ESP8266 Wifi To Serial Module AT Commands Set

1. AT Commands List

1.1 Basic AT Commands List

- AT : Tests AT startup.
- AT+RST : Restarts a module.
- AT+GMR : Checks version information.
- AT+GSLP : Enters Deep-sleep mode.
- ATE : Configures echoing of AT commands.
- AT+RESTORE : Restores the factory default settings of the module.
- AT+UART_CUR : Current UART configuration.
- AT+UART_DEF : Default UART configuration, saved in flash.
- AT+SLEEP : Sets the sleep mode.
- AT+SYSRAM : Checks the remaining space of RAM.
- AT+SYSMSG : Set message format.
- AT+RFPOWER : Set RF TX Power.
- AT+SYSROLLBACK : Roll back to the previous firmware.
- AT+SYSTIMESTAMP: Set local time stamp.
- AT+SYSLOG : Enable or disable the AT error code prompt.
- AT+SYSLSPCFG : Config the light-sleep wakeup source.
- AT+SYSLSP : Enters light-sleep mode.

1.2 Wi-Fi AT Commands List

- AT+CWMODE : Sets the Wi-Fi mode (STA/AP/STA+AP).
- AT+CWJAP : Connects to an AP.
- AT+CWLAPOPT : Sets the configuration of command AT+CWLAP.
- AT+CWLAP : Lists available APs.
- AT+CWQAP : Disconnects from the AP.
- AT+CWSAP : Sets the configuration of the ESP SoftAP.
- AT+CWLIF : Gets the Station IP to which the ESP SoftAP is connected.
- AT+CWQIF : Disconnect Station from the ESP SoftAP.
- AT+CWDHCP : Enables/disables DHCP.
- AT+CWDHCPS : Sets the IP range of the ESP SoftAP DHCP server. Saves the setting in flash.
- AT+CWAUTOCONN : Connects to the AP automatically on power-up.
- AT+CIPSTAMAC : Sets the MAC address of ESP Station.
- AT+CIPAPMAC : Sets the MAC address of ESP SoftAP.
- AT+CIPSTA : Sets the IP address of ESP Station.
- AT+CIPAP : Sets the IP address of ESP SoftAP.
- AT+CWSTARTSMART : Starts SmartConfig.
- AT+CWSTOPSMART : Stops SmartConfig.
- AT+WPS : Enables the WPS function.

- AT+MDNS : Configurates the MDNS function
- AT+CWHOSTNAME : Configures the Name of ESP Station

1.3 TCP/IP-Related AT Commands List

- AT+CIPSTATUS : Gets the connection status.
- AT+CIPDOMAIN : Domain Name Resolution Function.
- AT+CIPSTART : Establishes TCP connection, UDP transmission or SSL connection.
- AT+CIPSTARTEX : Establishes TCP connection, UDP transmission or SSL connection with automatically assigned ID.
- AT+CIPSEND : Sends data.
- AT+CIPSENDEX : Sends data when length of data is \ or when \0 appears in the data.
- AT+CIPCLOSE : Closes TCP/UDP/SSL connection.
- AT+CIFSR : Gets the local IP address.
- AT+CIPMUX : Configures the multiple connections mode.
- AT+CIPSERVER : Deletes/Creates TCP or SSL server.
- AT+CIPSERVERMAXCONN : Set the Maximum Connections Allowed by Server.
- AT+CIPMODE : Configures the transmission mode.
- AT+SAVETRANSLINK : Saves the transparent transmission link in flash.
- AT+CIPSTO : Sets timeout when ESP runs as a TCP server.
- AT+CIPSNTPCFG : Configures the time domain and SNTP server.
- AT+CIPSNTPTIME : Queries the SNTP time.
- AT+CIUPDATE : Updates the software through Wi-Fi.
- AT+CIPDINFO : Shows remote IP and remote port with +IPD.
- AT+CIPSSLCCONF : Config SSL client.
- AT+CIPRECONNINTV: Set Wi-Fi transparent transmitting auto-connect interval.
- AT+CIPRECVMODE: Set Socket Receive Mode.
- AT+CIPRECVDATA: Get Socket Data in Passive Receive Mode.
- AT+CIPRECVLEN: Get Socket Data Length in Passive Receive Mode.
- AT+PING: Ping Packets
- AT+CIPDNS : Configures Domain Name System. The configuration will be saved in flash.

1.4 MQTT AT Commands List

- AT+MQTTUSERCFG : Set MQTT User Config
- AT+MQTTCLIENTID : Set MQTT Client ID
- AT+MQTTUSERNAME : Set MQTT Username
- AT+MQTTPASSWORD : Set MQTT Password
- AT+MQTTCONNCFG : Set configuration of MQTT Connection
- AT+MQTTCONN : Connect to MQTT Broker
- AT+MQTTPUB : Publish MQTT Data in string
- AT+MQTTPUBRAW : Publish MQTT message in binary
- AT+MQTTSUB : Subscribe to MQTT Topic
- AT+MQTTUNSUB : Unsubscribe from MQTT Topic
- AT+MQTTCLEAN : Close the MQTT Connection

2. Basic AT Commands

2.1 AT—Tests AT Startup

Execute Command:

```
AT
```

Response:

```
OK
```

2.2 AT+RST—Restarts the Module

Execute Command:

```
AT+RST
```

Response:

```
OK
```

2.3 AT+GMR—Checks Version Information

Execute Command:

```
AT+GMR
```

Response:

```
<AT version info>
<SDK version info>
<compile time>

OK
```

Parameters:

- **AT version info**: information about the AT version.

- **SDK version info**: information about the SDK version.
- **compile time**: the duration of time for compiling the BIN.

2.4 AT+GSLP—Enters Deep-sleep Mode

Set Command:

```
AT+GSLP=<time>
```

Response:

```
<time>

OK
```

Parameters:

- **time**: the duration of ESP's sleep. Unit: ms.
  ESP will wake up after Deep-sleep for as many milliseconds (ms) as \ indicates.

2.5 ATE—AT Commands Echoing

Execute Command:

```
ATE
```

Response:

```
OK
```

Parameters:

- **ATE0**: Switches echo off.
- **ATE1**: Switches echo on.

2.6 AT+RESTORE—Restores the Factory Default Settings

Execute Command:

```
AT+RESTORE
```

Response:

```
OK
```

Note:

- The execution of this command will reset all parameters saved in flash, and restore the factory default settings of the module.
- The chip will be restarted when this command is executed.

2.7 AT+UART_CUR—Current UART Configuration, Not Saved in Flash

Query Command:

```
AT+UART_CUR?
```

Response:

```
+UART_CUR:<baudrate>,<databits>,<stopbits>,<parity>,<flow control>

OK
```

Note:

- Command `AT+UART_CUR?` will return the actual value of UART configuration parameters, which may have allowable errors compared with the set value because of the clock division.

Set Command:

```
AT+UART_CUR=<baudrate>,<databits>,<stopbits>,<parity>,<flow control>
```

Response:

```
OK
```

Parameters:

- **baudrate**: UART baud rate
- **databits**: data bits
  - 5: 5-bit data
  - 6: 6-bit data
  - 7: 7-bit data
  - 8: 8-bit data
- **stopbits**: stop bits
  - 1: 1-bit stop bit
  - 2: 1.5-bit stop bit
  - 3: 2-bit stop bit
- **parity**: parity bit
  - 0: None
  - 1: Odd
  - 2: Even
- **flow control**: flow control

    - 0: flow control is not enabled
    - 1: enable RTS
    - 2: enable CTS
    - 3: enable both RTS and CTS

Notes:

- The configuration changes will NOT be saved in flash.
- The use of flow control requires the support of hardware:
    - IO15 is UART0 CTS
    - IO14 is UART0 RTS
- The range of baud rates supported: 80 ~ 5000000.

Example:

```
AT+UART_CUR=115200,8,1,0,3
```

2.8 AT+UART_DEF—Default UART Configuration, Saved in Flash

Query Command:

```
AT+UART_DEF?
```

Response:

```
+UART_DEF:<baudrate>,<databits>,<stopbits>,<parity>,<flow control>

OK
```

Set Command:

```
AT+UART_DEF=<baudrate>,<databits>,<stopbits>,<parity>,<flow control>
```

Response:

```
OK
```

Parameters:

- **baudrate**: UART baud rate
- **databits**: data bits
    - 5: 5-bit data
    - 6: 6-bit data
    - 7: 7-bit data
    - 8: 8-bit data
- **stopbits**: stop bits

    - 1: 1-bit stop bit
    - 2: 1.5-bit stop bit
    - 3: 2-bit stop bit
- **parity**: parity bit
    - 0: None
    - 1: Odd
    - 2: Even
- **flow control**: flow control
    - 0: flow control is not enabled
    - 1: enable RTS
    - 2: enable CTS
    - 3: enable both RTS and CTS

Notes:

- The configuration changes will be saved in the NVS area, and will still be valid when the chip is powered on again.
- The use of flow control requires the support of hardware:
    - IO15 is UART0 CTS
    - IO14 is UART0 RTS
- The range of baud rates supported: 80 ~ 5000000.

Example:

```
AT+UART_DEF=115200,8,1,0,3
```

2.9 AT+SLEEP—Sets the Sleep Mode

Set Command:

```
AT+SLEEP=<sleep mode>
```

Response:

```
OK
```

Parameters:

- **sleep mode**:
    - 0: disable the sleep mode.
    - 1: Modem-sleep mode.

Example:

```
AT+SLEEP=0
```

2.10 AT+SYSRAM—Checks the Remaining Space of RAM

Query Command:

```
AT+SYSRAM?
```

Response:

```
+SYSRAM:<remaining RAM size>
OK
```

Parameters:

- **remaining RAM size**: remaining space of RAM, unit: byte

Example:

```
AT+SYSRAM?
+SYSRAM:148408
OK
```

2.11 AT+SYSMSG—Control to use new or old information

Query Command:

```
AT+SYSMSG?
Function:
Query the current system message state.
```

Response:

```
+SYSMSG:<state>
OK
```

Set Command:

```
AT+SYSMSG=<state>
Function:
Control to use new or old information.
```

Response:

```
OK
```

Parameters:

- **state**:
  - Bit0: Quit transparent transmission 0: Quit transparent transmission no information. 1: Quit transparent transmission will supply information.
  - Bit1: Connection info 0: Use old connection info. 1: Use new connection info.

