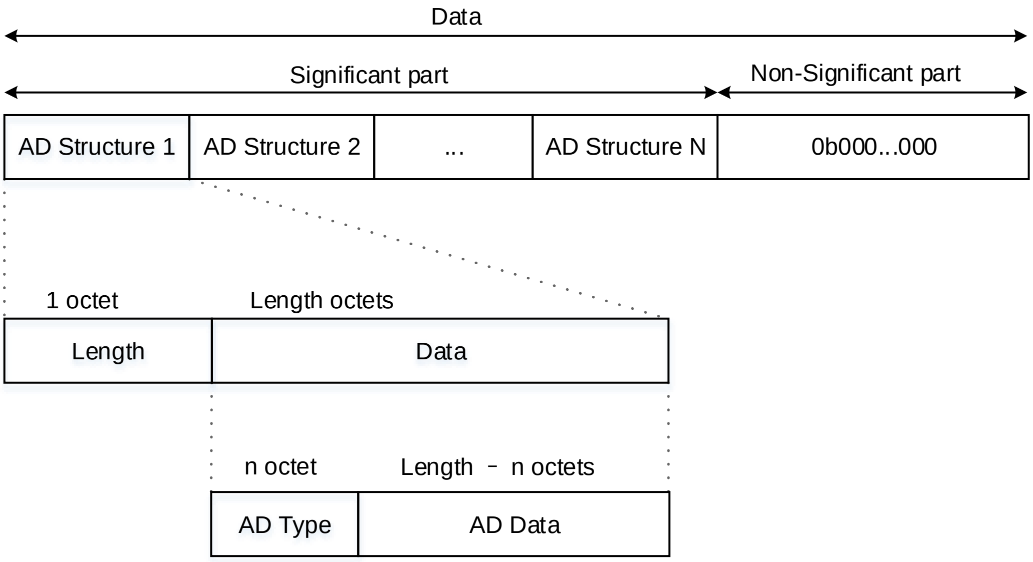


Figure 1: BLE Advertising Data Format

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 1-byte length of the Length field itself. The 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

1. The advertisement data header 020106 is automatically added at the bottom layer, and there is no need to set it.
2. Types and values of AD Type values will be automatically added in 020106 firmware. For details of AD types, See *Core Specification 5.2* (https://www.bluetooth.com/specifications/specs/core-specification/).

2.2.6. AT+QBLESRDATA Set BLE Scan Response Data

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

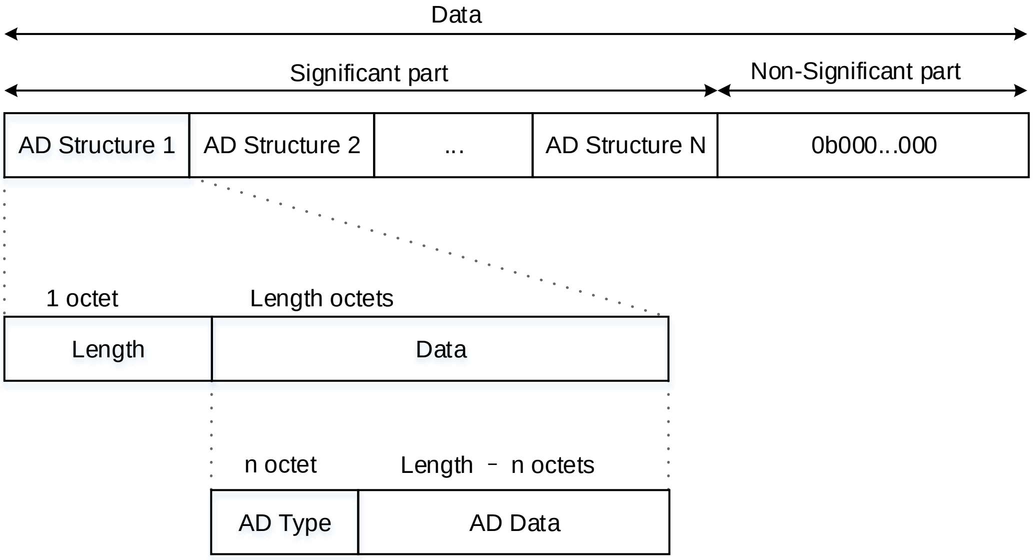


Figure 2: BLE Scan Response Data Format

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

Parameter

<sr_data>	String type. Scan response data. 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 1-byte length of the Length field itself. The maximum length is 0x1e, i.e., the maximum length of a data field is 30 bytes.
	AD Type	Data response type, such as TX Power Level (0x0A), Local Name (0x09), Le Role (0x1C) and Service UUIDs (0x16). After the peer scans the data response, the meaning of the data response can be determined from the AD Type.
	AD Data	Data response in big-endian byte order.

2.2.7. AT+QBLEGATTSSRV Establish a BLE Service

This command establishes a BLE service when the 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	This command takes effect immediately. The configuration is not saved

Parameter

<srv_UUID>	String type. BLE service UUID. Length: 4 bytes or 32 bytes.

NOTE

1. Only two BLE services can be established at most through the command, and the same service cannot be created again.
2. You must first initialize the BLE through **AT+QBLEINIT** to set the module as a peripheral device, and then establish a BLE service through **AT+QBLEGATTSSRV**. Otherwise, the Bluetooth advertising, scan and connection functions cannot be used normally.

2.2.8. AT+QBLEGATTSCHAR Set BLE Characteristic UUID

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

AT+QBLEGATTSCHAR Set BLE Characteristic UUID	
Write Command **AT+QBLEGATTSCHAR=<char_UUID>[,<att_cfg>]**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	This command takes effect immediately. The configurations are not saved.

Parameter

<char_UUID>	String type. Characteristic UUID. Length: 4 bytes or 32 bytes.
<att_cfg>	Integer type. Property configuration of characteristic UUID. Range: 0–FF. bit7–bit0 represent different properties, which are shown below. You can configure the corresponding bit, 0 indicates that the property is disabled and 1 indicates that the property is enabled. For example, FF indicates that properties corresponding to bit7–bit0 are enabled. bit7–bit3 Reserve bit2 Notify bit1 Write no response bit0 Write

NOTE

1. Only two BLE characteristic UUIDs can be set at most through this command, and the same UUID cannot be set again. When **<att_cfg>** is omitted, the property of each characteristic UUID is Notify and Write no response by default.
2. When the module is operating as a peripheral device, you must establish a BLE service through **AT+QBLEGATTSSRV**, and then set BLE characteristic UUID through this command. Otherwise, the Bluetooth advertising, scan and connection functions cannot be used normally.

2.2.9. AT+QBLEGATTSSRVDONE Complete Adding BLE Service

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

AT+QBLEGATTSSRVDONE Complete Adding BLE Service	
Execution Command **AT+QBLEGATTSSRVDONE**	Response **OK** Or

	ERROR
Maximum Response Time	300 ms
Characteristics	-

NOTE

When the module is operating as a peripheral device, you must set BLE characteristic UUID through **AT+QBLEGATTSCHAR**, and then complete the BLE service adding through this command. Otherwise, the Bluetooth advertising, scan and connection functions cannot be used normally.

2.2.10. AT+QBLEADVSTART Start BLE Advertising

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

AT+QBLEADVSTART Start BLE Advertising	
Read Command **AT+QBLEADVSTART?**	Response **+QBLEADVSTART: <adv_state>** **OK** If there is any error: **ERROR**
Execution Command **AT+QBLEADVSTART**	Response **OK** Or **ERROR**
Maximum Response Time	1000 ms
Characteristics	-

Parameter

<adv_state>	Integer type. Advertising state. 0 The advertising is stopped 1 The advertising is started

2.2.11. 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	-

2.2.12. AT+QBLEGATTSNTFY Send GATT Data by Notification

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

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

Parameter

<conn_idx>	Integer type. The peer device index assigned by the module automatically when the BLE connection is established. Range: 0–1. Default value: 0.
<UUID>	String type. Characteristic UUID. Length: 4 bytes or 32 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 converts the data to 0x123456 and sends it. When this parameter is omitted, the module sends **<data>** directly without conversion. For example, if **<data>** is 123456, the module sends the data as 123456.
<data>	String type. GATT data. The maximum data length is **<MTU_value>**-3, and the length of **<MTU_value>** can be obtained through the URC **+QBLEMTU**.

2.2.13. AT+QBLESCAN Enable/Disable BLE Scan

This command enables or disables BLE scan when the module is operating as a central device.

AT+QBLESCAN Enable/Disable BLE Scan	
Read Command **AT+QBLESCAN?**	Response **+QBLESCAN: <scan_state>** **OK** If there is any error: **ERROR**
Write Command **AT+QBLESCAN=<scan>[,<timeout>]**	Response If **<scan>** is 0: **OK** If **<scan>** is 1: **OK** **+QBLESCAN: <name>,<address_type>,<BLE_addr>** If **<scan>** is 2 or 3: **OK** **+QBLESCAN: <name>,<address_type>,<BLE_addr>,<RSSI>,<adv_data>** If there is any error: **ERROR**
Maximum Response Time	1000 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<scan_state>	Integer type. The current scan state. 0 The scan is disabled 1 The scan is enabled
<scan>	Integer type. Enable/disable BLE scan. 0 Stop 1 Start BLE scan but do not output **<RSSI>** and **<adv_data>**. 2 Start BLE scan and output **<RSSI>** and **<adv_data>**. **<timeout>** should be specified in this mode.

	3 Start BLE scan and output **<RSSI>** and **<adv_data>**. Only connectable advertising is scanned in this mode.
<timeout>	Integer type. Scan duration. Range: 100–10000. Unit: ms. It must be set when **<scan>** is 2.
<name>	String type. Scanned BLE device name.
<address_type>	Integer type. BLE device address type. 0 Public address 1 Random address
<BLE_addr>	String type. BLE device address. A 48-bit address is represented in a string of hexadecimal numbers, such as "58D391010203".
<RSSI>	Integer type. The signal strength value.
<adv_data>	String type. Scanned BLE data and response data.

2.2.14. AT+QBLESCANPARAM Query/Set BLE Scan Parameters

The command queries or sets BLE scan parameters when the module is operating as a central device.

AT+QBLESCANPARAM	Query/Set BLE Scan Parameters
Read Command **AT+QBLESCANPARAM?**	Response **+QBLESCANPARAM: <scan_interval>,<scan_window>** **OK** If there is any error: **ERROR**
Write Command **AT+QBLESCANPARAM=<scan_interval>,<scan_window>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<scan_interval>	Integer type. BLE scan interval. Range: 18–4096 (Time range: 11.25 ms to 2.56 s). Unit: timeslot (1 timeslot = 0.625 ms). Default value: 100.
<scan_window>	Integer type. BLE scan duration. **<scan_window>** shall be less than or equal to **<scan_interval>**. Range: 17–4096 (Time range: 10.625 ms–2.56 s). Unit: timeslot (1 timeslot = 0.625 ms). Default value: 30.

NOTE

The configured scan parameters do not take effect immediately during the scan process. You must disable the scan and then enable the scan again. After that, the parameters can take effect.

2.2.15. AT+QBLECONN Connect to a Peripheral Device

The command connects to a peripheral device when the module is operating as a central device.

AT+QBLECONN Connect to a Peripheral Device	
Write Command **AT+QBLECONN=<addr_type>,<peer_addr>[,<timeout>]**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<addr_type>	Integer type. Peripheral device address type. 0 Public address 1 Random address
<peer_addr>	String type. Peripheral device address.
<timeout>	Integer type. The timeout from the connection initiation to connection completion. Range: 1000–60000. Unit: ms. Default value: 15000.

2.2.16. AT+QBLECONNPARAM Configure Connection Parameters

The command configures connection parameters when the module is operating as a peripheral device or a central device.

AT+QBLECONNPARAM Configure Connection Parameters	
Read Command **AT+QBLECONNPARAM?**	Response **+QBLECONNPARAM: <conn_idx>,<con_interval>,<timeout>,<latency>** **OK** If there is any error: **ERROR**

Write Command **AT+QBLECONNPARAM=<conn_idx>,<con_interval>,<timeout>,<latency>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<conn_idx>	Integer type. The peer device index assigned by the module automatically when the BLE connection is established. Range: 0–1. Default value: 0.
<con_interval>	Integer type. Connection interval. Range: 6–3200 (Time range: 7.5 ms to 4 s). Unit: 1.25 ms.
<timeout>	Integer type. BLE Link supervision timeout. Range: 10–3200 (Time range: 100 ms to 32 s). Unit: 10 ms.
<latency>	Integer type. Number of connection events delayed by the peripheral device. Range: 0–499 (0x0000 to 0x01F3).

NOTE

The relationship among **<con_interval>**, **<timeout>** and **<latency>** should be as follows: **<con_interval>** × **<latency>** is less than the value of **<timeout>**.

2.2.17. AT+QBLEGATTCRD Read GATT Data

This command reads GATT data when the module is operating as a central device.

