LTE Standard(A) series

HTTP(S) Application Guide

LTE Standard Module Series

Version: 1.3

Date: 2022-08-05

Status: Controlled file

Shanghai Quectel Communications Technology Co., Ltd. (hereinafter referred to as "Quectel") always aims to provide customers with the most timely and comprehensive services. If you need any help, please feel free to contact our Shanghai headquarters, the contact information is as follows:

Shanghai Quectel Communications Technology Co.,
Ltd. Building 5, Phase 3 (B Zone), Science and Technology Oasis, No. 1016 Tianlin Road, Minhang District, Shanghai
Postcode: 200233 Tel: +86 21 5108 6236 Email: info@quectel.com

Or contact our local office, for details, please log in: http://www.quectel.com/cn/support/sales.htm.

If you need technical support or give feedback on the problems in our technical documents, please feel free to visit the website: http://www.quectel.com/cn/support/technical.htm or send an email to: support@quectel.com.

foreword

Quectel provides this document content to support customers' product design. The customer shall design the product according to the specifications and parameters provided in the document. At the same time, you understand and agree that the reference design provided by Quectel is only an example. You agree to use your independent analysis, evaluation and judgment in designing your intended product. Please read this statement carefully before using any hardware, software or service guided by this document. You hereby acknowledge and agree that although Quectel has taken commercially reasonable efforts to provide the best possible experience, this document and the services it refers to are provided to you on an "as available" basis. Quectel may, at its sole discretion, add, modify or restate this document at any time without prior notice.

Use and Disclosure Restrictions

License

Agreement Unless specifically authorized by Quectel, the recipient of the hardware, software, materials and documents provided by our company must keep the received content confidential and shall not use it for any purpose other than the implementation and development of this project.

Copyright

statement Quectel products and third-party products under this agreement may contain related materials protected by Quectel or third-party materials, hardware, software and documents. Unless you have obtained prior written consent, you shall not obtain, use, or disclose the documents and information provided by our company to a third party, or copy, reprint, plagiarize, publish, display, translate, distribute, Incorporate, modify, or create derivative works thereof. Quectel or third parties have exclusive rights to the copyrighted materials and do not grant or transfer any license to patent, copyright, trademark or service mark rights. For the avoidance of doubt, no purchase of any kind shall be deemed to confer a license other than a normal non-exclusive, royalty-free product use license. Quectel has the right to pursue legal responsibility for any violation of confidentiality obligations, unauthorized use or other illegal forms of malicious use of the documents and information.

Trademarks Unless otherwise specified, nothing in this document grants the right to use any trademarks, trade names and names of Quectel or third parties, or their acronyms, or imitations thereof, in advertising, publicity or otherwise.

Third Party Rights

You understand that this documentation may refer to one or more hardware, software and documentation belonging to third parties ("Third Party Materials"). Your use of such third-party materials shall be subject to all limitations and obligations of this document.

Quectel does not make any express or implied warranties or representations with respect to third-party materials, including but not limited to any implied or statutory merchantability or fitness for a particular purpose, quiet benefit rights, system integration, information accuracy, and licensed technology or the licensee's warranty that it does not infringe any third party's intellectual property rights in relation to the use of the licensed technology. Nothing in this Agreement shall constitute Quectel's development, enhancement, modification, distribution, marketing, sale, offer for sale or otherwise of any Quectel product or any other hardware, software, equipment, tools, information or products. Representation or warranty to maintain production. Further, Quectel disclaims any and all warranties arising out of course of dealing, use or trade.

Privacy statement

In order to realize the functions of Quectel products, specific device data will be uploaded to Quectel or third-party servers (including operators, chip suppliers or servers you designate). Quectel strictly abides by relevant laws and regulations, and only retains, uses, discloses or otherwise processes relevant data for the purpose of realizing product functions or under the circumstances permitted by applicable laws. Before you interact with a third party, please understand its privacy protection and data security policies.

disclaimer

1) Quectel shall not be liable for any damages caused by failure to comply with relevant operating or design specifications.

2) Quectel shall not be liable for any inaccuracies, omissions, or use of information in this document. 3) Quectel tries its best to ensure the completeness, accuracy and timeliness of the functions under development, but does not rule out the possibility of errors or omissions in the above functions. Unless otherwise stipulated in the agreement, Quectel does not make any implied or statutory guarantees for the use of functions under development. To the maximum extent permitted by applicable laws, Quectel shall not be liable for any damages suffered as a result of using functions under development, regardless of whether such damages are foreseeable or not.

4) Quectel is not responsible for the accessibility, security, accuracy, availability, legality and completeness of information, content, advertisements, commercial offers, products, services and materials on third-party websites and third-party resources. responsibility.

document history

revision history

version	date	author	change expression
-	2020-10-30	Luffy LIU	document creation
1.0	2020-11-30	Luffy LIU	Controlled version
1.1	2021-04-13	Luffy LIU	1. Add documentation for EC200N-CN and EC600N-CN modules. 2. Update example (Chapter 3). 1. Added documents applicable to modules EC200A series, EC800N-CN and EG915N-EU. 2. Delete the EG912Y-CN module. 3. Add commands AT+QHTTPCFG: "reqheader/add",<header_name>,<header_str> and AT+QHTTPCFG: "reqheader/add",<header_name>,<header_str> (Chapter 2.3.1). 4. Update the <closedind> parameter of AT+QHTTPCFG="closed/ind" number interpretation (chapter 2.3.1). 1. Add applicable modules EC200M-CN, EC600M-CN, EC800M-CN, EG915N-LA and EG912N-EN. 2. Delete the applicable module EC200T series. 3. Add data types "application/json" and "image/jpeg" for parameter <content_type> (Chapter 2.3.1).
1.2	2022-03-02	Larson LI	
1.3	2022-08-05	Larson LI	

Table of contents

Document History .... **....3**
Contents ....4 **Table index**.... **....6**

1 Introduction ....7 Applicable
1.1. Modules.... ....7 1.2. HTTP(S) command usage process.... ....8 HTTP(S) request header
1.3. information description.... ....8 1.3.1. Customize HTTP(S) request header information.... ....8 1.3.2. Output HTTP(S) response header information.... ....9 Data Mode
1.4. Description 9

2 Detailed explanation of HTTP(S) AT commands.... **....10**
2.1. AT command description.... 10
2.1.1. Definitions.... ....10 2.1.2 .AT command sentence.... ....10 2.2. AT example statement.... .11 2.3. Detailed explanation of AT commands.... ....11 2.3.1. AT+QHTTPCFG Configure HTTP(S) server parameters.. ....11 2.3.2. AT+QHTTPURL Set HTTP(S) server URL....ÿ… ....15 2.3.3. AT+QHTTPGET Send GET request to HTTP(S) server.... ...16 2.3.4. AT+QHTTPGETEX Send range GET request to HTTP(S) server.... ....17 2.3.5. AT+QHTTPPOST Send POST request to HTTP(S) server via UART/USB.... 18 2.3.6. AT+QHTTPPOSTFILE Send a POST request to HTTP(S) server through a file....20 2.3.7. AT+ QHTTPREAD Read HTTP(S) server response information via UART/USB....21 2.3.8. AT+QHTTPREADFILE Pass File read HTTP(S) server response information....22 2.3.9. AT+QHTTPSTOP Canceling an HTTP(S) request.... ....twenty three

