

# EG800Q&EG91xQ Series

## TCP/IP Application Note

**LTE Standard Module Series**

Version: 1.2

Date: 2024-11-22

Status: Released



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](mailto: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](mailto: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.

**Copyright © Quectel Wireless Solutions Co., Ltd. 2024. All rights reserved.**

# About the Document

## Revision History

Version	Date	Author	Description
-	2022-12-06	Orange LI	Creation of the document
1.0	2023-02-13	Orange LI	First official release
1.1	2024-07-30	Orange LI	<ol style="list-style-type: none"> <li>Updated applicable modules: <ul style="list-style-type: none"> <li>Added EG916Q-GL;</li> <li>Updated EG800Q-EU to EG800Q Series;</li> <li>Updated EG915Q-NA to EG915Q Series.</li> </ul> </li> <li>Added the following Write Commands (Chapter 2.3.1): <ul style="list-style-type: none"> <li>AT+QICFG="send/auto"</li> <li>AT+QICFG="formatcfg"</li> <li>AT+QICFG="close/mode"</li> <li>AT+QICFG="tcp/delay_ack"</li> <li>AT+QICFG="tcp/twrecycle"</li> <li>AT+QICFG="dns/retry"</li> </ul> </li> </ol>
1.2	2024-11-22	Elmo HUANG	<ol style="list-style-type: none"> <li>Updated the declaration of AT command examples (Chapter 2.2).</li> <li>Updated URLs in the examples (Chapter 3).</li> </ol>

## Contents

About the Document.....	3
Contents .....	4
Table Index .....	6
<b>1 Introduction .....</b>	<b>7</b>
1.1. Using TCP/IP AT Commands.....	7
1.2. Description of Data Access Modes.....	9
<b>2 Description of TCP/IP AT Commands .....</b>	<b>11</b>
2.1. AT Command Introduction .....	11
2.1.1. Definitions.....	11
2.1.2. AT Command Syntax .....	11
2.2. Declaration of AT Command Examples .....	12
2.3. Description of TCP/IP AT Commands.....	12
2.3.1. AT+QICFG Configure Optional Parameters.....	12
2.3.2. AT+QICSGP Configure Parameters of a TCP/IP Context .....	21
2.3.3. AT+QIACT Activate a PDP Context .....	22
2.3.4. AT+QIDEACT Deactivate a PDP Context .....	23
2.3.5. AT+QIOPEN Open a Socket Service .....	24
2.3.6. AT+QICLOSE Close a Socket Service.....	26
2.3.7. AT+QISTATE Query Socket Service Status.....	27
2.3.8. AT+QISEND Send Data .....	29
2.3.9. AT+QIRD Retrieve Received TCP/IP Data .....	31
2.3.10. AT+QISENDEX Send Hex String Data .....	32
2.3.11. AT+QIACCEPT Manually Accept New TCP Incoming Connection.....	33
2.3.12. AT+QISWTMD Switch Data Access Mode.....	34
2.3.13. AT+QPING Ping a Remote Host .....	35
2.3.14. AT+QNTTP Synchronize Local Time with NTP Server .....	36
2.3.15. AT+QIDNSCFG Configure Address of DNS Server.....	37
2.3.16. AT+QIDNSGIP Get IP Address by Domain Name .....	38
2.3.17. AT+QISDE Control Whether to Echo Data for AT+QISEND .....	39
2.3.18. AT+QIGETERROR Query Last Error Code .....	40
2.4. Description of URCs .....	40
2.4.1. +QIURC: "closed" Indicate Closed Connection .....	41
2.4.2. +QIURC: "recv" Indicate Incoming Data .....	41
2.4.3. +QIURC: "incoming full" Indicate Incoming Connection Reached Limit.....	42
2.4.4. +QIURC: "incoming" Indicate Incoming Connection in Automatic Acceptance Mode.....	42
2.4.5. +QIURC: "accept" Indicate Incoming Connection in Manual Acceptance Mode .....	43
2.4.6. +QIURC: "pdpdeact" Indicate PDP Deactivation .....	43
<b>3 Examples .....</b>	<b>44</b>
3.1. Configure and Activate a Context .....	44
3.1.1. Configure a Context .....	44

3.1.2.	Activate a Context .....	44
3.1.3.	Deactivate a Context.....	44
3.2.	TCP Client Works in Buffer Access Mode under IPv4 .....	44
3.2.1.	Set up a TCP Client Connection and Enter Buffer Access Mode .....	44
3.2.2.	Send Data in Buffer Access Mode .....	45
3.2.3.	Retrieve Data from Remote Server in Buffer Access Mode .....	45
3.2.4.	Close a Connection.....	46
3.3.	TCP Client Works in Transparent Transmission Mode.....	46
3.3.1.	Set up a TCP Client Connection and Enter Transparent Transmission Mode .....	46
3.3.2.	Send Data in Transparent Transmission Mode .....	47
3.3.3.	Retrieve Data from Remote Server in Transparent Transmission Mode.....	47
3.3.4.	Close a TCP Client.....	47
3.4.	TCP Client Works in Direct Push Mode.....	47
3.4.1.	Set up a TCP Client Connection and Enter Direct Push Mode .....	47
3.4.2.	Send Data in Direct Push Mode.....	48
3.4.3.	Receive Data from Remote Server in Direct Push Mode .....	48
3.4.4.	Close a TCP Client.....	48
3.5.	TCP Server Works in Buffer Access Mode under IPv4 .....	48
3.5.1.	Start a TCP Server .....	48
3.5.2.	Accept TCP Incoming Connection .....	49
3.5.3.	Retrieve Data from Incoming Connection.....	49
3.5.4.	Close a TCP Server .....	49
3.6.	UDP Service under IPv4 .....	50
3.6.1.	Start a UDP Service .....	50
3.6.2.	Send UDP Data to Remote Client.....	50
3.6.3.	Retrieve Data from Remote Client.....	50
3.6.4.	Close a UDP Service .....	51
3.7.	PING under IPv4 .....	51
3.8.	Synchronize Local Time.....	51
3.9.	Get Last Error Code .....	52
<b>4</b>	<b>Summary of Error Codes .....</b>	<b>53</b>
<b>5</b>	<b>Appendix References .....</b>	<b>55</b>

## Table Index

Table 1: Types of AT Commands .....	11
Table 2: Summary of Error Codes.....	53
Table 3: Terms and Abbreviations .....	55

# 1 Introduction

Quectel EG800Q series and EG91xQ family (EG915Q series and EG916Q-GL) modules feature an embedded TCP/IP stack, which enables the host to access the Internet directly via AT commands. This significantly reduces the dependence on external PPP and TCP/IP protocol stacks and thus minimizes the cost.

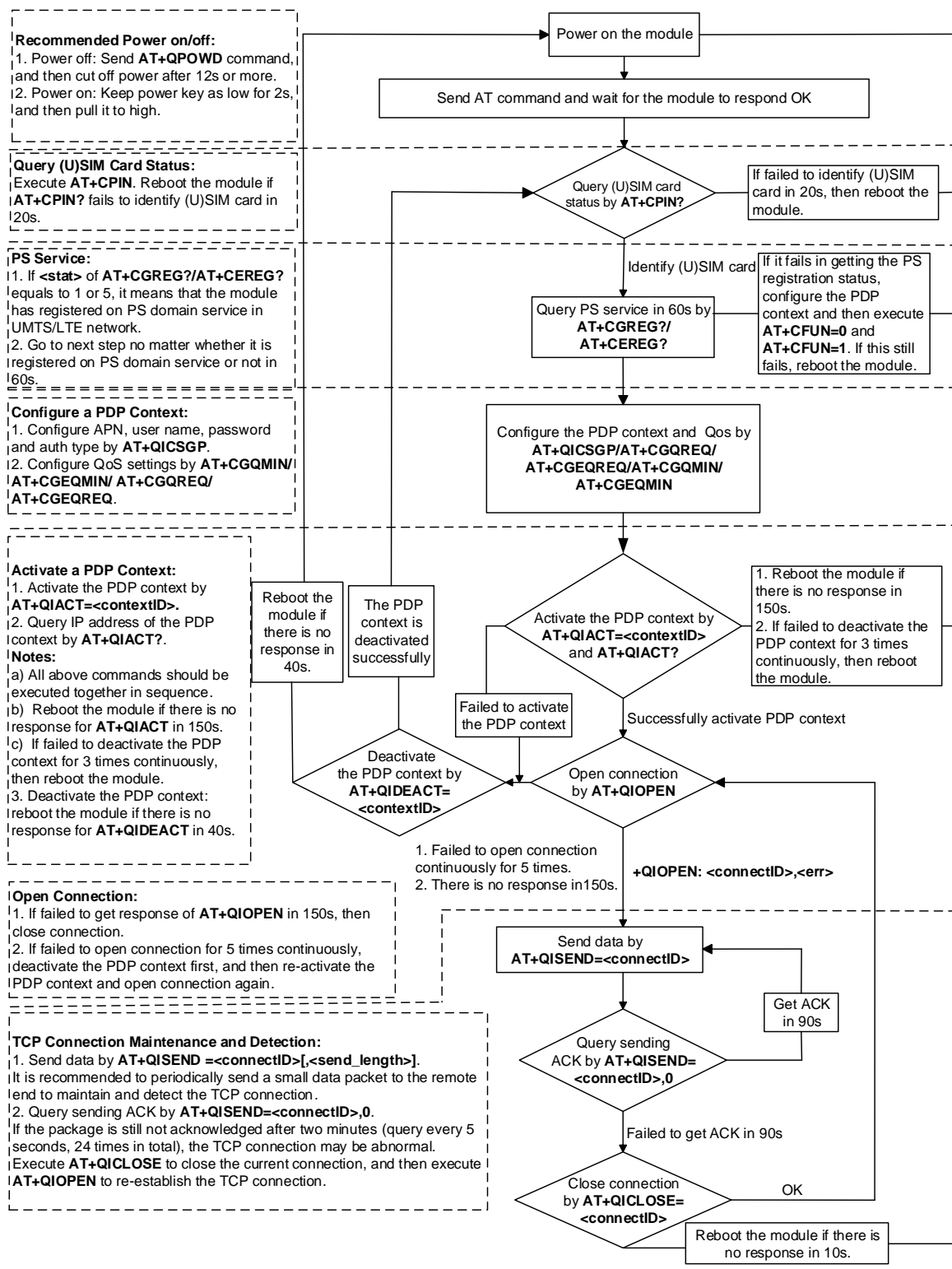
EG800Q and EG91xQ family modules provide the following socket services:

- TCP client
- UDP client
- TCP server
- UDP server

## 1.1. Using TCP/IP AT Commands

Through TCP/IP AT commands, the host can configure a PDP context, activate/deactivate the PDP context, open/close a socket service and send/retrieve data via the socket service. The following figure illustrates how to use TCP/IP AT commands.





