User's Guide

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

TEXAS INSTRUMENTS

ABSTRACT

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.

Table of Contents

List of Tables

Trademarks

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:

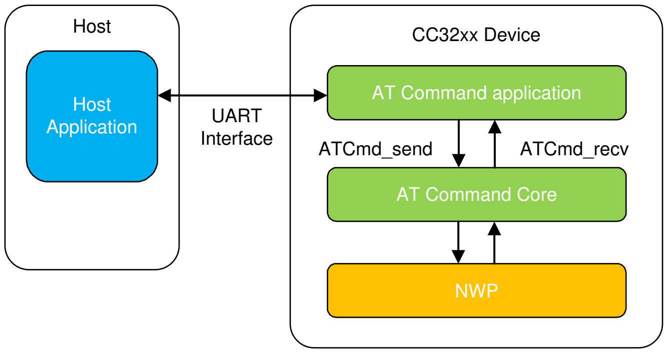
- *ATCmd_create* creates the AT Command core task and initializes the RX event queue.
- *ATCmd_send* transmits string from the AT Command application to the AT Command Core.

  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 2-1 shows the basic architecture.



Copyright © 2017, Texas Instruments Incorporated

Figure 2-1. Basic Architecture Scheme

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 4-1. Device Commands

Command	Definition
AT+Start	Starts the network processor (NWP)
AT+Stop	Stops the NWP
AT+Get	Gets device configurations
AT+Set	Sets device configurations
AT+Test	Test command

Table 4-2. Socket Commands

Command	Definition
AT+Socket	Create an endpoint for communication
AT+Close	Close socket
AT+Accept	Accept a connection on a socket
AT+Bind	Assign a name to a socket
AT+Listen	Listen for connections on a socket
AT+Connect	Initiate a connection on a socket
AT+Select	Monitor socket activity
AT+SetSockOpt	Set socket options
AT+GetSockOpt	Get socket options
AT+Recv	Read data from TCP socket
AT+RecvFrom	Read data from socket
AT+Send	Write data to TCP socket
AT+SendTo	Write data to socket

Table 4-3. WLAN Commands

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

Table 4-4. File System Commands

Command	Definition
AT+FileOpen	Open file in storage device
AT+FileClose	Close file in storage device
AT+FileCtl	Controls various file system operations
AT+FileDel	Delete file from storage device
AT+FileGetFilelist	Get list of a files
AT+FileGetInfo	Get information of a file
AT+FileRead	Read block of data from a file in storage device
AT+FileWrite	Write block of data to a file in storage device

Table 4-5. Network Application Commands

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

Table 4-6. Network Configuration Commands

Command	Definition
AT+NetCfgSet	Setting network configurations
AT+NetCfgGet	Getting network configurations

Table 4-7. Network Utility Commands

Command	Definition
AT+NetUtilGet	Getting utilities configurations
AT+NetUtilCmd	Performing utilities-related commands

Table 4-8. Asynchronous Events

Command	Definition
+EventFatalError	Fatal Error event for inspecting fatal error
+EventGeneral	General asynchronous event for inspecting general events
+EventWlan	WLAN asynchronous event
+EventNetApp	Network Application asynchronous event
+EventSock	Socket asynchronous event

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 enclosed with square brackets ([ ]).
- 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 6-1. *AT+Start* Starts the NWP

Request:	**Response:**
AT+Start	OK
Arguments: none	Arguments: none

Table 6-2. *AT+Stop* Stops the NWP

Request:	**Response:**
AT+Stop = [Timeout]	OK
Arguments: 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. • 0: Enter to hibernate immediately • 0xFFFF: Host waits for the response from the device before hibernating, without timeout protection • 0 <Timeout[msec] <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.	Arguments: none

Table 6-3. *AT+Get* Getting Device Configurations

Request:		**Response:**
AT+Get = [ID],[Option]		+Get:[Value1],..,[ValueX] OK
Arguments:		Arguments:
ID	Option	Return Values
Status	*Device*	Value1: bitmask: General error
	WLAN	Value1: bitmask: • WLANASYNCONNECTEDRESPONSE • WLANASYNCDISCONNECTEDRESPONSE • STA_CONNECTED • STA_DISCONNECTED • P2P_DEV_FOUND • CONNECTION_FAILED • P2P_NEG_REQ_RECEIVED • RX_FILTERS
	BSD	Value1: bitmask: • TX_FAILED
	NETAPP	Value1: bitmask: • IPACQUIRED • IPACQUIRED_V6 • IP_LEASED • IP_RELEASED • IPV4_LOST • DHCP_ACQUIRE_TIMEOUT • IP_COLLISION • IPV6_LOST
General	*Version*	• Value1: Chip ID • Value2: FW Version (x.x.x.x) • Value3: PHY Version (x.x.x.x) • Value4: NWP Version (x.x.x.x) • Value5: ROM Version
	Time	• Value1: Hour = Current hours • Value2: Minute = Current minutes • Value3: Second = Current seconds • Value4: Day = Current Date, 1–31 • Value5: Month = Current Month, 1–12 • Value6: Year = Current year
	Persistent	Value1: • 1: Enable • 0: Disable
IOT	*UDID*	16 bytes

Table 6-4. *AT+Set* Setting Device Configurations

Request:			**Response:**
AT+Set = [ID],[Option],[Value1],..,[ValueX]			OK
Arguments:			
ID	Option	Value	
General	*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.	Value1: • 1: Enable • 0: Disable	
	Time sets the device time and date	• Value1: Hour = Current hours • Value2: Minute = Current minutes • Value3: Second = Current seconds • Value4: Day = Current Date, 1–31 • Value5: Month = Current Month, 1–12 • Value6: Year = Current year	

Table 6-5. *AT+Test* Test Command

Request:	**Response:**
AT+Test	OK
Arguments: none	Arguments: none

6.2 Socket Commands

Table 6-6. *AT+Socket* Create an End-Point for Communication

Request:	**Response:**
AT+Socket = [Domain],[Type],[Protocol]	+Socket: [socket] OK
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 6-7 through Table 6-18.

