EC2x&EG9x&EG2x-G&EM05 Series

HTTP(S) Application Note

LTE Standard Module Series

Version: 1.2

Date: 2022-06-07

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
-	2017-11-20	Duke XIN/ Haley HUANG	Creation of the document
1.0	2017-11-20	Duke XIN/ Haley HUANG	First official release
1.1	2020-05-15	Domingo DENG	1. Added the applicable modules (Chapter 1.1). 2. Extended the value of parameter <content_type> of AT+QHTTPCFG (Chapter 2.1). 3. Updated the description of AT+QHTTPGETEX (Chapter 2.4). 4. Added the parameter <file_type> for AT+QHTTPPOSTFILE (Chapter 2.6).
1.2	2022-06-07	Larson LI	1. Added AT+QHTTPCFG="contenttype" and AT+QHTTPCFG="custom_header" AT+QHTTPCFG="windowsize" AT+QHTTPCFG="closewaittime" AT+QHTTPCFG="auth"(Chapter 2.1). 2. Updated the description of <closedind> of AT+QHTTPCFG="closed/ind" (Chapter 2.1). 3. Added AT+QHTTPPUT (Chapter 2.7). 4. Added AT+QHTTPPUTFILE (Chapter 2.8). 5. Added AT+QHTTPCFGEX (Chapter 2.11). 6. Added AT+QHTTPSEND (Chapter 2.12). 7. Added examples about sending HTTP PUT requests (Chapter 3.1.3). 8. Added examples about sending HTTP SEND requests (Chapter 3.1.4). 9. Added examples about sending HTTPS PUT requests (Chapter 3.2.3). 10. Added examples about sending HTTPS SEND requests (Chapter 3.2.4).

Contents

Table Index

1 Introduction

Quectel LTE Standard EC2x, EG9x series, EG2x-G and EM05 series modules provide HTTP(S) applications to HTTP(S) server. This document is a reference guide to all the AT commands defined for HTTP(S).

1.1. Applicable Modules

Table 1: Applicable Modules

Module Series	Module
EC2x	EC21 Series
	EC25 Series
	EC20-CE
EG2x-G	EG21-G
	EG25-G
EG9x	EG91 Series
	EG95 Series
EM05	EM05 Series

1.2. AT Command Introduction

1.2.1. 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.

1.2.2. 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 range 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.

1.3. 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.

1.4. The Process of Using HTTP(S) AT Commands

With TCP/IP AT commands you can configure a PDP context, activate/deactivate the PDP context, and query the 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 **<APN>**, **<username>**, **<password>** and other parameters of a PDP context with **AT+QICSGP**. Please refer to ***document [1]*** for details. If QoS settings need to be updated, configure them with **AT+CGQMIN**, **AT+CGEQMIN**, **AT+CGQREQ** and **AT+CGEQREQ**. For more details, please refer to ***document [2]***.

Step 2: Activate the PDP context with **AT+QIACT**, then the assigned IP address can be queried with **AT+QIACT?**. Please refer to ***document [1]*** for details.

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

Step 4: Configure SSL context parameters with **AT+QSSLCFG**. For more details, please refer to ***doucment [3]***.

Step 5: Set HTTP(S) URL with **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 GET request to HTTP(S) server to get data with specified range; **AT+QHTTPPOST**, **AT+QHTTPSEND** or **AT+QHTTPPOSTFILE** can be used for sending HTTP(S) POST request; **AT+QHTTPPUT** or **AT+QHTTPPUTFILE** can be used for sending HTTP(S) PUT request.

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

Step 8: Deactivate the PDP context with **AT+QIDEACT**. For more details, please refer to ***document [1]***.

