RG50xQ&RM5xxQ Series
HTTP(S) Application Note

5G Module Series

Version: 1.0

Date: 2024-06-12

Status: Released

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

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

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

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

Legal Notices

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

Use and Disclosure Restrictions

License Agreements

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

Copyright

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

Trademarks

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

Third-Party Rights

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

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

Privacy Policy

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

Disclaimer

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

About the Document

Revision History

Version	Date	Author	Description
-	2024-04-11	Yaalon PAN	Creation of the document
1.0	2024-06-12	Yaalon PAN	First official release

Contents

Table Index

1 Introduction

Quectel 5G RG50xQ family and RM5xxQ family modules support HTTP(S) applications by accessing HTTP(S) servers.

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

HTTPS (Hypertext Transfer Protocol Secure) is a variant of the standard web transfer protocol (HTTP) that adds a layer of security on the data in transit through a secure socket layer (SSL) or transport layer security (TLS) protocol connection. The main purpose of HTTPS development is to provide identity authentication for website servers and protect the privacy and integrity of exchanged data.

This document is a reference guide to all the AT commands defined for HTTP(S).

Table 1: Applicable Modules

Module Family	Module
RG50xQ	RG500Q Series
	RG501Q-EU
	RG502Q Series
RM5xxQ	RM500Q Series
	RM502Q-AE
	RM505Q-AE
	RM510Q-GL

1.1. Process of Using HTTP(S) AT Commands

By TCP/IP AT commands you can configure a PDP context, activate/deactivate the PDP context, and query the context status. Whereas, by HTTP(S) AT commands you can send HTTP(S) GET/POST/PUT requests to the HTTP(S) server and read the response from the HTTP(S) server. In general, the process

is as follows:

Step 1: Configure **<APN>**, **<username>**, **<password>** and other parameters of a PDP context by **AT+QICSGP**, and update optional QoS settings by using **AT+CGQMIN** and **AT+CGQREQ**. For more details, see ***document [1]***.

Step 2: Activate the PDP context by **AT+QIACT**, then the assigned IP address can be queried by **AT+QIACT?**. For more details, see ***document [1]***.

Step 3: Configure the PDP context ID and SSL context ID by **AT+QHTTPCFG**.

Step 4: Configure SSL context parameters by **AT+QSSLCFG**. For more details, see ***document [2]***.

Step 5: Set HTTP(S) URL by **AT+QHTTPURL**.

Step 6: Send HTTP(S) request.

- **AT+QHTTPGET** can be used for sending HTTP(S) GET request.
- **AT+QHTTPGETEX** can be used for sending to HTTP(S) GET request within specified range.
- **AT+QHTTPPOST** or **AT+QHTTPPOSTFILE** can be used for sending HTTP(S) POST request.
- **AT+QHTTPPUT** or **AT+QHTTPPUTFILE** can be used for sending an HTTP(S) PUT request.

Step 7: Read HTTP(S) response information by **AT+QHTTPREAD** or **AT+QHTTPREADFILE**.

Step 8: Deactivate the PDP context by **AT+QIDEACT**. For more details, See ***document [1]***.

1.2. Description of HTTP(S) Request Header

1.2.1. Customize HTTP(S) Request Header

HTTP(S) request header is filled by the module automatically. Set **<request_header>** to 1 via **AT+QHTTPCFG** to customize HTTP(S) request header, and then input HTTP(S) request header according to the following requirements:

- Follow HTTP(S) request header syntax.
- The URI in HTTP(S) POST request header must be in line with the URL configured by **AT+QHTTPURL**.
- The HTTP(S) request header must end with **<CR><LF>**.

The following example shows a valid HTTP(S) POST request header:

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>

1.2.2. Output HTTP(S) Response Header

HTTP(S) response header is not outputted automatically by the module. HTTP(S) response header can be outputted by setting **<response_header>** as 1 in **AT+QHTTPCFG**, and then HTTP(S) response header is outputted with the format of HTTP(S) response body after executing **AT+QHTTPREAD** or **AT+QHTTPREADFILE**.

1.3. Description of Data Mode

The COM port of the module has two working modes: AT command mode and data mode. In AT command mode, the data inputted via the COM port are treated as AT commands, while they are treated as data in data mode.

- **Enter Data Mode**

To enter the data mode, execute **AT+QHTTPURL**, **AT+QHTTPPOST**, **AT+QHTTPGET**, **AT+QHTTPPUT** or **AT+QHTTPREAD**. If you input **+++** or pull up the DTR pin to make the port exit data mode before the response is returned, the execution of these commands will be interrupted. In such a case, the COM port cannot re-enter data mode if you execute **ATO**.

- **Exit Data Mode**

Inputting **+++** or pulling up the DTR pin can make the COM port exit data mode. To prevent **+++** from being misinterpreted as data, the following sequence should be followed:

1) Do not input any character within 1 s before and after inputting **+++**.
2) Input **+++** within 1 s, and wait until **OK** is returned. When **OK** is returned, COM port exits the data mode.

If you are exiting the data mode by pulling up DTR pin, make sure to set **AT&D1** first.

2 Description of HTTP(S) AT Commands

2.1. AT Command Introduction

2.1.1. Definitions

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

2.1.2. AT Command Syntax

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

Table 2: Type of AT Commands

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

2.2. Declaration of AT Command Examples

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

2.3. AT Command Description

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

The command configures the parameters for HTTP(S) server, including configuring a PDP context ID, customizing HTTP(S) request header, outputting HTTP(S) response header and configuring SSL context ID. If optional parameters are omitted while executing Write Command, it will query the current settings.

AT+QHTTPCFG Configure Parameters for HTTP(S) Server	
Test Command **AT+QHTTPCFG=?**	Response **+QHTTPCFG: "contextid",(**range of supported **<contextID>**s**)** **+QHTTPCFG: "requestheader",(**list of supported **<request_header>**s**)** **+QHTTPCFG: "responseheader",(**list of supported **<response_header>**s**)** **+QHTTPCFG: "sslctxid",(**range of supported **<sslctxID>**s**)** **+QHTTPCFG: "contenttype",(**range of supported **<content_type>**s**)** **+QHTTPCFG: "rspout/auto",(**list of supported **<auto_outrsp>**s**)** **+QHTTPCFG: "closed/ind",(**list of supported **<closedind>**s**)** **OK**
Read 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>** **OK**

Write Command **AT+QHTTPCFG="contextid"[,<contextID>]**	Response If the optional parameter is omitted, query the current settings: **+QHTTPCFG: "contextid",<contextID>** **OK** If the optional parameter is specified, configure the PDP context ID: **OK** If there is any error: **+CME ERROR: <err>**
Write Command **AT+QHTTPCFG="requestheader"[,<request_header>]**	Response If the optional parameter is omitted, query the current settings: **+QHTTPCFG: "requestheader",<request_header>** **OK** If the optional parameter is specified, disable or enable to customize HTTP(S) request header: **OK** If there is any error: **+CME ERROR: <err>**
Write Command **AT+QHTTPCFG="responseheader"[,<response_header>]**	Response If the optional parameter is omitted, query the current settings: **+QHTTPCFG: "responseheader",<response_header>** **OK** If the optional parameter is specified, disable or enable to output HTTP(S) response header: **OK** If there is any error: **+CME ERROR: <err>**
Write Command **AT+QHTTPCFG="sslctxid"[,<sslctxID>]**	Response If the optional parameter is omitted, query the current settings: **+QHTTPCFG: "sslctxid",<sslctxID>** **OK** If the optional parameter is specified, configure SSL context ID used for HTTP(S):

	OK If there is any error: **+CME ERROR: <err>**
Write Command **AT+QHTTPCFG="contenttype"[,<content_type>]**	Response If the optional parameter is omitted, query the current settings: **+QHTTPCFG: "contenttype",<content_type>** **OK** If the optional parameter is specified, configure the data type of HTTP(S) body: **OK** If there is any error: **+CME ERROR: <err>**
Write Command **AT+QHTTPCFG="rspout/auto"[,<auto_outrsp>]**	Response If the optional parameter is omitted, query the current settings: **+QHTTPCFG: "rspout/auto",<auto_outrsp>** **OK** If the optional parameter is specified, disable or enable auto output of HTTP(S) response header: **OK** If there is any error: **+CME ERROR: <err>**
Write Command **AT+QHTTPCFG="closed/ind"[,<closedind>]**	Response If the optional parameter is omitted, query the current settings: **+QHTTPCFG: "closed/ind",<closedind>** **OK** If the optional parameter is specified, disable or enable the report of HTTP(S) session closing URC **+QHTTPURC: "closed"**: **OK** If there is any error: **+CME ERROR: <err>**
Maximum Response Time	-
Characteristics	The command takes effect immediately.

