QCA4010 Hostless SDK API 简介

修订说明

版本	内容	备注
V1	初稿	
V1.1	修正 ADC 相关 API 说明	
V1.2	添加 DHCPS 相关 API	qcom_dhcps_set_pool qcom_dhcps_release_pool

目录

一、系统简述

QCA4010/QCA4012 SoC 提供 802.11 无线局域网（WLAN）的连接，可以作为一个基本编程环境来运行应用程序。它的 WLAN 子系统包括：

- 无线媒体访问控制（MAC）
- 射频 Radio
- 基带 Baseband
- 片上网络处理器处理（进行 IEEE 802.11 协议处理）

非 WLAN 部分包括：

- Tensilica Xtensa 处理器，主频高达 130 MHz
- 1M Bytes ROM，1.4MBytes RAM
- 1024 Bytes 一次性可编程存储（OTP）
- 内容集成开关和线性稳压器的 PMU
- SPI 控制器，支持 quad 模式
- 2*UART
- GPIOs、I^2C、I^2S、PWM、ADC、JTAG
- 硬件加密

二、API 简介

QCA4010 SDK 可分为如下几个部分：

- I/O API

  ——GPIO/I2C/I2S/IrDA/SPI/UART/PWM
- System API

  ——Timer、Socket、Power mode、NVRAM、Encryption
- Network Protocol API

——DHCP、DNS、HTTP、SNTP、OTA、SSL

- Wireless Network API

——WLAN/P2P/WPS/WIFI security、WIFI Scan/WIFI STA/WIFI AP

GPIO API

qcom_gpio_pin_dir

Definition	Set the direction for the given hardware pin when it operates in GPIO mode.	
Prototype	A_STATUS qcom_gpio_ pin_dir(	
	A_UINT32 pin,	Hardware pin number
	A_BOOL input)	A_TRUE = Configure the pin as input. A_FLASE = Configure the pin as output.
Return Value	A_OK on success, A_ERROR on failure (for example, the given hardware pin is not in GPIO mode)	

qcom_gpio_pin_set

Definition	Set the output value for the given hardware pin when it operates in GPIO mode. This API is used only when the given pin’ s direction is set to output by calling qcom_gpio_pin_dir().	
Prototype	A_STATUS qcom_gpio_ pin_set(	
	A_UINT32 pin,	Hardware pin number
	A_BOOL input)	A_TRUE = Drive logic high. A_FLASE = Drive logic low.
Return Value	A_OK on success, A_ERROR on failure (for example, the given hardware pin is not in GPIO mode)	

qcom_gpio_pin_source

Definition	Set the source of the given hardware pin. This API is used only when the given hardware pin is configured to GPIO mode.	
Prototype	A_STATUS qcom_gpio_pin_source(	

	A_UINT32 pin,	Hardware pin
	A_BOOL pwm)	A_TRUE = Sigma-Delta PWM source A_FALSE = WLAN_GPIO_OUT register
Return Value	A_OK on success, A_ERROR on failure (for example, the given hardware pin is not in GPIO mode).	

qcom_gpio_interrupt_wakeup

Definition	Set the wakeup interrupt for the given hardware pin. This API is used only when the given hardware pin is configured to GPIO mode.	
Prototype	A_STATUS qcom_gpio_interrupt_wakeup(	
	A_UINT32 pin,	Hardware pin
	A_BOOL enable)	A_TRUE = Set the pin as wakeup interrupt. A_FALSE = The pin is not a wakeup interrupt.
Return Value	A_OK on success, A_ERROR on failure (for example the given hardware pin is not in GPIO mode).	

qcom_gpio_interrupt_register

Definition	Register an interrupt handler for the given hardware pin. The pin number defined in the qcom_gpio_interrupt_info_t specifies the hardware pin number and the interrupt handler function pointer. Since the hardware pin number is not exported to application, application needs to obtain the hardware pin number by calling qcom_gpio_get_interrupt_pin_num() and pass it on to this API.	
Prototype	A_STATUS qcom_gpio_ interrupt_register(	
	qcom_gpio_interrupt_info_t *gpio_interrupt_info)	This parameter contains the information such as hardware pin number, function pointer, and flags for interrupt handler. The structure looks like the following: typedef struct _qcom_gpio_interrupt_info_t { int pin; void