1.5. Description of HTTP(S) Header

1.5.1. Customize HTTP(S) Request Header

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

- Apply HTTP(S) request header syntax.
- The value of a URL in HTTP(S) request line and the “Host:” header must be in line with the URL configured with **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>
Message=1111&Appleqty=2222&Orangeqty=3333&find=1
```

1.5.2. Output HTTP(S) Response Header

HTTP(S) response header will not be outputted automatically. 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 with the HTTP(S) response body after executing **AT+QHTTPREAD** or **AT+QHTTPREADFILE**.

1.6. Description of Data Mode

The COM port of the above applicable EC2x, EG9x series, EG2x-G and EM05 series modules 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.

- **Exit Data Mode**

Inputting **+++** or pulling up the DTR pin can make the COM port exit 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 DTR pin up, make sure to set **AT&D1** first.

- **Enter Data Mode**

To enter the data mode, execute **AT+QHTTPURL**, **AT+QHTTPPOST**, **AT+QHTTPPUT**, **AT+QHTTPSEND** and **AT+QHTTPREAD**. If you input **+++** or pull the DTR pin to make the port exit 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**.

2 Description of HTTP(S) AT Commands

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

The command configures the parameters for 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. If the Write Command omits the optional parameter(s), 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**)** **+QHTTPCFG: "windowsize",(**range of supported **<window_size>**s**)** **+QHTTPCFG: "closewaittime",(**range of supported **<close_wait_time>**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: "rspout/auto",<auto_outrsp>** **+QHTTPCFG: "closed/ind",<closedind>**

	+QHTTPCFG: "windowsize",<window_size> **+QHTTPCFG: "closewaittime",<close_wait_time>** **+QHTTPCFG: "auth","<username>:<password>"** **+QHTTPCFG: "custom_header","<custom_value>"** **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: <err>**
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, enable/disable to customize HTTP(S) request header: **OK** Or **+CME ERROR: <err>**
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, enable/disable to output HTTP(S) response header: **OK** Or **+CME ERROR: <err>**
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: <err>**
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: <err>**
Write Command **AT+QHTTPCFG="rspout/auto"[,<auto_outrsp>]**	Response If the optional parameter is omitted, query the current setting: **+QHTTPCFG: "rspout/auto",<auto_outrsp>** **OK** If the optional parameter is specified, enable/disable auto output of HTTP(S) response data: **OK** Or **+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, enable/disable the report of HTTP(S) session closing URC **+QHTTPURC: "closed"**: **OK** Or **+CME ERROR: <err>**
Write Command **AT+QHTTPCFG="windowsize"[,<window_size>]**	Response If the optional parameter is omitted, query the current settings: **+QHTTPCFG: "windowsize",<window_size>** **OK**

	If the optional parameter is specified, configure the size of HTTP socket sliding window: **OK** Or **ERROR**
Write Command **AT+QHTTPCFG="closewaittime"[,<close_wait_time>]**	Response If the optional parameter is omitted, query the current settings: **+QHTTPCFG: "closewaittime",<close_wait_time>** **OK** If the optional parameter is specified, configure the wait time for closing HTTP socket: **OK** Or **ERROR**
Write Command **AT+QHTTPCFG="auth"[,<username>:<password>]**	Response If the optional parameter is omitted, query the current settings: **+QHTTPCFG: "auth","<username>:<password>"** **OK** If the optional parameter is specified, configure the username and password for logging in HTTP: **OK** Or **ERROR**
Write Command **AT+QHTTPCFG="custom_header"[,"<custom_value>"]**	Response If the optional parameter is omitted, query the current settings: **+QHTTPCFG:"custom_header","<custom_value>"** **OK** If the optional parameter is specified, configure the user defined HTTP header: **OK** Or **ERROR**
Maximum Response Time	200 ms
Characteristics	The command takes effect immediately. The configurations will not be 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 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 with **AT+QSSLCFG**. For details, please refer to ***document [3]***.
<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 data. If auto output of HTTP(S) response data is enabled, then **AT+QHTTPREAD** and **AT+QHTTPREADFILE** execution will fail. 0 Disable 1 Enable
<closedind>	Integer type. Disable or enable the report of HTTP(S) session closing URC **+QHTTPURC: "closed"**. 0 Disable 1 Enable
<username>	String type. The username for logging in HTTP.
<password>	String type. The password for logging in HTTP.
<window_size>	Integer type. The size of HTTP socket sliding window. Range: 1–4294967295. Default value: 16384 (16 KB). Unit: byte.
<close_wait_time>	Integer type. The wait time for closing HTTP socket. Range: 0–4294967295. Default value: 60000. Unit: ms.
<custom_header>	String type. User defined HTTP header.
<err>	Error code. Please refer to ***Chapter 5***.

NOTE

AT+QHTTPCFG="auth" can be used to configure the username and password for logging in HTTP, but it is only applicable for basic authentication of HTTP server. For more details, please refer to *RFC2616 14.8*.

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

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: (**range of supported **<URL_length>**s**),(**range of supported **<timeout>**s**)** **OK**
Read Command **AT+QHTTPURL?**	Response **[+QHTTPURL: <URL><CR><LF>]** **OK**
Write Command **AT+QHTTPURL=<URL_length>[,<timeout>]**	Response a) If the parameter format is correct, and it is not sending HTTP(S) GET/POST requests: **CONNECT** TA switches to 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 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 command mode and report the following code: **+CME ERROR: <err>** b) If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	Determined by **<timeout>**
Characteristics Description	The command takes effect immediately. The configurations will not be saved.

Parameter

<URL_length>	Integer type. The length of URL. Range: 1–2048. Unit: byte.
<timeout>	Integer type. The maximum time for inputting URL. Range: 1–65535. Default value: 60. Unit: second.
<err>	Error code. Please refer to ***Chapter 5***.

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

According to the configured **<request_header>** parameter in **AT+QHTTPCFG="requestheader"[,<request_header>]**, **AT+QHTTPGET** has two different formats.

If **<request_header>** is set to 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: <err>** will be returned. It is recommended to wait for a specific period of time (**<rsptime>**) for **+QHTTPGET: <err>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is reported.

In **+QHTTPGET: <err>[,<httprspcode>[,<content_length>]]**, the **<httprspcode>** can only be reported when **<err>** equals 0. If HTTP(S) response header contains CONTENT-LENGTH information, 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**
If **<request_header>** equals 0 (disable to customize HTTP(S) request header) Write Command **AT+QHTTPGET[=<rsptime>]**	Response a) 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>]]** b) If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
If **<request_header>** equals 1 (enable to customize HTTP(S) GET request header) Write Command **AT+QHTTPGET=<rsptime>,<data_length>[,<input_time>]**	Response a) If HTTP(S) server is connected successfully: **CONNECT** TA switches to 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 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 received data length is less than **<data_length>**, TA will return to command mode and report the following code: **+CME ERROR: <err>** b) If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	Determined by **<rsptime>**
Characteristics Description	The command takes effect immediately. The configurations will not be saved.

Parameter

<rsptime>	Integer type. Timeout value for the HTTP(S) GET response **+QHTTPGET: <err>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is returned. Range: 1–65535. Default value: 60. Unit: second.
<data_length>	Integer type. The 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. The maximum time for inputting HTTP(S) request information, including HTTP(S) request header and HTTP(S) request body. Range: 1–65535. Default value: 60. Unit: second.
<err>	Error code. Please refer to ***Chapter 5***.
<httprspcode>	Integer type. HTTP server response code. Please refer to ***Chapter 6***.