	The configurations are not saved.

Parameter

<contextID>	Integer type. PDP context ID. Range: 1–16. Default value: 1.
<request_header>	Integer type. Disable or enable to customize HTTP(S) request header. 0 Disable 1 Enable
<response_header>	Integer type. Disable or enable to output HTTP(S) response header. 0 Disable 1 Enable
<sslctxID>	Integer type. SSL context ID used for HTTP(S). Range: 0–5. Default value: 1. SSL parameters should be configured by **AT+QSSLCFG**. For more details, see ***document [2]***.
<content_type>	Integer type. Data type of HTTP(S) body. 0 application/x-www-form-urlencoded 1 text/plain 2 application/octet-stream 3 multipart/form-data 4 application/json 5 image/jpeg
<auto_outrsp>	Integer type. Disable or enable auto output of HTTP(S) response header. If auto output of HTTP(S) response header is enabled, **AT+QHTTPREAD** and **AT+QHTTPREADFILE** will fail to be executed. 0 Disable 1 Enable
<closedind>	Integer type. Disable or enable the report of HTTP(S) session closing URC **+QHTTPURC: "closed"**. 0 Disable 1 Enable
<err>	Integer type. Error code. For more details, see ***Chapter 5***.

2.3.2. AT+QHTTPURL Set URL of HTTP(S) Server

This command sets URL of HTTP(S) server. HTTP(S) server URL must begin with http:// or https://, which indicates the access to an HTTP or HTTPS server.

AT+QHTTPURL Set URL of HTTP(S) Server	
Test Command **AT+QHTTPURL=?**	Response **+QHTTPURL: (**range of supported **<URL_length>**s**),(**range of supported **<timeout>**s**)**

	OK
Read Command **AT+QHTTPURL?**	Response **[+QHTTPURL: <URL>]** **OK**
Write Command **AT+QHTTPURL=<URL_length>[,<timeout>]**	Response If the parameter format is correct, and HTTP(S) GET/POST/PUT requests are not sent: **CONNECT** TA switches to transparent transmission mode (data mode), and the URL can be inputted. When the total size of the inputted data reaches **<URL_length>**, TA will return to command mode and report the following code: **OK** If the **<timeout>** has been reached, but the length of received URL is less than **<URL_length>**, TA will return to command mode and report the following code: **+CME ERROR: <err>** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	Determined by **<timeout>**
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<URL_length>	Integer type. The length of URL. Range: 1–1024. Unit: byte.
<timeout>	Integer type. The maximum time for inputting URL. Range: 1–65535. Default value: 60. Unit: second.
<URL>	String type. HTTP(S) server URL.
<err>	Integer type. Error code. For more details, see ***Chapter 5***.

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

This command sends GET request to HTTP(S) server. According to the configured **<request_header>** in **AT+QHTTPCFG="requestheader"[,<request_header>]**, **AT+QHTTPGET** has two different formats.

- If **<request_header>** is set to 0, disable to customize HTTP(S) GET request header.
- If **<request_header>** is set to 1, enable to customize HTTP(S) GET request header.

After sending **AT+QHTTPGET**:

- **CONNECT** is outputted within 125 seconds (HTTP) or 425 seconds (HTTPS) to indicate successful establishment of HTTP(S) server connection.
- If that is not the case, then **+CME ERROR: <err>** will be outputted.

After executing **AT+QHTTPGET** and **OK** is returned, it is recommended to wait for a specific period of time (determined by network and **<rsptime>**) for URC **+QHTTPGET: <err>[,<httprspcode>[,<content_length>]]** to be outputted. **<httprspcode>** can only be reported when **<err>** is set to 0. If HTTP(S) response header contains the length of HTTP(S) response body, then **<content_length>** information will be reported.

AT+QHTTPGET Send GET Request to HTTP(S) Server	
Test Command **AT+QHTTPGET=?**	Response **+QHTTPGET: (**range of supported **<rsptime>**s**),(**range of supported **<data_length>**s**),(**range of supported **<input_time>**s**)** **OK**
Write/Execution Command If **<request_header>** is set to 0 (disable to customize HTTP(S) request header) **AT+QHTTPGET[=<rsptime>]**	Response If the parameter format is correct and no other errors occur: **OK** When the module has received response from 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>**
Write Command If **<request_header>** is set to 1 (enable to customize HTTP(S) GET request header) **AT+QHTTPGET=<rsptime>,<data_length>[,<input_time>]**	Response If the parameter format is correct and HTTP(S) server is connected successfully: **CONNECT** TA switches to transparent transmission mode (data mode), and the HTTP(S) GET request header can be inputted. When the length of inputted data reaches **<data_length>**, TA will return to command mode and report the following code: **OK** When the module has received response from HTTP(S) server, it will report the following URC: **+QHTTPGET: <err>[,<httprspcode>[,<content_length>]]**

	If the **<input_time>** has been reached, but the length of received data is less than **<data_length>**, TA will return to command mode and report the following code: **+CME ERROR: <err>** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	Determined by **<rsptime>**
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<rsptime>	Integer type. Timeout value for URC **+QHTTPGET: <err>[,<httprspcode>[,<content_length>]]** to be outputted. Range: 1–65535. Default value: 60. Unit: second.
<data_length>	Integer type. The length of HTTP(S) GET request, including HTTP(S) GET request header and request body. Range: 1–2048. Unit: byte.
<input_time>	Integer type. The maximum time for inputting HTTP(S) GET request, including HTTP(S) GET request header and request body. Range: 1–65535. Default value: 60. Unit: second.
<httprspcode>	Integer type. HTTP(S) response code. For more details, see ***Chapter 6***.
<request_header>	Integer type. Disable or enable to customize HTTP(S) request header. 0 Disable 1 Enable
<content_length>	Integer type. The length of HTTP(S) response body. Unit: byte.
<err>	Integer type. Error code. For more details, see ***Chapter 5***.

2.3.4. AT+QHTTPGETEX Send GET Request to HTTP(S) Server to Get Data Within Specified Range

This command sends GET request to the HTTP(S) server to get data within a specified range. MCU can get data with specified position and length from the HTTP(S) server by **AT+QHTTPGETEX**, and this command is only executable if **AT+QHTTPCFG="requestheader",0**. After that, HTTP(S) server will always respond with **206** code (**206**: Get data successfully) to the GET request to get data with specified position and length.

AT+QHTTPGETEX	**Send GET Request to HTTP(S) Server to Get Data Within Specified Range**
Test Command **AT+QHTTPGETEX=?**	Response **+QHTTPGETEX: (**range of supported **<rsptime>**s**),<start_postion>,<read_len>**

	OK
Write Command **AT+QHTTPGETEX=<rsptime>,<start_position>,<read_len>**	Response If the parameter format is correct and no other errors occur: **OK** When the module has received response from HTTP(S) server, it will report the following URC: **+QHTTPGET: <err>[,<httprspcode>[,<content_length>]]** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	Determined by **<rsptime>**
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<rsptime>	Integer type. Timeout value for the URC **+QHTTPGETEX: <err>[,<httprspcode>[,<content_length>]]** to be outputted. Range: 1–65535. Default value: 60. Unit: second.
<start_postion >	Integer type. The start position of the data that HTTP(S) client requires to get.
<read_len>	Integer type. The length of the data that HTTP(S) client requires to get. Unit: byte.
<httprspcode>	Integer type. HTTP(S) response code. For more details, see ***Chapter 6***.
<content_length>	Integer type. The length of HTTP(S) response body. Unit: byte.
<err>	Integer type. Error code. For more details, see ***Chapter 5***.