3 Examples .... ..24 3.1. Access to HTTP server.... ....24 3.1.1. Send HTTP GET request and read response information.... ....24 3.1.2. Send HTTP POST request and read response information.... ..25 3.1.2.1. Get POST request body via UART/ USB.... 25 3.1.2.2. Get the POST request body from the file system.... 26 3.2. Access HTTPS server.... ....27 3.2.1. Send HTTPS GET request and read Fetch Response Information.... ....27 3.2.2. Send HTTPS POST request and read response information.... ....29 3.2.2.1. From UART/ USB get POST request body.... 29 3.2.2.2. Get the POST request body from the file system.... 31

4 Handling of common problems.... **....33** 4.1. HTTP(S) AT command execution fail.... .....33 4.2. PDP Activation Failed.... 33

4.3. DNS resolution failed..... ....33 4.4. Failed to enter data mode..... ....34 4.5. Failed to send GET/POST request..... ....34 4.6. Failed to read response information..... 34

5 error codes 35

6 HTTP(S) response error codes..... 37

7 Appendix Reference Documents and Terminology Abbreviations 38

table index

Table 1: Applicable Modules..... .... 7 Table 2: AT command type..... .... 10 Table 3: List of Error Codes..... .... 35 Table 4: HTTP(S) Response Code List..... .... 37 Table 5: Reference Documents..... .... 38 Table 6: Terminology abbreviations..... 38

1 Introduction

Quectel LTE Standard(A) series modules provide HTTP(S) applications for HTTP(S) servers.

HTTP (HyperText Transfer Protocol, Hypertext Transfer Protocol) is an application layer protocol for distributed, collaborative and hypermedia information systems.

HTTPS (HyperText Transfer Protocol Secure) is a transmission protocol for secure communication over a computer network. HTTPS communicates over HTTP, but utilizes SSL/TLS to encrypt packets. The main purpose of HTTPS development is to provide identity authentication to web servers and protect the privacy and integrity of exchanged data.

This document mainly introduces AT commands related to HTTP(S).

1.1. Applicable modules

Table 1: Applicable modules

Module series	module
LTE Standard(A)	EC200A series
	EC200M-CN
	EC200N-CN
	EC200S series
	EC600M-CN
	EC600N-CN
	EC600S-CN
	EC800M-CN
	EC800N-CN
	EG912N-EN
	EG912Y-EU
	EG915N series

1.2. HTTP(S) command usage process

Through the TCP/IP AT command of the LTE Standard(A) series module, you can configure the PDP context, activate or deactivate the context and query the PDP context status. Through the HTTP(S) AT command of the LTE Standard(A) series module, HTTP(S) GET/POST requests can be sent to the HTTP(S) server, and the response results from the HTTP(S) server can be read. The general process is as follows:

The first step is to use **AT+QICSGP** to configure <APN>, <username>, <password> and other parameters of the PDP **context. For details ,** please refer to document ***[1].***

The second step, after activating the PDP context through **AT+QIACT ,** use **AT+QIACT? to query the allocated** IP address, refer to document ***[1].***

The third step is to configure PDP and SSL context ID through **AT+QHTTPCFG .**

The fourth step is to configure the SSL context parameters through **AT+QSSLCFG ,** refer todocument ***[2].***

The fifth step is to set HTTP(S) URL by **AT+QHTTPURL .**

The sixth step is to send **HTTP(S) request. AT+QHTTPGET** is used to send HTTP(S) GET **request, AT+QHTTPGETEX** is used to send HTTP(S) range GET **request. AT+QHTTPPOST** or **AT+QHTTPPOSTFILE** is used to send HTTP(S) POST request.

The seventh step is to read the HTTP(S) response information through **AT+QHTTPREAD** or **AT+QHTTPREADFILE .**

The eighth step, deactivate the PDP context by **AT+QIDEACT ,** refer to document ***[1].***

1.3. Description of HTTP(S) request header information

1.3.1. Custom **HTTP(S) request header information**

The module automatically fills in HTTP(S) request header information, users can **configure <request_header> to** 1 through **AT+QHTTPCFG** Define HTTP(S) request header information, but must follow the following standards:

ÿ Follow the HTTP(S) request header statement specification.

ÿ The URI value and Host: request header information in the HTTP(S) request line must be consistent with the URL configured by **AT+QHTTPURL .** ÿ HTTP(S) request header information must end with <CR><LF>.

The following is an example of standard HTTP(S) POST request header information:

POST /processorder.php HTTP/1.1<CR><LF> Host:
220.180.239.212:8011<CR><LF>
Accept: */*<CR><LF>
User-Agent: QUECTEL_MODULE<CR><LF>

Connection: Keep-Alive<CR><LF>

Content-Type: application/x-www-form-urlencoded<CR><LF>

Content-Length: 48<CR><LF>

<CR><LF>

Message=1111&Appleqty=2222&Orangeqty=3333&find=1

1.3.2. Output **HTTP(S) response header information**

The module does not automatically output HTTP(S) response header information, you can **configure <response_header> to** 1 through **AT+QHTTPCFG** to get HTTP(S) response header information, and then execute **AT+QHTTPREAD** or **AT+QHTTPREADFILE, HTTP(S) response header information Will be** output as HTTP(S) response body.

1.4. Data mode description

The COM port of the LTE Standard(A) series module has two working modes, one is the AT command mode, and the other is the data mode. AT order In command mode, the data input through COM port is considered as AT command; in data mode, it is considered as data.

Users can exit the data mode by +++ or DTR (AT&D1 needs to be set first). In order to prevent +++ from being sent as data, the following standards must be followed in actual operation:

ÿ +++ No other data can be input within 1 second before input. ÿ

+++ must be entered within 1 second, and no other data can be entered. ÿ +++

No other data can be input within 1 second after input.

The COM port can enter the data mode by executing **AT+QHTTPURL, AT+QHTTPPOST** and **AT+QHTTPREAD ,** but before these commands respond, if you exit the data mode through +++ or pull high DTR, the execution of these commands will be interrupted. In this case, the COM port cannot be re-entered into data mode by executing **ATO .**

2 Detailed Explanation of HTTP(S) AT Commands

2.1. AT command description

2.1.1. Definition

ÿ **<CR>** carriage

ÿ **<LF>** return. line

ÿ **<...>** break. parameter name. Angle brackets are not included in the actual

ÿ **[...]** command line. Optional parameter or optional part of the TA info response. The square brackets are not included in the actual command line. Unless otherwise specified, when an optional parameter in a configuration command is omitted, its previously set value or its default value will be used by default. ÿ Default settings for underlined parameters.

2.1.2. AT command statement

The prefix **AT** or **at** must be added at the beginning of each command line. Typing <CR> will terminate the command line. Usually, the command is followed by the form <CR><LF><response><CR><LF>'s response. In tables representing commands and responses in this document, <CR><LF> are omitted, and only commands and responses are shown.

Table **2: AT** Command Types

AT command type statement	describe
Test command **AT+<cmd>=?**	Tests for the existence of a corresponding command and returns information about the type, value, or range of its arguments.
Query command **AT+<cmd>?**	Query the current parameter value of the corresponding command.
Set command **AT+<cmd>=<p1>[,<p2>[,<p3>[...]]]** to set user-definable parameter value.	
Execute the command **AT+<cmd>**	Return specific parameter information or perform specific operations.

