EC200U&EG91xU&EG915G Series

HTTP(S) Application Note

LTE Standard Module Series

Version: 1.3

Date: 2024-04-29

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

Use and Disclosure Restrictions

License Agreements

Copyright

About the Document

Revision History

Version	Date	Author	Description
-	2021-04-28	Herry GENG	Creation of the document
1.0	2020-05-06	Herry GENG	First official release
1.1	2021-08-20	Herry GENG	Added an applicable module series EG915U.
1.2	2023-09-11	Lambert ZHAO	1. Added an applicable module EG912U-GL. 2. Updated the range of <content_type> (Chapter 2.3.1). 3. Updated the range of <URL_length> (Chapter 2.3.2). 4. Updated the range of <data_length> (Chapter 2.3.3). 5. Updated the URC format of AT+QHTTPGET, AT+QHTTPPOST, AT+QHTTPPOSTFILE, AT+QHTTPPUT and AT+QHTTPPUTFILE (Chapters 2.3.3 & 2.3.5–2.3.8 & 4.6).
1.3	2024-04-29	Kevin WANG	Updated the applicable modules: ● Updated EG912U-GL to EG912U series. ● Added EG915G-EU.

Contents

Table Index

1 Introduction

Quectel LTE Standard EC200U series, EG91xU family and EG915G-EU modules provide HTTP(S) applications to HTTP(S) server.

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 the AT commands defined for HTTP(S).

1.1. Applicable Modules

Table 1: Applicable Modules

Module Family	Module
-	EC200U Series
EG91xU	EG912U Series
	EG915U Series
-	EG915G-EU

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

With TCP/IP AT commands applicable for EC200U series, EG91xU family and EG915G-EU modules, a PDP context can be configured, namely activate/deactivate the PDP context and query the context status. With EC200U series, EG91xU family and EG915G-EU modules HTTP(S) AT commands, HTTP(S) GET/POST requests can be sent to HTTP(S) server, HTTP(S) response can be read from HTTP(S) server. The general process is as follows:

Step 1: Configure **<APN>**, **<username>**, **<password>** and other parameters of a PDP context by **AT+QICSGP**. See ***document [1]*** for details.