Table 6-7. *AT+Close* Close Socket

Request:	**Response:**
AT+Close = [socket]	+Close: [socket] OK

Table 6-7. *AT+Close* Close Socket (continued)

Request:	Response:
Arguments: socket: Socket descriptor received from AT+Socket command	

Table 6-8. *AT+Accept* Accept a Connection on a Socket

Request:	Response:
AT+Accept = [socket],[family]	OK +Accept: [New Socket],[Family],[Port],[Address]
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 6-9. *AT+Bind* Assign a Name to a Socket

Request:	Response:
AT+Bind = [Socket],[Family],[Port],[Address]	OK
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 6-10. *AT+Listen* Listen for Connections on a Socket

Request:	Response:
AT+Listen = [socket],[backlog]	OK
Arguments: • socket: Received from AT+Socket command • backlog: Listen	

Table 6-11. *AT+Connect* Initiate a Connection on a Socket

Request:	**Response:**
AT+Connect = [Socket],[Family],[Port],[Address]	OK +Connect : [Port], [Address]
Arguments: • Socket: Received from AT+Socket command • Family: internet protocol: – **INET**: For network protocol IPv4 – **INET6**: For network protocol IPv6 • Port: Address port • Address: Peer socket address (“x.x.x.x”)	

Table 6-12. *AT+Select* Monitor Socket Activity

Request:	**Response:**
AT+Select = [nfds],[readsds],[timeout sec],[timeout usec]	OK +Select: [readsds]
Arguments: • nfds: The highest-numbered file descriptor in any of the three sets (read, write, and except) • readsds: Socket descriptors as bit list (for example, 0|2 for monitoring socket 0 and socket 2) • timeout sec: Time in seconds is an upper bound on the amount of time elapsed before select() returns. 0 means return immediately. • timeout usec: Time in microseconds	Arguments: readsds: Socket descriptors list for read monitoring and accept monitoring

Table 6-13. *AT+SetSockOpt* Set Socket Options

Request:			Response:
AT+SetSockOpt = [sd],[Level],[Option],[Value1],..,[ValueX]			OK
Arguments: sd: Socket descriptor			
Level: Defines the protocol level for this option	**Option**	**Value**	
SOCKET	*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	
	RCVTIMEO 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_ECDHE_ECDSA_WITH_AES_128_CBC_SHA • TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA • TLS_RSA_WITH_AES_128_GCM_SHA256 • TLS_RSA_WITH_AES_256_GCM_SHA384 • 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_ECDSA_WITH_AES_128_GCM_SHA256 • TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 • TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_ SHA256 • TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_ SHA256 • TLS_DHE_RSA_WITH_CHACHA20_POLY1305_ SHA256	
SOCKET (continued)	*SECURE_FILES_CA_FILE_NAME* Map secured socket to CA file by name	Value1: File name	

Table 6-13. *AT+SetSockOpt* Set Socket Options (continued)

Request:			Response:
	SECURE_FILES_PRIVATE_KEY_FILE_NAME Map secured socket to private key by name	Value1: File name	
	SECURE_FILES_CERTIFICATE_FILE_NAME Map secured socket to certificate file by name	Value1: File name	
	SECURE_FILES_DH_KEY_FILE_NAME Map secured socket to Diffie Hellman file by name	Value1: File name	
	CHANGE_CHANNEL Sets channel in transceiver mode	Value1: Channel number (range is 1–13)	
	SECURE_ALPN Sets the ALPN list	Value1: The parameter is a bit map consist of or of the following values: H1 | H2 | H2C | H2_14 | H2_16 | FULL_LIST	
	LINGER Socket lingers on close pending remaining send and receive packets	• Value1: – 1: Enable – 0: Disable • Value2: Linger time in seconds	
	SECURE_EXT_CLIENT_CHLNG_RESP Set with no parameter to indicate that the client uses external signature using Network Application request	Value1: Ignore	
	SECURE_DOMAIN_NAME_VERIFICATION Set a domain name, to check in SSL client connection	Value1: Domain name	
IP	*MULTICAST_TTL* Set the time-to-live value of outgoing multicast packets for this socket	Value1: Number of hops	
	ADD_MEMBERSHIP UDP socket, join a multicast group	• Value1: IPv4 multicast address to join • Value2: Multicast interface address	
	DROP_MEMBERSHIP UDP socket, leave a multicast group	• Value1: IPv4 multicast address to join • Value2: Multicast interface address	
	RAW_RX_NO_HEADER Raw socket remove IP header from received data	Value1: • 1: Remove header • 0: Keep header	
	HDRINCL 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 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	
PHY	*PHY_RATE* Set WLAN PHY transmit rate on RAW socket	Value1: Rate	
	PHY_TX_POWER RAW socket, set WLAN PHY TX power	Value1: Power rage is 1–15	
	PHY_NUM_FRAMES_TO_TX RAW socket, set number of frames to transmit in transceiver mode	Value1: Number of frames	
	PHY_PREAMBLE RAW socket, set WLAN PHY preamble for long or short	Value1: Preamble value	
	PHY_TX_INHIBIT_THRESHOLD RAW socket, set WLAN TX inhibit threshold (CCA).	Value1: Threshold value: • MIN • LOW • DEFAULT • MED • HIGH • MAX	
	PHY_TX_TIMEOUT RAW socket, changes the TX timeout (lifetime) of transceiver frames	Value1: Time in milliseconds, maximum value is 10 ms	

Table 6-13. *AT+SetSockOpt* Set Socket Options (continued)

Request:			Response:
	PHY_ALLOW_ACKS RAW socket, enable sending ACKs in transceiver mode	Value1: • 1: Enable • 0: Disable	

Table 6-14. *AT+GetSockOpt* Get Socket Options

Request:	Response:
AT+GetSockOpt = [sd],[level],[option]	+GetSockOpt: [value1],..,[valueX] OK
Arguments: • sd: Socket handle • level: Defines the protocol level for this option (see Table 6-13) • option: Defines the option name to interrogate (see Table 6-13)	Arguments: value1,..,valueX (see the AT +SetSockOpt command in Table 6-13)

Table 6-15. *AT+Recv* Read Data From TCP Socket

Request:	Response:
AT+Recv = [sd],[format],[length]	OK +Recv: [sd],[format],[length],[data]
Arguments: • 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 6-16. *AT+RecvFrom* Read Data From Socket