2.3.5. AT+QHTTPPOST Send POST Request to HTTP(S) Server via UART/USB

The command sends POST request to HTTP(S) server via UART/USB. According to the configured **<request_header>** in **AT+QHTTPCFG="requestheader"[,<request_header>]**, **AT+QHTTPPOST** has two different formats.

- If **<request_header>** is set to 0, only HTTP(S) POST body should be inputted via UART/USB.
- If **<request_header>** is set to 1, both the HTTP(S) POST header and body should be inputted via UART/USB.

After sending **AT+QHTTPPOST**:

- **CONNECT** is outputted within 125 seconds (HTTP) or 425 seconds (HTTPS) to indicate successful establishment of HTTP(S) server connection.
- If that is not the case, then **+CME ERROR: <err>** will be returned.

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

AT+QHTTPPOST Send POST Request to HTTP(S) Server via UART/USB	
Test Command **AT+QHTTPPOST=?**	Response **+QHTTPPOST: (**range of supported **<data_length>**s**),(**range of supported **<input_time>**s**),(**range of supported **<rsptime>**s**)** **OK**
Write Command If **<request_header>** is set to 0 (disable to customize HTTP(S) request header) **AT+QHTTPPOST=<data_length>[,<input_time>,<rsptime>]**	Response If the parameter format is correct, HTTP(S) server is connected successfully and HTTP(S) request header is sent: **CONNECT** TA switches to transparent transmission mode (data mode), and the HTTP(S) POST request body can be inputted. When the length of inputted data reaches **<data_length>**, TA will return to command mode and report the following code: **OK** When the module has received the response from HTTP(S) server, it will report the following URC: **+QHTTPPOST: <err>[,<httprspcode>[,<content_length>]]** If the **<input_time>** has been reached, but the length of received data is less than **<data_length>**, TA will return to command mode and report the following code: **+CME ERROR: <err>** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Write Command If **<request_header>** is set to 1 (enable to customize HTTP(S) request header) **AT+QHTTPPOST=<data_length>[,<input_time>,<rsptime>]**	Response If the parameter format is correct and HTTP(S) server is connected successfully: **CONNECT** TA switches to the transparent transmission mode (data mode), and the HTTP(S) POST request header and body can be inputted. When the length of inputted data reaches **<data_length>**, TA will return to command mode and report the following code: **OK**

	When the module has received response from HTTP(S) server, it will report the following URC: **+QHTTPPOST: <err>[,<httprspcode>[,<content_length>]]** If **<input_time>** has been reached, but the length of received data is less than **<data_length>**, TA will return to command mode and report the following code: **+CME ERROR: <err>** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	Determined by network and **<rsptime>**
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<data_length>	Integer type. If **<request_header>** is set to 0, it indicates the length of HTTP(S) POST request body. If **<request_header>** is set to 1, it indicates the length of HTTP(S) POST request information, including HTTP(S) POST request header and body. Range: 1–1024000. Unit: byte.
<input_time>	Integer type. If **<request_header>** is set to 0, it indicates the maximum input time of HTTP(S) POST request body; if **<request_header>** is set to 1, it indicates the maximum input time of HTTP(S) POST request information (including request header and request body). Range: 1–65535. Default value: 60. Unit: second.
<rsptime>	Integer type. Timeout value for URC **+QHTTPPOST: <err>[,<httprspcode>[,<content_length>]]** to be outputted. Range: 1–65535. Default value: 60. Unit: second.
<httprspcode>	Integer type. HTTP(S) response code. For more details, see ***Chapter 6***.
<request_header>	Integer type. Disable or enable customizing HTTP(S) request header. 0 Disable 1 Enable
<content_length>	Integer type. The length of HTTP(S) response body. Unit: byte.
<err>	Integer type. Error code. For more details, see ***Chapter 5***.

2.3.6. AT+QHTTPPOSTFILE Send POST Request to HTTP(S) Server via File

This command sends POST request to HTTP(S) server via file. According to the configured **<request_header>** in **AT+QHTTPCFG="requestheader"[,<request_header>]**, the file of **AT+QHTTPPOSTFILE** has two different formats.

- If **<request_header>** is set to 0, the file in file system will be HTTP(S) POST body only.
- If **<request_header>** is set to 1, the file in file system will be both HTTP(S) POST header and body.

After executing **AT+QHTTPPOSTFILE** and **OK** is returned, it is recommended to wait for a specific period of time (determined by network and **<rsptime>**) for **+QHTTPPOSTFILE: <err>[,<httprspcode>[,<content_length>]]** to be outputted. **<httprspcode>** can only be reported when **<err>** is set to 0.

AT+QHTTPPOSTFILE Send POST Request to HTTP(S) Server via File	
Test Command **AT+QHTTPPOSTFILE=?**	Response **+QHTTPPOSTFILE: <file_name>,(**range of supported **<rsptime>**s**),(**range of supported **<post_mode>**s**)** **OK**
Write Command **AT+QHTTPPOSTFILE=<file_name>[,<rsptime>,<post_mode>]**	Response If the parameter format is correct and HTTP(S) server is connected successfully: **OK** When the module has received response from HTTP(S) server, it will report the following URC: **+QHTTPPOSTFILE: <err>[,<httprspcode>[,<content_length>]]** If parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	Determined by **<rsptime>**
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<file_name>	String type. File name. Maximum length:80. Unit: byte.
<rsptime>	Integer type. Timeout value for URC **+QHTTPPOSTFILE: <err>[,<httprspcode>[,<content_length>]]** to be outputted. Range: 1–65535. Default value: 60. Unit: second.
<post_mode>	Integer type. HTTP(S) sending file mode. 0 Send the current file directly 1 Store the file to be sent (not send the file currently, waiting to send with the file configured when **<post_mode>**=2 2 Send all the files configured when **<post_mode>**=1 and 2 in order (The two files can only be sent together)
<httprspcode>	Integer type. HTTP(S) response code. For more details, see ***Chapter 6***.

<request_header>	Integer type. Disable or enable to customize HTTP(S) request header. 0 Disable 1 Enable
<content_length>	Integer type. The length of HTTP(S) response body. Unit: byte.
<err>	Integer type. Error code. For more details, see ***Chapter 5***.

2.3.7. AT+QHTTPPUT Send PUT Request to HTTP(S) Server via UART/USB

This command sends PUT request to HTTP(S) server via UART/USB. According to the configured **<request_header>** in **AT+QHTTPCFG="requestheader"[,<request_header>]**, **AT+QHTTPPUT** has two different formats.

- If **<request_header>** is set to 0, only HTTP(S) PUT body should be inputted via UART/USB port.
- If **<request_header>** is set to 1, both the HTTP(S) PUT header and body should be inputted via UART/USB port.

After sending **AT+QHTTPPUT**:

- **CONNECT** is outputted within 125 seconds (HTTP) or 425 seconds (HTTPS) to indicate successful establishment of HTTP(S) server connection.
- If that is not the case, then **+CME ERROR: <err>** will be returned.

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

AT+QHTTPPUT Send PUT Request to HTTP(S) Server via UART/USB	
Test Command **AT+QHTTPPUT=?**	Response **+QHTTPPUT: (**range of supported **<data_length>**s**),(**range of supported **<input_time>**s**),(**range of supported **<rsptime>**s**)** **OK**
Write Command If **<request_header>** is set to 0 (disable to customize HTTP(S) request header) **AT+QHTTPPUT=<data_length>[,<input_time>,<rsptime>]**	Response If the parameter format is correct, HTTP(S) server is connected successfully and HTTP(S) request header is sent: **CONNECT** TA switches to transparent transmission mode (data mode), and then the HTTP(S) PUT request body can be inputted. When the length of inputted data reaches **<data_length>**, TA will return to command mode and report the following code: **OK**

	When the module has received a response from HTTP(S) server, it will report the following URC: **+QHTTPPUT: <err>[,<httprspcode>[,<content_length>]]** If the **<input_time>** has been reached, but the length of received data is less than **<data_length>**, TA will return to command mode and report the following code: **+QHTTPPUT: <err>** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Write Command If **<request_header>** is set to 1 (enable customizing HTTP(S) request header) **AT+QHTTPPUT=<data_length>[,<input_time>,<rsptime>]**	Response If the parameter format is correct and HTTP(S) server is connected successfully: **CONNECT** TA switches to the transparent transmission mode (data mode), and then the HTTP(S) PUT request header and body can be inputted. When the length of inputted data reaches **<data_length>**, TA will return to command mode and report the following code: **OK** When the module has received response from HTTP(S) server, it will report the following URC: **+QHTTPPUT: <err>[,<httprspcode>[,<content_length>]]** If **<input_time>** has been reached, but the length of received data is less than **<data_length>**, TA will return to command mode and report the following code: **+QHTTPPUT: <err>** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	Determined by **<rsptime>**
Characteristics	This command takes effect immediately. The configurations are not saved.