**Notes:**

- Please note that users need to wait for the final response (for example "OK", "CME ERROR", "CMS ERROR") of the last AT command you entered before you enter the next AT command. You can reboot the module if the module fails to get response in 60s.
- Reboot the module if the module has not got response of **AT+QIAC?** in 150s or response of **AT+QICLOSE** in 10s and in 40s.
- It is NOT recommended to frequently reboot the module. When the module has been continuously rebooted for 3 times due to failed AT command execution, it can be rebooted immediately for the first time after that. If it still fails, reboot the module after 10 minutes for the second time, and reboot after 30 minutes for the third time, one hour for the fourth time, etc.

Figure 1: Flow Chart of TCP/IP AT Command Use

## 1.2. Description of Data Access Modes

EG800Q and EG91xQ family modules support three data access modes:

- Buffer access mode
- Direct push mode
- Transparent transmission mode

When opening a socket service via **AT+QIOPEN**, you can specify the data access mode with **<access\_mode>**. After the socket service is opened, the data access mode can be switched with **AT+QISWTMD**.

1. In buffer access mode, data can be sent via **AT+QISEND**, and if the module receives the data from the Internet, it will buffer them and report a URC **+QIURC: "recv",<connectID>**. Data can be read via **AT+QIRD**.
2. In direct push mode, data can be sent via **AT+QISEND**, and if the module receives the data from the Internet, the data will be outputted to COM port directly in the following format:  
**+QIURC: "recv",<connectID>,<currentrecvlength><CR><LF><data>**.
3. In transparent transmission mode, the corresponding port (such as UART, USB modem port, etc.) is exclusively used for sending/receiving data directly to the Internet. The data received from COM port will be sent to the Internet directly, and the data received from Internet will be outputted via COM port directly.

- **Exit transparent transmission mode**

To make the module exit transparent transmission mode either:

- 1) Execute **+++**. Follow the requirements below to prevent **+++** from being misinterpreted as data:
  - a) Do not input any character for at least 1 second before and after inputting **+++**.
  - b) Input **+++** within 1 s, and wait until **OK** is returned. After **OK** is returned, the module switches to buffer access mode.

**OR**

- 2) Change DTR from LOW to HIGH to make the module enter command mode (the COM port can now be used for running AT commands, as well as for sending/retrieving data). In this case, set **AT&D1** before the module enters transparent transmission mode.

- **Return back to transparent transmission mode**

To return to transparent transmission mode either:

- 1) Execute **AT+QISWTMD**. Before execution specify **<access\_mode>** as 2. Once transparent transmission mode is entered successfully, **CONNECT** is returned.

OR

- 2) Execute **ATO**. After a connection exits transparent transmission mode, executing **ATO** switches the data access mode back to transparent transmission mode. Once transparent transmission mode is entered successfully, **CONNECT** is returned. If no connection has entered transparent transmission mode before, **ATO** returns **NO CARRIER**.

**NOTE**

1. In buffer access mode, if the buffer is not empty, and the module receives data again, it does not report a new URC until all the received data have been retrieved via **AT+QIRD** from the buffer.
2. In transparent transmission mode, AT commands cannot be executed. If the socket connection is closed because of a network error or other errors, the module reports **NO CARRIER** and exits the transparent transmission mode. In this case, execute **AT+QICLOSE** to close the socket service.

## 2 Description of TCP/IP AT Commands

This chapter describes AT commands related to TCP/IP.

### 2.1. AT Command Introduction

#### 2.1.1. 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.

#### 2.1.2. 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.

Table 1: Types of AT Commands

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

## 2.2. 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.

## 2.3. Description of TCP/IP AT Commands

### 2.3.1. AT+QICFG Configure Optional Parameters

This command configures optional parameters.

AT+QICFG Configure Optional Parameters	
Test Command AT+QICFG=?	Response +QICFG: "transpktsize",(range of supported <transpktsize>s) +QICFG: "transwaittm",(range of supported <transwaittm>s) +QICFG: "tcp/retranscfg",(range of supported <max_backoffs>s),(range of supported <max_rto>s) +QICFG: "tcp/accept",(list of supported <accept_mode>s) +QICFG: "passiveclosed",(list of supported <closed>s) +QICFG: "dataformat",(list of supported <send_data_format>s),(list of supported <recv_data_format>s) +QICFG: "viewmode",(list of supported <viewmode>s) +QICFG: "qisend/timeout",(range of supported <time>s) +QICFG: "send/auto",(range of supported <connectID>s),(range of supported <period>s),<msg_auto>,(range of supported <data_format>s),(range of supported <try_times>s) +QICFG: "recvind",(list of supported <enable>s) +QICFG: "formatcfg",(list of supported <format>s) +QICFG: "close/mode",(list of supported <close_mode>s) +QICFG: "udp/sendmode",(list of supported <send_mode>s) +QICFG: "udp/readmode",(list of supported <read_mode>s) +QICFG: "tcp/keepalive",(list of supported <enable>s,(range of supported <idle_time>s),(range of supported <interval_time>s),(range of supported <probe_cnt>s) +QICFG: "tcp/delay_ack",(list of supported <delay_ack>s)

	<p>+QICFG: "dns/cache",(list of supported &lt;DNS_cache&gt;s)</p> <p>+QICFG: "tcp/twrecycle",(range of supported &lt;timewait_timeout&gt;s)</p> <p>+QICFG: "dns/retry",(range of supported &lt;retry_ack&gt;s),(range of supported &lt;retry_timeout&gt;s)</p> <p>+QICFG: "send/buffersize",(range of supported &lt;send_buffersize&gt;s)</p> <p>+QICFG: "window/scale",(list of supported &lt;window_scale_enable&gt;s)</p> <p>OK</p>
<p>Write Command</p> <p>Query/set the packet size for transparent transmission mode</p> <p><b>AT+QICFG="transpktsize"[,&lt;transpktsize&gt;]</b></p>	<p>Response</p> <p>If the optional parameter is omitted, query the current setting:</p> <p><b>+QICFG: "transpktsize",&lt;transpktsize&gt;</b></p> <p>OK</p> <p>If the optional parameter is specified, set the packet size for transparent transmission mode:</p> <p>OK</p> <p>If there is any error:</p> <p><b>ERROR</b></p>
<p>Write Command</p> <p>Query/set the waiting time for transparent transmission mode</p> <p><b>AT+QICFG="transwaittm"[,&lt;transwaittm&gt;]</b></p>	<p>Response</p> <p>If the optional parameter is omitted, query the current setting:</p> <p><b>+QICFG: "transwaittm",&lt;transwaittm&gt;</b></p> <p>OK</p> <p>If the optional parameter is specified, set the waiting time for transparent transmission mode:</p> <p>OK</p> <p>If there is any error:</p> <p><b>ERROR</b></p>
<p>Write Command</p> <p>Query/set the maximum interval and the number of TCP retransmissions</p> <p><b>AT+QICFG="tcp/retranscfg"[,&lt;max_backoffs&gt;,&lt;max_rto&gt;]</b></p>	<p>Response</p> <p>If the optional parameter is omitted, query the current setting:</p> <p><b>+QICFG: "tcp/retranscfg",&lt;max_backoffs&gt;,&lt;max_rto&gt;</b></p> <p>OK</p> <p>If the optional parameter is specified, set the maximum interval and the number of TCP retransmissions:</p> <p>OK</p>

	<p>If there is any error: <b>ERROR</b></p>
<p>Write Command</p> <p>Enable or disable the automatic reception of the TCP connection from the client</p> <p><b>AT+QICFG="tcp/accept",&lt;accept_mode&gt;]</b></p>	<p>Response</p> <p>If the optional parameter is omitted, query the current setting: <b>+QICFG: "tcp/accept",&lt;accept_mode&gt;</b></p> <p><b>OK</b></p> <p>If the optional parameter is specified, enable or disable the automatic reception of the TCP connection from the client: <b>OK</b></p> <p>If there is any error: <b>ERROR</b></p>
<p>Write Command</p> <p>Enable or disable the passive closing of TCP connection when the server is closed</p> <p><b>AT+QICFG="passiveclosed",&lt;closed&gt;]</b></p>	<p>Response</p> <p>If the optional parameter is omitted, query the current setting: <b>+QICFG: "passiveclosed",&lt;closed&gt;</b></p> <p><b>OK</b></p> <p>If the parameter is specified, enable or disable the passive closing of TCP connection when the server is closed: <b>OK</b></p> <p>If there is any error: <b>ERROR</b></p>
<p>Write Command</p> <p>Query/set the format of the data to be sent or received (only for buffer access mode and direct push mode)</p> <p><b>AT+QICFG="dataformat",&lt;send_data_format&gt;,&lt;recv_data_format&gt;]</b></p>	<p>Response</p> <p>If the optional parameters are omitted, query the current setting: <b>+QICFG: "dataformat",&lt;send_data_format&gt;,&lt;recv_data_format&gt;</b></p> <p><b>OK</b></p> <p>If the optional parameters are specified, set the format of the data to be sent or received: <b>OK</b></p> <p>If there is any error: <b>ERROR</b></p>
<p>Write Command</p> <p>Query/set the output format of received data (only for buffer access mode and direct push mode)</p>	<p>Response</p> <p>If the optional parameter is omitted, query the current setting: <b>+QICFG: "viewmode",&lt;view_mode&gt;</b></p> <p><b>OK</b></p>

<p>AT+QICFG="viewmode"[,&lt;view_mode&gt;]</p>	<p>If the optional parameter is specified, set the output format of the received data:</p> <p><b>OK</b></p> <p>If there is any error:</p> <p><b>ERROR</b></p>
<p>Write Command</p> <p>Query/set the input data timeout</p> <p>AT+QICFG="qisend/timeout"[,&lt;time&gt;]</p>	<p>Response</p> <p>If the optional parameter is omitted, query the current setting:</p> <p><b>+QICFG: "qisend/timeout",&lt;time&gt;</b></p> <p><b>OK</b></p> <p>If the parameter is specified, set the input data timeout:</p> <p><b>OK</b></p> <p>If there is any error:</p> <p><b>ERROR</b></p>
<p>Write Command</p> <p>Set to send the specified data on the specified channel</p> <p>AT+QICFG="send/auto",&lt;connectID&gt;[,&lt;period&gt;[,&lt;msg_auto&gt;][,&lt;data_format&gt;][,&lt;try_times&gt;]]</p>	<p>Response</p> <p>If the optional parameters are omitted, query the current setting:</p> <p><b>+QICFG: "send/auto",&lt;connectID&gt;,&lt;period&gt;,&lt;msg_auto&gt;,&lt;data_format&gt;,&lt;try_times&gt;</b></p> <p><b>OK</b></p> <p>If the optional parameters are specified, set to send the specified data on the specified channel:</p> <p><b>OK</b></p> <p>If there is any error:</p> <p><b>ERROR</b></p>
<p>Write Command</p> <p>Query/set whether to display the data length in the URC format reported by the module after receiving data in buffer access mode</p> <p>AT+QICFG="recvind"[,&lt;enable&gt;]</p>	<p>Response</p> <p>If the parameter is omitted, query the current setting:</p> <p><b>+QICFG: "recvind",&lt;enable&gt;</b></p> <p><b>OK</b></p> <p>If the parameter is specified, set whether to display the data length in the URC reported by the module after receiving data in buffer access mode:</p> <p><b>OK</b></p> <p>If there is any error:</p> <p><b>ERROR</b></p>