Notes:

- The configuration changes will be saved in the NVS area.
- If set Bit0 to 1 will supply information "+QUITT" when quit transparent transmission.
- If set Bit1 to 1 will impact the infomation of command `AT+CIPSTART` and `AT+CIPSERVER`,
  - It will supply "+LINK_CONN : status_type,link_id,ip_type,terminal_type,remote_ip,remote_port,local_port" instead of "XX,CONNECT". Example:

// Use new connection info and quit transparent transmission no information AT+SYSMSG=2

2.14 AT+RFPOWER-Set RF TX Power

Query Command:

```
AT+RFPOWER?
Function: to query the RF TX Power.
```

Response:

```
+RFPOWER:<wifi_power>,<ble_adv_power>,<ble_scan_power>,<ble_conn_power>
OK
```

Set Command:

```
AT+RFPOWER=<wifi_power>[,<ble_adv_power>,<ble_scan_power>,<ble_conn_power>]
```

Response:

```
OK
```

Parameters:

- **wifi_power**: range [40, 82], the unit is 0.25dBm, for example, if the value is 78, then RF max power is 78*0.25 dBm=19.5dBm
- **ble_adv_power**: RF TX Power of BLE advertising, range: [0, 7]
  - 0:7dBm
  - 1:4dBm
  - 2:1dBm

    - 3:-2 dBm
    - 4:-5 dBm
    - 5:-8 dBm
    - 6:-11 dBm
    - 7:-14 dBm

- **ble_scan_power**: RF TX Power of BLE scanning, range: [0, 7], the same as **ble_adv_power**
- **ble_conn_power**: RF TX Power of BLE connecting, range: [0, 7], the same as **ble_adv_power**

Notes: Since the RF TX power is actually divided into several levels, and each level has its own value range, so the `wifi_power` value queried by the `esp_wifi_get_max_tx_power` may differ from the value set by `esp_wifi_set_max_tx_power`. And the query value will not be larger than the set one.

2.15 AT+SYSROLLBACK-Roll back to the previous firmware

Execute Command:

```
AT+SYSROLLBACK
```

Response:

```
OK
```

Note:

- This command will not upgrade via OTA, only roll back to the firmware which is in the other ota partition.

2.16 AT+SYSTIMESTAMP—Set local time stamp.

Query Command:

```
AT+SYSTIMESTAMP?
Function: to query the time stamp.
```

Response:

```
+SYSTIMESTAMP:<Unix_timestamp>
OK
```

Set Command:

```
AT+SYSTIMESTAMP=<Unix_timestamp>
Function: to set local time stamp. It will be the same as SNTP time
when the SNTP time updated.
```

Response:

```
OK
```

Parameters:

- **Unix_timestamp**: Unix timestamp, the unit is seconds.

Example:

```
AT+SYSTIMESTAMP=1565853509    //2019-08-15 15:18:29
```

2.17 AT+SYSLOG : Enable or disable the AT error code prompt.

Query Command:

```
AT+SYSLOG?
Function: to query the AT error code prompt for whether it is enabled or disabled.
```

Response:

```
+SYSLOG:<status>

OK
```

Set Command:

```
AT+SYSLOG=<status>
Function: Enable or disable the AT error code prompt.
```

Response:

```
OK
```

Parameters:
- **status**: : enable or disable - 0: disable - 1: enable

Example:
If enable AT error code prompt:

```
AT+SYSLOG=1

OK
AT+FAKE
ERR CODE:0x01090000

ERROR
```

If disable AT error code prompt:

```
AT+SYSLOG=0

OK
AT+FAKE
//No `ERR CODE:0x01090000`

ERROR
```

3 Wi-Fi AT Commands

3.1 AT+CWMODE—Sets the Wi-Fi Mode (Station/SoftAP/Station+SoftAP)

Query Command:

```
AT+CWMODE?
Function: to query the Wi-Fi mode of ESP.
```

Response:

```
+CWMODE:<mode>
OK
```

Set Command:

```
AT+CWMODE=<mode>
Function: to set the Wi-Fi mode of ESP.
```

Response:

```
OK
```

Parameters:

- **mode**:
  - 0: Null mode, WiFi RF will be disabled
  - 1: Station mode
  - 2: SoftAP mode
  - 3: SoftAP+Station mode

Note:

- The configuration changes will be saved in the NVS area.

Example:

```
AT+CWMODE=3
```

3.2 AT+CWJAP—Connects to an AP

Query Command:

```
AT+CWJAP?
Function: to query the AP to which the ESP Station is already connected.
```

Response:

```
+CWJAP:<ssid>,<bssid>,<channel>,<rssi>,<pci_en>,<reconn>,<lisnten_interval>
OK
```

Parameters:

- **ssid**: a string parameter showing the SSID of the AP.
- **bssid**: the AP's MAC address.
- **channel**: channel
- **rssi**: signal strength
- **pci_en**: PCI Authentication, which will disable connect OPEN and WEP AP.
- **reconn**: Wi-Fi reconnection, when beacon timeout, ESP will reconnect automatically.
- **listen_interval**: the interval of listening to the AP's beacon,the range is (0,100]

Set Command:

```
AT+CWJAP=<ssid>,<pwd>[,<bssid>][,<pci_en>][,<reconn>][,<listen_interval>]
Function: to set the AP to which the ESP Station needs to be connected.
```

Response:

```
OK
```

or +CWJAP: ERROR Parameters:

- **ssid**: the SSID of the target AP.
    - Escape character syntax is needed if SSID or password contains any special characters, such as , or " or \.
- **pwd**: password, MAX: 64-byte ASCII.
- **bssid**: the target AP's MAC address, used when multiple APs have the same SSID.
- **pci_en**: enable PCI Authentication, which will disable connect OPEN and WEP AP.
- **reconn**: enable Wi-Fi reconnection, when beacon timeout, ESP will reconnect automatically.
- **listen_interval**: the interval of listening to the AP's beacon,the range is (0,100],
- **error code**: (for reference only)
    - 1: connection timeout.

- 2: wrong password.
- 3: cannot find the target AP.
- 4: connection failed.
- others: unknown error occurred.

Note:

- The configuration changes will be saved in the NVS area.
- This command requires Station mode to be active.

Examples:

```
AT+CWJAP="abc","0123456789"
- For example, if the target AP's SSID is "ab\,c" and the password
is "0123456789"\", the command is as follows:
AT+CWJAP="ab\\\,c","0123456789\"\\"
- If multiple APs have the same SSID as "abc", the target AP can be
found by BSSID:
AT+CWJAP="abc","0123456789","ca:d7:19:d8:a6:44"
```

3.3 AT+CWLAPOPT—Sets the Configuration for the Command AT+CWLAP

Set Command:

```
AT+CWLAPOPT=<sort_enable>,<mask>
```

Response:

```
OK
```

Parameters:

- **sort_enable**: determines whether the result of command AT+CWLAP will be listed according to RSSI:
  - 0: the result is not ordered according to RSSI.
  - 1: the result is ordered according to RSSI.
- **mask**: determines the parameters shown in the result of `AT+CWLAP` ;
  - 0 means not showing the parameter corresponding to the bit, and 1 means showing it.
  - bit 0: determines whether \ will be shown in the result of `AT+CWLAP` .
  - bit 1: determines whether \ will be shown in the result of `AT+CWLAP` .
  - bit 2: determines whether \ will be shown in the result of `AT+CWLAP` .
  - bit 3: determines whether \ will be shown in the result of `AT+CWLAP` .
  - bit 4: determines whether \ will be shown in the result of `AT+CWLAP` .

Example:

```
AT+CWLAPOPT=1,31
- The first parameter is 1, meaning that the result of the command AT+CWLAP will be ordered according to RSSI;
- The second parameter is 31, namely 0x1F, meaning that the corresponding bits of <mask> are set to 1. All parameters will be shown in the result of AT+CWLAP.
```

3.4 AT+CWLAP—Lists the Available APs

Set Command:

```
AT+CWLAP=[<ssid>,<mac>,<channel>,<scan_type>,<scan_time_min>,<scan_time_max>]
Function: to query the APs with specific SSID and MAC on a specific channel.
```

Execute Command:

```
AT+CWLAP
Function: to list all available APs.
```

Response:

```
+CWLAP:<ecn>,<ssid>,<rssi>,<mac>,<channel>
OK
```

Parameters:

- **ecn**: encryption method.
  - 0: OPEN
  - 1: WEP
  - 2: WPA_PSK
  - 3: WPA2_PSK
  - 4: WPA_WPA2_PSK
  - 5: WPA2_Enterprise (AT can NOT connect to WPA2_Enterprise AP for now.)
- **ssid**: string parameter, SSID of the AP.
- **rssi**: signal strength.
- **mac**: string parameter, MAC address of the AP.
- **scan_type**: Wi-Fi scan type, active or passive.
  - 0: active scan
  - 1: passive scan
- **scan_time_min**: minimum active scan time per channel, units: millisecond, range [0,1500], if the scan type is passive, this param is invalid.
- **scan_time_max**: maximum active scan time per channel, units: millisecond, range [0,1500]. if this param is zero, the firmware will use the default time, active scan type is 120ms , passive scan type is 360ms.

Examples:

```
AT+CWLAP="Wi-Fi","ca:d7:19:d8:a6:44",6,0,400,1000
Or search for APs with a designated SSID:
AT+CWLAP="Wi-Fi"
```

3.5 AT+CWQAP—Disconnects from the AP

Execute Command:

```
AT+CWQAP
```

Response:

```
OK
```

3.6 AT+CWSAP—Configuration of the ESP SoftAP

Query Command:

```
AT+CWSAP?
Function: to obtain the configuration parameters of the ESP SoftAP.
```

Response:

```
+CWSAP:<ssid>,<pwd>,<channel>,<ecn>,<max conn>,<ssid hidden>
OK
```

Set Command:

```
AT+CWSAP=<ssid>,<pwd>,<chl>,<ecn>[,<max conn>][,<ssid hidden>]
Function: to set the configuration of the ESP SoftAP.
```

Response:

```
OK
```

Parameters:

- **ssid**: string parameter, SSID of AP.
- **pwd**: string parameter, length of password: 8 ~ 64 bytes ASCII.
- **channel**: channel ID.
- **ecn**: encryption method; WEP is not supported.
    - 0: OPEN
    - 2: WPA_PSK
    - 3: WPA2_PSK

    - 4: WPA_WPA2_PSK
- **max conn**(optional parameter): maximum number of Stations to which ESP SoftAP can be connected; within the range of [1, 10].
- **ssid hidden**(optional parameter):
    - 0: SSID is broadcast. (the default setting)
    - 1: SSID is not broadcast.