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

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

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

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

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

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

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

1.3. Description of HTTP(S) Request Header

1.3.1. Customize HTTP(S) Request Header

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

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

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

```
POST /processorder.php HTTP/1.1<CR><LF>
Host: 220.180.239.212:8011<CR><LF>
Accept: */*<CR><LF>
User-Agent: QUECTEL_MODULE<CR><LF>
Connection: Keep-Alive<CR><LF>
Content-Type: application/x-www-form-urlencoded<CR><LF>
Content-Length: 48<CR><LF>
<CR><LF>
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. HTTP(S) response header information can be obtained by configuring **<response_header>** to 1 via **AT+QHTTPCFG**, and then HTTP(S) response header will be outputted with HTTP(S) response body after executing **AT+QHTTPREAD** or **AT+QHTTPREADFILE**.

1.4. Description of Data Mode

The COM port of the above applicable EC200U series, EG91xU family and EG915G-EU modules have two working modes: AT command mode and data mode. In AT command mode, the inputted data via COM port will be regarded as AT command. While in data mode, it will be regarded as data.

Inputting **+++** or pulling up DTR (**AT&D1** should be set first) can make the COM port exit from data mode. To prevent the **+++** from being misinterpreted as data, the following sequence should be followed:

1) Do not input any character within 1s or longer before inputting **+++**.
2) Input **+++** within 1 s, and no other characters can be inputted during the time.
3) Do not input any character within 1 s after **+++** has been inputted.

When **AT+QHTTPURL**, **AT+QHTTPPOST**, **AT+QHTTPPUT** and **AT+QHTTPREAD** are executed, the COM port will enter data mode. If **+++** or DTR is used to make the port exit from data mode, the executing procedure of these commands will be interrupted before the response is returned. In such case, the COM port cannot reenter data mode by executing **ATO** command.

2 Description of HTTP(S) AT Commands

2.1. AT Command Introduction

2.1.1. Definitions

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

2.1.2. AT Command Syntax

The **AT** or **at** prefix must be added at the beginning of each command line. Entering **<CR>** will terminate a command line. Commands are usually followed by a response that includes **<CR><LF><response><CR><LF>**. Throughout this document, only the response **<response>** will be presented, **<CR><LF>** are omitted intentionally.

Table 2: Type of AT Commands and Responses

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.

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. AT Command Description

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

The command configures the parameters for HTTP(S) server, including configuring a PDP context ID, customizing HTTP(S) request header, outputting HTTP(S) response header and 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: "url",<url_value>** **+QHTTPCFG: "header",<header_value>** **+QHTTPCFG: "auth",<user_pwd>** **+QHTTPCFG: "form/data",<name>,<file_name>,<content_type>** **+QHTTPCFG: "reset"** **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: **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: **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: **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: **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: **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: **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: **OK** Or **+CME ERROR: <err>**
Write Command **AT+QHTTPCFG="url"[,<url_value>]**	Response If the optional parameter is omitted, query the current settings: **+QHTTPCFG: "url",<url_value>** **OK** If the optional parameter is specified: **OK** Or **+CME ERROR: <err>**
Write Command **AT+QHTTPCFG="header"[,<header_value>]**	Response If the optional parameter is omitted, query the current settings: **+QHTTPCFG: "header",<header_value>** **[…]** **OK** If the optional parameter is specified: **OK** Or **+CME ERROR: <err>**

Write Command **AT+QHTTPCFG="auth"[,<user_pwd>]**	Response If the optional parameter is omitted, query the current settings: **+QHTTPCFG: "auth",<user_pwd>** **OK** If the optional parameter is specified: **OK** Or **+CME ERROR: <err>**
Write Command **AT+QHTTPCFG="form/data"[,<name>[,<file_name>[,<content_type>]]]**	Response If the optional parameters are omitted, query the current settings: **+QHTTPCFG: "form/data",<name>,<file_name>,<content_type>** **[…]** **OK** If any of the optional parameters is specified: **OK** Or **+CME ERROR: <err>**
Write Command **AT+QHTTPCFG="reset"**	Response **OK** Or **+CME ERROR: <err>**
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: "url",<url_value>** **+QHTTPCFG: "auth",<user_pwd>** **OK**
Maximum Response Time	/
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<contextID>	Integer type. PDP context ID. Range: 1–7. Default: 1.
<request_header>	Integer type. Disable or enable to customize HTTP(S) request header. 0 Disable 1 Enable
