SimpleLink™ Wi-Fi® AT Command User's Guide

The SimpleLink™ Wi-Fi® Internet-on-a chip™ family of devices from Texas Instruments™ provides a suite of integrated protocols for Wi-Fi and internet connectivity to dramatically simplify the implementation of internet-enabled devices and applications.

This document describes the AT command protocol for SimpleLink, which is a widely used method to configure and control embedded networking systems due to its simplicity, textual parameter representation, and inherent flexibility.

Contents
1 Supported Platforms ................................................................................................................. 4
2 Architecture Overview ................................................................................................................. 4
3 Getting Started ....................................................................................................................... 5
4 Commands Summary ................................................................................................................ 5
5 Protocol Syntax .......................................................................................................................... 7
6 Command Description ................................................................................................................ 8

List of Tables
1 Device Commands .................................................................................................................. 5
2 Socket Commands ................................................................................................................... 5
3 WLAN Commands .................................................................................................................. 6
4 File System Commands .......................................................................................................... 6
5 Network Application Commands ............................................................................................. 6
6 Network Configuration Commands ........................................................................................ 6
7 Network Utility Commands ...................................................................................................... 7
8 Asynchronous Events ............................................................................................................... 7
9 AT+Start Starts the NWP .......................................................................................................... 8
10 AT+Stop Stops the NWP .......................................................................................................... 8
11 AT+Get Getting Device Configurations .................................................................................. 9
12 AT+Set Setting Device Configurations .................................................................................... 10
13 AT+Test Test Command ......................................................................................................... 10
14 AT+Socket Create an End-Point for Communication .............................................................. 10
15 AT+Close Close Socket .......................................................................................................... 11
16 AT+Accept Accept a Connection on a Socket ........................................................................ 11
17 AT+Bind Assign a Name to a Socket ....................................................................................... 11
18 AT+Listen Listen for Connections on a Socket ...................................................................... 11
19 AT+Connect Initiate a Connection on a Socket ........................................................................ 12
20 AT+Select Monitor Socket Activity ......................................................................................... 12
21 AT+SetSockOpt Set Socket Options ....................................................................................... 13
22 AT+GetSockOpt Get Socket Options ...................................................................................... 15
23 AT+Recv Read Data From TCP Socket .................................................................................... 15
24 AT+RecvFrom Read Data From Socket .................................................................................... 16
25 AT+Send Write Data to TCP Socket ....................................................................................... 16
26 AT+SendTo Write Data to Socket ............................................................................................. 16

Copyright © 2017–2020, Texas Instruments Incorporated
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+WlanConnect</td>
<td>Connect to WLAN Network as a Station</td>
<td>17</td>
</tr>
<tr>
<td>AT+WlanDisconnect</td>
<td>Disconnect the Connection</td>
<td>17</td>
</tr>
<tr>
<td>AT+WlanProfileAdd</td>
<td>Add Profile</td>
<td>18</td>
</tr>
<tr>
<td>AT+WlanProfileGet</td>
<td>Get Profile</td>
<td>18</td>
</tr>
<tr>
<td>AT+WlanProfileDel</td>
<td>Delete Profile</td>
<td>19</td>
</tr>
<tr>
<td>AT+WlanPolicySet</td>
<td>Set Policy Values</td>
<td>19</td>
</tr>
<tr>
<td>AT+WlanPolicyGet</td>
<td>Get Policy Values</td>
<td>21</td>
</tr>
<tr>
<td>AT+WlanScan</td>
<td>Gets the WLAN Scan Operation Results</td>
<td>21</td>
</tr>
<tr>
<td>AT+WlanSetMode</td>
<td>WLAN Set Mode</td>
<td>22</td>
</tr>
<tr>
<td>AT+WlanSet</td>
<td>Setting WLAN Configurations</td>
<td>22</td>
</tr>
<tr>
<td>AT+WlanGet</td>
<td>Getting WLAN Configurations</td>
<td>24</td>
</tr>
<tr>
<td>AT+FileOpen</td>
<td>Open File in Storage Device</td>
<td>26</td>
</tr>
<tr>
<td>AT+FileClose</td>
<td>Close File in Storage Device</td>
<td>26</td>
</tr>
<tr>
<td>AT+FileCtl</td>
<td>Controls Various File System Operations</td>
<td>27</td>
</tr>
<tr>
<td>AT+FileDel</td>
<td>Delete File From Storage Device</td>
<td>28</td>
</tr>
<tr>
<td>AT+FileGetFilelist</td>
<td>Get a List of Files</td>
<td>28</td>
</tr>
<tr>
<td>AT+FileGetInfo</td>
<td>Get Information About a File</td>
<td>28</td>
</tr>
<tr>
<td>AT+FileRead</td>
<td>Read a Block of Data From a File in Storage Device</td>
<td>28</td>
</tr>
<tr>
<td>AT+FileWrite</td>
<td>Write Block of Data to a File in Storage Device</td>
<td>29</td>
</tr>
<tr>
<td>AT+NetAPPStart</td>
<td>Starts a Network Application</td>
<td>30</td>
</tr>
<tr>
<td>AT+NetAPPStop</td>
<td>Stops a Network Application</td>
<td>30</td>
</tr>
<tr>
<td>AT+NetAPPGetHostName</td>
<td>Get Host IP by Name</td>
<td>30</td>
</tr>
<tr>
<td>AT+NetAPPGetHostByService</td>
<td>Get Host IP by Service</td>
<td>31</td>
</tr>
<tr>
<td>AT+NetAPPSet</td>
<td>Setting Network Application Configurations</td>
<td>31</td>
</tr>
<tr>
<td>AT+NetAPPGet</td>
<td>Getting Network Application Configurations</td>
<td>33</td>
</tr>
<tr>
<td>AT+NetAPPSend</td>
<td>Sends Network Application Response or Data Following a Network Application</td>
<td>33</td>
</tr>
<tr>
<td>AT+NetAPPRecv</td>
<td>Receives Data From the Network Processor Following a Network Application</td>
<td>33</td>
</tr>
<tr>
<td>AT+NetAPPPing</td>
<td>Send Ping to Network Hosts</td>
<td>34</td>
</tr>
<tr>
<td>AT+NetAPPGetServiceList</td>
<td>Get Service List</td>
<td>35</td>
</tr>
<tr>
<td>AT+NetAPPRegisterService</td>
<td>Register a New mDNS Service</td>
<td>35</td>
</tr>
<tr>
<td>AT+NetAPPUnRegisterService</td>
<td>Unregister mDNS Service</td>
<td>36</td>
</tr>
<tr>
<td>AT+NetCfsSet</td>
<td>Setting Network Configurations</td>
<td>37</td>
</tr>
<tr>
<td>AT+NetCfsGet</td>
<td>Getting Network Configurations</td>
<td>38</td>
</tr>
<tr>
<td>AT+NetUtilGet</td>
<td>Getting Utilities Configurations</td>
<td>40</td>
</tr>
<tr>
<td>AT+NetUtilCmd</td>
<td>Performing Utilities-Related Commands</td>
<td>41</td>
</tr>
<tr>
<td>+EventFatalError</td>
<td>Fatal Error Event for Inspecting Fatal Error</td>
<td>42</td>
</tr>
<tr>
<td>+EventGeneral</td>
<td>General Asynchronous Event for Inspecting General Events</td>
<td>42</td>
</tr>
<tr>
<td>+EventWlan</td>
<td>WLAN Asynchronous Event</td>
<td>43</td>
</tr>
<tr>
<td>+EventNetApp</td>
<td>Network Application Asynchronous Event</td>
<td>44</td>
</tr>
<tr>
<td>+EventSock</td>
<td>Socket Asynchronous Event</td>
<td>44</td>
</tr>
<tr>
<td>+EventMqtt</td>
<td>MQTT Asynchronous Event</td>
<td>45</td>
</tr>
<tr>
<td>AT+MqttCreate</td>
<td>MQTT Client Create</td>
<td>46</td>
</tr>
<tr>
<td>AT+MqttDelete</td>
<td>MQTT Client Delete</td>
<td>47</td>
</tr>
<tr>
<td>AT+MqttConnect</td>
<td>MQTT Client Connect to Broker</td>
<td>47</td>
</tr>
<tr>
<td>AT+MqttDisconnect</td>
<td>MQTT Client Disconnect From Broker</td>
<td>47</td>
</tr>
<tr>
<td>AT+MqttPublish</td>
<td>MQTT Client Send Message to Broker</td>
<td>48</td>
</tr>
<tr>
<td>AT+MqttSubscribe</td>
<td>MQTT Client Subscribe for Topic</td>
<td>48</td>
</tr>
</tbody>
</table>
74  **AT+MqttUnsubscribe** MQTT Client Unsubscribe for Topic ................................................................. 48
75  **AT+MqttSet** MQTT Client Set Option ............................................................................................ 49
76  **AT+HttpCreate**  Http Client Create ............................................................................................... 50
77  **AT+HttpDestroy**  Http Client Delete .............................................................................................. 50
78  **AT+HttpConnect** Http Client Connect to Host ................................................................................ 50
79  **AT+HttpDisconnect** Http Client Disconnect From Host .................................................................. 50
80  **AT+HttpSendReq** Http Client Send Request to Host ...................................................................... 51
81  **AT+HttpReadResBody** Http Client Read Response Body From Host ................................................. 51
82  **AT+HttpSetHeader** Http Client Set Header ..................................................................................... 52
83  **AT+HttpGetHeader** Http Client Get Header ..................................................................................... 53
84  **AT+HttpSetOpt** Http Client Set Option .......................................................................................... 54
85  **AT+HttpSetProxy** Http Client Set Proxy Address .......................................................................... 54

**Trademarks**

SimpleLink, Internet-on-a chip, Texas Instruments are trademarks of Texas Instruments. Wi-Fi, Wi-Fi Direct are registered trademarks of Wi-Fi Alliance. All other trademarks are the property of their respective owners.
1 Supported Platforms

Hardware platforms that support the AT command library are:
- CC3220R
- CC3220S
- CC3220SF

2 Architecture Overview

SimpleLink Wi-Fi AT Command consists of two main modules:
- AT Commands Application
  The application is one of the following application demos:
  - The AT_Commands application provides control by the AT Commands on the local device.
  - The Serial_wifi application provides control by the AT Commands on the local and the remote device.
  - The user-customized application is based on the two previous applications.
- AT Command Core
  - The core includes the command parser, execution, and return status.
  - The AT Command Core should already be compiled into the library.

The following API communicate between the two modules:
- ATCmd_create creates the AT Command core task and initializes the RX event queue.
- ATCmd_send transmits string from the AT Command application to the AT Command Core.
  The function takes one parameter, Buffer, which stores the sent string.
- ATCmd_recv transmits a string from the AT Command Core to the AT Command application.
  The function takes two parameters:
  - Buffer stores the received string.
  - Nonblock variant set to 0 for waits forever on the RX queue, otherwise set to 1.

All send and receive buffers should be allocated by the AT Commands application.

Figure 1 shows the basic architecture.

![Basic Architecture Scheme](image-url)
3 Getting Started

The following describes the procedure to build the AT Command Core. For building and executing the application binary file, refer to the README.html file that is located in each AT Command application. Ensure that the AT Command library includes in the application linking list.

The AT Command Core is prebuilt into the library “atcmd.a” per two OS (TI-RTOS and FreeRTOS) and per three compilers (CCS, GCC, and IAR). In the case where changes must be made to the core and you need to recompile it, there are two ways to build it:

- For CCS (TI-RTOS or FreeRTOS), import the CCS project located under (SDK ROOT)\source\ti\net\atcmd\ccs and build the library.

  **NOTE:** Pay attention to choose the appropriate product number.

- For all other favorites (including CCS), open the command prompt line under the directory (SDK ROOT)\source\ti\net\atcmd, and execute `gmake` from the XDC tool root directory. To clean all outputs, execute `gmake clean`.