Note:

- This command is only available when SoftAP is active.
- The configuration changes will be saved in the NVS area.

Example:

```
AT+CWSAP="ESP","1234567890",5,3
```

3.7 AT+CWLIF—IP of Stations to Which the ESP SoftAP is Connected

Execute Command:

```
AT+CWLIF
```

Response:

```
<ip addr>,<mac>
OK
```

Parameters:

- **ip addr**: IP address of Stations to which ESP SoftAP is connected.
- **mac**: MAC address of Stations to which ESP SoftAP is connected.

Note:

- This command cannot get a static IP. It only works when both DHCPs of the ESP SoftAP, and of the Station to which ESP is connected, are enabled.

3.8 AT+CWQIF—Disconnect Station from the ESP SoftAP

Execute Command:

```
AT+CWQIF
Function: Disconnect all stations that connected to the ESP SoftAP.
```

Response:

```
OK
```

Set Command:

```
AT+CWQIF=<mac>
Function: Disconnect the station whose mac is "<mac>" from the ESP SoftAP.
```

Response:

```
OK
```

Parameters:

- **mac**: MAC address of the station to disconnect to.

3.9 AT+CWDHCP—Enables/Disables DHCP

Query Command:

```
AT+CWDHCP?
```

Response: state

Set Command:

```
AT+CWDHCP=<operate>,<mode>
Function: to enable/disable DHCP.
```

Response:

```
OK
```

Parameters:

- **operate**:
  - 0: disable
  - 1: enable
- **mode**:
  - Bit0: Station DHCP
  - Bit1: SoftAP DHCP
- **state**:DHCP disabled or enabled now? Bit0: 0: Station DHCP is disabled. 1: Station DHCP is enabled. Bit1: 0: SoftAP DHCP is disabled. 1: SoftAP DHCP is enabled. ***Notes:***
- The configuration changes will be stored in the NVS area.
- This set command interacts with static-IP-related AT commands(AT+CIPSTA-related and AT+CIPAP-

related commands):

- If DHCP is enabled, static IP will be disabled;
- If static IP is enabled, DHCP will be disabled;
- Whether it is DHCP or static IP that is enabled depends on the last configuration.

Examples:

```
AT+CWDHCP=1,1    //Enable Station DHCP. If the last DHCP mode is 2, then the current
                 //DHCP mode will be 3.
AT+CWDHCP=0,2    //Disable SoftAP DHCP. If the last DHCP mode is 3, then the current
                 //DHCP mode will be 1.
```

3.10 AT+CWDHCPS—Sets the IP Address Allocated by ESP SoftAP DHCP (The configuration is saved in Flash.)

Query Command:

```
AT+CWDHCPS?
```

Response:

```
+CWDHCPS=<lease time>,<start IP>,<end IP>
OK
```

Set Command:

```
AT+CWDHCPS=<enable>,<lease time>,<start IP>,<end IP>
Function: sets the IP address range of the ESP SoftAP DHCP server.
```

Response:

```
OK
```

Parameters:

- **enable**:
  - 0: Disable the settings and use the default IP range.
  - 1: Enable setting the IP range, and the parameters below have to be set.
- **lease time**: lease time, unit: minute, range [1, 2880].
- **start IP**: start IP of the IP range that can be obtained from ESP SoftAP DHCP server.
- **end IP**: end IP of the IP range that can be obtained from ESP SoftAP DHCP server.

Notes:

- The configuration changes will be saved in the NVS area.
- This AT command is enabled when ESP8266 runs as SoftAP, and when DHCP is enabled.

- The IP address should be in the same network segment as the IP address of ESP SoftAP.

Examples:

```
AT+CWDHCPS=1,3,"192.168.4.10","192.168.4.15"
or
AT+CWDHCPS=0 //Disable the settings and use the default IP range.
```

3.11 AT+CWAUTOCONN—Auto-Connects to the AP or Not

Set Command:

```
AT+CWAUTOCONN=<enable>
```

Response:

```
OK
```

Parameters:

- **enable**:
  - 0: does NOT auto-connect to AP on power-up.
  - 1: connects to AP automatically on power-up.

Note:

- The configuration changes will be saved in the NVS area.
- The ESP Station connects to the AP automatically on power-up by default.

Example:

```
AT+CWAUTOCONN=1
```

3.12 AT+CIPSTAMAC—Sets the MAC Address of the ESP Station

Query Command:

```
AT+CIPSTAMAC?
Function: to obtain the MAC address of the ESP Station.
```

Response:

```
+CIPSTAMAC:<mac>
OK
```

Set Command:

```
AT+CIPSTAMAC=<mac>
Function: to set the MAC address of the ESP Station.
```

Response:

```
OK
```

Parameters:

- **mac**: string parameter, MAC address of the ESP8266 Station.

Notes:

- The configuration changes will be saved in the NVS area.
- The MAC address of ESP SoftAP is different from that of the ESP Station. Please make sure that you do not set the same MAC address for both of them.
- Bit 0 of the ESP MAC address CANNOT be 1.
    - For example, a MAC address can be "1a:..." but not "15:...".
- FF:FF:FF:FF:FF:FF and 00:00:00:00:00:00 are invalid MAC and cannot be set.

Example:

```
AT+CIPSTAMAC="1a:fe:35:98:d3:7b"
```

3.13 AT+CIPAPMAC—Sets the MAC Address of the ESP SoftAP

Query Command:

```
AT+CIPAPMAC?
Function: to obtain the MAC address of the ESP SoftAP.
```

Response:

```
+CIPAPMAC:<mac>
OK
```

Set Command:

```
AT+CIPAPMAC=<mac>
Function: to set the MAC address of the ESP SoftAP.
```

Response:

```
OK
```

Parameters:

- **mac**: string parameter, MAC address of the ESP8266 SoftAP.

Notes:

- The configuration changes will be saved in the NVS area.
- The MAC address of ESP SoftAP is different from that of the ESP Station. Please make sure that you do not set the same MAC address for both of them.
- Bit 0 of the ESP MAC address CANNOT be 1.
    - For example, a MAC address can be "18:..." but not "15:...".
- FF:FF:FF:FF:FF:FF and 00:00:00:00:00:00 are invalid MAC and cannot be set.

Example:

```
AT+CIPAPMAC="18:fe:35:98:d3:7b"
```

3.14 AT+CIPSTA—Sets the IP Address of the ESP Station

Query Command:

```
AT+CIPSTA?
Function: to obtain the IP address of the ESP Station.
Notice: Only when the ESP Station is connected to an AP can its IP address be queried.
```

Response:

```
+CIPSTA:<ip>
OK
```

Set Command:

```
AT+CIPSTA=<ip>[,<gateway>,<netmask>]
Function: to set the IP address of the ESP Station.
```

Response:

```
OK
```

Parameters:

- **ip**: string parameter, the IP address of the ESP Station.
- **gateway**: gateway.
- **netmask**: netmask.

Notes:

- The configuration changes will be saved in the NVS area.

- The set command interacts with DHCP-related AT commands (AT+CWDHCP-related commands):
  - If static IP is enabled, DHCP will be disabled;
  - If DHCP is enabled, static IP will be disabled;
  - Whether it is DHCP or static IP that is enabled depends on the last configuration.

Example:

```
AT+CIPSTA="192.168.6.100","192.168.6.1","255.255.255.0"
```

3.15 AT+CIPAP—Sets the IP Address of the ESP SoftAP

Query Command:

```
AT+CIPAP?
Function: to obtain the IP address of the ESP SoftAP.
```

Response:

```
+CIPAP:<ip>,<gateway>,<netmask>
OK
```

Set Command:

```
AT+CIPAP=<ip>[,<gateway>,<netmask>]
Function: to set the IP address of the ESP SoftAP.
```

Response:

```
OK
```

Parameters:

- **ip**: string parameter, the IP address of the ESP SoftAP.
- **gateway**: gateway.
- **netmask**: netmask.

Notes:

- The configuration changes will be saved in the NVS area.
- The set command interacts with DHCP-related AT commands (AT+CWDHCP-related commands):
  - If static IP is enabled, DHCP will be disabled;
  - If DHCP is enabled, static IP will be disabled;
  - Whether it is DHCP or static IP that is enabled depends on the last configuration.

Example:

```
AT+CIPAP="192.168.5.1","192.168.5.1","255.255.255.0"
```

3.16 AT+CWSTARTSMART—Starts SmartConfig

Execute Command:

```
AT+CWSTARTSMART
Function: to start SmartConfig. (The type of SmartConfig is ESP-TOUCH + AirKiss.
```

Set Command:

```
AT+CWSTARTSMART=<type>
Function: to start SmartConfig of a designated type.
```

Response:

```
OK
```

Parameters:

- **type**:
  - 1: ESP-TOUCH
  - 2: AirKiss
  - 3: ESP-TOUCH+AirKiss

Notes:

- For details on SmartConfig please see ESP-TOUCH User Guide.
- SmartConfig is only available in the ESP Station mode.
- The message `Smart get Wi-Fi info` means that SmartConfig has successfully acquired the AP information. ESP will try to connect to the target AP.
- Message `Smartconfig connected Wi-Fi` is printed if the connection is successful.
- Use command `AT+CWSTOPSMART` to stop SmartConfig before running other commands. Please make sure that you do not execute other commands during SmartConfig.

Example:

```
AT+CWMODE=1
AT+CWSTARTSMART
```

3.17 AT+CWSTOPSMART—Stops SmartConfig

Execute Command:

```
AT+CWSTOPSMART
```

Response:

```
OK
```

Note:

- Irrespective of whether SmartConfig succeeds or not, before executing any other AT commands, please always call `AT+CWSTOPSMART` to release the internal memory taken up by SmartConfig.