		(*gpio_pin_int_handler_fn)(void *arg); void *arg; unsigned int internal_flags; } qcom_gpio_interrupt_info_t; internal_flags should contain the interrupt mode. The interrupt mode can take one of the following values. typedef enum { QCOM_GPIO_PIN_INT_NONE = 0, QCOM_GPIO_PIN_INT_RISING_EDGE, QCOM_GPIO_PIN_INT_FALLING_EDGE, QCOM_GPIO_PIN_INT_BOTH_EDGE, QCOM_GPIO_PIN_INT_LEVEL_LOW, QCOM_GPIO_PIN_INT_LEVEL_HIGH, } QCOM_GPIO_PIN_INTERRUPT_MODE;
Return Value	A_OK on success, A_ERROR on failure (for example, invalid hardware pin, internal memory allcation failure).	

qcom_gpio_interrupt_mode

Definition	Configure the interrupt trigger mode (level 0/1 or rising/falling/any edge) for the given hardware pin. This API is used only when the given hardware pin is configured to GPIO mode. When a pin is configured in peripheral mode, the interrupt mode is expected to change in the runtime and is providd as part of the tunable active configuration. The pin number defined in the qcom_gpio_interrupt_info_t specifies the hardware pin number and the interrupt handler function pointer. Since the hardware pin number is not exported to application, application needs to obtain the hardware pin number by calling qcom_gpio_get_interrupt_pin_num() and pass it on to this API.	
Prototype	A_STATUS qcom_gpio_interrupt_mode(	
	qcom_gpio_interrupt_info_t *gpio_interrupt_info,	This parameter contains the information such as hardware pin number, function pointer, and flags for interrupt handler. The structure looks like the following: typedef struct _qcom_gpio_interrupt_info_t { int pin; void

		(*gpio_pin_int_handler_fn)(void *arg); void *arg; unsigned int internal_flags; } qcom_gpio_interrupt_info_t; Do not modify the internal_flags parameter. It is used by GPIO framework internally. The gpio_pin_int_handler_fn parameter can be set to NULL for this API.
	QCOM_GPIO_PIN_INTERRUPT_MODE mode	Specifies the interrupt mode to be configured for the given pin specified in gpio_interrupt_info. The mode can take one of the following enumerations: typedef enum { QCOM_GPIO_PIN_INT_NONE = 0, QCOM_GPIO_PIN_INT_RISING_EDGE, QCOM_GPIO_PIN_INT_FALLING_EDGE, QCOM_GPIO_PIN_INT_BOTH_EDGE, QCOM_GPIO_PIN_INT_LEVEL_LOW, QCOM_GPIO_PIN_INT_LEVEL_HIGH, } QCOM_GPIO_PIN_INTERRUPT_MODE;
Return Value	A_OK on success, A_ERROR on failure (for example the given hardware pin is not in GPIO mode)	

qcom_gpio_apply_peripheral_configuration

Definition	This function obtains the GPIO pin map for the given peripheral ID using the GPIO active configuration set (for enabling) and GPIO inactive configuration set (for disabling) and configures the pins to the requested power state. If active configuration is not found for the requested peripheral ID, this API returns an error.	
Prototype	A_STATUS qcom_gpio_apply_peripheral_configuration(	
	A_UINT32 peripheral_id,	Peripheral ID to be configured to a specific mode. Refer to qcom_gpio.h for the list of peripheral IDs.
	A_BOOL low_power_mode)	Specifies the mode for the given peripheral ID. A_TRUE = Set the pins in inactive configuration. A_FALSE = Set the pins in active configuration.
Return Value	A_OK on success, A_ERROR or A_HW_CONFIG_ERROR on failure.	

	If A_HW_CONFIG_ERROR is returned, the application or caller can invoke qcom_gpio_peripheral_pin_conflict_check() to get the conflicting peripherals that caused the error.

I²C API

qcom_i2cm_init

Definition	Initialize I²C master bus.	
Prototype	A_STATUS qcom_i2cm_init (	
	I2C_PIN_PAIR pin_pair_id	I²C sda/sck GPIO pin pair ID. Range: 0-4
	I2C_FREQ freq	I²C clock frequency. Range: 0-9
	A_UINT32 timeout)	Clock stretch timeout. Range: 0-1285000 µs depending on frequency. 0 for default value: 35000 µs
Return Value	0 on success or error code on failure	

qcom_i2cm_fini

Definition	This function is used to de-initialize I²C module in I²C fd.	
Prototype	A_INT32 qcom_i2cm_fini (	
	I2C_PIN_PAIR pin_pair_id)	I²C sda/sck GPIO pin pair ID
Return Value	0 on success or error code on failure	
Implantation	Restore setting, disable module NOTE: This API does not work before calling corresponding qcom_i2c_init.	

qcom_i2cm_read

Definition	This function is used to I²C master read control.