<response_header>	Integer type. Disable or enable to output HTTP(S) response header. 0 Disable 1 Enable
<sslctxID>	Integer type. SSL context ID used for HTTP(S). Range: 0–5. Default: 1. SSL parameters should be configured by **AT+QSSLCFG**. For details, See ***document [2]***.
<content_type>	Integer type. Data type of HTTP(S) body. 0 application/x-www-form-urlencoded 1 text/plain 2 application/octet-stream 3 multipart/form-data 4 application/json 5 image/jpeg
<auto_outrsp>	Integer type. Disable or enable auto output of HTTP(S) response data. If auto output of HTTP(S) response data is enabled, then **AT+QHTTPREAD** and **AT+QHTTPREADFILE** will fail to execute. 0 Disable 1 Enable
<closedind>	Integer type. Disable or enable report indication of closed HTTP(S) session. 0 Disable 1 Enable
<url_value>	String type. The URL of HTTP(S).
<header_value>	String type. HTTP(S) request header line/header field name, such as: "Content-type: text/plain" or "Content-type".
<user_pwd>	String type. User name and password, the format is: "username:password".
<name>	String type. The name value of form-data.
<file_name>	String type. The filename value of form-data.
<content_type>	String type. The content-type value of form-data.
<err>	The error code of the operation. See ***Chapter 5***.

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

URL must begin with http:// or https://, which indicates the access to an HTTP or HTTPS server.

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

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

Parameter

<URL_length>	Integer type. The length of URL. Range: 7–2048. Unit: byte.
<timeout>	Integer type. The maximum time for inputting URL. Range: 1–65535. Default: 60. Unit: second.
<err>	The error code of the operation. See ***Chapter 5***.

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

According to the configured **<request_header>** parameter in **AT+QHTTPCFG="requestheader"[,<request_header>]**, **AT+QHTTPGET** Write Command has two different formats. If **<request_header>** is set to 1, after **AT+QHTTPGET** has been sent, **CONNECT** may be outputted in 125 s to indicate that the HTTP(S) is connected successfully. If it is not outputted during the time, then **+CME ERROR: <err>** will be outputted.

After **AT+QHTTPGET** Write Command has been sent, it is recommended to wait for a specific period of time (see the maximum response time below) for URC **+QHTTPGET: <err>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is reported.

In **+QHTTPGET: <err>[,<httprspcode>[,<content_length>]]**, the **<httprspcode>** parameter 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**
Write Command If **<request_header>** equals 0 (disable to customize HTTP(S) request header) **AT+QHTTPGET[=<rsptime>]**	Response If the parameter format is correct and no other errors occur: **OK** When the module has received response from HTTP(S) server, it will report the following URC: **+QHTTPGET: <err>[,<httprspcode>[,<content_length>]]** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Write Command If **<request_header>** equals 1 (enable to customize HTTP(S) GET request header) **AT+QHTTPGET=<rsptime>,<data_length>[,<input_time>]**	Response If HTTP(S) server is connected successfully: **CONNECT** TA switches to transparent access mode, and 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 reached, but the length of received data is less than **<data_length>**, TA will return to command mode and report the following code: **+QHTTPGET: <err>**

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

Parameter

<rsptime>	Integer type. Range: 1–65535. Default: 60. Unit: second. It is used to configure the timeout for the HTTP(S) GET response **+QHTTPGET: <err>,<httprspcode>[,<content_length>]** to be outputted after **OK** is returned.
<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: 60. Unit: second.
<httprspcode>	Http response code. See ***Chapter 6*** for details.
<request_header>	Integer type. Disable or enable to customize HTTP(S) request header. 0 Disable 1 Enable
<content_length>	Integer type. The length of HTTP(S) response body. Unit: byte.
<err>	The error code of the operation. See ***Chapter 5***.

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

Like the way of reading files, MCU can get data from HTTP(S) server with specified position and specified length by **AT+QHTTPGETEX**, and this command is only executable in the condition of **AT+QHTTPCFG="requestheader",0**. After that, HTTP(S) server will always respond to the GET request that is used to get data with specified position and length with **206** code.

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	The command takes effect immediately. The configurations are not saved.

Parameter

<rsptime>	Integer type. Range: 1–65535. Default: 60. Unit: second. It is used to configure the timeout for the HTTP(S) GET response **+QHTTPGET: <err>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is returned.
<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.
<httprspcode>	HTTP response code. See ***Chapter 6*** for details.
<content_length>	Integer type. The length of HTTP(S) response body. Unit: byte.
<err>	The error code of the operation. See ***Chapter 5***.

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