Parameter

<data_length>	Integer type. If **<request_header>** is set to 0, it indicates the length of HTTP(S) PUT request body. If **<request_header>** is set to 1, it indicates the length of HTTP(S) PUT request information, including HTTP(S) PUT request

	header and body. Range: 1–1024000. Unit: byte.
<input_time>	Integer type. If **<request_header>** is set to 0, it indicates the maximum input time of HTTP(S) POST request body; if **<request_header>** is set to 1, it indicates the maximum input time of HTTP(S) POST request information (including request header and request body). Range: 1–65535. Default value: 60. Unit: second.
<rsptime>	Integer type. Timeout value for URC **+QHTTPPUT: <err>[,<httprspcode>[,<content_length>]]** to be outputted. Range: 1–65535. Default value: 60. Unit: second.
<httprspcode>	Integer type. HTTP(S) server response code. For more details, see ***Chapter 6***.
<request_header>	Integer type. Disable or enable customizing HTTP(S) request header. 0 Disable 1 Enable
<content_length>	Integer type. Length of HTTP(S) response body. Unit: byte.
<err>	Integer type. Error code. For more details, see ***Chapter 5*** for details.

2.3.8. AT+QHTTPPUTFILE Send PUT Request to HTTP(S) Server via File

The command sends PUT request HTTP(S) server via file. According to the configured **<request_header>** in **AT+QHTTPCFG="requestheader"[,<request_header>]**, the file of **AT+QHTTPPUTFILE** has two different formats.

- If **<request_header>** is set to 0, the file in file system will be HTTP(S) PUT body only.
- If **<request_header>** is set to 1, the file in file system will be both HTTP(S) PUT header and body.

After executing **AT+QHTTPPUTFILE** and **OK** is returned, it is recommended to wait for a specific period of time (determined by network and **<rsptime>**) for **+QHTTPPUTFILE: <err>[,<httprspcode>[,<content_length>]]** to be outputted. **<httprspcode>** can only be reported when **<err>** is set to 0.

AT+QHTTPPUTFILE Send PUT Request to HTTP(S) Server via File	
Test Command **AT+QHTTPPUTFILE=?**	Response **+QHTTPPUTFILE: <file_name>,(**range of supported **<rsptime>**s**)** **OK**
Write Command **AT+QHTTPPUTFILE=<file_name>[,<rsptime>]**	Response If parameter format is correct and HTTP(S) server is connected successfully: **OK** When the module has received response from HTTP(S) server, it will report the following URC:

	+QHTTPPUTFILE: <err>[,<httprspcode>[,<content_length>]] If parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	Determined by **<rsptime>**
Characteristics	This command takes effect immediately. The configurations are not saved.

Parameter

<file_name>	String type. File name. Maximum length:80. Unit: byte.
<rsptime>	Integer type. Timeout value for URC **+QHTTPPUTFILE: <err>[,<httprspcode>[,<content_length>]]** to be outputted. Range: 1–65535. Default value: 60. Unit: second.
<httprspcode>	Integer type. HTTP(S) response code. For more details, see ***Chapter 6***.
<request_header>	Integer type. Disable or enable to customize HTTP(S) request header. 0 Disable 1 Enable
<content_length>	Integer type. The length of HTTP(S) response body. Unit: byte.
<err>	Integer type. Error code. For more details, see ***Chapter 5***.

2.3.9. AT+QHTTPREAD Read Response from HTTP(S) Server via UART/USB

This command reads the HTTP(S) response from HTTP(S) server via UART/USB after HTTP(S) GET/POST/PUT requests are sent. It must be executed after any one of the following URCs is received:

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

AT+QHTTPREAD Read Response from HTTP(S) Server via UART/USB	
Test Command **AT+QHTTPREAD=?**	Response **+QHTTPREAD: (**range of supported **<wait_time>**s**)** **OK**
Write Command **AT+QHTTPREAD[=<wait_time>]**	Response If the parameter format is correct and read successfully: **CONNECT** <Output HTTP(S) response information>

	OK When the response information is read or **<wait_time>** is reached, it will report: **+QHTTPREAD: <err>** If **<wait_time>** has been reached or other errors occur, but the request body is not outputted completely: **+CME ERROR: <err>** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	Determined by **<wait_time>**
Characteristics	The command takes effect immediately. The configuration is not saved.

Parameter

<wait_time>	Integer type. The maximum interval time between receiving two packets of data. Range: 1–65535. Default value: 60. Unit: second.
<httprspcode>	Integer type. HTTP(S) response code. For more details, see ***Chapter 6***.
<content_length>	Integer type. The length of HTTP(S) response body. Unit: byte.
<err>	Integer type. Error code. For more details, see ***Chapter 5***.

2.3.10. AT+QHTTPREADFILE Read Response from HTTP(S) Server via File and Store Response Information

This command reads the HTTP(S) response from HTTP(S) server via file after HTTP(S) GET/POST/PUT requests are sent and stores response information. It must be executed after any one of the following URCs is received.

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

AT+QHTTPREADFILE Read Response from HTTP(S) Server via File and Store Response Information	
Test Command **AT+QHTTPREADFILE=?**	Response **+QHTTPREADFILE: <file_name>,(**range of supported **<wait_time>**s**)** **OK**
Write Command **AT+QHTTPREADFILE=<file_name>[, <wait_time>]**	Response If the parameter format is correct: **OK** When the response information is read or **<wait_time>** is reached, it will report: **+QHTTPREADFILE: <err>** If **<wait_time>** has been reached or other errors occur, but the request body is not outputted completely: **+CME ERROR: <err>** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	Determined by **<wait_time>**
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<wait_time>	Integer type. The maximum interval time between receiving two packets of data. Range: 1–65535. Default value: 60. Unit: second.
<file_name>	String type. Name of the file used to store response information. Maximum length: 80. Unit: byte.
<err>	Integer type. Error code. For more details, see ***Chapter 5***.

2.3.11. AT+QHTTPSTOP Cancel HTTP(S) Request

MCU can cancel HTTP(S) GET/POST/PUT request, and disconnect session with HTTP(S) server through this command.

AT+QHTTPSTOP Cancel HTTP(S) Request	
Test Command **AT+QHTTPSTOP=?**	Response **OK**

Execution Command **AT+QHTTPSTOP**	Response **OK** If there is any error: **+CME ERROR: <err>**
Maximum Response Time	10 s
Characteristics	-

Parameter

<err>	Integer type. Error code. For more details, see ***Chapter 5***.

3 Examples

3.1. Access to HTTP Server

3.1.1. Send HTTP GET Request and Read Response

The following examples show how to send a HTTP GET request and enable the output of HTTP response header, as well as how to read the HTTP GET response.

```
//Example of how to send HTTP GET request.
AT+QHTTPCFG="contextid",1                  //Configure the PDP context ID as 1.
OK
AT+QHTTPCFG="responseheader",1             //Enable to output HTTP response header.
OK
AT+QIACT?                                  //Query the state of PDP context.
OK                                         //Only returning OK means that there is no activated PDP context currently.
AT+QICSGP=1,1,"UNINET","","",1             //Configure PDP context as 1 and APN as China Unicom "UNINET". (Set AT+CFUN=1,1 for the configuration to take effect.)
OK
//The first PDP is activated by default. If it is queried inactivated, use AT+QIACT=1 to activate it.
AT+QIACT=1                                 //Activate PDP context 1.
OK                                         //Activated successfully.
AT+QIACT?                                  //Query the state of PDP context.
+QIACT: 1,1,1,"10.7.157.1"

OK
AT+QHTTPURL=22,80                          //Set the URL which will be accessed and timeout value as 80 s.

CONNECT
http://httpbin.org/get                     //Input URL whose length is 22 bytes. (This URL is only an example. Please input the correct URL in practical test.)

OK
AT+QHTTPGET=80                             //Send HTTP GET request and the maximum response time is 80 s.

OK
```

+QHTTPGET: 0,200,259 //If HTTP response header contains the length of HTTP response body, then the **<content_length>** information (259 bytes) will be reported.

