SIM7500_SIM7600_SIM7800 Series_TCPIP_AT Command Manual_V1.00

LTE Module

Shanghai SIMCom Wireless Solutions Ltd.
Building A, SIM Technology Building, No.633, Jinzhong Road
Changning District200335
Tel:86-21-31575100/31575200
support@simcom.com
www.simcom.com

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

General Notes

SIMCom offers this information as a service to its customers, to support application and engineering efforts that use the products designed by SIMCom. The information provided is based upon requirements specifically provided to SIMCom by the customers. SIMCom has not undertaken any independent search for additional relevant information, including any information that may be in the customer's possession. Furthermore, system validation of this product designed by SIMCom within a larger electronic system remains the responsibility of the customer or the customer's system integrator. All specifications supplied herein are subject to change.

Copyright

Version History

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

Content

1 Introduction

1.1 The process of Using TCP/IP Commands

Through TCP/IP AT commands, host can activate/deactivate PDP context, start/close socket service and send/receive data via socket service. Figure 1 illustrates how to use TCP/IP AT commands:

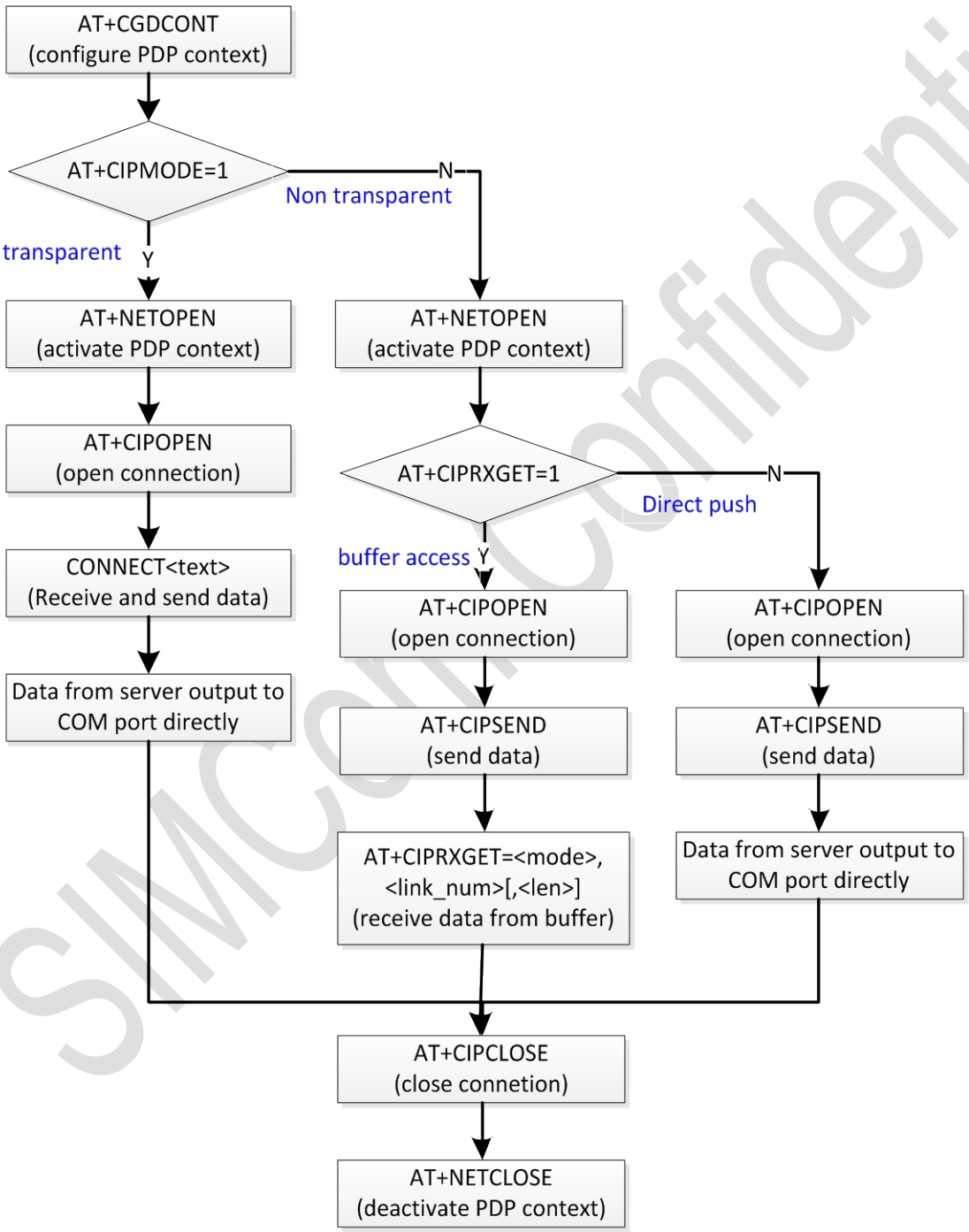
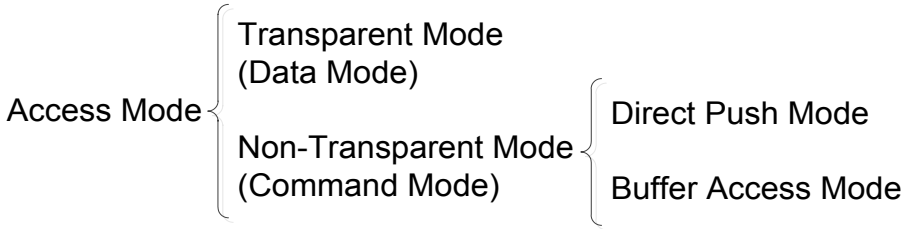


Figure 1: Flow Chart of Using TCP/IP Commands

1.2 Description of Data Access Mode



The default mode is direct push mode.

1. Direct Push Mode

In direct push mode, user can send data by AT+CIPSEND. the received data will be outputted to COM port directly by URC as "+RECV FROM:<IP ADDRESS>:<PORT><CR><LF>+IPD(data length)<CR><LF><data>".

2. Buffer Access Mode

AT+CIPRXGET=1 is used to enter into buffer access mode. In buffer access mode, user send data by AT+CIPSEND. after receiving data, the module will buffer it and report a URC as "+CIPRXGET: 1,<link_num>" to notify the host. Then host can retrieve data by AT+CIPRXGET.