The command sends HTTP(S) POST request via UART/USB. According to the configured **<request_header>** parameter in **AT+QHTTPCFG="requestheader"[,<request_header>]**, the **AT+QHTTPPOST** Write Command has two different formats. If **<request_header>** is set to 0, HTTP(S) POST body should be inputted via UART/USB port. If **<request_header>** is set to 1, then both HTTP(S) POST header and body should be inputted via UART/USB port.

After **AT+QHTTPPOST** has been sent, **CONNECT** may be outputted in 125 s to indicate the connection is successful. If it is not received during the time, **+CME ERROR: <err>** will be outputted.

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

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 to customize HTTP(S) request header) **AT+QHTTPPOST=<data_length>[,<input_time>,<rsptime>]**	Response If the parameter format is correct and HTTP(S) server is connected successfully and HTTP(S) request header is sent completely, it will prompt to input body: **CONNECT** TA switches to transparent access mode, and 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: **+QHTTPPOST: <err>** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Write Command If **<request_header>** equals 1 (enable to customize HTTP(S) request header) **AT+QHTTPPOST=<data_length>[,<input_time>,<rsptime>]**	Response If the parameter format is correct and HTTP(S) server is connected successfully: **CONNECT** TA switches to the transparent access mode, and 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: **+QHTTPPOST: <err>** If the parameter format is incorrect or other errors occur:

	+CME ERROR: <err>
Maximum Response Time	Determined by network and **<rsptime>**
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<data_length>	Integer type. If **<request_header>** is 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. Range: 1–65535. Default: 60. Unit: second. It is used to configure the timeout for the HTTP(S) POST response **+QHTTPPOST: <err>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is returned.
<httprspcode>	Http response code. See ***Chapter 6*** for details.
<request_header>	Integer type. Disable or enable to customize HTTP(S) request header. 0 Disable 1 Enable
<content_length>	Integer type. The length of HTTP(S) response body. Unit: byte.
<err>	The error code of the operation. See ***Chapter 5***.

2.3.6. 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 by **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.

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

It is recommended to wait for a specific period of time (see 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	Response

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

Parameter

<file_name>	String type. File name. The max length of file name is 132 bytes.
<rsptime>	Integer type. Range: 1–65535. Default: 60. Unit: second. It is used to configure the timeout for the HTTP(S) POST response **+QHTTPPOSTFILE: <err>[,<httprspcode>[,<content_length>]]** to be outputted after **OK** is returned.
<httprspcode>	HTTP response code. See ***Chapter 6*** for details.
<request_header>	Integer type. Disable or enable to customize HTTP(S) request header. If **<request_header>** equals 1, the specified file must contain HTTP(S) request header information. 0 Disable 1 Enable
<content_length>	Integer type. The 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 (not send the file currently, waiting to be sent with the file configured when **<post_mode>**=2. 2 Send the files configured when **<post_mode>**=1 and 2 in order (only support two files sent together)
<err>	The error code of the operation. See ***Chapter 5***.

2.3.7. 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 configured **<request_header>** parameter in **AT+QHTTPCFG="requestheader"[,<request_header>]** command, the **AT+QHTTPPUT** Write Command 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, then both HTTP(S) PUT header and body should be inputted via UART/USB port.

After **AT+QHTTPPUT** command has been sent, **CONNECT** may be outputted in 125 s to indicate the connection is successful. Otherwise, **+CME ERROR: <err>** will be outputted.

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

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 If the parameter format is correct and HTTP(S) server is connected successfully and HTTP(S) request header is sent completely: **CONNECT** TA switches to transparent access mode, and the HTTP(S) PUT body can be inputted. When the total size of the inputted data reaches **<data_length>**, TA will return to command mode and report the following code: **OK** When the module has received 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: **+QHTTPPUT: <err>** 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 If the parameter format is correct and HTTP(S) server is connected successfully: **CONNECT** TA switches to the transparent access mode, and the HTTP(S) PUT header and body can be inputted. When the total size of the inputted data reaches **<data_length>**, TA will return to command mode and report the following code: **OK** When the module has received 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: **+QHTTPPUT: <err>** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	Determined by network and **<rsptime>**