<request_header>	Integer type. Disable or enable to customize HTTP(S) request header. Please refer to ***Chapter 2.1***.
<content_length>	Integer type. The length of HTTP(S) response body. Unit: byte.

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

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

AT+QHTTPGETEX Send GET Request to HTTP(S) Server to Get Data With Specified Range	
Test Command **AT+QHTTPGETEX=?**	Response **+QHTTPGET: (**range of supported **<rsptime>**s**),<start_postion>,<read_len>** **OK**
Write Command **AT+QHTTPGETEX=<rsptime>,<start_position>,<read_len>**	Response a) 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>]]** b) If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	Determined by **<rsptime>**
Characteristics Description	The command takes effect immediately. The configurations will not be saved.

Parameter

<rsptime>	Integer type. Timeout value for the HTTP(S) GET response **+QHTTPGET: <err>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is returned. Range: 1–65535. Default value: 60. Unit: second.
<start_postion>	Integer type. The start position of the data that the HTTP(S) client wants to get.
<read_len>	Integer type. The length of the data that the HTTP(S) client wants to get.
<err>	Error code. Please refer to ***Chapter 5***.
<httprspcode>	Integer type. HTTP server response code. Please refer to ***Chapter 6***.
<content_length>	Integer type. The length of HTTP(S) response body. Unit: byte.

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

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

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

It is recommended to wait for a specific period of time (refer to the maximum response time below) for **+QHTTPPOST: <err>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is reported. The **<httprspcode>** can only be reported when **<err>** equals 0.

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**
If **<request_header>** equals 0 (disable to customize HTTP(S) request header) Write Command **AT+QHTTPPOST=<data_length>[,<input_time>,<rsptime>]**	Response a) If the parameter format is correct, the connection to HTTP(S) server has been established successfully, and the HTTP(S) request header has been sent: **CONNECT** TA switches to 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 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 the **<input_time>** has reached, but the received length of

	data is less than **<data_length>**, TA will return to command mode and report the following code: **+CME ERROR: <err>** b) If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
If **<request_header>** equals 1 (enable to customize HTTP(S) request header) Write Command **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 response from HTTP(S) server, it will report the following URC: **+QHTTPPOST: <err>[,<httprspcode>[,<content_length>]]** If the **<input_time>** has 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>** b) If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	Determined by network and **<rsptime>**
Characteristics Description	The command takes effect immediately. The configurations will not be saved.

Parameter

<data_length>	Integer type. If **<request_header>** is 0, it indicates the length of HTTP(S) POST body. If **<request_header>** is 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. The maximum time for inputting HTTP(S) POST body or HTTP(S) POST request information. Range: 1–65535. Default: 60. Unit: second.
<rsptime>	Integer type. Timeout value for the HTTP(S) POST response **+QHTTPPOST: <err>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is

	returned. Range: 1–65535. Default value: 60. Unit: second.
<err>	Error code. Please refer to ***Chapter 5***.
<httprspcode>	Integer type. HTTP server response code. Please refer to ***Chapter 6***.
<request_header>	Integer type. Disable or enable to customize HTTP(S) request header. Please refer to ***Chapter 2.1***.
<content_length>	Integer type. The length of HTTP(S) response body. Unit: byte.

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

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

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

After executing **AT+QHTTPPOSTFILE**, the module will report **+QHTTPPOSTFILE: <err>[,<httprspcode>[,<content_length>]]** information to indicate the execution result. The **<httprspcode>** parameter can only be reported when **<err>** equals 0.

It is recommended to wait for a specific period of time (refer to the maximum response time below) for **+QHTTPPOSTFILE: <err>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is reported.

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 **<file_type>**s**)]** **OK**
Write Command **AT+QHTTPPOSTFILE=<file_name>[,<rsptime>[,<file_type>]]** If **<request_header>** equals 1, the specified file must contain HTTP(S) request header information.	Response a) 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: **+QHTTPPOSTFILE: <err>[,<httprspcode>[,<content_length>]]** b) 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 will not be saved.

Parameter

<file_type>	String type. HTTP(S) sending files in segments. 0 Send the current file directly 1 Record the file name to be sent 2 Send file 1 and 2 in order
<file_name>	String type. File name. The max length of file name is 80 bytes.
<rsptime>	Integer type. Timeout value for the HTTP(S) POST response **+QHTTPPOSTFILE: <err>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is returned. Range: 1–65535. Default: 60. Unit: second.
<err>	Error code. Please refer to ***Chapter 5***.
<httprspcode>	Integer type. HTTP server response code. Please refer to ***Chapter 6***.
<request_header>	Integer type. Disable or enable to customize HTTP(S) request header. Please refer to ***Chapter 2.1***.
<content_length>	Integer type. The length of HTTP(S) response body. Unit: byte.

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

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

- If **<request_header>** is set to 0, HTTP(S) PUT body should be inputted via UART/USB port.
- If **<request_header>** is set to 1, 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: <err>** will be returned.

It is recommended to wait for a specific period of time (refer to the maximum response time below) for **+QHTTPPUT: <err>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is reported. The **<httprspcode>** can only be reported when **<err>** equals 0.

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**
If **<request_header>** equals 0 (disable to customize HTTP(S) request header) Write Command **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 the HTTP(S) request header has been sent: **CONNECT** TA switches to transparent transmission mode, and then 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 response from HTTP(S) server, it will report the following URC: **+QHTTPPUT: <err>[,<httprspcode>[,<content_length>]]** If the **<input_time>** has reached, but the received length of data is less than **<data_length>**, TA will return to command mode and report the following code: **+CME ERROR: <err>** b) If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
If **<request_header>** equals 1 (enable to customize HTTP(S) request header) Write Command **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 then 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 response from HTTP(S) server, it will report the following URC:

	+QHTTPPUT: <err>[,<httprspcode>[,<content_length>]] If the **<input_time>** has 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>** b) 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 will not be saved.

Parameter

<data_length>	Integer type. If **<request_header>** is 0, it indicates the length of HTTP(S) PUT body. If **<request_header>** is 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. The 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 **+QHTTPPOST: <err>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is returned. Range: 1–65535. Default value: 60. Unit: second.
<err>	Error code. Please refer to ***Chapter 5***.
<httprspcode>	Integer type. HTTP server response code. Please refer to ***Chapter 6***.
<request_header>	Integer type. Disable or enable to customize HTTP(S) request header. Please refer to ***Chapter 2.1***.
<content_length>	Integer type. The length of HTTP(S) response body. Unit: byte.

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

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

After executing **AT+QHTTPPUTFILE**, the module will report **+QHTTPPUTFILE:**

<err>[,<httprspcode>[,<content_length>]] information to indicate the execution result. The **<httprspcode>** can only be reported when **<err>** equals 0.