3. Transparent Access Mode

AT+CIPMODE=1 is used to enter into transparent access mode. In transparent mode, the data received from COM port will be sent to internet directly, and the received data from Internet will be output to COM port directly as well. "+++" is used to exit from transparent access mode. When "+++" returns OK, the module will be switched to command mode. In transparent access mode, host cannot execute any AT command.

Note: Currently, only one socket is available under transparent mode, either TCP client or TCP server.**In** transparent mode, the first server (<server_index> = 0) and the first client socket(<link_num> = 0) are used for transparent mode operation. Other servers (<server_index> = 1-3) and other client sockets (<link_num> = 1-9) are still used in command mode.

4. Switch Between Data Mode and Command Mode

Hardware flow control is recommended.

Currently, USB->modem port, USB->AT port and UART port all support hardware flow control.

Data mode -> Command mode

Software switching: escape sequence +++. Please take care, this is a complete command, do not separate each character, also take care that the time delay before and after this sequence should be more than 1000 milliseconds, the interval of each character should not be more than 900 milliseconds.

Hardware switching: DTR pin could be used to trigger data mode and command mode.Command AT&D1 should be configured before application.

Command Mode -> Data Mode

ATO is used to enter into transparent access mode from command mode. If it enters into transparent access mode successfully, CONNECT<text> will be returned.

2 Description of AT Command

2.1 AT+NETOPEN Start Socket Service

AT+NETOPEN is used to start socket service by activating PDP context. You must execute AT+NETOPEN before any other TCP/UDP related operations.

AT+NETOPEN Start Socket Service	
Read Command **AT+NETOPEN?**	Response **+NETOPEN: <net_state>** **OK**
Execute Command **AT+NETOPEN**	Response If the PDP context has not been activated or the network closed abnormally, response: **OK** **+NETOPEN: <err>** when the PDP context has been activated successfully, if you execute AT+NETOPEN again, response: **+IP ERROR: Network is already opened** **ERROR** other: **ERROR**
Maximum Response Time	Range: 3000ms-120000ms default: 120000ms (it can be set by AT+CIPTIMEOUT)

Defined Values

<net_state>	Integer type, indicates the state of PDP context activation. 0 network close (deactivated) 1 network open(activated)
<err>	Integer type,the result of operation. 0 is success, other value is failure, please refer to Chapter 4 for details

2.2 AT+NETCLOSE Stop Socket Service

AT+NETCLOSE is used to stop socket service by deactivating PDP context. It also can close all the opened socket connections when you didn't close these connections by AT+CIPCLOSE.

AT+NETCLOSE Stop Socket Service	
Execute Command **AT+NETCLOSE**	Response If the PDP context has been activated, response: **OK** **+NETCLOSE: <err>** If the PDP context has not been activated, response: **+NETCLOSE: <err>** **ERROR** other: **ERROR**

Defined Values	
<err>	Integer type,the result of operation. 0 is success, other value is failure, please refer to Chapter 4 for details

2.3 AT+CIPOPEN Establish Connection in Multi-Socket Mode

You can use AT+CIPOPEN to establish a connection with TCP server and UDP server, the maximum of the connections is 10.

AT+CIPOPEN Establish Connection in Multi-Socket Mode	
Test Command **AT+CIPOPEN=?**	Response **+CIPOPEN: (0-9),("TCP","UDP")** **OK**
Read Command **AT+CIPOPEN?**	Response **+CIPOPEN: <link_num> [,<type>,<serverIP>,<serverPort>,<index>]** **+CIPOPEN: <link_num> [,<type>,<serverIP>,<serverPort>,<index>]** **[…]**

	OK If a connection identified by <link_num>has not been established successfully, +CIPOPEN: <link_num> will be returned.
TCP connection **AT+CIPOPEN=<link_num>,"TCP",<serverIP>,<serverPort>[,<localPort>]**	Response if PDP context has been activated successfully, response: **OK** **+CIPOPEN: <link_num>,<err>** when the <link_num> is greater than 10, response： **+IP ERROR: Invalid parameter** **ERROR** If PDP context has not been activated, or the connection has been established, or parameter is incorrect, or other errors, response: **+CIPOPEN: <link_num>,<err>** **ERROR** Transparent mode for TCP connection: When you want to use transparent mode to transmit data, you should set AT+CIPMODE=1 before AT+NETOPEN. And if AT+CIPMODE=1 is set, the <link_num> is restricted to be only 0. if success **CONNECT [<text>]** if failure **CONNECT FAIL** other: **ERROR**
UDP Connection **AT+CIPOPEN=<link_num>,"UDP",,,<localPort>**	if PDP context has been activated successfully, response: **+CIPOPEN: <link_num>,0** **OK** when the <link_num> is greater than 10, response： **+IP ERROR: Invalid parameter** **ERROR** If PDP context has not been activated, or the connection has been established, or

	parameter is incorrect, or other errors, response: **+CIPOPEN: <link_num>,<err>** **ERROR** Other: **ERROR**
Maximum Response Time	Range: 3000ms-120000ms default: 120000ms (it can be set by AT+CIPTIMEOUT)

Defined Values

<link_num>	Integer type,identifies a connection. Range is 0-9. If AT+CIPMODE=1 is set, the <link_num> is restricted to be only 0.
<type>	String type, identifies the type of transmission protocol. If AT+CIPMODE=1 is set, the <type> is restricted to be only “TCP”. TCP Transmission Control Protocol UDP User Datagram Protocol
<serverIP>	String type, identifies the IP address of server. The IP address format consists of 4 octets, separated by decimal point, like ”AAA.BBB.CCC.DDD”. Also the domain name is supported here. **NOTE:** If the domain name is inputted here, the timeout value for the AT+CIPOPEN shall be decided by **AT+CIPDNSSET.**
<serverPort>	Integer type, identifies the port of TCP server, range is 0-65535. **NOTE:** When open port as TCP, the port must be the opened TCP port; When open port as UDP, the port may be any port. But, for Qualcomm, connecting the port 0 is regarded as an invalid operation.
<localPort>	Integer type,identifies the port of local socket, range is 0-65535.
<index>	Integer type, indicates whether the module is used as a client or server. When used as server, the range is 0-3, <index> is the server index to which the client is linked. -1 -- TCP client 0-3 -- TCP server index
<text>	String type, indicates CONNECT result code, please refer to ATX/AT\V/AT&E command for the string formats.
<err>	Integer type,the result of operation. 0 is success, other value is failure, please refer to Chapter 4 for details

2.4 AT+CIPSEND Send data through TCP or UDP Connection

AT+CIPSEND is used to send data to remote side. If service type is TCP, the data first is sent to the module's internal TCP/IP stack, then sent to server by protocol stack. The <length> field can be empty, when it is empty, Each <Ctrl+Z> character present in the data should be coded as <ETX><Ctrl+Z>. Each <ESC> character present in the data should be coded as <ETX><ESC>. Each <ETX> character will be coded as <ETX><ETX>. Single <Ctrl+Z> means end of the input data. Single <ESC> is used to cancel the sending.
<ETX> is 0x03, and <Ctrl+Z> is 0x1A, <ESC> is 0x1B.