//Example of how to read HTTP GET response.
//Solution 1: Read HTTP response and output it via UART.
AT+QHTTPREAD=80 //Read HTTP response and output it via UART. The maximum time to wait for HTTP session to be closed is 80 s.

CONNECT
HTTP/1.1 200 OK<CR><LF> //HTTP response header and body.
Date: Thu, 07 Jan 2021 05:40:15 GMT
Content-Type: application/json
Content-Length: 259
Connection: keep-alive
Server: gunicorn/19.9.0
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true

```
{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Host": "httpbin.org",
    "User-Agent": "QUECTEL_MODULE",
    "X-Amzn-Trace-Id": "Root=1-5ff69ebf-0531ade06a92e98d1cfad7a3"
  },
  "origin": "223.246.244.153",
  "url": "http://httpbin.org/get"
}
```

OK

+QHTTPREAD: 0 //Read HTTP response header and body successfully.

//Solution 2: Read HTTP response and store it to RAM file.
AT+QHTTPREADFILE="RAM:1.txt",80 //Read HTTP response header and body and store them to *RAM:1.txt*. The maximum time to wait for HTTP session to be closed is 80 s.

OK

+QHTTPREADFILE: 0 //HTTP response header and body are stored successfully.

3.1.2. Send HTTP POST Request and Read the Response

3.1.2.1. HTTP POST Body Obtained via UART/USB

The following examples show how to send an HTTP POST request and read HTTP POST request body via UART, as well as how to read HTTP POST response.

```
AT+QHTTPCFG="contextid",1                  //Configure the PDP context ID as 1.
OK
AT+QHTTPCFG="responseheader",1             //Enable to output HTTP response header.
OK
AT+QIACT?                                  //Query the state of PDP context.
OK                                         //Only returning OK means that there is no activated PDP
                                             context ID currently.
AT+QICSGP=1,1,"UNINET","","",1             //Configure PDP context as 1 and APN as China Unicom
                                           "UNINET". (Set AT+CFUN=1,1 for the configuration to
                                           take effect.)
OK
//The first PDP is activated by default. If it is queried inactivated, use AT+QIACT=1 to activate it.
AT+QIACT=1                                 //Activate PDP context 1.
OK                                         //Activated successfully.
AT+QIACT?                                  //Query the state of PDP context.
+QIACT: 1,1,1,"172.22.86.226"

OK
AT+QHTTPURL=23,80                          //Set the URL which will be accessed and timeout value as 80 s.
CONNECT
http://httpbin.org/post                    //Input URL whose length is 23 bytes. (This URL is only an
                                             example. Please input the correct URL in practical test.)
OK
AT+QHTTPPOST=20,80,80                      //Send HTTP POST request and HTTP POST request body is
                                           obtained via UART. The maximum input and response time of
                                           the POST request body are both 80 s.
CONNECT
Message=HelloQuectel                       //Input HTTP POST request body whose length is 20 bytes. (The
                                           POST request body is only an example. Please input the correct
                                           POST request body in practical test.)
OK

+QHTTPPOST: 0,200,443                      //If the HTTP response header contains the length of HTTP
                                           response body, <content_length> (443 bytes) will be reported.
AT+QHTTPREAD=80                            //Read HTTP response body and output it via UART. The
                                           maximum time to wait for HTTP session to be closed is 80 s.
CONNECT
```

```
{
  "args": {},
  "data": "",
  "files": {},
  "form": {
    "Message": "HelloQuectel"
  },
  "headers": {
    "Accept": "*/*",
    "Content-Length": "20",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org",
    "User-Agent": "QUECTEL_MODULE",
    "X-Amzn-Trace-Id": "Root=1-5ff6a07d-528fed3e7d8d9f0d58e5491d"
  },
  "json": null,
  "origin": "223.246.244.153",
  "url": "http://httpbin.org/post"
}

OK

+QHTTPREAD: 0                    //HTTP response has been outputted successfully.
```

3.1.2.2. HTTP POST Body Obtained via File System

The following examples show how to send an HTTP POST request and read POST request body via file system, as well as how to store the HTTP POST response to file system.

```
AT+QHTTPCFG="contextid",1          //Configure the PDP context ID as 1.
OK
AT+QIACT?                          //Query the state of PDP context.
OK                                  //Only returning OK means that there is no activated PDP
                                     context ID currently.
AT+QICSGP=1,1,"UNINET","","",1     //Configure PDP context as 1 and APN as China Unicom
                                     "UNINET". (Set AT+CFUN=1,1 for the configuration to
                                     take effect.)
OK
//The first PDP is activated by default. If it is queried inactivated, use AT+QIACT=1 to activate it.
AT+QIACT=1                         //Activate PDP context 1.
OK                                 //Activated successfully.
AT+QIACT?                          //Query the state of PDP context.
+QIACT: 1,1,1,"172.22.86.226"
```

```
OK
AT+QHTTPURL=23,80                    //Set the URL which will be accessed and timeout value as 80 s.
CONNECT
http://httpbin.org/post              //Input URL whose length is 23 bytes. (This URL is only an
                                     example. Please input the correct URL in practical test.)
OK

//POST request from a RAM file, read HTTP response and store it to a RAM file.
AT+QHTTPPOSTFILE="RAM:2.txt",80      //Send HTTP POST request and obtain POST request body
                                     from RAM:2.txt,. The maximum response time is 80 s.
OK

+QHTTPPOSTFILE: 0,200,443            //After HTTP POST request is sent successfully, the HTTP
                                     response can be read by executing AT+QHTTPREADFILE.
AT+QHTTPREADFILE="RAM:3.txt",80      //Read HTTP response and store it to RAM:3.txt. The
                                     maximum time to wait for HTTP session to be closed is 80 s.
OK

+QHTTPREADFILE: 0                    //HTTP response has been stored successfully.
```

3.1.3. Send HTTP PUT Request and Read the Response

3.1.3.1. HTTP PUT Body Obtained via UART/USB

The following examples show how to send an HTTP PUT request and read the HTTP PUT request body via UART, as well as how to read the HTTP PUT response.

```
AT+QHTTPCFG="contextid",1            //Configure the PDP context ID as 1.
OK
AT+QHTTPCFG="responseheader",1       //Enable to output HTTP response header.
OK
AT+QIACT?                            // Query the state of PDP context.
OK                                   //Only returning OK means that there is no activated PDP
                                       context currently.
AT+QICSGP=1,1,"UNINET","","",1       //Configure PDP context as 1 and APN as China Unicom
                                     "UNINET". (Set AT+CFUN=1,1 for the configuration to
                                     take effect.)
OK
//The first PDP is activated by default. If it is queried inactivated, use AT+QIACT=1 to activate it.
AT+QIACT=1                           //Activate PDP context 1.
OK                                   //Activated successfully.
AT+QIACT?                            //Query the state of PDP context.
+QIACT: 1,1,1,"172.22.86.226"
```

```
OK
AT+QHTTPURL=46,80                    //Set the URL that will be accessed and  timeout value as 80 s.
CONNECT
http://220.180.239.212:8252/uploads/put_01.txt    //Input URL whose length is 46 bytes.
                                                  (This URL is only an example. Input the correct
                                                  URL used in practice.)
OK
AT+QHTTPPUT=20,80,80                 //Send HTTP PUT request and HTTP PUT REQUEST body is
                                     obtained via UART. The maximum input and response time of
                                     the PUT request body are  both 80 s.
CONNECT
Message=HelloQuectel                 //Input HTTP PUT request body whose length is 20 bytes. (The
PUT                                  body is only an example. Please input the correct PUT request
                                     body in practical test.)
OK

+QHTTPPUT: 0,200,177                 //If the HTTP response header contains the length of HTTP
                                     response body, <content_length> (177 bytes) will be reported.
AT+QHTTPREAD=80                      //Read the HTTP response and output it via UART. The
                                     maximum time to wait for HTTP session to be closed is 80 s.
CONNECT
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>201 Created</title>
</head><body>
<h1>Created</h1>
<p>Resource /uploads/put_01.txt has been created.</p>
</body></html>                       //Output HTTP response.
OK

+QHTTPREAD: 0                        //HTTP response has been outputted successfully.
```