Example:

```
AT+CWMODE=1
AT+CWSTARTSMART
AT+CWSTOPSMART
```

3.18 AT+WPS—Enables the WPS Function

Set Command:

```
AT+WPS=<enable>
```

Response:

```
OK
```

Parameters:

- **enable**:
  - 1: enable WPS/Wi-Fi Protected Setup (implemented by PBC/Push Button Configuration).
  - 0: disable WPS (implemented by PBC).

Notes:

- WPS must be used when the ESP Station is enabled.
- WPS does not support WEP/Wired-Equivalent Privacy encryption.

Example:

```
AT+CWMODE=1
AT+WPS=1
```

3.19 AT+MDNS—Configurates the MDNS Function

Set Command:

```
AT+MDNS=<enable>[,<hostname>,<service_name>,<port>]
```

Response:

```
OK
```

Parameters:

- **enable**:
  - 1: enables the MDNS function; the following three parameters need to be set.
  - 0: disables the MDNS function; the following three parameters need not to be set.
- **hostname**: MDNS host name
- **service_name**: MDNS service name
- **port**: MDNS port

Example:

```
AT+MDNS=1,"espressif","_iot",8080
AT+MDNS=0
```

3.20 AT+CWJEAP—Connects to an WPA2 Enterprise AP.

Query Command:

```
AT+CWJEAP?
Function: to query the Enterprise AP to which the ESP Station is already connected.
```

Response:

```
+CWJEAP:<ssid>,<method>,<identity>,<username>,<password>,<security>
OK
```

Set Command:

```
AT+CWJEAP=<ssid>,<method>,<identity>,<username>,<password>,<security>
Function: to set the Enterprise AP to which the ESP Station needs to be connected.
```

Response:

```
OK
```

or +CWJEAP:Timeout ERROR Parameters:

- **ssid**: the SSID of the Enterprise AP.
  - Escape character syntax is needed if SSID or password contains any special characters, such as , or " or \.
- **method**: wpa2 enterprise authentication method.
  - 0: EAP-TLS.

    - 1: EAP-PEAP.
    - 2: EAP-TTLS.
- **identity**: identity for phase 1, string limited to 1~32.
- **username**: username for phase 2, must set for EAP-PEAP and EAP-TTLS mode, nor care for EAP-TLS, string limited to 1~32.
- **password**: password for phase 2, must set for EAP-PEAP and EAP-TTLS mode, nor care for EAP-TLS, string limited to 1~32.
- **security**:
    - Bit0: Client certificate
    - Bit1: Server certificate

Example:

```
 1. Connect to EAP-TLS mode enterprise AP, set identity, verify server certificate
and load client certificate
AT+CWJEAP="dlink11111",0,"example@espressif.com",,,3
2. Connect to EAP-PEAP mode enterprise AP, set identity, username and password,
not verify server certificate and not load client certificate
AT+CWJEAP="dlink11111",1,"example@espressif.com","espressif","test11",0
```

Note:

- The configuration changes will be saved in the NVS area.
- This command requires Station mode to be active.
- TLS mode will use client certificate, make sure enabled.

3.21 AT+CWHOSTNAME : Configures the Name of ESP Station

Query Command:

```
AT+CWHOSTNAME?
Function: Checks the host name of ESP Station.
```

Response:

```
+CWHOSTNAME:<hostname>

OK
```

Set Command:

```
AT+CWHOSTNAME=<hostname>
Function: Sets the host name of ESP Station.
```

Response:

```
OK
```

If the Station mode is not enabled, the command will return:

```
ERROR
```

Parameters:

- **hostname**: the host name of the ESP Station, the maximum length is 32 bytes.

Note:

- The configuration changes are not saved in the flash.

Example:

```
AT+CWMODE=3
AT+CWHOSTNAME="my_test"
```

4. TCP/IP-Related AT Commands

4.1 AT+CIPSTATUS—Gets the Connection Status

Execute Command:

```
AT+CIPSTATUS
```

Response:

```
STATUS:<stat>
+CIPSTATUS:<link ID>,<type>,<remote IP>,<remote port>,<local port>,<tetype>
```

Parameters:

- **stat**: status of the ESP Station interface.
  - 0: The ESP station is inactive.
  - 1: The ESP station is idle.
  - 2: The ESP Station is connected to an AP and its IP is obtained.
  - 3: The ESP Station has created a TCP or UDP transmission.
  - 4: The TCP or UDP transmission of ESP Station is disconnected.
  - 5: The ESP Station does NOT connect to an AP.
- **link ID**: ID of the connection (0~4), used for multiple connections.
- **type**: string parameter, "TCP" or "UDP".
- **remote IP**: string parameter indicating the remote IP address.
- **remote port**: the remote port number.
- **local port**: ESP local port number.

- **tetype**:
  - 0: ESP runs as a client.
  - 1: ESP runs as a server.

4.2 AT+CIPDOMAIN—Domain Name Resolution Function

Execute Command:

```
AT+CIPDOMAIN=<domain name>
```

Response:

```
+CIPDOMAIN:<IP address>
```

Parameter:

- **domain name**: the domain name.

Example:

```
 AT+CWMODE=1                        // set Station mode
AT+CWJAP="SSID","password"         // access to the internet
AT+CIPDOMAIN="iot.espressif.cn"    // Domain Name Resolution function
```

4.3 AT+CIPSTART—Establishes TCP Connection, UDP Transmission or SSL Connection

4.3.1 Establish TCP Connection

Set Command:

```
 Single TCP connection (AT+CIPMUX=0):
AT+CIPSTART=<type>,<remote IP>,<remote port>[,<TCP keep alive>][,<local IP>]
Multiple TCP Connections (AT+CIPMUX=1):
AT+CIPSTART=<link ID>,<type>,<remote IP>,<remote port>[,<TCP keep alive>][,<local IP>]
```

Response:

```
OK
```

Or if the TCP connection is already established, the response is: ALREADY CONNECTTED ERROR Parameters:

- **link ID**: ID of network connection (0~4), used for multiple connections.
- **type**: string parameter indicating the connection type: "TCP", "UDP" or "SSL".
- **remote IP**: string parameter indicating the remote IP address.
- **remote port**: the remote port number.

- **TCP keep alive**(optional parameter): detection time interval when TCP is kept alive; this function is

disabled by default.

  - 0: disable TCP keep-alive.
  - 1 ~ 7200: detection time interval; unit: second (s).

- **local IP**(optional parameter): select which IP want to use, this is useful when using both ethernet and WiFi; this parameter is disabled by default. If you want to use this parameter, must be specified firstly, null also is valid.

Examples:

```
AT+CIPSTART="TCP","iot.espressif.cn",8000
AT+CIPSTART="TCP","192.168.101.110",1000
AT+CIPSTART="TCP","192.168.101.110",1000,,"192.168.101.100"
```

4.3.2 Establish UDP Transmission

Set Command:

```
Single connection (AT+CIPMUX=0):
AT+CIPSTART=<type>,<remote IP>,<remote port>[,(<UDP local port>),
            (<UDP mode>)][,<local IP>]

Multiple connections (AT+CIPMUX=1):
AT+CIPSTART=<link ID>,<type>,<remote IP>,<remote port>[,(<UDP local port>),
            (<UDP mode>)][,<local IP>]
```

Response:

```
OK
```

Or if the UDP transmission is already established, the response is: ALREADY CONNECTTED ERROR
Parameters:

- **link ID**: ID of network connection (0~4), used for multiple connections.
- **type**: string parameter indicating the connection type: "TCP", "UDP" or "SSL".
- **remote IP**: string parameter indicating the remote IP address.
- **remote port**: remote port number.
- **UDP local port**(optional parameter): UDP port of ESP.
- **UDP mode**(optional parameter): In the UDP transparent transmission, the value of this parameter has to be 0.
  - 0: the destination peer entity of UDP will not change; this is the default setting.
  - 1: the destination peer entity of UDP can change once.
  - 2: the destination peer entity of UDP is allowed to change.
- **local IP**(optional parameter): select which IP want to use, this is useful when using both ethernet and WiFi; this parameter is disabled by default. If you want to use this parameter, and must be specified

firstly, null also is valid.

Notice:

- To use parameter \, parameter \ must be set first.

Example:

```
AT+CIPSTART="UDP","192.168.101.110",1000,1002,2
AT+CIPSTART="UDP","192.168.101.110",1000,,,"192.168.101.100"
```

4.3.3 Establish SSL Connection

Set Command:

```
AT+CIPSTART=[<link ID>,]<type>,<remote IP>,<remote port>[,<TCP keep alive>]
           [,<local IP>]
```

Response:

```
OK
```

Or if the TCP connection is already established, the response is: ALREADY CONNECTTED ERROR Parameters:

- **link ID**: ID of network connection (0~4), used for multiple connections.
- **type**: string parameter indicating the connection type: "TCP", "UDP" or "SSL".
- **remote IP**: string parameter indicating the remote IP address.
- **remote port**: the remote port number.
- **TCP keep alive**(optional parameter): detection time interval when TCP is kept alive; this function is disabled by default.
  - 0: disable the TCP keep-alive function.
  - 1 ~ 7200: detection time interval, unit: second (s).
- **local IP**(optional parameter): select which IP want to use, this is useful when using both ethernet and WiFi; this parameter is disabled by default. If you want to use this parameter, must be specified firstly, null also is valid.

Notes:

- ESP can only set one SSL connection at most.
- SSL connection does not support UART-WiFi passthrough mode (transparent transmission).
- SSL connection needs a large amount of memory; otherwise, it may cause system reboot.

Example:

```
AT+CIPSTART="SSL","iot.espressif.cn",8443
AT+CIPSTART="SSL","192.168.101.110",1000,,"192.168.101.100"
```

4.4 AT+CIPSTARTEX—Establishes TCP connection, UDP transmission or SSL connection with automatically assigned ID

This command is similar to AT+CIPSTART, but you need not to assign an ID by yourself when it is the multiple connections mode (AT+CIPMUX=1), the system will assign an ID to the new connection automatically.