It is recommended to wait for a specific period of time (refer to the maximum response time below) for **+QHTTPPUTFILE: <err>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is reported.

AT+QHTTPPUTFILE	**Send PUT Request to HTTP(S) Server via File**
Test Command **AT+QHTTPPUTFILE=?**	Response **+QHTTPPUTFILE: <file_name>,(**range of supported **<rsptime>**s**)[,(**range of supported **<file_type>**s**)]** **OK**
Write Command **AT+QHTTPPOSTFILE=<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 response from HTTP(S) server, it will report the following URC: **+QHTTPPUTFILE: <err>[,<httprspcode>[,<content_length>]]** b) If 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 will not be saved.

Parameter

<file_name>	String type. File name. The maximum length of file name is 80 bytes.
<rsptime>	Integer type. Timeout value for the HTTP(S) POST response **+QHTTPPOSTFILE: <err>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is returned. Range: 1–65535. Default: 60. Unit: second.
<file_type>	Integer type. The file information to be sent. This parameter can only be omitted when **<request_header>**=0. 0 If **<request_header>** is set to 0, it indicates request body If **<request_header>** is set to 1, it indicates request header and request body 1 Request header, **<request_header>** must be set to 1 2 Request body, **<request_header>** must be set to 1
<err>	Error code. Please refer to ***Chapter 5***.

<httprspcode>	Integer type. HTTP server response code. Please refer to ***Chapter 6***.
<request_header>	Integer type. Disable or enable to customize HTTP(S) request header. Please refer to ***Chapter 2.1***.
<content_length>	Integer type. The length of HTTP(S) response body. Unit: byte.

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

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

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 a) If the parameter format is correct and the HTTP(S) response is read successfully: **CONNECT** <Output HTTP(S) response information> **OK** **+QHTTPREAD: <err>** 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: <err>** b) 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 will not be saved.

Parameter

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

2.10. AT+QHTTPREADFILE Read Response from HTTP(S) Server via File

This command retrieves 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: <err>[,<httprspcode>[,<content_length>]],
+QHTTPPOST: <err>[,<httprspcode>[,<content_length>]],
+QHTTPPUT: <err>[,<httprspcode>[,<content_length>]],
+QHTTPPOSTFILE: <err>[,<httprspcode>[,<content_length>]] or
+QHTTPPUTFILE: <err>[,<httprspcode>[,<content_length>]] is received.

AT+QHTTPREADFILE Read Response from HTTP(S) Server via File	
Test Command **AT+QHTTPREADFILE=?**	Response **+QHTTPREADFILE: <file_name>,(**range of supported **<wait_time>**s**)** **OK**
Write Command **AT+QHTTPREADFILE=<file_name>[,<wait_time>]**	Response a) If the parameter format is correct: **OK** 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: **+QHTTPREADFILE: <err>** b) 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 will not be saved.

Parameter

<wait_time>	Integer type. The maximum interval time between receiving two data packets. Range: 1–65535. Default value: 60. Unit: second.
<file_name>	String type. File name. The maximum length of the file name is 80 bytes.
<err>	Error code. Please refer to ***Chapter 5***.
<httprspcode>	Integer type. HTTP server response code. Please refer to ***Chapter 6***.
<content_length>	Integer type. The length of HTTP(S) response body. Unit: byte.

2.11. AT+QHTTPCFGEX Configure Files and Parameters to be Sent

This command configures the files and parameters to be sent by **AT+QHTTPSEND**.

AT+QHTTPCFGEX Configure Files and Parameters to be Sent	
Test Command **AT+QHTTPCFGEX=?**	Response **+QHTTPCFGEX: "send_add","name","file_name","content_type","value"** **+QHTTPCFGEX: "send_del",(**range of supported **<index>**s**)** **OK**
Write Command **AT+QHTTPCFGEX="send_add"[,<index>[,<name>,<file_name>,<content_type>,<value>]]**	Response If the optional parameters are omitted, query whether the **<index>**s are configured or not: **+QHTTPCFGEX: "send_add",<x>,<x>,<x>,<x>,<x>,<x>,<x>,<x>,<x>,<x>,<x>,<x>** **OK** If optional parameters **<name>**, **<file_name>**, **<content_type>** and **<value>** are omitted, query the current **<index>** configuration: **+QHTTPCFGEX: "send_add",<index>,not_exit** **OK** Or **+QHTTPCFGEX: "send_add",<name>,<file_name>,<content_type>,<value>** **OK** If the optional parameters are specified, configure the

	corresponding **<index>**: **OK** Or **+CME ERROR: <err>**
Maximum Response Time	200 ms
Characteristics	The command takes effect immediately. The configurations will not be saved.

Parameter

<index>	Integer type. File upload sequence. 12 files can be uploaded in total. Range: 0–11. If a **<index>** has already been configured, the configurations will be covered when it is reconfigured by this command.
<x>	Integer type. It indicates whether the **<index>** is configured or not. 0 Configurations have not been added 1 Configurations have been added
<name>	String type. The name of parameters to be configured.
<file_name>	String type. The file to be sent. For example, UFS:xxx.
<content_type>	String type. The data type of the content to be sent. for example, application/x-www-form-urlencoded, text/plain, application/octet-stream or multipart/form-data. The difference between **<content_type>** in **AT+QHTTPCFGEX** and **AT+QHTTPCFG="contenttype",<content_type>** is: **<content_type>** in **AT+QHTTPCFG** affects the content of HTTP request body when assembling package, while the **<content_type>** in **AT+QHTTPCFGEX** affects the data type of the request body.
<value>	Data that needs to be sent to the server.
<err>	Error code. Please refer to ***Chapter 5***.

NOTE

Parameter **<value>** cannot be configured with **<file_name>** and **<content_type>** at the same time; **<value>** should be configured after **<file_name>** and **<content_type>** are set.

2.12. AT+QHTTPSEND Send POST Request to HTTP(S) via File

This command sends HTTP(S) POST request via file. The parameter **<request_header>** in **AT+QHTTPCFG="requestheader"[,<request_header>]** should be set to 0, which indicates that customizing HTTP(S) request header is disabled and standard request header is used. The parameter **<content_type>** in **AT+QHTTPCFG="contenttype",<content_type>** should be set to 3, which indicates that the data type of HTTP(S) content is multipart/form-data, i.e. form data is consist of several parts, including text data and binary data, such as files.

After executing **AT+QHTTPSEND**, the module will report **+QHTTPSEND: <err>[,<httprspcode>[,<content_length>]]** information to indicate the execution result. The **<httprspcode>** parameter can only be reported when **<err>** equals 0.

It is recommended to wait for a specific period of time (refer to the maximum response time below) for **+QHTTPSEND: <err>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is reported.

AT+QHTTPSEND Send POST Request to HTTP(S) via File	
Test Command **AT+QHTTPSEND=?**	Response **+QHTTPSEND: (**range of supported **<content_length>**s**)** **OK**
Write Command **AT+QHTTPSEND[=<rsptime>]**	Response a) 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: **+QHTTPSEND: <err>[,<httprspcode>[,<content_length>]]** b) If 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 configuration will not be saved.

Parameter

<rsptime>	Integer type. Timeout value for the HTTP(S) POST response. URC **+QHTTPSEND: <err>[,<httprspcode>[,<content_length>]]**will be outputted after **OK** is returned. Range: 1–65535. Default value: 60. Unit: second.
<err>	Error code. Please refer to ***Chapter 5***.
<httprspcode>	Integer type. HTTP server response code. Please refer to ***Chapter 6***.
<request_header>	Integer type. Disable or enable to customize HTTP(S) request header. Please refer to ***Chapter 2.1***.
<content_length>	Integer type. The length of HTTP(S) response body. Unit: byte.
<content_type>	Integer type. Data type of HTTP(S) body. Please refer to ***Chapter 2.1***.

2.13. AT+QHTTPSTOP Cancel HTTP(S) Request

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

AT+QHTTPSTOP Cancel HTTP(S) Request	
Test Command **AT+QHTTPSTOP=?**	Response **OK**
Execution Command **AT+QHTTPSTOP**	Response If the parameter format is correct and no other errors occur: **OK** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	10 s
Characteristics	/

Parameter

<err>	Error code. Please refer to ***Chapter 5***.

3 Examples

3.1. Access to HTTP Server

3.1.1. Send HTTP GET Request and Read the Response

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