AT+CIPSENDSend data through TCP or UDP Connection	
Test Command **AT+CIPSEND=?**	Response **AT+CIPSEND: (0-9),(1-1500)** **OK**
If service type is "TCP", send data with changeable length **AT+CIPSEND=<link_num>,** Response ">", then type data to send, tap CTRL+Z to send data, tap ESC to cancel the operation	Response: If the connection identified by <link_num> has been established successfully, response: **>** **<input data>** **CTRL+Z** **OK** **+CIPSEND: <link_num>,<reqSendLength>, <cnfSendLength>** If <reqSendLength> is equal <cnfSendLength>, it means that the data has been sent to TCP/IP protocol stack successfully. If the connection has not been established, abnormally closed, or parameter is incorrect, response: **+CIPERROR: <err>** **ERROR** Other: **ERROR**

If service type is “TCP”, send data with fixed length **AT+CIPSEND=<link_num>,<length>** Response “>”, type data until the data length is equal to <length>	Response: If the connection identified by <link_num> has been established successfully, response: **>** **<input data with specified length>** **OK** **+CIPSEND: <link_num>,<reqSendLength>, <cnfSendLength>** If <reqSendLength> is equal <cnfSendLength>, it means that the data has been sent to TCP/IP protocol stack successfully. If the connection has not been established, abnormally closed, or parameter is incorrect, response: **+CIPERROR: <err>** **ERROR** Other: **ERROR**
If service type is “UDP”, send data with changeable length **AT+CIPSEND=<link_num>,,<serverIP>,<serverPort>** Response “>”, then type data to send, tap CTRL+Z to send data, tap ESC to cancel the operation	Response: If the connection identified by <link_num> has been established successfully, response: **>** **<input data>** **CTRL+Z** **OK** **+CIPSEND: <link_num>,<reqSendLength>, <cnfSendLength>** If the connection has not been established, abnormally closed, or parameter is incorrect, response: **+CIPERROR: <err>** **ERROR** Other: **ERROR**
If service type is “UDP”, send data with fixed length **AT+CIPSEND=<link_num>,[<l**	Response: If the connection identified by <link_num> has been established successfully, response: **>**

ength>],<serverIP>,<serverPort> Response ">", type data until the data length is equal to <length>	**<input data with specified length>** **OK** **+CIPSEND: <link_num>,<reqSendLength>, <cnfSendLength>** If the connection has not been established, abnormally closed, or parameter is incorrect, response: **+CIPERROR: <err>** **ERROR** Other: **ERROR**
Maximum Response Time	Range: 3000ms-120000ms default: 120000ms (it can be set by AT+CIPTIMEOUT)

Defined Values

<link_num>	Integer type,identifies a connection. Range is 0-9.
<length>	Integer type, indicates the length of sending data, range is 1-1500.
<serverIP>	String type, identifies the IP address of server. The IP address format consists of 4 octets, separated by decimal point, like "AAA.BBB.CCC.DDD".
<serverPort>	Integer type, identifies the port of TCP server, range is 0-65535. **NOTE:** When open port as TCP, the port must be the opened TCP port; When open port as UDP, the port may be any port. But, for Qualcomm, connecting the port 0 is regarded as an invalid operation.
<reqSendLength>	Integer type,the length of the data requested to be sent
<cnfSendLength>	Integer type, the length of the data confirmed to have been sent -1 the connection is disconnected. 0 own send buffer or other side's congestion window are full. **Note**: If the <cnfSendLength> is not equal to the <reqSendLength>, the socket then cannot be used further.
<err>	Integer type,the result of operation. 0 is success, other value is failure, please refer to Chapter 4 for details

2.5 AT+CIPRXGET Set the Mode to Retrieve Data

If set <mode> to 1, after receiving data, the module will buffer it and report a URC as "+CIPRXGET: 1,<link_num>" to notify the host. Then host can retrieve data by AT+CIPRXGET.
If set <mode> to 0, the received data will be outputted to COM port directly by URC as "+RECV FROM:<IP ADDRESS>:<PORT><CR><LF>+IPD(data length)<CR><LF><data>".
The default value of <mode> is 0.

Note:

1. If the buffer is not empty, and the module receives data again, then it will not report a new URC until all the received data has been retrieved by AT+CIPRXGET from buffer.
2. When <mode> is set to 1 and the 2-4 mode will take effect.
3. If initially set <mode> to 1, after doing some data transmitting , set <mode> to 0, then the buffered data of the previously established connection will be output to the serial port directly, and the maximum length of output data at a time is 1500.

AT+CIPRXGET SettheModeto Retrieve Data	
Test Command **AT+CIPRXGET=?**	Response **+CIPRXGET: (0-4),(0-9),(1-1500)** **OK**
Read Command **AT+CIPRXGET?**	Response **+CIPRXGET: <mode>** **OK**
AT+CIPRXGET=<mode> In this case, <mode> canonly be 0 or 1	Response If the parameter is correct, response: **OK** Else, response: **ERROR**
AT+CIPRXGET=2,<link_num>[,<len>] Retrieve data in ACSIIform	Response If <length> field is empty, the default value to read is 1500. Ifthebufferis not empty, response: **+CIPRXGET: <mode>,<link_num>,<read_len>,<rest_len>** **<data>**ACSII form **OK** If the buffer is empty, response: **+IP ERROR: No data** **ERROR**