AT+QBLEGATTCRD Read GATT Data	
Write Command **AT+QBLEGATTCRD=<conn_idx>,<UUID>**	Response **OK** **<data>** If there is any error: **ERROR**
Maximum Response Time	3000 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<conn_idx>	Integer type. The peer device index assigned by the module automatically when the BLE connection is established. Range: 0–1. Default value: 0.
<UUID>	String type. Characteristic UUID. Length: 2 bytes or 16 bytes.
<data>	String type without double quotation marks. Read GATT data.

2.2.18. AT+QBLEGATTCWRCMD Send GATT Data by Writing Commands

The command sends GATT data by writing commands when the module is operating as a central device.

AT+QBLEGATTCWRCMD Send GATT Data by Writing Commands	
Write Command **AT+QBLEGATTCWRCMD=<conn_idx>,<UUID>[,<hex_length>],<data>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<conn_idx>	Integer type. The peer device index assigned by the module automatically when the BLE connection is established. Range: 0–1. Default value: 0.
<UUID>	String type. Characteristic UUID. Length: 4 bytes or 32 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 converts the data to 0x123456 and sends it. When this parameter is omitted, the module sends **<data>** directly without conversion. For example, if **<data>** is 123456, the module sends the data as 123456.
<data>	String type. GATT data.

2.2.19. AT+QBLEGATTCWRREQ Send GATT Data by Writing Requests

The command sends GATT data by writing requests when the module is operating as a central device.

AT+QBLEGATTCWRREQ Send GATT Data by Writing Requests	
Write Command **AT+QBLEGATTCWRREQ=<conn_idx>,<UUID>[,<hex_length>],<data>**	Response **OK** Or

	ERROR
Maximum Response Time	1000 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<conn_idx>	Integer type. The peer device index assigned by the module automatically when the BLE connection is established. Range: 0–1. Default value: 0.
<UUID>	String type. Characteristic UUID. Length: 4 bytes or 32 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 converts the data to 0x123456 and sends it. When this parameter is omitted, the module sends **<data>** directly without conversion. For example, if **<data>** is 123456, the module sends the data as 123456.
<data>	String type. GATT data.

2.2.20. AT+QBLEGATTCNTFCFG Enable/Disable Notification

The command enables or disables notification when the module is operating as a central device.

AT+QBLEGATTCNTFCFG Enable/Disable Notification	
Write Command **AT+QBLEGATTCNTFCFG=<conn_idx>,<UUID>,<enable>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<conn_idx>	Integer type. The peer device index assigned by the module automatically when the BLE connection is established. Range: 0–1. Default value: 0.
<UUID>	String type. Characteristic UUID. Length: 4 bytes or 32 bytes.
<enable>	Integer type. Enable/Disable notification. 0 Disable 1 Enable

2.2.21. AT+QBLEDISCONN Disconnect BLE Connection

The command disconnects a BLE connection.

AT+QBLEDISCONN Disconnect BLE Connection	
Write Command **AT+QBLEDISCONN[=<conn_idx>]**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configuration is not saved.

Parameter

<conn_idx>	Integer type. The peer device index assigned by the module automatically when the BLE connection is established. Range: 0–1. Default value: 0.

2.2.22. AT+QBLESTAT Query BLE Device State

The command queries the state of the BLE device when the module is operating as a central device or a peripheral device.

AT+QBLESTAT Query BLE Device State	
Execution Command **AT+QBLESTAT**	Response **+QBLESTAT: <BLE_state>** **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	-

Parameter

<BLE_state>	String type. Current state of BLE device.	
	"NOINIT"	Uninitialized
	"INIT"	Initialized
	"ADVERTISING"	Advertising
	"CONNECTED"	Connected

"DISCONNECTED"	Disconnected

2.2.23. AT+QBLETRANMODE Specify Bluetooth Characteristic UUID and Control Module to Enter Transparent Transmission Mode

This command specifies Bluetooth characteristic UUID and controls the module to enter transparent transmission mode when the module is operating as a peripheral device or central device.

AT+QBLETRANMODE Specify Bluetooth Characteristic UUID and Control Module to Enter Transparent Transmission Mode	
Write Command **AT+QBLETRANMODE=<UUID>[,<state>]**	Response **OK** Or **ERROR**
Maximum Response Time	500 ms
Characteristics	This command takes effect immediately. The configurations are not saved.

Parameter

<UUID>	String type. Characteristic UUID. Length: 4 bytes or 32 bytes.
<state>	Integer type. 0 When the module is operating as a peripheral device, after entering transparent transmission mode for the first time, it will automatically exit transparent transmission mode after disconnecting from a BLE connection. Then the module will not enter transparent transmission mode by default after subsequent re-establishing the BLE connection. 1 When the module is operating as a peripheral device, after entering transparent transmission mode for the first time, it will automatically exit transparent transmission mode after disconnecting from a BLE connection. Then the module will enter transparent transmission mode by default after subsequent re-establishing the BLE connection.

NOTE

1. When the module is operating as a central device, after establishing the BLE connection and having completed the discovering of peer device service (URC: **+QBLEINFO: "DOWN"**), execute the command to set the module to transparent transmission mode. After disconnecting the BLE connection, the module exits transparent transmission mode automatically. You can only send data in transparent transmission mode by writing commands.
2. When the module is operating as a peripheral device, after enabling advertising or establishing the BLE connection, execute the command to set the module to transparent transmission mode. If

automatic advertising is enabled, the module does not exit transparent transmission mode after the BLE connection is terminated. If the automatic advertising is disabled, the module exits transparent transmission mode automatically after the BLE connection is terminated. You can only send data in transparent transmission mode by Notify.

3. If you execute **+++** to exit the transparent transmission mode, the module will not enter the transparent transmission mode again after establishing a BLE connection by default.

2.2.24. +++ Exit Transparent Transmission Mode

This command controls the module to exit transparent transmission mode.

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

2.2.25. AT+QBLECFGMTU Configure Maximum Transmission Unit for BLE

This command configures the maximum transmission unit for BLE when the module is operating as a central device. The maximum BLE transmission unit can only be set once in a BLE connection.

AT+QBLECFGMTU Configure Maximum Transmission Unit for BLE	
Read Command **AT+QBLECFGMTU?**	Response **+QBLECFGMTU: <MTU_value>** **OK** If there is any error: **ERROR**
Write Command **AT+QBLECFGMTU=<MTU_value>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	This command takes effect immediately. The configuration is not saved.

Parameter

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

2.2.26. AT+QBLEDEINIT De-initialize BLE Service

This command de-initializes the BLE service.

AT+QBLEDEINIT De-initialize BLE Service	
Execution Command **AT+QBLEDEINIT**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	-

2.3. Description of TCP/UDP Related AT Commands

2.3.1. AT+QICFG Configure Optional Parameters for TCP/UDP Socket Service

This command configures optional parameters for TCP/UDP socket service.

AT+QICFG Configure Optional Parameters for TCP/UDP Socket Service	
Test Command **AT+QICFG=?**	Response **+QICFG: "transpktsize",(**list of supported**<transpktsize>**s**)** **+QICFG: "transwaittm",(**list of **<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",(**list of supported **<timeout>**s**)** **+QICFG: "close/mode",(**list of supported **<close_mode>**s**)** **+QICFG: "tcp/kalive",(**list of supported **<kalive_onoff>**s**),(**list of supported **<kalive_idle>**s**),(**list of supported **<kalive_interval>**s**),(**list of supported **<kalive_cnt>**s**)** **+QICFG: "tcp/tw_cycle",(**list of supported **<tw_enable>**s**)** **OK**
Write Command Set the size of packet to be sent in	Response If the optional parameter is omitted, query the current setting.

transparent transmission mode **AT+QICFG="transpktsize"[,<transpktsize>]**	**+QICFG: "transpktsize",<transpktsize>** **OK** If the optional parameter is specified, set the size of packet to be sent in transparent transmission mode. **OK** If there is any error: **ERROR**
Write Command Set the waiting time before automatically sending data in transparent transmission mode **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, set the waiting time before automatically sending data in transparent transmission mode. **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 parameter is 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 the 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 the optional parameter is specified, set whether to passively close the TCP connection when the server is shut down. **OK**

	If there is any error: **ERROR**
Write Command Enable or disable automatically accepting TCP connections from clients. **AT+QICFG="tcp/accept"[,<state>]**	Response If the optional parameter is omitted, query the setting. **+QICFG: "tcp/accept",<state>** **OK** If the 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 setting. **+QICFG: "qisend/timeout",<timeout>** **OK** If the optional parameter is specified, set the timeout after output **>**. **OK** If there is any error: **ERROR**
Write Command Set asynchronous TCP connection closure **AT+QICFG="close/mode"[,<close_mode>]**	Response If the optional parameter is omitted, query the current setting. **+QICFG: "close/mode",<close_mode>** **OK** If the optional parameter is specified, set asynchronous disconnection of TCP connections. **OK** If there is any error: **ERROR**
Write Command Set to enable or disable TCP keepalive function. **AT+QICFG="tcp/kalive"[,<kalive_onoff>[,<kalive_idle>,<kalive_interval,<kalive_cnt>]]**	Response If the optional parameter is omitted, query the current setting. **+QICFG: "tcp/kalive",<kalive_onoff>[,<kalive_idle>,<kalive_interval>,<kalive_cnt>]** **OK**

	If the optional parameter is specified, enable or disable the TCP keepalive function. **OK** If there is any error: **ERROR**
Write Command Set 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 the optional parameter is specified, enable or disable a quick release of TCP connection. **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	This command takes effect immediately The configurations are not saved.

Parameter

<transpktsize>	Integer type. The length of packet to be sent in transparent transmission mode. Range: 1–1460. Default value: 1024. Unit: byte.
<transwaittm>	Integer type. Waiting time before automatically sending data if the data is less than **<transpktsize>** in transparent transmission mode. Range: 0–20. Default value: 2. Unit: 100 ms.
<send_data_format>	Integer Type. Format of the data to be sent. 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 Mode 1 Hexadecimal Mode
<recv_data_format>	Integer Type. Format of the data to be received. 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 Mode 1 Hexadecimal Mode
<closed>	Integer type. Enable or disable TCP connection, and the connection will be automatically disconnected after server shutdown.

	0 Disable 1 Enable
<state>	Integer type. Enable or disable automatic acceptance of TCP connections from clients. 0 Disable 1 Enable
<timeout>	Integer type. Timeout for sending data. Range: 0–120. Unit: second.
<close_mode>	Integer type. Enable or disable asynchronous disconnection of TCP. connections. 0 Disable. 1 Enable
<kalive_onoff>	Integer type. Enable or disable TCP keepalive function. 0 Disable 1 Enable
<kalive_idle>	Integer type. Keep-alive timer. Range: 60–7200. Unit: second.
<kalive_interval>	Integer type. Keep-alive interval. Range: 10–20. Unit: second.
<kalive_cnt>	Integer type. Keep-alive probe count. Range: 1–8.
<tw_enable>	Integer type. Enable or disable a quick release of TCP connection. 0 Enable 1 Disable

2.3.2. AT+QIOPEN Open Socket Service

This 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 as **+QIURC: "incoming",<connectID>,<serverID>,<remoteIP>,<remote_port>**. The range of **<connectID>** is 0–5. The type of this new incoming connection is "TCP INCOMING" and the **<access_mode>** of "TCP INCOMING" is the same as 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 as **+QIURC: "recv",<connectID>,<currentrecvlength>,<remoteIP>,<remote_port><CR><LF><data>**.
- Receive data in buffer access mode: the module reports the URC as **+QIURC: "recv",<connectID>**, and then data can be retrieved 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 after executing the Write Command, **AT+QICLOSE** should be used to close the socket.

AT+QIOPEN Open Socket Service	
Test Command **AT+QIOPEN=?**	Response **+QIOPEN: (**list of supported **<connectID>**s**),(**list of supported **<service_type>**s**),<IP_address>/<domain_name>,(**list of supported **<remote_port>**s**),(**list of supported **<local_port>**s**),(**list 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 got 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	This command takes effect immediately. The configurations are not saved.

Parameter

<connectID>	Integer type. Socket service index. Range: 0–5.	
<service_type>	String type. Socket service type.	
	"TCP"	Start a TCP connection as a client
	"UDP"	Start a UDP connection as a client
	"TCP LISTENER"	Start a TCP server to listen for TCP connection
	"UDP SERVICE"	Start a UDP service
<IP_address>	String type.	
	If **<service_type>** is "TCP" or "UDP", **<IP_address>** indicates the IP address of	

remote server, such as "192.0.2.2".
If **<service_type>** is "TCP LISTENER" or "UDP SERVICE", enter "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 or is specified to 0, then the local port will be automatically assigned, otherwise this parameter must be specified.

<access_mode> Integer type. Data access mode of Socket service.
0 Buffer access mode
1 Direct push mode
2 Transparent access mode

<err> Result code. See ***Chapter 5*** for details.

2.3.3. AT+QSERVER Open Socket Listening Service

This command opens the socket listening service. It can be used before/after AP/STA starts. If it is configured before AP/STA starts, the listening service will be automatically enabled on the port according to the configuration when AP/STA starts, and the results will be reported through URC: **+QSERVER: <connectID>,<err>**.