```
//Example of how to send HTTP GET response.

AT+QHTTPCFG="contextid",1                //Configure the PDP context ID as 1.
OK
AT+QHTTPCFG="responseheader",1           //Allow to output HTTP response header.
OK
AT+QIACT?                                //Query the state of PDP context.
OK
AT+QICSGP=1,1,"UNINET","","",1           //Configure PDP context 1. APN is UNINET for China
                                         Unicom.
OK
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=23,80                        //Set the URL which will be accessed and the timeout value
                                         is 80 seconds.
CONNECT
HTTP://www.sina.com.cn/                  //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 HTTP GET request and the maximum response time
                                         is 80 seconds.
OK

+QHTTPGET: 0,200,601710                  //If HTTP response header contains CONTENT-LENGTH
                                         information, then the <content_length> information will be
```

```
                                   reported.

//Example of how to read HTTP response.

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

+QHTTPREAD: 0                      //Read HTTP response header and body successfully.
//Solution 2: Read HTTP response information 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 seconds.
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 from UART/USB

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

```
AT+QHTTPCFG="contextid",1                                     //Configure the PDP context
                                                              ID as 1.
OK
AT+QIACT?                                                     //Query the state of PDP
                                                              context.
OK
AT+QICSGP=1,1,"UNINET","","",1                                //Configure PDP context 1.
                                                              APN is UNINET for China
                                                              Unicom.
OK
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=59,80                                             //Set the URL which will be
                                                              accessed and the timeout
                                                              value is 80 seconds.
CONNECT
http://api.efxnow.com/DEMOWebServices2.8/Service.asmx/Echo?   //Input URL whose length is
                                                              59 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 body is
                                                              obtained via UART. The
                                                              maximum input body time
                                                              and the maximum response
                                                              time are both 80 seconds.
CONNECT
Message=HelloQuectel                                          //Input HTTP POST body
                                                              whose length is 20 bytes.
                                                              (The POST body is only an
                                                              example. Please input the
                                                              correct POST body in
                                                              practical test.)
OK

+QHTTPPOST: 0,200,177                                         //If the HTTP response
                                                              header contains
```

```
                                                                  CONTENT-LENGTH
                                                                  information, then the
                                                                  <content_length>
                                                                  information 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 seconds.
CONNECT
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="httpHTTPs://api.efxnow.com/webservices2.3">Message='HelloQuectel' ASCII:72
101 108 108 111 81 117 101 99 116 101 108 </string>               //Output HTTP response
                                                                  body.
OK

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

3.1.2.2. HTTP POST Body Obtained from File System

The following examples show how to send HTTP POST request and retrieve POST body via file system, as well as how to store 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
AT+QICSGP=1,1,"UNINET","","",1                                    //Configure PDP context 1.
                                                                  APN is UNINET for China
                                                                  Unicom.
OK
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=59,80                                                 //Set the URL which will be
                                                                  accessed and the timeout
                                                                  value is 80 seconds.
```

```
CONNECT
http://api.efxnow.com/DEMOWebServices2.8/Service.asmx/Echo?       //Input URL whose length is
                                                                  59 bytes. (This URL is only
                                                                  an example. Please input
                                                                  the correct URL in practical
                                                                  test.)
OK
//POST request information from RAM file, and read HTTP response information and store it to RAM file.
AT+QHTTPPOSTFILE="RAM:2.txt",80                                   //Send HTTP(S) POST
                                                                  request. POST body is
                                                                  obtained from RAM:2.txt,
                                                                  and the maximum response
                                                                  time is 80 s.
OK

+QHTTPPOSTFILE: 0,200,177                                         //After HTTP POST request
                                                                  is sent successfully,
                                                                  AT+QHTTPREADFILE
                                                                  command can be executed.
AT+QHTTPREADFILE="RAM:3.txt",80                                   //Read HTTP response
                                                                  body and store it to
                                                                  RAM:3.txt. The maximum
                                                                  time to wait for HTTP
                                                                  session to be closed is 80
                                                                  seconds.
OK

+QHTTPREADFILE: 0                                                 //HTTP response body is
                                                                  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 HTTP PUT request and retrieve HTTP PUT body via UART port, as well as how to read HTTP PUT response.

```
AT+QHTTPCFG="contextid",1                                         //Configure the PDP context
                                                                  ID as 1.
OK
AT+QIACT?                                                         //Query the state of PDP
                                                                  context.
```

```
OK
AT+QICSGP=1,1,"UNINET","","",1                                //Configure PDP context 1.
                                                              APN is UNINET for China
                                                              Unicom. AT+CFUN=1,1
                                                              should be set to make the
                                                              configurations take effects.
OK
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=59,80                                             //Set the URL which will be
                                                              accessed and the timeout
                                                              value is 80 seconds.
CONNECT
http://api.efxnow.com/DEMOWebServices2.8/Service.asmx/Echo?   //Input URL whose length is
                                                              59 bytes. (This URL is only
                                                              an example. Please input
                                                              the correct URL in practical
                                                              test.)
OK
AT+QHTTPPUT=20,80,80                                          //Send HTTP PUST request
                                                              and HTTP PUT body is
                                                              obtained via UART. The
                                                              maximum input body time
                                                              and the maximum response
                                                              time are both 80 seconds.
CONNECT
Message=HelloQuectel                                          //Input HTTP PUT body
                                                              whose length is 20 bytes.
                                                              (The PUT body is only an
                                                              example. Please input the
                                                              correct PUT body in
                                                              practical test.)
OK

+QHTTPPOST: 0,200,177                                         //If the HTTP response
                                                              header contains
                                                              CONTENT-LENGTH
                                                              information, then the
                                                              <content_length>
```

```
                                                        information 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 seconds.
CONNECT
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="httpHTTPs://api.efxnow.com/webservices2.3">Message='HelloQuectel' ASCII:72
101 108 108 111 81 117 101 99 116 101 108 </string>     //Output HTTP response
                                                        body.
OK