4.5 AT+CIPSEND—Sends Data

Set Command:

```
 Single connection: (+CIPMUX=0)
AT+CIPSEND=<length>
Multiple connections: (+CIPMUX=1)
AT+CIPSEND=<link ID>,<length>
Remote IP and ports can be set in UDP transmission:
AT+CIPSEND=[<link ID>,]<length>[,<remote IP>,<remote port>]
Function: to configure the data length in normal transmission mode.
```

Response:

```
OK
>
```

Begin receiving serial data. When the requirement of data length is met, the transmission of data starts. If the connection cannot be established or gets disrupted during data transmission, the system returns:

```
ERROR
```

If data is transmitted successfully, the system returns:

```
SEND OK
```

Execute Command:

```
AT+CIPSEND
Function: to start sending data in transparent transmission mode.
```

Response:

```
OK
>
```

Enter transparent transmission, with a 20-ms interval between each packet, and a maximum of 2048 bytes per packet. When a single packet containing +++ is received, ESP returns to normal command mode. Please wait for at least one second before sending the next AT command. This command can only be used in transparent transmission mode which requires single connection. For UDP transparent transmission, the value of has to be

0 when using AT+CIPSTART.

Or

```
ERROR
```

Parameters:

- **link ID**: ID of the connection (0~4), for multiple connections.
- **length**: data length, MAX: 2048 bytes.
- **remote IP**(optional parameter): remote IP can be set in UDP transmission.
- **remote port**(optional parameter): remote port can be set in UDP transmission.

4.6 AT+CIPSENDEX—Sends Data

Set Command:

```
 Single connection: (+CIPMUX=0)
AT+CIPSENDEX=<length>
Multiple connections: (+CIPMUX=1)
AT+CIPSENDEX=<link ID>,<length>
Remote IP and ports can be set in UDP transmission:
AT+CIPSENDEX=[<link ID>,]<length>[,<remote IP>,<remote port>]
Function: to configure the data length in normal transmission mode.
```

Response:

```
OK
>
```

Send data of designated length. Wrap return > after the set command. Begin receiving serial data. When the requirement of data length, determined by , is met, or when \0 appears in the data, the transmission starts. If connection cannot be established or gets disconnected during transmission, the system returns:

```
ERROR
```

If data are successfully transmitted, the system returns: SEND OK Parameters:

- **link ID**: ID of the connection (0~4), for multiple connections.
- **length**: data length, MAX: 2048 bytes.
    - When the requirement of data length, determined by \, is met, or when string \0 appears, the transmission of data starts. Go back to the normal command mode and wait for the next AT command.
    - When sending \0, please send it as \\0.

4.7 AT+CIPCLOSE—Closes TCP/UDP/SSL Connection

Set Command (for multiple connections):

```
AT+CIPCLOSE=<link ID>
Function: to close TCP/UDP connection.
```

Parameters:

- **link ID**: ID number of connections to be closed; when ID=5, all connections will be closed.

Execute Command (for single connection):

```
AT+CIPCLOSE
```

Response:

```
OK
```

4.8 AT+CIFSR—Gets the Local IP Address

Execute Command:

```
AT+CIFSR
```

Response:

```
+CIFSR:<SoftAP IP address>
+CIFSR:<Station IP address>
OK
```

Parameters:

- **IP address**:
  - IP address of the ESP SoftAP;
  - IP address of the ESP Station.

Notes:

- Only when the ESP Station is connected to an AP can the Station IP can be queried.

4.9 AT+CIPMUX—Enables/Disables Multiple Connections

Query Command:

```
AT+CIPMUX?
Function: to query the connection type.
```

Response:

```
+CIPMUX:<mode>
OK
```

Set Command:

```
AT+CIPMUX=<mode>
Function: to set the connection type.
```

Response:

```
OK
```

Parameters:

- **mode**:
  - 0: single connection
  - 1: multiple connections

Notes:

- The default mode is single connection mode.
- Multiple connections can only be set when transparent transmission is disabled (`AT+CIPMODE=0`).
- This mode can only be changed after all connections are disconnected.
- If the TCP server is running, it must be deleted (`AT+CIPSERVER=0`) before the single connection mode is activated.

Example:

```
AT+CIPMUX=1
```

4.10 AT+CIPSERVER—Deletes/Creates TCP or SSL Server

Set Command:

```
AT+CIPSERVER=<mode>[,<port>]
```

Response:

```
OK
```

Parameters:

- **mode**:
  - 0: delete server.
  - 1: create server.
- **port**: port number; 333 by default.

Notes:

- A TCP server can only be created when multiple connections are activated (`AT+CIPMUX=1`).
- A server monitor will automatically be created when the TCP server is created. And only one server is allowed.
- When a client is connected to the server, it will take up one connection and be assigned an ID.

Example:

```
// To create a TCP server
AT+CIPMUX=1
AT+CIPSERVER=1,80
```

4.11 AT+CIPSERVERMAXCONN—Set the Maximum Connections Allowed by Server

Query Command:

```
AT+CIPSERVERMAXCONN?
Function: obtain the maximum number of clients allowed to connect to the TCP
          or SSL server.
```

Response:

```
+CIPSERVERMAXCONN:<num>
OK
```

Set Command:

```
AT+CIPSERVERMAXCONN=<num>
Function: set the maximum number of clients allowed to connect to the TCP
          or SSL server.
```

Response:

```
OK
```

Parameters:

- **num**: the maximum number of clients allowed to connect to the TCP or SSL server.

Notes:

- To set this configuration, you should call the command `AT+CIPSERVERMAXCONN=<num>` before creating a server.

Example:

```
AT+CIPMUX=1
AT+CIPSERVERMAXCONN=2
AT+CIPSERVER=1,80
```

4.12 AT+CIPMODE—Configures the Transmission Mode

Query Command:

```
AT+CIPMODE?
Function: to obtain information about transmission mode.
```

Response:

```
+CIPMODE:<mode>
OK
```

Set Command:

```
AT+CIPMODE=<mode>
Function: to set the transmission mode.
```

Response:

```
OK
```

Parameters:

- **mode**:
  - 0: normal transmission mode.
  - 1: UART-Wi-Fi passthrough mode (transparent transmission), which can only be enabled in TCP single connection mode or in UDP mode when the remote IP and port do not change.

Notes:

- The configuration changes will NOT be saved in flash.
- During the UART-WiFi passthrough transmission, if the TCP connection breaks, ESP will keep trying to reconnect until `+++` is input to exit the transmission.
- If it is a normal TCP transmission and the TCP connection breaks, ESP will give a prompt and will not attempt to reconnect.

Example:

```
AT+CIPMODE=1
```

4.13 AT+SAVETRANSLINK—Saves the Transparent Transmission Link in Flash

4.13.1 Save TCP Single Connection in Flash

Set Command:

```
AT+SAVETRANSLINK=<mode>,<remote IP or domain name>,<remote port>[,<type>,
                 <TCP keep alive>]
```

Response:

```
OK
```

Parameters:

- **mode**:
  - 0: normal mode, ESP will NOT enter UART-WiFi passthrough mode on power-up.
  - 1: ESP will enter UART-WiFi passthrough mode on power-up.
- **remote IP**: remote IP or domain name.
- **remote port**: remote port.
- **type**(optional parameter): TCP or UDP, TCP by default.
- **TCP keep alive**(optional parameter): TCP is kept alive. This function is disabled by default.
  - 0: disables the TCP keep-alive function.
  - 1 ~ 7200: keep-alive detection time interval; unit: second (s).

Notes:

- This command will save the UART-WiFi passthrough mode and its link in the NVS area. ESP will enter the UART-WiFi passthrough mode on any subsequent power cycles.
- As long as the remote IP (or domain name) and port are valid, the configuration will be saved in flash.

Example:

```
AT+SAVETRANSLINK=1,"192.168.6.110",1002,"TCP"
```

4.13.2 Save UDP Transmission in Flash

Set Command:

```
AT+SAVETRANSLINK=<mode>,<remote IP>,<remote port>,<type>[,<UDP local port>]
```

Response:

```
OK
```

Parameters:

- **mode**:
  - 0: normal mode; ESP will NOT enter UART-WiFi passthrough mode on power-up.

    - 1: ESP enters UART-WiFi passthrough mode on power-up.
- **remote IP**: remote IP or domain name.
- **remote port**: remote port.
- **type**(optional parameter): UDP, TCP by default.
- **UDP local port**(optional parameter): local port when UDP transparent transmission is enabled on power-up.

Notes:

- This command will save the UART-WiFi passthrough mode and its link in the NVS area. ESP will enter the UART-WiFi passthrough mode on any subsequent power cycles.
- As long as the remote IP (or domain name) and port are valid, the configuration will be saved in flash.

Example:

```
AT+SAVETRANSLINK=1,"192.168.6.110",1002,"UDP",1005
```

4.14 AT+CIPSTO—Sets the TCP Server Timeout

Query Command:

```
AT+CIPSTO?
Function: to check the TCP server timeout.
```

Response:

```
+CIPSTO:<time>
OK
```

Set Command:

```
AT+CIPSTO=<time>
Function: to set the TCP server timeout.
```

Response:

```
OK
```

Parameter:

- **time**: TCP server timeout within the range of 0 ~ 7200s.

Notes:

- ESP configured as a TCP server will disconnect from the TCP client that does not communicate with it until timeout.
- If `AT+CIPSTO=0`, the connection will never time out. This configuration is not recommended.

Example:

```
AT+CIPMUX=1
AT+CIPSERVER=1,1001
AT+CIPSTO=10
```

4.15 AT+CIPSNTPCFG—Sets the Time Zone and the SNTP Server

Query Command:

```
AT+CIPSNTPCFG?
```

Response:

```
+CIPSNTPCFG:<enable>,<timezone>,<SNTP server1>[,<SNTP server2>,<SNTP server3>]
OK
```

Execute Command:

```
AT+CIPSNTPCFG
Function: to clear the SNTP server information.
```

Response:

```
OK
```

Set Command:

```
AT+CIPSNTPCFG=<timezone>[,<SNTP server1>,<SNTP server2>,<SNTP server3>]
```

Response:

```
OK
```

Parameters:

- **enable**:
  - 1: the SNTP server is configured.
  - 0: the SNTP server is not configured.
- **timezone**: time zone, range: [-11,13].
- **SNTP server1**: the first SNTP server.
- **SNTP server2**: the second SNTP server.
- **SNTP server3**: the third SNTP server.

Note:

- If the three SNTP servers are not configured, the following default configuration is used: "cn.ntp.org.cn", "ntp.sjtu.edu.cn", "us.pool.ntp.org".