In addition, the socket listening service opened through **AT+QSERVER** will automatically be disabled when the AP/STA stops, and will automatically enable the listening service when the AP/STA starts again. When you execute **AT+QICLOSE** to close the socket service, this listening service will be completely released.

AT+QSERVER Open Socket Listening Service	
Test Command **AT+QSERVER=?**	Response **+QSERVER: (**list of supported **<connectID>**s**),(**list of supported **<listen_type>**s**),(**list of supported **<netif_type>**s**),(**list of supported **<listen_port>**s**),(**list of supported **<access_mode>**s**)** **OK**
Read Command **AT+QSERVER?**	Response **[+QSERVER: <connectID>,<listen_type>,<netif_type>,<listen_port>]** **[…]** **OK**

Write Command **AT+QSERVER=<connectID>,<listen_type>,<netif_type>,<listen_port>[,<access_mode>**	Response **+QSERVER: <connectID>,<err>** **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	This command takes effect immediately. The configurations are not saved.

Parameter

<connectID>	Integer type. Socket service index. Range: 0–5.	
<listen_type>	String type.	
	"TCP"	Start a TCP server to listen to TCP connection
<netif_type>	String type.	
	"STA"	Module works as STA
	"AP"	Module works as AP
<listen_port>	Integer type. Module's listening port. Range: 1–65535.	
<access_mode>	Integer type. Data access mode of Socket listening service.	
	0	Buffer access mode
	1	Direct push mode
	2	Transparent transmission mode
<err>	Result code. See ***Chapter 5*** for details.	

2.3.4. AT+QICLOSE Close a Socket Service

This command closes a specified socket service. Determined by 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: (**list of supported **<connectID>**s**),(**list of supported **<timeout>**s**)** **OK**
Write Command **AT+QICLOSE=<connectID>[,<timeout>]**	Response **OK** Or **ERROR**

Maximum Response Time	Default: 10 seconds / Determined by **<timeout>**
Characteristics	This command takes effect immediately. The configurations are not saved.

Parameter

<connectID>	Integer type. The socket service index. Range: 0–5.
<timeout>	Integer type. The timeout value for the response to be outputted. If the FIN ACK of the other peer is not received within **<timeout>**, the module will be forced to close the socket. Range: 0–120. Default value: 10. Unit: second.

2.3.5. 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 service index. Range: 0–5.	
<service_type>	String type. The socket service type.	
	"TCP"	Start a TCP connection as a client

	"UDP" Start a UDP connection as a client "TCP LISTENER" Start a TCP server to listen for TCP connection "TCP INCOMING" Start a TCP connection accepted by a TCP server "UDP SERVICE" Start a UDP service
<IP_address>	String type. IP address. If **<service_type>**="TCP" or "UDP", **<IP_address>** is the IP address of remote server. If **<service_type>**="TCP LISTENER" or "UDP SERVICE", **<IP_address>** is the local IP address. If **<service_type>**="TCP INCOMING", **<IP_address>** is the IP address of remote client.
<remote_port>	Integer type. Remote port number. If **<service_type>**="TCP" or "UDP", **<remote_port>** is the port of remote server. If **<service_type>**="TCP LISTENER" or "UDP SERVICE", the port is invalid. If **<service_type>**="TCP INCOMING", **<remote_port>** is the port of remote client.
<local_port>	Integer type. Local port number. If **<local_port>** is not specified, then the local port is assigned automatically.
<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/incoming connection has been established 3 "Listening": server is listening 4 "Closing": connection is closing
<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

2.3.6. 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 it will return **SEND FAIL** or **ERROR**.

- **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 whether the data has reached the server by **AT+QISEND=<connectID>,0**.

AT+QISEND Send Data	
Test Command **AT+QISEND=?**	Response **+QISEND: (**list of supported **<connectID>**s**),(**list of supported **<send_length>**s**)** **OK**
Write Command Send variable-length data when **<service_type>** is "TCP", "UDP" or "TCP INCOMING" **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 has been established and the data is sent successfully: **SEND OK** If the connection has been established but the data is not sent successfully: **SEND FAIL** If there is any error: **ERROR**
Write Command Send fixed-length data when **<service_type>** is "TCP", "UDP" or "TCP INCOMING" **AT+QISEND=<connectID>,<send_length>**	Response **>** After the response **>**, input the data until the data length equals to **<send_length>** If the connection has been established and the sending buffer space is not full: **SEND OK** If the connection has been established but the sending buffer is full: **SEND FAIL** If there is any error: **ERROR**
Write Command If **<service_type>**="UDP SERVICE" **AT+QISEND=<connectID>,<send_length>,<remoteIP>,<remote_port>**	Response This command is used to send 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 until the data length equals to **<send_length>**:

	If the connection has been established and the sending buffer is not full: **SEND OK** If the connection has been established but the sending buffer 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>,<unackedbytes>** **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	This command takes effect immediately. The configurations are not saved.

Parameter

<connectID>	Integer type. Socket service index. Range: 0–5.
<send_length>	Integer type. The length of data to be sent. Range: 0–1460; Unit: byte.
<remoteIP>	String type. Remote IP address (must be dotted decimal 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 acknowledged data. Unit: byte
<unackedbytes>	Integer type. The total length of unacknowledged data. Unit: byte.

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

This command reads the received TCP/IP data. In buffer access mode, after receiving data, 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: (**list of supported **<connectID>**s**),(**list of supported **<read_length>**s**)** **OK**
Write Command When**<service_type>**is "TCP"/"UDP"/"TCP INCOMING" **AT+QIRD=<connectID>[,<read_length>]**	Response If the specified connection has received the data: **+QIRD: <read_actual_length><CR><LF><data>** **OK** If there is no data: **+QIRD: 0** **OK** If the connection does not exist: **ERROR**
Write Command When **<service_type>** is "UDP SERVICE" **AT+QIRD=<connectID>**	Response If data exists: **+QIRD: <read_actual_length>,<remoteIP>,<remote_port><CR><LF><data>** **OK** If there is no data: **+QIRD: 0** **OK** If the connection does not exist: **ERROR**
Write Command When **<read_length>** is 0, query the retrieved data length **AT+QIRD=<connectID>,0**	Response If the specified connection exists: **+QIRD: <total_receive_length>,<have_read_length>,<unread_length>** **OK**

	If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	-

Parameter

<connectID>	Integer type. The socket service index. Range: 0–5.
<read_length>	Integer type. The maximum length of data to be read. Range: 0–1500. Unit: byte.
<read_actual_length>	Integer type. The length of data that has been actually read. 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. The data that has been read. Unit: byte.
<total_receive_length>	Integer type. The total length of the read data. Unit: byte.
<have_read_length>	Integer type. The length of data that has been read. Unit: byte.
<unread_length>	Integer type. The length of data that has not been read. Unit: byte.

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

This command accepts/rejects the 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: (**list of supported **<listener_socketID>**s**),(**list of supported **<accept>**s**),(**list 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	This command takes effect immediately. The configurations are not saved.

Parameter

<listener_socketID>	Integer type. TCP server socket ID. Range: 0–5.
<accept>	Integer type. Reject or accept the remote incoming connection request from TCP/UDP socket service. 0 Reject 1 Accept
<incoming_socketID>	Integer type. The Socket ID for incoming connection. It is only valid when **<accept>**=1. Range: 0–5.
<remote_addr>	String type. Source address of the incoming connection.
<remote_port>	Integer type. Source port of the incoming connection.

2.3.9. AT+QISWTMD Switch Data Access Mode

The command switches the data access mode which includes buffer access mode, direct push mode and transparent transmission mode. When establishing a socket service, the data access mode can be specified via **<access_mode>** of **AT+QIOPEN**. After a socket service has been opened, the data access mode can be switched via **AT+QISWTMD**.

AT+QISWTMD Switch Data Access Mode	
Test Command **AT+QISWTMD=?**	Response **+QISWTMD: (**list of supported **<connectID>**s**),(**list 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	300 ms
Characteristics	The command takes effect immediately; The configurations are not saved.

Parameter

<connectID>	Integer type. The socket service index. Range: 0–5.
<access_mode>	Integer type. The data access modes of the connection. 0 Buffer access mode 1 Direct push mode 2 Transparent transmission mode

2.3.10. AT+QIGETERROR Query the Last Error Code

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

AT+QIGETERROR Query the Last Error Code	
Test Command **AT+QIGETERROR=?**	Response **OK**
Execution Command **AT+QIGETERROR**	Response **+QIGETERROR: <err>,<errcode_description>** **OK**
Maximum Response Time	300 ms
Characteristics	-

Parameter

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

2.3.11. 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
Characteristics	-

NOTE

1. If the socket connection has not been established before, **ATO** returns **NO CARRIER**.
2. After exiting the transparent transmission mode with **+++**, if the socket connection is active, you can enter the transparent transmission mode again with **ATO**.

2.4. Description of SSL Related AT Commands

2.4.1. AT+QSSLCFG Configure SSL Context Parameters

This command configures SSL configuration such as SSL version, cipher suite, trusted CA certificate path, authentication mode, client certificate, client key path, etc. of the SSL context. These parameters will be used during the handshake.

<SSL_ctxID> is the index of the SSL context. The module supports up to 2 SSL contexts. Multiple SSL connections can be established based on one SSL context. Settings for SSL version and cipher suite are stored in the SSL context and will be applied to new SSL connections associated with that SSL context.

AT+QSSLCFG Configure SSL Context Parameters	
Test Command **AT+QSSLCFG=?**	Response **+QSSLCFG:"sslversion",(**list of supported **<SSL_ctxID>**s**),(**list of supported **<SSL_version>**s**)** **+QSSLCFG: "ciphersuite",(**list of supported **<SSL_ctxID>**s**),(**list of supported **<cipher_suites>**s**)** **+QSSLCFG: "cacert",(**list of supported **<SSL_ctxID>**s**),<cacertpath>** **+QSSLCFG: "clientcert",(**list of supported **<SSL_ctxID>**s**),<client_cert_path>** **+QSSLCFG: "clientkey",(**list of supported **<SSL_ctxID>),<client_key_path>,<key_pwd>** **+QSSLCFG: "seclevel",(**list of supported **<SSL_ctxID>**s**),(**list of supported **<seclevel>**s**)** **+QSSLCFG: "ignorelocaltime",(**list of supported**<SSL_ctxID>**s**),(**list of supported**<ignore_ltime>)** **+QSSLCFG: "negotiatetime",(**list of supported**<SSL_ctxID>**s**),(**list of supported **<negotiate_time>)** **+QSSLCFG: "sni",(**list of supported**<SSL_ctxID>**s**),(**list of supported **<SNI>**s**)** **+QSSLCFG: "session_cache",(**list of supported **<SSL_ctxID>),(**list of supported **<session_cache_enable>**s**)** **OK**

Write command Configure the SSL version of the specified 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, configure the SSL version of the specified SSL context. **OK** If there is any error: **ERROR**
Write Command Configure the cipher suite for the specified SSL context **AT+QSSLCFG="ciphersuite",<SSL_ctxID>[,<cipher_suites>]**	Response If the optional parameter is omitted, query the current setting. **+QSSLCFG: "ciphersuite",<SSL_ctxID>,<cipher_suites>** **OK** If the optional parameter is specified, configure the cipher suite for the specified SSL context. **OK** If there is any error: **ERROR**
Write Command Configure the trusted CA certificate path for the specified SSL context **AT+QSSLCFG="cacert",<SSL_ctxID>[,<cacertpath>]**	Response If the optional parameter is omitted, query the current setting. **+QSSLCFG: "cacert",<SSL_ctxID>,<cacertpath>** **OK** If the optional parameter is specified, configure the trusted CA certificate path for the specified SSL context. **OK** If there is any error: **ERROR**
Write Command Configure the client certificate path for the specified SSL context **AT+QSSLCFG="clientcert",<SSL_ctxID>[,<client_cert_path>]**	Response If the optional parameter is omitted, query the current setting. **+QSSLCFG: "clientcert",<SSL_ctxID>,<client_cert_path>** **OK** If the optional parameter is specified, configure the client certificate path for the specified SSL context.

	OK If there is any error: **ERROR**
Write Command Configure the client key for the specified SSL context **AT+QSSLCFG="clientkey",<SSL_ctxID>[,<client_key_path>[,<key_pwd>]]**	Response If the optional parameters are omitted, query the current setting. **+QSSLCFG: "clientkey",<SSL_ctxID>,<client_key_path>[,<key_pwd>]** **OK** If the optional parameters are specified, configure the client key for the specified SSL context. **OK** If there is any error: **ERROR**
Write Command Configure 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, configure the authentication mode of the specified SSL context. **OK** If there is any error: **ERROR**
Write Command Configure whether the specified SSL context ignores certificate validity verification **AT+QSSLCFG="ignorelocaltime",<SSL_ctxID>[,<ignore_ltime>]**	Response If the optional parameter is omitted, query the current setting. **+QSSLCFG: "ignorelocaltime",<SSL_ctxID>,<ignore_ltime>** **OK** If the optional parameter is specified, configure whether the specified SSL context ignores certificate validity verification. **OK** If there is any error: **ERROR**

Write Command Configure the timeout period of the specified SSL context during the SSL negotiation phase **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, configure the maximum timeout period of the specified SSL context during the SSL negotiation phase. **OK** If there is any error: **ERROR**
Write Command Enable or disable the server name indication for the 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 the server name indication of the 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 the 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 identifier. Range: 0–2.	
<SSL_version>	Integer type. SSL version.	
	0	SSL3.0
	1	TLS1.0
	2	TLS1.1
	3	TLS1.2
	4	all
<cipher_suites>	Hexadecimal value. SSL encryption 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	Supports all cipher suites.
<ignore_ltime>	Integer type. Whether to ignore certificate validity verification.	
	0	Do not ignore
	1	Ignore
<cacertpath>	String type. Trusted CA certificate path.	
<client_cert_path>	String type. Client certificate path.	

