BG95&BG77&BG600L Series
HTTP(S) Application Note

LPWA Module Series

Version: 1.2

Date: 2024-12-26

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
1.0	2019-08-12	Terrence YANG/ Sherlock ZHAO	Initial
1.1	2021-11-30	Water WANG/ Adonis CHEN	1. Added the applicable module BG600L-M3. 2. Extended the value range of <content_type> in AT+QHTTPCFG (Chapter 2.3.1). 3. Added the following AT commands (Chapter 2.3.1): AT+QHTTPCFG="auth",<username>:<password>; AT+QHTTPCFG="custom_header"[,"<custom_header>"]. 4. Extended the value range of <URL_Length> in AT+QHTTPURL (Chapter 2.3.2). 5. Added AT+QHTTPPUT (Chapter 2.3.6). 6. Added AT+QHTTPPUTFILE (Chapter 2.3.7). 7. Added examples about sending HTTP(S) PUT requests (Chapter 3.1.3 and Chapter 3.2.3).
1.2	2024-12-26	Water WANG	1. Updated the declaration of AT command examples (Chapter 2.2). 2. Updated IP and URL addresses in examples (Chapter 1.3.1, 3.1.1, 3.1.2.1, 3.1.2.2, 3.1.3.1, 3.1.3.2, 3.2.1, 3.2.2.1, 3.2.2.2, 3.2.3.1, and 3.2.3.2).

Contents

Table Index

1 Introduction

Quectel LPWA BG95 series, BG77 and BG600L-M3 modules support HTTP(S) applications through accessing HTTP(S) servers. This document is a reference guide that includes all AT commands defined for HTTP(S).

1.1. Applicable Modules

Table 1: Applicable Modules

Module Series	Model	Description
BG95	BG95-M1	Cat M1 only
	BG95-M2	Cat M1/Cat NB2
	BG95-M3	Cat M1/Cat NB2/EGPRS
	BG95-M4	Cat M1/Cat NB2, 450 MHz Supported
	BG95-M5	Cat M1/Cat NB2/EGPRS, Power Class 3
	BG95-M6	Cat M1/Cat NB2, Power Class 3
	BG95-MF	Cat M1/Cat NB2, Wi-Fi Positioning
BG77	BG77	Cat M1/Cat NB2
BG600L	BG600L-M3	Cat M1/Cat NB2/EGPRS

1.2. Using HTTP(S) AT Commands

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

Step 1: Configure the **<APN>**, **<username>**, **<password>** and other parameters of a PDP context with **AT+QICSGP**.

Step 2: Activate the PDP context with **AT+QIACT**. You can query the assigned IP address with **AT+QIACT?**.

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

Step 4: Configure the SSL context parameters with **AT+QSSLCFG**.

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

Step 6: Send an HTTP(S) request. **AT+QHTTPGET** can be used for sending an HTTP(S) GET request, while **AT+QHTTPPOST** or **AT+QHTTPOSTFILE** can be used for sending an HTTP(S) POST request. **AT+QHTTPPUT** or **AT+QHTTPPUTFILE** can be used for sending an HTTP(S) PUT request.

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

Step 8: Deactivate the PDP context with **AT+QIDEACT**.

NOTE

1. See ***document [1]*** for more information on **AT+QICSGP**, **AT+QIACT** and **AT+QIDEACT**.
2. See ***document [2]*** for more information on **AT+QSSLCFG**.

1.3. Description of HTTP(S) Header

1.3.1. Customize HTTP(S) Request Header

By default, an HTTP(S) request header is filled by the module automatically. It can also be customized by configuring **<request_header>** to 1 via **AT+QHTTPCFG** first, and then by inputting the HTTP(S) request header according to the following requirements:

1. Apply the HTTP(S) header syntax.
2. The value of a URI in the HTTP(S) request line and the “Host:” the header must be in line with the URL configured with **AT+QHTTPURL**.
3. The HTTP(S) request header must end with **<CR><LF>**.

