LTE Standard(A) Series HTTP(S) Application Note

LTE Standard Module Series

Version: 1.3.0

Date: 2022-06-20

Status: Preliminary

Legal Notices

Use and Disclosure Restrictions

License Agreements

Copyright

About the Document

Revision History

Version	Date	Author	Description
-	2020-10-30	Luffy LIU	Creation of the document
1.0	2020-11-30	Luffy LIU	First official release
1.1	2021-04-13	Luffy LIU	1. Added the applicable modules EC200N-CN and EC600N-CN. 2. Updated the examples (Chapter 3).
1.2	2022-03-02	Larson LI	1. Added the applicable modules EC200A series, EC800N-CN and EG915N-EU. 2. Deleted EG912Y-CN module. 3. Added AT+QHTTPCFG="reqheader/add" and AT+QHTTPCFG=""reqheader/remove" (Chapter 2.3.1). 4. Updated the description of <closedind> of AT+QHTTPCFG="closed/ind" (Chapter 2.3.1).
1.3.0	2022-06-20	Larson LI	Preliminary: 1. Added applicable modules EC200M-CN, EC600M-CN, EC800M-CN and EG912N-EN. 2. Deleted applicable module EC200T series. 3. Updated EG915N-EU to EG915N series. 4. Added data types "application/json" and "image/jpeg" for the parameter <content_type> (Chapter 2.3.1).

Contents

Table Index

1 Introduction

Quectel LTE Standard(A) series modules support HTTP(S) applications by accessing HTTP(S) servers.

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

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

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

1.1. Applicable Modules

Table 1: Applicable Modules

Module Series	Module
LTE Standard(A)	EC200A Series
	EC200M-CN
	EC200N-CN
	EC200S Series
	EC600M-CN
	EC600N-CN
	EC600S-CN
	EC800M-CN
	EC800N-CN
	EG912N-EN

	EG912Y-EU
	EG915N Series

1.2. 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 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**. See ***document [1]*** for more information.

Step 2: Activate the PDP context with **AT+QIACT**. You can query the assigned IP address with **AT+QIACT?**. See ***document [1]*** for more information.

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 information, see ***document [2]***.

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

Step 6: Send an HTTP(S) request. You can use **AT+QHTTPGET** for sending an HTTP(S) GET request, while **AT+QHTTPPOST** or **AT+QHTTPPOSTFILE** can be used for sending an HTTP(S) POST request.

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

Step 8: Deactivate the PDP context with **AT+QIDEACT**. For more information, see ***document [1]***.

1.3. Description of HTTP(S) Header

1.3.1. 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 URI 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>.**

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

```
POST /processorder.php HTTP/1.1<CR><LF>
Host: 220.180.239.212:8011<CR><LF>
Accept: */*<CR><LF>
User-Agent: QUECTEL_MODULE<CR><LF>
Connection: Keep-Alive<CR><LF>
Content-Type: application/x-www-form-urlencoded<CR><LF>
Content-Length: 48<CR><LF>
<CR><LF>
Message=1111&Appleqty=2222&Orangeqty=3333&find=1
```

1.3.2. Output HTTP(S) Response Header

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**.

1.4. Description of Data Mode

The COM port of the above LTE Standard(A) 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** 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 Command Introduction

2.1.1. Definitions

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

2.1.2. AT Command Syntax

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

Table 2: Type of AT Commands

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

2.2. Declaration of AT Command Examples

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

2.3. Description of AT Commands

2.3.1. 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 only executes one parameter, it will query the current settings.