Prototype	A_STATUS qcom_i2cm_read (	
	I2C_PIN_PAIR pin_pair_id	I²C sda/sck GPIO pin pair ID. Range: 0-4
	A_UINT8 dev_addr	I²C device ID, unique 7-bit address
	A_UINT32 word_addr	word/reg address of I²C device to read. Ignore it in simple read format
	A_UINT8 addr_len	Length of word_addr data, 1-6 byte is preferred to combined format, 0 for simple read format
	A_UCHAR *data	The buffer pointer to the Rx data
	A_UINT32 data_len	Length of Rx chunk data, 0~8 byte is preferred
Return Value	0 on success, -1 on failure, -2 on timeout	

qcom_i2cm_write

Definition	This function is used to I²C write control.	
Prototype	A_INT32 qcom_i2cm_write (	
	I2C_PIN_PAIR pin_pair_id	I²C sda/sck GPIO pin pair ID. Range: 0-4
	A_UINT8 dev_addr	I²C device ID, unique 7-bit address
	A_UINT32 word_addr	word/reg address of I²C device to read. Ignore it in simple write format
	A_UINT8 addr_len	Length of word_addr data, 1-7 byte is preferred to combined format, 0 for simple write format
	A_UCHAR *data	The buffer pointer to the data to write
	A_UINT32 data_len	Length of write data, 1-8 byte is preferred
Return Value	0 on success, -1 on failure, -2 on timeout	

qcom_i2cs_csr_init

Definition	This function is used to initialize I²C slave csr module (single byte read/write)	
Prototype	A_INT32 qcom_i2cs_csr_init(	
	I2CS_CSR_PARA *csr_params)	CSR service registration parameters
Return Value	0 on success or error code on failure	

qcom_i2cs_reg_init

Definition	This function is used to initialize I²C slave register module (4 byte read/write)	
Prototype	A_INT32 qcom_i2cs_reg_init(	
	I2CS_REG_PARA *reg_params	Register service registration parameters
Return Value	0 on success or error code on failure	

qcom_i2cs_fifo_init

Definition	This function is used to initialize I²C slave fifo module(page read/write)	
Prototype	A_INT32 qcom_i2cs_fifo_init(	
	I2CS_FIFO_PARA *fifo_params	FIFO service registration parameters
Return Value	0 on success or error code on failure	

qcom_i2cs_control

Definition	This function is used to initialize I²C slave module	
Prototype	A_INT32 qcom_i2cs_control (	
	int enable	Enable/disable module
	I2CS_PIN_PAIR pin_pair_id)	I²C slave sda/sck GPIO pin pair ID. Range: 0-1
Return Value	0 on success or error code on failure	

qcom_i2cs_cmd

Definition	This function is used to send command to I²C slave module	
Prototype	A_INT32 qcom_i2cs_cmd (	
	A_UINT32 cmd	I²C slave command types
	A_UINT32 *data	I²C slave data used for both user to service and service to user info exchange
Return Value	0 on success or error code on failure	