A valid HTTP(S) POST request header is shown in the following example:

```
POST /processorder.php HTTP/1.1<CR><LF>
Host: 192.0.2.2:8011<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

By default, the HTTP(S) response header will not be outputted. Outputting of the HTTP(S) response header can be enabled by configuring **<response_header>** to 1 via **AT+QHTTPCFG**. The HTTP(S) response header will be outputted together with the HTTP(S) response body after executing **AT+QHTTPREAD** or **AT+QHTTPREADFILE**.

1.4. Description of Data Mode

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

- **Exit Data Mode**

Inputting **+++** or pulling the MAIN_DTR pin up can make the COM port exit the data mode.

To prevent the **+++** 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 the MAIN_DTR pin up, make sure to set **AT&D1** first.

- **Enter Data Mode**

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

NOTE

See ***document [3]*** for more information on **AT&D** and **ATO**.

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 the command line.
- **[...]** Optional parameter of a command or an optional part of TA information response. Square brackets do not appear on the command line. When an optional parameter is not given in a command, the new value equals its previous value or the default settings, unless otherwise specified.
- **Underline** Default setting of a parameter.

2.1.2. AT Command Syntax

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

Table 2: Types of AT Commands

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

2.2. Declaration of AT Command Examples

The AT command examples in this document are provided to help you learn about the use of the AT commands introduced herein. The examples, however, should not be taken as Quectel's recommendations or suggestions about how to design a program flow or what status to set the module into. Sometimes multiple examples may be provided for one AT command. However, this does not mean that there is a correlation among these examples, or that they should be executed in a given sequence. The URLs, domain names, IP addresses, usernames/accounts, and passwords (if any) in the AT command examples are provided for illustrative and explanatory purposes only, and they should be modified to reflect your actual usage and specific needs.

2.3. AT Commands Description

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

This command configures the parameters for an HTTP(S) server, such as configuring a PDP context ID, customizing the HTTP(S) request header, outputting the HTTP(S) response header, and querying SSL settings. When the Write Command only executes one parameter, it will query the current settings.

AT+QHTTPCFG Configure Parameters for HTTP(S) Server	
Test Command **AT+QHTTPCFG=?**	Response **+QHTTPCFG: "contextid",(**list of supported **<contextID>**s**)** **+QHTTPCFG: "requestheader",(**list of supported **<request_header>**s**)** **+QHTTPCFG: "responseheader",(**list of supported **<response_header>**s**)** **+QHTTPCFG: "sslctxid",(**list of supported **<sslctxID>**s**)** **+QHTTPCFG: "contenttype",(**list of supported **<content_type>**s**)** **+QHTTPCFG: "auth",("username:password")** **+QHTTPCFG: "custom_header",("custom_header")** **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: "auth",<username>:<password>** **+QHTTPCFG: "custom_header",<custom_header>**

	OK
Write Command **AT+QHTTPCFG="contextid"[,<contextID>]**	Response If the optional parameter is omitted, query the current setting: **+QHTTPCFG: "contextid",<contextID>** **OK** If the optional parameter is specified, configure the PDP context ID: **OK** Or **+CME ERROR: <result>**
Write Command **AT+QHTTPCFG="requestheader"[,<request_header>]**	Response If the optional parameter is omitted, query the current setting: **+QHTTPCFG: "requestheader",<request_header>** **OK** If the optional parameter is specified, configure whether to enable customization of HTTP(S) request header: **OK** Or **+CME ERROR: <result>**
Write Command **AT+QHTTPCFG="responseheader"[,<response_header>]**	Response If the optional parameter is omitted, query the current setting: **+QHTTPCFG: "responseheader",<response_header>** **OK** If the optional parameter is specified, configure whether to enable the outputting of the HTTP(S) response header: **OK** Or **+CME ERROR: <result>**
Write Command **AT+QHTTPCFG="sslctxid"[,<sslctxID>]**	Response If the optional parameter is omitted, query the current setting: **+QHTTPCFG: "sslctxid",<sslctxID>** **OK** If the optional parameter is specified, configure the SSL context ID used for HTTP(S): **OK**

	Or **+CME ERROR: <result>**
Write Command **AT+QHTTPCFG="contenttype"[,<content_type>]**	Response If the optional parameter is omitted, query the current setting: **+QHTTPCFG: "contenttype",<content_type>** **OK** If the optional parameter is specified, configure the data type of HTTP(S) body: **OK** Or **+CME ERROR: <result>**
Write Command **AT+QHTTPCFG="auth"[,<username>:<password>]**	Response If the optional parameter is omitted, query the current setting: **+QHTTPCFG: "auth","<username>:<password>"** **OK** If the optional parameter is specified, configure the username and password for logging in to the HTTP(S) server: **OK** Or **ERROR**
Write Command **AT+QHTTPCFG="custom_header"[,<custom_header>]**	Response If the optional parameter is omitted, query the current setting: **+QHTTPCFG:"custom_header",<custom_header>** **OK** If the optional parameter is specified, configure the user-defined HTTP(S) header: **OK** Or **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. The configuration is not saved.

Parameter

<contextID>	Integer type. PDP context ID. Range: 1–16. Default value: 1.

<request_header>	Integer type. Disable or enable customization of HTTP(S) request header. 0 Disable 1 Enable
<response_header>	Integer type. Disable or enable the outputting of 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 can be configured with **AT+QSSLCFG.** For more information on this command, 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
<username>	String type. Username for logging in to the HTTP(S) server.
<password>	String type. Password for logging in to the HTTP(S) server.
<custom_header>	String type. User-defined HTTP(S) request header.
<result>	Integer type. Result code. See ***Chapter 5*** for more information.

NOTE

AT+QHTTPCFG="auth" configures the username and password for logging in to the HTTP(S) server, but it is only applicable to basic authentication of the HTTP(S) server. For more information, see *RFC2616 14.8*.

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

This command sets the URL of an HTTP(S) Server. The URL must begin with “http://” or “https://”, which indicates that an HTTP or HTTPS server will be accessed.

AT+QHTTPURL Set URL of HTTP(S) Server	
Test Command **AT+QHTTPURL=?**	Response **+QHTTPURL: (**list of supported **<URL_length>**s**),(**list of supported **<timeout>**s**)** **OK**
Read Command **AT+QHTTPURL?**	Response **[+QHTTPURL: <URL>]** **OK**
Write Command	Response

AT+QHTTPURL=<URL_length>[,<timeout>]	a) If the parameter format is correct, but it is not sending HTTP(S) GET/POST/PUT requests at present: **CONNECT** TA switches to the transparent transmission mode, and then the URL can be inputted. When the total size of the inputted data reaches **<URL_length>**, TA will return to the command mode and report the following code: **OK** If **<timeout>** has been reached, but the received URL length is less than **<URL_length>**, TA will return to the command mode and report the following code: **+CME ERROR: <result>** b) If the parameter format is incorrect or other errors occur: **+CME ERROR: <result>**
Maximum Response Time	Determined by **<timeout>**
Characteristics	/

Parameter

<URL>	String type. HTTP(S) server URL.
<URL_length>	Integer type. URL length. Range: 1–3000. Unit: byte.
<timeout>	Integer type. Maximum time for inputting a URL. Range: 1–65535. Default value: 60. Unit: second.
<result>	Integer type. Result code. See ***Chapter 5*** for more information.

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

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

If **<request_header>**=1, after sending **AT+QHTTPGET**, **CONNECT** is outputted within 125 s to indicate successful establishment of the connection. If that is not the case, then **+CME ERROR: <result>** will be returned. It is recommended to wait for a specific period of time (**<rsptime>**) for **+QHTTPGET: <result>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is returned.

In **+QHTTPGET: <result>[,<httprspcode>[,<content_length>]]**, the **<httprspcode>** parameter can only be reported when **<result>** is 0. If HTTP(S) response header contains “Content-Length”, then **<content_length>** will be reported.

AT+QHTTPGET Send GET Request to HTTP(S) Server	
Test Command **AT+QHTTPGET=?**	Response **+QHTTPGET: (**list of supported **<rsptime>**s**),(**list of supported **<data_length>**s**),(**list of supported **<input_time>**s**)** **OK**
Write Command If **<request_header>**=0 (disabled to customize HTTP(S) request header) **AT+QHTTPGET[=<rsptime>]**	Response a) If the parameter format is correct and no other errors occur: **OK** When the module has received a response from the HTTP(S) server, it will report the following URC: **+QHTTPGET: <result>[,<httprspcode>[,<content_length>]]** b) If the parameter format is incorrect or other errors occur: **+CME ERROR: <result>**
Write Command If **<request_header>**=1 (enabled to customize HTTP(S) request header) **AT+QHTTPGET=<rsptime>,<data_length>[,<input_time>]**	Response a) If the connection to the HTTP(S) server has been established successfully: **CONNECT** TA switches to the transparent transmission mode, and then the HTTP(S) GET request header can be inputted. When the total size of the inputted data reaches **<data_length>**, TA will return to the 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: **+QHTTPGET: <result>[,<httprspcode>[,<content_length>]]** If the **<input_time>** has been reached, but the received data length is less than **<data_length>**, TA will return to the command mode and report the following code: **+CME ERROR: <result>** b) If the parameter format is incorrect or other errors occur: **+CME ERROR: <result>**

Maximum Response Time	Determined by **<rsptime>**
Characteristics	-

Parameter

<rsptime>	Integer type. Timeout value for the HTTP(S) GET response **+QHTTPGET: <result>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is returned. Range: 1–65535. Default value: 60. Unit: second.
<data_length>	Integer type. Length of HTTP(S) request information, including HTTP(S) request header and HTTP(S) request body. Range: 1–2048. Unit: byte.
<input_time>	Integer type. Maximum time for inputting HTTP(S) request information. Range: 1–65535. Default value: 60. Unit: second.
<httprspcode>	Integer type. HTTP(S) server response code. See ***Chapter 6*** for more information.
<request_header>	Integer type. Disable or enable customizing the HTTP(S) request header. 0 Disable 1 Enable
<content_length>	Integer type. Length of the HTTP(S) response body. Unit: byte.
<result>	Integer type. Result code. See ***Chapter 5*** for more information.

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

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

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

After sending **AT+QHTTPPOST**, **CONNECT** is outputted within 125 s to indicate successful establishment of the connection. If that is not the case, **+CME ERROR: <result>** will be returned. It is recommended to wait for a specific period of time (see the maximum response time below) for **+QHTTPPOST: <result>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is returned.

AT+QHTTPPOST Send POST Request to HTTP(S) Server via UART/USB	
Test Command **AT+QHTTPPOST=?**	Response **+QHTTPPOST: (**list of supported **<data_length>**s**),(**list of supported **<input_time>**s**),(**list of supported **<rsptime>**s**)** **OK**
Write Command If **<request_header>**=0 (disabled to customize HTTP(S) request header)	Response a) If the parameter format is correct, the connection to HTTP(S) server has been established successfully, and the

AT+QHTTPPOST=<data_length>[,<input_time>,<rsptime>]	HTTP(S) request header has been sent: **CONNECT** TA switches to the transparent transmission mode, and then the HTTP(S) POST body can be inputted. When the total size of the inputted data reaches **<data_length>**, TA will return to the 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: **+QHTTPPOST: <result>[,<httprspcode>[,<content_length>]]** If the **<input_time>** has been reached, but the received data length is less than **<data_length>**, TA will return to the command mode and report the following code: **+CME ERROR: <result>** b) If the parameter format is incorrect or other errors occur: **+CME ERROR: <result>**