AT+QHTTPCFG Configure Parameters for HTTP(S) Server	
Test Command **AT+QHTTPCFG=?**	Response **+QHTTPCFG: "contextid",(**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: "reqheader/add",<header_name>,<header_str>** **+QHTTPCFG: "reqheader/remove",<header_name>** **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: "reqheader/add",<add_num>[,<header_name>:<header_str>,...]** **+QHTTPCFG: "reqheader/remove",<add_num>[,<header_name>,...]** **OK**
Write Command **AT+QHTTPCFG="contextid"[,<contextID>]**	Response If the optional parameter is omitted, query the current settings: **+QHTTPCFG: "contextid",<contextID>** **OK** If the optional parameter is specified, configure the PDP context ID: **OK** Or **+CME ERROR: <err>**
Write Command **AT+QHTTPCFG="requestheader"[,<request_header>]**	Response If the optional parameter is omitted, query the current settings: **+QHTTPCFG: "requestheader",<request_header>** **OK** If the optional parameter is specified, configure whether to enable customization of 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 settings: **+QHTTPCFG: "responseheader",<response_header>** **OK** If the optional parameter is specified, configure whether to enable the outputting of the HTTP(S) response header: **OK** Or **+CME ERROR: <err>**
Write Command **AT+QHTTPCFG="sslctxid"[,<sslctxID>]**	Response If the optional parameter is omitted, query the current settings: **+QHTTPCFG: "sslctxid",<sslctxID>**

	OK If the optional parameter is specified, configure 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 settings: **+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 settings: **+QHTTPCFG: "rspout/auto",<auto_outrsp>** **OK** If the optional parameter is specified, configure whether to enable auto output of HTTP(S) response: **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="reqheader/add"[,<header_name>[,<header_str>]]**	Response If all optional parameters are omitted, query the customized header(s) added: **+QHTTPCFG: "reqheader/add",<add_num>[,<header_nam**

	e>:<header_str>,...] **OK** If the optional parameter **<header_str>** is omitted, query the attributes of **<header_name>**: **+QHTTPCFG: "reqheader/add",<header_name>:<header_str>** **OK** Or **+CME ERROR: <err>** If the optional parameters are specified, set the customized header: **OK** Or **+CME ERROR: <err>**
Write Command **AT+QHTTPCFG="reqheader/remove",<header_name>**	Response **OK** Or **+CME ERROR: <err>**
Maximum Response Time	300 ms
Characteristics	This command takes effect immediately. The configurations will not be saved.

Parameter

<contextID>	Integer type. PDP context ID. Range: 1–15. 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 output of HTTP(S) response header. 0 Disable 1 Enable
<sslctxID>	Integer type. SSL context ID used for HTTP(S). Range: 0–5. Default value: 1. SSL parameters can be configured with **AT+QSSLCFG**. For more information, see ***document [2]***.
<content_type>	Integer type. Data type of HTTP(S) body. 0 "application/x-www-form-urlencoded" 1 "text/plain" 2 "application/octet-stream" 3 "multipart/form-data"

	4 "application/json"
	5 "image/jpeg"
<auto_outrsp>	Integer type. Disable or enable auto output of HTTP(S) response data. If auto outputting 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
<header_name>	String type. Customized header.
<header_str>	String type. Content of customized header.
<add_num>	Integer type. Number of added customized headers. Default value: 0.
<err>	Error code. See ***Chapter 5*** for more information.

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

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

AT+QHTTPURL Set URL of HTTP(S) Server	
Test Command **AT+QHTTPURL=?**	Response **+QHTTPURL: (**range of supported **<URL_length>**s**),(**range of supported **<timeout>**s**)** **OK**
Read Command **AT+QHTTPURL?**	Response **[+QHTTPURL: <URL>]** **OK**
Write Command **AT+QHTTPURL=<URL_length>[,<timeout>]**	Response a) If the parameter format is correct, but HTTP(S) GET/POST requests are not being sent: **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	This command takes effect immediately. The configurations will not be saved.

Parameter

<URL_length>	Integer type. URL length. Range: 1–2048. Unit: byte.
<timeout>	Integer type. Maximum time for inputting URL. Range: 1–65535. Default value: 60. Unit: second.
<URL>	String type. HTTP(S) server URL.
<err>	Error code. See ***Chapter 5*** for more information.