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

3.1.3.2. HTTP PUT Body Obtained from File System

The following examples show how to send HTTP PUT request and retrieve PUT body via file system, as well as how to store 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
AT+QICSGP=1,1,"UNINET","","",1                          //Configure PDP context 1.
                                                        APN is UNINET for China
                                                        Unicom. AT+CFUN=1,1
                                                        should be set to make the
                                                        configurations take effects.
OK
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=59,80                                       //Set the URL which will be
                                                        accessed and the timeout
                                                        value is 80 seconds.
CONNECT
```

```
http://api.efxnow.com/DEMOWebServices2.8/Service.asmx/Echo?      //Input URL whose length is
                                                                  59 bytes. (This URL is only
                                                                  an example. Please input
                                                                  the correct URL in practical
                                                                  test.)
OK
//PUT request information from UFS file, and read HTTP response information and store it to UFS file.
AT+QHTTPPUTFILE="UFS:2.txt",80                                    //Send HTTP(S) PUT
                                                                  request. PUT body is
                                                                  obtained from UFS:2.txt,
                                                                  and the maximum response
                                                                  time is 80 seconds.
OK

+QHTTPPOSTFILE: 0,200,177                                         //After HTTP PUT request
                                                                  is sent successfully,
                                                                  AT+QHTTPREADFILE
                                                                  command can be executed.
AT+QHTTPREADFILE="RAM:3.txt",80                                   //Read HTTP response
                                                                  body 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 body is
                                                                  stored successfully.
```

3.1.4. Send POST Request to HTTP via File

```
AT+QHTTPCFGEX="send_add"                            //Query whether file upload sequence is
                                                    configured.
+QHTTPCFGEX: "send_add",0,0,0,0,0,0,0,0,0,0,0,0

OK
AT+QHTTPCFGEX="send_add",1,"app_key","SD:1.TXT"     //Add sending content 1: SD:1.txt.
OK
AT+QHTTPCFGEX="send_add",2,"phone_number","","","xx-xxxx-xxxx-xxxx"
                                                    //Add sending content 2:
                                                    xx-xxxx-xxxx-xxxx.
OK
AT+QHTTPCFGEX="send_add",3,"longitude","","","DD"   //Add sending content 3: DD.
OK
```

```
AT+QHTTPCFGEX="send_add",0,"sn","","","AA"       //Add sending content 0: AA.
OK
AT+QHTTPCFG="CONTENTTYPE",3                      //Configure the data type pf HTTP body
                                                 as multipart/form-data.
OK
AT+QHTTPURL=41,80                                //Set the URL which will be accessed and
                                                 the timeout value is 80 seconds.
CONNECT                                          //Input the URL of HTTP server that
                                                 supports sending POST request via file
                                                 after CONNECT is returned.

OK
AT+QHTTPCFGEX="send_add",0                       //Query the added content 0.
+QHTTPCFGEX: "send_add","sn","","","AA"

OK
AT+QHTTPCFGEX="send_add"                         //Query whether file upload sequence is
                                                 configured.
+QHTTPCFGEX: "send_add",1,1,1,1,0,0,0,0,0,0,0,0

OK
AT+QHTTPCFGEX="send_add",0                       //Query the added content 0.
+QHTTPCFGEX: "send_add","sn","","","AA"

OK
AT+QHTTPCFGEX="send_del",0                       //Delete content 0 that has been added
                                                 previously.
OK
AT+QHTTPCFGEX="send_add"                         //Query whether file upload sequence is
                                                 configured.
+QHTTPCFGEX: "send_add",0,1,1,1,0,0,0,0,0,0,0,0

OK
AT+QHTTPSEND=60                                  //Send POST request.
OK

+QHTTPSEND: 0,200
```

3.2. Access to HTTPS Server

3.2.1. Send HTTPS GET Request and Read the Response

The following examples show how to send HTTPS GET request and enable output of HTTPS response header, as well as how to read 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         //Allow to output HTTPS response header.
OK
AT+QIACT?                              //Query the state of PDP context.
OK
AT+QICSGP=1,1,"UNINET","","",1         //Configure PDP context 1. APN is UNINET for China
                                       Unicom.
OK
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",1,0x0005//Set SSL cipher suite as 0x0005 which means RC4-SHA.
OK
AT+QSSLCFG="seclevel",1,0              //Set SSL verify level as 0 which means CA certificate is not
                                       needed.
OK
AT+QHTTPURL=22,80                      //Set the URL which will be accessed and the timeout value
                                       is 80 seconds.
CONNECT
https://www.alipay.com                 //Input URL whose length is 19 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 seconds.
OK

+QHTTPGET: 0,200,21472                 //If the HTTPS response header contains
```

```
                                 CONTENT-LENGTH information, then the
                                 <content_length> information will be reported.
//Example of how to read HTTPS response.

//Solution 1: Read HTTPS response information and output it via UART.
AT+QHTTPREAD=80                  //Read HTTPS response information and output it via UART.
                                 The maximum time to wait for HTTPS session to be closed
                                 is 80 seconds.
CONNECT                          //HTTPS response header and body.
HTTP/1.1 200 OK<CR><LF>
Server: Tengine/2.1.0<CR><LF>
Date: Tue, 12 Sep 2017 05:54:34 GMT <CR><LF>
Content-Type: text/html; charset=utf-8<CR><LF>
Content-Length: 21451<CR><LF>
Connection: keep-alive <CR><LF>
…… <CR><LF>                      //Lines are omitted here.
<CR><LF>
<body>
OK

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

//Solution 2: Read HTTPS response information 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 seconds.
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 from UART/USB

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

```
AT+QHTTPCFG="contextid",1                          //Configure the PDP context ID as 1.
OK
AT+QIACT?                                          //Query the state of PDP context.
OK
```

```
AT+QICSGP=1,1,"UNINET","","",1                      //Configure PDP context 1. APN is
                                                    UNINET for China Unicom.
OK
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.
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 with
                                                    AT+QFUPL command.
OK
AT+QSSLCFG="cacert",1,"RAM:cacert.pem"              //Configure the path of trusted CA
                                                    certificate for the specified SSL
                                                    context.
OK
AT+QSSLCFG="clientcert",1,"RAM:clientcert.pem"      //Configure the path of client
                                                    certificate for the specified SSL
                                                    context.
OK
AT+QSSLCFG="clientkey",1,"RAM:clientkey.pem"        //Configure the path of client private
                                                    key for the specified SSL context.
OK
AT+QHTTPURL=45,80                                   //Set the URL which will be accessed
                                                    and the timeout value is 80 seconds.
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 is obtained from UART.
```