Write Command If **<request_header>**=1 (enabled to customize HTTP(S) request header) **AT+QHTTPPOST=<data_length>[,<input_time>,<rsptime>]**	Response a) If the parameter format is correct and the connection to HTTP(S) server has been established successfully: **CONNECT** TA switches to the transparent transmission mode, and then the HTTP(S) POST header and body can be inputted. When the total size of the 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: **+QHTTPPOST: <result>[,<httprspcode>[,<content_length>]]** If the **<input_time>** has been reached, but the received data length is less than **<data_length>**, TA will return to command mode and report the following code: **+CME ERROR: <result>** b) If the parameter format is incorrect or other errors occur: **+CME ERROR: <result>**

Maximum Response Time	Determined by network and **<rsptime>**
Characteristics	-

Parameter

<data_length>	Integer type. If **<request_header>**=0, it indicates the length of HTTP(S) POST body. If **<request_header>**=1, it indicates the length of HTTP(S) POST request information, including the HTTP(S) POST request header and body. Range: 1–1024000. Unit: byte.
<input_time>	Integer type. Maximum time for inputting the HTTP(S) POST body or HTTP(S) POST request information. Range: 1–65535. Default value: 60. Unit: second.
<rsptime>	Integer type. Timeout value for the HTTP(S) POST response **+QHTTPPOST: <result>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is returned. Range: 1–65535. Default value: 60. Unit: second.
<httprspcode>	Integer type. HTTP(S) server response code. See ***Chapter 6*** for more information.
<request_header>	Integer type. Disable or enable customizing the HTTP(S) request header. 0 Disable 1 Enable
<content_length>	Integer type. Length of the HTTP(S) response body. Unit: byte.
<result>	Integer type. Result code. See ***Chapter 5*** for more information.

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

This command sends HTTP(S) POST request via a file. According to the **<request_header>** configuration in **AT+QHTTPCFG="requestheader"[,<request_header>]**, the file operated with **AT+HTTPPOSTFILE** has two different formats:

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

After executing **AT+QHTTPPOSFILE**, the module will report **+QHTTPPOSTFILE: <result>[,<httprspcode>[,<content_length>]]** to indicate the execution result. The **<httprspcode>** can only be reported when **<result>**=0. It is recommended to wait for a specific period of time (see the maximum response time below) for **+QHTTPPOSTFILE: <result>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is returned.

AT+QHTTPPOSTFILE Send POST Request to HTTP(S) Server via File	
Test Command **AT+QHTTPPOSTFILE=?**	Response **+QHTTPPOSTFILE: <file_name>,(**list of supported **<rsptime>**s**)**

	OK
Write Command **AT+QHTTPPOSTFILE=<file_name>[,<rsptime>]** If **<request_header>**=1, the specified file must contain both HTTP(S) request header and body.	Response a) If the parameter format is correct and the connection to HTTP(S) server has been established successfully: **OK** When the module has received a response from the HTTP(S) server, it will report the following URC: **+QHTTPPOSTFILE: <result>[,<httprspcode>[,<content_length>]]** b) If the parameter format is incorrect or other errors occur: **+CME ERROR: <result>**
Maximum Response Time	Determined by **<rsptime>**
Characteristics	/

Parameter

<file_name>	String type. File name. Max file name length: 80 bytes.
<rsptime>	Integer type. Timeout value for the HTTP(S) POST response **+QHTTPPOSTFILE: <result>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is returned. Range: 1–65535. Default value: 60. Unit: second.
<httprspcode>	Integer type. HTTP(S) server response code. See ***Chapter 6*** for more information.
<request_header>	Integer type. Disable or enable customizing the HTTP(S) request header. 0 Disable 1 Enable
<content_length>	Integer type. Length of HTTP(S) response body.
<result>	Integer type. Result code. See ***Chapter 5*** for more information.

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

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

- If **<request_header>**=0, HTTP(S) PUT body should be inputted via UART/USB port.
- If **<request_header>**=1, then both HTTP(S) PUT header and body should be inputted via UART/USB port.

After sending **AT+QHTTPPUT**, **CONNECT** is outputted within 125 s to indicate successful establishment of the connection. If that is not the case, **+CME ERROR: <result>** will be outputted. It is recommended to wait for a specific period of time (see the maximum response time below) for **+QHTTPPUT: <result>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is returned.

AT+QHTTPPUT Send PUT Request to HTTP(S) Server via UART/USB	
Test Command **AT+QHTTPPUT=?**	Response **+QHTTPPUT: (**list of supported **<data_length>**s**),(**list of supported **<input_time>**s**),(**list of supported **<rsptime>**s**)** **OK**
Write Command If **<request_header>**=0 (disabled to customize HTTP(S) request header) **AT+QHTTPPUT=<data_length>[,<input_time>,<rsptime>]**	Response a) If the parameter format is correct, the connection to HTTP(S) server has been established successfully, and HTTP(S) request header has been sent: **CONNECT** TA switches to transparent transmission mode, and the HTTP(S) PUT body can be inputted. When the total size of the 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: <result>[,<httprspcode>[,<content_length>]]** If the **<input_time>** has been reached, but the received data length is less than **<data_length>**, TA will return to command mode and report the following code: **+CME ERROR: <result>** b) If the parameter format is incorrect or other errors occur: **+CME ERROR: <result>**

Write Command If **<request_header>**=1 (enabled to customize HTTP(S) request header) **AT+QHTTPPUT=<data_length>[,<input_time>,<rsptime>]**	Response a) If the parameter format is correct and the connection to HTTP(S) server has been established successfully: **CONNECT** TA switches to the transparent transmission mode, and the HTTP(S) PUT header and body can be inputted. When the total size of the 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 the HTTP(S) server, it will report the following URC: **+QHTTPPUT: <result>[,<httprspcode>[,<content_length>]]** If the **<input_time>** has been reached, but the received data length is less than **<data_length>**, TA will return to command mode and report the following code: **+CME ERROR: <result>** b) If the parameter format is incorrect or other errors occur: **+CME ERROR: <result>**
Maximum Response Time	Determined by network and **<rsptime>**
Characteristics	This command takes effect immediately. The configuration is not saved.

Parameter

<data_length>	Integer type. If **<request_header>**=0, it indicates the length of HTTP(S) PUT body. If **<request_header>**=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. Maximum time for inputting HTTP(S) PUT body or HTTP(S) PUT request information. Range: 1–65535. Default value: 60. Unit: second.
<rsptime>	Integer type. Timeout value for the HTTP(S) PUT response **+QHTTPPUT: <result>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is returned. Range: 1–65535. Default value: 60. Unit: second.
<httprspcode>	Integer type. HTTP(S) server response code. See ***Chapter 6*** for more information.
<request_header>	Integer type. Disable or enable customizing the HTTP(S) request header. 0 Disable 1 Enable

<content_length>	Integer type. Length of HTTP(S) response body. Unit: byte.
<result>	Integer type. Result code. See ***Chapter 5*** for more information.

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

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

- If **<request_header>**=0, the file in file system will be the PUT body only.
- If **<request_header>**=1, the file in file system will be the PUT header and body.

After executing **AT+QHTTPPUTFILE**, the module will report **+QHTTPPUTFILE: <result>[,<httprspcode>[,<content_length>]]** to indicate the execution result. The **<httprspcode>** can only be reported when **<result>**=0. It is recommended to wait for a specific period of time (see the maximum response time below) for **+QHTTPPUTFILE: <result>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is returned.

AT+QHTTPPUTFILE Send PUT Request to HTTP(S) Server via File	
Test Command **AT+QHTTPPUTFILE=?**	Response **+QHTTPPUTFILE: <file_name>,(**list of supported **<rsptime>**s)**[,(**list of supported **<file_type>**s)**]** **OK**
Write Command **AT+QHTTPPUTFILE=<file_name>[,<rsptime>[,<file_type>]]**	Response a) If parameter format is correct and the connection to HTTP(S) server has been established successfully: **OK** When the module has received a response from HTTP(S) server, it will report the following URC: **+QHTTPPUTFILE: <result>[,<httprspcode>[,<content_length>]]** b) If parameter format is incorrect or other errors occur: **+CME ERROR: <result>**
Maximum Response Time	Determined by the network and **<rsptime>**
Characteristics	This command takes effect immediately. The configuration is not saved.

Parameter

<file_name>	String type. File name. Max file name length: 80 bytes.
<rsptime>	Integer type. Timeout value for the HTTP(S) POST response **+QHTTPPUTFILE: <result>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is returned. Range: 1–65535. Default: 60. Unit: second.
<file_type>	Integer type. File information to be sent. This parameter can only be omitted when **<file_type>**=0. 0 If **<request_header>**=0, it indicates request body If **<request_header>**=1, it indicates both request header and body 1 Request header (**<request_header>** must be set to 1) 2 Request body (**<request_header>** must be set to 1)
<httprspcode>	Integer type. HTTP(S) server response code. See ***Chapter 6*** for more information.
<request_header>	Integer type. Disable or enable customizing the HTTP(S) request header. 0 Disable 1 Enable
<content_length>	Integer type. Length of HTTP(S) response body. Unit: byte.
<result>	Integer type. Result code. See ***Chapter 5*** for more information.

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

This command retrieves the HTTP(S) response from an HTTP(S) server via the UART/USB port, after HTTP(S) GET/POST/PUT requests are sent. It must be executed after
+QHTTPGET: <result>[,<httprspcode>[,<content_length>]],
+QHTTPPOST: <result>[,<httprspcode>[,<content_length>]],
+QHTTPPOSTFILE: <result>[,<httprspcode>[,<content_length>]],
+QHTTPPUT: <result>[,<httprspcode>[,<content_length>]] or
+QHTTPPUTFILE: <result>[,<httprspcode>[,<content_length>]] is received.

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

	+QHTTPREAD: <result> If **<wait_time>** is reached or other errors occur, but the HTTP(S) response has not been outputted completely, it will report the following code: **+CME ERROR: <result>** b) If the parameter format is incorrect or other errors occur: **+CME ERROR: <result>**
Maximum Response Time	Determined by **<wait_time>**
Characteristics	The command takes effect immediately. The configuration is not saved.

Parameter

<wait_time>	Integer type. Maximum time between receiving two packets of data. Range: 1–65535. Default value: 60. Unit: second.
<result>	Integer type. Result code. See ***Chapter 5*** for more information.
<httprspcode>	Integer type. HTTP(S) server response code. See ***Chapter 6*** for more information.
<content_length>	Integer type. Length of HTTP(S) response body. Unit: byte.

2.3.9. AT+QHTTPREADFILE Store the Response from HTTP(S) Server to File

This command stores the HTTP(S) response from an HTTP(S) server to a specified file, after HTTP(S) GET/POST/PUT requests are sent, thus allowing users to retrieve the response information from the file. It must be executed after
+QHTTPGET: <result>[,<httprspcode>[,<content_length>]],
+QHTTPPOST: <result>[,<httprspcode>[,<content_length>]],
+QHTTPPOSTFILE: <result>[,<httprspcode>[,<content_length>]],
+QHTTPPUT: <result>[,<httprspcode>[,<content_length>]] or
+QHTTPPUTFILE: <result>[,<httprspcode>[,<content_length>]] is reported.

AT+QHTTPREADFILE Store the Response from HTTP(S) Server to File	
Test Command **AT+QHTTPREADFILE=?**	Response **+QHTTPREADFILE: <file_name>,(**list of supported **<wait_time>**s**)** **OK**
Write Command **AT+QHTTPREADFILE=<file_name>[,<wait_time>]**	Response a) If the parameter format is correct: **OK**

	When the response from the HTTP(S) server is read or **<wait_time>** is reached, it will report: **+QHTTPREADFILE: <result>** b) If the parameter format is incorrect or other errors occur: **+CME ERROR: <result>**
Maximum Response Time	Determined by **<wait_time>**
Characteristics	The command takes effect immediately. The configuration is not saved.

Parameter

<wait_time>	Integer type. Maximum time between receiving two packets of data. Range: 1–65535. Default value: 60. Unit: second.
<file_name>	String type. File name. Max file name length: 80 bytes.
<result>	Integer type. Result code. See ***Chapter 5*** for more information.
<httprspcode>	Integer type. HTTP(S) server response code. See ***Chapter 6*** for more information.
<content_length>	Integer type. Length of HTTP(S) response body. Unit: byte.

3 Examples

3.1. Access HTTP Server

3.1.1. Send HTTP GET Request and Read the Response

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