	If the parameter is incorrect or other error, response: **+IP ERROR: <err_info>** **ERROR** Other **ERROR**
AT+CIPRXGET=3,<link_num>[,<len>] Retrieve data in hex form	Response If <length> field is empty, the default value to read is 750. If the buffer is not empty, response: **+CIPRXGET: <mode>,<link_num>,<read_len>,<rest_len>** **<data>**hex form **OK** If the buffer is empty, response: **+IP ERROR: No data** **ERROR** If the parameter is incorrect or other error, response: **+IP ERROR: <err_info>** **ERROR** Other **ERROR**
AT+CIPRXGET=4,<link_num>	If the parameter is correct, response: **+CIPRXGET: 4,<link_num>,<rest_len>** **OK** If the parameter is incorrect or other error, response: **+IP ERROR: <err_info>** **ERROR** Other: **ERROR**

Defined Values

<mode>	Integer type, sets the mode to retrieve data 0 – set the way to get the network data automatically 1 – set the way to get the network data manually 2 – read data, the max read length is 1500 3 – read data in HEX form, the max read length is 750 4 – get the rest data length
<link_num>	Integer type,identifies a connection. Range is 0-9.
<len>	Integer type, the data length to be read. Not required, the default value is 1500 when <mode>=2, and 750 when <mode>=3.
<read_len>	Integer type, the length of data that has been read.
<rest_len>	Integer type, the length of data which has not been read in the buffer.
<err_info>	String type, displays the cause of occurring error, please refer to Chapter 3 for details.

2.6 AT+CIPCLOSE Close TCP or UDP Socket

AT+CIPCLOSE is used to close TCP or UDP Socket

AT+CIPCLOSECloseTCPor UDP Socket	
Test Command **AT+CIPCLOSE=?**	Response **+CIPCLOSE: (0-9)** **OK**
Read Command **AT+CIPCLOSE?**	Response **+CIPCLOSE:<link0_state>,<link1_state>,<link2_state>, <link3_state>,<link4_state>,<link5_state>,<link6_state>, <link7_state>,<link8_state>,<link9_state>** **OK**
Write Command **AT+CIPCLOSE=<link_num>**	Response: If service type is TCP and the connection identified by <link_num> has been established, response: **OK** **+CIPCLOSE: <link_num>,<err>** If service type is UDP and the connection identified by <link_num> has been established, response: **+CIPCLOSE: <link_num>,0** **OK**

	If the connection has not been established, abnormally closed, or parameter is incorrect, response: **+CIPCLOSE: <link_num>,<err>** **ERROR** Other: **ERROR**

Defined Values

<link_num>	Integer type,identifies a connection. Range is 0-9.
<linkX_state>	Integer type, indicates state ofconnection identified by <link_num>. Range is 0-1. 0 -- disconnected 1 -- connected
<err>	Integer type,the result of operation. 0 is success, other value is failure, please refer to Chapter 4 for details

2.7 AT+IPADDR Inquire Socket PDP address

You can use AT+HTTPDATA to input data to post when you send a HTTP/HTTPS POST request.

AT+IPADDR Inquire Socket PDP Address	
Execute Command **AT+IPADDR**	Response Response If PDP context has been activated successfully, response **+IPADDR: < ip_address>** **OK** Else, response: **+IP ERROR: Network not opened** **ERROR**

Defined Values

<ip_address>	String type, identifies the IP address of current active socket PDP.

2.8 AT+CIPHEAD Add an IP Header When Receiving Data

AT+CIPHEAD is used to add an IP header when receiving data.

AT+CIPHEAD Add an IP Header When Receiving Data	
Test Command **AT+CIPHEAD=?**	Response **+CIPHEAD: (0-1)** **OK**
Read Command **AT+CIPHEAD?**	Response **+CIPHEAD: <mode>** **OK**
Write Command **AT+CIPHEAD=<mode>**	Response If the parameter is correct, response: **OK** Else, response: **ERROR**
Execute Command **AT+CIPHEAD**	Response Set default value:(<mode>=1) **OK**

Defined Values

<mode>	Integer type, indicates whether adding an IP header or not when receiving data 0– not add IP header 1– add IP header, the format is "+IPD(data length)"

2.9 AT+CIPSRIP Show Remote IP Address and Port

AT+CIPSRIP is used to set whether to display IP address and port of server when receiving data.

AT+CIPSRIP Show Remote IP Address and Port	
Test Command **AT+CIPSRIP =?**	Response **+CIPSRIP: (0-1)** **OK**
Read Command **AT+CIPSRIP?**	Response **+CIPSRIP: <mode>** **OK**

Write Command **AT+CIPSRIP =<mode>**	Response If the parameter is correct, response: **OK** Else, response: **ERROR**
Execute Command **AT+CIPSRIP**	Response Set default value:(<mode>=1) **OK**

Defined Values

<mode>	Integer type, indicates whether to show IP address and port of server or not when receiving data. 0 –not show 1– show,the format is as follows: "RECV FROM:<IP ADDRESS>:<PORT>"

2.10 AT+CIPMODE Set TCP/IP Application Mode

AT+CIPMODE is used to select transparent mode (data mode) or non-transparent mode (command mode).The default mode is non-transparent mode.

AT+CIPMODE Set TCP/IP Application Mode	
Test Command **AT+CIPMODE =?**	Response **+CIPMODE: (0-1)** **OK**
Read Command **AT+CIPMODE?**	Response **+CIPMODE: <mode>** **OK**
Write Command **AT+CIPMODE=<mode>**	Response If the parameter is correct, response: **OK** Else, response: **ERROR**
Execute Command **AT+CIPMODE**	Response Set default value:(<mode> =0) **OK**

Defined Values

<mode>	Integer type, sets TCP/IP application mode 0–Non transparent mode

	1–Transparent mode

2.11 AT+CIPSENDMODE Set Sending Mode

AT+CIPSENDMODE is used to select sending mode when service type is "TCP".
If set <mode> to 1, when sending data by AT+CIPSEND, the URC "+CIPSEND: <link_num>,<reqSendLength>, <cnfSendLength>" will not be returned until module receives the server's ACK message to the sent data last time.
If set <mode> to 0, the URC "+CIPSEND: <link_num>,<reqSendLength>, <cnfSendLength>" will be returned If the data has been sent to module's internal TCP/IP protocol stack. In this case, the module doesn't need to wait for the server's ACK message.
The default mode is sending without waiting peer TCP ACK mode.