3.1.3.2. HTTP PUT Body Obtained via File System

The following examples show how to send an HTTP PUT request and read PUT request body via file system, as well as how to store the HTTP PUT response to file system.

```
AT+QHTTPCFG="contextid",1            //Configure the PDP context ID as 1.
OK
AT+QIACT?                            //Query the state of PDP context.
OK                                   //Only returning OK means that there is no activated PDP
                                       context currently.
AT+QICSGP=1,1,"UNINET","","",1       //Configure PDP context as 1 and APN as China Unicom
```

```
                                    "UNINET". (Set AT+CFUN=1,1 for the configuration to
                                    take effect.)
OK
//The first PDP is activated by default. If it is queried inactivated, use AT+QIACT=1 to activate it.
AT+QIACT=1                          //Activate PDP context 1.
OK                                  //Activated successfully.
AT+QIACT?                           //Query the state of PDP context.
+QIACT: 1,1,1,"172.22.86.226"

OK
AT+QHTTPURL=46,80                   //Set the URL that will be accessed and timeout value as 80 s.
CONNECT
http://220.180.239.212:8252/uploads/put_02.txt   //Input URL whose length is 46 bytes. (This URL is
                                                 only an example. Please input the correct URL
                                                 used in practical test.)
OK
//PUT request from a UFS file, and read HTTP response and store it to a UFS file.
AT+QHTTPPUTFILE="UFS:2.txt",80      //Send HTTP PUT request and obtain PUT request body
                                    from UFS:2.txt, and the maximum response time is 80 s.
OK

+QHTTPPUTFILE: 0,200,177            //After HTTP PUT request is sent successfully, the HTTP
                                    response can be read by executing AT+QHTTPREADFILE.
AT+QHTTPREADFILE="UFS:3.txt",80     //Read HTTP response and store it to UFS:3.txt. The
                                    maximum time to wait for HTTP session to be closed is 80 s.
OK

+QHTTPREADFILE: 0                   //HTTP response has been stored successfully.
```

3.2. Access to HTTPS Server

3.2.1. Send HTTPS GET Request and Read the Response

The following examples show how to send a HTTPS GET request and enable the output of HTTPS response header, as well as how to read the HTTPS GET response.

```
//Example of how to send HTTPS GET request.
AT+QHTTPCFG="contextid",1           //Configure the PDP context ID as 1.
OK
AT+QHTTPCFG="responseheader",1      //Enable to output HTTPS response header.
OK
AT+QIACT?                           //Query the state of PDP context.
OK                                  //Only returning OK means that there is no activated PDP
```

```
                                        context currently.
AT+QICSGP=1,1,"UNINET","","",1          //Configure PDP context as 1 and APN as China Unicom
                                        "UNINET". (Then set AT+CFUN=1,1 for the configuration to
                                        take effect.)
OK
//The first PDP is activated by default. If it is queried inactivated, use AT+QIACT=1 to activate it.
AT+QIACT=1                              //Activate PDP context 1.
OK                                      //Activated successfully.
AT+QIACT?                               //Query the state of PDP context.
+QIACT: 1,1,1,"10.7.157.1"

OK
AT+QHTTPCFG="sslctxid",1                //Set SSL context ID.
OK
AT+QSSLCFG="sslversion",1,1             //Set SSL version as 1, which means TLSV1.0.
OK
AT+QSSLCFG="ciphersuite",1,0xC02F       //Set SSL cipher suite as 0xC02F, which means
                                        AES128-SHA.
OK
AT+QSSLCFG="seclevel",1,0               //Set SSL verify level as 0, which means CA certificate is not
                                        needed.
OK
AT+QHTTPURL=23,80                       //Set the URL that will be accessed and timeout value as 80 s.
CONNECT
https://httpbin.org/get                 //Input URL whose length is 23 bytes. (This URL is only an
                                        example. Please input the correct URL in practical test.)
OK
AT+QHTTPGET=80                          //Send HTTPS GET request and the maximum response
                                        time is 80 s.
OK

+QHTTPGET: 0,200,257                    //If HTTPS response header contains the length of
                                        HTTPS response body, then the <content_length>
                                        information (257 bytes) will be reported.

//Example of how to read HTTPS GET response.
//Solution 1: Read HTTPS response information and output it via UART.

AT+QHTTPREAD=80                         //Read HTTPS response and output it via UART. The
                                        maximum time to wait for HTTPS session to be closed is
                                        80 s.
CONNECT                                 //HTTPS response header and body.
HTTP/1.1 200 OK
Date: Fri, 08 Jan 2021 10:28:00 GMT
Content-Type: application/json
```

```
Content-Length: 257
Connection: keep-alive
Server: gunicorn/19.9.0
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true

{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Host": "httpbin.org",
    "User-Agent": "QUECTEL_MODULE",
    "X-Amzn-Trace-Id": "Root=1-5ff833b0-60e6d2373b6e15556f1801d9"
  },
  "origin": "36.61.88.160",
  "url": "https://httpbin.org/get"
}

OK

+QHTTPREAD: 0                    //Read HTTPS response header and body successfully.

//Solution 2: Read HTTPS response and store it to RAM file.
AT+QHTTPREADFILE="RAM:4.txt",80   //Read HTTPS response header and body and store them to
                                  RAM:4.txt. The maximum time to wait for HTTPS session to
                                  be closed is 80 s.
OK

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

3.2.2. Send HTTPS POST Request and Read the Response

3.2.2.1. HTTPS POST Body Obtained via UART/USB

The following examples show how to send an HTTPS POST request and read the POST request body via UART port, as well as how to read the HTTPS POST response.

```
AT+QHTTPCFG="contextid",1          //Configure the PDP context ID as 1.
OK
AT+QHTTPCFG="responseheader",1     //Enable to output HTTPS response header.
OK
AT+QIACT?                          //Query the state of PDP context.
OK                                 //Only returning OK means that there is no activated PDP
```

context currently.

AT+QICSGP=1,1,"UNINET","","",1 //Configure PDP context as 1 and APN as China Unicom "UNINET". (Then set **AT+CFUN=1,1** for the configuration to take effect.)

OK

//The first PDP is activated by default. If it is queried inactivated, use **AT+QIACT=1** to activate it.

AT+QIACT=1 //Activate PDP context 1.

OK //Activated successfully.

AT+QIACT? //Query the state of PDP context.

+QIACT: 1,1,1,"172.22.86.226"

OK

AT+QHTTPCFG="sslctxid",1 //Set SSL context ID as 1.

OK

AT+QSSLCFG="sslversion",1,1 //Set SSL version as 1, which means TLSV1.0.

OK

AT+QSSLCFG="ciphersuite",1,0xC02F //Set SSL cipher suite as 0xC02F, which means AES128-SHA.

OK

AT+QSSLCFG="seclevel",1,2 //Set SSL verify level as 2, which means CA certificate, client certificate and client private key should be uploaded by **AT+QFUPL**. For more details about **AT+QFUPL**, See ***document [3]***.

OK

AT+QSSLCFG="cacert",1,"RAM:cacert.pem" //Configure the path of trusted CA certificate for SSL context 1.

OK

AT+QSSLCFG="clientcert",1,"RAM:clientcert.pem" //Configure the path of client certificate for SSL context 1.

OK

AT+QSSLCFG="clientkey",1,"RAM:clientkey.pem" //Configure the path of client private key for SSL context 1.

OK

AT+QHTTPURL=45,80 //Set the URL which will be accessed and timeout value as 80 s.

CONNECT

HTTPs://220.180.239.212:8011/processorder.php //Input URL whose length is 45 bytes. (This URL is only an example. Please input the correct URL in practical test.)

OK

AT+QHTTPPOST=48,80,80 //Send HTTPS POST request. HTTPS POST request body is obtained via UART. The maximum input and response time are both 80 s.

CONNECT

Message=1111&Appleqty=2222&Orangeqty=3333&find=1 //Input HTTPS POST request body

whose length is 48 bytes. (This post body is only an example. Please input the correct one in practical test.)

OK

+QHTTPPOST: 0,200,285 //If the HTTPS response header contains the length of HTTPS response body, **<content_length>** (285 bytes) will be reported.

AT+QHTTPREAD=80 //Read HTTPS response and output it via UART. The maximum time to wait for HTTPS session to be closed is 80 s.

CONNECT //HTTPS response has been 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 has been outputted successfully.

3.2.2.2. HTTPS POST Body Obtained via File System

The following examples show how to send a HTTPS POST request and read the HTTPS POST body from a file system, as well as how to store the HTTPS POST response to a file system.

AT+QHTTPCFG="contextid",1 //Configure the PDP context ID as 1.

OK

AT+QIACT? //Query the state of PDP context.

OK //Only returning **OK** means that there is no activated PDP context currently.

AT+QICSGP=1,1,"UNINET","","",1 //Configure PDP context as 1 and APN China Unicom as "UNINET". (Set **AT+CFUN=1,1** for the configuration to take effect.)

OK

//The first PDP is activated by default. If it is queried inactivated, use **AT+QIACT=1** to activate it.

AT+QIACT=1 //Activate PDP context 1.

OK //Activated successfully.

AT+QIACT? //Query the state of PDP context.