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

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

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

AT+QHTTPGET Send GET Request to HTTP(S) Server	
Test Command **AT+QHTTPGET=?**	Response **+QHTTPGET:** (range of supported **<rsptime>**s),(range of supported **<data_length>**s),(range of supported **<input_time>**s) **OK**
Write Command If **<request_header>** equals 0 (disable customizing HTTP(S) request header)	Response a) If the parameter format is correct and no other errors occur: **OK**

AT+QHTTPGET[=<rsptime>]	When the module has received a 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>**
Write Command If **<request_header>** equals 1 (enable customizing HTTP(S) GET request header) **AT+QHTTPGET=<rsptime>,<data_length>[,<input_time>]**	Response a) If the connection to the HTTP(S) server has been established successfully: **CONNECT** TA switches to 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 a 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: **+QHTTPGET: <err>** b) If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	Determined by **<rsptime>**
Characteristics	/

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. Length of HTTP(S) request, including HTTP(S) request header and HTTP(S) request body. Range: 1–2048. Unit: byte.
<input_time>	Integer type. Maximum time for inputting HTTP(S) request including HTTP(S) request header and HTTP(S) request body. Range: 1–65535. Default value: 60. Unit: second.

<httprspcode>	Integer type. HTTP(S) server response code. See ***Chapter 6*** for more information.
<request_header>	Integer type. Disable or enable customizing HTTP(S) request header. 0 Disable 1 Enable
<content_length>	Integer type. Length of HTTP(S) response body. Unit: byte.
<err>	Error code. See ***Chapter 5*** for more information.

2.3.4. 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**),(**range of supported **<start_postion>**s**,(**range of supported **<read_len>**s**)** **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	/

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. Start position of the data that the HTTP(S) client wants to get.
<read_len>	Integer type. Length of the data that the HTTP(S) client wants to get.
<httprspcode>	Integer type. HTTP(S) server response code. See ***Chapter 6*** for more information.
<content_length>	Integer type. Length of HTTP(S) response body. Unit: byte.
<err>	Error code. See ***Chapter 5*** for more information.

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

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

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

AT+QHTTPPOST Send POST Request to HTTP(S) Server via UART/USB	
Test Command **AT+QHTTPPOST=?**	Response **+QHTTPPOST: (**range of supported **<data_length>**s**),(**range of supported **<input_time>**s**),(**range of supported **<rsptime>**s**)** **OK**
Write Command If **<request_header>** equals 0 (disable customizing HTTP(S) request header) **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 a response from HTTP(S) server, it will report the following URC: **+QHTTPPOST: <err>[,<httprspcode>[,<content_length>]]**

	If **<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: **+QHTTPPOST: <err>** b) If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Write Command If **<request_header>** equals 1 (enable customizing HTTP(S) request header) **AT+QHTTPPOST=<data_length>[,<input_time>,<rsptime>]**	Response a) If the parameter format is correct and the connection to HTTP(S) server has been established successfully: **CONNECT** TA switches to the transparent transmission mode, and then the HTTP(S) POST header and body can be inputted. When the total size of the inputted data reaches **<data_length>**, TA will return to command mode and report the following code: **OK** When the module has received a response from HTTP(S) server, it will report the following URC: **+QHTTPPOST: <err>[,<httprspcode>[,<content_length>]]** If the **<input_time>** has reached, but the received data length is less than **<data_length>**, TA will return to command mode and report the following code: **+QHTTPPOST: <err>** b) If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	Determined by network and **<rsptime>**
Characteristics	/

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, including HTTP(S) POST request header and body. Range: 1–1024000. Unit: byte.
<input_time>	Integer type. Maximum time for inputting HTTP(S) POST body or HTTP(S) POST request. Range: 1–65535. Default value: 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.
<httprspcode>	Integer type. HTTP(S) server response code. See ***Chapter 6*** for more information.
<request_header>	Integer type. Disable or enable customizing HTTP(S) request header. 0 Disable 1 Enable
<content_length>	Integer type. Length of HTTP(S) response body. Unit: byte.
<err>	Error code. See ***Chapter 5*** for more information.