AT+CIPSENDMODE Set Sending Mode	
Test Command **AT+CIPSENDMODE =?**	Response **+CIPSENDMODE: (0-1)** **OK**
Read Command **AT+CIPSENDMODE?**	Response **+CIPSENDMODE: <mode>** **OK**
Write Command **AT+CIPSENDMODE=<mode>**	Response If the parameter is correct, response: **OK** Else, response: **ERROR**

Defined Values

<mode>	Integer type, sets sending mode 0–sending without waiting peer TCP ACK mode 1–sending wait peer TCP ACK mode

2.12 AT+CIPTIMEOUT Set TCP/IP Timeout Value

AT+CIPTIMEOUT is used to set timeout value for AT+NETOPEN/AT+CIPOPEN/AT+CIPSEND.

AT+CIPTIMEOUTSet TCP/IP Timeout Value	
Read Command **AT+CIPTIMEOUT?**	Response **+CIPTIMEOUT:** **<netopen_timeout>,<cipopen_timeout>,<cipsend_timeout>**

	OK
Write Command **AT+CIPTIMEOUT=[<netopen_timeout>][,[<cipopen_timeout>][,[<cipsend_timeout>]]]**	Response If the parameter is correct, response: **OK** Else, response: **ERROR**

Defined Values

<netopen_timeout>	Integer type, timeout value for AT+NETOPEN. default is120000ms. Range is 3000ms-120000ms.
<cipopen_timeout>	Integer type, timeout value for AT+CIPOPEN. default is120000ms. Range is 3000ms-120000ms.
<cipsend_timeout>	Integer type, timeout value for AT+CIPSEND. default is120000ms. Range is 3000ms-120000ms.

2.13 AT+CIPCCFG Configure Parameters of Socket

AT+CIPCCFG is used to configure parameters of socket.

AT+CIPCCFG Configure Parameters of Socket	
Test Command **AT+CIPCCFG=?**	Response **+CIPCCFG: (0-10),(0-1000),(0),(0-1),(0-1),(0-1),(500-120000)** **OK**
Read Command **AT+CIPCCFG?**	Response **+CIPCCFG:<NmRetry>,<DelayTm>,<Ack>,<errMode>,<Header-Type>,<AsyncMode>,<TimeoutVal>** **OK**
Write Command **AT+CIPCCFG=[<NmRetry>][,[<DelayTm>][,[<Ack>][,[<errMode>][,]<HeaderType>][,[[<AsyncMode>][,[<TimeoutVal>]]]]]]]**	Response If the parameter is correct, response: **OK** Else, response: **ERROR**
Execute Command **AT+CIPCCFG**	Response Set default value: **OK**

Defined Values

<NmRetry>	Integer type, number of retransmission to be made for an IP packet. Range is 0-10. The default value is 10.

<DelayTm>	Integer type, number of milliseconds to delay to output data of Receiving. Range is 0-1000. The default value is 0.
<Ack>	Integer type, it can only be set to 0. It's used to be compatible with old TCP/IP command set.
<errMode>	Integer type,sets mode of reporting <err_info>, default value is 1. 0 error result code with numeric values 1 error result code with string values
<HeaderType>	Integer type, select which data header is used when receiving data, it only takes effect in multi-client mode. Default value is 0. 0 add data header, the format is"+IPD<data length>" 1 add data header, the format is "+RECEIVE,<link num>,<data length>"
<AsyncMode>	Integer type, range is 0-1. Default value is 0. It's used to be compatible with old TCP/IP command set.
<TimeoutVal>	Integer type, set the minimum retransmission timeout value for TCP connection. Range is 500ms-120000ms. Default is 500ms.

2.14 AT+SERVERSTART Startup TCP Sever

AT+SERVERSTART is used to startup a TCP server, and the server can receive the request of TCP client. After the command executes successfully, an unsolicited result code is returned when a client tries to connect with module and module accepts request. The unsolicited result code is+CLIENT: < link_num>,<server_index>,<client_IP>:<port>.

AT+SERVERSTART Startup TCP Sever	
Test Command **AT+SERVERSTART=?**	Response **+SERVERSTART: (0-65535),(0-3)** **OK**
Read Command **AT+SERVERSTART?**	Response If the PDP context has not been activated successfully, response: **+CIPERROR: <err>** **ERROR** If there exists opened server, response: **[+SERVERSTART: <server_index>,< port>** **…]** **OK** Other:

	ERROR
Write Command **AT+SERVERSTART=<port>,<server_index>[,<backlog>]**	Response if there is no error, response: **OK** If the PDP context has not been activated, or the server identified by <server_index> has been opened, or the parameter is not correct, or other errors, response: **+CIPERROR: <err>** **ERROR** Other: **ERROR**

Defined Values

<port>	Integer type, identifies the listening port of module when used as a TCP server.range is 0-65535.
<server_index>	Integer type, the TCP server index, range is 0-3.
<backlog>	Integer type, the maximum connections can be queued in listening queue. Range is 1-3 . Default is 3.

2.15 AT+SERVERSTOP Stop TCP Sever

AT+SERVERSTOP is used to stop TCP server. Before stopping a TCP server, all sockets <server_index> of which equals to the closing TCP server index must be closed first.

AT+SERVERSTOP Stop TCP Sever	
Write Command **AT+SERVERSTOP=<server_index>**	Response If there exists open connection with the server identified by <server_index>, or the server identified by <server_index> has not been opened, or the parameter is incorrect, response: **+SERVERSTOP: <server_index>,<err>** **ERROR** If the server socket is closed immediately, response: **+SERVERSTOP: <server_index>,0** **OK** (In general, the result is shown as below.)

	If the server socket starts to close, response: **OK** **+SERVERSTOP: <server_index>,<err>** Other: **ERROR**

Defined Values

<server_index>	Integer type, the TCP server index, range is 0-3.
<err>	Integer type,the result of operation. 0 is success, other value is failure, please refer to Chapter 4 for details

2.16 AT+CIPACK Query TCP Connection Data Transmitting Status

AT+CIPACK is used to query TCP connection data transmitting status.