2.2. AT example statement

The examples in this article are only for the convenience of users to understand how to use AT commands, and do not constitute Quectel's suggestions or opinions on the terminal process design, nor does it mean that the module should be set to the state in the corresponding examples. Multiple instances of some AT commands exist without succession or continuity between the instances.

2.3. Detailed explanation of AT commands

2.3.1. AT+QHTTPCFG configure HTTP(S) server parameters

This command is used to configure HTTP(S) server parameters, including configuring PDP context ID, customizing HTTP(S) request header information, and outputting HTTP(S) response headers and query SSL settings. If only one parameter is reserved when executing the setting command, it means to query the current configuration.

AT+QHTTPCFG Configure **HTTP(S) server parameters**	
test command **AT+QHTTPCFG=?**	response **+QHTTPCFG:** "contextid", (supported <contextID> range) **+QHTTPCFG:** "requestheader", (list of supported <request_header> s) **+QHTTPCFG: "responseheader", (list of supported <response_header>** s) **+QHTTPCFG:** "sslctxid", (supported <sslctxID> range) **+QHTTPCFG:** "contenttype", (supported <content_type> range) **+QHTTPCFG:** "rspout/auto", (list of supported <auto_outrsp>) **+QHTTPCFG:** "closed/ind", (list of supported <closedind>) **+QHTTPCFG: "reqheader/add",<header_name>,<header _str>** **+QHTTPCFG: "reqheader/remove",<header_name>** **OK**
query command **AT+QHTTPCFG?**	Response **+QHTTPCFG: "contextid",<contextID>** **+QHTTPCFG: "requestheader",<request_header> +QHTTPCFG: "responseheader",<response_header> +QHTTPCFG: "sslctxid",<sslctxID> +QHTTPCFG: "contenttype",<content_type > +QHTTPCFG: "rspout/auto",<auto_outrsp> +QHTTPCFG: "closed/ ind",<closedind> +QHTTPCFG: "reqheader/ add",<add_num>[,<header_na me>:<header_str>,.. .]** **+QHTTPCFG: "reqheader/remove",<add_num>[,<header_name>,...]**

	OK
set command **AT+QHTTPCFG="contextid"[,<contextID>]**	Response If the optional parameter is omitted, the current configuration is queried: **+QHTTPCFG: "contextid",<contextID>** **OK** If an optional parameter is specified, configure the PDP context ID: **OK** or **+CME ERROR: <err>**
set command **AT+QHTTPCFG="requestheader"[,<request_header>]**	Response If the optional parameter is omitted, the current configuration is queried: **+QHTTPCFG: "requestheader",<request_header>** **OK** If an optional parameter is specified, disable or enable custom HTTP(S) request header information: **OK** or **+CME ERROR: <err>**
set command **AT+QHTTPCFG="responseheader"[, <response_header>]**	Response If the optional parameter is omitted, the current configuration is queried: **+QHTTPCFG: "response header",<response_header>** **OK** If an optional parameter is specified, disables or enables the output of HTTP(S) response headers: **OK** or **+CME ERROR: <err>**
set command **AT+QHTTPCFG="sslctxid"[,<sslctxI D>]**	Response If the optional parameter is omitted, query the current configuration: **+QHTTPCFG: "sslctxid",<sslctxID>** **OK** If optional parameter is specified, configures the SSL context ID used for HTTP(S): **OK** or **+CME ERROR: <err>**

set command **AT+QHTTPCFG="contenttype"[,<content_type>]**	Response If the optional parameter is omitted, the current configuration is queried: **+QHTTPCFG: "contenttype",<content_type>** **OK** If an optional parameter is specified, configure the data type of the HTTP(S) body: **OK** or **+CME ERROR: <err>**
set command **AT+QHTTPCFG="rspout/auto"[,<aut o_outrsp>]**	Response If the optional parameter is omitted, the current configuration is queried: **+QHTTPCFG: "rspout/auto",<auto_outrsp>** **OK** If an optional parameter is specified, disable or enable automatic output of HTTP(S) response headers: **OK** or **+CME ERROR: <err>**
set command **AT+QHTTPCFG="closed/ind"[,<clos edind>]**	Response If the optional parameter is omitted, query the current configuration: **+QHTTPCFG: "closed/ind",<closedind>** **OK** If optional parameter is specified, disable or enable reporting HTTP(S) session close URC **+QHTTPURC: "closed":** **OK** or **+CME ERROR: <err>**
set command **AT+QHTTPCFG="reqheader/ add"[,<header_name>[,<header_str>]]**	If all optional parameters are omitted in the response, query the currently added custom header: **+QHTTPCFG: "reqheader/ add",<add_num>[,<header_na me>:<header_str>,...]** **OK** If the optional parameter **<header_str>** is omitted , query **the attribute value of** the packet header <header_name>: **+QHTTPCFG: "reqheader/add",<header_name>:<header _str>**

	OK or **+CME ERROR: <err>** If an optional parameter is specified, a custom header is set: **OK** or **+CME ERROR: <err>**
set command **AT+QHTTPCFG="reqheader/remove ",<header_name>**	response **OK** or **+CME ERROR: <err>**
maximum response time	300 milliseconds
Feature Description	This command takes effect immediately; the parameter configuration is not saved.

parameter

<contextID>	Integer. PDP context ID. Range: 1~15; Default value: 1.
<request_header>	integer. Disable or enable custom HTTP(S) request headers. 0 disable 1 enable **<response_header>** integer. Disable or enable output of HTTP(S) response headers. 0 disable 1 enable
<sslctxID>	integer. SSL context ID for HTTP(S). Range: 0~5; Default value: 1. Configure SSL parameters through **AT+QSSLCFG ,** refer to ***[2].*** Integer. The data type of the HTTP(S) body. 0 "application/x-www-form-urlencoded" "text/plain" "application/octet-stream" "multipart/form-data" "application/json" "image/jpeg" Integer. Disable or enable automatic output of HTTP(S) response headers. If it is enabled, **AT+QHTTPREAD** and **AT+QHTTPREADFILE** will fail to execute. 0 disable 1 enable integer. Disable or enable reporting HTTP(S) session closed URC **+QHTTPURC: "closed".** 0 disables
<content_type>	1 enables string type. Customize the header name.
	1
	2
	3
	4
	5
<auto_outrsp>	
	–
<closedind>	
	–
<header_name>	

<header_str>	String type. Customize the header content.
<add_num>	Integer. The number of custom headers added. Default value:
<err>	0. error code. For details, please refer to No. 5 chapter.

2.3.2. AT+QHTTPURL set HTTP(S) server URL

The URL of the HTTP(S) server must start with http:// or https://, indicating access to the HTTP or HTTPS server.

AT+QHTTPURL set HTTP(S) server URL	
test command **AT+QHTTPURL=?**	response **+QHTTPURL: (supported <URL_length>** range), **(supported <timeout> range)** **OK**
Query command **AT+QHTTPURL?**	response **[+QHTTPURL: <URL>]** **OK**
set command **AT+QHTTPURL=<URL_length>[,<tim eout>]**	Response If the parameter format is correct and no HTTP(S) GET/POST request is sent: **CONNECT** TA switches to the transparent transmission mode and can enter the URL. When the total size of input data **reaches** <URL_length>, TA will switch back to command mode and report the following results: **OK** If the input time reaches **<timeout> ,** but the length of the received URL is less than **<URL_length>, TA** will switch back to command mode and report the following results: **+CME ERROR: <err>** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
maximum response time	**depends on <timeout>**
Feature Description	This command takes effect immediately; the parameter configuration is not saved.

parameter

<URL_length> integer. URL length. Range: 1~2048; unit: byte. **<timeout>**	Integer. Maximum input time for a URL. Range: 1~65535; Default value: 60; Unit: second. error code.
<err>	For details, please refer to No. 5 chapter.