<client_key_path> String type. Client key path.

<key_pwd> String type. User public key password. Length: 0–255. Unit: byte.

<seclevel> Integer type. Authentication mode

0	No authentication mode
1	one way- Perform server authentication
2	two way- Server and client authentication

<negotiate_time> Integer type. Timeout period of the SSL negotiation phase. 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 or disable the SSL session buffer.

0	Disable
1	Enable

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

This command establishes an SSL connection, that is, opens an SSL socket to connect to the remote server. During the negotiation between the module and network, **AT+QSSLCFG** will be used for parameter configuration in the handshake process. After a successful handshake with the network, the module can send or receive data through this SSL connection, and can also establish multiple SSL connections based on one SSL context.

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 to Remote Server	
Test Command **AT+QSSLOPEN=?**	Response **+QSSLOPEN: (**list of supported **<SSL_ctxID>**s**),(**list of supported **<clientID>**s**),<server_address>,(**list of supported **<server_port>**s**),(**list of supported **<access_mode>**s**)** **OK**
Write Command **AT+QSSLOPEN=<SSL_ctxID>,<clientID>,<serveraddr>,<server_port>[,<access_mode>]**	Response If **<access_mode>**=2 and the SSL connection is established: **CONNECT** If**<access_mode>**=0/1: **OK** **+QSSLOPEN: <clientID>,<err>**

	When **<err>**=0, it indicates the SSL socket is opened successfully; otherwise, SSL socket failed to be opened. 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>**.
Characteristics	This command takes effect immediately; The configurations are not saved.

Parameter

<SSL_ctxID>	Integer type. SSL context ID. Range: 0–2.
<clientID>	Integer type. Socket index. Range: 0–5.
<serveraddr>	String type. Remote server address.
<server_port>	Integer type. Remote server listening port. Range: 1–65535.
<access_mode>	Integer type. Access mode of SSL connection. 0 Buffer access mode 1 Direct push mode 2 Transparent transmission mode
<err>	Error codes. See ***Chapter 5*** for details.
<negotiate_time>	Integer type. Timeout period of the SSL negotiation phase. Range: 10–300. Default Value: 300. Unit: Second.

2.4.3. AT+QSSLSEND Send Data Through SSL Connection

This command sends data through the SSL Socket connection.

AT+QSSLSEND Send Data Through SSL Connection	
Test Command **AT+QSSLSEND=?**	Response **+QSSLSEND: (**list of supported **<clientID>**s**),(**list of supported **<send_length>**s**)** **OK**
Write Command Send the data with variable length **AT+QSSLSEND=<clientID>**	Response **>** After the response **>**, enter the data to be sent. Tap **Ctrl + Z** to send, and tap **Esc** to cancel the operation. If the SSL socket has been opened and the data is sent successfully: **SEND OK**

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

Parameter

<clientID>	Integer type. Socket index. Range: 0–5.
<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 variable length is 1460 bytes.

2.4.4. AT+QSSLRECV Read Data Received 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 Data Received via SSL Connection	
Test Command **AT+QSSLRECV=?**	Response **+QSSLRECV: (**list of supported **<clientID>**s**),(**list of supported **<read_length>**s**)** **OK**
Write Command **AT+QSSLRECV=<clientID>,<read_length>**	Response If the specified socket connection receives data: **+QSSLRECV: <have_readlen><CR><LF><data>** **OK** If the buffer is empty: **+QSSLRECV: 0** **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	This command takes effect immediately; The configurations are not saved.

Parameter

<clientID>	Integer type. Socket index. Range: 0–5.
<read_length>	Integer type. The maximum length of data to be read. Range:1–1500. Unit: byte.
<have_readlen>	Integer type. Read the actual length of data through **AT+QSSLRECV**. Unit: byte.
<data>	The actual read data.

2.4.5. AT+QSSLCLOSE Close SSL Connection

This command closes the SSL connection. If all SSL connections based on the same SSL context have been closed, the module will release the SSL context.

AT+QSSLCLOSE Close SSL Connet	
Test Command **AT+QSSLCLOSE=?**	Response **+QSSLCLOSE: (**list of supported **<clientID>**s**),(**list of supported **<close_timeout>**s**)** **OK**

Write Command **AT+QSSLCLOSE=<clientID>[,<close_timeout>]**	Response **OK** Or **ERROR**
Maximum Response Time	Determined by the time configured in **<close_timeout>**
Characteristics	This command takes effect immediately. The configurations are not saved.

Parameter

<clientID>	Integer type. Socket index. Range: 0–5.
<close_timeout>	Integer type. Timeout of closing an SSL connection. Range: 0–120. Default value: 10. Unit: Second. 0 indicates that the command will be executed immediately.

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**
Read Command **AT+QSSLSTATE?**	Response Returns the status of all existing SSL connections. **[+QSSLSTATE: <clientID>,"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>,<clientID>**	Response Returns the status of all existing SSL connections for a specific context. **+QSSLSTATE: <clientID>,"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>,<clie**	Response **+QSSLSTATE: <clientID>,"SSLClient",<IP_address>,<remote_port>,<local_port>,<socket_state>,<serverID>,<access_mode>,<devname>,<SSL_ctxID>**

ntID>	**OK**
Execution Command **AT+QSSLSTATE**	Response Returns the status of all existing SSL connections. **[+QSSLSTATE: <clientID>,"SSLClient",<IP_address>,<remote_port>,<local_port>,<socket_state>,<serverID>,<access_mode>,<devname>,<SSL_ctxID>]** **[…]** **OK**
Maximum Response Time	300 ms
Characteristics	-

Parameter

<clientID> Integer type. Socket index. Range: 0–5.

<IP_address> String type. Remote server address.

<remote_port> Integer type. Remote server port. Range: 0–65535.

<local_port> Integer type. Local port. Range: 0–65535

<socket_state> Integer type. SSL connection status.

0 "Initial" Connection not established.
1 "Opening" Client is connecting.
2 "Connected" Client connection established.
4 "Closing" Connection is closing.

<serverID> Integer type. It is valid only when **<service_type>** is "TCP INCOMING". **<serverID>** represents which server accepted this TCP connection. This parameter value is the same as the **<connectID>** value of the server "TCP LISTENER".

<access_mode> Integer type. Represents the data access mode of an SSL connection

0 Buffer access mode
1 Direct push mode
2 Transparent transmission mode

<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:0–2.

<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 Configure MQTT Parameters

This command configures MQTT parameters when the MQTT connection is not created.

AT+QMTCFG Configure MQTT Parameters	
Test Command **AT+QMTCFG=?**	Response **+QMTCFG:"version",(**list of supported **<client_idx>**s**),(**list of supported **<vsn>**s**)** **+QMTCFG: "ssl",(**list of supported **<client_idx>**s**),(**list of supported **<SSL_enable>**s**),(**list of supported **<SSL_ctx_idx>**s**)** **+QMTCFG:"keepalive",(**list of supported **<client_idx>**s**),(**list of supported **<keep_alive_time>**s**)** **+QMTCFG:"session",(**list of supported **<client_idx>**s**),(**list of supported **<clean_session>**s**)** **+QMTCFG: "will",(**list of supported **<client_idx>s),(**list of supported **<will_flag>**s**),(**list of supported **<will_qos>**s**),(**list of supported **<will_retain>**s**),<will_topic>,<will_message>** **+QMTCFG: "recv/mode",(**list of supported **<client_idx>**s**),(**list of supported **<msg_recv_mode>**s**),(**list of supported **<msg_len_enable>**s**)** **+QMTCFG: "dataformat",(**list of supported **<client_idx>**s**),(**list of supported **<send_mode>**s**),(**list of supported **<recv_mode>**s**)** **+QMTCFG:"timeout",(**list of supported **<client_idx>**s**),(**list of supported **<pkt_timeout>**s**),(**list of supported **<retry_times>**s**),(**list of supported **<timeout_notice>**s**)** **+QMTCFG: "aliauth",(**list of supported **<client_idx>**s**),(**list of supported **<product_ket>**s**),(**list of supported **<device_name>**s**),(**list of supported **<device_secret>**s**)** **OK**
Write Command Configure 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, configure the MQTT protocol version. **OK**

	If there is any error: **ERROR**
Write Command Configure 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_ctx_idx>]** OK If optional parameters are specified, configure the MQTT SSL mode and SSL context index. **OK** If there is any error: **ERROR**
Write Command Configure 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 the optional parameter is specified, configure the keep-alive time. **OK** If there is any error: **ERROR**
Write Command Configure 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 the optional parameter is specified, configure the session type. **OK** If there is any error: **ERROR**
Write Command Configure Will Information **AT+QMTCFG="will",<client_idx>[,<will_flag>[,<will_qos>,<will_retain>,<will_topic>,<will_message>]]**	Response If the optional parameters are omitted, query the current setting. **+QMTCFG: "will",<will_flag>[,<will_qos>,<will_retain>,<will_topic>,<will_message>]** **OK**

	If optional parameters are specified, configure Will information. **OK** If there is any error: **ERROR**
Write Command Configure the reception mode of server data **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 the optional parameter is pecified, configure the reception mode of server data. **OK** If there is any error: **ERROR**
Write Command Configure 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, configure the MQTT data format. **OK** If there is any error: **ERROR**
Write Command Configure 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, configure the message transmission timeout. **OK** If there is any error: ERROR
Write Command Configure Alibaba Cloud device	Response If the optional parameters are omitted, query the current setting.

information **AT+QMTCFG="aliauth",<client_idx>[,<product_key>,<device_name>,<device_secret>]**	**+QMTCFG: "aliauth",<product_key>,<device_name>,<device_secret>** **OK** If the optional parameters are specified and the MQTT connection is not established, configure Alibaba Cloud device information. **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 index. Range :0–5.
<vsn>	Integer type. MQTT Protocol version. 3 MQTT protocol v3.1 4 MQTT protocol v3.1.1
<SSL_enable>	Integer type. MQTT SSL connection type. 0 Enable normal SSL connection 1 Enable SSL TCP security connection
<SSL_ctx_idx>	Integer type. SSL context index. Range: 0–2.
<keep_alive_time>	Integer type. Keep-alive time. Range: 0–3600. Default value: 120. Unit: second. 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, then the server will disconnect from the client. If the value of this parameter is 0, it means that the client connection is not disconnected.
<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: second.
<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 packets 0 Not report 1 Report
<will_fg>	Integer type. Whether to configure Will Flag. 0 Not configure 1 Configure
<will_qos>	Integer type. QoS level corresponding to sending **<will_message>**. 0 At most once 1 At least once 2 Exactly once
<will_retain>	Integer type. Whether to retain PUBLISH message. 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_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: 0–256. Unit: byte.
<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.
<msg_len_enable>	Integer type. Whether the length of MQTT message received from server will be contained in URC. 0 Not contained 1 Contained
<send_mode>	Integer type. MQTT message sending format. 0 String type 1 Hexadecimal type
<recv_mode>	Integer type. MQTT message receiving format. 0 String type 1 Hexadecimal type
<product_key>	String type. Product key released by Alibaba Cloud.
<device_name>	String type. Device name released by Alibaba Cloud.
<device_secret>	String type. Device secret released by Alibaba Cloud.

NOTE

1. If **<will_flag>**=1, then **<will_qos>**, **<will_retain>**, **<will_topic>** and **<will_message>** must be specified; if **<will_flag>** is not 1, the above parameters can 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 to SSL mode, **<SSL_ctx_idx>** must be specified. In addition, during the MQTT SSL handshake process, you need to configure the SSL version, cipher suite, security level, CA certificate, client certificate and client key through **AT+QSSLCFG**.

2.5.2. AT+QMTOPEN Open MQTT Client

This command opens the MQTT client network.

AT+QMTOPEN Open MQTT Client	
Test Command **AT+QMTOPEN=?**	Response **+QMTOPEN: (**list of supported **<client_idx>**s**),<host_name>,(**list 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 s, determined by the network.
Characteristics	This command takes effect immediately. The configurations are not saved.

Parameter

<client_idx>	Integer type. MQTT client index. 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 Failed to activate PDP 4 Failed to parse domain name 5 Network disconnection causing error

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: (**list 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
Characteristics	This command takes effect immediately. The configuration is not saved.