```
+QIACT: 1,1,1,"172.22.86.226"

OK
AT+QHTTPCFG="sslctxid",1          //Set SSL context ID as 1.
OK
AT+QSSLCFG="sslversion",1,1       //Set SSL version as 1, which means TLSV1.0.
OK
AT+QSSLCFG="ciphersuite",1,0x0005     //Set SSL cipher suite as 0x0005, which means RC4-SHA.
OK
AT+QSSLCFG="seclevel",1,2         //Set SSL verify level as 2, which means CA certificate, client
                                  certificate and client private key should be uploaded by
                                  AT+QFUPL. For more details about AT+QFUPL, See
                                  document [3].
OK
AT+QSSLCFG="cacert",1,"RAM:cacert.pem"          //Configure the path of CA certificate for SSL
                                                context 1.
OK
AT+QSSLCFG="clientcert",1,"RAM:clientcert.pem"  //Configure the path of client certificate for
                                                SSL context 1.
OK
AT+QSSLCFG="clientkey",1,"RAM:clientkey.pem"    //Configure the path of client private key for
                                                SSL context 1.
OK
AT+QHTTPURL=45,80             //Set the URL which will be accessed and timeout value as 80 s.
CONNECT
https://220.180.239.212:8011/processorder.php   //Input URL whose length is 45 bytes. (This URL is
                                                only an example. Please input the correct URL in
                                                practical test.)
OK

//POST request from RAM file, and read HTTPS response and store it to RAM file.
AT+QHTTPPOSTFILE="RAM:5.txt",80  //Send HTTPS POST request and obtain HTTPS POST request
                                 body from RAM:5.txt. The maximum response time is 80 s.
OK

+QHTTPPOSTFILE: 0,200,177        //After HTTPS POST request is sent successfully, the HTTPS
                                 response can be read by executing AT+QHTTPREADFILE.
AT+QHTTPREADFILE="RAM:6.txt",80  //Read HTTPS response and store it to RAM:6.txt. The
                                   maximum time to wait for HTTPS session to be closed is
                                   80 s.
OK

+QHTTPREADFILE: 0                //HTTPS response has been stored successfully.
```

3.2.3. Send HTTPS PUT Request and Read the Response

3.2.3.1. HTTPS PUT Body Obtained from UART/USB

The following examples show how to send an HTTPS POST request and read the PUT request body via UART, as well as how to read the HTTPS PUT response.

```
AT+QHTTPCFG="contextid",1                  //Configure the PDP context ID as 1.
OK
AT+QHTTPCFG="responseheader",1             //Enable to output HTTPS response header.
OK
AT+QIACT?                                  //Query the state of PDP context.
OK                                         //Only returning OK means that there is no activated PDP
                                             context currently.
AT+QICSGP=1,1,"UNINET","","",1             //Configure PDP context as 1 and China Unicom APN as
                                           "UNINET" (Then set AT+CFUN=1,1 for the configuration
                                           to take effect.)
OK
//The first PDP is activated by default. If it is queried inactivated, use AT+QIACT=1 to activate it.
AT+QIACT=1                                 //Activate PDP context 1.
OK                                         //Activated successfully.
AT+QIACT?                                  // Query the state of PDP context.
+QIACT: 1,1,1,"172.22.86.226"

OK
AT+QHTTPCFG="sslctxid",1                   //Set SSL context ID as 1.
OK
AT+QSSLCFG="sslversion",1,1                //Set SSL version as 1, which means TLSV1.0.
OK
AT+QSSLCFG="ciphersuite",1,0x0005          //Set SSL cipher suite as 0x0005, which means RC4-SHA.
OK
AT+QSSLCFG="seclevel",1,2                  //Set SSL verification level as 2, which means that a CA,
                                           certificate, client certificate and client private key should be
                                           uploaded with 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"      //Configure the path of trusted CA certificate
for                                         SSL context 1.
OK
AT+QSSLCFG="clientcert",1,"UFS:clientcert.pem"  //Configure the path of client certificate for
                                                SSL context 1.
OK
AT+QSSLCFG="clientkey",1,"UFS:clientkey.pem"    //Configure the path of client private key for
                                                SSL context 1.
OK
AT+QHTTPURL=45,80                           //Set the URL which will be accessed and
                                            timeout value as 80 s.
CONNECT
HTTPs://220.180.239.212:8011/processorder.php   //Input the URL whose length is 45 bytes.
                                                (This URL is only an example. Please input
                                            the correct URL in practical test.)
OK
AT+QHTTPPUT=48,80,80                        //Send HTTPS PUT request and HTTPS PUT
                                            request body is obtained via UART. The maximum
                                            input and response time are both 80 s.
CONNECT
Message=1111&Appleqty=2222&Orangeqty=3333&find=1    //Input HTTPS PUT body whose length
                                                    is 48 bytes. (This POST body is only an
                                                    example. Please input the correct one in
                                                    practical test.)
OK

+QHTTPPUT: 0,200,285                        //If the HTTPS response header contains the length of
                                            HTTPS response body, <content_length> (285 bytes)
                                            will be reported.
AT+QHTTPREAD=80                             //Read HTTPS response and output it via UART.
                                            The maximum time to wait for HTTPS session to be
                                            closed is 80 s.
CONNECT                                     //HTTPS response has been read successfully.
<html>
<head>
```

```
<title>Quectel's Auto Parts - Order Results</title>
</head>
<body>
<h1>Quectel's Auto Parts</h1>
<h2>Order Results</h2>
Content-Type:application/x-www-form-urlencoded
<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 has been outputted successfully.
```

3.2.3.2. HTTPS PUT Body Obtained from File System

The following examples show how to send an HTTPS PUT request and read the HTTPS PUT body from a file system, as well as how to store the HTTPS PUT response to a file system.