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

The command sends an HTTP(S) POST request via a 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 only.
- 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>]]** 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 **+QHTTPPOSTFILE: <err>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is returned.

AT+QHTTPPOSTFILE Send POST Request to HTTP(S) Server via File	
Test Command **AT+QHTTPPOSTFILE=?**	Response **+QHTTPPOSTFILE: <file_name>,(**range of supported **<rsptime>**s**)[,(**range of supported **<post_mode>**s**)]** **OK**
Write Command If **<request_header>** equals 1, the specified file must contain both HTTP(S) request header and body. **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 a 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	/

Parameter

<file_name>	String type. File name. Max. file name length: 80 bytes.
<rsptime>	Integer type. Timeout value for the HTTP(S) POST response **+QHTTPPOSTFILE: <err>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is returned. Range: 1–65535. Default value: 60. Unit: second.
<httprspcode>	Integer type. HTTP(S) server response code. See ***Chapter 6*** for more information.
<request_header>	Integer type. Disable or enable customizing HTTP(S) request header. 0 Disable 1 Enable
<content_length>	Integer type. Length of HTTP(S) response body. Unit: byte.
<post_mode>	String type. HTTP(S) sending files in segments. 0 Send the current file directly 1 Record the file name to be sent (do not send the file for now, wait to send it together with the file configured when **<post_mode>=2** 2 Send the files configured when **<post_mode>**=1 and 2 in order (only sending two files together is supported)
<err>	Error code. See ***Chapter 5*** for more information.

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

This command retrieves the HTTP(S) response from an HTTP(S) server via the UART/USB port, after sending HTTP(S) GET/POST requests. **AT+QHTTPREAD** must be executed after **+QHTTPGET: <err>[,<httprspcode>[,<content_length>]]**, **+QHTTPPOST: <err>[,<httprspcode>[,<content_length>]]** or **+QHTTPPOSTFILE: <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 **<Outputs HTTP(S) response information>** **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 : **+QHTTPREAD: <err>** b) If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	Determined by **<wait_time>**
Characteristics	/

Parameter

<wait_time>	Integer type. Max interval between receiving two data packets. Range: 1–65535. Default value: 60. Unit: second.
<err>	Error code. See ***Chapter 5*** for more information.

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

This command retrieves the HTTP(S) response from HTTP(S) server via file after sending HTTP (S) GET/POST requests. **AT+QHTTPREADFILE** must be executed after **+QHTTPGET: <err>[,<httprspcode>[,<content_length>]]**, **+QHTTPPOST: <err>[,<httprspcode>[,<content_length>]]** or **+QHTTPPOSTFILE: <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** When response from the HTTP(S) server is read or **<wait_time>** is reached, it will report: **+QHTTPREADFILE: <err>** b) If the parameter format is incorrect or other errors occur:

	+CME ERROR: <err>
Maximum Response Time	Determined by **<wait_time>**
Characteristics	/

Parameter

<wait_time>	Integer type. Max time between receiving two data packets. Range: 1–65535. Default value: 60. Unit: second.
<file_name>	String type. File name. Maximum length of file name: 80 bytes.
<err>	Error code. See ***Chapter 5*** for more information.

2.3.9. AT+QHTTPSTOP Cancel HTTP(S) Request

MCU can cancel HTTP(S) GET/POST 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 a) If the parameter format is correct and no other errors occur: **OK** b) If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	10 s
Characteristics	/

Parameter

<err>	Error code. See ***Chapter 5*** for more information.