Parameter

<client_idx>	Integer type. MQTT client index. Range: 0–5.
<result>	Integer type. Command execution result. -1 Failed execution 0 successful execution

2.5.4. AT+QMTCONN Connect Client to MQTT Server

The client requests to connect to the MQTT server through this command. When a client establishes a TCP/IP socket connection with a server, it must create a protocol layer session using the TCP connection.

AT+QMTCONN Connect Client to MQTT Server	
Test Command **AT+QMTCONN=?**	Response **+QMTCONN: (**list of supported **<client_idx>**s**),<clientID>,<username>,<password>** **OK**
Read Command **AT+QMTCONN?**	Response **[+QMTCONN: <client_idx>,<state>]** **OK** If there is any error: **ERROR**
Write Command **AT+QMTCONN=<client_idx>,<clientID>[,<username>,<password>]**	Response **OK** **+QMTCONN: <client_idx>,<result>[,<ret_code>]** If there is any error: **ERROR**
Maximum Response Time	**<pkt_timeout>** value (default 5 seconds), determined by the network.
Characteristics	This command takes effect immediately. The configurations are not saved.

Parameter

<client_idx>	Integer type. MQTT client index. Range: 0–5.
<state>	Integer type. MQTT connection status. 1 MQTT is initialization 2 MQTT is connecting 3 MQTT is connected successfully 4 MQTT is disconnecting
<clientID>	String type. MQTT client identifier. The maximum length is 127 bytes.
<username>	String type. MQTT client username, which can be used for authorization. The maximum length is 256 bytes.
<password>	String type. The password corresponding to the MQTT client username, which can be used for authorization. The maximum length is 256 bytes.
<result>	Integer type. Command execution result

	0 The data packet is sent successfully and the ACK from the server is received. 1 Retransmit the data packet 2 Failed to send the data packet
<ret_code>	Integer type. Connection status. 0 Accept connection 1 Connection refused: Unaccepted protocol version 2 Connection refused: identifier is refused 3 Connection refused: Server is unavailable 4 Connection refused: Incorrect username or password 5 Connection refused: Unauthorized
<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>]**.

NOTE

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

2.5.5. AT+QMTDISC Disconnect Client from MQTT Server

The client requests to disconnect from the MQTT server through this command. The client sends a DISCONNECT message to the server to indicate that it is about to disconnect the MQTT connection with the server.

AT+QMTDISC Disconnect Client from MQTT Server	
Test Command **AT+QMTDISC=?**	Response **+QMTDISC: (**list 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
Characteristics	This command takes effect immediately. The configuration is not saved.

Parameter

<client_idx>	Integer type. MQTT client index. 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: (**list of supported **<client_idx>**s**),(**list of supported **<msgID>**s**),<topic>,(**list of supported **<qos>**s**)** **OK**
Write Command **AT+QMTSUB=<client_idx>,<msgID>,<topic1>,<qos1>[,<topic2>,<qos2>[,…]]**	Response **OK** **+QMTSUB: <client_idx>[,<msgID>],<result>[,<value>]** If there is any error: **ERROR**
Maximum Response Time	**<pkt_timeout>** × **<retry_times>** (default 15 seconds), determined by the network
Characteristics	-

Parameter

<client_idx>	Integer type. MQTT client index. Range: 0–5.
<msgID>	Integer type. SUBSCRIBE message identifier. Range: 0–65535.
<topic>	String type. Topics that the client subscribes to or unsubscribes from.
<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 ACK message was received. 1 SUBSCRIBE message was sent successfully but SUBACK message was not received within the specified response time. Retransmission was performed.

	2 Failed to send SUBSCRIBE message.
<value>	Integer type. If **<result>**=0, **<value>** indicates QoS level of SUBSCRIBE message. If **<result>**=1, **<value>** indicates the number of SUBSCRIBE message retransmissions. If **<result>**=2, **<value>** indicates this parameter is meaningless and the field is empty.
<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>]**.
<retry_times>	Integer type. Number of retries after failed packet transmission. Range: 0–10. Default value: 3.

NOTE

<msgID> is only displayed in the 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 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: (**list of supported **<client_idx>),(**list 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 15 seconds), determined by the network
Characteristics	-

Parameter

<client_idx>	Integer type. MQTT client index. Range: 0–5.
<msgID>	Integer type. UNSUBSCRIBE message identifier. Range: 0–65535.
<topic>	String type. The topic that the client wants to subscribe to or unsubscribe from
<result>	Integer type. Command execution result 0 UNSUBSCRIBE message was sent successfully and ACK message was received. 1 UNSUBSCRIBE message was sent successfully but UNSUBACK message was not received within the specified response time. Retransmission was performed. 2 Failed to send UNSUBSCRIBE message.
<value>	Integer type. If **<result>**=0, **<value>** indicates QoS level of UNSUBSCRIBE message. If **<result>**=1, **<value>** indicates the number of SUBSCRIBE message retransmissions. If **<result>**=2, **<value>** indicates this parameter is meaningless and the field is empty.
<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>]**.
<retry_times>	Integer type. Number of retries after failed packet transmission. Range: 0–10. Default value: 3.

2.5.8. AT+QMTPUB Publish Messages

The client publishes 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 send these messages to the client as PUBLISH messages.

AT+QMTPUB Publish Messages	
Test Command **AT+QMTPUB=?**	Response **+QMTPUB: (**list of supported **<client_idx>**s**),(**list of supported **<msgID>**s**),(**list of supported **<qos>**s**),(**list of supported **<retain>**s**),<topic>,(**list of supported **<length>**s**)** **OK**
Write Command **AT+QMTPUB=<client_idx>,<msgID>,<qos>,<retain>,<topic>[,<length>]**	Response **>** After response **>**, 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 tap **Ctrl + Z** to send the data.

	OK **+QMTPUB: <client_idx>,<msgID>,<result>[,<value>]** If there is any error: **ERROR**
Maximum Response Time	**<pkt_timeout>** × **<retry_times>** (default 15 seconds), determined by the network.
Characteristics	-

Parameter

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

<msgID> Integer type. PUBLISH message identifier. Range :0–65535. Only when **<qos>**=0, this parameter is 0.

<qos> Integer type. QoS level of PUBLISH message.
0 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.
0 Do not save
1 Save

<topic> String type. The topic associated with the message 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. When data is published and **<qos>**=0, no ACK is required.
1 PUBLISH message was sent successfully but ACK message was not received within the specified response time. Retransmission was performed.
2 Failed to send PUBLISH message.

<value> Integer type
If **<result>**=1, **<value>** indicates the number of PUBLISH message retransmissions.
If **<result>**=0 or 2, **<value>** indicates this parameter is meaningless and the field is empty.

<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>]**.

<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 a new data packet. 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 the data, that is, payload information. The maximum length of sent data each time is 2048 bytes. If it exceeds, please send it in segments.
3. The subscriber can send PUBLISH messages to the server, and the server can also send PUBLISH messages to subscribers. When the server sends a PUBLISH 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 about URC description, see ***Chapter 3.5.2***.

2.5.9. AT+QMTRECV Read Messages from Buffer

This command reads messages from 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**
Read Command **AT+QMTRECV?**	Response **+QMTRECV: <client_idx>,<store_status_0>,<store_status_1>,<store_status_2>,<store_status_3>,<store_status_4>** **OK** If there is no MQTT connection: **OK**
Write Command **AT+QMTRECV=<client_idx>[,<receiveID>]**	Response **[+QMTRECV: <client_idx>,<msgID>,<topic>[,<payload_length>],<payload>** **[…]]** **OK** If there is no MQTT connection: **ERROR**
Maximum Response Time	300 ms
Characteristics	-

Parameter

<client_idx>	Integer type. MQTT Client index. Range: 0–5.
<store_status_x>	Integer type. Whether there is a message in the buffer. Up to 5 pieces of messages can be stored in the buffer, so up to 5 URCs can be reported at the same time. 0 No message in the buffer 1 One message in the buffer
<receiveID>	Integer type. Sequence number of each received data. Range: 0–4.
<msgID>	Integer type. Packet message identifier. Range: 0–65535. When **<qos>**=0, this parameter is 0.
<topic>	String type. The topic associated with the message to be published.
<payload_length>	Integer type. The length of the payload message. Range: 0–10240. Unit: byte
<payload>	String type. Payload message related to the topic name.

2.6. Description of HTTP Related AT Commands

2.6.1. AT+QHTTPCFG Configure Parameters for HTTP(S) Server

This command configures parameters for the HTTP(S) server, including configuring PDP context ID, customizing HTTP(S) request header information, outputting HTTP(S) response header information, and configuring SSL context ID.

AT+QHTTPCFG Configure Parameters for HTTP(S) Server	
Test Command **AT+QHTTPCFG=?**	Response **+QHTTPCFG: "url",<urlstring>** **+QHTTPCFG: "header",<header_value>** **+QHTTPCFG: "auth",<username_password>** **+QHTTPCFG: "sslctxid",(**list of supported **<sslctxID>**s**)** **+QHTTPCFG: "rsp/header",(**list of supported **<response_header>**s**)** **+QHTTPCFG: "rspout/auto",(**list of supported **<auto_outrsp>**s**),(**list of supported **<pkt_size>**s**),(**list of supported **<pkg_freq>**s**)** **+QHTTPCFG: "closed/ind",(**list of supported **<closed_ind>**s**)** **+QHTTPCFG: "form/option",<name>,<file_name>,<content_type>** **+QHTTPCFG: "reset"** **OK**
Write Command	Response

AT+QHTTPCFG="url"[,<urlstring>]	If the optional parameter is omitted, query the current setting. **+QHTTPCFG: "url",<urlstring>** **OK** If the optional parameter is specified, configure the HTTP(S) URL. **OK** If there is any error: **+CME ERROR: <err>**
Write Command **AT+QHTTPCFG="header"[,<header_value>]**	Responses If the optional parameter is omitted, query the current setting. **+QHTTPCFG: "header",<header_value>** **[…]** **OK** If the optional parameter is 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 the optional parameter is omitted, query the current setting. **+QHTTPCFG: "auth",<username_password>** **OK** If the optional parameter is specified, configure the username and password. **OK** If there is any error: **+CME ERROR: <err>**
Write Command **AT+QHTTPCFG="sslctxid"[,<sslctxID>]**	Response If the optional parameter is omitted, query the current setting. **+QHTTPCFG: "sslctxid",<sslctxID>** **OK** If the optional parameter is 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"[,<response_header>]**	Response If the optional parameter is omitted, query the current setting. **+QHTTPCFG: "rsp/header",<response_header>** **OK** If the optional parameter is specified, disable or enable 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>[,<pkt_size>[,<pkg_freq>]]]**	Response If optional parameters are omitted, query the current setting. **+QHTTPCFG: "rspout/auto",<auto_outrsp>,<pkt_size>,<pkg_freq>** **OK** If optional parameters are specified, disable or enable the automatic output of HTTP(S) response header information: **OK** If there is any error: **+CME ERROR: <err>**
Write Command **AT+QHTTPCFG="closed/ind"[,<closed_ind>]**	Response If the optional parameter is omitted, query the current setting. **+QHTTPCFG: "closed/ind",<closed_ind>** **OK** If the optional parameter is specified, disable or enable reporting of HTTP(S) session closure 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 setting. **+QHTTPCFG: "form/option",<name>,<file_name>,<content_type>** **[…]** **OK** If 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	-
Characteristics	The command takes effect immediately; The configurations are not saved.

Parameter

<response_header>	Integer type. Disable or enable the output of HTTP(S) response headers. 0 Disable 1 Enable
<sslctxID>	Integer type. SSL context ID for HTTP(S). Range: 0–2. Default value: 1. SSL parameters can be configured in **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 the automatic output of HTTP(S) response header information is enabled, **AT+QHTTPREAD** and **AT+QHTTPREADFILE** will fail to execute. 0 Disable 1 Enable
<pkt_size>	Integer type. The packet size of the automatically output HTTP(S) response header information. Range: 512–1500. Unit: byte.
<pkg_freq>	Integer type. The interval time at which each packet is automatically output. Range: 10–2000. Unit: ms.
<closed_ind>	Integer type. Disable or enable reporting of HTTP(S) session closure URC **+QHTTPURC: "closed"**. 0 Disable 1 Enable

<urlstring>	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. Name value in form/option.
<file_name>	String type. Filename value in form/option.
<content_type>	String type. Content-type value in form/option.
<err>	Error codes. See ***Chapter 5*** for details.

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

This command sends the GET request to HTTP(S) server. After executing **AT+QHTTPGET** Write Command and returning **OK**, it is necessary 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 the 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: (**list of supported **<rsptime>**s**)** **OK**
Write Command **AT+QHTTPGET[=<rsptime>]**	Response If the parameter format is correct and no other error occurs: **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	Determined by **<rsptime>**
Characteristics	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 value: 60. Unit: second.
<httprspcode>	HTTP(S) server response code. See ***Chapter 5*** for details.