Request:	**Response:**
AT+RecvFrom = [sd],[family],[port],[addr],[format],[length]	OK +RecvFrom: [sd],[format],[length],[data]
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 6-17. *AT+Send* Write Data to TCP Socket

Request:	**Response:**
AT+Send = [sd],[format],[length],[data]	OK
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 6-18. *AT+SendTo* Write Data to Socket

Request:	**Response:**
AT+SendTo = [sd],[family],[port],[addr],[format],[length],[data]	OK
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 6-19. *AT+WlanConnect* Connect to WLAN Network as a Station

Request:	Response:
AT+WlanConnect = [SSID],[BSSID],[SecurityType],[SecurityKey],[SecurityExtUser], [SecurityExtAnonUser], [SecurityExtEapMethod]	OK

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 6-20. *AT+WlanDisconnect* Disconnect the Connection

Request:	Response:
AT+WlanDisconnect	OK
Arguments: none	

Table 6-21. *AT+WlanProfileAdd* Add Profile

Request:	Response:
AT+WlanProfileAdd = [SSID],[BSSID],[SecurityType],[SecurityKey],[SecurityExtUser], [SecurityExtAnonUser], [SecurityExtEapMethod],[Priority]	+WlanProfileAdd: [index] OK
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	Arguments: index: Profile stored index

Table 6-22. *AT+WlanProfileGet* Get Profile

Request:	Response:
AT+WlanProfileGet = [index]	+WlanProfileGet: [SSID],[BSSID],[SecurityType],[SecurityExtUser], [SecurityExtAnonUser],[SecurityExtEapMethod],[priority] OK
Arguments: index: Profile stored index received from +WlanProfileAdd	Arguments: See the AT+WlanProfileAdd command in Table 6-21.

Table 6-23. *AT+WlanProfileDel* Delete Profile

Request:	Response:
AT+ WlanProfileDel = [index]	OK
Arguments: index: Number of profile to delete received from +WlanProfileAdd To delete all profiles, use index = 0xFF	

Table 6-24. *AT+WlanPolicySet* Set Policy Values

Request:			**Response:**
AT+WlanPolicySet = [Type],[Option],[Value]			OK
Type	Option	Value	
CONNECTION Defines options available to connect to the AP (Options could be set as bit masked). No option selected = disable all	*Auto* Reconnect to one of the stored profiles each time the connection fails or the device is rebooted	Ignore	
	Fast Establish a fast connection to AP	Ignore	
	P2P Automatically connect to the first P2P device available	Ignore	
	Auto_Provisioning Start the provisioning process after a long period of disconnection when profiles exist	Ignore	
SCAN Defines system scan time interval. An interval is 10 minutes. After settings scan interval, an immediate scan is activated	*Hidden_SSID*	Scan interval in seconds	
	No_Hidden_SSID	Scan interval in seconds	
	Disable_Scan	Ignore	
PM Defines a power management policy for Station mode	*Normal*	Ignore	
	Low_Latency	Ignore	
	Low_Power	Ignore	
	Always_On	Ignore	
	Long_Sleep	Maximum sleep time in milliseconds	
P2P Defines P2P negotiation policy parameters for P2P role	• *CLIENT* Indicates that the device is forced to be CLIENT • *GROUP_OWNER* Indicates that the device is forced to be P2P GO • *NEGOTIATE* Indicates that the device can be either CLIENT or GO, depending on the Wi-Fi Direct® negotiation tiebreaker	• *ACTIVE* When the remote peer is found after the discovery process, the device immediately sends the negotiation request to the peer device. • *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. • *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.	

Table 6-25. *AT+WlanPolicyGet* Get Policy Values

Request:	Response:
AT+WlanPolicyGet = [Type]	+WlanPolicyGet: [Option],[Value] OK
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	Arguments: • Option: See the AT+WlanPolicySet command in Table 6-24 • Value: See the AT+WlanPolicySet command in Table 6-24

Table 6-26. *AT+WlanScan* Gets the WLAN Scan Operation Results

Request:	Response:
AT+WlanScan = [Index],[Count]	+WlanScan: [SSID],[BSSID],[RSSI],[Channel],[Security_Type],[Hidden_SSID], [Cipher],[Key_Mgmt]; OK
Arguments: • Index: Starting index identifier (range 0–29) for getting scan results. • Count: How many entries to fetch; maximum is 30	Arguments: • 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 6-27. *AT+WlanSetMode* WLAN Set Mode

Request:	Response:
AT+WlanSetMode = [Mode]	OK
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 6-28. *AT+WlanSet* Setting WLAN Configurations

Request:			Response:
AT+WlanSet = [ID],[Option],[Value1],..,[ValueX]			OK
ID	Option	Value	
AP	*SSID* Set SSID for AP mode	String up to 32 characters	
	CHANNEL Set channel for AP mode	Channel in the range of [1–11]	
	HIDDEN_SSID Set Hidden SSID Mode for AP mode	• 0: Disabled • 1: Send empty (length = 0) SSID in beacon and ignore probe request for broadcast SSID • 2: Clear SSID (ASCII 0), but keep the original length (this may be required with some clients that do not support empty SSID) and ignore probe requests for broadcast SSID	
	SECURITY Set Security type for AP mode	• OPEN: Open security • WEP: WEP security • WPA_WPA2: WPA security	
	PASSWORD Set Password for AP mode (for WEP or for WPA)	Password for WPA: 8–63 characters Password for WEP: 5 or 13 characters (ASCII)	
	MAX_STATIONS Set Max AP stations	1...4 Note: can be less than the number of currently connected stations	
	MAX_STA_AGING Set Max station aging time	Number of seconds	
	ACCESS_LIST_MODE Set AP access list mode	• *DISABLE* • *DENY_LIST*: Set Black List Mode	
	ACCESS_LIST_ADD_MAC Add MAC address to the AP access list	MAC address: 6 characters	
	ACCESS_LIST_DEL_MAC Delete MAC address from the AP access list	MAC address: 6 characters	
	ACCESS_LIST_DEL_IDX Delete MAC address from index in the AP access list	Index	
GENERAL	*COUNTRY_CODE* Set Country Code for AP mode	Two characters country code	
	STA_TX_POWER Set STA mode TX power level	Number between 0–15, as dB offset from maximum power (0 sets maximum power)	
	AP_TX_POWER Set AP mode TX power level	Number between 0–15, as dB offset from maximum power (0 sets maximum power)	