AT+CIPACK Query Connection Data Transmitting State	
Test Command **AT+CIPACK=?**	Response **+CIPACK: (0-9)** **OK**
Write Command **AT+CIPACK=<link_num>**	Response If the PDP context has not been activated, or the connection identified by <link_num> has not been established, abnormally closed, or the parameter is incorrect, or other errors, response: **+IP ERROR: <err_info>** **ERROR** If the connection has been established, and the service type is “TCP”, response: **+CIPACK: <sent_data_size>,<ack_data_size>,<recv_data_size>** **OK**

Defined Values

<link_num>	Integer type,identifies a connection. Range is 0-9.
<sent_data_size>	Integer type, the total length of sent data
<ack_data_size>	Integer type, the total length of acknowledged data .
<recv_data_size>	Integer type, the total length of received data
<err>	Integer type,the result of operation.

	0 is success, other value is failure, please refer to Chapter 4 for details
<err_info>	String type, displays the cause of occurring error, please refer to Chapter 3 for details.

2.17 AT+CDNSGIP Query the IP Address of Given Domain Name

AT+CDNSGIP is used to query the IP address of given domain name.

AT+CDNSGIP Query the IP Address of Given Domain Name	
Test Command **AT+CDNSGIP=?**	Response **OK**
Write Command **AT+CDNSGIP=<domain name>**	Response If the given domain name has related IP, response: **+CDNSGIP: 1,<domain name>,<IP address>** **OK** If the given name has no related IP, response: **+CDNSGIP: 0,<dns error code>** **ERROR** Other: **ERROR**

Defined Values

<domain name>	String type(string should be included in quotation marks), indicates the do main name. The maximum length of domain name is 254. Valid characters allowed in the domain name area-z, A-Z, 0-9, "-" (hyphen) and ".". A domain name is made up of onelabel name or more l abel names separated by "." (eg: AT+CDNSGIP="aa.bb.cc"). For labelnames separated by ".", length of each label must be no more than 63 characters. The be-ginning character of the domain name an d of labels should be an alphanumeric character.
<IP address>	String type, indicates the IP address corresponding to the domain name.
<dns error code>	Integer type, indicates the error code. 10 DNS GENERAL ERROR

Example

```
AT+CDNSGIP="www.baidu.com"
+CDNSGIP: 1,"www.baidu.com","61.135.169.121"

OK
```

2.18 AT+CDNSGHNAME Query the Domain Name of given IP Address

AT+CDNSGHNAME is used to query the domain name of given IP address.

AT+CDNSGHNAME Query the Domain Name of Given IP Address	
Test Command **AT+CDNSGHNAME=?**	Response **OK**
Write Command **AT+CDNSGHNAME=<IP address>**	Response If the given IP address has related domain name, response: **+CDNSGHNAME: <index>,<domain name>,<IP address>** **OK** If the given IP address has no related domain name, response: **+CDNSGHNAME: 0,<dns error code>** **ERROR** Other: **ERROR**

Defined Values

<domain name>	String type(string should be included in quotation marks), indicates the do main name. The maximum length of domain name is 254. Valid characters allowed in the domain name area-z, A-Z, 0-9, "-" (hyphen) and ".". A domain name is made up of onelabel name or more l abel names separated by "." (eg: AT+CDNSGIP="aa.bb.cc"). For labelnames separated by ".", length of each label must be no more than 63 characters. The be-ginning character of the domain name an d of labels should be an alphanumeric character.
<IP address>	String type(string should be included in quotation marks), indicates the IP address corresponding to the domain name.
<dns error code>	Integer type, indicates the error code. 10 DNS GENERAL ERROR
<index>	Integer type, indicates DNS result index. This value is always 1 if performing successfully. Currently only the first record returned from the DNS server will be reported.

Example

```
AT+CDNSGHNAME="58.32.231.148"
+CDNSGHNAME: 1,"mail.sim.com.","58.32.231.148"
```

OK

2.19 AT+CIPDNSSET Set DNS Query Parameters

AT+CIPDNSSET is used to set DNS Query Parameters.

AT+CIPDNSSET Set DNS Query Parameters	
Read Command **AT+CIPTIMEOUT?**	Response **+CIPDNSSET: 3,30000,7** **OK**
Write Command **AT+CIPDNSSET=[<max_net_retries>][,[<net_timeout>][,[<max_query_retries>]]]**	Response If the parameter is correct, response: **OK** Else, response: **ERROR**

Defined Values

<max_net_retries>	Integer type, maximum retry times for opening PS network to perform DNS query. Range is 0-3. Default is 3.
<netopen_timeout>	Integer type, timeout value for each opening PS network operation when performing DNS query. Range is 30000ms-120000ms. Default value is 30000ms.
<max_query_retries>	Integer type, maximum retry times for performing DNS query using UDP packet. Range is 0-7. Default value is 7.

3 Description of <err_info>

The fourth parameter <errMode> of AT+CIPCCFG is used to determine how <err_info> is displayed.
If <errMode> is set to 0, the <err_info> is displayed with numeric value.
If <errMode>is set to 1, the <err_info> is displayed with string value.
The default is displayed with string value.

Numeric Value	String Value
21	Operation failed
0	Connection time out
1	Bind port failed
2	Port overflow

Numeric Value	String Value
3	Create socket failed
4	Network is already opened
5	Network is already closed
6	No clients connected
7	No active client
8	Network not opened
9	Client index overflow
10	Connection is already created
11	Connection is not created
12	Invalid parameter
13	Operation not supported
14	DNS query failed
15	TCP busy
16	Netclose failed for socket opened
17	Sending time out
18	Sending failure for network error
19	Open failure for network error
20	Server is already listening
22	No data

4 Description of <err>

<err>	Description of <err>
0	operation succeeded
1	Network failure
2	Network not opened
3	Wrong parameter
4	Operation not supported
5	Failed to create socket
6	Failed to bind socket
7	TCP server is already listening
8	Busy
9	Sockets opened
10	Timeout
11	DNS parse failed for AT+CIPOPEN
12	Unknown error

5 Information Elements related to TCP/IP

Information	Description
+CIPEVENT: NETWORK CLOSED UNEXPECTEDLY	Network is closed for network error(Out of service, etc). When this event happens, user's application needs to check and close all opened sockets, and then uses AT+NETCLOSE to release the network library if AT+NETOPEN? shows the network library is still opened.
+IPCLOSE: <client_index>, <close_reason>	Socket is closed passively. <client_index> is the link number. <close_reason>: 0 - Closed by local, active 1 - Closed by remote, passive 2 - Closed for sending timeout
+CLIENT: < link_num>,<server_index>,<client_IP>:<port>	TCP server accepted a new socket client, the index is<link_num>, the TCP server index is <server_index>. The peer IP address is <client_IP>, the peer port is <port>.