3 Examples

3.1. Access HTTP Server

3.1.1. Send HTTP GET Request and Read the Response

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

```
//Example of how to send an HTTP GET request.
AT+QHTTPCFG="contextid",1           //Configure the PDP context ID as 1.
OK
AT+QHTTPCFG="responseheader",1      //Allow the output of HTTP response header.
OK

AT+QIACT?                           //Query the state of context.
OK                                  //No context activated currently.

AT+QICSGP=1,1,"UNINET","","",1      //Configure PDP context 1. China Unicom APN: UNINET.
                                    (Then set AT+CFUN=1,1 to make the configuration take effect.)
OK

AT+QIACT?                           //Query the state of context.
+QIACT: 1,1,1,"10.7.157.1"

OK

//The first PDP is activated by default. If it is queried inactivated, use AT+QIACT=1 to activate it.
AT+QIACT=1                          //Activate context 1.
OK                                  //Activated successfully.

AT+QHTTPURL=23,80                   //Set the URL of the HTTP server that will be accessed and
                                    timeout value as 80 s.
CONNECT
HTTP://www.sina.com.cn/             //Input URL whose length is 23 bytes. (This URL is only an
                                    example. Input the correct URL in a practical test.)
OK
AT+QHTTPGET=80                      //Send HTTP GET request with the maximum response time of
```

```
                                      80 s.
OK

+QHTTPGET: 0,200,601710               //If HTTP response header contains CONTENT-LENGTH
                                      information, then the <content_length> (601710) is reported.

//Example of how to read an HTTP response.
//Solution 1: Read the HTTP response and output it via the UART port.
AT+QHTTPREAD=80                       //Read HTTP response and output it via UART. The maximum
                                      time to wait for an HTTP session to be closed is 80 s.
CONNECT
HTTP/1.1 200 OK <CR><LF>              //HTTP response header and body.
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>                               //Response is omitted here.
<CR><LF>
<body>
OK

+QHTTPREAD: 0                            // HTTP response header and body have been read
                                         successfully.
//Solution 2: Read HTTP response and store it to a UFS file.
AT+QHTTPREADFILE="UFS:1.txt",80          //Read HTTP response header and body and store them to
                                         UFS:1.txt. The maximum time to wait for HTTP session to be
                                         closed is 80 s.
OK

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

3.1.2. Send HTTP POST Request and Read the Response

3.1.2.1. HTTP POST Body Obtained from UART/USB

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

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

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

OK                                  //No context activated currently.

AT+QICSGP=1,1,"UNINET","","",1      //Configure PDP context 1. China Unicom APN: UNINET.
                                    (Then set AT+CFUN=1,1 to make the configuration take effect.)
OK

AT+QIACT?                           //Query the state of context.
+QIACT: 1,1,1,"172.22.86.226"

OK
//The first PDP is activated by default. If it is queried inactivated, use AT+QIACT=1 to activate it.
AT+QIACT=1                          //Activate context 1.
OK                                  //Activated successfully.

AT+QHTTPURL=59,80                   //Set the URL of the HTTP server that will be accessed and
                                    timeout value as 80 s.
CONNECT
http://api.efxnow.com/DEMOWebServices2.8/Service.asmx/Echo?   //Input URL whose length is 59
                                                              bytes. (This URL is only an
                                                              example. Input the correct URL
                                                              in a practical test.)
OK

AT+QHTTPPOST=20,80,80               //Send HTTP POST request and HTTP POST body is obtained
                                    via UART. The maximum input time and the maximum response
                                    time are 80 s each.
CONNECT
Message=HelloQuectel                //Input HTTP POST body whose length is 20 bytes. (The POST
                                    body is only an example. Input the correct POST body in a
                                    practical test.)
OK
```

```
+QHTTPPOST: 0,200,177             //If the HTTP response header contains CONTENT-LENGTH
                                  information, then the <content_length> (177) will be reported.

AT+QHTTPREAD=80                   //Read the HTTP response body and output it via UART. The
                                  maximum time to wait for HTTP session to be closed is 80 s.
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 has been outputted successfully.
```

3.1.2.2. HTTP POST Body Obtained from File System

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

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

AT+QIACT?                         //Query the state of context.
OK                                //No context activated currently.