<p>Write Command</p> <p>Set the "&gt;" format configuration when <b>AT+QISEND</b> enters the transparent transmission mode</p> <p><b>AT+QICFG="formatcfg",&lt;format&gt;]</b></p>	<p>Response</p> <p>If the optional parameter is omitted, query the current setting:</p> <p><b>+QICFG: "formatcfg",&lt;format&gt;</b></p> <p><b>OK</b></p> <p>If the optional parameter is specified, set the "&gt;" format configuration when <b>AT+QISEND</b> enters the transparent transmission mode:</p> <p><b>OK</b></p> <p>If there is any error:</p> <p><b>ERROR</b></p>
<p>Write Command</p> <p>Enable or disable displaying the return information of <b>AT+QICLOSE</b> in URC form</p> <p><b>AT+QICFG="close/mode",&lt;close_mode&gt;</b></p>	<p>Response</p> <p>If the optional parameter is omitted, query the current setting:</p> <p><b>+QICFG: "close/mode",&lt;close_mode&gt;</b></p> <p><b>OK</b></p> <p>If the optional parameter is specified, enable or disable displaying the return information of <b>AT+QICLOSE</b> in the form of URC:</p> <p><b>OK</b></p> <p>If there is any error:</p> <p><b>ERROR</b></p>
<p>Write Command</p> <p>Query/set the mode for sending UDP data</p> <p><b>AT+QICFG="udp/sendmode",&lt;send_mode&gt;]</b></p>	<p>Response</p> <p>If the optional parameter is omitted, query the current setting:</p> <p><b>+QICFG: "udp/readmode",&lt;send_mode&gt;</b></p> <p><b>OK</b></p> <p>If the optional parameter is specified, set the UDP data sending mode:</p> <p><b>OK</b></p> <p>If there is any error:</p> <p><b>ERROR</b></p>
<p>Write Command</p> <p>Query/set the mode for reading UDP data</p> <p><b>AT+QICFG="udp/readmode",&lt;read_mode&gt;]</b></p>	<p>Response</p> <p>If the optional parameter is omitted, query the current setting:</p> <p><b>+QICFG: "udp/readmode",&lt;read_mode&gt;</b></p> <p><b>OK</b></p> <p>If the optional parameter is specified, set the UDP data reading mode:</p> <p><b>OK</b></p>

	<p>If there is any error: <b>ERROR</b></p>
<p>Write Command</p> <p>Query/set whether to send TCP keep-alive information</p> <p><b>AT+QICFG="tcp/keepalive",&lt;enable&gt;,&lt;idle_time&gt;,&lt;interval_time&gt;,&lt;probe_cnt&gt;]</b></p>	<p>Response</p> <p>If the optional parameters are omitted, query the current setting: <b>+QICFG: "tcp/keepalive",&lt;enable&gt;,&lt;idle_time&gt;,&lt;interval_time&gt;,&lt;probe_cnt&gt;</b></p> <p><b>OK</b></p> <p>If the optional parameters are specified, set whether to send TCP keep-alive information: <b>OK</b></p> <p>If there is any error: <b>ERROR</b></p>
<p>Write Command</p> <p>Enable or disable the delayed acknowledgment of TCP packets</p> <p><b>AT+QICFG="tcp/delay_ack",&lt;delay_ack&gt;]</b></p>	<p>Response</p> <p>If the optional parameters are omitted, query the current setting: <b>+QICFG: "tcp/delay_ack",&lt;delay_ack&gt;</b></p> <p><b>OK</b></p> <p>If the optional parameters are specified, enable or disable the delayed acknowledgment of TCP packets: <b>OK</b></p> <p>If there is any error: <b>ERROR</b></p>
<p>Write Command</p> <p>Enable or disable the DNS cache</p> <p><b>AT+QICFG="dns/cache",&lt;DNS_cache&gt;]</b></p>	<p>Response</p> <p>If the optional parameter is omitted, query the current setting: <b>+QICFG: "dns/cache",&lt;DNS_cache&gt;</b></p> <p><b>OK</b></p> <p>If the optional parameter is specified, enable or disable the DNS cache: <b>OK</b></p> <p>If there is any error: <b>ERROR</b></p>
<p>Write Command</p> <p>Set the MSL time for TIME_WAIT after the TCP fourth-way handshake</p> <p><b>AT+QICFG="tcp/twrecycle",&lt;t</b></p>	<p>Response</p> <p>If the optional parameter is omitted, query the current setting: <b>+QICFG: "tcp/twrecycle",&lt;timewait_timeout&gt;</b></p> <p><b>OK</b></p>

<p><b>imewait_timeout&gt;]</b></p>	<p>If the optional parameter is specified, set the MSL time for TIME_WAIT after the TCP fourth-way handshake.</p> <p><b>OK</b></p> <p>If there is any error:</p> <p><b>ERROR</b></p>
<p>Write Command</p> <p>Set the retry count and retry time of DNS</p> <p><b>AT+QICFG="dns/retry",&lt;retry_ack&gt;,&lt;retry_timeout&gt;]</b></p>	<p>Response</p> <p>If the optional parameter is omitted, query the current setting:</p> <p><b>+QICFG: "dns/retry",&lt;retry_ack&gt;,&lt;retry_timeout&gt;</b></p> <p><b>OK</b></p> <p>If the optional parameter is specified, set the retry count and retry time of DNS.</p> <p><b>OK</b></p> <p>If there is any error:</p> <p><b>ERROR</b></p>
<p>Write Command</p> <p>Set the maximum length of a single transmission data</p> <p><b>AT+QICFG="send/buffersize",&lt;send_buffersize&gt;]</b></p>	<p>Response</p> <p>If the optional parameter is omitted, query the current setting:</p> <p><b>+QICFG: "send/buffersize",&lt;send_buffersize&gt;</b></p> <p><b>OK</b></p> <p>If the optional parameter is specified, set the maximum length of a single transmission data:</p> <p><b>OK</b></p> <p>If there is any error:</p> <p><b>ERROR</b></p>
<p>Write Command</p> <p>Set the TCP protocol extension option windows scale</p> <p><b>AT+QICFG="windows/scale",&lt;window_scale_enable&gt;]</b></p>	<p>Response</p> <p>If the optional parameters are omitted, query the current setting:</p> <p><b>+QICFG: "windows/scale",&lt;window_scale_enable&gt;</b></p> <p><b>OK</b></p> <p>If the parameter is specified, set the TCP protocol extension option windows scale:</p> <p><b>OK</b></p> <p>If there is any error:</p> <p><b>ERROR</b></p>

Maximum Response Time	/
Characteristic	The command takes effect immediately. The configurations are not saved.

## Parameter

<b>&lt;transpktsize&gt;</b>	Integer type. Maximum length of the data packet to be sent. Range: 1–1460. Default value: 1024. Unit: byte.
<b>&lt;transwaittm&gt;</b>	Integer type. In transparent transmission mode, if the length of data received from the port is less than the specified value of <b>&lt;transpktsize&gt;</b> , after exceeding the time specified by <b>&lt;transwaittm&gt;</b> , data will be sent. Range: 0–20. Default value: 2. Unit: 100 ms.
<b>&lt;max_backoffs&gt;</b>	Integer type. Maximum number of TCP retransmissions. Range: 3–20. Default value: 12.
<b>&lt;max_rto&gt;</b>	Integer type. Maximum interval between TCP retransmissions. Range: 5–1000. Default value: 600. Unit: 100 ms.
<b>&lt;accept_mode&gt;</b>	Integer type. Acceptance mode for the incoming TCP connection from client. 0 Manually accept the incoming TCP connection from the client 1 Automatically accept the incoming TCP connection from the client
<b>&lt;closed&gt;</b>	Integer type. Enable or disable the passive closing of TCP connection when the server is closed 0 Disable 1 Enable
<b>&lt;send_data_format&gt;</b>	Integer type. Format of the data to be sent. The suffix “0x” is not needed when the mode is set as Hex mode as the module will automatically form two bytes to one ASCII code. 0 Text mode 1 Hex mode
<b>&lt;recv_data_format&gt;</b>	Integer type. Format of the data to be received. The suffix “0x” is not needed when the mode is set as Hex mode as the module will automatically form two bytes to one ASCII code. 0 Text mode 1 Hex mode
<b>&lt;view_mode&gt;</b>	Integer type. Output format of received data. 0 data header\r\n\data. 1 data header,data.
<b>&lt;time&gt;</b>	Integer type. Timeout of <b>AT+QISEND</b> . After <b>&gt;</b> is returned, if no data is inputted within the timeout period, <b>AT+QISEND</b> will be executed. Range: 0–3600. Default value: 0. Unit: ms.
<b>&lt;connectID&gt;</b>	Integer type. The socket ID. Range: 1–11.
<b>&lt;period&gt;</b>	Integer type. The period that is configured for the specified channel to send the specified data. Range: 0, 20–86400. Unit: second. 0 indicates disabling automatic transmission; Default value: 60. Unit: seconds.

<b>&lt;msg_auto&gt;</b>	String type. Data sent in hexadecimal. The max length is 256 bytes.
<b>&lt;data_format&gt;</b>	Integer type. Set the data format of the input <b>&lt;msg_auto&gt;</b> . <u>0</u> The input data is in hexadecimal ASCII code (automatically converted to hexadecimal string) 1 The input data is in hexadecimal string 2 The input data is in string
<b>&lt;try_times&gt;</b>	Integer type. Retry times. Range: 0–10.
<b>&lt;enable&gt;</b>	Integer type. In <b>AT+QICFG="tcp/keepalive"</b> , it means whether to send TCP keepalive. In <b>AT+QICFG="recvind"</b> , it means whether to display the data length in the URC format of TCP/IP buffer access mode. <u>0</u> Disable corresponding function 1 Enable corresponding function
<b>&lt;format&gt;</b>	Integer type. Set the prompt format after <b>AT+QISEND</b> . The configuration values of S3 and S4 are ATS3 and ATS4. <u>0</u> "S3S4>" 1 "S3S4>S3S4"
<b>&lt;close_mode&gt;</b>	Integer type. Enable or disable display <b>AT+QISEND</b> execution information in URC form. 0 Disable 1 Enable
<b>&lt;send_mode&gt;</b>	Integer type. 0 Block mode <u>1</u> Stream mode
<b>&lt;read_mode&gt;</b>	Integer type. <u>0</u> Block mode 1 Stream mode
<b>&lt;idle_time&gt;</b>	Integer type. Trigger keepalive time cycle. Range: 1–120. Default value: 1. Unit: minute.
<b>&lt;interval_time&gt;</b>	Integer type. Interval between keepalive probes. If the sender does not receive a response packet, it continues to send probe packets according to this interval. Range: 25–100. Default value: 25. Unit: second.
<b>&lt;probe_cnt&gt;</b>	Integer type. Number of keepalive probes. Range: 3–10. Default: 3.
<b>&lt;delay_ack&gt;</b>	Integer type. Enable or disable the delayed acknowledgment of TCP packets. <u>0</u> Disable 1 Enable
<b>&lt;DNS_cache&gt;</b>	Integer type. Enable or disable DNS cache. 0 Disable <u>1</u> Enable
<b>&lt;timewait_timeout&gt;</b>	Integer type. The MSL time for TIME_WAIT after the TCP fourth-way handshake. Range: 2–60000. Default value: 2. Unit: ms.
<b>&lt;retry_ack&gt;</b>	Integer type. DNS retry count. Range: 1–4. Default value: 2. Unit: times.
<b>&lt;retry_timeout&gt;</b>	Integer type. DNS retry times. Range: 100–2000. Default value: 200. Unit: ms.
<b>&lt;send_buffersize&gt;</b>	Integer type. Maximum number of bytes sent at one time. Range: 1460–4096. Default value: 1460.