Table 6-28. *AT+WlanSet* Setting WLAN Configurations (continued)

Request:			**Response:**
	INFO_ELEMENT Set Info Element for AP mode	• Value1: Index of the info element • Value2: Role: – AP – P2P • Value3: Info element ID • Value4: Organization unique ID first Byte • Value5: Organization unique ID second Byte • Value6: Organization unique ID third Byte • Value7: Info element (maximum 252 chars)	
	SCAN_PARAMS Set scan parameters	• Value1: Channel mask • Value2: RSSI threshold	
	SUSPEND_PROFILES Set suspended profiles mask	Suspended bitmask	
	DISABLE_ENT_SERVER_AUTH This option enables to skip server authentication and is valid for one use, when manually connection to an enterprise network	• 1: Disable the server authentication • 0: Enable	
P2P	*DEV_TYPE* Set P2P Device type	Device type is published under P2P I.E (maximum length of 17 characters)	
	CHANNEL_N_REGS Set P2P Channels	• Value1: Listen channel (either 1/6/11 for 2.4 GHz) • Value2: Listen regulatory class (81 for 2.4 GHz) • Value3: Operating channel (channel 1, 6, or 11 for 2.4 GHz) • Value4: Operating regulatory class (81 for 2.4 GHz)	
RX_FILTER	*STATE* Enable or disable filters	Filter Bitmap array (16 bytes in format xx:xx)	
	SYS_STATE Enable or disable system filters	Filter Bitmap array (4 bytes in format xx:xx)	
	REMOVE Remove filters	Filter Bitmap array (16 bytes in format xx:xx)	
	STORE Save the filters as persistent	null	
Network Assisted Roaming	*SL_WLAN_ROAMING_TRIGGERING_ENABLE* Enable or disable Roaming by RSSI trigger	• Value 1: 1 - Enable the roaming by RSSI trigger 0 - Disable • Value 2 : RSSI threshold for roaming in dBm units, range [-85, 0]	
	SL_WLAN_AP_TRANSITION_ENABLE Enable or disable Agile MBO	• 1 - Enable Agile MBO • 0 - Disable	

Table 6-29. *AT+ WlanGet* Getting WLAN Configurations

Request:		**Response:**
AT+WlanGet = [ID],[Option]		+WlanGet: [Value1],..,[ValueX] OK
Arguments:		Arguments: See the AT+WlanSet command in Table 6-28.
ID	Option	
AP	*SSID* Get SSID for AP mode	
	CHANNEL Get channel for AP mode	
	HIDDEN_SSID Get Hidden SSID Mode for AP mode	
	SECURITY Get Security type for AP mode	
	PASSWORD Get Password for AP mode (for WEP or for WPA)	
	MAX_STATIONS Get Max AP allowed stations	
	MAX_STA_AGING Get AP aging time in seconds	
	ACCESS_LIST_NUM_ENTRIES Get AP access list number of entries	
ACCESS_LIST Get the AP access list from start index	The start index in the access list	
GENERAL	*COUNTRY_CODE* Get Country Code for AP mode	
	STA_TX_POWER Get STA mode TX power level	
	AP_TX_POWER Get AP mode TX power level	
	SCAN_PARAMS Get scan parameters	
P2P	*CHANNEL_N_REGS* Get P2P Channels	
RX_FILTER	*STATE* Retrieves the filters enable/disable status	
	SYS_STATE Retrieves the system filters enable or disable status	

Table 6-29. *AT+ WlanGet* Getting WLAN Configurations (continued)

Request:		**Response:**
Connection	Ignore	• 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 6-30. *AT+FileOpen* Open File in Storage Device

Request:	**Response:**
AT+FileOpen = [Filename],[Options],[File size]	+FileOpen:[FileID],[Secure Token] OK
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) /* Creation flags 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 6-31. *AT+FileClose* Close File in Storage Device

Request:	**Response:**
AT+FileClose = [FileID],[CeritificateFileName],[Signature]	OK
Arguments: • FileID: Assigned from AT+FileOpen • CeritificateFileName: Certificate file with full path (Optional) • Signature: The signature is SHA-1, the certificate chain may include SHA-256 (Optional)	

Table 6-32. *AT+FileCtl* Controls Various File System Operations

Request:				**Response:**	
AT+FileCtl = [Command],[Secure_Token],[Filename],[Data]				+FileCtl:[NewSecureToken],[OutputData] OK	
Arguments:				Arguments:	
Command	Token	Filename	Data	Token	Output Data
RESTORE Return to factory default	Ignore	Ignore	*FACTORY_IMAGE* The system will be back to the production image. *FACTORY_DEFAULT* Return to factory default	Ignore	Ignore
ROLLBACK Roll-back file	Token assigned from AT+FileOpen	Filename to roll back	Ignore	New secure token	Ignore
COMMIT Commit file	Token assigned from AT+FileOpen	Filename to commit	Ignore	New secure token	Ignore
RENAME Rename file	Token assigned from AT+FileOpen	Filename to rename	New file name	Ignore	Ignore
GET_STORAGE_INFO Get storage information	Ignore	Ignore	Ignore	Ignore	• DeviceBlockSize • DeviceBlocks Capacity • NumOfAllocatedBlocks • NumOfReservedBlocks • NumOfReservedBlocksFor Systemfiles • LargestAllocatedGapInBlocks • NumOfAvailableBlocks • ForUserFiles • MaxFsFiles • IsDevlopment FormatType • Bundlestate • MaxFsFilesReservedForSysFiles • ActualNumOf UserFiles • ActualNumOf SysFiles NumOfAlerts • NumOfAlerts Threshold • FATWrite Counter
BUNDLE_ROLLBACK Roll back a bundle	Ignore	Ignore	Ignore	Ignore	Ignore
BUNDLE_COMMIT Commit a bundle	Ignore	Ignore	Ignore	Ignore	Ignore

Table 6-33. *AT+FileDel* Delete File From Storage Device

Request:	Response:
AT+FileDel = [FileName],[SecureToken]	OK
Arguments: • FileName: Full path File Name • SecureToken: Token assigned from AT+FileOpen (optional)	