2.3.3. AT+QHTTPGET send GET request to HTTP(S) server

According to **the <request_header>** configured in **AT+QHTTPCFG="requestheader"[,<request_header>] , the AT+QHTTPGET setting command has two forms. If <request_header> is** 1, after sending **AT+QHTTPGET ,** if CONNECT is output within 125 seconds , it means that the HTTP(S) server connection is successful; if CONNECT is not reported within 125 seconds, +CME **ERROR will be output: <err>.**

After sending **AT+QHTTPGET** to report **OK ,** it needs to wait for a while (refer to the maximum response time) before outputting URC **+QHTTPGET: <err>,<httprspcode>[,<content_length>].**

+QHTTPGET: In **<err>,<httprspcode>[,<content_length>] ,** report only when **<err>** is 0 **<httprspcode>. If** the HTTP(S) response header information includes **CONTENT-LENGTH, the <content_length> information will be reported.**

AT+QHTTPGET Send **GET** request to **HTTP(S) server**	
Test command **AT+QHTTPGET=?**	response **+QHTTPGET: (supported <rsptime>** range), **(supported** <data_length> range), (supported <input_time> range) **OK**
Set command **if <request_header> is** 0 (disable custom HTTP(S) request header information) **AT+QHTTPGET[=<rsptime>]**	Response If the parameter format is correct and no other errors occur: **OK** After the module receives the response from the HTTP(S) server, it will report the following URC: **+QHTTPGET: <err>,<httprspcode>[,<content_length>]** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Set the command **if <request_header> is** 1 (enable custom HTTP(S) request header information) **AT+QHTTPGET=<rsptime>,<data_length>[,<input_time>]**	Response If the HTTP(S) server connection is successful: **CONNECT** TA switches to transparent transmission mode to input HTTP(S) GET request header information. When **the total size of input data reaches <data_length>, TA** will switch back to command mode and report the following results: **OK** After the module receives the response from the HTTP(S) server, it will report the following URC: **+QHTTPGET: <err>,<httprspcode>[,<content_length>]** **If the input time reaches <input_time>, but the received data length is less than <data_length>, TA** will switch back to command mode and report the following results: **+QHTTPGET: <err>**

	If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
maximum response time	**depends on <rsptime>**
Feature Description	This command takes effect immediately; the parameter configuration is not saved.

parameter

<rsptime>	Integer. After reporting **OK ,** this parameter can be used to configure **the maximum response time of HTTP(S) GET response + QHTTPGET: <err>,<httprspcode>[,<content_length>]. Range: 1~65535; Default** value: 60; Unit: second.
<data_length>	integer. Length of HTTP(S) request information, including HTTP(S) request header information and HTTP(S) request body. Range: 1~2048; unit: byte. Integer. The maximum input time for HTTP(S) request information. Range: 1~65535;
<input_time>	Default value: 60; Unit: second.
<httprspcode>	integer. HTTP(S) reply code. For details, please refer to No. 6 chapter.
<request_header>	integer. Disable or enable custom HTTP(S) request headers. 0 disable 1 enable
<content_length>	integer. HTTP(S) response body length. Unit: byte.
<err>	error code. For details, please refer to No. 5 chapter.

2.3.4. AT+QHTTPGETEX send range GET request to HTTP(S) server

Similar to reading a file, the MCU can obtain data at a specified location and a specified length from the HTTP(S) server through **AT+QHTTPGETEX ,** and this command can only be executed when **AT+QHTTPCFG="requestheader",0 .** Afterwards, the HTTP(S) server will always respond to range GET requests with **a 206** code.

AT+QHTTPGETEX Send range **GET** request to **HTTP(S) server**	
test command **AT+QHTTPGETEX=?**	response **+QHTTPGETEX: (supported <rsptime> range), <start_position>, <read_len>** **OK**
set command **AT+QHTTPGETEX=<rsptime>,<start_position>,<read_len>**	Response If the parameter format is correct and no other errors occur: **OK** After the module receives the response from the HTTP(S) server, it will report the following URC: **+QHTTPGET: <err>,<httprspcode>[,<content_length>]**

	If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
maximum response time	**depends on <rsptime>**
Feature Description	/

parameter

<rsptime>	Integer. After reporting **OK ,** this parameter can be used to configure **the maximum response time of HTTP(S) GET response + QHTTPGET: <err>,<httprspcode>[,<content_length>]. Range: 1~65535; Default** value: 60; Unit: second.
<start_postion> integer. HTTP(S) clients need the initial location of the GET data. **<read_len>** integer. The HTTP(S) client expects the length of the GET data.	
<httprspcode> integer. HTTP(S) reply code. For details, please refer to **<content_length>** integer. HTTP(S) response body length. Unit: byte. error code. For details, please refer to	No. 6 chapter
<err>	No. 5 chapter

2.3.5. AT+QHTTPPOST send POST request to HTTP(S) server via UART/USB

This command is used to send HTTP(S) POST request to HTTP(S) server via UART/USB. According to **The <request_header>** configured in **AT+QHTTPCFG="requestheader"[,<request_header>] , the AT+QHTTPPOST setting command has two forms. If <request_header> is** 0, then input POST **request** body through UART/USB port ; if <request_header> is 1, then input POST request header information and POST request body through UART/USB port.

After sending **AT+QHTTPPOST ,** if CONNECT is returned within 125 seconds , **it means that** the HTTP(S) server is successfully connected; if CONNECT is not returned within 125 seconds, +CME **ERROR: <err> will be output.**

After reporting **OK ,** you need to wait for a while (refer to the maximum response time) before outputting URC **+QHTTPPOST: <err>,<httprspcode>[,<content_length>].**

AT+QHTTPPOST Send **POST** request to **HTTP(S) server** via **UART/USB**	
test command **AT+QHTTPPOST=?**	response **+QHTTPPOST: (supported <data_length>** range), **(supported <input_time> range), (supported <rsptime> range)** **OK**
Set command **if <request_header> is** 0 (disable custom HTTP(S) request header information) **AT+QHTTPPOST=<data_length>[,<input_time>,<rsptime>]**	If the response parameter format is correct, the HTTP(S) server connection is successful and the HTTP(S) request header information is sent: **CONNECT**

	TA switches to the transparent transmission mode, and can input the HTTP(S) POST request body. When the total size of **input** data reaches <data_length>, TA will switch back to command mode and report the following results: **OK** After the module receives the response from the HTTP(S) server, it will report the following URC: **+QHTTPPOST: <err>,<httprspcode>[,<content_length>]** **If the input time reaches <input_time>, but the received data length is less than <data_length>, TA** will switch back to command mode and report the following results: **+QHTTPPOST: <err>** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Set the command **if <request_header> is** 1 (enable custom HTTP(S) request header information) **AT+QHTTPPOST=<data_length>[,<in put_time>,<rsptime>]**	Response If the parameter format is correct and the HTTP(S) server connection is successful: **CONNECT** TA switches to transparent transmission mode to input HTTP(S) POST request body and HTTP(S) POST request header information. When the total size of input data reaches **<data_length>, TA** will switch back to command mode and report the following results: **OK** After the module receives the response from the HTTP(S) server, it will report the following URC: **+QHTTPPOST: <err>,<httprspcode>[,<content_length>]** **If the input time reaches <input_time>, but the received data length is less than <data_length>, TA** will switch back to command mode and report the following results: **+QHTTPPOST: <err>** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
maximum response time	**Depends on network and <rsptime>**
Feature Description	/

parameter

<data_length>	Integer. If **<request_header>** is 0, it means the length of POST request body; if **<request_header> is** 1, it means the length of HTTP(S) request information, including HTTP(S) POST request header information and HTTP(S) POST request body. Range: 1~1024000; unit: byte. Integer. Maximum input time for POST request body or HTTP(S)
<input_time>	request information. Range: 1~65535; Default value: 60; Unit: second.