I²S API

qcom_i2s_init

Definition	Initialize I²S function. This API can be used to: ✓ Initialize I²S interface 0/1, and select the frequency. ✓ Call the callback function in the descriptor when the receive interrupt is triggered. ✓ Use qcom_i2s_rcv_contrl to start and stop the receive DMA	
Prototype	A_INT32 qcom_i2s_init (	
	typedef struct { A_UINT32 port; I2S_FREQ freq; A_UINT8 dsize; A_UINT32 i2s_buf_size; A_UINT32 num_rx_desc; A_UINT32 num_tx_desc; A_UINT8 i2s_mode; } i2s_app_config;	Port: 0 = I2S0, 1 = I2S1 Freq: typedef enum { I2S_FREQ_44_1K = 0, I2S_FREQ_48K= 1, I2S_FREQ_32K = 2, I2S_FREQ_96K = 3 }I2S_FREQ; Dsize: The word size of each channel (8, 16, 24, 32 bits).. Tx buffer count in DMA. Buffer size over 4 is recommended. Rx buffer count in DMA. Buffer size over 4 is recommended. Buffer size. 1024 is recommended. i2s_mode: 1 = master mode; 0 = slave mode.
	void *intr_cb_txcomp	void intr_cb_txcomp (void *) This callback is used when I2S DMA transmit is completed. The user can continue to write DMA when this interrupt is received, otherwise it means DMA is not finished. If this parameter is NULL, the user will use the polling method to transmit the data.
	void *intr_cb_rx	void intr_cb_rx (void *) This callback is used in ISR mode, e.g., it calls WMI interface to release a signal to notify or wake some thread. The user then calls qcom_i2s_rcv_data(...) to receive the data. If this parameter is NULL, the user will use the polling method to receive the data.
Return Value	成功返回 A_OK，失败返回 A_ERROR	

qcom_i2s_xmt_data

Definition	Transmit data to an I²S interface. This API sends the data to I²S interface from high level.	
Prototype	A_UINT32 qcom_i2s_xmt_data (	
	A_UINT8 port,	0 = I²S0, 1 = I²S1
	A_UINT8 *data,	Data buffered to be transmitted to the I²S interface .
	A_UINT32 *len	Data length
Return Value	Greater than 0 = Data remaining to be processed 0 = No data remaining to be processed	

qcom_i2s_rcv_data

Definition	Receive data from an I²S interface. This API gets the data in I²S functionality from high level.	
Prototype	A_UINT32 qcom_i2s_rcv_data (	
	A_UINT8 port,	0 = I²S0, 1 = I²S1
	A_UINT8 *data,	Data buffered from the I²S interface
	A_UINT32 bufLen,	Maximum data buffer length. Usually 1k.
	A_UINT32 *len	Valid data length returned
Return Value	1 = Data remaining to be received 0 = No data remaining to be received	

qcom_i2s_rcv_control

Definition	Start processing data from an I²S interface. This API is used to start or stop the DMA channel to get data from the I²S interface.	
Prototype	void qcom_i2s_rcv_control (	
	A_UINT8 port,	0 = I²S0, 1 = I²S1
	I2S_REV_CONTROL control)	typedef enum { I2S_RCV_START = 0, I2S_RCV_STOP = 1, }I2S_RCV_CONTROL;
Return Value	N/A	

IrDA API

ir_tx_pin

void ir_tx_pin(int pin);

ir_send

void ir_send(unsigned char addr1,unsigned char addr2, unsigned char key);

PWM API

qcom_pwm_port_set

Definition	Configure frequency, duty cycle and dimming level of a given set of PWM function .	
Prototype	void qcom_pwm_port_set(	
	A_UINT32 freq,	Configure the frequency, has a maximum precision of 0.01Hz 10000 equal to 100.00Hz
	A_UINT32 duty,	Range: 0-10000; for duty cycle of percentage format, has a maximum precision of 0.01% ,10000 means 100.00%
	A_UINT32 phase,	Range: 0-10000; for phase shift of percentage format, has a maximum precision of 0.01% ,10000 equal to 100.00% .
	A_UINT32 pwm_id,	Indicate which channel of PWM to be configured. Should not be larger than the MACRO PWM_MAX_CHANNEL
	A_UINT32 src_clk)	Range: 0-2. Source clock. 0 = CPU_CLK 1 = REF_CLK

		2 = LPO_CLK
Return Value	A_OK / A_ERROR	

qcom_pwm_sdm_set

Definition	Configure frequency, duty cycle and dimming level of a given set of PWM function.	
Prototype	void qcom_pwm_set(	
	A_UINT32 freq,	Configure the frequency, unit: KHz
	A_UINT32 duty,	Range: 0-255; for duty cycle of percentage format, divided by 256.
	A_UINT32 pwm_id,	Indicate which channel of PWM to be configured. It should not be larger than the MACRO PWM_MAX_CHANNEL.
Return Value	A_OK / A_ERROR	
Implantation	Set and calculate parameters NOTE: If a very stable and reliable driving waveform is required, call qcom_pwm_sdm_set before enabling the port (call qcom_pwm_port_control(1,1,port_id)) when initializing a port, so the PWM wave will not generate a spur at the beginning.	