Table 6-34. *AT+FileGetFilelist* Get a List of Files

Request:	Response:
AT+FileGetFileList	+FileGetFileList: [FileName],[FileMaxSize],[Properties], [FileAllocatedBlocks] OK
Arguments:	Arguments: • FileName: File name • FileMaxSize: Maximum file size • Properties: Info flag bitmask • FileAllocatedBlocks: Allocated blocks

Table 6-35. *AT+FileGetInfo* Get Information About a File

Request:	Response:
AT+FileGetInfo = [FileName],[SecureToken]	+FileGetInfo: [Flags],[File Size],[Allocated Size], [Tokens], [Storage Size],[Write Counter] OK
Arguments: • FileName: Full path file name • SecureToken: token assigned from AT+FileOpen (optional)	

Table 6-36. *AT+FileRead* Read a Block of Data From a File in Storage Device

Request:	Response:
AT+FileRead = [FileID],[Offset],[Format],[Length]	+FileRead:[format],[NumberOfReadBytes],[ReceivedData] OK
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 6-37. *AT+FileWrite* Write Block of Data to a File in Storage Device

Request:	Response:
AT+FileWrite = [FileID],[Offset],[Format],[Length],[Data]	+FileWrite:[NumberOfWrittenBytes] OK
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 6-38. ***AT+NetAPPStart*** **Starts a Network Application**

Request:	Response:
AT+NetAPPStart = [APP Bitmap]	OK
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 6-39. ***AT+NetAPPStop*** **Stops a Network Application**

Request:	Response:
AT+NetAPPStop = [APP Bitmap]	OK
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 6-40. ***AT+NetAPPGetHostByName*** **Get Host IP by Name**

Request:	Response:
AT+NetAPPGetHostByName = [HostName],[Family]	OK +NetAPPGetHostByName: [HostName],[Host IP address]
Arguments: • HostName • Family: Protocol Family: – **INET**: For network protocol IPv4 – **INET6**: For network protocol IPv6	Arguments: • HostName • Host IP address: IP address according to the family (IPv4 or IPv6)

Table 6-41. ***AT+NetAPPGetHostByService*** **Get Host IP by Service**

Request:	**Response:**
AT+NetAPPGetHostByService = [ServiceName],[Family]	OK +NetAPPGetHostByService: [ServiceName],[Port],[HostIPAddress],[Text]
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 6-42. ***AT+NetAPPSet*** **Setting Network Application Configurations**

Request:			**Response:**
AT+NetAPPSet = [App ID],[Option],[Value1],…,[ValueX]			OK
Arguments:			
App ID	Option	Values	
DHCP_SERVER	*BASIC*	• Value1: Lease time (in seconds) of the IP Address • Value2: First IP Address for allocation • Value3: Last IP Address for allocation	
HTTP_SERVER	*PRIM_PORT_NUM*	Value1: port number	
	AUTH_CHECK	Value1: • 1: Authentication enable • 0: Authentication disable	
	AUTH_NAME	Value1: Authentication name (maximum length is 20 bytes)	
	AUTH_PASSWORD	Value1: Authentication password (maximum length is 20 bytes)	
	AUTH_REALM	Value1: Authorization realm (maximum length is 20 bytes)	
	ROM_PAGES_ACCESS	Value1: • 1: Access enable • 0: Access disable	
	SECOND_PORT_NUM	Value1: port number	
	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)	
MDNS	*CONT_QUERY*	Value1: Service name (maximum length is 80 bytes)	

Table 6-42. *AT+NetAPPSet* Setting Network Application Configurations (continued)

Request:			Response:
	QEVETN_MASK	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	• Value1: Period in ticks (100 ticks = 1 second) • Value2: Repetitions • Value3: Telescopic factor • Value4: Retransmission interval • Value5: Maximum period interval • Value6: Maximum time	
DEVICE	*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 6-43. *AT+NetAPPGet* Getting Network Applications Configurations

Request:		**Response:**
AT+NetAPPGet = [App ID],[Option]		+NetAPPGet: [return values] OK
Arguments:		Arguments: See AT+NetAPPSet command values
App ID	Option	
DHCP_SERVER	*BASIC*	
HTTP_SERVER	*PRIM_PORT_NUM*	
	AUTH_CHECK	
	AUTH_NAME	
	AUTH_PASSWORD	
	AUTH_REALM	
	ROM_PAGES_ACCESS	
	SECOND_PORT_NUM	
	SECOND_PORT_EN	
	PRIM_PORT_SEC_EN	
MDNS	*CONT_QUERY*	
	QEVETN_MASK	
	TIMING_PARAMS	
DEVICE	*URN*	
	DOMAIN	
DNS_CLIENT	*TIME*	

Table 6-44. *AT+NetAPPSend* Sends Network Application Response or Data Following a Network Application Request Event

Request:	**Response:**
AT+NetAPPSend = [Handle],[Flags],[Format],[Length],[Data]	OK
Arguments: • Handle: Handle to send the data to. Should match the handle received in the Network Application request event • Flags: Bitmask: – CONTINUATION: More data will arrive in subsequent calls to AT+NetAPPSend – METADATA: Define data as metadata, otherwise data is payload – ACCUMULATION: The network processor should accumulate the data chunks and will process it when it is completely received • 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. Can be just data payload or metadata (depends on flags)	

Table 6-45. *AT+NetAPPRecv* Receives Data From the Network Processor Following a Network Application Response Event

Request:	**Response:**
AT+NetAPPRecv = [Handle],[Format],[Length]	OK +NetAPPRecv:[Handle],[Flags],[Format],[Length],[Data]
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	Arguments: • Handle • Flags: Can have the following value: • **CONTINUATION**: More data is pending in the network processor. Application should continue reading the data by calling *AT+NetAPPRecv* again • Format: Data format: – 0: Binary data format – 1: Base64 data format • Length: Number of bytes received • Data: Data received

Table 6-46. *AT+NetAPPPing* Send Ping to Network Hosts

Request:	**Response:**
AT+NetAPPPing = [Family],[Destination],[Size],[Delay],[Timeout],[Max],[Flags]	OK +NetAPPPing: [PacketsSent],[PacketsReceived],[RoundTime]
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 6-47. *AT+NetAPPGetServiceList* Get Service List