4 Commands Summary

**Table 1. Device Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+Start</td>
<td>Starts the network processor (NWP)</td>
</tr>
<tr>
<td>AT+Stop</td>
<td>Stops the NWP</td>
</tr>
<tr>
<td>AT+Get</td>
<td>Gets device configurations</td>
</tr>
<tr>
<td>AT+Set</td>
<td>Sets device configurations</td>
</tr>
<tr>
<td>AT+Test</td>
<td>Test command</td>
</tr>
</tbody>
</table>

**Table 2. Socket Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+Socket</td>
<td>Create an endpoint for communication</td>
</tr>
<tr>
<td>AT+Close</td>
<td>Close socket</td>
</tr>
<tr>
<td>AT+Accept</td>
<td>Accept a connection on a socket</td>
</tr>
<tr>
<td>AT+Bind</td>
<td>Assign a name to a socket</td>
</tr>
<tr>
<td>AT+Listen</td>
<td>Listen for connections on a socket</td>
</tr>
<tr>
<td>AT+Connect</td>
<td>Initiate a connection on a socket</td>
</tr>
<tr>
<td>AT+Select</td>
<td>Monitor socket activity</td>
</tr>
<tr>
<td>AT+SetSockOpt</td>
<td>Set socket options</td>
</tr>
<tr>
<td>AT+GetSockOpt</td>
<td>Get socket options</td>
</tr>
<tr>
<td>AT+Recv</td>
<td>Read data from TCP socket</td>
</tr>
<tr>
<td>AT+RecvFrom</td>
<td>Read data from socket</td>
</tr>
<tr>
<td>AT+Send</td>
<td>Write data to TCP socket</td>
</tr>
<tr>
<td>AT+SendTo</td>
<td>Write data to socket</td>
</tr>
</tbody>
</table>
### Table 3. WLAN Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+WlanConnect</td>
<td>Connect to WLAN network as a station</td>
</tr>
<tr>
<td>AT+WlanDisconnect</td>
<td>Disconnect connection</td>
</tr>
<tr>
<td>AT+WlanProfileAdd</td>
<td>Add profile</td>
</tr>
<tr>
<td>AT+WlanProfileGet</td>
<td>Get profile</td>
</tr>
<tr>
<td>AT+WlanProfileDel</td>
<td>Delete profile</td>
</tr>
<tr>
<td>AT+WlanPolicySet</td>
<td>Set policy values</td>
</tr>
<tr>
<td>AT+WlanPolicyGet</td>
<td>Get policy values</td>
</tr>
<tr>
<td>AT+WlanScan</td>
<td>Gets the WLAN scan operation results</td>
</tr>
<tr>
<td>AT+WlanSetMode</td>
<td>WLAN set mode</td>
</tr>
<tr>
<td>AT+WlanSet</td>
<td>Setting WLAN configurations</td>
</tr>
<tr>
<td>AT+WlanGet</td>
<td>Getting WLAN configurations</td>
</tr>
</tbody>
</table>

### Table 4. File System Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+FileOpen</td>
<td>Open file in storage device</td>
</tr>
<tr>
<td>AT+FileClose</td>
<td>Close file in storage device</td>
</tr>
<tr>
<td>AT+FileCtl</td>
<td>Controls various file system operations</td>
</tr>
<tr>
<td>AT+FileDel</td>
<td>Delete file from storage device</td>
</tr>
<tr>
<td>AT+FileGetFilelist</td>
<td>Get list of files</td>
</tr>
<tr>
<td>AT+FileGetInfo</td>
<td>Get information of a file</td>
</tr>
<tr>
<td>AT+FileRead</td>
<td>Read block of data from a file in storage device</td>
</tr>
<tr>
<td>AT+FileWrite</td>
<td>Write block of data to a file in storage device</td>
</tr>
</tbody>
</table>

### Table 5. Network Application Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+NetAPPStart</td>
<td>Starts a network application</td>
</tr>
<tr>
<td>AT+NetAPPStop</td>
<td>Stops a network application</td>
</tr>
<tr>
<td>AT+NetAPPGetHostName</td>
<td>Get host IP by name</td>
</tr>
<tr>
<td>AT+NetAPPGetHostByService</td>
<td>Host IP by service</td>
</tr>
<tr>
<td>AT+NetAPPSet</td>
<td>Setting network applications configurations</td>
</tr>
<tr>
<td>AT+NetAPPGet</td>
<td>Getting network applications configurations</td>
</tr>
<tr>
<td>AT+NetAPPSend</td>
<td>Sends Network Application response or data following a Network Application request event</td>
</tr>
<tr>
<td>AT+NetAPPRecv</td>
<td>Receives data from the network processor following a Network Application response event</td>
</tr>
<tr>
<td>AT+NetAPPing</td>
<td>Send ping to network hosts</td>
</tr>
<tr>
<td>AT+NetAPPGetServiceList</td>
<td>Get service list</td>
</tr>
<tr>
<td>AT+NetAPPRegisterService</td>
<td>Register a new mDNS service</td>
</tr>
<tr>
<td>AT+NetAPPUnRegisterService</td>
<td>Unregister mDNS service</td>
</tr>
</tbody>
</table>

### Table 6. Network Configuration Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+NetCfgSet</td>
<td>Setting network configurations</td>
</tr>
<tr>
<td>AT+NetCfgGet</td>
<td>Getting network configurations</td>
</tr>
</tbody>
</table>
5 Protocol Syntax

5.1 Commands

Syntax:
```
AT<command name>=<param1>, <param2>, ..., <paramX>
```

- Commands that contain parameters should include an equal sign (=) between the command name and the first parameter.
- Commands that contain parameters should include a comma mark (,) as a delimiter between them—comma delimiters are mandatory.
- In case the parameter is defined as "ignore" or "optional", it could be left empty but the comma delimiter should be mentioned—it looks like two conjunction delimiters (,,).
- Parameters that are left empty must be treated as 0 or NULL (according to the parameter type), and in case it was not defined as "ignore" or "optional", an error should be raised.
- String parameters containing spaces must be enclosed with quotes (" ").
- String parameters containing a comma delimiter (,) must be enclosed with quotes (" ").
- Numeric value parameters could be one of the following:
  - Decimal
  - Hexadecimal—must have a prefix of zero x notation (0x)
- Numeric array parameters could be one of the following:
  - IPv4 address—contains four numeric values (8 bits each) with a point mark (.) as a delimiter between them enclosed with or without square brackets—x.x.x.x or [x.x.x.x]
  - IPv6 address—contains four numeric values (32 bit each) with a colon mark (:) as a delimiter between them enclosed with or without square brackets—x:x:x:x or [x:x:x:x]
  - MAC address—contains six numeric values (8 bit each) with a colon mark (:) as a delimiter between them enclosed with or without square brackets—x:x:x:x:x:x or [x:x:x:x:x:x]
- Bitmask parameters should contain values with a vertical bar ( | ) as delimiter between them enclosed with or without square brackets—x|x|x or [x|x|x]
- The AT command handler allows for the AT commands to be entered in uppercase or lowercase with spaces between the arguments.
- Data parameter should be one of the following formats:
  - Binary format
  - Base64 format—binary to text encoding
5.2 Command Return Status

Command return status could be one of the following cases:

- Command that returns values:
  <command name>: <value1>, ..., <valueX>

- Command that returns success:
  OK

- Command that returns failure:
  ERROR:<error description>, <error code>

Command return status should include a colon mark (:) between the command name and the first value. Command return status that contains list values should include a semicolon mark (;) as a delimiter between the list members.

5.3 Asynchronous Event

The events may arrive at any time. Asynchronous events are always built in the following format:

<event name>: <event ID>,<value1>,...,<valueX>

The event should include a colon mark (:) between the event name and the event ID.

6 Command Description

6.1 Device Commands

Table 9. AT+Start Starts the NWP

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+Start</td>
<td>OK</td>
</tr>
<tr>
<td>Arguments:</td>
<td>none</td>
</tr>
</tbody>
</table>

Table 10. AT+Stop Stops the NWP

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+Stop = [Timeout]</td>
<td>OK</td>
</tr>
<tr>
<td>Arguments:</td>
<td>none</td>
</tr>
<tr>
<td>Timeout: Stop timeout in milliseconds should be used to give the device time to finish any transmission or reception that is not completed when the function was called.</td>
<td></td>
</tr>
<tr>
<td>0: Enter to hibernate immediately</td>
<td></td>
</tr>
<tr>
<td>0xFFFF: Host waits for the response from the device before hibernating, without timeout protection</td>
<td></td>
</tr>
<tr>
<td>0 &lt;Timeout[msec] &lt;0xFFFF: Host waits for the response from the device before hibernating, with a defined timeout protection This timeout defines the maximum time to wait. The NWP response can be sent earlier than this timeout.</td>
<td></td>
</tr>
</tbody>
</table>
## Table 11. AT+Get Getting Device Configurations

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+Get = [ID],[Option]</td>
<td>+Get:[Value1],...,[ValueX]</td>
</tr>
</tbody>
</table>

### Arguments:

<table>
<thead>
<tr>
<th>ID</th>
<th>Option</th>
<th>Return Values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Device</td>
<td>Value1: bitmask: General error</td>
</tr>
<tr>
<td></td>
<td>WLAN</td>
<td>Value1: bitmask:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- WLANASYNC_CONNECTEDRESPONSE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- WLANASYNC_DISCONNECTEDRESPONSE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- STA_CONNECTED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- STA_DISCONNECTED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- P2P_DEV_FOUND</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- CONNECTION_FAILED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- P2P_NEG_REQ_RECEIVED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- RX_FILTERS</td>
</tr>
<tr>
<td></td>
<td>BSD</td>
<td>Value1: bitmask:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- TX_FAILED</td>
</tr>
<tr>
<td></td>
<td>NETAPP</td>
<td>Value1: bitmask:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- IPACQUIRED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- IPACQUIRED_V6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- IP_LEASED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- IP_RELEASED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- IPV4_LOST</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- DHCP_ACQUIRE_TIMEOUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- IP_COLLISION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- IPV6_LOST</td>
</tr>
<tr>
<td></td>
<td>Version</td>
<td>Value1: Chip ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value2: FW Version (x.x.x.x)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value3: PHY Version (x.x.x.x)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value4: NWP Version (x.x.x.x)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value5: ROM Version</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>Value1: Hour = Current hours</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value2: Minute = Current minutes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value3: Second = Current seconds</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value4: Day = Current Date, 1–31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value5: Month = Current Month, 1–12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value6: Year = Current year</td>
</tr>
<tr>
<td></td>
<td>Persistent</td>
<td>Value1:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1: Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 0: Disable</td>
</tr>
<tr>
<td></td>
<td>IOT</td>
<td>UDID 16 bytes</td>
</tr>
</tbody>
</table>

---

**Note:** The table provides a comprehensive overview of the Return Values for various ID options, including Device, WLAN, BSD, NETAPP, Version, Time, and Persistent categories. Each category lists specific values and their corresponding descriptions, offering a clear understanding of the configuration details available through the AT+Get command.
### Table 12. AT+Set Setting Device Configurations

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+Set = [ID],[Option],[Value1],..,[ValueX]</td>
<td>OK</td>
</tr>
</tbody>
</table>

**Arguments:**

<table>
<thead>
<tr>
<th>ID</th>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>Persistent sets the default system-wide configuration persistence mode. In case true, all APIs that follow system configured persistence (see persistence attribute noted per API) shall maintain the configured settings. In case false, all calls to APIs that follow system configured persistence shall be volatile. Configuration should revert to default after reset or power recycle.</td>
<td>Value1:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Disable</td>
</tr>
<tr>
<td></td>
<td>Time sets the device time and date</td>
<td>Value2: Hour = Current hours</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value3: Minute = Current minutes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value4: Day = Current Date, 1–31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value5: Month = Current Month, 1–12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value6: Year = Current year</td>
</tr>
</tbody>
</table>

### Table 13. AT+Test Test Command

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+Test</td>
<td>OK</td>
</tr>
</tbody>
</table>

**Arguments:**

<table>
<thead>
<tr>
<th>Arguments:</th>
<th>none</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Arguments:</th>
<th>none</th>
</tr>
</thead>
</table>

### 6.2 Socket Commands

### Table 14. AT+Socket Create an End-Point for Communication

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+Socket = [Domain],[Type],[Protocol]</td>
<td>+Socket: [socket] OK</td>
</tr>
</tbody>
</table>

**Arguments:**

- Domain: Specifies the protocol family of the created socket:
  - INET: For network protocol IPv4
  - INET6: For network protocol IPv6
  - RF: For starting transceiver mode
- Type: Specifies the communication semantic:
  - STREAM: Reliable stream-oriented service or Stream Sockets
  - DGRAM: Datagram service or Datagram Sockets
  - RAW: Raw protocols atop the network layer
- Protocol: Specifies a particular transport to be used with the socket:
  - TCP
  - UDP
  - RAW
  - SEC

**Arguments:**

- Socket: Socket descriptor that will be used in the socket commands described in Table 15 through Table 26.
Table 15. **AT+Close** Close Socket

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+Close = [socket]</td>
<td>+Close: [socket]</td>
</tr>
<tr>
<td></td>
<td>OK</td>
</tr>
</tbody>
</table>

Arguments:
- **socket**: Socket descriptor received from AT+Socket command

Table 16. **AT+Accept** Accept a Connection on a Socket

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+Accept = [socket],[family]</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>+Accept:</td>
</tr>
<tr>
<td></td>
<td>[New Socket],[Family],[Port],[Address]</td>
</tr>
</tbody>
</table>

Arguments:
- **socket**: Socket descriptor received from AT+Socket command
- **family**: Specifies the protocol family of the created socket:
  - **INET**: For network protocol IPv4
  - **INET6**: For network protocol IPv6
- **NewSocket**: New connected socket
- **Family**: internet protocol (AF_INET)
- **Port**: Address port
- **Address**: Peer socket address

Table 17. **AT+Bind** Assign a Name to a Socket

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+Bind = [Socket],[Family],[Port],[Address]</td>
<td>OK</td>
</tr>
</tbody>
</table>

Arguments:
- **Socket**: Socket descriptor received from AT+Socket command
- **Family**: Specifies the protocol family of the created socket:
  - **INET**: For network protocol IPv4
  - **INET6**: For network protocol IPv6
- **Port**: Address port
- **Address**: Local socket address

Table 18. **AT+Listen** Listen for Connections on a Socket

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+Listen = [socket],[backlog]</td>
<td>OK</td>
</tr>
</tbody>
</table>

Arguments:
- **socket**: Received from AT+Socket command
- **backlog**: Listen
### Table 19. AT+Connect Initiate a Connection on a Socket

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+Connect = [Socket],[Family],[Port],[Address]</td>
<td>OK</td>
</tr>
<tr>
<td>Arguments:</td>
<td>[Port], [Address]</td>
</tr>
<tr>
<td>• Socket: Received from AT+Socket command</td>
<td></td>
</tr>
<tr>
<td>• Family: internet protocol:</td>
<td></td>
</tr>
<tr>
<td>– INET: For network protocol IPv4</td>
<td></td>
</tr>
<tr>
<td>– INET6: For network protocol IPv6</td>
<td></td>
</tr>
<tr>
<td>• Port: Address port</td>
<td></td>
</tr>
<tr>
<td>• Address: Peer socket address (“x.x.x.x”)</td>
<td></td>
</tr>
</tbody>
</table>