<rsptime>	Integer. After reporting **OK ,** this parameter can be used to configure **the maximum output time of HTTP(S) POST response + QHTTPPOST: <err>,<httprspcode>[,<content_length>]. Range: 1~65535;** Default value: 60; Unit: second. Integer. HTTP(S) reply code. For details, please refer to **<httprspcode> <request_header>** integer. Disable or enable custom HTTP(S) request headers. No. 6 0 disable 1 enable integer.
<content_length>	HTTP(S) response body length. Unit: byte. error code.
<err>	For details, please refer to No. 5 chapter.

2.3.6. AT+QHTTPPOSTFILE Send POST request to HTTP(S) server through file

This command can be used to send a POST request to an HTTP(S) server via a file. According to **the <request_header> configured in AT+QHTTPCFG="requesthead r"**[,<request_header>], the file operated by AT+QHTTPPOSTFILE has two forms. If **<request_header> is** 0, the file in the file system will be the POST **request body; if <request_header> is** 1, the file in the file system will be the POST request header information and POST request body information.

After executing **AT+QHTTPPOSFILE ,** the module will report URC **+QHTTPPOSTFILE: <err>,<httprspcode>[,<con** tent_length>] indicates the command execution result. **Report <httprspcode>** only when <err> is 0 .

After reporting **OK ,** it takes a while (refer to the maximum response time) before outputting URC **+QHTTPPOSTFILE: <err>,<httprspcode>[,<content_length>] information.**

AT+QHTTPPOSTFILE Send **POST** request to **HTTP(S) server** through file	
test command **AT+QHTTPPOSTFILE=?**	response **+QHTTPPOSTFILE: <file_name>, (supported <rsptime> range), (supported <post_mode> range)** **OK**
Set command **AT+QHTTPPOSTFILE=<file_name>[, <rsptime>,<post_mode>] (If <request_header> is** 1, the specified file must contain HTTP(S) request header information.)	Response If the parameter format is correct and the HTTP(S) server connection is successful: **OK** When the module receives the response from the HTTP(S) server: **+QHTTPPOSTFILE: <err>,<httprspcode>[,<content_length>]** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
maximum response time	**depends on <rsptime>**
Feature Description	This command takes effect immediately; the parameter configuration is not saved.

parameter	
<file_name>	String type. file name. The maximum length is 80 bytes.
<rsptime>	Integer. After reporting **OK ,** this parameter can be used to configure **the maximum output time of HTTP(S) POST response + QHTTPPOST FILE: <err>,<httprspcode>[,<content_length>].** **Range: 1** ~65535; Default value: 60; Unit: second. Integer. HTTP(S) reply code. For details, please
headers. 0 disable 1 enable integer. HTTP(S) **<httprspcode>** **<request_header>** integer. Disable or enable sending HTTP(S) request	
file content directly 1 Record and save the file, do not send it temporarily, wait to send it together with the file configured when <post_mode>=2 2 Send the file, send it together with the file saved when **<post_mode>=1** (only two files are supported together send) error code. For details, please refer to	
<content_length>	
<post_mode>	
<err>	

2.3.7. **AT+QHTTPREAD** read HTTP(S) server response information via **UART/USB**

After sending the HTTP(S) GET/POST request, you can use **AT+QHTTPREAD** to read the HTTP(S) response information from the HTTP(S) server through the UART/USB port. Must receive +QHTTPGET: **<err>,<httprspcode>[,<content_length>], +QHTTPPOST: <err>,<httprspcode>[,<content_length>] or +QHTTPPOSTFILE: <err>,<httprspcode>[,< content_length>] information to execute AT+QHTTPREAD.**

AT+QHTTPREAD read HTTP(S) server response information via **UART/USB**	
test command **AT+QHTTPREAD=?**	response **+QHTTPREAD: (supported <wait_time> range)** **OK**
set command **AT+QHTTPREAD[=<wait_time>]**	Response If the parameter format is correct: **CONNECT** <output **HTTP(S) response information>** **OK** When the reading of the response information is completed or the interval between receiving two data packets reaches **<wait_time>:** **+QHTTPREAD: <err>** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**

maximum response time	**depends on <wait_time>**
Feature Description	/

parameter

<wait_time> integer. The maximum interval between receiving two packets. Range: 1~65535; Default value: 60; Unit: second.	
<err>	error code. For details, please refer toNo. 5 chapter.

2.3.8. AT+QHTTPREADFILE read HTTP(S) server response information through file

After sending the HTTP(S) GET/POST request, you can use **AT+QHTTPREADFILE** to read the HTTP(S) response information from the HTTP(S) server through the file, you must receive +QHTTPGET: **<err>,<httprspcode>[,<content_length>], +QHTTPPO ST: <err>,<httprspcode>[,<content_length>] or +QHTTPPOSTFILE: <err>,<httprspcode>[,<content_length>] information before executing AT+QHTTPREADFILE.**

AT+QHTTPREADFILE read HTTP(S) server response information through file	
test command **AT+QHTTPREADFILE=?**	response **+QHTTPREADFILE:** <file_name>, (supported <wait_time> range) **OK**
set command **AT+QHTTPREADFILE=<file_name>[, <wait_time>]**	Response If the parameter format is correct: **OK** When the body reading is completed or the interval between receiving two data packets reaches **<wait_time>: +QHTTPREADFILE: <err>** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
maximum response time	**depends on <wait_time>**
Feature Description	/

parameter

<wait_time>	Integer. The maximum interval between receiving two packets. Range: 1~65535; Default value: 60; Unit: second. String type. file name. The maximum length is 80 bytes. error code. For details, please refer to
<file_name>	
<err>	No. 5 chapter.

2.3.9. AT+QHTTPSTOP cancel HTTP(S) request

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

AT+QHTTPSTOP cancel HTTP(S) request	
test command **AT+QHTTPSTOP=?**	response **OK**
Excuting an order **AT+QHTTPSTOP**	Response If the parameter format is correct and no other errors occur: **OK** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
maximum response time	10 seconds
Feature Description	/

parameter

<err>	error code. For details, please refer toNo. 5 chapter.