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<data_length>	Integer type. If **<request_header>** is 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. Range: 1–65535. Default value: 60. Unit: second. It is used to configure the timeout for the HTTP(S) PUT response **+QHTTPPOST: <err>,[<httprspcode>[,<content_length>]]** to be outputted after **OK** is returned.
<httprspcode>	HTTP server response code. See ***Chapter 6***.
<request_header>	Integer type. Disable or enable to customize HTTP(S) request header. 0 Disable 1 Enable

<content_length>	Integer type. The length of HTTP(S) response body. Unit: byte.
<err>	The error code of the operation. See ***Chapter 5***.

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

This command sends HTTP(S) PUT request via file. According to the **<request_header>** in **AT+QHTTPCFG="requestheader"[,<request_header>]** command, the file operated by **AT+QHTTPPUTFILE** command 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>** parameter can only be reported when **<err>** equals 0.

It is recommended to wait for a specific period of time (see 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 **<put_mode>**s**)]** **OK**
Write Command **AT+QHTTPPOSTFILE=<file_name>,<rsptime>[,<put_mode>]**	Response If parameter format is correct and HTTP(S) server is connected successfully: **OK** When the module has received response from HTTP(S) server, it will report the following URC: **+QHTTPPUTFILE: <err>[,<httprspcode>[,<content_length>]]** If parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	Determined by network and **<rsptime>**
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<file_name>	String type. File name. The max length of file name is 132 bytes.
<rsptime>	Integer type. Range: 1–65535. Default: 60. Unit: second. It is used to configure the timeout for the HTTP(S) POST response **+QHTTPPOSTFILE: <err>[,<httprspcode>,<content_length>]** to be outputted after **OK** is returned.
<httprspcode>	HTTP server response code. See ***Chapter 6***.
<request_header>	Integer type. Disable or enable to customize HTTP(S) request header. If **<request_header>** equals 1, the specified file must contain HTTP(S) request header information. 0 Disable 1 Enable
<content_length>	Integer type. The length of HTTP(S) response body. Unit: byte.
<put_mode>	Integer type. The mode of HTTP(S) sending files. 0 Send file content directly 1 Record and save the file, do not send it temporarily, wait to send it with the file configured when **<put_mode>**=2 2 Send the file, together with the file saved when **<put_mode>**=1 (only support two files sent together)
<err>	The error code of the operation. See ***Chapter 5***.

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

After sending HTTP(S) GET/POST requests, HTTP(S) response information can be retrieved from HTTP(S) server via UART/USB port by **AT+QHTTPREAD**. And **+QHTTPGET: <err>[,<httprspcode>[,<content_length>]]**, **+QHTTPPOST: <err>[,<httprspcode>[,<content_length>]]** or **+QHTTPPOSTFILE: <err>[,<httprspcode>[,<content_length>]]** information must be received before executing **AT+QHTTPREAD**.

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 read successfully: **CONNECT** <Output HTTP(S) response information> **OK** When body is read over or **<wait_time>** reaches, it will report: **+QHTTPREAD: <err>**

	b) If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	/
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<wait_time>	Integer type. The maximum interval time between receiving two packets of data. Range: 1–65535. Default: 60. Unit: second.
<err>	The error code of the operation. See ***Chapter 5***.

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

After sending HTTP(S) GET/POST requests, HTTP(S) response information can be retrieved from HTTP(S) server via file by **AT+QHTTPREADFILE**. And **+QHTTPGET: <err>[,<httprspcode>[,<content_length>]]**, **+QHTTPPOST: <err>[,<httprspcode>[,<content_length>]]** or **+QHTTPPOSTFILE: <err>[,<httprspcode>[,<content_length>]]** information must be received before executing **AT+QHTTPREADFILE**.

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 If the parameter format is correct: **OK** When body is read over or **<wait_time>** reaches, it will report: **+QHTTPREADFILE: <err>** If the parameter format is incorrect or other errors occur: **+CME ERROR: <err>**
Maximum Response Time	Determined by **<wait_time>**
Characteristics	The command takes effect immediately. The configurations are not saved.

Parameter

<wait_time>	Integer type. The maximum interval time between receiving two packets of data. Range: 1–65535. Default: 60. Unit: second.
<file_name>	String type. File name. The maximum length of the file name is 132 bytes.
<err>	The error code of the operation. See ***Chapter 5***.

2.3.11. 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 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	The command takes effect immediately. The configurations are not saved.

Parameter

<err>	The error code of the operation. See ***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                                    //Only returning OK means that there is no activated PDP
                                       context currently.
AT+QICSGP=1,1,"UNINET","","",1        //Configure PDP context 1. APN is UNINET for China
                                      Unicom. (Then set AT+CFUN=1,1 to make the
                                      configuration take effect.)