### Table 20. AT+Select Monitor Socket Activity

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+Select = [nfds],[readsds],[timeout sec],[timeout usec]</td>
<td>OK</td>
</tr>
<tr>
<td>Arguments:</td>
<td>[readsds]</td>
</tr>
<tr>
<td>• nfds: The highest-numbered file descriptor in any of the three sets (read, write, and except)</td>
<td>Arguments: Socket descriptors list for read monitoring and accept monitoring</td>
</tr>
<tr>
<td>• readsds: Socket descriptors as bit list (for example, 0</td>
<td>2 for monitoring socket 0 and socket 2)</td>
</tr>
<tr>
<td>• timeout sec: Time in seconds is an upper bound on the amount of time elapsed before select() returns. 0 means return immediately.</td>
<td></td>
</tr>
<tr>
<td>• timeout usec: Time in microseconds</td>
<td></td>
</tr>
</tbody>
</table>
## Table 21. AT+SetSockOpt Set Socket Options

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+SetSockOpt = [sd], [Level], [Option], [Value1], .., [ValueX]</td>
<td>OK</td>
</tr>
</tbody>
</table>

### Arguments:
- sd: Socket descriptor

### Description:

#### Level:
Defines the protocol level for this option

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOCKET</td>
<td>Value1 security method:</td>
</tr>
<tr>
<td>SECMETHOD</td>
<td>Value1 Cipher type:</td>
</tr>
<tr>
<td>KEEPALIVE</td>
<td>Value1:</td>
</tr>
<tr>
<td>RX_NO_IP_BOUNDARY</td>
<td>Value1:</td>
</tr>
<tr>
<td>RCVTIME</td>
<td>Value1:</td>
</tr>
<tr>
<td>RCVBUF</td>
<td>Value1:</td>
</tr>
<tr>
<td>NONBLOCKING</td>
<td>Value1:</td>
</tr>
<tr>
<td>KEEPALIVETIME</td>
<td>Value1:</td>
</tr>
</tbody>
</table>

### Option Details:

- **KEEPALIVE**: Enable or disable periodic keep alive. Keeps TCP connections active by enabling the periodic transmission of messages.
  - Value1: 
    - 1: Enable
    - 0: Disable

- **KEEPALIVETIME**: Set keep alive timeout
  - Value1: Timeout in seconds

- **RX_NO_IP_BOUNDARY**: Enable or disable RX IP boundary
  - Value1: 
    - 1: Enable
    - 0: Disable

- **RCVTIME**: Sets the timeout value that specifies the maximum amount of time an input function waits until it completes
  - Value1: Seconds
  - Value2: Microseconds. 10000 microseconds resolution

- **RCVBUF**: Sets TCP maximum receive window size
  - Value1: Size in bytes

- **NONBLOCKING**: Sets socket to nonblocking
  - Value1: 
    - 1: Enable
    - 0: Disable

- **SECMETHOD**: Sets method to TCP secured socket
  - Value1 security method:
    - SSLV3: Security method SSL v3
    - TLSV1: Security method TLS v1
    - TLSV1_1: Security method TLS v1.1
    - TLSV1_2: Security method TLS v1.2
    - SSLV3_TLSV1_2: Use highest possible version from SSLv3–TLS 1.2

- **SECURE_MASK**: Sets specific ciphers as OR bitmask to TCP secured socket (default value: all ciphers)
  - Value1 Cipher type:
    - SSL_RSA_WITH_RC4_128_SHA
    - SSL_RSA_WITH_RC4_128_MD5
    - TLS_RSA_WITH_AES_256_CBC_SHA
    - TLS_DHE_RSA_WITH_AES_256_CBC_SHA
    - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
    - TLS_ECDHE_RSA_WITH_RC4_128_SHA
    - TLS_RSA_WITH_AES_128_CBC_SHA256
    - TLS_RSA_WITH_AES_256_CBC_SHA256
    - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
    - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
    - TLS_RSA_WITH_AES_128_GCM_SHA256
    - TLS_RSA_WITH_AES_256_GCM_SHA256
    - TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
    - TLS_DHE_RSA_WITH_AES_256_GCM_SHA256
    - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
    - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA256
    - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
    - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA256
    - TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
    - TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
    - TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256
    - TLS_DHE_RSA_WITH_CHACHA20 POLY1305_SHA256
    - TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256
    - TLS_DHE_RSA_WITH_CHACHA20 POLY1305_SHA256
<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SECURE_FILES_CA_FILE_NAME</strong>&lt;br&gt;Map secured socket to CA file by name</td>
<td>Value1: File name</td>
</tr>
<tr>
<td><strong>SECURE_FILES_PRIVATE_KEY_FILE_NAME</strong>&lt;br&gt;Map secured socket to private key by name</td>
<td>Value1: File name</td>
</tr>
<tr>
<td><strong>SECURE_FILES_CERTIFICATE_FILE_NAME</strong>&lt;br&gt;Map secured socket to certificate file by name</td>
<td>Value1: File name</td>
</tr>
<tr>
<td><strong>SECURE_FILES_DH_KEY_FILE_NAME</strong>&lt;br&gt;Map secured socket to Diffie Hellman file by name</td>
<td>Value1: File name</td>
</tr>
<tr>
<td><strong>CHANGE_CHANNEL</strong>&lt;br&gt;Sets channel in transceiver mode</td>
<td>Value1: Channel number (range is 1–13)</td>
</tr>
<tr>
<td><strong>SECURE_ALPN</strong>&lt;br&gt;Sets the ALPN list</td>
<td>Value1: The parameter is a bit map consist of or of the following values: H1</td>
</tr>
</tbody>
</table>
| **LINGER**<br>Socket lingers on close pending remaining send and receive packets | • Value1:  
  – 1: Enable  
  – 0: Disable  
  • Value2: Linger time in seconds |
| **SECURE_EXT_CLIENT_CHLNGL_RESP**<br>Set with no parameter to indicate that the client uses external signature using Network Application request | Value1: Ignore |
| **SECURE_DOMAIN_NAME_VERIFICATION**<br>Set a domain name, to check in SSL client connection | Value1: Domain name |
| **MULTICAST_TTL**<br>Set the time-to-live value of outgoing multicast packets for this socket | Value1: Number of hops |
| **ADD_MEMBERSHIP**<br>UDP socket, join a multicast group | • Value1: IPv4 multicast address to join  
  • Value2: Multicast interface address |
| **DROP_MEMBERSHIP**<br>UDP socket, leave a multicast group | • Value1: IPv4 multicast address to join  
  • Value2: Multicast interface address |
| **RAW_RX_NO_HEADER**<br>Raw socket remove IP header from received data | Value1:  
  • 1: Remove header  
  • 0: Keep header |
| **HDRINCL**<br>RAW socket only, the IPv4 layer generates an IP header when sending a packet unless this option is enabled on the socket | Value1:  
  • 1: Enable  
  • 0: Disable |
| **RAW_IPV6_HDRINCL**<br>RAW socket only, the IPv6 layer generates an IP header when sending a packet unless this option is enabled on the socket | Value1:  
  • 1: Enable  
  • 0: Disable |
Table 21. AT+SetSockOpt Set Socket Options (continued)

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PHY_RATE</strong>&lt;br&gt;Set WLAN PHY transmit rate on RAW socket</td>
<td>Value1: Rate</td>
</tr>
<tr>
<td><strong>PHY_TX_POWER</strong>&lt;br&gt;RAW socket, set WLAN PHY TX power</td>
<td>Value1: Power range is 1–15</td>
</tr>
<tr>
<td><strong>PHY_NUM_FRAMES_TO_TX</strong>&lt;br&gt;RAW socket, set number of frames to transmit in transceiver mode</td>
<td>Value1: Number of frames</td>
</tr>
<tr>
<td><strong>PHY_PREAMBLE</strong>&lt;br&gt;RAW socket, set WLAN PHY preamble for long or short</td>
<td>Value1: Preamble value</td>
</tr>
</tbody>
</table>
| **PHY_TX_INHIBIT_THRESHOLD**<br>RAW socket, set WLAN TX inhibit threshold (CCA), | Value1: Threshold value:  
  • MIN  
  • LOW  
  • DEFAULT  
  • MED  
  • HIGH  
  • MAX |
| **PHY_TX_TIMEOUT**<br>RAW socket, changes the TX timeout (lifetime) of transceiver frames | Value1: Time in milliseconds, maximum value is 10 ms |
| **PHY_ALLOW_ACKS**<br>RAW socket, enable sending ACKs in transceiver mode | Value1:  
  • 1: Enable  
  • 0: Disable |

Table 22. AT+GetSockOpt Get Socket Options

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
</table>
| AT+GetSockOpt = [sd],[level],[option]                                  | +GetSockOpt:  
  [value1]...[valueX]  
  OK |
| Arguments:                                                             | Arguments:  
  value1...valueX (see the AT+SetSockOpt command in Table 21) |
  • sd: Socket handle  
  • level: Defines the protocol level for this option (see Table 21)  
  • option: Defines the option name to interrogate (see Table 21) |

Table 23. AT+Recv Read Data From TCP Socket

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
</table>
| AT+Recv = [sd],[format],[length]                                       | OK  
  +Recv:  
  [sd],[format],[length],[data] |
| Arguments:                                                             | Arguments:  
  [sd],[format],[length],[data] |
  • sd: Socket handle  
  • format: Data format:  
    • 0: Binary data format  
    • 1: Base64 data format (binary to text encoding)  
  • length: Maximum number of bytes to receive |
### Table 24. AT+RecvFrom Read Data From Socket

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+RecvFrom = [sd],[family],[port],[addr],[format],[length]</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>+RecvFrom: [sd],[format],[length],[data]</td>
</tr>
</tbody>
</table>

**Arguments:**
- `sd`: Socket handle
- `family`: internet protocol
  - INET: For network protocol IPv4
  - INET6: For network protocol IPv6
- `port`: Address port (16 bits)
- `addr`: Internet address (32 bits)
- `format`: Data format:
  - 0: Binary data format
  - 1: Base64 data format (binary to text encoding)
- `length`: Maximum number of bytes to receive

### Table 25. AT+Send Write Data to TCP Socket

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+Send = [sd],[format],[length],[data]</td>
<td>OK</td>
</tr>
</tbody>
</table>

**Arguments:**
- `sd`: Socket handle
- `format`: Data format:
  - 0: Binary data format
  - 1: Base64 data format (binary to text encoding)
- `length`: Number of bytes to send
- `data`: Data to send

### Table 26. AT+SendTo Write Data to Socket

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+SendTo = [sd],[family],[port],[addr],[format],[length],[data]</td>
<td>OK</td>
</tr>
</tbody>
</table>

**Arguments:**
- `sd`: Socket handle
- `family`: internet protocol:
  - INET: For network protocol IPv4
  - INET6: For network protocol IPv6
- `port`: Address port (16 bits)
- `addr`: Internet address (32 bits)
- `format`: Data format:
  - 0: Binary data format
  - 1: Base64 data format (binary to text encoding)
- `length`: Maximum number of bytes to receive
- `data`: Data to send
### 6.3 WLAN Commands

#### Table 27. AT+WlanConnect Connect to WLAN Network as a Station

<table>
<thead>
<tr>
<th>Request</th>
<th>Response</th>
</tr>
</thead>
</table>

**Arguments:**
- **SSID**: Name of the Access Point
- **BSSID**: Access Point MAC address (Optional)
- **SecurityType**: Security type:
  - OPEN
  - WEP
  - WEP_SHARED
  - WPA_WPA2
  - WPA2_PLUS
  - WPA3
  - WPA_ENT
  - WPS_PBC
  - WPS_PIN
- **SecurityKey**: Password (Optional in case it is not needed)
- **SecurityExtUser**: Enterprise user name parameters (Ignored in case WPA_ENT was not selected)
- **SecurityExtAnonUser**: Enterprise anonymous user name parameters (Ignored in case WPA_ENT was not selected)
- **SecurityExtEapMethod**: Extensible Authentication Protocol (Ignored in case WPA_ENT was not selected):
  - TLS
  - TTLS_TLS
  - TTLS_MSCHAPv2
  - TTLS_PSK
  - PEAP0_TLS
  - PEAP0_MSCHAPv2
  - PEAP0_PSK
  - PEAP1_TLS
  - PEAP1_PSK

#### Table 28. AT+WlanDisconnect Disconnect the Connection

<table>
<thead>
<tr>
<th>Request</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+WlanDisconnect</td>
<td>OK</td>
</tr>
</tbody>
</table>

**Arguments:**
- none
### Table 29. AT+WlanProfileAdd Add Profile

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
</table>

Arguments:
- **SSID**: Name of the Access Point
- **BSSID**: Access Point MAC address (Optional)
- **SecurityType**: Security type:
  - OPEN
  - WEP
  - WEP_SHARED
  - WPA_WPA2
  - WPA2_PLUS
  - WPA3
  - WPA_ENT
  - WPS_PBC
  - WPS_PIN
- **SecurityKey**: Password (Optional in case it is not needed)
- **SecurityExtUser**: Enterprise user name parameters (Ignored in case WPA_ENT was not selected)
- **SecurityExtAnonUser**: Enterprise anonymous user name parameters (Ignored in case WPA_ENT was not selected)
- **SecurityExtEapMethod**: Extensible Authentication Protocol (Ignored in case WPA_ENT was not selected):
  - TLS
  - TTLS_TLS
  - TTLS_MSCHAPv2
  - TTLS_PSK
  - PEAP0_TLS
  - PEAP0_MSCHAPv2
  - PEAP0_PSK
  - PEAP1_TLS
  - PEAP1_PSK
- **Priority**: Profile priority:
  - Lowest priority: 0
  - Highest priority: 15

### Table 30. AT+WlanProfileGet Get Profile

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
</table>

Arguments: See the AT+WlanProfileAdd command in Table 29.