<content_length>	Integer type. HTTP(S)Response length. Unit: byte.
<err>	Error codes. See ***Chapter 5*** 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 returning **OK**, it is necessary 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: (**list of supported **<data_length>**s**),(**list of supported **<input_time>**s**),(**list 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** After TA switches to transparent transmission mode (that is, data mode), the HTTP(S) POST request body can be inputted. When the total size of input data reaches **<data_length>**, TA will switch back to command mode and report the following result: **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 URC: **+QHTTPPOST: <err>** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**

Maximum Response Time	Determined by network and **<rsptime>**
Characteristics	The command takes effect immediately; The configurations are not saved.

Parameter

<data_length>	Integer type. The length of the POST request body. Range: 1–1024000. Unit: byte.
<input_time>	Integer type. The maximum input time for POST request body. Range: 1–65535. Default value: 60. Unit: second.
<rsptime>	Integer type. Timeout value for URC **+QHTTPPOST: <err>[,<httprspcode>[,<content_length>]]** to be outputted. Range: 1–65535. Default value: 60. Unit: second.
<httprspcode>	HTTP(S) server response code. See ***Chapter 5*** for details.
<content_length>	Integer type. HTTP(S) response length. Unit: byte.
<err>	Error codes. See ***Chapter 5*** 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 outputted.

After executing **AT+QHTTPPUT** and returning **OK**, it is necessary 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: (**list of supported **<data_length>s),(**list of supported **<input_time>**s)**,(**list of supported **<rsptime>**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** After TA switches to transparent transmission mode (that is, data mode), the HTTP(S) PUT request body can be inputted. When the total size of input data reaches **<data_length>**, TA will return to command mode and report the following result:

	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 URC: **+QHTTPPUT: <err>** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	Determined by network and **<rsptime>**
Characteristics	The command takes effect immediately; The configuration is not saved.

Parameter

<data_length>	Integer type. The length of the PUT request body. Range: 1–1024000. Unit: byte.
<input_time>	Integer type. The maximum input time for PUT request body. Range: 1–65535. Default value: 60. Unit: second.
<rsptime>	Integer type. Timeout value for URC **+QHTTPPUT: <err>[,<httprspcode>[,<content_length>]]** to be outputted. Range: 1–65535. Default value: 60. Unit: seconds.
<httprspcode>	HTTP(S) server response code. See ***Chapter 5*** for details.
<content_length>	Integer type. HTTP(S) response body length. Unit: byte.
<err>	Error code. See ***Chapter 5*** 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 from the HTTP(S) 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: (**list of supported **<wait_time>**s**)**

	OK
Write Command **AT+QHTTPREAD[=<wait_time>]**	Response If the parameter format is correct: **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	Determined by **<wait_time>**
Characteristics	The command takes effect immediately; The configuration is not saved.

Parameter

<wait_time>	Integer type. Maximum interval time between receiving two packets. Range: 1–65535. Default value: 60. Unit: second.
<err>	Error codes. See ***Chapter 5*** for details.

2.6.6. AT+QHTTPSTOP Cancel HTTP(S) Request

MCU uses this command to cancel HTTP(S) GET/POST/PUT request and disconnect the session with HTTP(S).

AT+QHTTPSTOP Cancel HTTP(S) Request	
Test Command **AT+QHTTPSTOP=?**	Response **OK**
Execution Command **AT+QHTTPSTOP**	Response **OK** Or **+CME ERROR: <err>**
Maximum Response Time	10 s
Characteristics	This command takes effect immediately; The configuration is not saved.

Parameter

<err>	Error codes. See ***Chapter 5*** for details.

2.7. Description of AT+QPING AT Command

2.7.1. AT+QPING PING a Remote Host

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

AT+QPING PING a Remote Host	
Test Command **AT+QPING=?**	Response **+QPING: <host>,(**list of supported **<timeout>**s**),(**list of supported **<pingnum>**s**)** **OK**
Write Command **AT+QPING=<host>[,<timeout>[,<pingnum>]]**	Response If the remote host is pinged 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	Determined by **<timeout>**
Characteristics	This command takes effect immediately; The configurations are not saved.

Parameter

<host>	Integer type. Host address, the format is a domain name or an IP address (must be dotted decimal format).
<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 number of Ping requests. Range:1–10. Default value: 4.

<ping_result> Integer type. Result for each Ping request.
0 A Ping response is received from the remote host, followed by **,<IP_address>,<bytes>,<time>,<ttl>**.
Others Error codes, see ***Chapter 5*** for details.

<IP_address> String type. IP address of the remote host. The format is dotted decimal.

<bytes> Integer type. The length of the Ping request to be sent. Unit: byte.

<time> Integer type. The response wait time after sending a Ping request. Unit: ms.

<ttl> Integer type. TTL value of Ping request response packet.

<finresult> Integer type. The final result after executing the command.
0 Works normally and the host is reached. In this case, followed by **,<sent>,<rcvd>,<lost>,<min>,<max>,<avg>**.
Others Error codes, see ***Chapter 5*** 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. Description of DNS Related AT Command

2.8.1. AT+QIDNSGIP Get IP Address Using Domain Name

This command gets the IP address using domain name. Before querying DNS, the host should first connect to the Wi-Fi.

.AT+QIDNSGIP Get IP Address Using 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	60 s, determined by the network.
Characteristics	-

Parameter

<hostname>	String type. Domain name
<err>	Error codes. See ***Chapter 5*** for details.
<IP_count>	Integer type. The number of IP addresses corresponds to **<hostname>**.
<DNS_TTL>	Integer type. DNS TTL value. Unit: second.
<hostIPaddr>	String type. IP address of **<hostname>**.

2.9. Description of NTP AT Commands

2.9.1. AT+QNTP Synchronize Local Time Through NTP Server

This command synchronizes local time to Coordinated Universal Time (UTC) through the NTP server.

AT+QNTP Synchronize Local Time Through NTP Server	
Test Command **AT+QNTP=?**	Response **+QNTP: <server>,**(list of supported **<port>**s)**,**(list of supported **<auto_set_time>**s) **OK**
Write Command **AT+QNTP=<server>[,<port>[,<auto_set_time>]]**	Response If the synchronization is successful: **OK** **+QNTP: <err>,<time>** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	-

Parameter

<server>	String type. NTP server address, the format of a domain name or an IP address (must be dotted decimal). Maximum length: 50 bytes.

<port>	Integer type. NTP server port number. Range: 1–65535. Default value: 123.
<auto_set_time>	Integer type. Whether to automatically synchronize local time to UTC 0 Not automatically synchronize 1 Automatically synchronize
<time>	String type. Time synchronized from the NTP server The format is "YY/MM/DD,hh:mm:ss", where (YY) represents the year, (MM) represents the month, (DD) represents the day, (hh) represents the hour, (mm) represents the minutes, (ss) represents the seconds, (±zz) represents the time zone.
<err>	Error codes. See ***Chapter 5*** for details.

NOTE

When **<auto_set_time>** is 1, RTC will automatically update to the synchronization time.

2.10. Description of FILE Related AT Commands

2.10.1. AT+QFOPEN Open a File

This command opens a file and obtains the file handle used in the command. Related commands include **AT+QFREAD**, **AT+QFWRITE**, **AT+QFSEEK**, **AT+QFPOSITION** and **AT+QFCLOSE**.

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

Maximum Response Time	300 ms
Characteristics	The command takes effect immediately; The configurations are not saved.

Parameter

<file_name>	String type. Name of the file to be opened in UFS. Maximum length is 80 bytes
<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 file in both cases is 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 file in both cases is 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 reported.

2.10.2. 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>** After TA switches to data mode, when the total data size exceeds **<length>**, TA will switch back to command mode: **OK** If there is any error: **ERROR**
Maximum Response Time	5 s
Characteristics	The command takes effect immediately; The configurations are not saved.

Parameter

<filehandle>	Integer type. File handle. The data type is 4 bytes.
<length>	Integer type. The expected length of the file to be read, Default value: the length of the file. Unit: byte.
<read_length>	Integer type. Actual read file length. Unit: byte.

2.10.3. 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** After TA switches to data mode, when the total size of written data exceeds **<length>** 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 s
Characteristics	The command takes effect immediately; The configurations are not saved.

Parameter

<filehandle>	Integer type. File handle. The data type is 4 bytes.
<length>	Integer type. The expected length of the file to be written. Range: 1–65535. Default value: 10240. Unit: byte.
<timeout>	Integer type. Waiting time for data input via USB/UART. Range: 1–180. Default value: 5. Unit: second.
<written_length>	Integer type. The length of the actual data written. Unit: byte.
<total_length>	Integer type. Total file length. Unit: byte.

2.10.4. AT+QFSEEK Set File Pointer Position

This command sets the file pointer to the specified position and determines the initial position of these commands, such as **AT+QFREAD**, **AT+QFWRITE** and **AT+QFPOSITION**.

AT+QFSEEK Set File Pointer Position	
Test Command **AT+QFSEEK=?**	Response **+QFSEEK: <filehandle>,<offset>,(**list of supported **<position>**s**)** **OK**
Write Command **AT+QFSEEK=<filehandle>,<offset>[,<position>]**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately; The configurations are not saved

Parameter

<filehandle> Integer type. File handle. The data type is 4 bytes.

<offset> Integer type. The number of bytes moved by the file pointer

<position> Integer type. File pointer movement mode.
0 Move from the initial location of the file
1 Move from the current position of file pointer
2 Move from the final location of the file

NOTE

If **<position>** is 2, the file handle will move forward.

2.10.5. AT+QFPOSITION Get File Pointer Offset

This command gets the offset of the file pointer from the initial position of the file.

AT+QFPOSITION Get File Pointer Offset	
Test Command **AT+QFPOSITION=?**	Response **+QFPOSITION: <filehandle>** **OK**

Write Command **AT+QFPOSITION=<filehandle>**	Response **+QFPOSITION: <offset>** **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately; The configuration is not saved.

Parameter

<filehandle>	Integer type. File handle. The data type is 4 bytes.
<offset>	Integer type. Offset of file pointer from the initial position.

2.10.6. AT+QFCLOSE Close the File

This command closes the file or ends 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** Or **ERROR**
Maximum Response Time	300ms
Characteristics	The command takes effect immediately; The configuration is not saved.

Parameter

<filehandle>	Integer type. File handle. The data type is 4 bytes.

2.10.7. AT+QFLST List File Information in Storage Medium

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

AT+QFLST List File Information in Storage Medium	
Test Command **AT+QFLST=?**	Response **OK**
Write Command **AT+QFLST=<name_pattern>**	Response **+QFLST: <file_name>,<file_size>** **[+QFLST: <file_name>,<file_size>** **[…]]** **OK** If there is any error: **ERROR**
Execution Command **AT+QFLST**	Response Returns all file information stored in UFS: **+QFLST: <file_name>,<file_size>** **[+QFLST: <file_name>,<file_size>** **[…]]** **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately; The configuration is not saved.

Parameter

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

2.10.8. AT+QFDEL Delete a File in Storage Medium

This command deletes a file in the current storage medium.

AT+QFDEL Delete a File in Storage Medium	
Test Command **AT+QFDEL=?**	Response **+QFDEL: <file_name>** **OK**
Write Command **AT+QFDEL=<file_name>**	Response **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately; The configuration is not saved.

Parameter

<file_name>	String type. File name.

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 the STA state changes.

Parameter

<event>	String types. Event reported when the STA state changes.	
	"WLAN DISCONNECTED"	STA disconnected
	"WLAN_CONNECTED"	STA connected
	"GOT_IP",2.4G/5G	STA got IP (2.4G/5G: 2.4 GHz or 5 GHz)
	"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 the AP state changes.

Parameter

<event>	String types. Event reported when the AP state changes.	
	"AP_UP"	AP enabled successfully
	"AP_DOWN"	AP disabled successfully
	"AP_CONNECT"	STA connected to AP
	"AP_DISCONNECT"	STA disconnected from AP

3.1.3. +QOTASTAT URC Indicating OTA State Change

+QOTASTAT URC Indicating OTA State Change	
+QOTASTAT: <event>	This URC indicates the OTA upgrade state changes

Parameter

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

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 the BLE state changes.

Parameter

<event>	String type. URC reported when the BLE state 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 MTU value changes of BLE.

Parameter

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

3.2.3. +QBLEINFO URC Indicating Peer Service Discovery Result Upon Successful Establishment of BLE Connection

When the module successfully connects to the peer device as a central device, the URC is reported to display the characteristic UUID of the peer device.

+QBLEINFO URC Indicating Peer Service Discovery Result Upon Successful Establishment of BLE Connection	
+QBLEINFO: <UUID>	The URC indicates the peer service discovery result upon successful establishment of the BLE connection.

Parameter

<UUID>	String type. "DOWN" Completion of discovering the peer device service Others Characteristic UUID of peer device. Length: 4 bytes or 32 bytes.

3.2.4. +QBLEPEERINFO URC Indicating Receiving Messages from Peer

This URC indicates BLE device receives messages from the peer after BLE connection, BLE disconnection or MTU update.

+QBLEPEERINFO URC Indicating Receiving Messages from Peer	
+QBLEPEERRINFO: <conn_idx>,<peer_role>,<BLE_addr>	This URC indicates BLE device receives messages from the peer after BLE connection, BLE disconnection or MTU update.