Example:

```
AT+CIPSNTPCFG=8,"cn.ntp.org.cn","ntp.sjtu.edu.cn"
```

4.16 AT+CIPSNTPTIME—Queries the SNTP Time

Query Command:

```
AT+CIPSNTPTIME?
```

Response:

```
+CIPSNTPTIME:SNTP time
OK
```

Example:

```
AT+CIPSNTPCFG=8,"cn.ntp.org.cn","ntp.sjtu.edu.cn"
OK
AT+CIPSNTPTIME?
+CIPSNTPTIME:Mon Dec 12 02:33:32 2016
OK
```

4.17 AT+CIUPDATE—Updates the Software Through Wi-Fi

Execute Command:

```
AT+CIUPDATE
Function: OTA the lastest version via TCP from server.
```

Response:

```
+CIPUPDATE:<n>
OK
```

Execute Command:

```
AT+CIUPDATE=<ota mode>[,version]
Function: OTA the specified version from server.
```

Response:

```
+CIPUPDATE:<n>
OK
```

Parameters:

- **ota mode**:
  - 0: OTA via TCP
  - 1: OTA via SSL, please ensure `make menuconfig` > `Component config` > `AT` > `OTA based upon ssl` is enabled.
- **version**: AT version, for example, `v1.2.0.0`, `v1.1.3.0`, `v1.1.2.0`
- **n**:
  - 1: find the server.
  - 2: connect to server.
  - 3: get the software version.
  - 4: start updating.

Example:

```
AT+CIUPDATE
```

Or

```
AT+CIUPDATE=1,"v1.2.0.0"
```

Notes:

- The speed of the upgrade is susceptible to the connectivity of the network.
- ERROR will be returned if the upgrade fails due to unfavourable network conditions. Please wait for some time before retrying.

Notice:

- If using Espressif's AT BIN, `AT+CIUPDATE` will download a new AT BIN from the Espressif Cloud.
- If using a user-compiled AT BIN, users need to implement their own AT+CIUPDATE FOTA function. esp-at project provides an example of FOTA.
- It is suggested that users call `AT+RESTORE` to restore the factory default settings after upgrading the AT firmware.

4.18 AT+CIPDINFO—Shows the Remote IP and Port with "+IPD"

Set Command:

```
AT+CIPDINFO=<mode>
```

Response:

```
OK
```

Parameters:

- **mode**:
  - 0: does not show the remote IP and port with "+IPD" and "+CIPRECVDATA".
  - 1: shows the remote IP and port with "+IPD" and "+CIPRECVDATA".

Example:

```
AT+CIPDINFO=1
```

4.19 +IPD—Receives Network Data

Command:

```
Single connection:
(+CIPMUX=0)+IPD,<len>[,<remote IP>,<remote port>]:<data>
multiple connections:
(+CIPMUX=1)+IPD,<link ID>,<len>[,<remote IP>,<remote port>]:<data>
```

Parameters:

- **remote IP**: remote IP string, enabled by command `AT+CIPDINFO=1`.
- **remote port**: remote port, enabled by command `AT+CIPDINFO=1`.
- **link ID**: ID number of connection.
- **len**: data length.
- **data**: data received.

Note:

- The command is valid in normal command mode. When the module receives network data, it will send the data through the serial port using the `+IPD` command.

4.20 AT+CIPSSLCCONF—Config SSL client

Query Command:

```
AT+CIPSSLCCONF?
Function: to get the configuration of each link that running as SSL client.
```

Response:

```
+CIPSSLCCONF:<link ID>,<auth_mode>,<pki_number>,<ca_number>
OK
```

Set Command:

```
 Single connection: (+CIPMUX=0)
AT+CIPSSLCCONF=<auth_mode>,<pki_number>,<ca_number>
Multiple connections: (+CIPMUX=1)
AT+CIPSSLCCONF=<link ID>,<auth_mode>,<pki_number>,<ca_number>
```

Response:

```
OK
```

Parameters:

- **link ID**: ID of the connection (0~max), for multiple connections, if the value is max, it means all connections. By default, max is 5.
- **auth_mode**:
  - 0: no authorization.
  - 1: load cert and private key for server authorization.
  - 2: load CA for client authorize server cert and private key.
  - 3: both authorization.
- **pki_number**: the index of cert and private key, if only one cert and private key, the value should be 0.
- **ca_number**: the index of CA, if only one CA, the value should be 0.

Notes:

- Call this command before establish SSL connection if you want configuration take effect immediately.
- The configuration changes will be saved in the NVS area. If you use AT+SAVETRANSLINK to set SSL passthrough mode, the ESP will establish an SSL connection based on this configuration after next power on.

4.21 AT+CIPRECONNINTV—Set Wi-Fi transparent transmitting auto-connect interval

Set Command:

```
 AT+CIPRECONNINTV=<interval>
Function: to set the interval of auto reconnecting when the TCP/UDP/SSL transmission
          broke in UART-WiFi transparent mode.
```

Parameters:

- **interval**: Time interval for automatic reconnection, default is 1, range is 1~36000, unit is 100ms.

Example:

```
AT+CIPRECONNINTV=10
```

4.22 +IPD—Receives Network Data

Command:

```
Single connection:
(+CIPMUX=0)+IPD,<len>[,<remote IP>,<remote port>]:<data>
multiple connections:
(+CIPMUX=1)+IPD,<link ID>,<len>[,<remote IP>,<remote port>]:<data>
```

Parameters:

- **remote IP**: remote IP, enabled by command `AT+CIPDINFO=1` .
- **remote port**: remote port, enabled by command `AT+CIPDINFO=1` .
- **link ID**: ID number of connection.
- **len**: data length.
- **data**: data received.

Note:

- The command is valid in normal command mode. When the module receives network data, it will send the data through the serial port using the `+IPD` command.

4.23 AT+CIPRECVMODE—Set Socket Receive Mode

Query Command:

```
AT+CIPRECVMODE?
Function: to query socket receive mode.
```

Response:

```
+CIPRECVMODE:<mode>
OK
```

Set Command:

```
AT+CIPRECVMODE=<mode>
```

Response:

```
OK
```

Parameters: - **mode**: the receive mode of socket data is active mode by default. - 0: active mode - ESP AT will send all the received socket data instantly to host MCU through UART with header “+IPD". - 1: passive mode - ESP AT will keep the received socket data in an internal buffer (default is 5840 bytes), and wait for host MCU to read the data. If the buffer is full, the socket transmission will be blocked.

Example:

```
AT+CIPRECVMODE=1
```

Notes:

- The configuration is for TCP and SSL transmission only, and can not be used on WiFi-UART passthrough mode. If it is a UDP transmission in passive mode，data will be missed when buffer full.
- If the passive mode is enabled, when ESP AT receives socket data, it will prompt the following message in different scenarios:
  - for multiple connection mode (AT+CIPMUX=1), the message is: `+IPD,<link ID>,<len>`
  - for single connection mode (AT+CIPMUX=0), the message is: `+IPD,<len>`
  - `<len>` is the total length of socket data in buffer

4.24 AT+CIPRECVDATA—Get Socket Data in Passive Receive Mode

Set Command:

```
 Single connection: (+CIPMUX=0)
AT+CIPRECVDATA=<len>
Multiple connections: (+CIPMUX=1)
AT+CIPRECVDATA=<link_id>,<len>
```

Response:

```
 +CIPRECVDATA:<actual_len>,<data>
OK
```

or

```
 +CIPRECVDATA:<actual_len>,<remote IP>,<remote port>,<data>
OK
```

Parameters: - **link_id**: connection ID in multiple connection mode. -**len**: data length that you want to get, max is 2048 bytes per time. - **actual_len**: length of the data you actually get -**data**: the data you get -**remote IP**: remote IP string, enabled by command `AT+CIPDINFO=1` . - **remote port**: remote port, enabled by command `AT+CIPDINFO=1` .

Example:

```
AT+CIPRECVMODE=1
For example, if host MCU gets a message of receiving 100 bytes data in connection
with No.0, the message will be as following: +IPD,0,100
then you can read those 100 bytes by using the command below
AT+CIPRECVDATA=0,100
```

Notes:

- In a case of disconnection, the buffered Socket data will still be there and can be read by MCU until you send `AT+CIPCLOSE`, or a new connection occupied the previous link_id instead.

4.25 AT+CIPRECVLEN—Get Socket Data Length in Passive Receive Mode

Query Command:

```
AT+CIPRECVLEN?
Function: query the length of the entire data buffered for the link.
```

Response:

```
+CIPRECVLEN:<data length of link0>,<data length of link1>,<data length of link2>,
            <data length of link3>,<data length of link4>
OK
```

Parameters: - **data length of link**: length of the entire data buffered for the link

Example:

```
AT+CIPRECVLEN?
+CIPRECVLEN:100,,,,,
OK
```

Notes:

- For ssl link, it will return the length of encrypted data, so the returned length will be more than the real data length.

4.26 AT+PING: Ping Packets

Set Command:

```
AT+PING=<IP>
Function: Ping packets.
```

Response:

```
+PING:<time>

OK
```

or

```
+timeout

ERROR
```

Parameters: - **IP**: string; host IP or domain name -**time**: the response time of ping, unit: millisecond.

Example:

```
AT+PING="192.168.1.1"
AT+PING="www.baidu.com"
```

4.27 AT+CIPDNS : Configures Domain Name System.

Query Command:

```
AT+CIPDNS?
Function: to obtain current Domain Name System information.
```

Response:

```
+CIPDNS:<enable>[,<"DNS IP1">,<"DNS IP2">,<"DNS IP3">]
OK
```

Set Command:

```
AT+CIPDNS=<enable>[,<"DNS IP1">,<"DNS IP2">,<"DNS IP3">]
Function: Configures Domain Name System.
```

Response:

```
OK
```

or

```
ERROR
```

Parameters: - **enable**: - 0: Enable automatic DNS settings from DHCP, the DNS will be restore to `222.222.67.208`, only when DHCP is updated will it take effect. - 1: Enable manual DNS settings, if not set `DNS IP`, It will use `222.222.67.208` by default. -**DNS IP1**: the first DNS IP. For set command, only for

manual DNS settings; for query command, it is current DNS setting. - **DNS IP2**: the second DNS IP. For set command, only for manual DNS settings; for query command, it is current DNS setting. - **DNS IP3**: the third DNS IP. For set command, only for manual DNS settings; for query command, it is current DNS setting.