Arguments: See the AT+WlanProfileAdd command in Table 29.
Table 31. \textit{AT+WlanProfileDel} Delete Profile

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{AT+WlanProfileDel = [index]}</td>
<td>OK</td>
</tr>
</tbody>
</table>

Arguments:
Index: Number of profile to delete received from +WlanProfileAdd
To delete all profiles, use index = 0xFF

Table 32. \textit{AT+WlanPolicySet} Set Policy Values

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{AT+WlanPolicySet = [Type],[Option],[Value]}</td>
<td>OK</td>
</tr>
</tbody>
</table>

Type | Option | Value |
---|---|---|
| \textit{CONNECTION} | Defines options available to connect to the AP (Options could be set as bit masked). No option selected = disable all | \texttt{Auto} Reconnect to one of the stored profiles each time the connection fails or the device is rebooted | Ignore |
| | | \texttt{Fast} Establish a fast connection to AP | Ignore |
| | | \texttt{P2P} Automatically connect to the first P2P device available | Ignore |
| | | \texttt{Auto_Provisioning} Start the provisioning process after a long period of disconnection when profiles exist | Ignore |
| | \textit{SCAN} | Defines system scan time interval. An interval is 10 minutes. After settings scan interval, an immediate scan is activated | \texttt{HiddenSSID} Scan interval in seconds | OK |
| | | | \texttt{No_HiddenSSID} Scan interval in seconds | OK |
| | | | \texttt{DisableScan} Ignore | OK |
| | \textit{PM} | Defines a power management policy for Station mode | \texttt{Normal} Ignore | OK |
| | | | \texttt{Low_Latency} Ignore | OK |
| | | | \texttt{Low_Power} Ignore | OK |
| | | | \texttt{Always_On} Ignore | OK |
| | | | \texttt{Long_Sleep} Maximum sleep time in milliseconds | OK |
| | \textit{P2P} | Defines P2P negotiation policy parameters for P2P role | \texttt{CLIENT} Indicates that the device is forced to be CLIENT | OK |
| | | | \texttt{GROUP_OWNER} Indicates that the device is forced to be P2P GO | OK |
| | | | \texttt{NEGOTIATE} Indicates that the device can be either CLIENT or GO, depending on the Wi-Fi Direct® negotiation tiebreaker | OK |
| | | | \texttt{ACTIVE} When the remote peer is found after the discovery process, the device immediately sends the negotiation request to the peer device. | OK |
| | | | \texttt{PASSIVE} When the remote peer is found after the discovery process, the device passively waits for the peer to start the negotiation, and only responds after. | OK |
| | | | \texttt{RAND_BACKOFF} When the remote peer is found after the discovery process, the device triggers a random timer (from 1 to 6 seconds). During this period, the device passively waits for the peer to start the negotiation. If the timer expires without negotiation, the device immediately sends the negotiation request to the peer device. | OK |
### Table 33. AT+WlanPolicyGet Get Policy Values

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
</table>

**Arguments:**
- **Type:** Type of policy. The options are:
  - **CONNECTION**
    - Get connection policy
  - **SCAN**
    - Get scan policy
  - **PM**
    - Get power management policy
  - **P2P**
    - Get P2P policy

### Table 34. AT+WlanScan Gets the WLAN Scan Operation Results

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+WlanScan = [Index],[Count]</td>
<td>+WlanScan: [SSID],[BSSID],[RSSI],[Channel],[Security_Type],[Hidden_SSID],[Cipher],[Key_Mgmt]; OK</td>
</tr>
</tbody>
</table>

**Arguments:**
- **Index:** Starting index identifier (range 0–29) for getting scan results.
- **Count:** How many entries to fetch; maximum is 30
- **SSID:** Wireless LAN identifier
- **BSSID:** MAC address of the wireless access point
- **Channel**
- **RSSI:** Relative received signal strength in a wireless environment
- **Security_Type:**
  - **OPEN**
  - **WEP**
  - **WPA**
  - **WPA2**
  - **WPA_WPA2**
  - **WPA3**
- **Hidden_SSID:**
  - 1: Hidden
  - 0: Not hidden
- **Cipher:**
  - None
  - WEP40
  - WEP104
  - TKIP
  - CCMP
  - TKIP_CCMP
- **Key_Mgmt:**
  - None
  - 802_1_X
  - PSK
Table 35. \textit{AT+WlanSetMode} WLAN Set Mode

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+WlanSetMode = [Mode]</td>
<td>OK</td>
</tr>
</tbody>
</table>

Arguments:
- Mode: WLAN mode to start the device:
  - STA: For WLAN station mode
  - AP: For WLAN Access Point mode
  - P2P: For WLAN P2P mode

Table 36. \textit{AT+WlanSet} Setting WLAN Configurations

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+WlanSet = [ID],[Option],[Value1]...[ValueX]</td>
<td>OK</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP</td>
<td>SSID</td>
<td>Set SSID for AP mode</td>
</tr>
<tr>
<td></td>
<td>CHANNEL</td>
<td>Set channel for AP mode</td>
</tr>
<tr>
<td></td>
<td>HIDDEN_SSID</td>
<td>Set Hidden SSID Mode for AP mode</td>
</tr>
<tr>
<td></td>
<td>SECURITY</td>
<td>Set Security type for AP mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PASSWORD</td>
<td>Set Password for AP mode (for WEP or for WPA)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MAX_STATIONS</td>
<td>Set Max AP stations</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MAX_STA_AGING</td>
<td>Set Max station aging time</td>
</tr>
<tr>
<td></td>
<td>ACCESS_LIST_MODE</td>
<td>Set AP access list mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ACCESS_LIST_ADD_MAC</td>
<td>Add MAC address to the AP access list</td>
</tr>
<tr>
<td></td>
<td>ACCESS_LIST_DEL_MAC</td>
<td>Delete MAC address from the AP access list</td>
</tr>
<tr>
<td></td>
<td>ACCESS_LIST_DEL_IDX</td>
<td>Delete MAC address from index in the AP access list</td>
</tr>
<tr>
<td>Request:</td>
<td>Response:</td>
<td></td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>COUNTRY_CODE</td>
<td>Set Country Code for AP mode</td>
<td>Two characters country code</td>
</tr>
<tr>
<td>STA_TX_POWER</td>
<td>Set STA mode TX power level</td>
<td>Number between 0–15, as dB offset from maximum power (0 sets maximum power)</td>
</tr>
<tr>
<td>AP_TX_POWER</td>
<td>Set AP mode TX power level</td>
<td>Number between 0–15, as dB offset from maximum power (0 sets maximum power)</td>
</tr>
<tr>
<td>GENERAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INFO_ELEMENT</td>
<td>Set Info Element for AP mode</td>
<td>• Value1: Index of the info element</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Value2: Role:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- AP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- P2P</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Value3: Info element ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Value4: Organization unique ID first Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Value5: Organization unique ID second Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Value6: Organization unique ID third Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Value7: Info element (maximum 252 chars)</td>
</tr>
<tr>
<td>SCAN_PARAMS</td>
<td>Set scan parameters</td>
<td>• Value1: Channel mask</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Value2: RSSI threshold</td>
</tr>
<tr>
<td>SUSPEND_PROFILES</td>
<td>Set suspended profiles mask</td>
<td>Suspended bitmask</td>
</tr>
<tr>
<td>DISABLE_ENT_SERVER_AUTH</td>
<td>This option enables to skip server authentication and is valid for one use, when manually connection to an enterprise network</td>
<td>• 1: Disable the server authentication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: Enable</td>
</tr>
<tr>
<td>P2P</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEV_TYPE</td>
<td>Set P2P Device type</td>
<td>Device type is published under P2P I.E (maximum length of 17 characters)</td>
</tr>
<tr>
<td>CHANNEL_N_REGS</td>
<td>Set P2P Channels</td>
<td>• Value1: Listen channel (either 1/6/11 for 2.4 GHz)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Value2: Listen regulatory class (81 for 2.4 GHz)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Value3: Operating channel (channel 1, 6, or 11 for 2.4 GHz)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Value4: Operating regulatory class (81 for 2.4 GHz)</td>
</tr>
<tr>
<td>RX_FILTER</td>
<td>Enable or disable filters</td>
<td>Filter Bitmap array (16 bytes in format xxx)</td>
</tr>
<tr>
<td>SYS_STATE</td>
<td>Enable or disable system filters</td>
<td>Filter Bitmap array (4 bytes in format xx:xx)</td>
</tr>
<tr>
<td>REMOVE</td>
<td>Remove filters</td>
<td>Filter Bitmap array (16 bytes in format xxx)</td>
</tr>
<tr>
<td>STORE</td>
<td>Save the filters as persistent</td>
<td>null</td>
</tr>
</tbody>
</table>
### Table 37. AT+ WlanGet Getting WLAN Configurations

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+WlanGet = [ID],[Option]</td>
<td>+WlanGet: [Value1],...,[ValueX] OK</td>
</tr>
</tbody>
</table>

**Arguments:**
Arguments: See the AT+WlanSet command in Table 36.

<table>
<thead>
<tr>
<th>ID</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP</td>
<td>SSID Get SSID for AP mode</td>
</tr>
<tr>
<td></td>
<td>CHANNEL Get channel for AP mode</td>
</tr>
<tr>
<td></td>
<td>HIDDEN_SSID Get Hidden SSID Mode for AP mode</td>
</tr>
<tr>
<td></td>
<td>SECURITY Get Security type for AP mode</td>
</tr>
<tr>
<td></td>
<td>PASSWORD Get Password for AP mode (for WEP or for WPA)</td>
</tr>
<tr>
<td></td>
<td>MAX_STATIONS Get Max AP allowed stations</td>
</tr>
<tr>
<td></td>
<td>MAX_STA_AGING Get AP aging time in seconds</td>
</tr>
<tr>
<td></td>
<td>ACCESS_LIST_NUM_ENTRIES Get AP access list number of entries</td>
</tr>
<tr>
<td>ACCESS_LIST</td>
<td>Get the AP access list from start index</td>
</tr>
<tr>
<td></td>
<td>The start index in the access list</td>
</tr>
<tr>
<td>GENERAL</td>
<td>COUNTRY_CODE Get Country Code for AP mode</td>
</tr>
<tr>
<td></td>
<td>STA_TX_POWER Get STA mode TX power level</td>
</tr>
<tr>
<td></td>
<td>AP_TX_POWER Get AP mode TX power level</td>
</tr>
<tr>
<td></td>
<td>SCAN_PARAMS Get scan parameters</td>
</tr>
<tr>
<td>P2P</td>
<td>CHANNEL_N_REGS Get P2P Channels</td>
</tr>
<tr>
<td>RX_FILTER</td>
<td>STATE Retrieves the filters enable/disable status</td>
</tr>
<tr>
<td></td>
<td>SYS_STATE Retrieves the system filters enable or disable status</td>
</tr>
</tbody>
</table>
Table 37. `AT+ WlanGet` Getting WLAN Configurations (continued)

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Connection</strong></td>
<td>Ignore</td>
</tr>
</tbody>
</table>
| • Value1: Role: | - sta  
  - ap  
  - p2p |
| • Value2: Status: | - disconnected  
  - station_connected  
  - p2pcl_connected  
  - p2pgo_connected  
  - ap_connected_stations |
| • Value3: Security: | - open  
  - wep  
  - wpa_wpa2  
  - wps_pbc  
  - wps_pin  
  - wpa_ent  
  - wep_shared |
| • Value4: SSID Name | |
| • Value5: BSSID | |
| • Value6: Device name (relevant to P2P Client only) | |
6.4 File System Commands

Table 38. AT+FileOpen Open File in Storage Device

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
</table>

Arguments:
- Filename: Full path File Name
- Options: Bitmask depend in option:
  - READ: Read a file (no bitmask)
  - WRITE: Open for write for an existing file (optionally bitmask with CREATE)
  - CREATE: Open for creating a new file (optionally bitmask with WRITE or OVERWRITE)
  - OVERWRITE: Opens an existing file (optionally bitmask with CREATE)
  - CREATE_FAILSAFE: Fail safe
  - CREATE_SECURE: Secure file
  - CREATE_NOSIGNATURE: Relevant to secure file only
  - CREATE_STATIC_TOKEN: Relevant to secure file only
  - CREATE_VENDOR_TOKEN: Relevant to secure file only
  - CREATE_PUBLIC_WRITE: Relevant to secure file only, the file can be opened for write without Token
  - CREATE_PUBLIC_READ: Relevant to secure file only, the file can be opened for read without Token
- File size: Maximum file size is defined in bytes (mandatory only for the CREATE option and is ignored for other options)

Table 39. AT+FileClose Close File in Storage Device

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+FileClose = [FileID],[CertificateFileName],[Signature]</td>
<td>OK</td>
</tr>
</tbody>
</table>

Arguments:
- FileID: Assigned from AT+FileOpen
- CertificateFileName: Certificate file with full path (Optional)
- Signature: The signature is SHA-1, the certificate chain may include SHA-256 (Optional)
### Table 40. AT+FileCtl Controls Various File System Operations