3 example

3.1. Access HTTP server

3.1.1. Send HTTP GET request and read response information

The following example shows how to send an HTTP GET request, enable the output of HTTP response header information, and read the HTTP GET response.

```
//Send an example HTTP GET response.
AT+QHTTPCFG="contextid",1 OK                    //Configure the PDP context ID as 1.
AT+QHTTPCFG="responseheader",1 //
Enable output HTTP response header information.
OK
AT+QIACT?                                       //Query PDP context status.
OK
AT+QICSGP=1,1,"UNINET","","",1 //Configure PDP context as 1, APN as "UNINET" of China Unicom. Need to set
                                                AT+CFUN=1,1 makes the configuration take effect.
OK
AT+QIACT?                                       //Query PDP context status.
+QIACT: 1,1,1,"10.7.157.1"

OK
//The first PDP is activated by default. If the query shows that it is not activated, it can be activated by executing
AT+QIACT=1 //                                   AT+QIACT=1 . //Activate PDP context 1.
OK          Activation is successful. //Set the URL to be accessed, and set
AT+QHTTPURL=23,80                               the timeout to 80 seconds.
CONNECT
http://www.sina.com.cn/                         //Enter a URL with a length of 23 bytes. (This URL is just an example. Please enter the correct
                                                URL according to the actual situation.)
OK
AT+QHTTPGET=80                                  //Send an HTTP GET request with a maximum response time of 80 seconds.
OK

+QHTTPGET: 0,200,601710                         //If the HTTP response header information includes CONTENT-LENGTH, the
                                                <content_length> information will be reported.
//Read an example of HTTP GET response information.

//Method 1: Read HTTP response information and output it through UART port.
```

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

                                         //The response information is omitted here.
<CR><LF>
<body>
OK

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

//Method 2: Read the HTTP response information and store it in the UFS file.

```
AT+QHTTPREADFILE="UFS:1.txt",80          //Read the HTTP response header information and response body and store them in UFS:1.txt.
                                         The maximum wait time for an HTTP session to close is 80 seconds.
OK

+QHTTPREADFILE: 0                        //Successfully store HTTP response header information and response body.
```

3.1.2. Send **HTTP POST** request and read response information

3.1.2.1. Get **POST** request body via **UART/USB**

The following examples illustrate how to send HTTP POST request, read POST request body and how to read HTTP POST response information through UART port.

```
AT+QHTTPCFG="contextid",1                //Configure the PDP context ID as 1.
OK
AT+QIACT?                                //Query PDP context status.
OK
AT+QICSGP=1,1,"UNINET","","",1           //Configure PDP context as 1 and APN as "UNINET" of China Unicom. Need to set
                                         AT+CFUN=1,1 makes the configuration take effect.
OK
```

```
AT+QIACT?                                   //Query PDP context status.
+QIACT: 1,1,1,"172.22.86.226"

OK
//The first PDP is activated by default. If the query shows that it is not activated, it can be activated by executing
AT+QIACT=1 //                               AT+QIACT=1 . //Activate PDP context 1.
OK                  Activation is successful. //Set the URL to be accessed, and set
AT+QHTTPURL=59,80                           the timeout to 80 seconds.
CONNECT
http://api.efxnow.com/DEMOWebServices2.8/Service.asmx/Echo? //Enter a URL with a length of 59 bytes. (This URL is just an
                                                                    example, please enter the correct URL
                                                                    according to the actual situation.)
OK
AT+QHTTPPOST=20,80,80                       //Send HTTP POST request, get POST request body through UART. The maximum input time
                                            and response time for the POST request body are both 80 seconds.
CONNECT
Message=Hello Quectel                       //Enter a POST request body with a length of 20 bytes. (This POST request body is just an
                                            example, please enter the correct POST request body according to the actual situation.)
OK

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

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

3.1.2.2. Get the POST request body from the file system

The following example illustrates how to send an HTTP POST request, read the POST request body through the file system, and store the HTTP POST response to the file system.

```
AT+QHTTPCFG="contextid",1                   //Configure the PDP context ID as 1.
OK
AT+QIACT?                                   //Query PDP context status.
OK
AT+QICSGP=1,1,"UNINET","","",1              //Configure PDP context as 1 and APN as "UNINET" of China Unicom. Need to set
                                            AT+CFUN=1,1 makes the configuration take effect.
OK
```

```
AT+QIACT?                                   //Query PDP context status.
+QIACT: 1,1,1,"172.22.86.226"

OK
//The first PDP is activated by default. If the query shows that it is not activated, it can be activated by executing
AT+QIACT=1 //                               AT+QIACT=1 . //Activate PDP context 1.
OK                  Activation is successful. //Set the URL to be accessed, and set
AT+QHTTPURL=59,80                           the timeout to 80 seconds.
CONNECT
http://api.efxnow.com/DEMOWebServices2.8/Service.asmx/Echo? //Enter a URL with a length of 59 bytes. (This URL is just an
                                                                    example, please enter the correct URL
                                                                    according to the actual situation.)
OK //
POST request information from UFS file, read HTTP response information and store it in UFS file.
AT+QHTTPPOSTFILE="UFS:2.txt",80 //Send HTTP POST request, get POST request body from UFS:2.txt ,
                                                    The maximum response time is 80 seconds.
OK

+QHTTPPOSTFILE: 0,200,177                   // Only after the HTTP POST request is sent successfully, it can be executed
                                            AT+QHTTPREADFILE.
AT+QHTTPREADFILE="UFS:3.txt",80 //Read HTTP response information and save it to UFS:3.txt. The maximum wait time for an
                                            HTTP session to close is 80 seconds.
OK

+QHTTPREADFILE: 0                           //HTTP response information stored successfully.
```

3.2. Access **HTTPS** server

3.2.1. Send **HTTPS GET** request and read response information

The following example shows how to send HTTPS GET request, enable HTTPS response header information output and how to read HTTPS GET response information.

```
//Example of sending HTTPS GET request
AT+QHTTPCFG="contextid",1                   //Configure the PDP context ID as 1.
OK
AT+QHTTPCFG="responseheader",1 //Enable output HTTPS response header information.
OK
AT+QIACT?                                   //Query PDP context status.
OK
AT+QICSGP=1,1,"UNINET","","",1              //Configure PDP context as 1 and APN as "UNINET" of China Unicom. Need to set
                                            AT+CFUN=1,1 to make the configuration take effect.
OK
```

```
AT+QIACT?                                   //Query PDP context status.
+QIACT: 1,1,1,"10.7.157.1"

OK
//The first PDP is activated by default. If the query shows that it is not activated, it can be activated by executing
AT+QIACT=1 //                               AT+QIACT=1 . //Activate PDP context 1.
OK                  Activation is successful. //Set the SSL context ID to 1.
AT+QHTTPCFG="sslctxid",1
OK
AT+QSSLCFG="sslversion",1,1                 //Set the SSL version to 1, which means TLSV1.0.
OK
AT+QSSLCFG="ciphersuite",1,0x0005 //Set the SSL cipher suite to 0x0005, which means RC4-SHA.
OK
AT+QSSLCFG="seclevel",1,0                   //Set the SSL verification level to 0, indicating no authentication mode.
OK
AT+QHTTPURL=22,80                           //Set the URL to be accessed, and set the timeout to 80 seconds.
CONNECT
https://www.alipay.com                      //Enter a URL with a length of 22 bytes. (This URL is just an example, please enter the correct
                                            URL according to the actual situation.)
OK
AT+QHTTPGET=80                              // Send HTTPS GET request, the maximum response time is 80 seconds.
OK

+QHTTPGET: 0,200,21472                      //If the HTTPS response header information contains CONTENT-LENGTH, the
                                            <content_length> information will be reported.
//Example of reading HTTPS response information.

//Method 1: Read HTTPS response information and output it through UART.