AT+QICSGP=1,1,"UNINET","","",1    //Configure PDP context 1. China Unicom APN: UNINET.
                              (Then set AT+CFUN=1,1 to make the configuration take effect.)
OK

AT+QIACT?                         //Query the state of context.
+QIACT: 1,1,1,"172.22.86.226"

OK
//The first PDP is activated by default. If it is queried inactivated, use AT+QIACT=1 to activate it.
AT+QIACT=1                        //Activate context 1.
OK                                //Activated successfully.

AT+QHTTPURL=59,80                 //Set the URL the HTTP server that will be accessed and
                                  timeout value as 80 s.
CONNECT
http://api.efxnow.com/DEMOWebServices2.8/Service.asmx/Echo?  //Input URL whose length is 59
                                                             bytes. (This URL is only an
                                                             example. Input the correct
                                                             URL in practical test.)
OK
```

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

AT+QHTTPPOSTFILE="UFS:2.txt",80      //Send HTTP(S) POST request. POST body is obtained
                                     from UFS:2.txt. The maximum response time is 80 s.
OK

+QHTTPPOSTFILE: 0,200,177            //After HTTP POST request is sent successfully, the HTTP
                                     response body can be read by executing AT+QHTTPREAD
                                     or AT+QHTTPREADFILE.

AT+QHTTPREADFILE="UFS: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 has been stored successfully.
```

3.2. Access HTTPS Server

3.2.1. Send HTTPS GET Request and Read the Response

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

```
//Example of how to send an HTTPS GET request.
AT+QHTTPCFG="contextid",1            //Configure the PDP context ID as 1.
OK

AT+QHTTPCFG="responseheader",1       //Allow the output of HTTPS response header.
OK

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

AT+QICSGP=1,1,"UNINET","","",1       //Configure PDP context 1. China Unicom APN: UNINET
                                     (Then set AT+CFUN=1,1 to make the configuration take
                                     effect.)
OK

AT+QIACT?                            //Query the state of context.
+QIACT: 1,1,1,"10.7.157.1"
```

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

AT+QIACT=1                                //Activate context 1.
OK                                        //Activated successfully.

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",1,0                 //Set SSL verification level as 0, which means that a CA
                                          certificate is not needed.
OK

AT+QHTTPURL=22,80                         //Set the URL of the HTTPS server that will be accessed
                                          and timeout value as 80 s.
CONNECT
https://www.alipay.com                    //Input a URL whose length is 22 bytes. (This URL is only an
                                          example. Input the correct URL in a practical test.)
OK

AT+QHTTPGET=80                            //Send HTTPS GET request with the maximum response
                                          time of 80 s.
OK

+QHTTPGET: 0,200,21472                    //If the HTTPS response header contains CONTENT-
                                          LENGTH information, the <content_length> (21472) will
                                          be reported.
//Example of how to read an HTTPS response.
//Solution 1: Read HTTPS response and output it via UART.
AT+QHTTPREAD=80                           //Read HTTPS response and output it via UART.
                                          The maximum time to wait for HTTPS session to be closed
                                          is 80 s.

CONNECT                                   //HTTPS response header and body.
HTTP/1.1 200 OK<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>                          //Response is omitted here.
<CR><LF>
<body>
OK

+QHTTPREAD: 0                        // HTTPS response header and body have been read
                                     successfully.

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

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

3.2.2. Send HTTPS POST Request and Read the Response

3.2.2.1. HTTPS POST Body Obtained from UART/USB

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

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

AT+QIACT?                            //Query the state of context.
OK                                   //No context activated currently.