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Token</th>
<th>Filename</th>
<th>Data</th>
<th>Token</th>
<th>Output Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESTORE</td>
<td>Ignore</td>
<td>Ignore</td>
<td>FACTORY_IMAGE</td>
<td>Ignore</td>
<td>Ignore</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The system will be back to the production image. FACTORY_DEFAULT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROLLBACK</td>
<td>Token assigned from AT+FileOpen</td>
<td>Filename to rollback</td>
<td>Ignore</td>
<td>New secure token</td>
<td>Ignore</td>
</tr>
<tr>
<td>COMMIT</td>
<td>Token assigned from AT+FileOpen</td>
<td>Filename to commit</td>
<td>Ignore</td>
<td>New secure token</td>
<td>Ignore</td>
</tr>
<tr>
<td>RENAME</td>
<td>Token assigned from AT+FileOpen</td>
<td>Filename to rename</td>
<td>New file name</td>
<td>Ignore</td>
<td>Ignore</td>
</tr>
<tr>
<td>GET_STORAGE_INFO</td>
<td>Ignore</td>
<td>Ignore</td>
<td>Ignore</td>
<td>Ignore</td>
<td>Ignore</td>
</tr>
<tr>
<td>BUNDLE_ROLLBACK</td>
<td>Ignore</td>
<td>Ignore</td>
<td>Ignore</td>
<td>Ignore</td>
<td>Ignore</td>
</tr>
<tr>
<td>BUNDLE_COMMIT</td>
<td>Ignore</td>
<td>Ignore</td>
<td>Ignore</td>
<td>Ignore</td>
<td>Ignore</td>
</tr>
</tbody>
</table>

- **DeviceBlockSize**
- **DeviceBlocks Capacity**
- **NumOfAllocated Blocks**
- **NumOfReserved Blocks**
- **NumOfReserved BlocksFor Systemfiles**
- **LargestAllocated GapInBlocks**
- **NumOfAvailable Blocks**
- **ForUserFiles**
- **MaxFsFiles**
- **IsDevelopment FormatType**
- **Bundlestate**
- **MaxFsFilesReservedForSysFiles**
- **ActualNumOf UserFiles**
- **ActualNumOf SysFiles**
- **NumOfAlerts**
- **NumOfAlerts Threshold**
- **FATWrite Counter**
### Table 41. AT+FileDel Delete File From Storage Device

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+FileDel = [FileName],[SecureToken]</td>
<td>OK</td>
</tr>
</tbody>
</table>

**Arguments:**
- FileName: Full path File Name
- SecureToken: Token assigned from AT+FileOpen (optional)

### Table 42. AT+FileGetFilelist Get a List of Files

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+FileGetFileList</td>
<td>+FileGetFileList: [FileName],[FileMaxSize],[Properties],[FileAllocatedBlocks] OK</td>
</tr>
</tbody>
</table>

**Arguments:**
- FileName: File name
- FileMaxSize: Maximum file size
- Properties: Info flag bitmask
- FileAllocatedBlocks: Allocated blocks

### Table 43. AT+FileGetInfo Get Information About a File

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
</table>

**Arguments:**
- FileName: Full path file name
- SecureToken: token assigned from AT+FileOpen (optional)

### Table 44. AT+FileRead Read a Block of Data From a File in Storage Device

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+FileRead = [FileID],[Offset],[Format],[Length]</td>
<td>+FileRead:[format],[NumberOfReadBytes],[ReceivedData] OK</td>
</tr>
</tbody>
</table>

**Arguments:**
- FileID: Assigned from AT+FileOpen
- Offset: Offset to specific read block
- Format: Data format:
  - 0: Binary data format
  - 1: Base64 data format (binary to text encoding)
- Length: Number of bytes to read
Table 45. AT+FileWrite Write Block of Data to a File in Storage Device

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+FileWrite = [FileID],[Offset],[Format],[Length],[Data]</td>
<td>+FileWrite:[NumberOfWrittenBytes] OK</td>
</tr>
</tbody>
</table>

Arguments:
- FileID: Assigned from AT+FileOpen
- Offset: Offset to specific block to be written
- Format: Data format:
  - 0: Binary data format
  - 1: Base64 data format (binary to text encoding)
- Length: Number of bytes to write
- Data: Transmitted data to the storage device
6.5 Network Application Commands

Activate networking applications, such as:
- HTTP Server
- DHCP Server
- Ping
- DNS
- mDNS

Table 46. AT+NetAPPStart Starts a Network Application

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+NetAPPStart = [APP Bitmap]</td>
<td>OK</td>
</tr>
</tbody>
</table>

Arguments:
- APP Bitmap: Application bitmap, could be one or a combination of the following with OR ("|") between them:
  - HTTP_SERVER
  - DHCP_SERVER
  - MDNS
  - DNS_SERVER

Table 47. AT+NetAPPStop Stops a Network Application

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+NetAPPStop = [APP Bitmap]</td>
<td>OK</td>
</tr>
</tbody>
</table>

Arguments:
- APP Bitmap: Application bitmap, could be one or a combination of the following with OR ("|") between them:
  - HTTP_SERVER
  - DHCP_SERVER
  - MDNS
  - DNS_SERVER

Table 48. AT+NetAPPGetHostByName Get Host IP by Name

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+NetAPPGetHostByName = [HostName],[Family]</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>+NetAPPGetHostByName: [HostName],[Host IP address]</td>
</tr>
</tbody>
</table>

Arguments:
- HostName
- Family: Protocol Family:
  - INET: For network protocol IPv4
  - INET6: For network protocol IPv6
### Table 49. AT+NetAPPGetHostByService Get Host IP by Service

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+NetAPPGetHostByService = [ServiceName],[Family]</td>
<td>OK +NetAPPGetHostByService: [ServiceName],[Port],[HostIPAddress],[Text]</td>
</tr>
</tbody>
</table>

**Arguments:**
- **ServiceName:** Service name can be full or partial
- **Family:** Protocol Family:
  - **INET:** For network protocol IPv4
  - **INET6:** For network protocol IPv6

**Arguments:**
- **ServiceName**
- **Port:** Service port
- **HostIPAddress:** Host IP address (IPv4 or IPv6)
- **Text:** Text of the service full or partial

### Table 50. AT+NetAPPSet Setting Network Application Configurations

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+NetAPPSet = [App ID],[Option],[Value1],…,[ValueX]</td>
<td>OK</td>
</tr>
</tbody>
</table>

**Arguments:**
- **App ID**
- **Option**
- **Values**

**DHCP_SERVER**

<table>
<thead>
<tr>
<th>Option</th>
<th>Values</th>
</tr>
</thead>
</table>
| **BASIC** | • Value1: Lease time (in seconds) of the IP Address  
| | • Value2: First IP Address for allocation  
| | • Value3: Last IP Address for allocation |

**HTTP_SERVER**

<table>
<thead>
<tr>
<th>Option</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PRIM_PORT_NUM</strong></td>
<td>Value1: port number</td>
</tr>
<tr>
<td><strong>SECOND_PORT_NUM</strong></td>
<td>Value1: port number</td>
</tr>
</tbody>
</table>
| **SECOND_PORT_EN** | Value1:  
| | • 1: Enable  
| | • 0: Disable |
| **PRIM_PORT_SEC_EN** | Value1:  
| | • 1: Enable  
| | • 0: Disable |
| **PRIV_KEY_FILE** | Value1: File name (maximum length is 96 bytes) |
| **DEV_CERT_FILE** | Value1: File name (maximum length is 96 bytes) |
| **CA_CERT_FILE** | Value1: File name (maximum length is 96 bytes) |
| **TMP_REGISTER_SERVICE** | Value1: Service name for MDNS (maximum length is 80 bytes) |
| **TMP_UNREGISTER_SERVICE** | Value1: Service name for MDNS (maximum length is 80 bytes) |
### Table 50. AT+NetAPPSet Setting Network Application Configurations (continued)

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
</table>
| **CONT_QUERY** | Value1: Event mask:  
- ipp  
- deviceinfo  
- http  
- https  
- workstation  
- guid  
- h323  
- ntp  
- objective  
- rdp  
- remote  
- rtsp  
- sip  
- smb  
- soap  
- ssh  
- telnet  
- tftp  
- xmpp  
- raop |
| **QEVETN_MASK** | Value1: Service name (maximum length is 80 bytes)  
Value1: Event mask:  
- ipp  
- deviceinfo  
- http  
- https  
- workstation  
- guid  
- h323  
- ntp  
- objective  
- rdp  
- remote  
- rtsp  
- sip  
- smb  
- soap  
- ssh  
- telnet  
- tftp  
- xmpp  
- raop |
| **TIMING_PARAMS** | **DEVICE**  
- Value1: Period in ticks (100 ticks = 1 second)  
- Value2: Repetitions  
- Value3: Telescopic factor  
- Value4: Retransmission interval  
- Value5: Maximum period interval  
- Value6: Maximum time  
**URN** | Value1: device name (maximum length is 33 bytes)  
**DOMAIN** | Value1: domain name (maximum length is 63 bytes)  
**DNS_CLIENT** | **TIME**  
- Value1: Maximum response time in milliseconds  
- Value2: Number of retries |
### Table 51. AT+NetAPPGet Getting Network Applications Configurations

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Arguments:</td>
<td>Arguments: See AT+NetAPPSet command values</td>
</tr>
<tr>
<td>App ID</td>
<td>Option</td>
</tr>
<tr>
<td>DHCP_SERVER</td>
<td>BASIC</td>
</tr>
<tr>
<td>HTTP_SERVER</td>
<td>PRIM_PORT_NUM, AUTH_CHECK, AUTH_NAME, AUTH_PASSWORD, AUTH_REALM, ROM_PAGES_ACCESS, SECOND_PORT_NUM, SECOND_PORT_EN, PRIM_PORT_SEC_EN</td>
</tr>
<tr>
<td>MDNS</td>
<td>CONT_QUERY, QEVETN_MASK, TIMING_PARAMS</td>
</tr>
<tr>
<td>DEVICE</td>
<td>URN, DOMAIN</td>
</tr>
<tr>
<td>DNS_CLIENT</td>
<td>TIME</td>
</tr>
</tbody>
</table>

### Table 52. AT+NetAPPSend Sends Network Application Response or Data Following a Network Application Request Event

<table>
<thead>
<tr>
<th>Request: AT+NetAPPSend = [Handle],[Flags],[Format],[Length],[Data]</th>
<th>Response: OK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arguments:</td>
<td></td>
</tr>
<tr>
<td>• Handle: Handle to send the data to. Should match the handle received in the Network Application request event</td>
<td></td>
</tr>
<tr>
<td>• Flags: Bitmask:</td>
<td></td>
</tr>
<tr>
<td>– CONTINUATION: More data will arrive in subsequent calls to AT+NetAPPSend</td>
<td></td>
</tr>
<tr>
<td>– METADATA: Define data as metadata, otherwise data is payload</td>
<td></td>
</tr>
<tr>
<td>– ACCUMULATION: The network processor should accumulate the data chunks and will process it when it is completely received</td>
<td></td>
</tr>
<tr>
<td>• Format: Data format:</td>
<td></td>
</tr>
<tr>
<td>– 0: Binary data format</td>
<td></td>
</tr>
<tr>
<td>– 1: Base64 data format (binary to text encoding)</td>
<td></td>
</tr>
<tr>
<td>• Length: Number of bytes to send</td>
<td></td>
</tr>
<tr>
<td>• Data: Data to send. Can be just data payload or metadata (depends on flags)</td>
<td></td>
</tr>
</tbody>
</table>
Table 53. **AT+NetAPPRecv** Receives Data From the Network Processor Following a Network Application Response Event

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+NetAPPRecv = [Handle],[Format],[Length]</td>
<td>OK +NetAPPRecv:[Handle],[Flags],[Format],[Length],[Data]</td>
</tr>
</tbody>
</table>

Arguments:
- **Handle**: Handle to receive data from. Should match the handle receive in the Network Application request event.
- **Format**: Data format:
  - 0: Binary data format
  - 1: Base64 data format (binary to text encoding)
- **Length**: Number of bytes to receive.

Table 54. **AT+NetAPPPing** Send Ping to Network Hosts

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+NetAPP Ping = [Family],[Destination],[Size],[Delay],[Timeout],[Max],[Flags]</td>
<td>OK +NetAPP Ping: [PacketsSent],[PacketsReceived],[RoundTime]</td>
</tr>
</tbody>
</table>

Arguments:
- **Family**:
  - **INET**: For network protocol IPv4
  - **INET6**: For network protocol IPv6
- **Destination**: Destination IP address. For stopping an ongoing ping activity, set destination to 0.
- **Size**: Size of ping, in bytes
- **Delay**: Delay between pings, in milliseconds
- **Timeout**: Timeout for every ping in milliseconds
- **Max**: Maximum number of ping requests
  - 0: Forever
- **Flags**:
  - Set to 0: Ping reports back once all requested pings are done
  - Set to 1: Ping reports back after every ping
  - Set to 2: Ping stops after the first successful ping and reports back for the successful ping, as well as any preceding failed pings.
## Table 55. AT+NetAPPGetServiceList Get Service List

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+NetAPPGetServiceList = [IndexOffset],[MaxServiceCount],[Flags]</td>
<td>+NetAPPGetServiceList:[ServiceInfo1];...:[ServiceInfoX] OK</td>
</tr>
</tbody>
</table>

**Arguments:**
- **IndexOffset:** The start index in the peer cache that from it the first service is returned
- **MaxServiceCount:** The maximum services that can be returned if existed or if not exceed the maximum index in the peer cache
- **Flags:** Which service to use (means which types of service to fill):
  - FULL_IPV4_WITH_TEXT
  - FULL_IPV4
  - SHORT_IPV4
  - FULL_IPV6_WITH_TEXT
  - FULL_IPV6
  - SHORT_IPV6

**Response:**
- Arguments:
  - ServiceInfo: Depends on flag type:
    - SHORT_IPV4
      - ip
      - port
    - FULL_IPV4
      - ip
      - port
      - service name
      - service host name
    - FULL_IPV6_WITH_TEXT
      - ip
      - port
      - service name
      - service host name
      - service text

## Table 56. AT+NetAPPRegisterService Register a New mDNS Service

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+NetAPPRegisterService = [ServiceName],[Text],[Port],[TTL],[Options]</td>
<td>OK</td>
</tr>
</tbody>
</table>

**Arguments:**
- **ServiceName:** The service name
- **Text:** The description of the service
- **Port:** The port on this target host port
- **TTL:** The TTL of the service
- **Options:** Bitwise parameters:
  - **IS_UNIQUE_BIT:** Service is unique per interface (means that the service needs to be unique)
  - **IPV6_IPV4_SERVICE:** Add this service to IPv6 interface, if exist (default is IPv4 service only)
  - **IPV6_ONLY_SERVICE:** Add this service to IPv6 interface, but remove it from IPv4 (only IPv6 is available)
  - **UPDATE_TEXT:** For update text fields (without reregistering the service)
  - **IS_NOT_PERSISTENT:** For setting a nonpersistent service
## Table 57. AT+NetAPPUnRegisterService Unregister mDNS Service

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+NetAPPUnRegisterService = [ServiceName],[Options]</td>
<td>OK</td>
</tr>
</tbody>
</table>

Arguments:
- **ServiceName**: Full service name
- **Options**: Bitwise parameters:
  - `IS_UNIQUE_BIT`: Service is unique per interface (means that the service needs to be unique)
  - `IPV6_IPV4_SERVICE`: Add this service to IPv6 interface, if exist (default is IPv4 service only)
  - `IPV6_ONLY_SERVICE`: Add this service to IPv6 interface, but remove it from IPv4 (only IPv6 is available)
  - `UPDATE_TEXT`: For update text fields (without reregistering the service)
  - `IS_NOT_PERSISTENT`: For setting a nonpersistent service
### 6.6 Network Configuration Commands

The Network Configuration Commands control the configuration of the device addresses (that is, IP and MAC addresses).