qcom_pwm_control

Definition	Start/stop one set of PWM function.	
Prototype	A_STATUS qcom_pwm_control(	
	A_BOOL module_select,	0 = PWM port module 1 = sigma delta module
	A_BOOL enable	TRUE: enable PWM output in port_id channel in module_select mode FALSE: disable PWM output in port_id channel in module_select mode
	A_UINT32 port_group)	Indicate which channel of PWM to be configured. Set it based on MACRO PWM_PORT_n
Return Value	A_OK / A_ERROR	
Implantation	Enable/disable the SDM/PWM PORT of port_id NOTE: sigma delta module is not supported in sleep mode	

SPI API

qcom_spi_init

Definition	Initiate legacy SPI interface.	
Prototype	int qcom_spi_init (	
	SPI_CS_PIN cs_pin	SPI chip select pin
	SPI_MOSI_PIN mosi_pin	SPI MOSI pin
	SPI_MISO_PIN miso_pin	SPI MISO pin
	SPI_SCLK_PIN sclk_pin	SPI sclk pin
	SPI_FREQ freq	SPI frequency
Return Value	A_OK / A_ERROR	

qcom_spi_fini

Definition	Restore GPIO pins used by SPI and disable SPI function.	
Prototype	void qcom_spi_fini ()	
Return Value	void	

qcom_spi_request

Definition	Initiate SPI session.	
Prototype	void qcom_spi_request (	
	A_UINT32 *req	Buffer pointer to be transfered
	unsigned int rxcnt	The number of bytes (from 0 to 8) to receive on the SPI
	unsigned int txcnt)	The number of bytes (from 0 to 8) to send on the SPI
Return Value	void	

qcom_spi_response

Definition	Read from the attached SPI device.	
Prototype	void qcom_spi_response (	
	A_UINT8 *buf	Buffer used for receiving data
	unsigned int nbytes)	The number of bytes (up to 8)
Return Value	void	

UART API

qcom_uart_init

Definition	Initiate UART.	
Prototype	A_UINT32 qcom_uart_init()	
Return Value	Number of UART.	

qcom_set_uart_config

Definition	Set parameters of UART.	
Prototype	A_STATUS qcom_set_uart_config (	
	A_CHAR *name,	UART port name

	qcom_uart_para *uartpara)	typedef struct _uart_para
		{
		int BaudRate; // baud rate
		char number; // number of data bits
		char parity; // parity (0 = no parity; 1 = odd parity check; 2 = even parity check) default 0
		char StopBits; //number of stop bits
		char FlowControl; //1 = support flow control
		} qcom_uart_para;

Return Value	A_OK / A_ERROR	

qcom_get_uart_config

Definition	Get parameters of UART.	
Prototype	A_STATUS qcom_get_uart_config (	
	A_CHAR *name,	UART port name

	qcom_uart_para *uartpara)	typedef struct _uart_para
		{
		int BaudRate; // baud rate
		char number; // number of data bits
		char parity; // parity (0 = no parity; 1 = odd
		parity check; 2 = even parity check) default 0
		char StopBits; //number of stop bits
		char FlowControl; //1 = support flow control
		} qcom_uart_para;

Return Value	A_OK / A_ERROR	

qcom_uart_open

Definition	Open UART and get a file descriptor.	
Prototype	A_UINT32 qcom_uart_open (	
	A_CHAR *name)	UART port name
Return Value	File descriptor	

qcom_uart_close

Definition	Close UART	
Prototype	A_UINT32 qcom_uart_close (	
	A_INT32 fd	File descriptor returned from uart_open
Return Value	0 on success, other values on failure	

qcom_uart_read

Definition	Read data from UART.	
Prototype	A_UINT32 qcom_uart_read (	
	A_INT32 fd,	File descriptor returned from uart_open
	A_CHAR *buff,	Address of the buffer. The buffer size must be big enough for the data.
	A_INT32 *len)	Input length of the buffer. Output length of bytes that have been read.
Return Value	Bytes remain in the Rx buffer.	

qcom_uart_write

Definition	Write data to UART.	
Prototype	A_UINT32 qcom_uart_write (	
	A_INT32 fd,	File descriptor returned from uart_open
	A_CHAR *buff,	Address of the buffer. The buffer size must be big enough for the data.
	A_INT32 *len)	Input length of the buffer. Output length of bytes that have been read.
Return Value	Bytes remain in the Tx buffer.	