AT+QICSGP=1,1,"UNINET","","",1       //Configure PDP context 1. China Unicom APN: UNINET
                                     (Then set AT+CFUN=1,1 to make the configuration take
                                     effect.)
OK

AT+QIACT?                            //Query the state of context.
+QIACT: 1,1,1,"172.22.86.226"

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

```
AT+QIACT=1                                  //Activate context 1.
OK                                          //Activated successfully.

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

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

AT+QSSLCFG="ciphersuite",1,0x0005           //Set SSL cipher suite as 0x0005, which means RC4-SHA.
OK

AT+QSSLCFG="seclevel",1,2                   //Set SSL verification level as 2, which means that a CA,
                                            certificate, client certificate and client private key should be
                                            uploaded with AT+QFUPL.
OK

AT+QFUPL="cacert.pem"                       //Upload the CA certificate to UFS.
CONNECT
<Input file bin data>
+QFUPL:1216,7648

OK

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

OK

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

OK

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

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

```
OK

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

AT+QHTTPURL=45,80                       //Set the URL of the HTTPS server that will be accessed
                                        and timeout value as 80 s.
CONNECT
HTTPs://220.180.239.212:8011/processorder.php    //Input the URL whose length is 45 bytes. (This
                                                  URL is only an example. Input the correct URL in
                                                  a practical test.)
OK

AT+QHTTPPOST=48,80,80                   //Send HTTPS POST request. The maximum input body time
                                        and the maximum response time are 80 s each.
CONNECT
Message=1111&Appleqty=2222&Orangeqty=3333&find=1      //Input HTTPS POST body whose length
                                                       is 48 bytes. (This post body is only an
                                                       example. Input the correct one in a
                                                       practical test.)
OK

+QHTTPPOST: 0,200,285                   //If the HTTPS response header contains CONTENT-
                                        LENGTH information, the <content_length> (285) 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
                                        s.
CONNECT                                 //HTTPS response body has been read successfully.
<html>
<head>
<title>Quectel's Auto Parts - Order Results</title>
</head>
<body>
<h1>Quectel's Auto Parts</h1>
<h2>Order Results</h2>

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

OK
```

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

3.2.2.2. HTTPS POST Body Obtained from File System

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

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

AT+QIACT?                           //Query the state of context.
OK                                  //No context activated currently.

AT+QICSGP=1,1,"UNINET","","",1      //Configure PDP context 1. China Unicom APN: UNINET.
                                  (Then set AT+CFUN=1,1 to make the configuration take effect.)
OK

AT+QIACT?                           //Query the state of context.
+QIACT: 1,1,1,"172.22.86.226"

OK
//The first PDP is activated by default. If it is queried unactivated, use AT+QIACT=1 to activate it.
AT+QIACT=1                          //Activate context 1.
OK                                  //Activated successfully.

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

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

AT+QSSLCFG="ciphersuite",1,0x0005//Set SSL cipher suite as 0x0005, which means RC4-SHA.
OK

AT+QSSLCFG="seclevel",1,2           //Set SSL verification level as 2, which means that a CA certificate,
                                    a client certificate and a client private key should be uploaded
                                    with AT+QFUPL.
OK

AT+QFUPL="cacert.pem"                //Upload the CA certificate to UFS.
CONNECT
<Input file bin data>
+QFUPL:1216,7648
```

```
OK

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

OK

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

OK

AT+QSSLCFG="cacert",1,"UFS:cacert.pem"
OK

AT+QSSLCFG="clientcert",1,"UFS:clientcert.pem"
OK

AT+QSSLCFG="clientkey",1,"UFS:clientkey.pem"
OK

AT+QHTTPURL=45,80                              //Set the URL of the HTTPS server that will be
                                               accessed and timeout value as 80 s.
CONNECT
https://220.180.239.212:8011/processorder.php  //Input URL whose length is 45 bytes. (This URL is
                                               only an example. Input the correct URL in a practical
                                               test.)
OK
//POST request information from UFS file, and read the HTTPS response and store it to a UFS file.
AT+QHTTPPOSTFILE="UFS:5.txt",80                //Send HTTPS POST request. HTTPS POST body
                                               is obtained from UFS:5.txt. The maximum response
                                               time is 80 s.
OK

+QHTTPPOSTFILE: 0,200,177                      //After HTTPS POST request is sent successfully,
                                               the HTTPS response body can be read via either
                                               AT+QHTTPREAD or AT+QHTTPREADFILE.
AT+QHTTPREADFILE="UFS:6.txt",80                //Read the HTTPS response body and store it to
```