#### Table 58. AT+NetCfgSet Setting Network Configurations

<table>
<thead>
<tr>
<th>Request: AT+NetCfgSet = [ConfigId],[ConfigOpt],[Value1],...,[ValueX]</th>
<th>Response: OK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arguments:</td>
<td></td>
</tr>
<tr>
<td><strong>ConfigId</strong></td>
<td><strong>ConfigOpt</strong></td>
</tr>
<tr>
<td><strong>IF</strong></td>
<td>STATE</td>
</tr>
<tr>
<td><strong>SET_MAC_ADDR</strong></td>
<td>Static</td>
</tr>
<tr>
<td><strong>IPV4_STA_ADDR</strong></td>
<td>Static</td>
</tr>
<tr>
<td><strong>IPV4_AP_ADDR</strong></td>
<td>Static</td>
</tr>
</tbody>
</table>

**Arguments:**

- **ConfigId**
- **ConfigOpt**
- **Value**

**IF**

- **STATE**
  - Enable or disable modes (bitmask)

**SET_MAC_ADDR**

- **STATIC**
  - Setting MAC address to the Device
- **DHCP**
  - Setting IP address by DHCP
- **DHCP_LLA**
  - Setting DHCP LLA

**IPV4_STA_ADDR**

- **STATIC**
  - Setting a static IP address
- **DHCP**
  - Setting IP address by DHCP
- **DHCP_LLA**
  - Setting DHCP LLA
- **RELEASE_IP_SET**
  - Setting release IP before disconnect enables sending a DHCP release frame to the server
- **RELEASE_IP_OFF**
  - Setting release IP before disconnect disables sending a DHCP release frame to the server

**IPV4_AP_ADDR**

- **STATIC**
  - Setting a static IP address to the device working in AP mode
Table 58. AT+NetCfgSet Setting Network Configurations (continued)

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPV6_ADDR_LOCAL</td>
<td></td>
</tr>
<tr>
<td>Static</td>
<td>Setting a IPv6 Local static IP address</td>
</tr>
<tr>
<td>Stateless</td>
<td>Setting a IPv6 Local stateless address</td>
</tr>
<tr>
<td>Stateful</td>
<td>Setting a IPv6 Local stateful address</td>
</tr>
<tr>
<td>IPV6_ADDR_GLOBAL</td>
<td></td>
</tr>
<tr>
<td>Static</td>
<td>Setting a IPv6 Global static IP address</td>
</tr>
<tr>
<td>Value1 : IP address</td>
<td></td>
</tr>
<tr>
<td>Value2: DNS Server IP</td>
<td></td>
</tr>
<tr>
<td>Stateful</td>
<td>Setting a IPv6 Global stateful address</td>
</tr>
<tr>
<td>AP_STATION_DISCONNECT</td>
<td>Disconnect AP station by MAC address</td>
</tr>
<tr>
<td>Ignore value</td>
<td>AP MAC address</td>
</tr>
<tr>
<td>IPV4_DNS_CLIENT</td>
<td>Set secondary DNS address</td>
</tr>
<tr>
<td>Ignore value</td>
<td>Secondary DNS Server address</td>
</tr>
</tbody>
</table>

Table 59. AT+NetCfgGet Getting Network Configurations

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+NetCfgGet = [ConfigId]</td>
<td>+NetCfgGet:[Value1],..,[ValueX]</td>
</tr>
<tr>
<td>Arguments: Configld: Configuration ID:</td>
<td>Arguments:</td>
</tr>
<tr>
<td>GET_MAC_ADDR</td>
<td>Get the device MAC address</td>
</tr>
<tr>
<td>Value1: MAC address</td>
<td></td>
</tr>
<tr>
<td>IPV4_STA_ADDR</td>
<td>Get IP address from WLAN station or P2P client</td>
</tr>
<tr>
<td>Value1: Address option:</td>
<td></td>
</tr>
<tr>
<td>DHCP</td>
<td></td>
</tr>
<tr>
<td>DHCP_LLA</td>
<td></td>
</tr>
<tr>
<td>STATIC</td>
<td></td>
</tr>
<tr>
<td>Value2: Address</td>
<td></td>
</tr>
<tr>
<td>Value3: Subnet mask</td>
<td></td>
</tr>
<tr>
<td>Value4: Gateway</td>
<td></td>
</tr>
<tr>
<td>Value5: DNS</td>
<td></td>
</tr>
<tr>
<td>IPV4_AP_ADDR</td>
<td>Get static IP address for AP or P2P go</td>
</tr>
<tr>
<td>Value1: State (bitmask):</td>
<td></td>
</tr>
<tr>
<td>ipv6_sta_local</td>
<td></td>
</tr>
<tr>
<td>ipv6_sta_global</td>
<td></td>
</tr>
<tr>
<td>disable_ipv4_dhcp</td>
<td></td>
</tr>
<tr>
<td>ipv6_local_static</td>
<td></td>
</tr>
<tr>
<td>ipv6_local_stateless</td>
<td></td>
</tr>
<tr>
<td>ipv6_local_stateful</td>
<td></td>
</tr>
<tr>
<td>ipv6_global-static</td>
<td></td>
</tr>
<tr>
<td>ipv6_global-stateless</td>
<td></td>
</tr>
<tr>
<td>disable_ipv4_lla</td>
<td></td>
</tr>
<tr>
<td>enable_dhcp_release</td>
<td></td>
</tr>
<tr>
<td>ipv6_global-stateless</td>
<td></td>
</tr>
<tr>
<td>disable_fast_renew</td>
<td></td>
</tr>
</tbody>
</table>
Table 59. AT+NetCfgGet Getting Network Configurations (continued)

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPV6_ADDR_LOCAL</td>
<td>• Value1: Address option:</td>
</tr>
<tr>
<td>Get IPV6 Local address</td>
<td>– stateless</td>
</tr>
<tr>
<td>IPV6_ADDR_GLOBAL</td>
<td>– stateful</td>
</tr>
<tr>
<td>Get IPV6 Global address</td>
<td>– STATIC</td>
</tr>
<tr>
<td>AP_STATIONS_CONNECTED</td>
<td>• Value2: Address</td>
</tr>
<tr>
<td>Get AP number of connected stations</td>
<td>Value1: Number of connected stations</td>
</tr>
<tr>
<td>AP_STATIONS_INFO</td>
<td>[address1],[MAC address1],[name1]; ...; [addressX],[MAC addressX],[nameX]</td>
</tr>
<tr>
<td>IPV4_DNS_CLIENT</td>
<td>Value1: DNS second server address</td>
</tr>
<tr>
<td>Set secondary DNS address</td>
<td>• Value1: Address</td>
</tr>
<tr>
<td>IPV4_DHCP_CLIENT</td>
<td>• Value2: Subnet mask</td>
</tr>
<tr>
<td>Get DHCP Client info</td>
<td>• Value3: Gateway</td>
</tr>
<tr>
<td>IPV4_DHCP_CLIENT</td>
<td>• Value4: DNS 1</td>
</tr>
<tr>
<td>Get DHCP Client info</td>
<td>• Value5: DNS 2</td>
</tr>
<tr>
<td></td>
<td>• Value6: DHCP server</td>
</tr>
<tr>
<td></td>
<td>• Value7: Lease time</td>
</tr>
<tr>
<td></td>
<td>• Value8: Time to renew</td>
</tr>
<tr>
<td></td>
<td>• Value9: DHCP State:</td>
</tr>
<tr>
<td></td>
<td>– unknown</td>
</tr>
<tr>
<td></td>
<td>– disabled</td>
</tr>
<tr>
<td></td>
<td>– enabled</td>
</tr>
<tr>
<td></td>
<td>– bound</td>
</tr>
<tr>
<td></td>
<td>– renew</td>
</tr>
<tr>
<td></td>
<td>– rebind</td>
</tr>
</tbody>
</table>
6.7 Network Utility Commands

Networking related commands and configuration.

Table 60. AT+NetUtilGet Getting Utilities Configurations

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+NetUtilGet = [ID],[Option]</td>
<td>+NetUtilGet: [Value1],...,[ValueX] OK</td>
</tr>
</tbody>
</table>

Arguments:

<table>
<thead>
<tr>
<th>ID Identifier of the specific “get” operation to perform</th>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>public_key</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Binary data format</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Base64 data format (binary to text encoding)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Value1: Public key format:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Binary data format</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Base64 data format</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Value2: Public key length (maximum length is 255 bytes or 370 bytes in base64 format)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Value3: Public key</td>
<td></td>
</tr>
<tr>
<td>true_random</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of random numbers (maximum is 172 numbers)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>List of random numbers</td>
<td></td>
</tr>
</tbody>
</table>
## Table 61. AT+NetUtilCmd Performing Utilities-Related Commands

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+NetUtilCmd = [Cmd],[Value1],..,[ValueX]</td>
<td>+NetUtilCmd:[Value1],..,[ValueX] OK</td>
</tr>
</tbody>
</table>

### Arguments:

<table>
<thead>
<tr>
<th>Cmd</th>
<th>Identifier of the specific command to perform</th>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>sign_msg</strong></td>
<td>Create a digital signature using the ECDSA algorithm</td>
<td>• Value1: Key index:</td>
<td>• Value1: Signature format:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Value2: Data format:</td>
<td>• Value2: 0: Binary data format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– 0: Binary data format</td>
<td>– 1: Base64 data format (binary to text encoding)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– 1: Base64 data format (binary to text encoding)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Value3: Data length (maximum length is 1500 bytes)</td>
<td>• Value2: Signature length (maximum length is 255 bytes)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Value4: Data</td>
<td>• Value3: Signature</td>
</tr>
<tr>
<td><strong>verify_msg</strong></td>
<td>verify a digital signature using the ECDSA algorithm</td>
<td>• Value1: Key index</td>
<td>Value1: Success or failure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Value2: Data and signature format:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>– 0: Binary data format</td>
<td>• Value2: 0: Binary data format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– 1: Base64 data format (binary to text encoding)</td>
<td>– 1: Base64 data format (binary to text encoding)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Value3: Data length (maximum length is 1500 bytes)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Value4: Signature length</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Value5: Data and signature (signature concatenate to end of data)</td>
<td></td>
</tr>
<tr>
<td><strong>temp_keys</strong></td>
<td>Create or remove a temporary ECC key pair with the SECP256R1 curve</td>
<td>• Value1: Key index</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Value2: Action:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>– create</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>– remove</td>
<td></td>
</tr>
<tr>
<td><strong>install_op</strong></td>
<td>Install or uninstall a key pair in one of the crypto utilities key pair management mechanism</td>
<td>• Value1: Key index</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Value2: Action:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>– install</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>– uninstall</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Value3: Key Algorithm (ignored for uninstall action):</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>– none</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>– ec</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Value4: EC Named Curve identifier (optional for Key Algorithm none) (ignored for uninstall action):</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>– none</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>– secp256r1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Value5: Certification file name (ignored for uninstall action)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Value6: Key file name (ignored for uninstall action)</td>
<td></td>
</tr>
</tbody>
</table>
## 6.8 Asynchronous Events

### Table 62. +EventFatalError Fatal Error Event for Inspecting Fatal Event

**Response:**

```plaintext
+EventFatalError:[EventID],[Value1],...,[ValueX]
```

**Arguments:**

<table>
<thead>
<tr>
<th>EventID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEVICE_ABORT</td>
<td>Indicates a severe error occurred and the device stopped</td>
</tr>
<tr>
<td>NO_CMD_ACK</td>
<td>Indicates that the command sent to the device had no ACK</td>
</tr>
<tr>
<td>CMD_TIMEOUT</td>
<td>Indicates that the command got a timeout while waiting for its asynchronous response</td>
</tr>
<tr>
<td>DRIVER_ABORT</td>
<td>Indicates a severe error occurred in the driver</td>
</tr>
<tr>
<td>SYNC_LOSS</td>
<td>Indicates a sync loss with the device</td>
</tr>
</tbody>
</table>