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

                                            //The response information is omitted here.
<CR><LF>
<body>
OK

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

```
//Method 2: Read the HTTPS response information and store it in a UFS file.
AT+QHTTPREADFILE="UFS:4.txt",80 //Read HTTPS response header information and response body and store them in UFS:4.txt.
                                The maximum wait time for an HTTP session to close is 80 seconds.
OK

+QHTTPREADFILE: 0               //HTTPS response header information and response body are stored successfully.
```

3.2.2. Send **HTTPS POST** request and read response information

3.2.2.1. Get **POST** request body from **UART/USB**

The following example illustrates how to send HTTPS POST request, read POST request body and how to read HTTPS POST response information through UART.

```
AT+QHTTPCFG="contextid",1                   //Configure the PDP context ID as 1.
OK
AT+QIACT?                                   //Query PDP context status.
OK
AT+QICSGP=1,1,"UNINET","","",1              //Configure PDP context as 1 and APN as "UNINET" of China Unicom. Need to set
                                            AT+CFUN=1,1 to make the configuration take effect.
OK
AT+QIACT?                                   //Query PDP context status.
+QIACT: 1,1,1,"172.22.86.226"

OK
//The first PDP is activated by default. If the query shows that it is not activated, it can be activated by executing AT+QIACT=1 .
AT+QIACT=1                                  //Activate PDP context 1. //
OK                                          Activation succeeded.
AT+QHTTPCFG="sslctxid",1                    //Set the SSL context ID to 1.
OK
AT+QSSLCFG="sslversion",1,1                 //Set the SSL version to 1, which means TLSV1.0.
OK
AT+QSSLCFG="ciphersuite",1,0x0005 //Set the SSL cipher suite to 0x0005, which means RC4-SHA.
OK
AT+QSSLCFG="seclevel",1,2                   //Set the SSL verification level to 2, indicating that the CA certificate, client certificate and
                                            client private key need to be uploaded through AT+QFUPL .
OK
AT+QFUPL="cacert.pem"                       //Upload the CA certificate to UFS.
CONNECT
<Input file bin data>
+QFUPL:1216,7648
```

```
OK

AT+QFUPL="clientcert.pem"                    //Upload the client certificate to UFS.
CONNECT
<Input file bin data>
+QFUPL:1216,5558

OK

AT+QFUPL="clientkey.pem"                     //Upload the client private key to UFS.
CONNECT
<Input file bin data>
+QFUPL:1706,538

OK

AT+QSSLCFG="cacert",1,"UFS:cacert.pem"
OK
AT+QSSLCFG="clientcert",1,"UFS:clientcert.pem"
OK
AT+QSSLCFG="clientkey",1,"UFS:clientkey.pem"
OK
AT+QHTTPURL=45,80                            //Set the URL to be accessed, and set the timeout to 80 seconds.
CONNECT HTTPs://
220.180.239.212:8011/processorder.php //Enter a URL with a length of 45 bytes. (This URL is an example only,
                                             Please enter the correct URL according to the actual situation. )
OK
AT+QHTTPPOST=48,80,80                        //Send HTTPS POST request, get POST request body through UART. The maximum input
                                             time and response time for a POST request body is 80 seconds.
CONNECT
Message=1111&Appleqty=2222&Orangeqty=3333&find=1 //Enter the POST request body with a length of 48 bytes. (This POST request
                                                 body is just an example, please enter the correct POST
                                                 request body according to the actual situation.)
OK

+QHTTPPOST: 0,200,285                        //If the HTTPS response header information contains CONTENT-LENGTH, the
                                             <content_length> information will be reported. //Read HTTPS response information and
AT+QHTTPREAD=80                              output it through UART. The maximum wait time for an HTTP session to close is 80 seconds.

CONNECT                                      //HTTPS response information read successfully.
<html>
<head>
<title>Quectel's Auto Parts - Order Results</title>
</head>
```

```
<body>
<h1>Quectel's Auto Parts</h1>
<h2>Order Results</h2>

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

OK

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

3.2.2.2. Get the POST request body from the file system

The following example illustrates how to send an HTTPS POST request, read the POST request body from the file system, and how to send the HTTPS POST request Response information is stored to the file system.

```
AT+QHTTPCFG="contextid",1                        //Configure the PDP context ID as 1.
OK
AT+QIACT?                                        //Query PDP context status.
OK
AT+QICSGP=1,1,"UNINET","","",1 //Configure PDP context as 1, APN as "UNINET" of China Unicom. Need to set
                                                 AT+CFUN=1,1 makes the configuration take effect.
OK
AT+QIACT?                                        //Query PDP context status.
+QIACT: 1,1,1,"172.22.86.226"

OK
//The first PDP is activated by default. If the query shows that it is not activated, it can be activated by executing
AT+QIACT=1 //                                     AT+QIACT=1 . //Activate PDP context 1.
OK              Activation is successful. //Set the SSL context ID to 1.
AT+QHTTPCFG="sslctxid",1
OK
AT+QSSLCFG="sslversion",1,1                      //Set the SSL version to 1, which means TLSV1.0.
OK
AT+QSSLCFG="ciphersuite",1,0x0005 //Set the SSL cipher suite to 0x0005, which means RC4-SHA.
OK
AT+QSSLCFG="seclevel",1,2                        //Set the SSL verification level to 2, indicating that the CA certificate, client certificate and client
                                                 private key need to be uploaded through AT+QFUPL .
OK
AT+QFUPL="cacert.pem"                            //Upload the CA certificate to UFS.
CONNECT
<Input file bin data>
+QFUPL:1216,7648
OK
```

```
AT+QFUPL="clientcert.pem"                    //Upload the client certificate to UFS.
CONNECT
<Input file bin data>
+QFUPL:1216,5558
OK

AT+QFUPL="clientkey.pem"                     //Upload the client private key to UFS.
CONNECT
<Input file bin data>
+QFUPL:1706,538
OK

AT+QSSLCFG="cacert",1,"UFS:cacert.pem"
OK
AT+QSSLCFG="clientcert",1,"UFS:clientcert.pem"
OK
AT+QSSLCFG="clientkey",1,"UFS:clientkey.pem"
OK
AT+QHTTPURL=45,80                            //Set the URL to be accessed, and set the timeout to 80 seconds.
CONNECT https://
220.180.239.212:8011/processorder.php //Enter a URL with a length of 45 bytes. (This URL is just an example, please enter the
                                             correct URL according to the actual situation.)
OK //
POST request information comes from UFS file, read HTTPS response information and store it in UFS file.
AT+QHTTPPOSTFILE="UFS:5.txt",80 //Send HTTPS POST request, get POST request body from UFS:5.txt ,
                                             The maximum response time is 80 seconds.
OK

+QHTTPPOSTFILE: 0,200,177                    // Only after the HTTPS POST request is sent successfully, it can be executed
                                             AT+QHTTPREADFILE.
AT+QHTTPREADFILE="UFS:6.txt",80 //Read HTTPS response information and store it in UFS:6.txt". The maximum waiting time for
                                             HTTP session closing is 80 seconds.
OK