<b>&lt;window_scale_enable&gt;</b>	Integer type. Whether to enable the TCP protocol extension option windows scale.
0	Disable
<u>1</u>	Enable

#### NOTE

The settings of **AT+QICFG="tcp/retranscfg",<max\_backoffs>,<max\_rto>** and **+QICFG: "tcp/keepalive",<enable>,<idle\_time>,<interval\_time>,<probe\_cnt>** take effect on all TCP sockets and cannot be saved to NV.

### 2.3.2. AT+QICSGP Configure Parameters of a TCP/IP Context

This command configures **<APN>**, **<username>**, **<password>** and other parameters of a TCP/IP context.

#### AT+QICSGP Configure Parameters of a TCP/IP Context

Test Command <b>AT+QICSGP=?</b>	Response <b>+QICSGP:</b> (range of supported <b>&lt;contextID&gt;</b> s),(range of supported <b>&lt;context_type&gt;</b> s), <b>&lt;APN&gt;</b> , <b>&lt;username&gt;</b> , <b>&lt;password&gt;</b> ,(range of supported <b>&lt;authentication&gt;</b> s)  <b>OK</b>
Write Command Query the configuration of a specified context <b>AT+QICSGP=&lt;contextID&gt;</b>	Response <b>+QICSGP:</b> <b>&lt;context_type&gt;</b> , <b>&lt;APN&gt;</b> , <b>&lt;username&gt;</b> , <b>&lt;password&gt;</b> , <b>&lt;authentication&gt;</b>  <b>OK</b>
Write Command Configure the context <b>AT+QICSGP=&lt;contextID&gt;[,&lt;context_type&gt;,&lt;APN&gt;[,&lt;username&gt;,&lt;password&gt;],&lt;authentication&gt;]]</b>	Response <b>OK</b>  If there is any error: <b>ERROR</b>
Maximum Response Time	/
Characteristics	The command takes effect immediately. The configurations are saved automatically.

#### Parameter

<b>&lt;contextID&gt;</b>	Integer type. Context ID. Range: 1–15.
<b>&lt;context_type&gt;</b>	Integer type. Protocol type.

	1	IPv4
	2	IPv6
	3	IPv4v6
<APN>	String type. Access point name.	
<username>	String type. Username. Maximum length: 127 bytes.	
<password>	String type. Password. Maximum length: 127 bytes.	
<authentication>	Integer type. Authentication methods.	
	0	None
	1	PAP
	2	CHAP
	3	PAP or CHAP

## Example

```

AT+QICSGP=1                                     //Query the configuration of context 1.
+QICSGP: 1,"", "", "", "",0

OK
AT+QICSGP=1,1,"UNINET","", "",1                //Configure context 1. APN is "UNINET" for China Unicom.
OK

```

### 2.3.3. AT+QIACT Activate a PDP Context

Before activating a PDP context via **AT+QIACT**, the context should be configured with **AT+QICSGP**. After activation, the IP address can be queried via **AT+QIACT?**.

The range of **<contextID>** is 1–15 and the module supports maximum three PDP contexts activated simultaneously under LTE Cat M1. Depending on the network, it may take maximum 150 s to return **OK** or **ERROR** after executing **AT+QIACT**. Other AT commands can be executed only after the response is returned.

AT+QIACT Activate a PDP Context	
Test Command <b>AT+QIACT=?</b>	Response <b>+QIACT:</b> (range of supported <b>&lt;contextID&gt;</b> s)  <b>OK</b>
Read command <b>AT+QIACT?</b>	Response Return the list of the currently activated contexts and their IP addresses: <b>+QIACT:</b> 1,<context_state>,<context_type>[,<IP_addresses>] [.....]

	<b>+QIACT: 15,&lt;context_state&gt;,&lt;context_type&gt;[,&lt;IP_addresses&gt;]]</b>  <b>OK</b>
Write Command Activate a specified PDP context <b>AT+QIACT=&lt;contextID&gt;</b>	Response <b>OK</b>  If there is any error: <b>ERROR</b>
Maximum Response Time	150 s, determined by the network.
Characteristics	/

## Parameter

<b>&lt;contextID&gt;</b>	Integer type. Context ID. Range: 1–15.
<b>&lt;context_state&gt;</b>	Integer type. Context state. 0 Deactivated 1 Activated
<b>&lt;context_type&gt;</b>	Integer type. Protocol type. 1 IPv4 2 IPv6 3 IPv4v6
<b>&lt;IP_address&gt;</b>	String type. Local IP address after the context is activated.

### 2.3.4. AT+QIDEACT Deactivate a PDP Context

This command deactivates a specific context and closes all TCP/IP connections set up in this context. Depending on the network, it may take maximum 40 seconds to return **OK** or **ERROR** after executing **AT+QIDEACT**. Other AT commands can be executed only after the response is returned.

#### AT+QIDEACT Deactivate a PDP Context

Test Command <b>AT+QIDEACT=?</b>	Response <b>+QIDEACT: (range of supported &lt;contextID&gt;s )</b>  <b>OK</b>
Write Command <b>AT+QIDEACT=&lt;contextID&gt;</b>	Response <b>OK</b>  If there is any error: <b>ERROR</b>



Maximum Response Time	40 s, determined by the network.
Characteristics	/

## Parameter

<contextID>	Integer type. Context ID. Range: 1–15.
-------------	--

### 2.3.5. AT+QIOPEN Open a 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 response **+QIOPEN: <connectID>,<err>** indicates whether the socket service has been opened successfully.

- If <service\_type> is "TCP LISTENER", the module works as a TCP server, and there are two kinds of <accept\_mode>, automatic acceptance and manual acceptance. In automatic acceptance mode, after accepting a new TCP connection, the module automatically specifies <connectID> and report the URC **+QIURC: "incoming",<connectID>,<serverID>,<remoteIP>,<remote\_port>**. The range of <connectID> is 0–11. The type of this new incoming connection is "TCP INCOMING" and the <access\_mode> of "TCP INCOMING" is the same as that of "TCP LISTENER". In manual acceptance mode, if there is a new incoming connection to be accessed, the module will report the URC **+QIURC: "accept",<connectID>**, and <connectID> is the ID of "TCP LISTENER".
- If <service\_type> is "UDP SERVICE", UDP data can be sent to or received from the remote IP via <local\_port>.
  - Send data: execute **AT+QISEND=<connectID>,<send\_length>,<remoteIP>,<remote\_port>**.
  - Receive data in direct push mode: the module reports the URC **+QIURC: "recv",<connectID>,<currentrecvlength>,<remoteIP>,<remote\_port><CR><LF><data>**.
  - Receive data in buffer access mode: the module reports the URC **+QIURC: "recv",<connectID>**, and then the received data can be retrieved via **AT+QIRD=<connectID>**.
- It is suggested to wait for 150 s for **+QIOPEN: <connectID>,<err>** to be outputted after executing the Write Command. If the response cannot be received in 150 s , use **AT+QICLOSE** to close the socket.

### AT+QIOPEN Open a Socket Service

Test Command <b>AT+QIOPEN=?</b>	Response <b>+QIOPEN:</b> (range of supported <contextID>s),(range of supported <connectID>s),"TCP/UDP/TCP LISTENER/UDP SERVICE", "<IP_address>/<domain_name>",<remote_por
------------------------------------	--

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

## Parameter

<contextID>	Integer type. Context ID. Range: 1–15.
<connectID>	Integer type. Socket service index. Range: 0–11.
<service_type>	String type. Socket service type.
	<p>"TCP" Start a TCP connection as a client</p> <p>"UDP" Start a UDP connection as a client</p> <p>"TCP LISTENER" Start a TCP server to listen to TCP connection</p> <p>"UDP SERVICE" Start a UDP service</p>
<IP_address>	<p>String type.</p> <p>If &lt;service_type&gt; is "TCP" or "UDP", it indicates the IP address of remote server, such as 192.0.2.2.</p> <p>If &lt;service_type&gt; is "TCP LISTENER" or "UDP SERVICE", it can be omitted.</p>
<domain_name>	String type. Domain name address of the remote server.
<remote_port>	<p>Integer type. Port number of the remote server. Range: 0-65535.</p> <p>If &lt;service_type&gt; is "TCP" or "UDP" this parameter must be specified.</p> <p>If &lt;service_type&gt; is "TCP LISTENER" or "UDP SERVICE", specify this parameter as 0.</p>

<b>&lt;local_port&gt;</b>	Local port. Range: 0–65535. If <b>&lt;service_type&gt;</b> is "TCP LISTENER" or "UDP SERVICE", this parameter must be specified. If <b>&lt;service_type&gt;</b> is "TCP" or "UDP" and <b>&lt;local_port&gt;</b> is 0, the local port will be assigned automatically. Otherwise, the local port is assigned as specified.
<b>&lt;access_mode&gt;</b>	Integer type. Data access mode of the socket service. 0          Buffer access mode 1          Direct push mode 2          Transparent transmission mode
<b>&lt;err&gt;</b>	Integer type. Err code. See <b>Chapter 4</b> for details.

### 2.3.6. AT+QICLOSE Close a Socket Service

This command closes a specified socket service. Depending on the network, it takes maximum 10 s (default value, can be modified by **<timeout>**) to return **OK** or **ERROR** after executing **AT+QICLOSE**. Other AT commands can be executed only after the response is returned.

<b>AT+QICLOSE Close a Socket Service</b>	
Test Command <b>AT+QICLOSE=?</b>	Response <b>+QICLOSE:</b> (range of supported <b>&lt;connectID&gt;</b> s),(range of supported <b>&lt;timeout&gt;</b> s)  <b>OK</b>
Write Command <b>AT+QICLOSE=&lt;connectID&gt;[,&lt;timeout&gt;]</b>	Response If the socket service is closed successfully: <b>OK</b>  If the command failed to close the socket service: <b>ERROR</b>
Maximum Response Time	Determined by <b>&lt;timeout&gt;</b>
Characteristics	/

#### Parameter

<b>&lt;connectID&gt;</b>	Integer type. Socket service index. Range: 0–11.
<b>&lt;timeout&gt;</b>	Integer type. Timeout value for the response to be outputted. If the <b>FIN ACK</b> of the other peer is not received within <b>&lt;timeout&gt;</b> , the module will be forced to close the socket. Range: 0–65535. Default value: 10. Unit: second.