Parameter

<conn_idx>	Integer type. The target device index assigned by the module automatically when the BLE connection is established. Range: 0–1. Default value: 0.
<peer_role>	Integer type. Peer device role. 0 Central device 1 Peripheral device
<BLE_addr>	String type. Peer device address. A 48-bit address is represented in a string of hexadecimal numbers, such as "58D391010203".

3.2.5. +QBLERECV URC Indicating BLE Device Receives Data

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

Parameter

<conn_idx>	Integer type. The target device index assigned by the module automatically when the BLE connection is established. Range: 0–1. Default value: 0.
<UUID>	String type. Characteristic UUID. Length: 4 bytes or 32 bytes.
<length>	Integer type. Length of received data. Unit: byte.
<data>	String type without double quotation marks. Received data

3.2.6. +QBLECONNPARAM URC Indicting Receiving Connection Parameters After BLE Connection Established

+QBLECONNPARAM URC Indicting Receiving Connection Parameters After BLE Connection Established	
+QBLECONNPARAM: <conn_idx>,<con_interval>,<timeout>,<latency>	This URC indicates receiving connection parameters after the BLE connection is established.

Parameter

<conn_idx>	Integer type. The target device index assigned by the module automatically when the BLE connection is established. Range: 0–1. Default value: 0.
<con_interval>	Integer type. Connection interval.
<timeout>	Integer type. BLE Link supervision timeout.
<latency>	Integer type. Number of connection events delayed by the peripheral device.

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 reported, and the status of the socket service will be in "Closing" state (**<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. The socket service index. Range: 0–5.

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

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

- In buffer access mode: The URC format is **+QIURC: "recv",<connectID>**; after the URC is reported, the host can read the data through **AT+QIRD**. Please note that if the buffer is not empty and the module receives data again, the module will report a new URC only after the host reads all received data through **AT+QIRD**.
- In direct push 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>,<current_recvlength><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. The socket service index. Range: 0–5.
<current_recvlength>	Integer type. The length of the actual received data.
<remoteIP>	String type. Remote IP address.
<remote_port>	Integer type. Remote port.
<data>	Integer type. The received data. Unit: byte.

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 as **+QIURC: "incoming full"** for the new incoming connection request.

+QIURC: "incoming full" URC Indicating Incoming Connection Full	
+QIURC: "incoming full"	This URC indicates the client connection request 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 an idle **<connectID>** to the new connection, where the range of **<connectID>** is 0–5. At this time, the module will report this 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 reported by this URC.

Parameter

<connectID>	Integer type. The socket service index. Range: 0–5.
<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 **<connectID>**.
<remote_port>	Integer type. Remote port of **<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>	The URC reported when receiving data in buffer access mode. SSL data can be received via **AT+QSSLRECV**.
+QSSLURC: "recv",<clientID>,<current_recvlength><CR><LF><data>	The URC reported when receiving data in direct push mode.

Parameter

<clientID>	Integer type. Socket ID. Range: 0–5.
<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 notifies 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>	The URC reported when the SSL connection based on the specified socket has been closed.

Parameter

<clientID>	Integer type. Socket ID. Range: 0–5.

3.5. MQTT Related URCs

3.5.1.+QMTSTAT URC Indicating MQTT Link Layer State Change

The URC indicates MQTT link layer state 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 State Change	
+QMTSTAT: <client_idx>,<err_code>	When the MQTT link layer state changes, the client will disconnect the MQTT connection and report this URC.

Parameter

<client_idx>	Integer type. MQTT client index. 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 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 MQTT Packet Data Read by Host

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

+QMTRECV URC Indicating MQTT Packet Data Read by Host	
+QMTRECV:<client_idx>,<msgID>,<topic>[,<payload_length>],<payload>	Notify the host to read the data packet sent by the MQTT Server.

+QMTRECV: <client_idx>,<recvID>	This URC is reported when the message received from the MQTT server is stored in the buffer.

Parameter

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

3.6. HTTP(S)-Related URCs

3.6.1. +QHTTPURC: "recv" URC Indicating Incoming Response Data

This URC is reported when the module receives HTTP data. Only when the automatic output of the HTTP(S) response header is enabled in **AT+QHTTPCFG="rspout/auto"** will the URC be reported.

+QHTTPURC: "recv" URC Indicating Incoming Response Data	
+QHTTPURC: "recv",<length>	This URC indicates incoming response data.

Parameter

<length>	Integer type. Length of received data. Unit: byte.

3.6.2. +QHTTPURC: "closed" URC Indicating HTTP Session Closing

This URC is reported to indicate that the HTTP session is closed. The reasons for closing include successful disconnection by HTTP request response or HTTP response timeout exception. Only when the report of the HTTP(S) session closing URC is enabled in **AT+QHTTPCFG="closed/ind"** will the URC be reported.

+QHTTPURC: "closed" URC Indicating HTTP Session Closing	
+QHTTPURC: "closed"	This URC is reported to indicate that the HTTP session is closed.

3.6.3. +QHTTPGET URC Indicating Result of GET Request Sending

After sending a GET request with **AT+QHTTPGET**, the URC is reported to indicate the result of GET request sending.

+QHTTPGET URC Indicating Result of GET Request Sending	
+QHTTPGET: <err>[,<httprspcode>[,<content_length>]]	The URC is reported to indicate the sending result of GET request.

Parameter

<err>	Result code. See ***Chapter 5*** for details.
<httprspcode>	HTTP(S) server response code. See ***Chapter 5*** for details.
<content_length>	Integer type. Length of HTTP(S) response body. Unit: byte.

3.6.4. +QHTTPPOST URC Indicating Result of POST Request Sending

After sending a POST request with **AT+QHTTPPOST**, the URC is reported to indicate the result of POST request sending.

+QHTTPPOST URC Indicating Result of POST Request Sending	
+QHTTPPOST: <err>[,<httprspcode>[,<content_length>]]	The URC is reported to indicate the result of POST request sending.

Parameter

<err>	Result code. See ***Chapter 5*** for details.
<httprspcode>	HTTP(S) server response code. See ***Chapter 5*** for details.
<content_length>	Integer type. Length of HTTP(S) response body. Unit: byte.

3.6.5. +QHTTPPUT URC Indicating Result of PUT Request Sending

After sending a PUT request with **AT+QHTTPPUT**, the URC is reported to indicate the result of the PUT request sending.

+QHTTPPUT URC Indicating Result of PUT Request Sending	
+QHTTPPUT: <err>[,<httprspcode>[,<content_length>]]	The URC is reported to indicate the result of PUT request sending.

Parameter

<err>	Result code. See ***Chapter 5*** for details.
<httprspcode>	HTTP(S) server response code. See ***Chapter 5*** for details.
<content_length>	Integer type. Length of HTTP(S) response body. Unit: byte.

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

This Chapter describes that the BLE device, which accepts the request to establish an active physical connection, is defined as 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=2,1                   //Set the module as a peripheral device for initializing BLE.
OK
AT+QBLEADVPARAM=150,150           //Set BLE advertising parameters.
OK
AT+QBLENAME="QuecFCM362K"         //Set BLE name.
OK
AT+QBLEADDR?                      //Query and obtain BLE address.
+QBLEADDR: "09c000455667"

OK
AT+QBLEGATTSSRV="FFF1"            //Establish a BLE service and set the service UUID to FFF1.
OK
AT+QBLEGATTSCHAR="FFF2"           //Set GATT characteristic UUID to FFF2.
OK
```