+QHTTPREADFILE: 0                            //HTTPS response information stored successfully.
```

4Common problem handling

4.1. HTTP(S) AT command execution failed

After executing the HTTP(S) AT command, if +CME **ERROR:** <err> is returned, please check whether the (U)SIM card is inserted, and pay attention to execute Whether +CPIN: **READY** is returned after AT+CPIN?.

4.2. PDP activation failed

If using **AT+QIACT** to activate PDP fails, please check the following configurations:

1. Query whether the PS domain **is attached through AT+CGATT?.** If the PS domain is not attached, please execute **AT+CGATT=1** to attach the PS domain first. 2. **Query** the PS domain status through AT+CGREG? and make sure the PS domain has been registered. 3. Query the PDP context parameters through **AT+QICSGP ,** and make sure that the APN of the specified PDP context has been set. 4. Make sure that the specified PDP context ID is neither used by PPP nor activated by **AT+CGACT .** 5. According to the 3GPP specification, the module only supports to activate three PDP contexts at the same time, so the number of activated PDP contexts must be less than

3.

If the above configuration is correct, the activation of the PDP context using **AT+QIACT** still fails, please restart the module. After restarting the module, check at least Three times of the above-mentioned configurations, with an interval of 10 minutes each time, to avoid frequent restarts of the module.

4.3. DNS resolution failed

After executing **AT+QHTTPGET, AT+QHTTPPOST** and **AT+QHTTPPOSTFILE ,** if +CME **ERROR: 714** is returned (714: HTTP(S) DNS error), please check the following configuration:

1. Confirm that the HTTP(S) server domain name is valid. 2. **Query** the PDP context status through AT+QIACT? to ensure that the specified PDP context is successfully activated. 3. Query the DNS server address by **AT+QIDNSCFG ,** make sure the DNS **server address is not "0.0.0.0".**

If **the address of the DNS server is "0.0.0.0", there are two solutions:**

1. Reassign a valid DNS server address via **AT+QIDNSCFG .** 2. Deactivate the PDP context by **AT+QIDEACT ,** and reactivate the PDP context by **AT+QIACT .**

4.4. Failed to enter data mode

After executing **AT+QHTTPURL, AT+QHTTPGET, AT+QHTTPPOST** and **AT+QHTTPREAD , if it returns +CME ERROR:** 704 (704: HTTP(S) UART is busy), please check whether there are other ports in the data mode, because the module only supports One port is in data mode, if there are other ports, please exit the data mode of other ports and execute these commands again.

4.5. GET/POST request failed to send

If execution of **AT+QHTTPGET, AT+QHTTPGETEX, AT+QHTTPPOST** and **AT+QHTTPPOSTFILE** fails, please check the following configurations:

1. Make sure the URL entered via AT+QHTTPURL is valid and accessible. 2. Make sure the specified server supports GET/POST related commands. 3. Make sure the PDP context is activated.

If the above configuration is correct, sending GET/POST request through AT+QHTTPGET, AT+QHTTPPOST and AT+QHTTPPOSTFILE still fails, please deactivate the PDP context through **AT+QIDEACT ,** and reactivate the PDP context through **AT+QIACT .** If activation of the PDP context fails, please refer to Solve this problem for details. No. ***4.2*** chapter

4.6. Failed to read response information

Before reading the response information through **AT+QHTTPREAD** and **AT+QHTTPREADFILE ,** first execute **AT+QHTTPGET, AT+QHTTPPOST** and **AT+QHTTPPOSTFILE, after which the following** URC will be reported:

+QHTTPGET: <err>,<httprspcode>[,<content_length>]
+QHTTPPOST: <err>,<httprspcode>[,<content_length>]
+QHTTPPOSTFILE: <err>,<httprspcode>[,<content_length>]

When executing AT + **QHTTTPREAD** and **AT+QHTTPREADFILE ,** if an error occurs , such as reporting +CME **ERROR :** 717 (717: HTTP(S) socket read error), please send HTTP(S) The server resends the HTTP(S) GET/POST request; if the request still fails, please refer to Solve this problem for details. No. ***4.5*** chapter

5 error codes

<err> Indicates an error code related to the mobile device or the network. See the table below for details.

Table 3: List of Error Codes

<err> error code	English description	Chinese description
0	Operation successful	Successful operation
701	HTTP(S) unknown error	HTTP(S) unknown error
702	HTTP(S) timeout	HTTP(S) timeout
703	HTTP(S) busy	HTTP(S) busy
704	HTTP(S) UART busy	HTTP(S) UART busy
705	HTTP(S) no GET/POST requests	HTTP(S) no GET/POST request
706	HTTP(S) network busy	HTTP(S) network is busy
707	HTTP(S) network open failed	HTTP(S) network connection failed
708	HTTP(S) network no configuration	HTTP(S) network not configured
709	HTTP(S) network deactivated	Deactivate HTTP(S) network
710	HTTP(S) network error	HTTP(S) network error
711	HTTP(S) URL error	HTTP(S) URL error
712	HTTP(S) empty URL	HTTP(S) Empty URL
713	HTTP(S) IP address error	HTTP(S) IP address error
714	HTTP(S) DNS error	HTTP(S) DNS error
715	HTTP(S) socket create error	HTTP(S) socket creation error
716	HTTP(S) socket connect error	HTTP(S) socket connection error
717	HTTP(S) socket read error	HTTP(S) socket read error

718	HTTP(S) socket write error	HTTP(S) socket write error
719	HTTP(S) socket closed	HTTP(S) socket closed
720	HTTP(S) data encode error	HTTP(S) data encoding error
721	HTTP(S) data decode error	HTTP(S) data decoding error
722	HTTP(S) read timeout	HTTP(S) read timeout
723	HTTP(S) response failed	HTTP(S) response failed
724	Incoming call busy	call busy
725	Voice call busy	voice call busy
726	Input timeout	input timeout
727	Wait data timeout	data wait timeout
728	Wait HTTP(S) response timeout	HTTP(S) timeout waiting for response
729	Memory allocation failed	Insufficient memory allocation
730	Invalid parameter	invalid parameter

6 HTTP(S) response error codes

<httprspcode> indicates the HTTP(S) server response code. See the table below for details.

Table **4: List of HTTP(S) Response Codes**

<httprspcode> response code	English description	Chinese description
200	OK	success
403	Forbidden	prohibit
404	not found	Not found
409	Conflict	conflict
411	Length required	Need to enter the length
500	Internal server error	internal server error

7 Appendix Reference Documents and Terminology Abbreviations

Table 5: Reference Documents

file name
[1] Quectel_LTE_Standard(A) Series_TCP(IP)_Application Guide
[2] Quectel_LTE_Standard(A) Series_SSL_Application Guide

Table 6: Terminology Abbreviations

abbreviation	English full name	Chinese full name
DNS	Domain Name Server	domain name server
DTR	Data Terminal Ready	data terminal ready
HTTP	Hypertext Transfer Protocol	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure hypertext transfer security protocol	
ID	Identification	Identification
PDPs	Packet Data Protocol	packet data protocol
PPP	Point-to-Point Protocol	peer-to-peer protocol
P.S.	Packet Switch	packet switching
SSL	Security Socket Layer	Secure Socket Layer
UART	Universal Asynchronous Receiver/Transmitter	Universal Asynchronous Transmitter
URIs	Uniform Resource Identifier	Uniform Resource Identifier
URL	Uniform Resource Locator	Uniform Resource Locator
URC	Unsolicited Result Code	unsolicited result code

USB	Universal Serial Bus	Universal Serial Bus
(U)SIM	(Universal) Subscriber Identity Module	(Universal) Subscriber Identity Module
UFS	UNIX File System	UNIX file system