```
//Example of how to send an HTTP GET response.
AT+QHTTPCFG="contextid",1                  //Configure the PDP context ID as 1.
OK
AT+QHTTPCFG="responseheader",1             //Allow the output of HTTP response header.
OK
AT+QIACT?                                  //Query the list of currently activated contexts and their IP addresses.
OK                                         //No context activated currently.
AT+QICSGP=1,1,"UNINET","","",1             //Configure PDP context 1. Protocol type: IPv4. China Unicom APN: UNINET. Authentication method: PAP.
OK
AT+QIACT=1                                 //Activate context 1.
OK                                         //Activated successfully.
AT+QIACT?                                  //Query the list of currently activated contexts and their IP addresses.
+QIACT: 1,1,1,"10.7.157.1"

OK
AT+QHTTPURL=26,80                          //Set the URL of the HTTP server that will be accessed.
CONNECT
http://www.example.com.cn/                 //Input the URL whose length should be 26 bytes. (This URL is only an example. Input the correct URL in a practical test.)
OK
AT+QHTTPGET=80                             //Send the HTTP GET request with the maximum response time of 80 s.
OK
```

+QHTTPGET: 0,200,547256 //If the HTTP response header contains "Content-Length" information, then the **<content_length>** (547256) will be reported.

//Example of how to read an HTTP response.

//Solution 1: Read the HTTP response information and output it via the UART port.

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

CONNECT
HTTP/1.1 200 OK <CR><LF> //HTTP response header and body.
Content-Type: text/html<CR><LF>
Vary: Accept-Encoding<CR><LF>
X-Powered-By: shci_v1.03<CR><LF>
Server: nginx<CR><LF>
Date: Fri, 27 Dec 2013 02:21:43 GMT<CR><LF>
Last-Modified: Fri, 27 Dec 2013 02:20:01 GMT<CR><LF>
Expires: Fri, 27 Dec 2013 02:22:43 GMT<CR><LF>
Cache-Control: max-age=60<CR><LF>
Age: 1<CR><LF>
Content-Length: 547256<CR><LF>
X-Cache: HIT from xd33-85.example.com.cn<CR><LF>
<CR><LF>
<body>
OK

+QHTTPREAD: 0 //Successful reading of HTTP response header and body.

//Solution 2: Read the HTTP response information through storing it to a UFS file.

AT+QHTTPREADFILE="1.txt",80 //Read the HTTP response header and body through storing them to *1.txt*. The maximum time to wait for an HTTP session to be closed is 80 s.

OK

+QHTTPREADFILE: 0 //The HTTP response header and body have been stored successfully.

3.1.2. Send HTTP POST Request and Read the Response

3.1.2.1. HTTP POST Body Obtained from UART/USB

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