qcom_uart_rx_pin_set

Definition	Set UART Rx pin.	
Prototype	void qcom_uart_rx_pin_set (	
	int pin0,	Rx pin for UART0. Default: 2
	int pin1)	Rx pin for UART1. Default: 10
Return Value	void	

qcom_uart_tx_pin_set

Definition	Set UART Tx pin.	
Prototype	void qcom_uart_tx_pin_set (	
	int pin0,	Tx pin for UART0. Default: 7
	int pin1)	Tx pin for UART1. Default: 11
Return Value	void	

ADC API

qcom_adc_init

Definition	Open the ADC according to the specified parameters	
Prototype	int qcom_adc_init(	
	A_UINT8 gain_scale,	0: 1.8V 1: 3.3V
	A_UINT8 accuracy,	Sample accuracy, 6bit~12bit
	A_UINT8 freq,	Sample frequency, 0~7 0: 31.25KHz 1: 1MHz 2: 500KHz 3: 250KHz 4: 100KHz 5: 50KHz 6: 25KHz 7: 10KHz
	A_UINT8 input_type,	0: single ended input 1: differential
	A_BOOL stream_bit)	Default is 0, set to 1 when using voice stream
Return Value	A_OK / A_ERROR	

qcom_adc_config

Definition	Configure the ADC	
Prototype	int qcom_adc_config (	
	A_UINT8 scan_mode,	Scan Mode Enable 0: one-channel working

		1: multi-channel working
	A_UINT8 trigger_mode,	Conversion Trigger Select 0: software trigger 1: hardware timer trigger 2: hardware pin trigger
	A_UINT8 adco_mode,	ADC work mode 0: sample-by-sample(single)(static scenario) 1: continuous(dynamic scenario), if SCAN=1, repeat scan;
	ADC_CHAN_CFG_T channel[]	Channel configuration
	A_UINT8 channel_cnt)	Channel number
Return Value	A_OK / A_ERROR	

qcom_adc_timer_config

Definition	Configure the ADC timer when using timer trigger mode	
Prototype	int qcom_adc_timer_config (	
	A_UINT32 time_start,	Timer Start Value (default is 300ms) Sets the timer start value. The timer will count down until it reaches 0, then load this register value again and count down.
	A_UINT32 time_unit)	timer start value unit 0: unit is clock period(CLK=66M) 1: 1us 2: 1ms 3: 10ms
Return Value	A_OK / A_ERROR	

qcom_adc_dma_config

Definition	Configure the ADC DMA	
Prototype	int qcom_adc_dma_config (	
	A_UINT32 threshold,	threshold for channel data cnt
	A_UINT32 data_cnt)	Total data cnt of all channels for DMA, when exhaust will disable dma input and transfer. When set to 0, no limit for transfer data byte number.

Return Value	A_OK / A_ERROR

qcom_adc_calibration

Definition	Do the ADC calibration process	
Prototype	int qcom_adc_calibration ()	
Return Value	A_OK / A_ERROR	

qcom_adc_conversion

Definition	Start/stop adc conversion	
Prototype	int qcom_adc_conversion (	
	A_BOOL start)	1: start conversion 0: stop conversion
Return Value	A_OK / A_ERROR	

qcom_adc_recv_data

Definition	Get ADC conversion results	
Prototype	int qcom_adc_recv_data(	
	A_UINT8 channel,	ADC channel, 1-8
	A_UINT8* buffer,	Data buffer
	A_UINT32 buf_len,	Buffer length
	A_UINT32* ret_len,	Return length
	A_UINT8* more,	1: there is more data, 0: there is no data
	A_UINT8* done)	1: ADC conversion done
Return Value	A_OK / A_ERROR	

qcom_adc_close

Definition	Close ADC, set ADC to default settings

Prototype	int qcom_adc_close()	
Return Value	A_OK / A_ERROR	

TIMER API

qcom_timer_init

Definition	Initialize a timer.	
Prototype	int qcom_mac_get (	
	qcom_timer_t* qtimer	A pointer to qcom_timer_t for saving internal data of timer.
	void (*fn)(unsigned int, void *),	Call back function for the timer
	void* arg,	The parameters of call back function
	int timeout,	Timeout value for the timer in milliseconds.
	qcom_timer_type_e type)	Oneshot or Periodic.
Return Value	0 on success, else on error.	