	UFS:6.txt. The maximum time to wait for an HTTPS session to be closed is 80 s.
OK	
+QHTTPREADFILE: 0	//HTTPS response body has been stored successfully.

4 Error Handling

4.1. Executing HTTP(S) AT Command Failure

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

4.2. PDP Activation Failure

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

1. Query whether the PS domain is attached or not with **AT+CGATT?**. If not, 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=<contextID>** and make sure the APN of the 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 supports maximum three PDP contexts activated simultaneously.

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

4.3. DNS Parse Failure

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

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 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 then re-activate the PDP context with **AT+QIACT**.

4.4. Entering Data Mode Failure

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

4.5. Sending GET/POST Requests Failure

If a failed result is received after executing **AT+QHTTPGET**, **AT+QHTTPGETEX**, **AT+QHTTPPOST** and **AT+QHTTPPOSTFILE**, 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** requests.
3. Make sure the PDP context has been activated successfully.

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

4.6. Reading Response Failure

Before reading responses with **AT+QHTTPREAD** and **AT+QHTTPREADFILE**, execute **AT+QHTTPGET**, **AT+QHTTPPOST** and **AT+QHTTPPOSTFILE** and wait until the following URC information is reported:
+QHTTPGET: <err>[,<httprspcode>[,<content_length>]]
+QHTTPPOST: <err>[,<httprspcode>[,<content_length>]]
+QHTTPPOSTFILE: <err>[,<httprspcode>[,<content_length>]]

In case of errors during the execution of **AT+QHTTPREAD** and **AT+QHTTPREADFILE**, such as **+CME ERROR: 717** (717: HTTP(S) socket read error), resend HTTP(S) GET/POST requests to HTTP(S) server with **AT+QHTTPGET**, **AT+QHTTPPOST** and **AT+QHTTPPOSTFILE**. If sending GET/POST requests to

HTTP(S) server fails, see ***Chapter 4.5***.

5 Summary of ERROR Codes

The error code **<err>** indicates an error related to mobile equipment or network. The detailed information about **<err>** is presented 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 meaning of **<httprspcode>** is presented in the following table.

Table 4: Summary of HTTP(S) Response Codes

<httprspcode>	**Meaning**
200	OK
403	Forbidden
404	Not found
409	Conflict
411	Length required
500	Internal server error

7 Appendix References

Table 5: Related Documents

Document Name
[1] Quectel_LTE_Standard(A)_Series_TCP(IP)_Application_Note
[2] Quectel_LTE_Standard(A)_Series_SSL_Application_Note
[3] Quectel_LTE_Standard(A)_Series_ AT_Commands_Manual

Table 6: Terms and Abbreviations

Abbreviation	Description
APN	Access Point Name
CA	Certification Authority
COM port	Communication Port
CR	Carriage Return
DNS	Domain Name Server
DTR	Data Terminal Ready
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ID	Identification
IP	Internet Protocol
LF	Line Feed (a new line)
LTE	Long-Term Evolution

PDP	Packet Data Protocol
PPP	Point-to-Point Protocol
PS	Packet Switch
SSL	Security Socket Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UART	Universal Asynchronous Receiver/Transmitter
UFS	UNIX File System
URC	Unsolicited Result Code
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal Serial Bus
(U)SIM	(Universal) Subscriber Identity Module