```
AT+QHTTPCFG="contextid",1          //Configure the PDP context ID as 1.
OK
AT+QIACT?                          // Query the state of PDP context.
OK                                 //Only returning OK means that there is no activated PDP
                                     context currently.
AT+QICSGP=1,1,"UINET","","",1      //Configure PDP context as 1 and China Unicom APN as
                                   "UINET". (Then set AT+CFUN=1,1 for the configuration to take
                                   effect.)
OK
//The first PDP is activated by default. If it is queried inactivated, use AT+QIACT=1 to activate it.
AT+QIACT=1                         //Activate PDP context 1.
OK                                 //Activated successfully.
AT+QIACT?                          //Query the state of PDP context.
+QIACT: 1,1,1,"172.22.86.226"

OK
AT+QHTTPCFG="sslctxid",1           //Set SSL context ID as 1.
OK
AT+QSSLCFG="sslversion",1,1        //Set SSL version as 1, which means TLSV1.0.
OK
AT+QSSLCFG="ciphersuite",1,0x0005//Set SSL cipher suite as 0x0005, which means RC4-SHA.
OK
AT+QSSLCFG="seclevel",1,2          //Set SSL verification level as 2, which means that a CA
                                   certificate, a client certificate and a client private key should be
                                   uploaded with 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"          //Configure the path of CA certificate for SSL
                                                 context 1.
OK
AT+QSSLCFG="clientcert",1,"UFS:clientcert.pem"  //Configure the path of client certificate for
                                                 SSL context 1.
OK
AT+QSSLCFG="clientkey",1,"UFS:clientkey.pem"    //Configure the path of client private key for
                                                 SSL context 1.
OK
AT+QHTTPURL=45,80                               //Set the URL which will be accessed and
                                                 timeout value as 80 s.
CONNECT
https://220.180.239.212:8011/processorder.php   //Input URL whose length is 45 bytes. (This
                                                 URL is only an example. Please input the
                                                 correct URL in practical test.)
OK
//PUT request from UFS file, and read HTTPS response and store it to a UFS file.
AT+QHTTPPUTFILE="UFS:5.txt",80                  //Send HTTPS PUT request and obtain HTTPS
                                                 POST request body from UFS:5.txt. The maximum
                                                 response time is 80 s.
OK

+QHTTPPUTFILE: 0,200,177                        //After HTTPS PUT request is sent successfully,
                                                 the HTTPS response can be read by executing
                                                 AT+QHTTPREADFILE.
```

AT+QHTTPREADFILE="UFS:6.txt",80	//Read HTTPS response and store it to *UFS:6.txt*. The maximum time to wait for an HTTPS session to be closed is 80 s.
OK	
+QHTTPREADFILE: 0	//HTTPS response has been stored successfully.

4 Solutions to Common Problems

4.1. Executing HTTP(S) AT Commands Fails

If **ERROR** returns after executing HTTP(S) AT commands, please check whether the (U)SIM card is inserted and check whether **+CPIN: READY** returns after executing **AT+CPIN?**.

4.2. PDP Activation Fails

If it fails to active a PDP context by **AT+QIACT**, please check the following configurations:

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

If all above configurations are correct but the PDP context activation by **AT+QIACT** still fails, please reboot the module. After rebooting the module, please check the configurations above for at least three times and each time at an interval of 10 minutes to avoid frequently rebooting.

4.3. DNS Parse Fails

When executing **AT+QHTTPGET**, **AT+QHTTPGETEX**, **AT+QHTTPPOST**, **AT+QHTTPPOSTFILE**, **AT+QHTTPPUT** and **AT+QHTTPPUTFILE**, if **+CME ERROR: 714** (714: HTTP(S) DNS error) is returned, please check the following aspects:

1. Make sure the domain name of HTTP(S) server is valid.
2. Query the status of the PDP context by **AT+QIACT?** to make sure the specified PDP context has

been activated successfully.
3. Query the address of DNS server by **AT+QIDNSCFG** to make sure the address of DNS server is not "0.0.0.0".

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

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

4.4. Data Mode Related Operation Fails

When executing **AT+QHTTPURL**, **AT+QHTTPGET**, **AT+QHTTPGETEX**, **AT+QHTTPPOST**, **AT+QHTTPPOSTFILE**, **AT+QHTTPPUT**, **AT+QHTTPPUTFILE**, **AT+QHTTPREAD** and **AT+QHTTPREADFILE**, if **+CME ERROR: 704** (704: HTTP(S) UART busy) is returned, please check whether there are other ports in data mode, since the module only supports one port in data mode at a time. If there is any other port in data mode, please re-execute these commands after other ports have exited from data mode.

4.5. Sending GET/POST/PUT Request Fails

When **AT+QHTTPGET**, **AT+QHTTPGETEX**, **AT+QHTTPPOST**, **AT+QHTTPPOSTFILE**, **AT+QHTTPPUT** and **AT+QHTTPPUTFILE** fail to be executed, please check the following configurations:

1. Make sure the URL inputted via **AT+QHTTPURL** is valid and can be accessed.
2. Make sure the specified server supports GET/POST/PUT commands.
3. Make sure the PDP context has been activated successfully.

If all above configurations are correct, but sending GET/POST/PUT requests by **AT+QHTTPGET**, **AT+QHTTPGETEX**, **AT+QHTTPPOST**, **AT+QHTTPPOSTFILE**, **AT+QHTTPPUT** and **AT+QHTTPPUTFILE** still fails, please deactivate the PDP context by **AT+QIDEACT** and re-activate the PDP context by **AT+QIACT**. If the PDP context activation fails, see ***Chapter 4.2***.

4.6. Reading Response Fails

Before reading response by **AT+QHTTPREAD** and **AT+QHTTPREADFILE**, execute **AT+QHTTPGET**, **AT+QHTTPGETEX**, **AT+QHTTPPOST**, **AT+QHTTPPOSTFILE**, **AT+QHTTPPUT** and **AT+QHTTPPUTFILE** first, and wait until the following URC information is reported:

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

In case of errors during the execution of **AT+QHTTPREAD** and **AT+QHTTPREADFILE**, such as **+CME ERROR: 717** (717: HTTP(S) Socket read error), please resend HTTP(S) GET/POST/PUT requests to HTTP(S) server by **AT+QHTTPGET**, **AT+QHTTPGETEX**, **AT+QHTTPPOST**, **AT+QHTTPPOSTFILE**, **AT+QHTTPPUT** and **AT+QHTTPPUTFILE**. If sending GET/POST/PUT requests to HTTP(S) server fails, see ***Chapter 4.5***.

5 Summary of Error Codes

The error code **<err>** indicates an error related to mobile equipment or network. The details about **<err>** are described in the following table.

Table 3: Summary of Error Codes

<err>	Description
0	Operation successful
701	HTTP(S) unknown error
702	HTTP(S) timeout
703	HTTP(S) busy
704	HTTP(S) UART busy
705	HTTP(S) no GET/POST requests
706	HTTP(S) network busy
707	HTTP(S) network open failed
708	HTTP(S) network no configuration
709	HTTP(S) network deactivated
710	HTTP(S) network error
711	HTTP(S) URL error
712	HTTP(S) empty URL
713	HTTP(S) IP address error
714	HTTP(S) DNS error
715	HTTP(S) socket create error
716	HTTP(S) socket connect error
717	HTTP(S) socket read error

718	HTTP(S) socket write error
719	HTTP(S) socket closed
720	HTTP(S) data encode error
721	HTTP(S) data decode error
722	HTTP(S) read timeout
723	HTTP(S) response failed
724	Incoming call busy
725	Voice call busy
726	Input timeout
727	Wait data timeout
728	Wait HTTP(S) response timeout
729	Memory allocation failed
730	Invalid parameter

6 Summary of HTTP(S) Response Codes

<httprspcode> indicates the response codes from HTTP(S) server. The details about **<httprspcode>** are described in the following table.

Table 4: Summary of HTTP(S) Response Codes

<httprspcode> Response Code	Description
200	OK
403	Forbidden
404	Not found
409	Conflict
411	Length required
500	Internal server error

7 Appendix References

Table 5: Related Documents

Document Name
[1] Quectel_RG50xQ&RM5xxQ_Series_TCP(IP)_Application_Note
[2] Quectel_RG50xQ&RM5xxQ_Series_SSL_Application_Note
[3] Quectel_RG50xQ&RM5xxQ_Series_FILE_Application_Note
[4] Quectel_RG50xQ&RM5xxQ_Series_AT_Commands_Manual

Table 6: Terms and Abbreviations

Abbreviation	Description
APN	Access Point Name
CA	Certification Authority
DNS	Domain Name Server
DTR	Data Terminal Ready
HTTP(S)	Hypertext Transfer Protocol (Secure)
ID	Identification
IP	Internet Protocol
MCU	Microprogrammed Control Unit
PDP	Packet Data Protocol
PPP	Point-to-Point Protocol
PS	Packet Switch

QoS	Quality of Service
RAM	Random Access Memory
SSL	Security Socket Layer
TA	Terminal Adapter
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UART	Universal Asynchronous Receiver/Transmitter
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URC	Unsolicited Result Code
USB	Universal Serial Bus
(U)SIM	(Universal) Subscriber Identity Module