qcom_timer_start

Definition	Start a timer.	
Prototype	int qcom_timer_start (	
	qcom_timer_t* qtimer)	A pointer to qcom_timer_t returned from qcom_timer_init.
Return Value	0 on success, else on error.	

qcom_timer_stop

Definition	Stop a timer.	
Prototype	int qcom_timer_stop (	
	qcom_timer_t* qtimer)	A pointer to qcom_timer_t returned from qcom_timer_init.
Return Value	0 on success, else on error.	

qcom_timer_delete

Definition	Delete a timer.	
Prototype	int qcom_timer_delete (	
	qcom_timer_t* qtimer)	A pointer to qcom_timer_t returned from qcom_timer_init.
Return Value	0 on success, else on error.	

qcom_thread_msleep

Definition	Thread sleep	
Prototype	void qcom_thread_msleep(	
	unsigned long ms)	Sleep time
Return Value	void	

Socket API

qcom_socket

Definition	Create a socket.	
Prototype	int qcom_socket (	
	int family,	Protocol family (AF_INET, AF_INET6)
	int type,	Socket type (SOCK_DGRAM or SOCK_STREAM)
	int protocol)	Protocol (usually 0)
Description	This call is blocked until a socket handle is returned. If no response is received or socket creation fails, error code is returned.	
Return Value	Socket handle on success, -1 on failure	

qcom_bind

Definition	Assign a local socket address to an unnamed socket.

Prototype	int qcom_bind(	
	int s,	Socket handler
	struct sockaddr* addr,	Sockaddr structure indicating the local address to bind with. The address is provided in the name field in the form of a sockaddr (or sockaddr_6) structure.
	int addrlen)	Length of sockaddr structure
Return Value	0 / -1	

qcom_listen

Definition	Listen on incoming connections.	
Prototype	int qcom_listen(	
	int s,	Socket handler
	int backlog)	Maximum length of pending connections
Description	If a connection request arrives, a qcom_accept call is triggered to accept it. qcom_listen is used only with socket type SOCK_STREAM. The backlog parameter defines the maximum length of the queue for pending connections. If a connection request arrives while the queue is full, the client receives error code ECONNREFUSED.	
Return Value	0 / -1	

qcom_accept

Definition	Accept incoming connections on an open socket.	
Prototype	int qcom_accept(	
	int s,	Socket handler
	struct sockaddr* addr,	Sockaddr structure indicating the local address to accept
	int *addrlen)	Length on sockaddr structure
Description	After qcom_listen() returns success (socket created and bound to an address), this API extracts the first connection from the queue of pending connections. This API is used only with socket type SOCK_STREAM. The call is blocked if no connection is present or the socket is blocking.	
Return Value	Non-negative socket descriptor on success, -1 on error	

qcom_connect

Definition	Connect to a socket.	
Prototype	int qcom_connect(	
	int s,	Socket handler

	struct sockaddr* addr,	struct sockaddr
		{
		u_short sa_family; /* address family */
		#ifdef IP_V6 /* V6 only or dual stacks */
		char sa_data[32]; /* big enough for unpacked sockaddr_in6 */
		#else /* Ancient IPv4 only version */
		char sa_data[14]; /* up to 14 bytes of direct address */
		#endif
		};

	int *addrlen)	Length on sockaddr structure
Description	If the socket type is UDP, specify the peer to be associated with, the address to which datagrams are sent, and the address from which datagrams are received. If the socket type is TCP, this call attempts to connect to another socket. The call is blocked until a connection is established or error is returned.	
Return Value	0 / -1	

qcom_setsockopt

Definition	Set options for an existing socket.	
Prototype	int qcom_setsockopt(	
	int s,	Socket handler
	int level,	SOL_SOCKET
	int name,	TCP_MAXSEG IP_ADD_MEMBERSHIP IP_DROP_MEMBERSHIP
	void* arg,	Option value
	int arglen)	Option length
Return Value	0 / -1	

qcom_getsockopt

Definition	Get options of an existing socket.	
Prototype	int qcom_getsockopt(	
	int s,	Socket handler
	int level,	SOL_SOCKET
	int name,	TCP_MAXSEG IP_ADD_MEMBERSHIP IP_DROP_MEMBERSHIP
	void* arg,	Option value
	int arglen)	Option length
Return Value	0 / -1	