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 timeout value as
                                      80 s.
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 s.
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 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>                             //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 s.
OK

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

3.1.2. Send HTTP POST Request and Read the Response

3.1.2.1. HTTP POST Body Obtained 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                                    //Only returning OK means that there is no activated PDP
                                        context currently.
AT+QICSGP=1,1,"UNINET","","",1        //Configure PDP context 1. APN is UNINET for China Unicom.
                                      (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
AT+QHTTPURL=59,80                     //Set the URL which 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. 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 is 80 s and the maximum
                                   response time is 80 s.
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,
                                the <content_length> information is reported.
AT+QHTTPREAD=80               //Read HTTP response body and output it via UART. The maximum time
                              to wait for HTTP session to be closed is 80 s.
CONNECT
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="httpHTTPs://api.efxnow.com/webservices2.3">Message='HelloQuectel' ASCII:72
101 108 108 111 81 117 101 99 116 101 108 </string>      //Output HTTP response information.
OK
```

```
+QHTTPREAD: 0                    //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                                     //Only returning OK means that there is no activated PDP context
                                         currently.
AT+QICSGP=1,1,"UNINET","","",1         //Configure PDP context 1. APN is UNINET for China Unicom.
                                       (Then set AT+CFUN=1,1 to make the configuration take effect.)
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 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. Please input the
                                                              correct URL in practical test.)
OK
//POST request information from UFS file, and read HTTP response information and store it to UFS file.
AT+QHTTPPOSTFILE="UFS:2.txt",80            //Send HTTP(S) POST request. POST body is obtained
                                           from UFS:2.txt, and the maximum response time is 80 s.
OK

+QHTTPPOSTFILE: 0,200,177                 //After HTTP POST request is sent successfully,
                                          AT+QHTTPREADFILE can be executed.
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 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                                     //Only returning OK means that there is no activated PDP
                                         context currently.
AT+QICSGP=1,1,"UNINET","","",1         //Configure PDP context 1. APN is UNINET for China Unicom.
                                       (Then set AT+CFUN=1,1 to make the configuration take effect.)
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 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. Please input the
                                                              correct URL in practical test.)
OK
AT+QHTTPPOST=20,80,80                  //Send HTTP PUT request and HTTP PUT body is obtained
                                       via UART. The maximum input body time is 80 s and the
                                       maximum response time is 80 s.
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,
                               the <content_length> information is reported.
AT+QHTTPREAD=80              //Read HTTP response body and output it via UART. The maximum time
                             to wait for HTTP session to be closed is 80 s.
CONNECT
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="httpHTTPs://api.efxnow.com/webservices2.3">Message='HelloQuectel' ASCII:72
```

```
101 108 108 111 81 117 101 99 116 101 108 </string>      //Output HTTP response information.
OK

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

3.1.3.2. HTTP PUTBody 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                                   //Only returning OK means that there is no activated PDP
                                       context currently.
AT+QICSGP=1,1,"UNINET","","",1       //Configure PDP context 1. APN is UNINET for China Unicom.
                                     (Then set AT+CFUN=1,1 to make the configuration take effect.)
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 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. 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+QHTTPPOSTFILE="UFS:2.txt",80          //Send HTTP(S) PUT request. PUT body is obtained
                                         from UFS:2.txt, and the maximum response time is 80 s.
OK

+QHTTPPOSTFILE: 0,200,177               //After HTTP POST request is sent successfully,
                                        AT+QHTTPREADFILE can be executed.
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 is stored successfully.
```

