# HTTP Service
# AT Commands

**GSM_HTTP_ATC_V1.2**

# Contents

# 1. Introduction

Module has an internal TCP/IP stack that is driven by AT commands and enables the host application to easily access the Internet service. It includes TCP service, UDP service, FTP service and HTTP service, etc. This document is a reference guide to all the AT commands and responses defined for HTTP Service. The advantage of this solution is that it eliminates the need for the application manufacturer to implement own HTTP protocol, thus minimizing cost and time to integrate Internet connectivity into a new or existing host application.

This document is applicable to all GSM modules.

## 1.1. Reference

**Table 1: Reference**

| SN | Document name | Remark |
|---|---|---|
| [1] | GSM_TCPIP_AN.pdf | TCPIP Application Notes |
| [2] | RFC 2616 | |

# 2. AT Commands for HTTP Service

## 2.1. Overview of AT Commands for HTTP Service

| Command | Description |
|---|---|
| AT+QHTTPURL | Set HTTP Server URL |
| AT+QHTTPGET | Send HTTP GET Request |
| AT+QHTTPREAD | Read HTTP Server Response |
| AT+QHTTPPOST | Send HTTP POST Request |
| AT+QHTTPDL | Download file from http server |

Execution of above HTTP related AT commands will switch to data mode. To switch back to AT mode, you can input "+++" and this will terminate the current HTTP AT command. The interval time between the first "+" and the character before the first "+" MUST NOT be less than 500 ms and the interval time between the last "+" and the character next to the last "+" MUST NOT be less than 500 ms and the interval time between each "+" MUST be less than 1000 ms.

## 2.2. Detailed Description of AT Commands for HTTP Service

### 2.2.1. AT+QHTTPURL Set HTTP Server URL

| AT+QHTTPURL Set HTTP Server URL | |
|---|---|
| Test Command<br>AT+QHTTPURL=? | Response<br>+QHTTPURL: (1-450),(1-65535)<br><br>OK |
| | Parameter<br>See Write Command |
| Write Command<br>AT+QHTTPURL=<url_len | Response<br>CONNECT<br><br>If error is related to ME functionality:<br>ERROR<br>+CME ERROR: <err> |
| | Parameter<br>**<url_len>** The length in bytes of the URL.<br>**<input_time>** Maximum time in seconds to input URL. |
| Reference | If sending HTTP GET Request, for example, input URL path: http://api.efxnow.com/DEMOWebServices2.8/Service.asmx/Echo? |

| | |
|---|---|
| | Message=hello<br>If sending HTTP POST Request, for example, input URL path: http://api.efxnow.com/DEMOWebServices2.8/Service.asmx/Echo<br>Server address must be provided as IP address in standard dot-format (e.g. "192.168.1.1") or as server address names resolvable by a DNS server (e.g. "api.efxnow.xom"). |

### 2.2.2. AT+QHTTPGET Send HTTP GET Request

| AT+QHTTPGET Send HTTP GET Request | |
|---|---|
| Test Command<br>AT+QHTTPGET=? | Response<br>+QHTTPGET: (1-65535)<br><br>OK |
| | Parameter<br>See Write Command |
| Write Command<br>AT+QHTTPGET=<to_read | Response<br>OK<br><br>If error is related to ME functionality:<br>ERROR<br>+CME ERROR: <err> |
| | Parameter<br>**<to_read_time>** Time in seconds. AT+QHTTPREAD will be invalid if the idle time after **AT+QHTTPGET** is longer than the time of <to_read_time>. |
| Reference | |

### 2.2.3. AT+QHTTPREAD Read HTTP Server Response

| AT+QHTTPREAD Read HTTP Server Response | |
|---|---|
| Test Command<br>AT+QHTTPREAD=? | Response<br>+QHTTPREAD: (1-65535)<br><br>OK |
| | Parameter<br>See Write Command |
| Write Command<br>AT+QHTTPREAD=<wait_t | Response<br>CONNECT<br><data><br>OK |

| | |
|---|---|
| | If error is related to ME functionality:<br>ERROR<br>+CME ERROR: <err> |
| | Parameter<br>**<wait_time>** Time in seconds. It will close http session when timeout.<br>**<data>** The data of HTTP server responds. |
| Reference | |