6 Example

6.1 Configure and Activate context

6.1.1 Network Environment

TCP/IP application is based on GPRS network; so, ensure GPRS network is available before TCP/IP setup.

```
AT+CSQ
+CSQ: 25,99

OK
AT+CREG?
+CREG: 0,1

OK
```

```
AT+CPSI?
+CPSI: LTE,Online,460-00,0x1816,27593483,139,EUTRAN-BAND39,38400,5,5,-88,-868,-578,18

OK
AT+CGREG?
+CGREG: 0,1

OK
```

6.1.2 Configure Context

```
AT+CGDCONT=1,"IP","CMNET"
OK
```

6.1.3 Activate context

```
AT+NETOPEN
OK

+NETOPEN: 0
AT+IPADDR
+IPADDR: 10.148.0.17

OK
```

6.1.4 Deactivate Context

```
AT+NETCLOSE
OK

+NETCLOSE: 0
```

6.2 TCP Client

6.2.1 TCP Client Works in Direct Push Mode

6.2.1.1 Set up TCP Client Connection

```
AT+NETOPEN
OK

+NETOPEN: 0
AT+CIPOPEN=1,"TCP","117.131.85.139",5253    // set up a TCP connection, <link_num> is 1. Before
                                               using AT+CIPOPEN, host should activate PDP
                                               Context with AT+NETOPEN first.
OK

+CIPOPEN: 1,0
```

6.2.1.2 Send data to Server

```
AT+CIPSEND=1,5    // send data with fixed length
>HELLO
OK

+CIPSEND: 1,5,5
AT+CIPSEND=1,    // send data with changeable length, <CTRL+Z> to end
>HELLOWORLD<CTRL+Z>
OK

+CIPSEND: 1,10,10
```

6.2.1.3 Receive Data From Server

```
RECV FROM:117.131.85.139:5253      // data from server directly output to COM
+IPD16
data from server
```

6.2.1.4 Close TCP Connection

```
AT+CIPCLOSE=1
OK

+CIPCLOSE: 1,0
```

6.2.2 TCP Client Works in Buffer Access Mode

6.2.2.1 Set up TCP Client connection

```
AT+NETOPEN
OK

+NETOPEN: 0
AT+CIPRXGET=1// buffer access mode, get data by AT+CIPRXGET
OK
AT+CIPOPEN=1,"TCP","117.131.85.139",5253//set up a TCP connection, <link_num> is 1
OK

+CIPOPEN: 1,0
```

6.2.2.2 Send Data to Server

```
AT+CIPSEND=1,5// send data with fixed length
>hello
OK

+CIPSEND: 1,5,5
```

6.2.2.3 Receive Data from Server

```
+CIPRXGET: 1,1// URC to notify host of data from server
AT+CIPRXGET=4,1// query the length of data in the buffer of socket with <link_num>=1
+CIPRXGET: 4,1,16

OK
AT+CIPRXGET=2,1,5       // get data in ASCII form
+CIPRXGET: 2,1,5,11
data
```

```
OK
AT+CIPRXGET=3,1,5// get data in hex form
+CIPRXGET: 3,1,5,6
66726F6D20

OK
AT+CIPRXGET=4,1// read the length of unread data in buffer
+CIPRXGET: 4,1,6

OK
AT+CIPRXGET=2,2 // the connection identified by link_num=2 has not been established
+IP ERROR: No data

ERROR
AT+CIPRXGET=2,1
+CIPRXGET: 2,1,6,0
server

OK
AT+CIPRXGET=4,1// all the data in buffer has been read, the rest_len is 0.
+CIPRXGET: 4,1,0

OK
```

6.2.2.4 Close TCP Connection

```
AT+CIPCLOSE=1
OK

+CIPCLOSE: 1,0
```

6.2.3 TCP Client Works in Transparent Access Mode

6.2.3.1 Set up TCP Client Connection

```
AT+CIPMODE=1// Enter into transparent mode by at+cipmode=1
OK
AT+NETOPEN
OK

+NETOPEN: 0
```

// only <link_num>=0 is allowed to operate with transparent mode.
AT+CIPOEPN=0,"TCP","117.131.85.139",5253
CONNECT 115200

6.2.3.2 Send Data to Server

All data got from com port will be sent to internet directly

6.2.3.3 Receive Data From Server

DATA FROM SERVERDATA FROM SERVER//all the received data from server will be output to com port directly
OK //// sequence of +++ to quit transparent mode
AT+CIPOPEN?
+CIPOPEN: 0,"TCP","117.131.85.139",5253,-1
+CIPOPEN: 1
+CIPOPEN: 2
+CIPOPEN: 3
+CIPOPEN: 4
+CIPOPEN: 5
+CIPOPEN: 6
+CIPOPEN: 7
+CIPOPEN: 8
+CIPOPEN: 9

OK
ATO//ATO to enter transparent mode again
CONNECT 115200
HELLO CLIENT
OK

6.2.3.4 Close TCP Connection

AT+CIPCLOSE=0
OK

CLOSED
+CIPCLOSE: 0,0

6.3 UDP Client

6.3.1 UDP Client Works in Direct Push Mode

6.3.1.1 Set up UDP Client Connection

```
AT+NETOPEN
OK

+NETOPEN: 0
// when set a UDP connection, the remote IP address and port is not necessary, but the local port
must be specified.
AT+CIPOPEN=1,"UDP",,,5000
+CIPOPEN: 1,0

OK
```

6.3.1.2 Send data to Server

```
// for UDP connection, when sending data, user must specify the remote IP address and port
AT+CIPSEND=1,,"117.131.85.139",5254 //send data with changeable length, <CTRL+Z> to end
>HELLOSERVER
OK     <CTRL+Z>

+CIPSEND: 1,11,11
AT+CIPSEND=1,5,"117.131.85.139",5254 //send data with fixed length
>HELLO
OK

+CIPSEND: 1,5,5
```

6.3.1.3 Receive Data From Server

```
RECV FROM:117.131.85.139:5254    //data from server output to COM port directly
+IPD14
HELLO CLIENT
```