Example:

```
AT+CIPDNS=0
AT+CIPDNS=1,"222.222.67.208","114.114.114.114","8.8.8.8"
```

Notes:

- The configuration changes will be saved in the NVS area.
- The three parameters cannot be set to the same server.
- The DNS server may change according to the configuration of the router which the ESP chip connected to.

5. MQTT AT Commands List

5.1 AT+MQTTUSERCFG - Set MQTT User Config

Set Command:

```
AT+MQTTUSERCFG=<LinkID>,<scheme>,<"client_id">,<"username">,<"password">,<cert_key_ID>,
               <CA_ID>,<"path">
```

Function:

```
Set MQTT User Config
```

Response:

```
OK
```

Parameters:

- **LinkID**: only supports link ID 0 for now
- **scheme**:
  - 1: `MQTT over TCP`
  - 2: `MQTT over TLS(no certificate verify)`
  - 3: `MQTT over TLS(verify server certificate)`
  - 4: `MQTT over TLS(provide client certificate)`
  - 5: `MQTT over TLS(verify server certificate and provide client certificate)`
  - 6: `MQTT over WebSocket(based on TCP)`
  - 7: `MQTT over WebSocket Secure(based on TLS, no certificate verify)`

    - 8: `MQTT over WebSocket Secure(based on TLS, verify server certificate)`
    - 9: `MQTT over WebSocket Secure(based on TLS, provide client certificate)`
    - 10: `MQTT over WebSocket Secure(based on TLS, verify server certificate and provide client certificate)`
- **client_id**: MQTT client ID, max length 256Bytes
- **username**: the user name to login to the MQTT broker, max length 64Bytes
- **password**: the password to login to the MQTT broker, max length 64Bytes
- **cert_key_ID**: certificate ID, only supports one certificate of ID 0 for now
- **CA_ID**: CA ID, only supports one CA of ID 0 for now
- **path**: path of the resource, max length 32Bytes

Note:

- The total length of the entire AT command should be less than 256Bytes.

4.2 AT+MQTTCLIENTID - Set MQTT Client ID

Set Command:

```
AT+MQTTCLIENTID=<LinkID><"client_id">
```

Function:

```
Set MQTT Client ID, will cover the parameter client_id in AT+MQTTUSERCFG
User can set a long client id by AT+MQTTCLIENTID.
```

Response:

```
OK
```

Parameters:

- **LinkID**: only supports link ID 0 for now
- **client_id**: MQTT client ID, max length 256Bytes

Note:

- The total length of the entire AT command should be less than 256Bytes.
- AT+MQTTCLIENTID command only could be set after AT+MQTTUSERCFG command

4.3 AT+MQTTUSERNAME - Set MQTT Username

Set Command:

```
AT+MQTTUSERNAME=<LinkID><"client_id">
```

Function:

```
Set MQTT Username, will cover the parameter username in AT+MQTTUSERCFG
User can set a long username by AT+MQTTUSERNAME.
```

Response:

```
OK
```

Parameters:

- **LinkID**: only supports link ID 0 for now
- **username**: the user name to login to the MQTT broker, max length 256Bytes

Note:

- The total length of the entire AT command should be less than 256Bytes.
- AT+MQTTUSERNAME command only could be set after AT+MQTTUSERCFG command

4.4 AT+MQTTPASSWORD - Set MQTT Password

Set Command:

```
AT+MQTTPASSWORD=<LinkID><"password">
```

Function:

```
Set MQTT Password, will cover the parameter password in AT+MQTTUSERCFG
User can set a long username by AT+MQTTPASSWORD.
```

Response:

```
OK
```

Parameters:

- **LinkID**: only supports link ID 0 for now
- **password**: the password to login to the MQTT broker, max length 256Bytes

Note:

- The total length of the entire AT command should be less than 256Bytes.
- AT+MQTTPASSWORD command only could be set after AT+MQTTUSERCFG command

4.5 AT+MQTTCONNCFG - Set configuration of MQTT Connection

Set Command:

```
AT+MQTTCONNCFG=<LinkID>,<keepalive>,<disable_clean_session>,<"lwt_topic">,<"lwt_msg">,
               <lwt_qos>,<lwt_retain>
```

Function:

```
Set configuration of MQTT Connection
```

Response:

```
OK
```

Parameters:

- **LinkID**: only supports link ID 0 for now
- **keepalive**: timeout of MQTT ping, range [60, 7200], unit:second. Default is 120s.
- **disable_clean_session**: set MQTT clean session
  - 0: enable clean session
  - 1: disable clean session
- **lwt_topic**: LWT (Last Will and Testament) message topic, max length 64Bytes
- **lwt_msg**: LWT message, max length 64Bytes
- **lwt_qos**: LWT QoS, can be set to 0, or 1, or 2. Default is 0.
- **lwt_retain**: LWT retain, can be set to 0 or 1. Default is 0.

4.6 AT+MQTTCONN - Connect to MQTT Broker

Set Command:

```
AT+MQTTCONN=<LinkID>,<"host">,<port>,<reconnect>
```

Function:

```
Connect to a MQTT broker.
```

Response:

```
OK
```

Query Command:

```
AT+MQTTCONN?
```

Function:

```
Get the MQTT broker that the ESP chip connected to.
```

Response:

```
+MQTTCONN:<LinkID>,<state>,<scheme><"host">,<port>,<"path">,<reconnect>
OK
```

Parameters:

- **LinkID**: only supports link ID 0 for now
- **host**: MQTT broker domain, max length 128Bytes
- **port**: MQTT broker port, max is port 65535
- **path**: path, max length 32Bytes
- **reconnect**:
  - 0: MQTT will not auto-reconnect
  - 1: MQTT will auto-reconnect, it will take more resource
- **state**: MQTT states
  - 0: MQTT uninitialized
  - 1: already set `AT+MQTTUSERCFG`
  - 2: already set `AT+MQTTCONNCFG`
  - 3: connection disconnected
  - 4: connection established
  - 5: connected, but did not subscribe to any topic
  - 6: connected, and subscribed to MQTT topic
- **scheme**:
  - 1: MQTT over TCP`
  - 2: MQTT over TLS(no certificate verify)
  - 3: MQTT over TLS(verify server certificate)
  - 4: MQTT over TLS(provide client certificate)
  - 5: MQTT over TLS(verify server certificate and provide client certificate)`
  - 6: MQTT over WebSocket(based on TCP)
  - 7: MQTT over WebSocket Secure(based on TLS, no certificate verify)
  - 8: MQTT over WebSocket Secure(based on TLS, verify server certificate)
  - 9: MQTT over WebSocket Secure(based on TLS, provide client certificate)
  - 10: MQTT over WebSocket Secure(based on TLS, verify server certificate and provide client certificate)

4.7 AT+MQTTPUB - Publish MQTT message in string

Set Command:

```
AT+MQTTPUB=<LinkID>,<"topic">,<"data">,<qos>,<retain>
```

Function:

```
Publish MQTT message in string to defined topic. If you need to publish message
in binary, please use command `AT+MQTTPUBRAW` instead.
```

Response:

```
OK
```

Parameters:

- **LinkID**: only supports link ID 0 for now
- **topic**: MQTT topic, max length 64Bytes
- **data**: MQTT message in string.
- **qos**: qos of publish message, can be set to 0, or 1, or 2. Default is 0.
- **retain**: retain flag

Note:

- The total length of the entire AT command should be less than 256Bytes.
- This command cannot send data `\0`, if you need to send `\0`, please use command `AT+MQTTPUBRAW` instead.

4.8 AT+MQTTPUBRAW - Publish MQTT message in binary

Set Command:

```
AT+MQTTPUBRAW=<LinkID>,<"topic">,<length>,<qos>,<retain>
```

Function:

```
Publish MQTT message in binary to defined topic.
```

Response:

```
OK
>
```

Wrap return > after the Set Command. Begin receiving serial data. The AT firmware will keep waiting until the data length defined by is met, all data received will be considered as the MQTT publish message. When the data is met, the transmission of data starts. And then it will respond as the following message.

```
+MQTTPUB:FAIL
```

Or

```
+MQTTPUB:OK
```

Parameters:

- **LinkID**: only supports link ID 0 for now
- **topic**: MQTT topic, max length 64Bytes
- **length**: length of MQTT message, max length is 1024 by default. Users can change the max length limitation by setting `MQTT_BUFFER_SIZE_BYTE` in `make menuconfig`
- **qos**: qos of publish message, can be set to 0, or 1, or 2. Default is 0.
- **retain**: retain flag

4.9 AT+MQTTSUB - Subscribe to MQTT Topic

Set Command:

```
AT+MQTTSUB=<LinkID>,<"topic">,<qos>
```

Function:

```
Subscribe to defined MQTT topic with defined QoS. It supports subscribing to
multiple topics.
```

Response:

```
OK
```

When received MQTT message of the subscribed topic, it will prompt:

```
+MQTTSUBRECV:<LinkID>,<"topic">,<data_length>,data
```

If the topic has been subscribed before, it will prompt:
`ALREADY SUBSCRIBE`

Query Command:

```
AT+MQTTSUB?
```

Function:

```
Get all MQTT topics that already subscribed.
```

Response:

```
+MQTTSUB:<LinkID>,<state>,<"topic1">,<qos>
+MQTTSUB:<LinkID>,<state>,<"topic2">,<qos>
+MQTTSUB:<LinkID>,<state>,<"topic3">,<qos>
...
OK
```

Parameters:

- **LinkID**: only supports link ID 0 for now
- **state**: MQTT states
  - 0: MQTT uninitialized
  - 1: already set `AT+MQTTUSERCFG`
  - 2: already set `AT+MQTTCONNCFG`
  - 3: connection disconnected
  - 4: connection established
  - 5: connected, but did not subscribe to any topic
  - 6: connected, and subscribed to MQTT topic
- **topic**: the topic that subscribed to
- **qos**: the QoS that subscribed to

4.10 AT+MQTTUNSUB - Unsubscribe from MQTT Topic

Set Command:

```
AT+MQTTUNSUB=<LinkID>,<"topic">
```

Function:

```
Unsubscribe the client from defined topic. This command can be called multiple times
to unsubscribe from differrent topics.
```

Response:

```
OK
```

Parameters:

- **LinkID**: only supports link ID 0 for now
- **topic**: MQTT topic, max length 64Bytes

Note:

- If the topic has not been subscribed, then the AT log will prompt `NO UNSUBSCRIBE`. And the AT command will still respond `OK`.