- **DEVICE_ABORT**
  - Value1: An indication of the abort type
  - Value2: The abort data
- **NO_CMD_ACK**
  - Value1: An indication of the CMD opcode
- **CMD_TIMEOUT**
  - Value1: An indication of the asynchronous event opcode
- **DRIVER_ABORT**
  - null
- **SYNC_LOSS**
  - null

### Table 63. +EventGeneral General Asynchronous Event for Inspecting General Events

**Response:**

```plaintext
+EventGeneral:[EventID],[Value1],...,[ValueX]
```

**Arguments:**

<table>
<thead>
<tr>
<th>EventID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESET_REQUEST</td>
<td>• Value1: An error code indication from the device</td>
</tr>
<tr>
<td></td>
<td>• Value2: The sender originator:</td>
</tr>
<tr>
<td></td>
<td>- WLAN</td>
</tr>
<tr>
<td></td>
<td>- NETCFG</td>
</tr>
<tr>
<td></td>
<td>- NETAPP</td>
</tr>
<tr>
<td></td>
<td>- SECURITY</td>
</tr>
<tr>
<td></td>
<td>- OTHER</td>
</tr>
<tr>
<td>ERROR</td>
<td>• Value1: An error code indication from the device</td>
</tr>
<tr>
<td></td>
<td>• Value2: The sender originator</td>
</tr>
</tbody>
</table>
Table 64. +EventWlan WLAN Asynchronous Event

<table>
<thead>
<tr>
<th>Response</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>+EventWlan:[EventID],[Value1],...,[ValueX]</td>
<td></td>
</tr>
</tbody>
</table>

Arguments:

<table>
<thead>
<tr>
<th>EventID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONNECT</td>
<td>Value1: SSID name</td>
</tr>
<tr>
<td>STA connection indication event</td>
<td>Value2: BSSID</td>
</tr>
<tr>
<td>P2P_CONNECT</td>
<td>Value1: SSID name</td>
</tr>
<tr>
<td>P2P client connection indication event</td>
<td>Value2: BSSID</td>
</tr>
<tr>
<td>P2P_DISCONNECT</td>
<td>Value1: SSID name</td>
</tr>
<tr>
<td>STA client disconnection event</td>
<td>Value2: BSSID</td>
</tr>
<tr>
<td>STA_ADDED</td>
<td>Value1: MAC address</td>
</tr>
<tr>
<td>AP connected STA</td>
<td></td>
</tr>
<tr>
<td>P2P_CLIENT_ADDED</td>
<td>Value1: MAC address</td>
</tr>
<tr>
<td>P2P(Go) connected P2P(Client)</td>
<td>Value2: Go Device Name</td>
</tr>
<tr>
<td>P2P_CLIENT_REMOVED</td>
<td>Value1: MAC address</td>
</tr>
<tr>
<td>P2P(Go) disconnected P2P(Client)</td>
<td>Value2: Go Device Name</td>
</tr>
<tr>
<td>P2P_DEVFOUND</td>
<td>Value1: Go Device Name</td>
</tr>
<tr>
<td>P2P_REQUEST</td>
<td>Value1: MAC address</td>
</tr>
<tr>
<td>P2P_CONNECTFAIL</td>
<td>Value2: MAC address</td>
</tr>
<tr>
<td>P2P only</td>
<td>Value3: WPS Method</td>
</tr>
<tr>
<td>PROVISIONING_STATUS</td>
<td>Value1: Status</td>
</tr>
<tr>
<td>PROVISIONING_PROFILE_ADDED</td>
<td>Value1: Status</td>
</tr>
<tr>
<td>PROVISIONING_PROFILE_ADDED</td>
<td>Value2: SSID name</td>
</tr>
</tbody>
</table>
### Table 65. +EventNetApp Network Application Asynchronous Event

<table>
<thead>
<tr>
<th>EventID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPV4_ACQUIRED</td>
<td>• Value1: IP address</td>
</tr>
<tr>
<td></td>
<td>• Value2: Gateway</td>
</tr>
<tr>
<td></td>
<td>• Value3: DNS</td>
</tr>
<tr>
<td>IPV6_ACQUIRED</td>
<td>• Value1: IP address</td>
</tr>
<tr>
<td></td>
<td>• Value2: DNS</td>
</tr>
<tr>
<td>ip_collision</td>
<td>• Value1: IP address</td>
</tr>
<tr>
<td></td>
<td>• Value2: DHCP MAC</td>
</tr>
<tr>
<td></td>
<td>• Value3: DNS</td>
</tr>
<tr>
<td>IPLEASED</td>
<td>• Value1: IP address</td>
</tr>
<tr>
<td></td>
<td>• Value2: Lease time</td>
</tr>
<tr>
<td></td>
<td>• Value3: MAC</td>
</tr>
<tr>
<td>IP_RELEASED</td>
<td>• Value1: IP address</td>
</tr>
<tr>
<td></td>
<td>• Value2: MAC</td>
</tr>
<tr>
<td></td>
<td>• Value3: Reason</td>
</tr>
<tr>
<td>IPV4_LOST</td>
<td>Value1: Status</td>
</tr>
<tr>
<td>dhcp_ipv4_acquire_timeout</td>
<td>Value1: Status</td>
</tr>
<tr>
<td>IPV6_LOST</td>
<td>Value1: IP lost</td>
</tr>
</tbody>
</table>

### Table 66. +EventSock Socket Asynchronous Event

<table>
<thead>
<tr>
<th>EventID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX_FAILED</td>
<td>• Value1: sd</td>
</tr>
<tr>
<td></td>
<td>• Value2: Status</td>
</tr>
<tr>
<td>ASYNC_EVENT</td>
<td>• Value1: sd</td>
</tr>
<tr>
<td></td>
<td>• Value2: Type:</td>
</tr>
<tr>
<td></td>
<td>− SSL_ACCEPT</td>
</tr>
<tr>
<td></td>
<td>− RX_FRAG_TOO_BIG</td>
</tr>
<tr>
<td></td>
<td>− OTHER_SIDE_CLOSE_SSL</td>
</tr>
<tr>
<td></td>
<td>− CONNECTED_SECURED</td>
</tr>
<tr>
<td></td>
<td>− WRONG_ROOT_CA</td>
</tr>
<tr>
<td></td>
<td>• Value3: Error value</td>
</tr>
</tbody>
</table>
Table 67. +EventMqtt MQTT Asynchronous Event

<table>
<thead>
<tr>
<th>Response:</th>
<th>+EventMqtt[EVENTID],[Value1],...,[ValueX]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arguments:</td>
<td></td>
</tr>
<tr>
<td>EventID</td>
<td>Value</td>
</tr>
<tr>
<td></td>
<td>• Value1: operation ID:</td>
</tr>
<tr>
<td></td>
<td>-- Connack: connection acknowledge</td>
</tr>
<tr>
<td></td>
<td>-- Value2: 16 bits:</td>
</tr>
<tr>
<td></td>
<td>-- 8 MSBs: Acknowledge Flags</td>
</tr>
<tr>
<td></td>
<td>-- 8 LSBs: return code:</td>
</tr>
<tr>
<td></td>
<td>-- 0: Connection Accepted</td>
</tr>
<tr>
<td></td>
<td>-- 1: Connection Refused, unacceptable protocol version</td>
</tr>
<tr>
<td></td>
<td>-- 2: Connection Refused, identifier rejected</td>
</tr>
<tr>
<td></td>
<td>-- 3: Connection Refused, Server unavailable</td>
</tr>
<tr>
<td></td>
<td>-- 4: Connection Refused, bad user name or password</td>
</tr>
<tr>
<td></td>
<td>-- 5: Connection Refused, not authorized</td>
</tr>
<tr>
<td></td>
<td>-- Puback: publish acknowledge</td>
</tr>
<tr>
<td></td>
<td>-- Value2: Packet Identifier from the PUBLISH Packet that is being acknowledged</td>
</tr>
<tr>
<td></td>
<td>-- Suback: subscribe acknowledge</td>
</tr>
<tr>
<td></td>
<td>-- Value2: Packet Identifier from the SUBSCRIBE Packet that is being acknowledged</td>
</tr>
<tr>
<td></td>
<td>-- Value3 to ValueX: return code per topic:</td>
</tr>
<tr>
<td></td>
<td>-- 0: Success, Maximum QoS 0</td>
</tr>
<tr>
<td></td>
<td>-- 1: Success, Maximum QoS 1</td>
</tr>
<tr>
<td></td>
<td>-- 2: Success, Maximum QoS 2</td>
</tr>
<tr>
<td></td>
<td>-- 128: Failure</td>
</tr>
<tr>
<td></td>
<td>-- Unsuback: unsubscribe acknowledge</td>
</tr>
<tr>
<td></td>
<td>-- Value2: Packet Identifier from the UNSUBSCRIBE Packet that is being acknowledged</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Topic: topic string</td>
</tr>
<tr>
<td></td>
<td>• QoS: Quality of service type:</td>
</tr>
<tr>
<td></td>
<td>-- QoS 0</td>
</tr>
<tr>
<td></td>
<td>-- QoS 1</td>
</tr>
<tr>
<td></td>
<td>-- QoS 2</td>
</tr>
<tr>
<td></td>
<td>• Retain:</td>
</tr>
<tr>
<td></td>
<td>-- 0: message should not be retained</td>
</tr>
<tr>
<td></td>
<td>-- 1: message should be retained</td>
</tr>
<tr>
<td></td>
<td>• Duplicate:</td>
</tr>
<tr>
<td></td>
<td>-- 0: first attempted to send the message</td>
</tr>
<tr>
<td></td>
<td>-- 1: might be re-delivery of an earlier attempt to send the message</td>
</tr>
<tr>
<td></td>
<td>• Message Format:</td>
</tr>
<tr>
<td></td>
<td>-- 0: Binary data format</td>
</tr>
<tr>
<td></td>
<td>-- 1: Base64 data format (binary to text encoding)</td>
</tr>
<tr>
<td></td>
<td>• Message length: number of bytes to send</td>
</tr>
<tr>
<td></td>
<td>• Message: message to send</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• recv</td>
</tr>
</tbody>
</table>
6.9 MQTT Client Commands

MQTT client commands and configuration.

Table 68. AT+MqttCreate MQTT Client Create

<table>
<thead>
<tr>
<th>Request</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+MqttCreate = [client ID],[flags],[address],[port],[method],[cipher],[private key],[Certificate],[CA],[DH key],[protocol],[blocking send],[data format]</td>
<td>+ MqttCreate: [index] OK</td>
</tr>
</tbody>
</table>

Arguments:

- **client ID**
- **flags**: bitmask of the following:
  - **ip4**: IPv4 connection
  - **ip6**: IPv6 connection
  - **url**: Server address is an URL and not IP address
  - **sec**: Connection to server must be secure (TLS)
  - **skip_domain_verify**: skip domain name verification
  - **skip_cert_verify**: skip certificate catalog verification
  - **skip_date_verify**: skip date verification
- **address**: server address (ip or url)
- **port**: address port (16 bits)
- **method**: security method (mandatory only in case of secure connection):
  - **SSLV3**: Security method SSL v3
  - **TLSV1**: Security method TLS v1
  - **TLSV1_1**: Security method TLS v1_1
  - **TLSV1_2**: Security method TLS v1_2
  - **SSLV3_TLSV1_2**: Use highest possible version from SSLv3–TLS 1.2
- **cipher**: security cipher as OR bitmask (optional), (default value: all ciphers):
  - SSL_RSA_WITH_RC4_128_SHA
  - SSL_RSA_WITH_RC4_128_MD5
  - TLS_RSA_WITH_AES_256_CBC_SHA
  - TLS_DHE_RSA_WITH_AES_256_CBC_SHA
  - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
  - TLS_ECDHE_RSA_WITH_RC4_128_SHA
  - TLS_RSA_WITH_AES_128_CBC_SHA256
  - TLS_RSA_WITH_AES_256_CBC_SHA256
  - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
  - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA256
  - TLS_RSA_WITH_AES_128_GCM_SHA256
  - TLS_RSA_WITH_AES_256_GCM_SHA256
  - TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
  - TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
  - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
  - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA384
  - TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
  - TLS_RSA_WITH_AES_128_GCM_SHA256
  - TLS_RSA_WITH_AES_256_GCM_SHA256
  - TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
  - TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
  - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
  - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
  - TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
  - TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
  - TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256

index: client handle
### Table 68. AT+MqttCreate MQTT Client Create (continued)

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• private key: private key file name (Optional)</td>
<td></td>
</tr>
<tr>
<td>• certificate: certificate file name (Optional)</td>
<td></td>
</tr>
<tr>
<td>• CA: certificate authority file name (mandatory only in case of secure connection)</td>
<td></td>
</tr>
<tr>
<td>• DH key: Diffie Hellman file name (Optional)</td>
<td></td>
</tr>
<tr>
<td>• protocol: MQTT protocol:</td>
<td>index: client handle</td>
</tr>
<tr>
<td>- v3_1: protocol v3.1</td>
<td></td>
</tr>
<tr>
<td>- v3_1_1: protocol v3.1.1</td>
<td></td>
</tr>
<tr>
<td>• blocking send:</td>
<td></td>
</tr>
<tr>
<td>- 0: do not wait for server response</td>
<td></td>
</tr>
<tr>
<td>- 1: wait for response</td>
<td></td>
</tr>
<tr>
<td>• data format: set format globally to all MQTT commands and events:</td>
<td></td>
</tr>
<tr>
<td>- 0: Binary data format</td>
<td></td>
</tr>
<tr>
<td>- 1: Base64 data format (binary to text encoding)</td>
<td></td>
</tr>
</tbody>
</table>