### 2.3.7. AT+QISTATE Query Socket Service Status

This command queries the socket service status. If the **<query\_type>** is 0, it returns the status of all existing socket services in the specified context. If the **<query\_type>** is 1, it returns the status of a specified socket service.

AT+QISTATE Query Socket Service Status	
Test Command <b>AT+QISTATE=?</b>	Response <b>OK</b>
Read/Execution Command <b>AT+QISTATE?</b> or <b>AT+QISTATE</b>	Response Return the status of all existing connections: <b>+QISTATE: &lt;connectID&gt;,&lt;service_type&gt;,&lt;IP_address&gt;,&lt;remote_port&gt;,&lt;local_port&gt;,&lt;socket_state&gt;,&lt;contextID&gt;,&lt;serverID&gt;,&lt;access_mode&gt;,&lt;AT_port&gt;</b> [...] <b>OK</b>
Write Command If <b>&lt;query_type&gt;</b> is 0, query the connection status of a specified context <b>AT+QISTATE=&lt;query_type&gt;,&lt;contextID&gt;</b>	Response Return the status of all existing connections in a specified context: <b>+QISTATE: &lt;connectID&gt;,&lt;service_type&gt;,&lt;IP_address&gt;,&lt;remote_port&gt;,&lt;local_port&gt;,&lt;socket_state&gt;,&lt;contextID&gt;,&lt;serverID&gt;,&lt;access_mode&gt;,&lt;AT_port&gt;</b> [...] <b>OK</b>
Write Command If <b>&lt;query_type&gt;</b> is 1, query the connection status of a specified socket service <b>AT+QISTATE=&lt;query_type&gt;,&lt;connectID&gt;</b>	Response <b>+QISTATE: &lt;connectID&gt;,&lt;service_type&gt;,&lt;IP_address&gt;,&lt;remote_port&gt;,&lt;local_port&gt;,&lt;socket_state&gt;,&lt;contextID&gt;,&lt;serverID&gt;,&lt;access_mode&gt;,&lt;AT_port&gt;</b> <b>OK</b>
Maximum Response Time	/
Characteristics	/

#### Parameter

<b>&lt;query_type&gt;</b>	Integer type. Query type.
0	Query connection status of all socket services in a specified context
1	Query connection status of a specified socket service

<b>&lt;contextID&gt;</b>	Integer type. Context ID. Range: 1–15.
<b>&lt;connectID&gt;</b>	Integer type. Socket service index. Range: 0–11.
<b>&lt;service_type&gt;</b>	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 incoming connection
	"TCP INCOMING" Start a TCP connection accepted by a TCP server
	"UDP SERVICE" Start a UDP service
	"SSL Client" Start an SSL connection as a client
<b>&lt;IP_address&gt;</b>	String type. IP address.
	If <b>&lt;service_type&gt;</b> ="TCP" or "SSL Client" or "UDP", it is the IP address of a remote server.
	If <b>&lt;service_type&gt;</b> ="TCP LISTENER" or "UDP SERVICE", it is the local IP address.
	If <b>&lt;service_type&gt;</b> ="TCP INCOMING", it is the IP address of a remote client.
<b>&lt;remote_port&gt;</b>	Integer type. Remote port number.
	If <b>&lt;service_type&gt;</b> ="TCP" or "SSL Client" or "UDP", it is the port of a remote server.
	If <b>&lt;service_type&gt;</b> ="TCP LISTENER" or "UDP SERVICE", the port is invalid.
	If <b>&lt;service_type&gt;</b> ="TCP INCOMING", it is the port of a remote client.
<b>&lt;local_port&gt;</b>	Integer type. Local port number.
	If <b>&lt;local_port&gt;</b> is 0, then the local port is assigned automatically.
<b>&lt;socket_state&gt;</b>	Integer type. 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
<b>&lt;serverID&gt;</b>	Integer type. It is valid only when <b>&lt;service_type&gt;</b> is "TCP INCOMING".
	<b>&lt;serverID&gt;</b> represents which server accepts this TCP incoming connection, and the value is the same as <b>&lt;connectID&gt;</b> when the <b>&lt;service_type&gt;</b> is "TCP LISTENER".
<b>&lt;access_mode&gt;</b>	Integer type. Data access mode.
	0 Buffer access mode
	1 Direct push mode
	2 Transparent transmission mode
<b>&lt;AT_port&gt;</b>	String type. COM port of socket service.
	"usbmodem" USB modem port
	"usbat" USB AT port
	"uart1" UART port1
	"cmux1" MUX port 1
	"cmux2" MUX port 2
	"cmux3" MUX port 3
	"cmux4" MUX port 4

### 2.3.8. AT+QISEND Send Data

In buffer access mode (<access\_mode>=0) or direct push mode (<access\_mode>=1), then the data can be sent via **AT+QISEND**. If the data is sent to the module successfully, **SEND OK** is returned. Otherwise, **SEND FAIL** or **ERROR** is returned.

- **SEND FAIL** indicates the sending buffer is full. In this case, the data can be resent.
- **ERROR** indicates an error in the data sending process. In this case, wait for some time before you resend the data. Maximum data length: 1460 bytes.
- **SEND OK** means that the data have been sent to the peer, but it does not mean they have reached the peer successfully. You can query whether the data have reached the peer by **AT+QISEND=<connectID>,0**.

#### AT+QISEND Send Data

Test Command <b>AT+QISEND=?</b>	Response <b>+QISEND:</b> (range of supported <connectID>s),(range of supported <send_length>s)  <b>OK</b>
Write Command Send variable-length data when <service_type> is "TCP", "UDP" or "TCP INCOMING" <b>AT+QISEND=&lt;connectID&gt;</b>	Response > After the response >, input the data to be sent. Tap <b>Ctrl+Z</b> to send the data, and tap <b>Esc</b> to cancel the sending operation  If the connection has been established and the data is sent successfully: <b>SEND OK</b>  If the connection has been established but the sending buffer is full: <b>SEND FAIL</b>  If the connection is not established, abnormally closed, or any parameter is incorrect: <b>ERROR</b>
Write Command Send fixed-length data when <service_type> is "TCP", "UDP" or "TCP INCOMING" <b>AT+QISEND=&lt;connectID&gt;,&lt;send_length&gt;</b>	Response > After the response >, input the data until the data length equals <send_length>.  If the connection has been established and the data is sent successfully: <b>SEND OK</b>

	<p>If the connection has been established but the sending buffer is full: <b>SEND FAIL</b></p> <p>If the connection has not been established, abnormally closed, or any parameter is incorrect: <b>ERROR</b></p>
<p>Write Command</p> <p>If <b>&lt;service_type&gt;</b> is "UDP SERVICE"</p> <p><b>AT+QISEND=&lt;connectID&gt;,&lt;send_length&gt;,&lt;remoteIP&gt;,&lt;remote_port&gt;</b></p>	<p>Response</p> <p>Sends fixed length data to a specified remote IP address and remote port. <b>&lt;service_type&gt;</b> must be "UDP SERVICE".</p> <p><b>&gt;</b></p> <p>After response <b>&gt;</b>, input the data until the data length equals <b>&lt;send_length&gt;</b></p> <p>If the connection has been established and the data is sent successfully: <b>SEND OK</b></p> <p>If the connection has been established but the sending buffer is full: <b>SEND FAIL</b></p> <p>If the connection has not been established, abnormally closed, or any parameter is incorrect: <b>ERROR</b></p>
<p>Write Command</p> <p>When <b>&lt;send_length&gt;</b> is 0, query the sent data</p> <p><b>AT+QISEND=&lt;connectID&gt;,0</b></p>	<p>Response</p> <p>If the specified connection exists: <b>+QISEND: &lt;total_send_length&gt;,&lt;ackedbytes&gt;,&lt;unacked bytes&gt;</b></p> <p><b>OK</b></p> <p>If the specified connection does not exist, or if there is any error: <b>ERROR</b></p>
Maximum Response Time	/
Characteristics	/

## Parameter

<b>&lt;connectID&gt;</b>	Integer type. Socket service index. Range: 0–11.
<b>&lt;send_length&gt;</b>	Integer type. Length of data to be sent. Range: 0–1460. Unit: byte.
<b>&lt;remoteIP&gt;</b>	String type. Remote IP address (must be dot format). It is valid only when <b>&lt;service_type&gt;</b> is "UDP SERVICE".
<b>&lt;remote_port&gt;</b>	Integer type. Remote port. It is only valid when <b>&lt;service_type&gt;</b> is "UDP SERVICE".
<b>&lt;total_send_length&gt;</b>	Integer type. Total length of sent data. Unit: byte.
<b>&lt;ackedbytes&gt;</b>	Integer type. Total length of acknowledged data. Unit: byte.
<b>&lt;unackedbytes&gt;</b>	Integer type. Total length of unacknowledged data. Unit: byte.

### 2.3.9. AT+QIRD Retrieve Received TCP/IP Data

In buffer access mode, after receiving data, the module buffers it and reports URC **+QIURC: "recv",<connectID>**, then the data can be retrieved with **AT+QIRD**.

Please note that if the buffer is not empty, and the module receives data again, it will not report a new URC until all received data has been retrieved via **AT+QIRD** from the buffer.

AT+QIRD Retrieve Received TCP/IP Data	
Test Command <b>AT+QIRD=?</b>	Response <b>+QIRD: (range of supported &lt;connectID&gt;s),(range of supported &lt;read_length&gt;s)</b>  <b>OK</b>
Write Command When <b>&lt;service_type&gt;</b> is "TCP", "UDP", "TCP INCOMING" <b>AT+QIRD=&lt;connectID&gt;[,&lt;read_length&gt;]</b>	Response If the specified connection has received the data: <b>+QIRD: &lt;read_actual_length&gt;&lt;CR&gt;&lt;LF&gt;&lt;data&gt;</b>  <b>OK</b>  If there is no data: <b>+QIRD: 0</b>  <b>OK</b>  If the connection does not exist: <b>ERROR</b>
Write Command When <b>&lt;service_type&gt;</b> is "UDP SERVICE" <b>AT+QIRD=&lt;connectID&gt;[,&lt;read_length&gt;]</b>	Response If data exists: <b>+QIRD: &lt;read_actual_length&gt;,&lt;remoteIP&gt;,&lt;remote_port&gt;&lt;CR&gt;&lt;LF&gt;&lt;data&gt;</b>



	<p><b>OK</b></p> <p>If there is no data: <b>+QIRD: 0</b></p> <p><b>OK</b></p> <p>If the connection does not exist: <b>ERROR</b></p>
<p>Write Command</p> <p>When <b>&lt;read_length&gt;</b> is 0, query the retrieved data length</p> <p><b>AT+QIRD=&lt;connectID&gt;,0</b></p>	<p>Response</p> <p>If the specified connection exists: <b>+QIRD: &lt;total_receive_length&gt;,&lt;have_read_length&gt;,&lt;unread_length&gt;</b></p> <p><b>OK</b></p> <p>If there is any error: <b>ERROR</b></p>
Maximum Response Time	/
Characteristics	/

## Parameter

<b>&lt;connectID&gt;</b>	Integer type. Socket service index. Range: 0–11.
<b>&lt;read_length&gt;</b>	Integer type. Maximum length of data to be retrieved. Range: 0–1500. Unit: byte.
<b>&lt;read_actual_length&gt;</b>	Integer type. Length of actually retrieved data. Unit: byte.
<b>&lt;remoteIP&gt;</b>	String type. Remote IP address. It is valid only when <b>&lt;service_type&gt;</b> is "UDP SERVICE".
<b>&lt;remote_port&gt;</b>	Integer type. Remote port number. It is valid only when <b>&lt;service_type&gt;</b> is "UDP SERVICE".
<b>&lt;data&gt;</b>	Integer type. Retrieved data.
<b>&lt;total_receive_length&gt;</b>	Integer type. Total length of received data. Unit: byte.
<b>&lt;have_read_length&gt;</b>	Integer type. Length of data that has been retrieved. Unit: byte.
<b>&lt;unread_length&gt;</b>	Integer type. Length of data that has not been retrieved. Unit: byte.