```
AT+QHTTPCFG="contextid",1                          //Configure the PDP context ID as 1.
OK
AT+QIACT?                                          //Query the list of currently activated
                                                   contexts and their IP addresses.
OK                                                 //No context activated currently.
AT+QICSGP=1,1,"UINET","","",1                      //Configure PDP context 1. Protocol type:
                                                   IPv4. China Unicom APN: UNINET.
                                                   Authentication method: PAP.
OK
AT+QIACT=1                                         //Activate context 1.
OK                                                 //Activated successfully.
AT+QIACT?                                          //Query the list of currently activated
                                                   contexts and their IP addresses.
+QIACT: 1,1,1,"172.22.86.226"

OK
AT+QHTTPURL=39,80                                  //Set the URL of the HTTP server that
                                                   will be accessed.
CONNECT
http://192.0.2.2:8300/processorder.php             //Input the URL whose length is 39 bytes.
                                                   (This URL is only an example. Input the
                                                   correct URL in a practical test.)
OK
AT+QHTTPPOST=48,80,80                              //Send an HTTP POST request. The
                                                   maximum input time and the maximum
                                                   response time are 80 s each.
CONNECT
Message=1111&Appleqty=2222&Orangeqty=3333&find=1   //Input the POST body whose length
                                                   should be 48 bytes. (The POST body is
                                                   only an example. Input the correct POST
                                                   body in a practical test.)
OK

+QHTTPPOST: 0,200,320                              //If the HTTP response header contains
                                                   "Content-Length" information, then the
                                                   <content_length> (320) will be reported.
```

```
AT+QHTTPREAD=80                                  //Read the HTTP response body and output it via a UART. The maximum time to wait for an HTTP session to be closed is 80 s.
CONNECT
HTTP/1.1 200 OK
Date: Tue, 06 Jul 2021 07:24:18 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.2.15
X-Powered-By: PHP/7.2.15
Access-Control-Allow-Origin: *
Content-Length: 320
Keep-Alive: timeout=60, max=9999
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded

<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 </p><p>Your order is as follows: </p>1111 message<br />2222  apple<br />3333 orange<br /></body>
</html>

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

3.1.2.2. HTTP POST Body Obtained from File System

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

```
AT+QHTTPCFG="contextid",1                        //Configure the PDP context ID as 1.
OK
AT+QIACT?                                        //Query the list of currently activated contexts and their IP addresses.
OK                                               //No context activated currently.
AT+QICSGP=1,1,"UINET","","",1                    //Configure PDP context 1. Protocol type: IPv4
                                                 China Unicom APN: UINET
                                                 Authentication method: PAP.
OK
```

```
AT+QIACT=1                                  //Activate context 1.
OK                                          //Activated successfully.
AT+QIACT?                                   //Query the list of currently activated contexts
                                            and their IP addresses.
+QIACT: 1,1,1,"172.22.86.226"

OK
AT+QHTTPURL=39,80                           //Set the URL of the HTTP server that will be
                                            accessed.
CONNECT
http://192.0.2.2:8300/processorder.php      //Input URL whose length is 39 bytes. (This URL is
                                            only an example. Input the correct URL in a
                                            practical test.)
OK
```

//POST the request information from a UFS file, and read the HTTP response information through storing it to a UFS file.

```
AT+QHTTPPOSTFILE="2.txt",80                 //Send the HTTP POST request. The  POST body
                                            is obtained from 2.txt. The maximum response time
                                            is 80 s.
OK

+QHTTPPOSTFILE: 0,200,295                   //HTTP POST request has been sent successfully.
                                            The HTTP response body can be read via either
                                            AT+QHTTPREAD or AT+QHTTPREADFILE.

AT+QHTTPREADFILE="3.txt",80                 //Read the HTTP response body through storing it
                                            to 3.txt. The maximum time to wait for an HTTP
                                            session to be closed is 80 s.
OK

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

3.1.3. Send HTTP PUT Request and Read the Response

3.1.3.1. HTTP PUT Body Obtained from UART/USB

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

```
AT+QHTTPCFG="contextid",1                   //Configure the PDP context ID as 1.
OK
AT+QHTTPCFG="contenttype",4                 //Configure the content type as application/json
```

```
OK
AT+QIACT?                                   //Query the list of currently activated contexts
                                            and their IP addresses.
OK
AT+QICSGP=1,1,"UNINET","","",1              //Configure the PDP context 1. Protocol type: IPv4
                                            China Unicom APN: UNINET. Authentication
                                            method: PAP.
OK
AT+QIACT=1                                  //Activate PDP context 1.
OK                                          //Activated successfully.
AT+QIACT?                                   //Query the list of currently activated contexts
                                            and their IP addresses.
+QIACT: 1,1,1,"172.22.86.226"