Request:	Response:
AT+NetAPPGetServiceList = [IndexOffset],[MaxServiceCount],[Flags]	+NetAPPGetServiceList:[ServiceInfo1];...;[ServiceInfoX] OK
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	Arguments: ServiceInfo: Depends on flag type: • SHORT_IPV4 • SHORT_IPV6 – ip – port • FULL_IPV4 • FULL_IPV6 – ip – port – service name – service host name • FULL_IPV4_WITH_TEXT • FULL_IPV6_WITH_TEXT – ip – port – service name – service host name – service text

Table 6-48. *AT+NetAPPRegisterService* Register a New mDNS Service

Request:	Response:
AT+NetAPPRegisterService = [ServiceName],[Text],[Port],[TTL],[Options]	OK
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 6-49. *AT+NetAPPUnRegisterService* Unregister mDNS Service

Request:	Response:
AT+NetAPPUnRegisterService = [ServiceName],[Options]	OK
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 6-50. *AT+NetCfgSet* Setting Network Configurations

Request:			**Response:**
AT+NetCfgSet = [ConfigId],[ConfigOpt],[Value1],..,[ValueX]			OK
Arguments:			
ConfigId	ConfigOpt	Value	
IF	*STATE* Enable or disable modes (bitmask)	• IPV6_STA_LOCAL: Enable ipv6 local • IPV6_STA_GLOBAL: Enable ipv6 global • DISABLE_IPV4_DHCP: Disable ipv4 DHCP • IPV6_LOCAL_STATIC: Enable ipv6 local static • IPV6_LOCAL_STATELESS: Enable ipv6 local stateless • IPV6_LOCAL_STATEFUL: Enable ipv6 local stateful • IPV6_GLOBAL_STATIC: Enable ipv6 global static • IPV6_GLOBAL_STATEFUL: Enable ipv6 global stateful • DISABLE_IPV4_LLA: Disable LLA feature • ENABLE_DHCP_RELEASE: Enables DHCP release • IPV6_GLOBAL_STATELESS: Enable ipv6 global stateless • DISABLE_FAST_RENEW: Fast renew disabled	
SET_MAC_ADDR Setting MAC address to the Device	Ignore value	New MAC address	
IPV4_STA_ADDR Setting IP address	*STATIC* Setting a static IP address	• Value1: IP address • Value2: Subnet mask • Value3: Default gateway address • Value4: DNS server address	
	DHCP Setting IP address by DHCP	Ignore value	
	DHCP_LLA Setting DHCP LLA	Ignore value	
	RELEASE_IP_SET Setting release IP before disconnect enables sending a DHCP release frame to the server	Ignore value	
	RELEASE_IP_OFF Setting release IP before disconnect disables sending a DHCP release frame to the server	Ignore value	
IPV4_AP_ADDR Setting a static IP address to the device working in AP mode	*STATIC* Setting a static IP address	• Value1: IP address • Value2: Subnet mask • Value3: Default gateway address • Value4: DNS server address	

Table 6-50. *AT+NetCfgSet* Setting Network Configurations (continued)

Request:			Response:
IPV6_ADDR_LOCAL	*STATIC* Setting a IPv6 Local static address	IP address	
	STATELESS Setting a IPv6 Local stateless address	Ignore value	
	STATEFUL Setting a IPv6 Local stateful address	Ignore value	
IPV6_ADDR_GLOBAL	*STATIC* Setting a IPv6 Global static address Value1 : IP address Value2: DNS Server IP STATEFUL	• Value1: IP address • Value2: DNS Server IP	
	STATEFUL Setting a IPv6 Global stateful address	Ignore value	
AP_STATION_DISCONNECT Disconnect AP station by MAC address	Ignore value	AP MAC address	
IPV4_DNS_CLIENT Set secondary DNS address	Ignore value	Secondary DNS Server address	

Table 6-51. *AT+NetCfgGet* Getting Network Configurations

Request:	Response:
AT+NetCfgGet = [ConfigId]	+NetCfgGet:[Value1],..,[ValueX] OK
Arguments: ConfigId: Configuration ID:	Arguments:
GET_MAC_ADDR Get the device MAC address	Value1: MAC address
IPV4_STA_ADDR Get IP address from WLAN station or P2P client	• Value1: Address option: – DHCP – DHCP_LLA – STATIC • Value2: Address • Value3: Subnet mask • Value4: Gateway • Value5: DNS
IPV4_AP_ADDR Get static IP address for AP or P2P go	
IF Get interface bitmap	Value1: State (bitmask): • ipv6_sta_local • ipv6_sta_global • disable_ipv4_dhcp • ipv6_local_static • ipv6_local_stateless • ipv6_local_stateful • ipv6_global_static • ipv6_global_stateful • disable_ipv4_lla • enable_dhcp_release • ipv6_global_stateless • disable_fast_renew

Table 6-51. *AT+NetCfgGet* Getting Network Configurations (continued)

Request:	Response:
IPV6_ADDR_LOCAL Get IPV6 Local address	• Vaule1: Address option: – stateless – stateful – STATIC • Value2: Address
IPV6_ADDR_GLOBAL Get IPV6 Global address	
AP_STATIONS_CONNECTED Get AP number of connected stations	Value1: Number of connected stations
AP_STATIONS_INFO Get AP full list of connected stations	[address1],[MAC address1],[name1]; ...; [addressX],[MAC addressX],[nameX]
IPV4_DNS_CLIENT Set secondary DNS address	Value1: DNS second server address
IPV4_DHCP_CLIENT Get DHCP Client info	• Value1: Address • Value2: Subnet mask • Value3: Gateway • Value4: DNS 1 • Value5: DNS 2 • Value6: DHCP server • Value7: Lease time • Value8: Time to renew • Value9: DHCP State: – unknown – disabled – enabled – bound – renew – rebind

6.7 Network Utility Commands

Networking related commands and configuration.