4.11 AT+MQTTCLEAN - Close the MQTT Connection

Set Command:

```
AT+MQTTCLEAN=<LinkID>
```

Function:

```
Close the MQTT connection, and release the resource.
```

Response:

```
OK
```

Parameters:

- **LinkID**: only supports link ID 0 for now

4.12 MQTT Error Codes

The MQTT Error code will be prompt as `ERR CODE:0x<%08x>`.

```
    AT_MQTT_NO_CONFIGURED,                          // 0x6001
    AT_MQTT_NOT_IN_CONFIGURED_STATE,                // 0x6002
    AT_MQTT_UNINITIATED_OR_ALREADY_CLEAN,           // 0x6003
    AT_MQTT_ALREADY_CONNECTED,                      // 0x6004
    AT_MQTT_MALLOC_FAILED,                          // 0x6005
    AT_MQTT_NULL_LINK,                              // 0x6006
    AT_MQTT_NULL_PARAMTER,                          // 0x6007
    AT_MQTT_PARAMETER_COUNTS_IS_WRONG,              // 0x6008
    AT_MQTT_TLS_CONFIG_ERROR,                       // 0x6009
    AT_MQTT_PARAM_PREPARE_ERROR,                    // 0x600A
    AT_MQTT_CLIENT_START_FAILED,                    // 0x600B
    AT_MQTT_CLIENT_PUBLISH_FAILED,                  // 0x600C
    AT_MQTT_CLIENT_SUBSCRIBE_FAILED,                // 0x600D
    AT_MQTT_CLIENT_UNSUBSCRIBE_FAILED,              // 0x600E
    AT_MQTT_CLIENT_DISCONNECT_FAILED,               // 0x600F
    AT_MQTT_LINK_ID_READ_FAILED,                    // 0x6010
    AT_MQTT_LINK_ID_VALUE_IS_WRONG,                 // 0x6011
    AT_MQTT_SCHEME_READ_FAILED,                     // 0x6012
    AT_MQTT_SCHEME_VALUE_IS_WRONG,                  // 0x6013
    AT_MQTT_CLIENT_ID_READ_FAILED,                  // 0x6014
    AT_MQTT_CLIENT_ID_IS_NULL,                      // 0x6015
    AT_MQTT_CLIENT_ID_IS_OVERLENGTH,                // 0x6016
    AT_MQTT_USERNAME_READ_FAILED,                   // 0x6017
    AT_MQTT_USERNAME_IS_NULL,                       // 0x6018
    AT_MQTT_USERNAME_IS_OVERLENGTH,                 // 0x6019
    AT_MQTT_PASSWORD_READ_FAILED,                   // 0x601A
    AT_MQTT_PASSWORD_IS_NULL,                       // 0x601B
    AT_MQTT_PASSWORD_IS_OVERLENGTH                  // 0x601C
```

```
    AT_MQTT_PASSWORD_IS_OVERLENGTH,                 // 0x601C
    AT_MQTT_CERT_KEY_ID_READ_FAILED,                // 0x601D
    AT_MQTT_CERT_KEY_ID_VALUE_IS_WRONG,             // 0x601E
    AT_MQTT_CA_ID_READ_FAILED,                      // 0x601F
    AT_MQTT_CA_ID_VALUE_IS_WRONG,                   // 0x6020
    AT_MQTT_CA_LENGTH_ERROR,                        // 0x6021
    AT_MQTT_CA_READ_FAILED,                         // 0x6022
    AT_MQTT_CERT_LENGTH_ERROR,                      // 0x6023
    AT_MQTT_CERT_READ_FAILED,                       // 0x6024
    AT_MQTT_KEY_LENGTH_ERROR,                       // 0x6025
    AT_MQTT_KEY_READ_FAILED,                        // 0x6026
    AT_MQTT_PATH_READ_FAILED,                       // 0x6027
    AT_MQTT_PATH_IS_NULL,                           // 0x6028
    AT_MQTT_PATH_IS_OVERLENGTH,                     // 0x6029
    AT_MQTT_VERSION_READ_FAILED,                    // 0x602A
    AT_MQTT_KEEPALIVE_READ_FAILED,                  // 0x602B
    AT_MQTT_KEEPALIVE_IS_NULL,                      // 0x602C
    AT_MQTT_KEEPALIVE_VALUE_IS_WRONG,               // 0x602D
    AT_MQTT_DISABLE_CLEAN_SESSION_READ_FAILED,      // 0x602E
    AT_MQTT_DISABLE_CLEAN_SESSION_VALUE_IS_WRONG,   // 0x602F
    AT_MQTT_LWT_TOPIC_READ_FAILED,                  // 0x6030
    AT_MQTT_LWT_TOPIC_IS_NULL,                      // 0x6031
    AT_MQTT_LWT_TOPIC_IS_OVERLENGTH,                // 0x6032
    AT_MQTT_LWT_MSG_READ_FAILED,                    // 0x6033
    AT_MQTT_LWT_MSG_IS_NULL,                        // 0x6034
    AT_MQTT_LWT_MSG_IS_OVERLENGTH,                  // 0x6035
    AT_MQTT_LWT_QOS_READ_FAILED,                    // 0x6036
    AT_MQTT_LWT_QOS_VALUE_IS_WRONG,                 // 0x6037
    AT_MQTT_LWT_RETAIN_READ_FAILED,                 // 0x6038
    AT_MQTT_LWT_RETAIN_VALUE_IS_WRONG,              // 0x6039
    AT_MQTT_HOST_READ_FAILED,                       // 0x603A
    AT_MQTT_HOST_IS_NULL,                           // 0x603B
    AT_MQTT_HOST_IS_OVERLENGTH,                     // 0x603C
    AT_MQTT_PORT_READ_FAILED,                       // 0x603D
    AT_MQTT_PORT_VALUE_IS_WRONG,                    // 0x603E
    AT_MQTT_RECONNECT_READ_FAILED,                  // 0x603F
    AT_MQTT_RECONNECT_VALUE_IS_WRONG,               // 0x6040
    AT_MQTT_TOPIC_READ_FAILED,                      // 0x6041
    AT_MQTT_TOPIC_IS_NULL,                          // 0x6042
    AT_MQTT_TOPIC_IS_OVERLENGTH,                    // 0x6043
    AT_MQTT_DATA_READ_FAILED,                       // 0x6044
    AT_MQTT_DATA_IS_NULL,                           // 0x6045
    AT_MQTT_DATA_IS_OVERLENGTH,                     // 0x6046
    AT_MQTT_QOS_READ_FAILED,                        // 0x6047
    AT_MQTT_QOS_VALUE_IS_WRONG,                     // 0x6048
    AT_MQTT_RETAIN_READ_FAILED,                     // 0x6049
    AT_MQTT_RETAIN_VALUE_IS_WRONG,                  // 0x604A
    AT_MQTT_PUBLISH_LENGTH_READ_FAILED,             // 0x604B
    AT_MQTT_PUBLISH_LENGTH_VALUE_IS_WRONG,          // 0x604C
```

```
    AT_MQTT_RECV_LENGTH_IS_WRONG,                   // 0x604D
    AT_MQTT_CREATE_SEMA_FAILED,                     // 0x604E
    AT_MQTT_CREATE_EVENT_GROUP_FAILED,              // 0x604F
```

4.13 MQTT Notes

- In general, AT MQTT commands will be responded within 10s, except command `AT+MQTTCONN`.For example, if the router fails to access to the internet, the command `AT+MQTTPUB` will respond within 10s. But the command `AT+MQTTCONN` may need more time due to the packet retransmission in bad network environment.
- If the `AT+MQTTCONN` is based on a TLS connection, the timeout of each packet is 10s, then the total timeout will be much longer depending on the handshake packets count.
- When the MQTT connection ends, it will prompt message `+MQTTDISCONNECTED:<LinkID>`
- When the MQTT connection established, it will prompt message `+MQTTCONNECTED:<LinkID>,<scheme>,<"host">,port,<"path">,<reconnect>`

4.14 Example 1: MQTT over TCP (with a Local MQTT Broker)

Create a local MQTT broker. For example, the MQTT broker's IP address is "192.168.31.113", port 1883. Then the example of communicating with the MQTT broker will be as the following steps.

```
AT+MQTTUSERCFG=0,1,"ESP","espressif","1234567890",0,0,""
AT+MQTTCONN=0,"192.168.31.113",1883,0
AT+MQTTSUB=0,"topic",1
AT+MQTTPUB=0,"topic","test",1,0
AT+MQTTCLEAN=0
```

4.15 Example 2: MQTT over TLS (with a Local MQTT Broker)

Create a local MQTT broker. For example, the MQTT broker's IP address is "192.168.31.113", port 1883. Then the example of communicating with the MQTT broker will be as the following steps.

```
AT+CIPSNTPCFG=1,8,"ntp1.aliyun.com"
AT+CIPSNTPTIME?
AT+MQTTUSERCFG=0,3,"ESP","espressif","1234567890",0,0,""
AT+MQTTCONNCFG=0,0,0,"lwtt","lwtm",0,0
AT+MQTTCONN=0,"192.168.31.113",1883,0
AT+MQTTSUB=0,"topic",1
AT+MQTTPUB=0,"topic","test",1,0
AT+MQTTCLEAN=0
```

4.16 Example 3: MQTT over WSS

This is an example of communicating with MQTT broker: iot.eclipse.org, of which port is 443.

```
 AT+CIPSNTPCFG=1,8,"ntp1.aliyun.com"
AT+CIPSNTPTIME?
AT+MQTTUSERCFG=0,7,"ESP","espressif","1234567890",0,0,"wss"
AT+MQTTCONN=0,"iot.eclipse.org",443,0
AT+MQTTSUB=0,"topic",1
AT+MQTTPUB=0,"topic","test",1,0
AT+MQTTCLEAN=0
```