### 2.3.10. AT+QISENDEX Send Hex String Data

This command sends hex string data and cannot be applied to "UDP SERVICE" and "TCP LISTENER" sockets.

### AT+QISENDEX Send Hex String Data

Test Command <b>AT+QISENDEX=?</b>	Response <b>+QISENDEX:</b> (range of supported <connectID>s),<hex_string>  <b>OK</b>
Write Command <b>AT+QISENDEX=&lt;connectID&gt;,&lt;hex_string&gt;</b>	Response If the hex string is sent successfully: <b>SEND OK</b>  If the sending buffer is full: <b>SEND FAIL</b>  If the connection does not exist: <b>ERROR</b>
Maximum Response Time	/
Characteristics	/

#### Parameter

<connectID>	Integer type. Socket service index. Range: 0–11.
<hex_string>	String type. Hex string data. Max length: 512 bytes.

### 2.3.11. AT+QIACCEPT Manually Accept New TCP Incoming Connection

If <accept\_mode> is set to manually accept a new TCP incoming connection, after URC **+QIURC: "accept",<connectID>** is returned, the new TCP incoming connection can be accepted via **AT+QIACCEPT**.

### AT+QIACCEPT Manually Accept New TCP Incoming Connection

Test Command <b>AT+QIACCEPT=?</b>	Response <b>+QIACCEPT:</b> (range of supported <TCP_listener_connectID>s),(list of supported <accept>s),(range of supported <incoming_connectID>s)  <b>OK</b>
Write Command <b>AT+QIACCEPT=&lt;TCP_listener_connectID&gt;,&lt;accept&gt;[,&lt;incoming_connectID&gt;]</b>	Response If <accept> is 0: <b>OK</b>

tID>]	<p>If &lt;accept&gt; is 1: <b>OK</b></p> <p><b>+QIACCEPT:</b> &lt;incoming_connectID&gt;,&lt;remote_addr&gt;,&lt;remote_port&gt;</p> <p>If there is any error: <b>ERROR</b></p>
Maximum Response Time	/
Characteristics	/

## Parameter

<TCP_listener_connectID>	Integer type. ID of "TCP LISTENER". Range: 0–11.
<accept>	Integer type. Accept or refuse the incoming connection. 0 Refuse 1 Accept
<incoming_connectID>	Integer type. ID of the incoming connection to be accepted.
<remote_addr>	String type. Remote client address.
<remote_port>	Integer type. Remote client port. Range: 1–65535.

## 2.3.12. AT+QISWTMD Switch Data Access Mode

This command switches the data access mode among buffer access mode, direct push mode and transparent transmission mode. When starting a socket service, the data access mode can be specified via the <access\_mode> parameter of **AT+QIOPEN**. After opening a socket, the data access mode can be changed via **AT+QISWTMD**.

AT+QISWTMD Switch Data Access Mode	
Test Command <b>AT+QISWTMD=?</b>	<p>Response</p> <p><b>+QISWTMD:</b> (range of supported &lt;connectID&gt;s),(range of supported &lt;access_mode&gt;s)</p> <p><b>OK</b></p>
Write Command <b>AT+QISWTMD=&lt;connectID&gt;,&lt;access_mode&gt;</b>	<p>Response</p> <p>If data access mode is switched successfully and &lt;access_mode&gt; is 0 or 1: <b>OK</b></p>

	<p>If data access mode is switched successfully and <b>&lt;access_mode&gt;</b> is 2, the module will enter the intended data mode:</p> <p><b>CONNECT</b></p> <p>If there is any error:</p> <p><b>ERROR</b></p>
Maximum Response Time	/
Characteristics	/

## Parameter

<b>&lt;connectID&gt;</b>	Integer type. Socket service index. Range: 0–11.
<b>&lt;access_mode&gt;</b>	Integer type. Data access modes of the connection.
0	Buffer access mode
1	Direct push mode
2	Transparent transmission mode

### 2.3.13. AT+QPING Ping a Remote Host

The command tests the reachability of a host on an Internet protocol network. Before using the ping utility, the host should activate the context of the corresponding **<contextID>** via **AT+QIACT**. The command returns the result within **<timeout>** and the default value of **<timeout>** is 4 seconds.

AT+QPING Ping a Remote Host	
<p>Test Command</p> <p><b>AT+QPING=?</b></p>	<p>Response</p> <p><b>+QPING:</b> (range of supported <b>&lt;connectID&gt;</b>s),<b>&lt;host&gt;</b>,(range of supported <b>&lt;timeout&gt;</b>s),(range of supported <b>&lt;pingnum&gt;</b>s)</p> <p><b>OK</b></p>
<p>Write Command</p> <p><b>AT+QPING=&lt;contextID&gt;,&lt;host&gt;[,&lt;timeout&gt;[,&lt;pingnum&gt;]]</b></p>	<p>Response</p> <p>If a remote server is pinged successfully:</p> <p><b>OK</b></p> <p><b>[+QPING: &lt;ping_result&gt;[,&lt;IP_address&gt;,&lt;bytes&gt;,&lt;time&gt;,&lt;ttl&gt;]&lt;CR&gt;&lt;LF&gt;...]</b></p> <p><b>[...]</b></p> <p><b>+QPING: &lt;finresult&gt;[,&lt;sent&gt;,&lt;rcvd&gt;,&lt;lost&gt;,&lt;min&gt;,&lt;ma</b></p>

	<b>x&gt;,&lt;avg&gt;]</b>  If there is any error: <b>ERROR</b>
Maximum Response Time	/
Characteristics	/

## Parameter

<b>&lt;contextID&gt;</b>	Integer type. Context ID. Range: 1–15.
<b>&lt;host&gt;</b>	String type. Host address. It is a domain name or a dotted decimal IP address.
<b>&lt;timeout&gt;</b>	Integer type. Maximum time to wait for the response of each ping request. Range: 1–255. Default value: 4. Unit: second.
<b>&lt;pingnum&gt;</b>	Integer type. Maximum number of times for sending ping request. Range: 1–10. Default value: 4.
<b>&lt;ping_result&gt;</b>	Integer type. Result of each ping request. 0 Received the ping response from the host. In this case, it is followed by <b>&lt;IP_address&gt;,&lt;bytes&gt;,&lt;time&gt;,&lt;tll&gt;</b> . Other values Err code. See <b>Chapter 4</b> for details.
<b>&lt;IP_address&gt;</b>	String type. IP address of the remote host formatted as a dotted decimal IP.
<b>&lt;bytes&gt;</b>	Integer type. Length of each sent ping request. Unit: byte.
<b>&lt;time&gt;</b>	Integer type. Waiting time for the response to ping request. Unit: ms.
<b>&lt;tll&gt;</b>	Integer type. Time to live value of the response packet for the ping request.
<b>&lt;finresult&gt;</b>	Integer type. Final result of the command. 0 Pinged successfully. It succeeded in activating the context and finding the host. In this case, it is followed by <b>&lt;sent&gt;,&lt;rcvd&gt;,&lt;lost&gt;,&lt;min&gt;,&lt;max&gt;,&lt;avg&gt;</b> . Other values Err code. See <b>Chapter 4</b> for details.
<b>&lt;sent&gt;</b>	Integer type. Total number of sent ping requests.
<b>&lt;rcvd&gt;</b>	Integer type. Total number of the ping requests that received responses.
<b>&lt;lost&gt;</b>	Integer type. Total number of timed out ping requests.
<b>&lt;min&gt;</b>	Integer type. Minimum response time. Unit: ms.
<b>&lt;max&gt;</b>	Integer type. Maximum response time. Unit: ms.
<b>&lt;avg&gt;</b>	Integer type. Average response time. Unit: ms.

### 2.3.14. AT+QNTTP Synchronize Local Time with NTP Server

This command synchronizes the local time with Universal Time Coordinated (UTC) through the NTP server. Before time synchronization, the host should activate the context corresponding to **<contextID>** via **AT+QIACT**. Depending on the network, it will take maximum 125 seconds to return the result.

AT+QNTTP Synchronize Local Time with NTP Server	
Test command <b>AT+QNTTP=?</b>	Response <b>+QNTTP:</b> (range of supported <connectID>s),<server>,(range of supported <port>s),(list of supported <autosettime>s)  <b>OK</b>
Read command <b>AT+QNTTP?</b>	Response If in the process of local time synchronization: <b>+QNTTP: &lt;server&gt;,&lt;port&gt;</b>  <b>OK</b>
Write command <b>AT+QNTTP=&lt;contextID&gt;,&lt;server&gt;[,&lt;port&gt;[,&lt;autosettime&gt;]]</b>	Response If the local time is synchronized with NTP server successfully: <b>OK</b>  <b>+QNTTP: &lt;err&gt;,&lt;time&gt;</b>  If there is any error: <b>ERROR</b>
Maximum Response Time	125 seconds, determined by the network.
Characteristics	/

## Parameter

<contextID>	Integer type. Context ID. Range: 1–15.
<server>	String type. NTP server address.
<port>	Integer type. NTP server port. Range:1–65535.
<autosettime>	Integer type. Whether to automatically set synchronized time as local time. 0 Do not set 1 Set
<err>	Integer type. Err code. See <b>Chapter 4</b> for details.
<time>	String type. Time synchronized from NTP server. Format: “YYYY/MM/DD,hh:mm:ss±zz”. Range of zz: -48 to +56.

### 2.3.15. AT+QIDNSCFG Configure Address of DNS Server

This command configures the address of DNS server. Before setting the DNS address, the host must activate the context of corresponding <contextID> via **AT+QIACT**.