Table 6-52. *AT+NetUtilGet* Getting Utilities Configurations

Request:		**Response:**
AT+NetUtilGet = [ID],[Option]		+NetUtilGet: [Value1],..,[ValueX] OK
Arguments:		Arguments:
ID Identifier of the specific "get" operation to perform	**Option**	**Value**
public_key	• 0: Binary data format • 1: Base64 data format (binary to text encoding)	• Value1: Public key format: – 0: Binary data format – 1: Base64 data format • Value2: Public key length (maximum length is 255 bytes or 370 bytes in base64 format) • Value3: Public key
true_random	Number of random numbers (maximum is 172 numbers)	List of random numbers

Table 6-53. *AT+NetUtilCmd* Performing Utilities-Related Commands

Request:		**Response:**
AT+NetUtilCmd = [Cmd],[Value1],..,[ValueX]		+NetUtilCmd:[Value1],..,[ValueX] OK
Arguments:		Arguments:
Cmd Identifier of the specific command to perform	**Option**	**Value**
sign_msg Create a digital signature using the ECDSA algorithm	• Value1: Key index: • Value2: Data format: – 0: Binary data format – 1: Base64 data format (binary to text encoding) • Value3: Data length (maximum length is 1500 bytes) • Value4: Data	• Value1: Signature format: – 0: Binary data format – 1: Base64 data format (binary to text encoding) • Value2: Signature length (maximum length is 255 bytes) • Value3: Signature
verify_msg verify a digital signature using the ECDSA algorithm	• Value1: Key index • Value2: Data and signature format: – 0: Binary data format – 1: Base64 data format (binary to text encoding) • Value3: Data length (maximum length is 1500 bytes) • Value4: Signature length • Value5: Data and signature (signature concatenate to end of data)	Value1: Success or failure
temp_keys Create or remove a temporary ECC key pair with the SECP256R1 curve	• Value1: Key index • Value2: Action: – *create* – *remove*	
install_op Install or uninstall a key pair in one of the crypto utilities key pair management mechanism	• Value1: Key index • Value2: Action: – *install* – *uninstall* • Value3: Key Algorithm (ignored for uninstall action): – *none* – *ec* • Value4: EC Named Curve identifier (optional for Key Algorithm none) (ignored for uninstall action): – *none* – *secp256r1* • Value5: Certification file name (ignored for uninstall action) • Value6: Key file name (ignored for uninstall action)	

6.8 Asynchronous Events

Table 6-54. *+EventFatalError* Fatal Error Event for Inspecting Fatal Error

Response:	
+EventFatalError:[EventID],[Value1],..,[ValueX]	
Arguments:	
EventID	Value
DEVICE_ABORT Indicates a severe error occurred and the device stopped	• Value1: An indication of the abort type • Value2: The abort data
NO_CMD_ACK Indicates that the command sent to the device had no ACK	Value1: An indication of the CMD opcode
CMD_TIMEOUT Indicates that the command got a timeout while waiting for its asynchronous response	Value1: An indication of the asynchronous event opcode
DRIVER_ABORT Indicates a severe error occurred in the driver	null
SYNC_LOSS Indicates a sync loss with the device	null

Table 6-55. *+EventGeneral* General Asynchronous Event for Inspecting General Events

Response:	
+EventGeneral:[EventID],[Value1],..,[ValueX]	
Arguments:	
EventID	Value
RESET_REQUEST	• Value1: An error code indication from the device • Value2: The sender originator: – WLAN – NETCFG – NETAPP – SECURITY – OTHER
ERROR	• Value1: An error code indication from the device • Value2: The sender originator

Table 6-56. +*EventWlan* WLAN Asynchronous Event

Response:	
+EventWlan:[EventID],[Value1],..,[ValueX]	
Arguments:	
EventID	Value
CONNECT STA connection indication event	• Value1: SSID name • Value2: BSSID
P2P_CONNECT P2P client connection indication event	• Value1: SSID name • Value2: BSSID • Value3: Go Device Name
DISCONNECT STA client disconnection event	• Value1: SSID name • Value2: BSSID • Value3: Reason
P2P_DISCONNECT P2P client disconnection event	• Value1: SSID name • Value2: BSSID • Value3: Reason • Value4: Go Device Name
STA_ADDED AP connected STA	Value1: MAC address
STA_REMOVED AP disconnected STA	Value1: MAC address
P2P_CLIENT_ADDED P2P(Go) connected P2P(Client)	• Value1: MAC address • Value2: Go Device Name • Value3: Own SSID
P2P_CLIENT_REMOVED P2P(Go) disconnected P2P(Client)	• Value1: MAC address • Value2: Go Device Name • Value3: Own SSID
P2P_DEVFOUND	• Value1: Go Device Name • Value2: MAC address • Value3: WPS Method
P2P_REQUEST	• Value1: Go Device Name • Value2: MAC address • Value3: WPS Method
P2P_CONNECTFAIL P2P only	Value1: Status
PROVISIONING_STATUS	Value1: Status
PROVISIONING_PROFILE_ADDED	• Value1: Status • Value2: SSID name

Table 6-57. +*EventNetApp* Network Application Asynchronous Event

Response:	
+EventNetApp:[EventID],[Value1],..,[ValueX]	
Arguments:	
EventID	Value
IPV4_ACQUIRED	• Value1: IP address • Value2: Gateway • Value3: DNS
IPV6_ACQUIRED	• Value1: IP address • Value2: DNS
ip_collision	• Value1: IP address • Value2: DHCP MAC • Value3: DNS
IP_LEASED AP or P2P go DHCP lease event	• Value1: IP address • Value2: Lease time • Value3: MAC
IP_RELEASED AP or P2P go DHCP IP release event	• Value1: IP address • Value2: MAC • Value3: Reason
IPV4_LOST	Value1: Status
dhcp_ipv4_acquire_timeout	Value1: Status
IPV6_LOST	Value1: IP lost

Table 6-58. +*EventSock* Socket Asynchronous Event

Response:	
+EventSock:[EventID],[Value1],..,[ValueX]	
Arguments:	
EventID	Value
TX_FAILED	• Value1: sd • Value2: Status

ASYNC_EVENT	• Value1: sd
	• Value2: Type:
	  – SSL_ACCEPT
	  – RX_FRAG_TOO_BIG
	  – OTHER_SIDE_CLOSE_SSL
	  – CONNECTED_SECURED
	  – WRONG_ROOT_CA
	• Value3: Error value

Table 6-59. *+EventMqtt* MQTT Asynchronous Event

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

6.9 MQTT Client Commands

MQTT client commands and configuration.