OK
AT+QHTTPURL=39,80                           //Set the URL that will be accessed.
CONNECT
http://192.0.2.2:8300/processorder.php      //Input the URL whose length is 39 bytes. (This URL
                                            is only an example. Input the correct URL in a
                                            practical test.)
OK
AT+QHTTPPUT=18,80,80                        //Send an HTTP PUT request. The HTTP PUT body
                                            is obtained via UART. The maximum input body
                                            time and the maximum response time are
                                            80 s each.
CONNECT
{"Message":"1234"}                          //Input the HTTP PUT body whose length is 18
                                            bytes. (The PUT body is only an example. Input
                                            the correct PUT body in a practical test.)
OK

+QHTTPPUT:  0,200,295                       //If the HTTP response header contains
                                            “Content-Length” information, then the <conte
                                            nt_length> information will be reported.
AT+QHTTPREAD=80                             //Read the HTTP response body and output it
                                            via a UART. The maximum time to wait for the
                                            HTTP session to be closed is 80 s.
CONNECT
<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/json
<p>Order processed at </p><p>Your order is as follows: </p>1234 message<br />  apple<br />
orange<br /></body>
</html>                                          //Output the HTTP response body.
OK

+QHTTPREAD: 0                                    //HTTP response body is output successfully.
```

3.1.3.2. HTTP PUT Body Obtained from File System

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

```
AT+QHTTPCFG="contextid",1                        //Configure the PDP context ID as 1.
OK
AT+QHTTPCFG=" contenttype",4                     //Configure the content type as application/json.
AT+QIACT?                                        //Query the list of currently activated contexts
                                                 and their IP addresses.
OK
AT+QICSGP=1,1,"UNINET","","",1                   //Configure PDP context 1. Protocol type: IPv4.
                                                 China Unicom APN: UNINET. Authentication
                                                 method: PAP.
OK
AT+QIACT=1                                       //Activate PDP context 1.
OK                                               //Activated successfully.
AT+QIACT?                                        //Query the list of currently activated contexts and
                                                 their IP addresses.
+QIACT: 1,1,1,"172.22.86.226"

OK
AT+QHTTPURL=39,80                                //Set the URL that will be accessed. Timeout
                                                 value: 80 s.
CONNECT
http://192.0.2.2:8300/processorder.php           //Input URL whose length is 39 bytes. (This URL is
                                                 only an example. Input the correct URL in a
                                                 practical test.)
OK

//PUT the request information from a UFS file, and read the HTTP response information and store it to a
UFS file.

AT+QHTTPPUTFILE="UFS:2.txt",80                   //Send an HTTP(S) PUT request. PUT body is
                                                 obtained from UFS:2.txt. The maximum response
                                                 time: 80 s.
```

```
OK

+QHTTPPUTFILE: 0,200,295                    //After an HTTP PUT request is sent successfully, AT+QHTTPREADFILE can be executed.
AT+QHTTPREADFILE="RAM:3.txt",80             //Read an HTTP response body and store it to UFS:3.txt. The maximum time to wait for an HTTP session to be closed is 80 s.
OK

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

3.2. Access HTTPS Server

3.2.1. Send HTTPS GET Request and Read the Response

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

```
//An example of how to send an HTTPS GET request.
AT+QHTTPCFG="contextid",1                   //Configure the PDP context ID as 1.
OK
AT+QHTTPCFG="responseheader",1              //Allow the output of the HTTPS response header.
OK
AT+QIACT?                                   //Query the list of currently activated contexts and their IP addresses.
OK
AT+QICSGP=1,1,"UNINET","","",1              //Configure PDP context 1. Protocol type: IPv4. China Unicom APN: UNINET. Authentication method: PAP.
OK
AT+QIACT=1                                  //Activate context 1.
OK                                          //Activated successfully.
AT+QIACT?                                   //Query the list of currently activated contexts and their IP addresses.
+QIACT: 1,1,1,"10.7.157.1"

OK
AT+QHTTPCFG="sslctxid",1                    //Set the SSL context ID as 1.
OK
AT+QSSLCFG="sslversion",1,1                 //Set the SSL verification as 1 which means TLSv1.0.
OK
AT+QSSLCFG="ciphersuite",1,0xFFFF           //Set the SSL cipher suite as 0xFFFF which means support all.
```

```
OK
AT+QSSLCFG="seclevel",1,0          //Set the SSL verify level as 0 which means CA certificate is not needed.
OK
AT+QHTTPURL=23,80                  //Set the URL of the HTTPS server that will be accessed.
CONNECT
https://www.example.com            //Input a URL whose length is 23 bytes. (This URL is only an example. Input the correct URL in a practical test.)
OK
AT+QHTTPGET=80                     //Send an HTTPS GET request. The maximum response time: 80 s.
OK

+QHTTPGET: 0,200,21472             //If the HTTPS response header contains "Content-Length" information, then the <content_length> (21472) will be reported.

//An example of how to read an HTTPS response.

//Solution 1: Read the HTTPS response information and output it via a UART.

AT+QHTTPREAD=80                    //Read the HTTPS response information and output it via a UART. The maximum time to wait for an HTTPS session to be closed is 80 s.
CONNECT                            //HTTPS response header and body.
HTTP/1.1 200 OK<CR><LF>
Server: nginx/1.2.7<CR><LF>
Date: Fri, 27 Dec 2013 02:38:27 GMT<CR><LF>
Content-Type: text/html; charset=GB18030<CR><LF>
Content-Length: 10750<CR><LF>
Connection: keep-alive<CR><LF>
<CR><LF>
<body>
OK

+QHTTPREAD: 0                      //Successful reading of HTTPS response header and body.

//Solution 2: Read the HTTPS response information through storing it to UFS file.

AT+QHTTPREADFILE="4.txt",80        //Read the HTTPS response header and body through storing it to 4.txt. The maximum time to wait for an HTTPS session to be closed is 80 s.
OK

+QHTTPREADFILE: 0                  //The HTTPS response header and body have been stored successfully.
```

3.2.2. Send HTTPS POST Request and Read the Response

3.2.2.1. HTTPS POST Body Obtained from UART/USB

The following examples show how to send an HTTPS POST request and retrieve the POST body via a 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+QIACT?                                    //Query the list of currently activated contexts and their IP addresses.
OK                                           //No context activated currently.
AT+QICSGP=1,1, "UINET","","",1               //Configure PDP context 1. Protocol type: IPv4. China Unicom APN: UNINET. Authentication method: PAP.
OK
AT+QIACT=1                                   //Activate context 1.
OK                                           //Activated successfully.
AT+QIACT?                                    //Query the list of currently activated contexts and their IP addresses.
+QIACT: 1,1,1,"172.22.86.226"