AT+QIDNSCFG Configure Address of DNS Server
---

Test command <b>AT+QIDNSCFG=?</b>	Response <b>+QIDNSCFG:</b> (range of supported <connectID>s),<pridnsaddr>,<secdnsaddr>  <b>OK</b>
Write Command <b>AT+QIDNSCFG=&lt;contextID&gt;[,&lt;pridnsaddr&gt;[,&lt;secdnsaddr&gt;]]</b>	Response If <pridnsaddr> and <secdnsaddr> are omitted, query the current DNS server addresses of a specified PDP context: <b>+QIDNSCFG: &lt;contextID&gt;,&lt;pridnsaddr&gt;,&lt;secdnsaddr&gt;</b>  <b>OK</b>  If <pridnsaddr> and <secdnsaddr> are specified, configure the primary and secondary DNS server addresses of a specified PDP context: <b>OK</b>  If there is any error: <b>ERROR</b>
Maximum Response Time	/
Characteristics	/

## Parameter

<contextID>	Integer type. PDP context ID. Range: 1–15.
<pridnsaddr>	String type. Primary DNS server address.
<secdnsaddr>	String type. Secondary DNS server address.

### 2.3.16. AT+QIDNSGIP Get IP Address by Domain Name

This command gets an IP address by domain name. Before querying the DNS, the host should activate the context of corresponding <contextID> via **AT+QIACT**. Depending on the network, it will take maximum 60 seconds to return the result.

<b>AT+QIDNSGIP Get IP Address by Domain Name</b>	
Test Command <b>AT+QIDNSGIP=?</b>	Response <b>+QIDNSGIP:</b> (range of supported <connectID>s),<hostname>  <b>OK</b>
Write Command <b>AT+QIDNSGIP=&lt;contextID&gt;,&lt;ho</b>	Response <b>OK</b>

stname>	<p>If there is any error: <b>ERROR</b></p> <p>The result will be returned as URC.  <b>+QIURC: "dnsgip",&lt;err&gt;,&lt;IP_count&gt;,&lt;DNS_ttl&gt;</b>  [.....  <b>+QIURC: "dnsgip",&lt;hostIPaddr&gt;</b> </p>
Maximum Response Time	60 seconds, determined by the network.
Characteristics	/

## Parameter

<contextID>	Integer type. PDP context ID. Range: 1–15.
<hostname>	String type. Domain name.
<err>	Integer type. Err code. See <b>Chapter 4</b> for details.
<IP_count>	Integer type. Number of IP addresses corresponding to <hostname>.
<DNS_ttl>	Integer type. Time to live of the DNS. Unit: second.
<hostIPaddr>	String type. IP address of <hostname>.

### 2.3.17. AT+QISDE Control Whether to Echo Data for AT+QISEND

This command controls whether to echo the data for **AT+QISEND**.

AT+QISDE Control Whether to Echo Data for AT+QISEND	
Test Command <b>AT+QISDE=?</b>	<p>Response  <b>+QISDE: (list of supported &lt;echo&gt;s)</b></p> <p><b>OK</b></p>
Read Command <b>AT+QISDE?</b>	<p>Response  <b>+QISDE: &lt;echo&gt;</b></p> <p><b>OK</b></p>
Write Command <b>AT+QISDE=&lt;echo&gt;</b>	<p>Response  <b>OK</b></p> <p>If there is any error:  <b>ERROR</b></p>
Maximum Response Time	/



Characteristics	/
-----------------	---

## Parameter

<b>&lt;echo&gt;</b>	Numeric type. Whether to echo the data for <b>AT+QISEND</b> .
0	Do not echo the data
<u>1</u>	Echo the data

### 2.3.18. AT+QIGETERROR Query Last Error Code

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

AT+QIGETERROR Query Last Error Code	
Test command <b>AT+QIGETERROR=?</b>	Response <b>OK</b>
Execution Command <b>AT+QIGETERROR</b>	Response <b>+QIGETERROR: &lt;err&gt;,&lt;err_code_description&gt;</b>  <b>OK</b>
Maximum Response Time	/
Characteristics	/

## Parameter

<b>&lt;err&gt;</b>	Integer type. Error codes of the operation. See <b>Chapter 4</b> for details.
<b>&lt;err_code_description&gt;</b>	String type. String parameter indicates the details of error information. See <b>Chapter 4</b> for details.

## 2.4. Description of URCs

The URC of TCP/IP AT commands will be reported to the host in the format that begins with **+QIURC:**. It contains the reports about incoming data, closed connection and incoming connection and so on. Actually, there is **<CR><LF>** both before and after URC, but **<CR><LF>** is intentionally not presented.

### 2.4.1. +QIURC: "closed" Indicate Closed Connection

When TCP socket service is closed by remote peer or due to network error, the URC will be outputted, and the status of socket service will be "closing" (<socket\_state>=4). **AT+QICLOSE=<connectID>** can be used to change the <socket\_state> to initial.

#### +QIURC: "closed" Indicate Closed Connection

<b>+QIURC: "closed",&lt;connectID&gt;</b>	Socket service connection is closed.
---	--------------------------------------

#### Parameter

<connectID>	Integer type. Socket service index. Range: 0–11.
-------------	--

### 2.4.2. +QIURC: "recv" Indicate Incoming Data

In buffer access mode or direct push mode, after receiving data, the module reports a URC to the host.

In buffer access mode, after receiving data, the module reports **+QIURC: "recv",<connectID>** to notify the host. Then host can retrieve data via **AT+QIRD**. If the buffer is not empty, and the module receives data again, it does not report a new URC until all the received data have been retrieved with **AT+QIRD** from the buffer.

In direct push mode, the received data are outputted to COM port directly.

#### +QIURC: "recv" Indicate Incoming Data

<b>+QIURC: "recv",&lt;connectID&gt;</b>	Indicates incoming data in buffer access mode. The host can retrieve data via <b>AT+QIRD</b> .
<b>+QIURC: "recv",&lt;connectID&gt;,&lt;currentrecvlength&gt;&lt;CR&gt;&lt;LF&gt;&lt;data&gt;</b>	Indicates the incoming data in direct push mode when <service_type> is "TCP", "UDP", "UPD INCOMING" or "TCP INCOMING".
<b>+QIURC: "recv",&lt;connectID&gt;,&lt;currentrecvlength&gt;,&lt;remoteIP&gt;,&lt;remote_port&gt;&lt;CR&gt;&lt;LF&gt;&lt;data&gt;</b>	Indicates data incoming in direct push mode when <service_type> is "UDP SERVICE".

#### Parameter

<connectID>	Integer type. Socket service index. Range: 0–11.
<currentrecvlength>	Integer type. Length of actually received data. Range: 0-1500. Unit:byte.
<remoteIP>	String type. Remote IP address.
<remote_port>	Integer type. Remote port number.

<data>	Received data. Unit: byte.
--------	----------------------------

### 2.4.3. +QIURC: "incoming full" Indicate Incoming Connection Reached Limit

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" Indicate Incoming Connection Reached Limit

+QIURC: "incoming full"	Indicates the number of incoming connections has reached the limit.
-------------------------	---

### 2.4.4. +QIURC: "incoming" Indicate Incoming Connection in Automatic

#### Acceptance Mode

If <service\_type> is "TCP LISTENER", when a remote client connects to this server, and if <accept\_mode> is automatic acceptance, the module will automatically assign a free <connectID> for the new connection. The range of <connectID> is 0 to 11. In such a case, the module will report the URC. <service\_type> of the new connection will be "TCP INCOMING", and <access\_mode> will be assigned as <access mode> of "TCP LISENER".

#### +QIURC: "incoming" Indicate Incoming Connection in Automatic Acceptance Mode

+QIURC: "incoming",<connectID>,<serverID>,<remoteIP>,<remote_port>	When the new incoming connection is accepted by <serverID>, the allocated <connectID>, <remoteIP> and <remote_port> are informed by this URC.
--	---

#### Parameter

<connectID>	Integer type. Index of the socket service assigned to the incoming connection, which is automatically specified by the module. Range: 0–11.
<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> when the <service_type> is "TCP LISTENER".
<remoteIP>	String type. Remote IP address of the incoming <connectID>.
<remote_port>	Integer type. Remote port number of the incoming <connectID>.

## 2.4.5. +QIURC: "accept" Indicate Incoming Connection in Manual Acceptance Mode

### Mode

If the `<service_type>` is "TCP LISTENER", when a remote client connects to this server, and if the `<accept_mode>` is manual acceptance, the module will report the URC.

### +QIURC: "accept" Indicate Incoming Connection in Manual Acceptance Mode

+QIURC: "accept",<connectID>	Indicates an incoming connection and you can execute <b>AT+QIACCEPT</b> to accept or refuse the incoming connection.
------------------------------	--

### Parameter

<code>&lt;connectID&gt;</code>	Integer type. ID of "TCP LISTENER". Range: 0–11.
--------------------------------	--

## 2.4.6. +QIURC: "pdpdeact" Indicate PDP Deactivation

PDP context may be deactivated by the network. The module reports the URC to the host about PDP deactivation. In this case, the host must execute **AT+QIDEACT** to deactivate the context and reset all connections.

### +QIURC: "pdpdeact" Indicate PDP Deactivation

+QIURC: "pdpdeact",<contextID>	<code>&lt;contextID&gt;</code> context is deactivated.
--------------------------------	--

### Parameter

<code>&lt;contextID&gt;</code>	Integer type. Context ID. Range: 1–15.
--------------------------------	--

# 3 Examples

## 3.1. Configure and Activate a Context

### 3.1.1. Configure a Context

```
AT+QICSGP=1,1,"UNINET","", "",1 //Configure context 1. APN is "UNINET" for China Unicom.
OK
```

### 3.1.2. Activate a Context

```
AT+QIACT=1 //Activate context 1. Depending on the network, the maximum
            response time is 150 s.
OK //Activated the context successfully.
AT+QIACT? //Query the context state, protocol type and IP address of
            context 1
+QIACT: 1,1,1,"10.7.157.1"
OK
```

### 3.1.3. Deactivate a Context

```
AT+QIDEACT=1 //Deactivate context 1.
OK //Deactivated the context successfully. Depending on the
    network, the maximum response time is 40 s.
```

## 3.2. TCP Client Works in Buffer Access Mode under IPv4

### 3.2.1. Set up a TCP Client Connection and Enter Buffer Access Mode

```
AT+QIOPEN=1,0,"TCP","192.0.2.2",8009,0,0 //Context is 1 and <connectID> is 0. Before
                                             using AT+QIOPEN, the host should activate
```

the context with **AT+QIACT** first.

OK

**+QIOPEN: 0,0**

//TCP client is connected successfully. It is suggested to wait for 150 s for the URC **+QIOPEN: <connectID>,<err>**. If the URC cannot be received in 150 s, the host could use **AT+QICLOSE** to close the socket.

**AT+QISTATE=1,0**

//Query connection status of socket service 1.

**+QISTATE: 0,"TCP","192.0.2.2",8009,65514,2,1,0,0,"usbmodem"**

OK

### 3.2.2. Send Data in Buffer Access Mode

**AT+QISEND=0**

//Send variable-length data.