Table 6-60. *AT+MqttCreate* MQTT Client Create

Request:	**Response:**
AT+MqttCreate = [client ID],[flags],[address],[port],[method],[cipher],[private key],[Certificate],[CA],[DH key],[protocol],[blocking send],[data format]	+ MqttCreate: [index] OK
Arguments:	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_ECDSA_WITH_AES_128_CBC_SHA256 – TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA – TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA – TLS_RSA_WITH_AES_128_GCM_SHA256 – TLS_RSA_WITH_AES_256_GCM_SHA384 – 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_ECDSA_WITH_AES_128_GCM_SHA256 – TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 – TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 – TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 – TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256	index: client handle

Table 6-60. *AT+MqttCreate* MQTT Client Create (continued)

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

Table 6-61. *AT+MqttDelete* MQTT Client Delete

Request:	Response:
AT+MqttDelete = [index]	OK
Arguments:	Arguments:
index: client handle received from At+MqttCreate	

Table 6-62. *AT+MqttConnect* MQTT Client Connect to Broker

Request:	Response:
AT+MqttConnect = [index]	OK
Arguments:	Arguments:
index: client handle received from At+MqttCreate	

Table 6-63. *AT+MqttDisconnect* MQTT Client Disconnect From Broker

Request:	Response:
AT+MqttDisconnect = [index]	OK
Arguments:	Arguments:
index: client handle received from At+MqttCreate	

Table 6-64. *AT+MqttPublish* MQTT Client Send Message to Broker

Request:	**Response:**
AT+MqttPublish = [index],[topic],[QoS],[retain],[message length],[message]	OK
Arguments:	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 6-65. *AT+MqttSubscribe* MQTT Client Subscribe for Topic

Request:	**Response:**
AT+MqttSubscribe = [index],[number of topics],[topic1][QoS1],[persistent1],…, [topicX][QoSX],[persistentX]	OK
Arguments:	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 6-66. *AT+MqttUnsubscribe* MQTT Client Unsubscribe for Topic

Request:	**Response:**
AT+MqttUnsubscribe = [index],[number of topics],[topic1],[persistent1],…, [topicX],[persistentX]	OK
Arguments:	Arguments:
• index: client handle received from At+MqttCreate • number of topics: maximum 4 topics • topic: topic string • persistent (optional for future use)	

Table 6-67. *AT+MqttSet* MQTT Client Set Option

Request:		**Response:**
AT+MqttSet = [index],[option],[value1],..,[valueX]		OK
Arguments:		Arguments:
index: client handle received from At+MqttCreate		
Option	**Value**	
user	Value1: User name string	
password	Value1: Password string	
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 6-68. *AT+HttpCreate* Http Client Create

Request:	**Response:**
AT+HttpCreate	+HttpCreate: [index] OK
Arguments:	Arguments:
	index: client handle

Table 6-69. *AT+HttpDestroy* Http Client Delete

Request:	**Response:**
AT+HttpDestroy = [index]	OK
Arguments:	Arguments:
index: client handle received from At+HttpCreate	

Table 6-70. *AT+HttpConnect* Http Client Connect to Host

Request:	**Response:**
AT+HttpConnect = [index],[host],[flags],[private key],[certificate],[ca]	OK
Arguments:	Arguments:
• index: client handle received from At+HttpCreate • host: host name • flags: bitmask: – ignore_proxy – host_exist • private key: private key file name (optional) • certificate: client certificate file name (optional) • ca: root ca file name (optional)	

Table 6-71. *AT+HttpDisconnect* Http Client Disconnect From Host

Request:	**Response:**
AT+HttpDisconnect = [index]	OK
Arguments:	Arguments:
index: client handle received from At+HttpCreate	

Table 6-72. ***AT+HttpSendReq*** **Http Client Send Request to Host**

Request:	**Response:**
AT+HttpSendReq = [index],[method],[uri],[flags],[format],[length],[data]	+HttpSendReq: [status] OK
Arguments:	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: Flushes 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 6-73. ***AT+HttpReadResBody*** **Http Client Read Response Body From Host**

Request:	**Response:**
AT+HttpReadResBody = [index],[format],[length]	+HttpReadResBody: [index],[flag],[format],[length],[body] OK
Arguments:	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 6-74. *AT+HttpSetHeader* Http Client Set Header

Request:	**Response:**
AT+HttpSetHeader = [index],[option],[flags],[format],[length],[data]	OK
Arguments:	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_auth – 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	• 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 6-74. *AT+HttpSetHeader* Http Client Set Header (continued)

Request:	Response:
• option: – req_if_none_match – req_if_range – req_if_unmodified_since – req_origin – req_proxy_auth – req_range – req_te – req_tx_encoding – req_upgrade – req_user_agent – req_via – req_warning • flags: bitmask: – not_persistent: Header Field added is not persistent – persistent: Header Field added is persistent • format: data format – 0: Binary data format – 1: Base64 data format (binary to text encoding) • length: length of data (optional) • data: (optional)	

Table 6-75. *AT+HttpGetHeader* Http Client Get Header

Request:	Response:
AT+HttpGetHeader = [index],[option],[format],[length]	+HttpGetHeader:[index],format],[length],[data] OK
Arguments:	Arguments:
• index: client handle received from At+HttpCreate • option: see option in AT+HttpSetHeader command (Table 6-74) • format: data format – 0: Binary data format – 1: Base64 data format (binary to text encoding) • length: maximum length of data	• index: client handle • format: data format – 0: Binary data format – 1: Base64 data format (binary to text encoding) • length: current length of data • data: received value

Table 6-76. *AT+HttpSetOpt*Http Client Set Option

Request:		**Response:**
AT+HttpSetOpt = [index],[option],[value]		OK
Arguments:		Arguments:
Index: client handle received from At+HttpCreate		
Option	**Value**	
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 6-77. *AT+HttpSetProxy* Http Client Set Proxy Address

Request:	**Response:**
AT+HttpSetProxy = [family],[port],[address]	OK
Arguments:	Arguments:
• family: Internet Protocol – **INET**: for network protocol IPv4 – **INET6**: for network protocol IPv6 • port: proxy port • address: proxy server address	

Revision History

Changes from Revision C (January 2020) to Revision D (October 2020) **Page**

- Added Network Assisted Roaming to AT+WlanSet Setting WLAN Configurations table.................................20