OK
AT+QHTTPCFG="sslctxid",1                     //Set the SSL context ID as 1.
OK
AT+QSSLCFG="sslversion",1,1                  //Set the SSL version as 1 which means TLSv1.0.
OK
AT+QSSLCFG="ciphersuite",1,0xFFFF            //Set the SSL cipher suite as 0xFFFF which means support all.
OK
AT+QSSLCFG="seclevel",1,2                    //Set the SSL verification level as 2 which means that a CA certificate, a client certificate and a client private key should all 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,"cacert.pem"          //Configure the path of CA certificate for
                                            SSL context 1.
OK
AT+QSSLCFG="clientcert",1,"clientcert.pem"  //Configure the path of client certificate for
                                            SSL context 1.
OK
AT+QSSLCFG="clientkey",1,"clientkey.pem"    //Configure the path of client private key
                                            for SSL context 1.
OK
AT+QHTTPURL=39,80                           //Set the URL of the HTTPS server that
                                            will be accessed.
CONNECT
https://192.0.2.2:8303/processorder.php     //Input the URL whose length is 39 bytes.
                                            (This URL is only an example. Input
                                            the correct URL in practical test.)
OK
AT+QHTTPPOST=48,80,80                       //Send the HTTPS POST request. The
                                            maximum input body time and the
                                            maximum response time are 80 s each.
CONNECT
Message=1111&Appleqty=2222&Orangeqty=3333&find=1  //Input the POST body whose length
                                                  should be 48 bytes. (This POST body is
                                                  only an example. Input the correct one in
                                                  a practical test.)
OK

+QHTTPPOST: 0,200,320                       //If the HTTPS response header contains
                                            "Content-Length" information, the
                                            <content_length> (285) will be reported.
AT+QHTTPREAD=80                             //Read the HTTPS response body and
                                            output it via a UART. The maximum time
                                            to wait for an HTTPS session to be closed
                                            is 80 s.
CONNECT
```

```
HTTP/1.1 200 OK
Date: Tue, 06 Jul 2021 07:35:55 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.2.15
X-Powered-By: PHP/7.2.15
Access-Control-Allow-Origin: *
Content-Length: 320
Keep-Alive: timeout=60, max=9999
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded

<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 </p><p>Your order is as follows: </p>1111 message<br />2222    apple<br
/>3333 orange<br /></body>
</html>

OK

+QHTTPREAD: 0                    //The HTTPS response body has been outputted successfully.
```

3.2.2.2. HTTPS POST Body Obtained from File System

The following examples show how to send an HTTPS POST request and retrieve the 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 list of currently activated
                                             contexts and their IP addresses.
OK                                           //No context activated currently.
AT+QICSGP=1,1, "UINET","","",1               //Configure PDP context 1. Protocol type:
                                             IPv4. China Unicom APN: UNINET.
                                             Authentication method: PAP.
OK
AT+QIACT=1                                   //Activate context 1.
OK                                           //Activated successfully.
AT+QIACT?                                    //Query the list of currently activated contexts
                                             and their IP addresses.
```

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

OK
AT+QHTTPCFG="sslctxid",1                    //Set the SSL context ID as 1.
OK
AT+QSSLCFG="sslversion",1,1                 //Set the SSL version as 1 which means
                                            TLSv1.0.
OK
AT+QSSLCFG="ciphersuite",1,0xFFFF           //Set the SSL cipher suite as 0xFFFF which
                                            means support all.
OK
AT+QSSLCFG="seclevel",1,2                   //Set the SSL verification level as 2 which
                                            means that a CA certificate, a client certificate
                                            and a client private key should all 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,"cacert.pem"          //Configure the path of CA certificate for SSL
                                            context 1.
OK
AT+QSSLCFG="clientcert",1,"clientcert.pem"  //Configure the path of client certificate for SSL
                                            context 1.
OK
AT+QSSLCFG="clientkey",1,"clientkey.pem"    //Configure the path of client private key for
                                            SSL context 1.
OK
```

```
AT+QHTTPURL=39,80                                   //Set the URL of HTTPS server that will be accessed.

CONNECT
https://192.0.2.2:8303/processorder.php             //Input the URL whose length should be 39 bytes. (This URL is only an example. Input the correct URL in a practical test.)

OK
```

//POST request information from UFS file, and read the HTTPS response information through storing it to a UFS file.

```
AT+QHTTPPOSTFILE="5.txt",80                         //Send the HTTPS POST request. The POST body is obtained from 5.txt. The maximum response time is 80 s.

OK
+QHTTPPOSTFILE: 0,200,320                           //The HTTPS POST request has been sent successfully. And then the HTTPS response body can be read via either AT+QHTTPREAD or AT+QHTTPREADFILE.
AT+QHTTPREADFILE="6.txt",80                         //Read the HTTPS response body through storing it to 6.txt. The maximum time to wait for an HTTPS session to be closed is 80 s.

OK

+QHTTPREADFILE: 0                                   //The HTTPS response body 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 PUT request and retrieve the PUT body via a UART port, as well as how to read an HTTPS PUT response.

```
AT+QHTTPCFG="contextid",1                           //Configure the PDP context ID as 1.
OK
AT+QHTTPCFG="contenttype",4                         //Configure the content type as application/json.

OK
AT+QIACT?                                           //Query the list of currently activated contexts and their IP addresses.

OK
AT+QICSGP=1,1,"UNINET","","",1                      //Configure PDP context 1. Protocol type: IPv4. China Unicom APN: UNINET. Authentication method: PAP.
```

```
OK
AT+QIACT=1                                  //Activate PDP context 1.
OK                                          //Activated successfully.
AT+QIACT?                                   //Query the list of currently activated
                                            contexts and their IP addresses.
+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",1,0xFFFF           //Set SSL cipher suite as 0xFFFF
                                            which means support all.
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 with AT+QFUPL.
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=39,80                           //Set the URL that will be accessed.
                                            Timeout value: 80 s
CONNECT
https://192.0.2.2:8303/processorder.php     //Input URL whose length is 39 bytes.
                                            (This URL is only an example. Input
                                            the correct URL in a practical test.)
OK
AT+QHTTPPUT=18,80,80                        //Send an HTTPS PUT request.
                                            The HTTPS PUT body is obtained
                                            from UART. The maximum input body
                                            time and the maximum response time
                                            are 80 s each.
CONNECT
{"Message":"1234"}                          //Input HTTPS PUT body whose
                                            length is 18 bytes. (This PUT body is
                                            only an example. Input the correct
                                            one in a practical test.)
```

```
OK

+QHTTPPUT: 0,200,295                    //If the HTTPS response header contains content length information, then the <content_length> information will be reported.
AT+QHTTPREAD=80                         //Read HTTPS response body and output it via a UART. The maximum time to wait for an HTTPS session to be closed is 80 s.
CONNECT                                 //Successful reading of the HTTPS response body.

<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/json
<p>Order processed at </p><p>Your order is as follows: </p>1234 message<br />  apple<br />
orange<br /></body>
</html>

OK

+QHTTPREAD: 0                           //HTTPS response body 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 retrieve the HTTPS PUT body from a file system, as well as how to store an HTTPS PUT response to file system.

```
AT+QHTTPCFG="contextid",1               //Configure the PDP context ID as 1.
OK
AT+QHTTPCFG="contenttype",4             //Configure the content type as application/json.
OK
AT+QIACT?                               //Query the list of currently activated contexts and their IP addresses.