```
                                                    The maximum input body time and
                                                    the maximum response time are both
                                                    80 seconds.
CONNECT
Message=1111&Appleqty=2222&Orangeqty=3333&find=1    //Input HTTPS POST 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 CONTENT-LENGTH
                                                    information, then the
                                                    <content_length> information will
                                                    be reported.
AT+QHTTPREAD=80                                     //Read HTTPS response body and
                                                    output it via UART. The maximum
                                                    time to wait for HTTPS session to be
                                                    closed is 80 seconds.
CONNECT                                             //Read HTTPS response body
                                                    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 body is outputted
                                                    successfully.
```

3.2.2.2. HTTPS POST Body Obtained from File System

The following examples show how to send HTTPS POST request and retrieve HTTPS POST body from file system, as well as how to store HTTPS 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
AT+QICSGP=1,1,"UNINET","","",1              //Configure PDP context 1. APN is UNINET for
                                            China Unicom.
OK
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.
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 with AT+QFUPL command.
OK
AT+QSSLCFG="cacert",1,"RAM:cacert.pem"      //Configure the path of trusted CA certificate for the
                                            specified SSL context.
OK
AT+QSSLCFG="clientcert",1,"RAM:clientcert.pem"//Configure the path of client certificate for the
                                            specified SSL context.
OK
AT+QSSLCFG="clientkey",1,"RAM:clientkey.pem" //Configure the path of client private key for the
                                            specified SSL context.
OK
AT+QHTTPURL=45,80                           //Set the URL which will be accessed and the
                                            timeout value is 80 seconds.
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 information from RAM file, and read HTTPS response information and store it to RAM file.
AT+QHTTPPOSTFILE="RAM:5.txt",80             //Send HTTPS POST request. HTTPS POST body
                                            is obtained from RAM:5.txt, and the maximum
```

```
                                           response time is 80 seconds.
OK

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

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

```
AT+QHTTPCFG="contextid",1                            //Configure the PDP context ID as 1.
OK
AT+QIACT?                                            //Query the state of PDP context.
OK
AT+QICSGP=1,1,"UNINET","","",1                       //Configure PDP context 1. APN is
                                                     UNINET for China Unicom.
                                                     AT+CFUN=1,1 should be set to
                                                     make the configurations take effects.
OK
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 with
                                                  AT+QFUPL command.
OK
AT+QSSLCFG="cacert",1,"UFS:cacert.pem"            //Configure the path of trusted CA
                                                  certificate for the specified SSL
                                                  context.
OK
AT+QSSLCFG="clientcert",1,"UFS:clientcert.pem"    //Configure the path of client
                                                  certificate for the specified SSL
                                                  context.
OK
AT+QSSLCFG="clientkey",1,"UFS:clientkey.pem"      //Configure the path of client private
                                                  key for the specified SSL context.
OK
AT+QHTTPURL=45,80                                 //Set the URL which will be accessed
                                                  and the timeout value is 80 seconds
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+QHTTPPUT=48,80,80                              //Send HTTPS PUT request.
                                                  HTTPS PUT is obtained from UART.
                                                  The maximum input body time and
                                                  the maximum response time are both
                                                  80 seconds.
CONNECT
Message=1111&Appleqty=2222&Orangeqty=3333&find=1  //Input HTTPS PUT body whose
                                                  length is 48 bytes. (This PUT body is
                                                  only an example. Please input
                                                  the correct one in practical test.)
OK

+QHTTPPOST: 0,200,285                             //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 UART. The maximum
                                                  time to wait for HTTPS session to be
```

```
                                                   closed is 80 seconds.
CONNECT                                            //Read HTTPS response body
                                                   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 body is outputted
                                                   successfully.
```

3.2.3.2. HTTPS PUT Body Obtained from File System

The following examples show how to send HTTPS PUT request and retrieve HTTPS PUT body from file system, as well as how to store HTTPS 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
AT+QICSGP=1,1,"UNINET","","",1           //Configure PDP context 1. APN is UNINET for
                                         China Unicom. AT+CFUN=1,1 should be set to
                                         make the configurations take effects.
OK
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 with AT+QFUPL command.
OK
AT+QSSLCFG="cacert",1,"UFS:cacert.pem"       //Configure the path of trusted CA certificate for the
                                              specified SSL context.
OK
AT+QSSLCFG="clientcert",1,"UFS:clientcert.pem"//Configure the path of client certificate for the
                                              specified SSL context.
OK
AT+QSSLCFG="clientkey",1,"UFS:clientkey.pem" //Configure the path of client private key for the
                                              specified SSL context.
OK
AT+QHTTPURL=45,80                            //Set the URL which will be accessed and the
                                              timeout value is 80 seconds.
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 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, and the maximum
                                              response time is 80 seconds.
OK

+QHTTPPUTFILE: 0,200,177                     //After HTTPS PUT request is sent successfully,
                                              AT+QHTTPREADFILE command 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 seconds.
OK

+QHTTPREADFILE: 0                            //HTTPS response body is stored successfully.
```