### 2.2.4. AT+QHTTPPOST Send HTTP POST Request

| AT+QHTTPPOST Send HTTP POST Request | |
|---|---|
| Test Command<br>AT+QHTTPPOST=? | Response<br>+QHTTPPOST: (1-29696),(1-65535),(1-65535)<br><br>OK |
| | Parameter<br>See Write Command |
| Write Command<br>AT+QHTTPPOST=<body_siz | Response<br>CONNECT<br><body_data><br>OK<br><br>If error is related to ME functionality:<br>ERROR<br>+CME ERROR: <err> |
| | Parameter<br>**<body_size>** Size in bytes of the body data to POST.<br>**<input_time>** Maximum time in seconds to input the body data.<br>**<to_read_time>** Time in seconds. AT+QHTTPREAD will be invalid if the idle time after **AT+QHTTPGET** is longer than the time of <to_read_time><br>**<body_data>** Input the body data to POST from UART. |
| Reference | |

### 2.2.5. AT+QHTTPDL Download file from http server

| AT+QHTTPDL Download file from http server | | |
|---|---|---|
| Test Command<br>AT+QHTTPDL=? | Response<br>+QHTTPDL:"filename"[,<length>[,(1-65535)]]<br><br>OK | |
| | Parameter<br>See Write Command | |
| Write Command<br>AT+QHTTPDL=<br>"<filename>"[,<length><br>[, wait_time]] | Response<br>OK<br><br>Finally, if download the file successfully, response<br>**+ QHTTPDL:<dl size>,<content-length>,<errcode>**<br><br>If error is related to ME functionality:<br>ERROR<br>+CME ERROR: <err> | |
| | Parameter | |
| | **<filename>** | The path of the file to be stored, such as "RAM:1.txt" |
| | **<length>** | The maximum size of the file to be download. Default is 10240. Unit: byte. It is only used for RAM file. |
| | **<wait_time>** | Time in seconds. It will close http session when timeout. |
| | **<dl size>** | The length of data has been download |
| | <**content-length**> | The content length. If the content-length is unknown, then set it to -1. |
| | **<errcode>** | If all data has been download, the <errcode> is 0, else it is **a** numeric to indicate the type of error, please refer to the chapter 3. |
| | **<err>** | A numeric to indicate the type of error, please refer to the chapter 3. |
| Reference | | |

# 3. Supported unsolicited result codes

## 3.1. Summary of CME ERROR Codes

Final result code +CME ERROR: <err> indicates an error related to mobile equipment or network. The operation is similar to ERROR result code. None of the following commands in the same Command line is executed. Neither ERROR nor OK result code shall be returned. The following <err> is just the new <err> code for HTTP. About other <err> codes, please refer to [1].

<err> values used by common messaging commands:

| Code of <err> | Meaning |
|---|---|
| 3801 | HTTP time out |
| 3802 | HTTP busy |
| 3803 | HTTP UART busy |
| 3804 | HTTP get no request |
| 3805 | HTTP network busy |
| 3806 | HTTP network open failed |
| 3807 | HTTP network no config |
| 3808 | HTTP network deactive |
| 3809 | HTTP network error |
| 3810 | HTTP url error |
| 3811 | HTTP empty url |
| 3812 | HTTP ip addr error |
| 3813 | HTTP DNS error |
| 3814 | HTTP socket create error |
| 3815 | HTTP socket connect error |
| 3816 | HTTP socket read error |
| 3817 | HTTP socket write error |
| 3818 | HTTP socket close |
| 3819 | HTTP data encode error |
| 3820 | HTTP data decode error |
| 3821 | HTTP to read timeout |
| 3822 | HTTP response failed |
| 3823 | incoming call busy |
| 3824 | voice call busy |
| 3825 | input timeout |
| 3826 | wait data timeout |
| 3827 | wait http response timeout |
| 3828 | alloc memory fail |