OK
AT+QICSGP=1,1,"UNINET","","",1          //Configure PDP context 1. Protocol type: IPv4 China Unicom APN: UNINET. Authentication method: PAP.
OK
```

```
AT+QIACT=1                                  //Activate PDP context 1.
OK                                          //Activated successfully.
AT+QIACT?                                   //Query the list of currently activated contexts and
                                            their IP addresses.
+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",1,0xFFFF           //Set SSL cipher suite as 0xFFFF, which means
                                            support all.
OK
AT+QSSLCFG="seclevel",1,2                   //Set SSL verify level as 2, which means that a CA
                                            certificate, client certificate and client private key
                                            should be uploaded with AT+QFUPL.
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=39,80                           //Set the URL that will be accessed. Timeout
                                            value: 80 s.
CONNECT
https://192.0.2.2:8303/processorder.php     //Input URL whose length is 39 bytes. (This URL is
                                            only an example. Input the correct URL in a practical
                                            test.)
OK

//PUT request information from UFS file, and read HTTPS response information and store it to UFS file.
AT+QHTTPPUTFILE="UFS:5.txt",80              //Send HTTPS PUT request. HTTPS PUT body
                                            is obtained from UFS:5.txt. The maximum response
                                            time: 80 s.
OK

+QHTTPPUTFILE: 0,200,295                    //After HTTPS PUT request is sent successfully,
                                            AT+QHTTPREADFILE can be executed.
AT+QHTTPREADFILE="UFS:6.txt",80             //Read HTTPS response body and store it to
                                            UFS:6.txt. The maximum time to wait for HTTPS
                                            session to be closed is 80 s.
OK
```

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

4 Error Handling

4.1. Executing HTTP(S) AT Command Failure

If **ERROR** response is received from the module after executing HTTP(S) AT commands, check whether the (U)SIM card has been inserted and whether **+CPIN: READY** is returned when executing **AT+CPIN?**.

4.2. PDP Activation Failure

In case of failure to activate a PDP context with **AT+QIACT**, check the following configurations:

1. Query the PS domain status with **AT+CEREG?** (for LTE Cat M1 and Cat NB2 networks) or **AT+CGREG?** (for EGPRS network) and make sure the PS domain has been registered.
2. Query the PDP context parameters with **AT+QICSGP=<contextID>** and make sure that the APN of the specified PDP context has been set.
3. Make sure the specified PDP context ID is neither used by PPP nor activated with **AT+CGACT**.
4. The module supports maximum three PDP contexts activated simultaneously under LTE Cat M1/ EGPRS and maximum two under LTE Cat NB2.

If all above configurations are correct, but activating the PDP context with **AT+QIACT** still fails, reboot the module. After rebooting, check the configurations above at least three times in 10-minute intervals to avoid frequent module rebooting.

4.3. DNS Parse Failure

If **+CME ERROR: 714** (714: HTTP(S) DNS error) is returned after executing **AT+QHTTPGET**, **AT+QHTTPPOST**, **AT+QHTTPPOSTFILE**, **AT+QHTTPPUT** and **AT+QHTTPPUTFILE**, check the following:

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

3. Query the address of the DNS server with **AT+QIDNSCFG** to make sure the address is not null or "0.0.0.0".

If the DNS server address is null or "0.0.0.0", there are three solutions:

1. Reassign a valid DNS server address with **AT+QIDNSCFG**.
2. Deactivate the PDP context with **AT+QIDEACT**, and then re-activate the PDP context with **AT+QIACT**.
3. If the module has registered to an NB-IoT network, execute **AT+QCFG="nccconf",101** to enable ePCO, and then re-register to the network with **AT+CFUN=0/1** or reboot the module.

4.4. Entering Data Mode Failure

If **+CME ERROR: 704** (704: HTTP(S) UART busy) is returned after executing **AT+QHTTPURL**, **AT+QHTTPGET**, **AT+QHTTPPOST**, **AT+QHTTPPUT** and **AT+QHTTPREAD**, check if there are several ports in data mode, since the module only supports one port in data mode at a time. If there are, re-execute these commands when all but one of the ports have exited the data mode.

4.5. Sending GET/POST/PUT Requests Failure

If a failed response is received after executing **AT+QHTTPGET**, **AT+QHTTPPOST**, **QHTTPPOSTFILE**, **AT+QHTTPPUT** and **AT+QHTTPPUTFILE**, check the following configurations:

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

If all above configurations are correct, but sending GET/POST/PUT requests with **AT+QHTTPGET**, **AT+QHTTPPOST**, **AT+QHTTPPOSTFILE**, **AT+QHTTPPUT** and **AT+QHTTPPUTFILE** still fails, deactivate the PDP context with **AT+QIDEACT** and then re-activate it with **AT+QIACT** to resolve this issue. If activating the PDP context fails, see ***Chapter 4.2***.

4.6. Reading Response Failure

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

+QHTTPGET: <result>[,<httprspcode>[,<content_length>]]

+QHTTPPOST: <result>[,<httprspcode>[,<content_length>]]
+QHTTPPOSTFILE: <result>[,<httprspcode>[,<content_length>]]
+QHTTPPUT: <result>[,<httprspcode>[,<content_length>]]
+QHTTPPUTFILE: <result>[,<httprspcode>[,<content_length>]]

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

5 Summary of Result Codes

The result code **<result>** indicates a result related to mobile equipment or network operation. The meaning of **<result>** is presented in the following table.

Table 3: Summary of Result Codes

<result>	Meaning
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/PUT 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 meaning of **<httprspcode>** is presented in the following table.

Table 4: Summary of HTTP(S) Response Codes

<httprspcode>	Meaning
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_BG95&BG77&BG600L_Series_TCP(IP)_AT_Commands_Manual
[2] Quectel_BG95&BG77&BG600L_Series_SSL_AT_Commands_Manual
[3] Quectel_BG95&BG77&BG600L_Series_AT_Commands_Manual
[4] Quectel_BG95&BG77&BG600L_Series_FILE_Application_Note

Table 6: Terms and Abbreviations

Abbreviation	Description
APN	Access Point Name
CA	Certification Authority
COM port	Communication Port
CR	Carriage Return
DNS	Domain Name Server
DTR	Data Terminal Ready
EGPRS	Enhanced General Packet Radio Service
ePCO	Extended Protocol Configuration Options
GMT	Greenwich Mean Time
HTTP	Hyper Text Transport Protocol
HTTPS	Hyper Text Transfer Protocol Secure

ID	Identifier
IP	Internet Protocol
LF	Line Feed (a new line)
LTE	Long-Term Evolution
LTE Cat	LTE Category
PAP	Password Authentication Protocol
PDP	Packet Data Protocol
PPP	Point-to-Point Protocol
PS	Packet Switch
SIM	Subscriber Identity Module
SSL	Security Socket Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UART	Universal Asynchronous Receiver/Transmitter.
UFS	Universal Flash Storage
URC	Unsolicited Result Code
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal Serial Bus
USIM	Universal Subscriber Identity Module