3.2.4. Send POST Request to HTTPS via File

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

```
AT+QIACT?                                          //Query the state of PDP context.
OK
AT+QICSGP=1,1,"UNINET","","",1                     //Configure PDP context 1. APN is
                                                   UNINET for China Unicom.
                                                   AT+CFUN=1,1 should be set to make the
                                                   configurations take effects.
OK
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 with
                                                   AT+QFUPL command.
OK
AT+QSSLCFG="cacert",1,"UFS:cacert.pem"             //Configure the path of trusted CA
                                                   certificate for the specified SSL context.
OK
AT+QSSLCFG="clientcert",1,"UFS:clientcert.pem"     //Configure the path of client certificate for
                                                   the specified SSL context.
OK
AT+QSSLCFG="clientkey",1,"UFS:clientkey.pem"       //Configure the path of client private key
                                                   for the specified SSL context.
OK
AT+QHTTPCFGEX="send_add"                           //Query whether file upload sequence is
                                                   configured.
+QHTTPCFGEX: "send_add",0,0,0,0,0,0,0,0,0,0,0,0

OK
AT+QHTTPCFGEX="send_add",1,"app_key","SD:1.TXT"    //Add sending content 1: SD:1.txt.

OK
AT+QHTTPCFGEX="send_add",2,"phone_number","","","xx-xxxx-xxxx-xxxx"
```

```
OK                                                  //Add sending content 2:
                                                    xx-xxxx-xxxx-xxxx.
AT+QHTTPCFGEX="send_add",3,"longitude","","","DD"   //Add sending content 1: DD.
OK
AT+QHTTPCFGEX="send_add",0,"sn","","","AA"          //Add sending content 0: AA
OK
AT+QHTTPCFG="CONTENTTYPE",3                         //Configure the data type pf HTTP body
                                                    as multipart/form-data.
OK
AT+QHTTPURL=41,80                                   //Set the URL which will be accessed and
                                                    the timeout value is 80 seconds.
CONNECT                                             //Input the URL of HTTP server that
                                                    supports sending POST request via file
                                                    after CONNECT is returned.

OK
AT+QHTTPCFGEX="send_add",0                          //Query the added content 0.
+QHTTPCFGEX: "send_add","sn","","","AA"

OK
AT+QHTTPCFGEX="send_add"                            //Query whether file upload sequence is
                                                    configured.
+QHTTPCFGEX: "send_add",1,1,1,1,0,0,0,0,0,0,0,0

OK
AT+QHTTPCFGEX="send_add",0                          //Query the added content 0.
+QHTTPCFGEX: "send_add","sn","","","AA"

OK
AT+QHTTPCFGEX="send_del",0                          //Delete content 0 that has been added
                                                    previously.
OK
AT+QHTTPCFGEX="send_add"                            //Query whether file upload sequence is
                                                    configured.
+QHTTPCFGEX: "send_add",0,1,1,1,0,0,0,0,0,0,0,0

OK
AT+QHTTPSEND=60                                     //Send POST request.
OK

+QHTTPSEND: 0,200
```

4 Error Handling

4.1. Executing HTTP(S) AT Commands Fails

When executing HTTP(S) AT commands, if **ERROR** response is received from the module, please check whether the (U)SIM card is inserted and whether it is **+CPIN: READY** returned when executing **AT+CPIN?**.

4.2. PDP Activation Fails

If it is failed to active a PDP context with **AT+QIACT**, please check the following configurations:

1. Query whether the PS domain is attached or not with **AT+CGATT?**. If not, please execute **AT+CGATT=1** to attach the PS domain.
2. Query the PS domain status with **AT+CGREG?** and make sure the PS domain has been registered.
3. Query the PDP context parameters with **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 with **AT+CGACT**.
5. According to 3GPP specifications, the module only supports 3 PDP contexts activated simultaneously, so the number of activated PDP contexts must be ensured less than 3.

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

4.3. DNS Parse Fails

When executing **AT+QHTTPGET**, **AT+QHTTPPOST**, **AT+QHTTPPOSTFIL**, **AT+QHTTPPUT**, **AT+QHTTPPUTFILE** and **AT+QHTTPSEND**, 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 with **AT+QIACT?** to make sure the specified PDP context has been activated successfully.
3. Query the address of DNS server with **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 with **AT+QIDNSCFG**.
2. Deactivate the PDP context with **AT+QIDEACT**, and re-activate the PDP context via **AT+QIACT**.

4.4. Entering Data Mode Fails

When executing **AT+QHTTPURL**, **AT+QHTTPGET**, **AT+QHTTPPOST**, **AT+QHTTPPOSTFILE, AT+QHTTPREAD**、**AT+QHTTPPUT** and **AT+QHTTPPUTFILE**, 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 any, please re-execute these commands after other ports have exited from data mode.

4.5. Sending GET/POST Requests Fails

When executing **AT+QHTTPGET**, **AT+QHTTPGETEX**, **AT+QHTTPPOST**, **AT+QHTTPPOSTFILE**, **AT+QHTTPPUT** and **AT+QHTTPPUTFILE**, if a failed result is received, 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 commands.
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+QHTTPGETEX**, **AT+QHTTPPOST**, **AT+QHTTPPOSTFILE**, **AT+QHTTPPUT** and **AT+QHTTPPUTFILE** still fails, please deactivate the PDP context with **AT+QIDEACT** and re-activate the PDP context with **AT+QIACT** to resolve this issue. If activating the PDP context fails, please refer to ***Chapter 4.2*** to resolve it.

4.6. Reading Response Fails

Before reading response with **AT+QHTTPREAD** and **AT+QHTTPREADFILE**, execute **AT+QHTTPGET**, **AT+QHTTPPOST**, **AT+QHTTPPOSTFILE, AT+QHTTPPUT, AT+QHTTPPUTFILE** and **AT+QHTTPSEND** and the following URC information will be 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>]]
+QHTTPSEND: <err>],<httprspcode>[,<content_length>]]

During executing **AT+QHTTPREAD** and **AT+QHTTPREADFILE**, if customers encounter some errors, such as **+CME ERROR: 717** (717: HTTP(S) socket read error), please resend HTTP(S) GET/POST/PUT requests to HTTP(S) server with **AT+QHTTPGET**, **AT+QHTTPPOST**, **AT+QHTTPPOSTFILE**, **AT+QHTTPPUT**, **AT+QHTTPPUTFILE** and **AT+QHTTPSEND**. If sending GET/POST requests to HTTP(S) server fails, please refer to ***Chapter 4.5***to resolve it.

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>	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 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>	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_LTE_Standard_TCP(IP)_Application_Note
[2] Quectel_EC2x&EG9x&EG2x-G&EM05_Series_AT_Commands_Manual
[3] Quectel_EC2x&EG9x&EG2x-G&EM05_Series_SSL_Application_Note
[4] Quectel_LTE_Standard_FILE_Application_Note

Table 6: Terms and Abbreviations

Abbreviation	Description
APN	Access Point Name
CA	Certification Authority
DNS	Domain Name Server
DTR	Data Terminal Ready
HTTP(S)	Hyper Text Transport Protocol (Secure)
ID	Identifier
IP	Internet Protocol
LTE	Long-Term Evolution
MCU	Microprogrammed Control Unit
PPP	Point-to-Point Protocol
PS	Packet Switch

RAM	Random Access Memory
SSL	Security Socket Layer
TA	Terminal Adapter
UART	Universal Asynchronous Receiver/Transmitter
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
(U)SIM	(Universal) Subscriber Identity Module