```
AT+QBLEGATTSSRVDONE                   //Peripheral device completes creating BLE service.
OK
AT+QBLEADVSTART                       //Start BLE advertising.
OK
```

4.2.1.2.nRF Connect Connectivity

1. Click "**SCAN**" to scan for peripheral devices, select "**QuecFCM362K**" in the scan results and then click "**CONNECT**", as shown in the figure below：

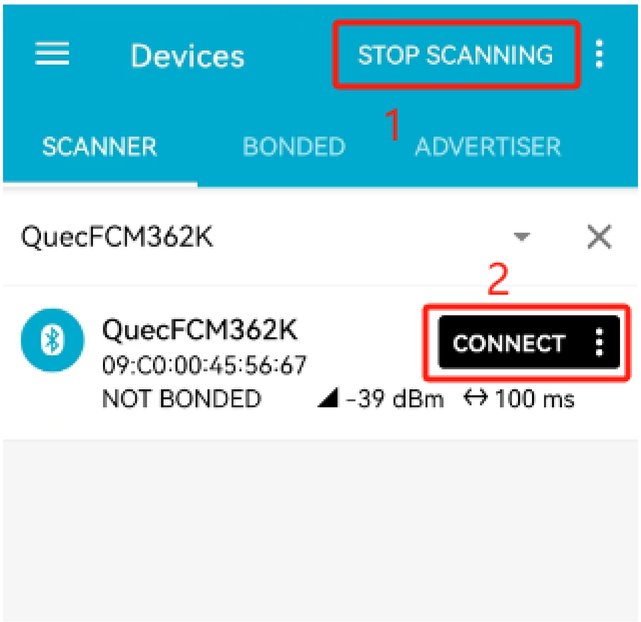
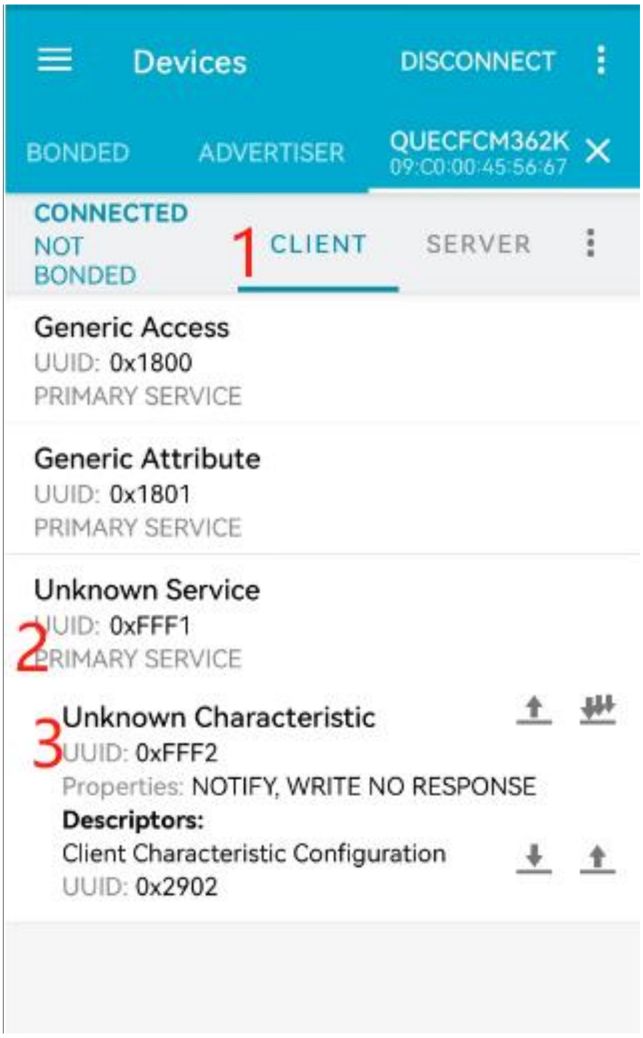


Figure 3: Scanning Result

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



Devices DISCONNECT

BONDED ADVERTISER QUECFCM362K 09:C0:00:45:56:67

CONNECTED
NOT BONDED

CLIENT SERVER

Generic Access
UUID: 0x1800
PRIMARY SERVICE

Generic Attribute
UUID: 0x1801
PRIMARY SERVICE

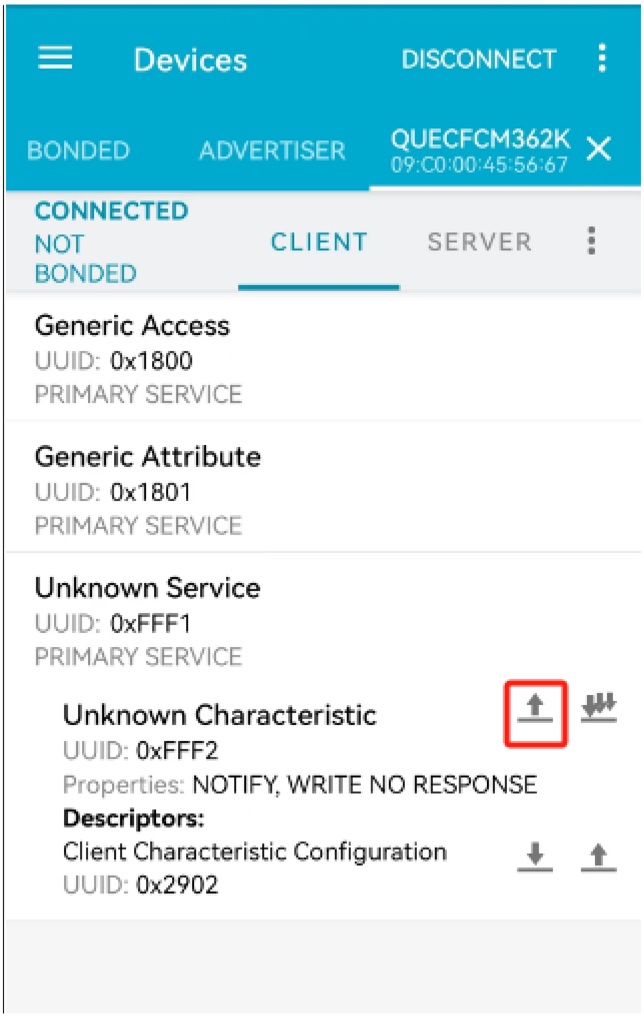
Unknown Service
UUID: 0xFFF1
PRIMARY SERVICE

Unknown Characteristic
UUID: 0xFFF2
Properties: NOTIFY, WRITE NO RESPONSE
Descriptors:
Client Characteristic Configuration
UUID: 0x2902

Figure 4: Connected Successfully

4.2.1.3.Send data to the module

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



Devices DISCONNECT

BONDED ADVERTISER QUECFCM362K 09:C0:00:45:56:67

CONNECTED NOT BONDED CLIENT SERVER

Generic Access
UUID: 0x1800
PRIMARY SERVICE

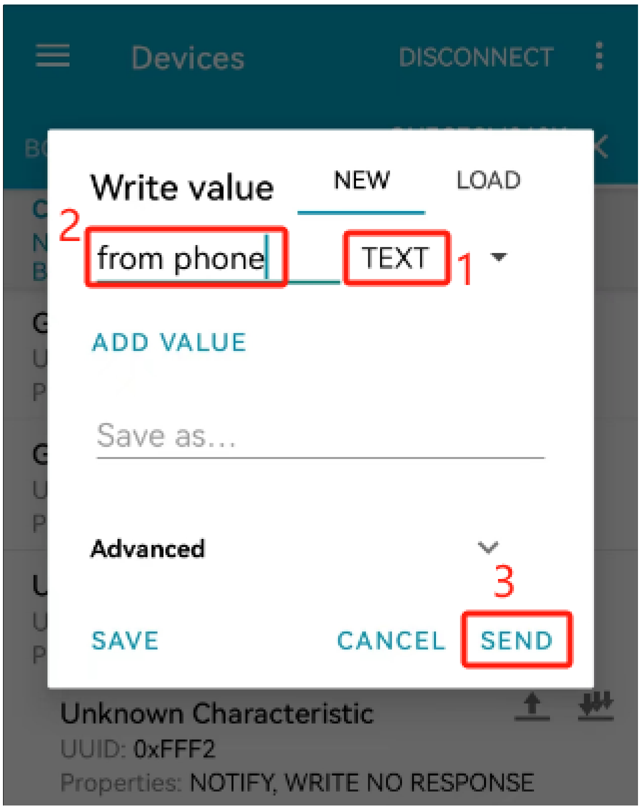
Generic Attribute
UUID: 0x1801
PRIMARY SERVICE

Unknown Service
UUID: 0xFFF1
PRIMARY SERVICE

Unknown Characteristic
UUID: 0xFFF2
Properties: NOTIFY, WRITE NO RESPONSE
Descriptors:
Client Characteristic Configuration
UUID: 0x2902

Figure 5: Edit Characteristic

2. Select the data format as "TEXT", and input the data to be sent in "Write value". Then click "**SEND**":



Write value

from phone

TEXT

Save as…

Unknown Characteristic

UUID: 0xFFF2

Properties: NOTIFY, WRITE NO RESPONSE

Figure 6: Send Data

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

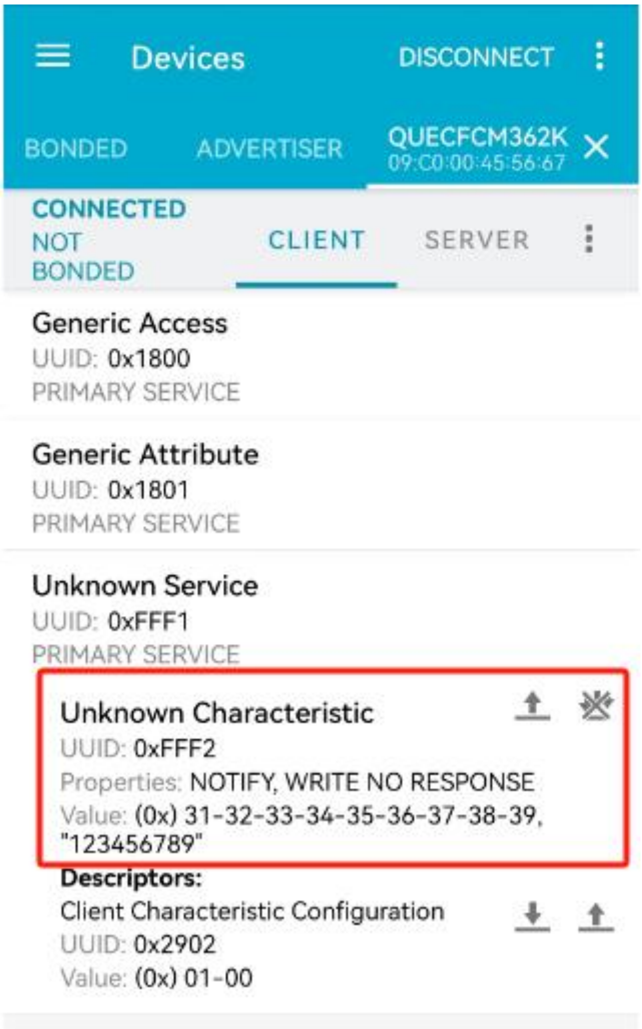
```
+QBLERECV: 0,"fff2",10
from phone
```

4.2.1.4.Send data to nRF Connect

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

```
AT+QBLEGATTSNTFY=0,"FFF2","123456789"
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 NO RESPONSE
Value: (0x) 31-32-33-34-35-36-37-38-39, "123456789"

Descriptors:
Client Characteristic Configuration
UUID: 0x2902
Value: (0x) 01-00

Figure 7: 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","192.0.2.2",8009,0,2	//**<connectID>** is 0. Before **AT+QIOPEN** is executed, the host needs to use **AT+QSTAAPINFO** to connect to the Wi-Fi.
CONNECT	//The connection is successful. It is recommended to wait 150 seconds for the URC response **CONNECT**. If there is no URC response within 150 seconds, the host should 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 the remote server in transparent transmission mode.

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

Disconnect from TCP client.

```
AT+QICLOSE=0                //After exiting transparent transmission mode through +++, the host can
                             disconnect the TCP connection through AT+QICLOSE. Determined by
                             the network, 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

OK
AT+QFCLOSE=1
OK
AT+QSSLCFG="cacert",1,"ca.pem"         //Set the trusted CA certificate path for SSL context.
OK
```

4.4.2. SSL Client in Buffer Access Mode

```
//Establish an SSL connection and enter buffer access mode
AT+QSSLOPEN=1,4,"192.0.2.2",8010,0
OK

+QSSLOPEN: 4,0                   //SSL connection is successfully established.
AT+QSSLSTATE                     //Query the status of all SSL connections.
+QSSLSTATE: 4,"SSLClient","192.0.2.2",8010,65344,2,4,0,"uart1",1

OK
//Send data in buffer 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 buffer access mode
+QSSLURC: "recv",4

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 the buffer.
OK
//Close SSL connection
AT+QSSLCLOSE=4                   //Close the SSL connection (<clientID>=4). Determined by the
                                   network, the maximum response time is 10 seconds.
OK
```

4.4.3. SSL Client in Direct Push Mode

```
//Establish an SSL connection and enter direct push mode
AT+QSSLOPEN=1,4,"192.0.2.2",8011,1
OK
```

```
+QSSLOPEN: 4,0                      //SSL connection is successfully established.
AT+QSSLSTATE                        //Query the status of all SSL connections.
+QSSLSTATE: 4,"SSLClient","192.0.2.2",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 data and the data length is 18 bytes.
>
Test data from SSL
SEND OK
//Receive data in direct push mode
+QSSLURC: "recv",4,18
Test data from SSL
//Close SSL connection
AT+QSSLCLOSE=4                      //Close the SSL connection (<clientID>=4). Determined by
                                     the network, 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,"192.0.2.2",8011,2        //Establish an 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,"192.0.2.2",8011,2        //Establish an 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).
                                          Determined by 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

```
//Set the reception mode as MQTT messages received from the server are reported in the form of URC.
AT+QMTCFG="recv/mode",0,0
OK
AT+QMTOPEN=?
+QMTOPEN: (0-5),<hostname>,(1-65535)

OK
//Open MQTT client network
AT+QMTOPEN=0,"192.0.2.2",8306
OK

+QMTOPEN: 0,0                                   //MQTT client has successfully opened the
                                                network.
AT+QMTOPEN?
+QMTOPEN: 0,"192.0.2.2",8306

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

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

+QMTCONN: 0,0,0                                 //The MQTT client is successfully connected to
                                                 the 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 server.
AT+QMTDISC=0
OK

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

4.5.2. Example of MQTT Operation with SSL

```
//Set the reception mode as MQTT messages received from the server are reported in the form of URC.
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 to 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 to 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 to 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                //Set SSL authorization method to server authentication.
OK
AT+QSSLCFG="sslversion",2,4              //Set SSL authorization version to all SSL versions.
OK
AT+QSSLCFG="ciphersuite",2,0xFFFF        //Set cipher suite to supporting all cipher-suites.
OK
AT+QSSLCFG="ignorelocaltime",2,1         //Ignore authorization time.
OK
AT+QSSLCFG="cacert",1,"ca.pem"           //Set the trusted CA certificate path for the SSL context.
OK
AT+QSSLCFG="clientcert",1,"user.pem"     //Set the client certificate path for the SSL context.
OK
AT+QSSLCFG="clientkey",1,"user_key.pem"  //Set the client key for the SSL context.
OK
//Start MQTT SSL connection
AT+QMTOPEN=0,"192.0.2.2",8307
OK

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

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

+QMTSUB: 0,1,0,1
//Publish the 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 "$aws/things/M26_0206/shadow/update/accepted" and other
devices publish a message with the same topic to the server, the module will report the following
information:
+QMTRECV: 0,1,"$aws/things/M26_0206/shadow/update/accepted",32,"This is publish data from
client"

//Disconnect the client from the 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.example.com"   //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 contains Content-Length,
                                           <content_length> information will be reported.

//Example of reading HTTP GET response information.
//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 the 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://example.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 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 the HTTP session to be
                                       closed is 80 seconds.
CONNECT
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="httpHTTPs://example.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://example.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 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 the HTTP session to be
                                       closed is 80 seconds.
CONNECT
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="httpHTTPs://example.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 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 to read the HTTPS GET response information.
AT+QHTTPCFG="rsp/header",1                //Enable the 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, indicating 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.example.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 contains Content-Length,
                                          <content_length> information will be reported.

//Example of reading HTTPS response information
//Read HTTPS response information and output it through UART.
AT+QHTTPREAD=80                           //Read the HTTPS response information and output it
```

```
                                           through UART. Maximum wait time for the 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"               //Set the trusted CA certificate path for the SSL context.
OK
AT+QSSLCFG="clientcert",1,"user.pem"         //Set the client certificate path for the SSL context.
OK
AT+QSSLCFG="clientkey",1,"user_key.pem"      //Set the client key for the SSL context.
OK
AT+QHTTPCFG="url","https://192.0.2.2:8011/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 contains Content-Length,
                                         <content_length> information will be reported.
AT+QHTTPREAD=80                          //Read the HTTPS response information and output it
                                         through UART. Maximum wait time for the 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 outputted 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, indicating 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"               //Set the trusted CA certificate path for the SSL context.
OK
AT+QSSLCFG="clientcert",1,"user.pem"         //Set the client certificate path for the SSL context.
OK
AT+QSSLCFG="clientkey",1,"user_key.pem"      //Set the client key for the SSL context.
OK
AT+QHTTPCFG="url","https://192.0.2.2:8011/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 contains Content-Length,
                                       <content_length> information will be reported.
AT+QHTTPREAD=80                        //Read the HTTPS response information and output it through
                                       UART. Maximum wait time for the 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.
```

5 Result Code

Table 3: TCP/UDP/SSL Result Codes

Result Code	Description
0	Operation successful
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
521	Timeout

Table 4: HTTP(S) Result Codes

Result Code	English Description
0	Operation successful
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 creation error
712	HTTP(S) socket connection error
713	HTTP(S) socket read error
714	HTTP(S) socket write error
715	HTTP(S) socket is 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
DTIM	Delivery Traffic Indication Message
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 Technology
PSK	Pre-Shared Key
QoS	Quality of Service
RAM	Random Access Memory

RI	Ring Indicator
SNI	Server Name Indication
SSID	Service Set Identifier
SSL	Secure Sockets Layer
STA	Station
TA	Terminal Adapter
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UFS	Universal Flash Storage
URC	Unsolicited Result Code
UUID	Universally Unique Identifier