3.2. Access to HTTPS Server

3.2.1. Send HTTPS GET Request and Read the Response

The following examples show how to send 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                                        //Only returning OK means that there is no activated PDP
                                           context currently.
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.
CONNECT
https://www.alipay.com                   //Input URL whose length is 22 bytes. (This URL is only an
                                         example. Please input the correct URL in practical test.)
OK
AT+QHTTPGET=80                           //Send HTTPS GET request and the maximum response
time                                     is 80 s.
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 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>                                     //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 s.
OK

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

3.2.2. Send HTTPS POST Request and Read the Response

3.2.2.1. HTTPS POST Body Obtained 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                                          //Only returning OK means that there is no activated PDP
                                            context currently.
```

```
AT+QICSGP=1,1,"UNINET","","",1        //Configure PDP context 1. APN is UNINET for China Unicom.
                                      (Then set AT+CFUN=1,1 to make the configuration take effect.)
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 by
                                      AT+QFUPL.
OK
AT+QSSLCFG="cacert",1,"UFS:cacert.pem"
OK
AT+QSSLCFG="clientcert",1,"UFS:clientcert.pem"
OK
AT+QSSLCFG="clientkey",1,"UFS:clientkey.pem"
OK
AT+QHTTPURL=45,80                     //Set the URL which will be accessed and timeout value as 80 s.
CONNECT
HTTPs://220.180.239.212:8011/processorder.php    //Input URL whose length is 45 bytes. (This URL
                                                 is only an example. Please input the correct URL
                                                 in practical test.)
OK
AT+QHTTPPOST=48,80,80                 //Send HTTPS POST request. HTTPS POST body is obtained
                                      from UART. The maximum input body time is 80 s and the
                                      maximum response time is 80 s.
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, the <content_length> information is 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                            //Read HTTPS response information 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                                 //Only returning OK means that there is no activated PDP
                                    context currently.
AT+QICSGP=1,1,"UNINET","","",1     //Configure PDP context 1. APN is UNINET for China Unicom.
                                    (Then set AT+CFUN=1,1 to make the configuration take effect.)
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 by AT+QFUPL.
```

```
OK
AT+QSSLCFG="cacert",1,"UFS:cacert.pem"
OK
AT+QSSLCFG="clientcert",1,"UFS:clientcert.pem"
OK
AT+QSSLCFG="clientkey",1,"UFS:clientkey.pem"
OK
AT+QHTTPURL=45,80                   //Set the URL which will be accessed and timeout value as 80 s.
CONNECT
https://220.180.239.212:8011/processorder.php    //Input URL whose length is 45 bytes. (This URL is
                                                 only an example. Please input the correct URL in
                                                 practical test.)
OK
//POST request information from UFS file, and read HTTPS response information and store it to 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,
                                 AT+QHTTPREAD 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 0 s.
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 HTTPS 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                               //Only returning OK means that there is no activated PDP
                                  context currently.
AT+QICSGP=1,1,"UNINET","","",1   //Configure PDP context 1. APN is UNINET for China Unicom.
                                 (Then set AT+CFUN=1,1 to make the configuration take effect.)
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 by AT+QFUPL.
OK
AT+QSSLCFG="cacert",1,"UFS:cacert.pem"
OK
AT+QSSLCFG="clientcert",1,"UFS:clientcert.pem"
OK
AT+QSSLCFG="clientkey",1,"UFS:clientkey.pem"
OK
AT+QHTTPURL=45,80                  //Set the URL which will be accessed and timeout value as 80 s.
CONNECT
HTTPs://220.180.239.212:8011/processorder.php     //Input URL whose length is 45 bytes. (This URL is
                                                  only an example. Please input the correct URL in
                                                  practical test.)
OK
AT+QHTTPPOST=48,80,80                  //Send HTTPS PUT request and HTTPS PUT body is obtained
                                            via UART. The maximum input body time is 80 s and the
                                            maximum response time is 80 s.
CONNECT
Message=1111&Appleqty=2222&Orangeqty=3333&find=1          //Input HTTPS PUT body whose
                                                          length is 48 bytes. (The PUT body is
                                                          only an example. Please input the
                                                          correct PUT body in practical test.)
OK