6.3.1.4 Close UDP Connection

```
AT+CIPCLOSE=1
+CIPCLOSE: 1,0

OK
```

6.3.2 UDP Client Works in Buffer Access Mode

6.3.2.1 Set up UDP Client connection

```
AT+NETOPEN
OK

+NETOPEN: 0
AT+CIPRXGET=1// buffer access mode, get data by AT+CIPRXGET
OK
// when set a UDP connection, the remote IP address and port is not necessary, but the local port
must be specified.
AT+CIPOPEN=1,"UDP",,,5000
+CIPOPEN: 1,0

OK
```

6.3.2.2 Send Data to Server

```
// for UDP connection, when sending data, user must specify the remote IP address and port
AT+CIPSEND=1,,"117.131.85.139",5254 //send data with changeable length, <CTRL+Z> to end
>HELLOSERVER
OK     <CTRL+Z>

+CIPSEND: 1,11,11
AT+CIPSEND=1,5,"117.131.85.139",5254 //send data with fixed length
>HELLO
OK

+CIPSEND: 1,5,5
```

6.3.2.3 Receive Data From Server

```
+CIPRXGET: 1,1// URC to notify host of data from server
AT+CIPRXGET=4,1// query the length of data in the buffer of socket with <link_num>=1
+CIPRXGET: 4,1,16

OK
AT+CIPRXGET=2,1,5       // get data in ASCII form
+CIPRXGET: 2,1,5,11
data

OK
AT+CIPRXGET=3,1,5// get data in hex form
+CIPRXGET: 3,1,5,6
66726F6D20

OK
AT+CIPRXGET=4,1// read the length of unread data in buffer
+CIPRXGET: 4,1,6

OK
AT+CIPRXGET=2,2 // the connection identified by link_num=2 has not been established
+IP ERROR: No data

ERROR
AT+CIPRXGET=2,1
+CIPRXGET: 2,1,6,0
server

OK
AT+CIPRXGET=4,1// all the data in buffer has been read, the rest_len is 0.
+CIPRXGET: 4,1,0

OK
```

6.3.2.4 Close UDP Connection

```
AT+CIPCLOSE=1
OK

+CIPCLOSE: 1,0
```

6.3.3 UDP Client Works in Transparent Access Mode

6.3.3.1 Set up UDP Client Connection

```
AT+CIPMODE=1
OK
AT+NETOPEN
OK

+NETOPEN: 0
AT+CIPOPEN=0,"UDP","117.131.85.139",5254,5000 //only <link_num>=0 is allowed to operate with
                                                transparent mode.

CONNECT 115200
```

6.3.3.2 Send Data to Server

```
All data got from com port will be sent to internet directly
```

6.3.3.3 Receive Data From Server

```
//data from server output to COM port directly
HELLO CLIENT
HELLO CLIENT

OK // sequence of +++ to quit transparent mode

AT+CIPOPEN?
+CIPOPEN: 0,"UDP","117.131.85.139",5254,-1
+CIPOPEN: 1
+CIPOPEN: 2
+CIPOPEN: 3
+CIPOPEN: 4
+CIPOPEN: 5
+CIPOPEN: 6
+CIPOPEN: 7
+CIPOPEN: 8
+CIPOPEN: 9

OK
```

6.3.3.4 Close UDP Connection

```
AT+CIPCLOSE=0
CLOSED

+CIPCLOSE: 0,0
OK
```

6.4 TCP Server

6.4.1 Transparent Mode

```
AT+CIPMODE=1
OK
AT+NETOPEN
OK

+NETOPEN: 0
AT+SERVERSTART=8080, 0//only <server_index>=0 is allowed to operate with transparent mode.
OK

+CLIENT: 0,0,192.168.108.5:57202//only <link_num> 0 can be used for transparent mode operation.
CONNECT 115200
// sequence of +++ to quit data mode
OK
AT+CIPCLOSE=0 // close client connection
OK

CLOSED
+CIPCLOSE: 0,0

AT+SERVERSTOP=0                    // close server socket
+SERVERSTOP: 0,0
OK
```

6.4.2 Non-Transparent Mode

Module supports 4 sockets to listen.

```
AT+NETOPEN
OK

+NETOPEN: 0,0
AT+SERVERSTART=8080,0
OK
AT+SERVERSTART=9090,1
OK
AT+SERVERSTART=7070,2
OK
AT+SERVERSTART=6060,3
OK

//If a socket is accepted, the following URC will be reported:
+CLIENT: 0,1,192.168.108.5:57202

//User can use AT+CIPOPEN? to check the accepted socket
AT+CIPOPEN?
+CIPOPEN: 0,"TCP","192.168.108.5",57202,1
// last parameter of 1 indicates this is an accepted socket, this server index is 1
+CIPOPEN: 1
+CIPOPEN: 2
+CIPOPEN: 3
+CIPOPEN: 4
+CIPOPEN: 5
+CIPOPEN: 6
+CIPOPEN: 7
+CIPOPEN: 8
+CIPOPEN: 9

OK

AT+CIPSEND=0,5 // only supports fixed-length to send
>HELLO
OK

+CIPSEND: 0,5,5
```

```
AT+SERVERSTOP=0      // if unspecified, it will close 0 channel
+SERVERSTOP: 0,0
OK
AT+SERVERSTOP=1
+SERVERSTOP: 1,0
OK
AT+SERVERSTOP=2
+SERVERSTOP: 2,0
OK
AT+SERVERSTOP=3
+SERVERSTOP: 3,0
OK

AT+NETCLOSE
OK

+NETCLOSE: 0
```

6.5 Extended Information

6.6 Query Connection Status

```
AT+CIPOPEN=1,"TCP","117.131.85.139",5253
OK

+CIPOPEN: 1,0

AT+CIPOPEN? // query the current state of all sockets
+CIPOPEN: 0
+CIPOPEN: 1,"TCP","117.131.85.139",5253,-1
+CIPOPEN: 2
+CIPOPEN: 3
+CIPOPEN: 4
+CIPOPEN: 5
+CIPOPEN: 6
+CIPOPEN: 7
+CIPOPEN: 8
+CIPOPEN: 9

OK

AT+CIPCLOSE?
```

```
+CIPCLOSE: 0,1,0,0,0,0,0,0,0,0

OK
AT+CIPCLOSE=1
OK

+CIPCLOSE: 1,0
AT+CIPCLOSE?
+CIPCLOSE: 0,0,0,0,0,0,0,0,0,0

OK
```