| | |
|---|---|
| 3829 | HTTP need relocation |
| 4000 | Exceed max length |
| 4001 | Open file fail |
| 4002 | Write file fail |
| 4003 | Get size fail |
| 4004 | Read fail |
| 4005 | List file fail |
| 4006 | Delete file fail |
| 4007 | Get Disk info fail |
| 4008 | No space |
| 4009 | Time out |
| 4010 | File not found |
| 4011 | File too large |
| 4012 | File already exist |
| 4013 | Invalid parameter |
| 4014 | Driver error |
| 4015 | Create fail |
| 4016 | Access denied |
| 4017 | File too large |

# 4. Examples

## 4.1. Send HTTP GET Request

```
AT+QIFGCNT=0
OK

AT+QICSGP=1,"CMNET"              //Set APN
OK

AT+QIREGAPP                      //Optional
OK

AT+QIACT                         //Optional
OK

AT+QHTTPURL=79,30                //Set the URL

CONNECT
….
//for example, input 79 bytes:
http://api.efxnow.com/DEMOWebServices2.8/Service.asmx/Echo?Message=hello

OK

AT+QHTTPGET=60                   //Send HTTP GET Request
OK


AT+QHTTPREAD=30                  //Read the response of HTTP server.

CONNECT
….                               //Output the response data of HTTP server to UART.

//for example, UART outputs:
<?xml version="1.0" encoding="utf-8"?>
<string  xmlns="https://api.efxnow.com/webservices2.3">Message='hello'  ASCII:104  101
108 108 111 113 117 101 99 116 101 108 </string>
OK

AT+QIDEACT                       //Deactivate GPRS PDP connect.
DEACT OK
```

## 4.2. Send HTTP POST Request

```
AT+QIFGCNT=0
OK

AT+QICSGP=1,"CMNET"              //Set APN
OK

AT+QIREGAPP                      //Optional
OK

AT+QIACT                         //Optional
OK

AT+QHTTPURL=58,30                //Set the URL,

CONNECT
….
//for example, input 58 bytes:
http://api.efxnow.com/DEMOWebServices2.8/Service.asmx/Echo

OK

AT+QHTTPPOST=18,50,10
                                 //POST the data whose size is 18 Bytes and the maximum
                                   latency time for inputting is 50 s. It is recommended to set
                                   the latency time as long as enough to download all the data
                                   in the latency time.
CONNECT
                                 //This means it is ready to receive data from UART. And DCD
                                   has been set to low. Receive data from UART and not echo.
//for example, input 18 bytes: Message=helloworld
OK
//This means all data has been received over, and DCD is set to high.

AT+QHTTPREAD=30                  //Read the response of HTTP server.

CONNECT
….                               //Output the response data of HTTP server to UART
//for example, UART outputs:
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="https://api.efxnow.com/webservices2.3">Message='helloworld' ASCII:104 101
108 108 111 119 111 114 108 100 </string>
```

```
OK

AT+QIDEACT                    //Deactivate GPRS PDP connect.
DEACT OK
```

### 4.3. Download file form HTTP server

```
AT+QIFGCNT=0
OK

AT+QICSGP=1,"CMNET"           //Set APN
OK

AT+QIREGAPP                   //Optional
OK

AT+QIACT                      //Optional
OK

AT+QHTTPURL=29,30             //Set the URL,

CONNECT
….
//for example, input 29 bytes:
http://api.efxnow.com/1.TXT

OK

AT+QHTTPGET=60                //Send HTTP GET Request
OK

AT+QHTTPDL=“RAM:1.TXT”,1024   //Download the file to “RAM:1.TXT”, max size is
                                1024 bytes

+ QHTTPDL：100,100,0

OK
```