+QHTTPPOST: 0,200,285          //If the HTTPS response header contains Content-Length information,
                               the <content_length> information is reported.
AT+QHTTPREAD=80                //Read HTTPS response body and output it via UART. The maximum time
                               to wait for HTTP session to be closed is 80 s.
CONNECT
<html>
<head>
<title>Quectel's Auto Parts - Order Results</title>
```

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

<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                    //HTTP 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 PUT body via 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                                     //Only returning OK means that there is no activated PDP
                                        context currently.
AT+QICSGP=1,1,"UNINET","","",1        //Configure PDP context 1. APN is UNINET for China Unicom.
                                      (Then set AT+CFUN=1,1 to make the configuration take effect.)
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 by AT+QFUPL.
OK
AT+QSSLCFG="cacert",1,"UFS:cacert.pem"
OK
AT+QSSLCFG="clientcert",1,"UFS:clientcert.pem"
OK
```

```
AT+QSSLCFG="clientkey",1,"UFS:clientkey.pem"
OK
AT+QHTTPURL=45,80                          //Set the URL which will be accessed and timeout value as 80 s.
CONNECT
https://220.180.239.212:8011/processorder.php    //Input URL whose length is 45 bytes. (This URL is
                                                   only an example. Please input the correct URL in
                                                   practical test.)
OK
//PUT request information from UFS file, and read HTTP response information and store it to UFS file.
AT+QHTTPPOSTFILE="UFS:5.txt",80        //Send HTTP(S) PUT request. PUT body is obtained
                                       from UFS:5.txt, and the maximum response time is 80 s.
OK

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

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

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 by **AT+QIACT**, please check the following configurations:

1. Query whether the PS domain is attached or not by **AT+CGATT?**. If not, please execute **AT+CGATT=1** to attach the PS domain.
2. Query the PS domain status by **AT+CGREG?** and make sure the PS domain has been registered.
3. Query the PDP context parameters by **AT+QICSGP** and make sure the APN of specified PDP context has been set.
4. Make sure the specified PDP context ID is neither used by PPP nor activated by **AT+CGACT**.

If all above configurations are correct, but activating the PDP context by **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+QHTTPPOSTFILE**, **AT+QHTTPPUT** and **AT+QHTTPPOSTFILE**, if **+CME ERROR: 714** (714: HTTP(S) DNS error) is returned, please check the following aspects:

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

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

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

4.4. Entering Data Mode Fails

When executing **AT+QHTTPURL**, **AT+QHTTPGET**, **AT+QHTTPPOST**, **AT+QHTTPPOSTFILE**、**AT+QHTTPPUT** ,**AT+QHTTPPUTFILE** and **AT+QHTTPREAD**, 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/PUT 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/PUT commands.
3. Make sure the PDP context has been activated successfully.

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

4.6. Reading Response Fails

Before reading response by **AT+QHTTPREAD** and **AT+QHTTPREADFILE**, execute **AT+QHTTPGET**, **AT+QHTTPPOST**, **AT+QHTTPPOSTFILE**、**AT+QHTTPPUT** and **AT+QHTTPPUTFILE** 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>]]

During executing **AT+QHTTPREAD** and **AT+QHTTPREADFILE**, if you 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 by **AT+QHTTPGET**, **AT+QHTTPPOST**, **AT+QHTTPPOSTFILE**, **AT+QHTTPPUT** and **AT+QHTTPPUTFILE.** If sending GET/POST/PUT requests to HTTP(S) server fails, see ***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
731	Wondblock
732	SSL Handshake Failed

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
206	Partial Content
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_EC200U&EG91xU&EG915G_Series_TCP(IP)_Application_Note
[2] Quectel_EC200U&EG91xU&EG915G_Series_SSL_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	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ID	Identification
IP	Internet Protocol
LTE	Long-Term Evolution
MCU	Microprogrammed Control Unit
PDP	Packet Data Protocol
PPP	Point-to-Point Protocol
PS	Packet Switch

SSL	Security Socket Layer
TA	Terminal Adapter
UART	Universal Asynchronous Receiver/Transmitter
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URC	Unsolicited Result Code
USB	Universal Serial Bus
(U)SIM	(Universal) Subscriber Identity Module
UFS	UNIX File System