**> test1<ctrl+Z>**

**SEND OK**

//**SEND OK** does not mean the data has been sent to the server successfully. The host can query whether the data has reached the server via **AT+QISEND=0,0**.

**AT+QISEND=0,4**

//Send fixed-length data and the data length is 4 bytes.

**> test**

**SEND OK**

**AT+QISEND=0,0**

//Query the length of sent data, acknowledged data and unacknowledged data.

**+QISEND: 9,9,0**

OK

**AT+QISENDEX=0,"3132333435"**

//Send Hex string data.

**SEND OK**

**AT+QISEND=0,0**

//Query the length of sent data, acknowledged data and unacknowledged data.

**+QISEND: 14,14,0**

OK

### 3.2.3. Retrieve Data from Remote Server in Buffer Access Mode

**+QIURC: "recv",0**

//The received data when **<connectID>=0**.

**AT+QIRD=0,1500**

//Retrieve data, and the maximum length of data to be retrieved is 1500

```

bytes.
+QIRD: 5 //The length of actually retrieved data is 5 bytes.
test1

OK
AT+QICFG="recvind",1
OK
+QIURC: "recv",0,5 //Received data is 5 bytes when <connectID>=0.
AT+QIRD=0,1500 //Retrieve data, and the length is 1500 bytes.
+QIRD: 5 //The length of actual received data is 5 bytes.
test1

OK
AT+QIRD=0,1500
+QIRD: 0 //No data in buffer.

OK
AT+QIRD=0,0 //Query the total length of received data, including read and unread data.
+QIRD: 10,10,0

OK

```

### 3.2.4. Close a Connection

```

AT+QICLOSE=0 //Close a connection whose <connectID> is 0. Depending on the
              network, the maximum response time is 10 s by default.

OK

```

## 3.3. TCP Client Works in Transparent Transmission Mode

### 3.3.1. Set up a TCP Client Connection and Enter Transparent Transmission Mode

```

AT+QIOPEN=1,0,"TCP","192.0.2.2",8009,0,2 //Context is 1 and <connectID> is 0. Before
                                           using AT+QIOPEN, the host should activate the
                                           context with AT+QIACT.

CONNECT //TCP client is connected successfully. It is
         suggested to wait for 150 s for the URC
         CONNECT. If the URC cannot be received in
         150 s, the host could use AT+QICLOSE to close
         the socket.

```

### 3.3.2. Send Data in Transparent Transmission Mode

<All data received from COM port will be sent directly to the Internet>

### 3.3.3. Retrieve Data from Remote Server in Transparent Transmission Mode

Test 1

//All data received from the Internet are outputted via COM port directly.

### 3.3.4. Close a TCP Client

**AT+QICLOSE=0**

//After using **+++** to exit the transparent transmission mode, the host could use **AT+QICLOSE** to close the TCP link. Depending on the network, the maximum response time is 10 s by default.

OK

## 3.4. TCP Client Works in Direct Push Mode

### 3.4.1. Set up a TCP Client Connection and Enter Direct Push Mode

**AT+QIOPEN=1,0,"TCP","192.0.2.2",8009,0,1**

//Context is 1 and **<connectID>** is 0. Before using **AT+QIOPEN**, the host should activate the context via **AT+QIACT**.

OK

**+QIOPEN: 0,0**

//TCP client is connected successfully. It is suggested to wait for 150 s for the URC **+QIOPEN: <connectID>,<err>**. If the URC cannot be received in 150 s, the host could use **AT+QICLOSE** to close the socket.

**AT+QISTATE=1,0**

//Query if the connection state socket service 0.

**+QISTATE: 0,"TCP","192.0.2.2",8009,65344,2,1,0,1,"usbmodem"**

OK



### 3.4.2. Send Data in Direct Push Mode

<b>AT+QISEND=0</b>	//Send variable-length data.
> test1<ctrl+Z>	
SEND OK	// <b>SEND OK</b> does not mean the data has been sent to the server successfully. Host can query whether the data has reached the server via <b>AT+QISEND=0,0</b> .
<b>AT+QISEND=0,5</b>	//Send fixed-length data and the data length is 5 bytes.
> test2	
SEND OK	
<b>AT+QISEND=0,0</b>	//Query the length of sent data, acknowledged data and unacknowledged data.
+QISEND: 10,10,0	//A total of 10 bytes of data have been sent, and all 10 bytes have been acknowledged.
OK	

### 3.4.3. Receive Data from Remote Server in Direct Push Mode

+QIURC: "recv",0,4	//Retrieve data from remote server.
test	

### 3.4.4. Close a TCP Client

<b>AT+QICLOSE=0</b>	//Close the connection whose <b>&lt;connectID&gt;</b> is 0. Depending on the network, the maximum response time is 10 s by default.
OK	

## 3.5. TCP Server Works in Buffer Access Mode under IPv4

### 3.5.1. Start a TCP Server

<b>AT+QIOPEN=1,1,"TCP LISTENER","127.0.0.1",0,2020,0</b>	//Context is 1 and <b>&lt;connectID&gt;</b> is 1. Before using <b>AT+QIOPEN</b> , the host should activate the context with <b>AT+QIACT</b> .
OK	
+QIOPEN: 1,0	//TCP server is opened successfully.

```

AT+QISTATE=0,1 //Query the connection the connection status
of context 1.
+QISTATE: 1,"TCP LISTENER","10.7.157.1",0,2020,3,1,1,0,"usbmodem"
OK

```

### 3.5.2. Accept TCP Incoming Connection

```

+QIURC: "incoming",11,1,"172.31.242.222",54091 //A new TCP connection is accepted.
<service_type> is "TCP incoming", and
<connectID> is 11.

```

### 3.5.3. Retrieve Data from Incoming Connection

```

+QIURC: "recv",11 //Received data from remote incoming connection.
AT+QIRD=11,1500 //Retrieve data received from incoming connection.
+QIRD: 4 //Length of actually retrieved data is 4 bytes.
test

OK
AT+QIRD=11,1500
+QIRD: 0 //No data in buffer.

OK
AT+QIRD=11,0 //Query the total length of received data, including
read and unread data.
+QIRD: 4,4,0

OK

```

### 3.5.4. Close a TCP Server

```

AT+QICLOSE=11 //Close the incoming connection. Depending on the
network, the maximum response time is 10 s by
default.

OK
AT+QICLOSE=1 //Close the listening TCP server.

OK

```

## 3.6. UDP Service under IPv4

### 3.6.1. Start a UDP Service

```

AT+QIOPEN=1,2,"UDP SERVICE","127.0.0.1",0,3030,0 //Start a UDP service whose <connectID> is 2
                                                    and <contextID> is 1. Before using
                                                    AT+QIOPEN, the host should activate the
                                                    context with AT+QIACT.

OK

+QIOPEN: 2,0                                     //UDP service is opened successfully.
AT+QISTATE=0,1                                    //Query the connection status of context 1.
+QISTATE: 2,"UDP SERVICE","10.7.157.1",0,3030,2,1,2,0,"usbmodem"

OK

```

### 3.6.2. Send UDP Data to Remote Client

```

AT+QISEND=2,10,"10.7.89.10",6969                //Send 10 bytes data to remote client whose IP
                                                    is 10.7.89.10 and the remote port is 6969.

>1234567890
SEND OK

```

### 3.6.3. Retrieve Data from Remote Client

```

+QIURC: "recv",2                                //Received data from remote client.
AT+QIRD=2                                         //Retrieve UDP data. One whole UDP packet will
                                                    be outputted. There is no need to specify the
                                                    read length.

+QIRD: 4,"10.7.76.34",7687                       //Retrieved data length is 4. The remote IP
                                                    address is 10.7.76.34 and remote port is 7687.

AAAA

OK

AT+QIRD=2                                         //Retrieve data.
+QIRD: 0                                          //No data in buffer.

OK

AT+QISEND=2,10,"10.7.76.34",7687                //Send data to the remote client whose IP is
                                                    10.7.76.34 and remote port is 7687.

>1234567890

```

SEND OK

### 3.6.4. Close a UDP Service

**AT+QICLOSE=2**

//Close the service.

OK

## 3.7. PING under IPv4

**AT+QPING=1,"www.example.com"**

//Ping www.example.com in context 1. Before pinging the destination IP address, the host should activate the context via **AT+QIACT**.

OK

+QPING: 0,"192.0.2.2",32,192,255

+QPING: 0,"192.0.2.2",32,240,255

+QPING: 0,"192.0.2.2",32,241,255

+QPING: 0,"192.0.2.2",32,479,255

+QPING: 0,4,4,0,192,479,288

## 3.8. Synchronize Local Time

**AT+QNTP=1,"192.0.2.2",123**

//Synchronize local time with NTP server "192.0.2.2:123". Before synchronizing the time, the host should activate the context with **AT+QIACT**.

OK

+QNTP: 0,"2019/07/21,06:10:59+00"

**AT+CCLK?**

+CCLK: "19/07/21,06:11:05+00"

OK

### 3.9. Get Last Error Code

```
AT+QIOPEN=1,"TCP","192.0.2.2",8009,0,1 //Send AT+QIOPEN with missing <connectID>.
```

```
ERROR
```

```
AT+QIGETERROR
```

```
+QIGETERROR: 552, invalid parameters
```

```
OK
```

# 4 Summary of Error Codes

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

**Table 2: Summary of Error Codes**

<err>	<errcode_description>
0	operate success
550	Unknown error
551	Operation blocked
552	Invalid parameters
553	Memory not enough
554	Socket creation failed
555	Operation not supported
556	Socket bind failed
557	Socket listen failed
558	Socket write failed
559	Socket read failed
560	Socket accept failed
561	Activate PDP context failed
562	Deactivate PDP context failed
563	Socket identity has been used
564	Dns busy

---

565	Dns parse failed
566	Socket connect failed
567	Socket has been closed
568	Operation busy
569	Operation timeout
570	PDP context broken down
571	Cancel sending
572	Operation not allowed
573	APN not configured
574	Port busy

---

# 5 Appendix References

**Table 3: Terms and Abbreviations**

Abbreviation	Description
3GPP	3rd Generation Partnership Project
ACK	Acknowledge
APN	Access Point Name
ASCII	American Standard Code for Information Interchange
CHAP	Challenge Handshake Authentication Protocol
CS	Circuit Switching
DNS	Domain Name System
DTR	Data Terminal Ready
FIN	Finish
ID	Identifier
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
LTE	Long-Term Evolution
NB-IoT	Narrowband Internet of Things
NTP	Network Time Protocol
NVM	Non-Volatile Memory
PAP	Password Authentication Protocol I
PDP	Packet Data Protocol
PPP	Point-to-Point Protocol



---

PSSACK	Packet SwitchingSelective Acknowledgment
PS	Packet Switching
QoS	Quality of Service
TCP	Transmission Control Protocol
TTL	Time To Live
UART	Universal Asynchronous Receiver& Transmitter
UDP	User Datagram Protocol
URC	Unsolicited Result Code
USB	Universal Serial Bus
(U)SIM	(Universal) Subscriber Identity Module
UTC	Coordinated Universal Time

---