### Table 69. AT+MqttDelete MQTT Client Delete

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+MqttDelete = [index]</td>
<td>OK</td>
</tr>
<tr>
<td>Arguments:</td>
<td>Arguments:</td>
</tr>
<tr>
<td>index: client handle received from AT+MqttCreate</td>
<td></td>
</tr>
</tbody>
</table>

### Table 70. AT+MqttConnect MQTT Client Connect to Broker

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+MqttConnect = [index]</td>
<td>OK</td>
</tr>
<tr>
<td>Arguments:</td>
<td>Arguments:</td>
</tr>
<tr>
<td>index: client handle received from AT+MqttCreate</td>
<td></td>
</tr>
</tbody>
</table>

### Table 71. AT+MqttDisconnect MQTT Client Disconnect From Broker

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+MqttDisconnect = [index]</td>
<td>OK</td>
</tr>
<tr>
<td>Arguments:</td>
<td>Arguments:</td>
</tr>
<tr>
<td>index: client handle received from AT+MqttCreate</td>
<td></td>
</tr>
</tbody>
</table>
### Table 72. AT+MqttPublish MQTT Client Send Message to Broker

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+MqttPublish = [index],[topic],[QoS],[retain],[message length],[message]</td>
<td>OK</td>
</tr>
</tbody>
</table>

Arguments:

- **index**: client handle received from AT+MqttCreate
- **topic**: topic string
- **QoS**: Quality of service type:
  - QoS 0
  - QoS 1
  - QoS 2
- **retain**:
  - 0: message should not be retained
  - 1: message should be retained
- **message length**: number of bytes to send
- **message**: message to send in format according to previous configuration in AT+MqttCreate (Data format field)

### Table 73. AT+MqttSubscribe MQTT Client Subscribe for Topic

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+MqttSubscribe = [index],[number of topics],[topic1][QoS1],[persistent1],..., [topicX][QoSX],[persistentX]</td>
<td>OK</td>
</tr>
</tbody>
</table>

Arguments:

- **index**: client handle received from AT+MqttCreate
- **number of topics**: maximum 4 topics
- **topic**: topic string
- **QoS**: Quality of service type:
  - QoS 0
  - QoS 1
  - QoS 2
- **persistent** (optional for future use)

### Table 74. AT+MqttUnsubscribe MQTT Client Unsubscribe for Topic

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+MqttUnsubscribe = [index],[number of topics],[topic1],[persistent1],..., [topicX],[persistentX]</td>
<td>OK</td>
</tr>
</tbody>
</table>

Arguments:

- **index**: client handle received from AT+MqttCreate
- **number of topics**: maximum 4 topics
- **topic**: topic string
- **persistent** (optional for future use)
### Table 75. AT+MqttSet MQTT Client Set Option

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+MqttSet = [index],[option],[value1]...[valueX]</td>
<td>OK</td>
</tr>
</tbody>
</table>

**Arguments:**
- `index`: client handle received from AT+MqttCreate

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>user</code></td>
<td>Value1: User name string</td>
</tr>
<tr>
<td><code>password</code></td>
<td>Value1: Password string</td>
</tr>
</tbody>
</table>
| `will`  | `Value1`: Topic: will topic string  
|         | `Value2`: QoS: Quality of service type:  
|         |   - QoS 0  
|         |   - QoS 1  
|         |   - QoS 2  
|         | `Value3`: Retain:  
|         |   - 0: will message should not be retained  
|         |   - 1: will message should be retained  
|         | `Value4`: Message length: number of bytes contain in will message  
|         | `Value5`: Message: will message to send in format according to previous configuration in AT+MqttCreate (Data format field)  |
| `keepalive` | Value1: keep alive time in seconds (16 bits) |
| `clean` | Value1:  
|         |   - 0: Persistent connection  
|         |   - 1: Enable clean connection  |
6.10 HTTP Client Commands

HTTP client commands and configuration.

Table 76. AT+HttpCreate Http Client Create

<table>
<thead>
<tr>
<th>Request</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+HttpCreate</td>
<td>+HttpCreate: [index] OK</td>
</tr>
<tr>
<td>Arguments:</td>
<td>Arguments:</td>
</tr>
<tr>
<td>index: client handle</td>
<td></td>
</tr>
</tbody>
</table>

Table 77. AT+HttpDestroy Http Client Delete

<table>
<thead>
<tr>
<th>Request</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+HttpDestroy = [index]</td>
<td>OK</td>
</tr>
<tr>
<td>Arguments:</td>
<td>Arguments:</td>
</tr>
<tr>
<td>index: client handle received from At+HttpCreate</td>
<td></td>
</tr>
</tbody>
</table>

Table 78. AT+HttpConnect Http Client Connect to Host

<table>
<thead>
<tr>
<th>Request</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+HttpConnect = [index],[host],[flags],[private key],[certificate],[ca]</td>
<td>OK</td>
</tr>
<tr>
<td>Arguments:</td>
<td>Arguments:</td>
</tr>
<tr>
<td>• index: client handle received from At+HttpCreate</td>
<td></td>
</tr>
<tr>
<td>• host: host name</td>
<td></td>
</tr>
<tr>
<td>• flags: bitmask:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>– ignore_proxy</td>
</tr>
<tr>
<td></td>
<td>– host_exist</td>
</tr>
<tr>
<td>• private key: private key file name (optional)</td>
<td></td>
</tr>
<tr>
<td>• certificate: client certificate file name (optional)</td>
<td></td>
</tr>
<tr>
<td>• ca: root ca file name (optional)</td>
<td></td>
</tr>
</tbody>
</table>

Table 79. AT+HttpDisconnect Http Client Disconnect From Host

<table>
<thead>
<tr>
<th>Request</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+HttpDisconnect = [index]</td>
<td>OK</td>
</tr>
<tr>
<td>Arguments:</td>
<td>Arguments:</td>
</tr>
<tr>
<td>index: client handle received from At+HttpCreate</td>
<td></td>
</tr>
</tbody>
</table>
Table 80. **AT+HttpSendReq** Http Client Send Request to Host

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+HttpSendReq = [index],[method],[uri],[flags],[format],[length],[data]</td>
<td>+HttpSendReq: [status] OK</td>
</tr>
</tbody>
</table>

**Arguments:**

- **index:** client handle received from At+HttpCreate
- **method:**
  - `get`
  - `post`
  - `head`
  - `options`
  - `put`
  - `del`
  - `connect`
- **uri:** request uri string
- **flags:** bitmask:
  - `chunk_start`: Sets the client's request state into chunked body
  - `chunk_end`: Sets the client's request state out of chunked body and sends last chunk
  - `drop_body`: Flashes the response body
- **format:** request data format (mandatory only in case of methods post or put)
  - `0`: Binary data format
  - `1`: Base64 data format (binary to text encoding)
- **length:** length of request data (mandatory only in case of methods post or put)
- **data:** request data (mandatory only in case of methods post or put)

**Status:** case of success status = 200, else failure

Table 81. **AT+HttpReadResBody** Http Client Read Response Body From Host

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+HttpReadResBody = [index],[format],[length]</td>
<td>+HttpReadResBody: [index],[flag],[format],[length],[body] OK</td>
</tr>
</tbody>
</table>

**Arguments:**

- **index:** client handle received from At+HttpCreate
- **format:** request data format
  - `0`: Binary data format
  - `1`: Base64 data format (binary to text encoding)
- **length:** maximum length of body

- **index:** client handle
- **flag:** more data flag
- **format:** request data format
  - `0`: Binary data format
  - `1`: Base64 data format (binary to text encoding)
- **length:** maximum length of body
- **body:** received data
Table 82. *AT+HttpSetHeader* Http Client Set Header

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+HttpSetHeader = [index],[option],[flags],[format],[length],[data]</td>
<td>OK</td>
</tr>
</tbody>
</table>

Arguments:

- **Index:** client handle received from At+HttpCreate
- **Option:**
  - res_age
  - res_allow
  - res_cache_control
  - res_connection
  - res_content_encoding
  - res_content_language
  - res_content_length
  - res_content_location
  - res_content_range
  - res_content_type
  - res_date
  - res_etag
  - res_expires
  - res_last_modified
  - res_location
  - res_proxy_auth
  - res_retry_after
  - res_server
  - res_set_cookie
  - res_trailer
  - res_tx_encoding
  - res_upgrade
  - res_vary
  - res_via
  - res_www_auth
  - res_warning
  - req_accept
  - req_accept_charset
  - req_accept_encoding
  - req_accept_language
  - req_allow
  - req_authorization
  - req_cache_control
  - req_connection
  - req_content_encoding
  - req_content_language
  - req_content_location
  - req_content_type
  - req_cookie
  - req_date
  - req_expect
  - req_forwarded
  - req_from
  - req_host
  - req_if_match
  - req_if_modified_since

Arguments:

- **Index:** client handle
- **Flag:** more data flag
- **Format:** request data format
  - 0: Binary data format
  - 1: Base64 data format (binary to text encoding)
- **Length:** maximum length of body
- **Body:** received data
### Table 82. AT+HttpSetHeader Http Client Set Header (continued)

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• option:</td>
<td></td>
</tr>
<tr>
<td>- req_if_none_match</td>
<td></td>
</tr>
<tr>
<td>- req_if_range</td>
<td></td>
</tr>
<tr>
<td>- req_if_unmodified_since</td>
<td></td>
</tr>
<tr>
<td>- req_origin</td>
<td></td>
</tr>
<tr>
<td>- req_proxy_auth</td>
<td></td>
</tr>
<tr>
<td>- req_range</td>
<td></td>
</tr>
<tr>
<td>- req_te</td>
<td></td>
</tr>
<tr>
<td>- req_tx_encoding</td>
<td></td>
</tr>
<tr>
<td>- req_upgrade</td>
<td></td>
</tr>
<tr>
<td>- req_user_agent</td>
<td></td>
</tr>
<tr>
<td>- req_via</td>
<td></td>
</tr>
<tr>
<td>- req_warning</td>
<td></td>
</tr>
<tr>
<td>• flags: bitmask:</td>
<td></td>
</tr>
<tr>
<td>- not_persistent: Header Field added is not persistent</td>
<td></td>
</tr>
<tr>
<td>- persistent: Header Field added is persistent</td>
<td></td>
</tr>
<tr>
<td>• format: data format</td>
<td></td>
</tr>
<tr>
<td>- 0: Binary data format</td>
<td></td>
</tr>
<tr>
<td>- 1: Base64 data format (binary to text encoding)</td>
<td></td>
</tr>
<tr>
<td>• length: length of data (optional)</td>
<td></td>
</tr>
<tr>
<td>• data: (optional)</td>
<td></td>
</tr>
</tbody>
</table>

### Table 83. AT+HttpGetHeader Http Client Get Header

<table>
<thead>
<tr>
<th>Request:</th>
<th>Response:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT+HttpGetHeader = [index],[option],[format],[length]</td>
<td>+HttpGetHeader:[index],[format],[length],[data] OK</td>
</tr>
<tr>
<td>Arguments:</td>
<td>Arguments:</td>
</tr>
<tr>
<td>• index: client handle received from AT+HttpCreate</td>
<td>• index: client handle</td>
</tr>
<tr>
<td>• option: see option in AT+HttpSetHeader command (Table 82)</td>
<td>• format: data format</td>
</tr>
<tr>
<td>• format: data format</td>
<td>- 0: Binary data format</td>
</tr>
<tr>
<td>- 0: Binary data format</td>
<td>- 1: Base64 data format (binary to text encoding)</td>
</tr>
<tr>
<td>- 1: Base64 data format (binary to text encoding)</td>
<td>• length: current length of data</td>
</tr>
<tr>
<td>• length: maximum length of data</td>
<td>• data: received value</td>
</tr>
</tbody>
</table>
### Table 84. AT+HttpSetOpt Http Client Set Option

<table>
<thead>
<tr>
<th>Request: AT+HttpSetOpt = [index],[option],[value]</th>
<th>Response: OK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arguments: Index: client handle received from At+HttpCreate</td>
<td>Arguments:</td>
</tr>
<tr>
<td><strong>Option</strong></td>
<td><strong>Value</strong></td>
</tr>
</tbody>
</table>
| redirect_feature | • 0: disable redirect feature  
• 1: enable redirect feature |
| res_filter_clear | • 1: clear response filter to default (all enabled) |
| redirect_tls_downgrade | • 0: disable the option for tls downgrade  
• 1: enable the option for tls downgrade |

### Table 85. AT+HttpSetProxy Http Client Set Proxy Address

<table>
<thead>
<tr>
<th>Request: AT+HttpSetProxy = [family],[port],[address]</th>
<th>Response: OK</th>
</tr>
</thead>
</table>
| Arguments: family: Internet Protocol  
- INET: for network protocol IPv4  
- INET6: for network protocol IPv6  
port: proxy port  
address: proxy server address | Arguments: |
## Revision History

### Changes from B Revision (April 2019) to C Revision

<table>
<thead>
<tr>
<th>Changes</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Added WPA2_PLUS and WPA3 to <strong>AT+WlanConnect</strong> Connect to WLAN Network as a Station table.</td>
<td>17</td>
</tr>
<tr>
<td>• Added WPA2_PLUS and WPA3 to <strong>AT+WlanProfileAdd</strong> Add Profile table.</td>
<td>18</td>
</tr>
<tr>
<td>• Added WPA3 to <strong>AT+WlanScan</strong> Gets the WLAN Scan Operation Results table.</td>
<td>21</td>
</tr>
</tbody>
</table>
IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2020, Texas Instruments Incorporated