qcom_recv

Definition	Receive data on a stream socket.	
Prototype	int qcom_recv(	
	int s,	Socket handler
	char* buf,	A pointer to data
	int len,	Payload length
	int flags)	Send flags
Description	The socket must be in connected state for receive operation. If no packet is available, the socket is blocked until it is set to non-blocking. The application must provide a valid buffer to receive the payload. If the buffer length is smaller than the available payload, the API will do a partial copy, and hold on the rest of the payload for a subsequent call to the API.	
Return Value	Number of bytes received on success, -1 on error	

qcom_recvfrom

Definition	Receive data on a datagram socket.	
Prototype	int qcom_recvfrom(	
	int s,	Socket handler
	char* buf,	A pointer to data

	int len,	Payload length
	int flags,	Send flags
	struct sockaddr* from,	Sockaddr structure (v4 or v6)
	int *fromlen)	Length of sockaddr
Description	If no packet is available, the socket is blocked until it is set to non-blocking. The application must provide a sufficiently large buffer to receive the payload. If the buffer length is smaller than the available payload, the API will do a partial copy and hold drop the rest of the payload.	
Return Value	Number of bytes received on success, -1 on error	

qcom_send

Definition	Send data on a stream socket.	
Prototype	int qcom_send(	
	int s,	Socket handler
	char* buf,	A pointer to data
	int len,	Payload length
	int flags)	Send flags, usually 0
Description	This API is used to send data on a stream socket in connected state. The API works for both IPv4 and IPv6.	
Return Value	On success: Number of bytes transmitted to the target. On failure: -100: No Tx buffers available. Caller is required to sleep and retry. -1: Fatal error, or packet length exceeded 1576 bytes. Others: socket error number	

qcom_sendto

Definition	Send data on a datagram socket.	
Prototype	int qcom_sendto(	
	int s,	Socket handler
	char* buf,	A pointer to data
	int len,	Payload length
	int flags,	Send flags, usually 0
	struct sockaddr* to,	Sockaddr structure (v4 or v6)

	int tolen)	Length of sockaddr
Description	The destination address must be specified in the "to" parameter. This API works for both IPv4 and IPv6 with the sockaddr and sockaddr_6 structures respectively in the name parameter.	
Return Value	On success: Number of bytes transmitted to the target. On failure: 100: No Tx buffers available. Caller is required to sleep and retry. 1: Fatal error, or packet length exceeded 1576 bytes.	

qcom_socket_close

Definition	Close a socket.	
Prototype	int qcom_socket_close(	
	int s)	Socket handler
Description	If the socket does not exist, error is returned.	
Return Value	0 / -1	

Power Management API

qcom_power_set_mode

Definition	Set power mode.	
Prototype	A_STATUS qcom_power_set_mode (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT32 powerMode)	1 = REC_POWER. Conserve power. 2 = MAX_PERF_POWER. Maximum performance. Other values = Reserved
Description	In the conserve power mode (recommended), the device enters sleep state during idle period to conserve power. Performance is impacted by this mechanism because it takes up to 2 ms to resume operation after exiting sleep state. In the maximum performance mode, the device never enters sleep state (no need to wakeup) to achieve better performance at the cost of higher power consumption.	

Return Value	A_OK / A_ERROR

qcom_power_get_mode

Definition	Get power mode.	
Prototype	A_STATUS qcom_power_get_mode (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT32 *powerMode)	A pointer to power mode
Description	In the conserve power mode (recommended), the device enters sleep state during idle period to conserve power. Performance is impacted by this mechanism because it takes up to 2 ms to resume operation after exiting sleep state. In the maximum performance mode, the device never enters sleep state (no need to wakeup) to achieve better performance at the cost of higher power consumption.	
Return Value	A_OK / A_ERROR	

qcom_suspend_enable

Definition	Enable suspend function.	
Prototype	A_STATUS qcom_suspend_enable (	
	A_BOOL enable)	1 = Enable; 0 = Disable.
Return Value	A_OK / A_ERROR	

qcom_suspend_start

Definition	Start the suspend function and restore after the specified time.	
Prototype	A_STATUS qcom_suspend_start (	
	A_UINT32 time)	Suspend period in seconds
Return Value	A_OK / A_ERROR	

qcom_power_set_parameters

Definition	Set power parameters.	
Prototype	A_STATUS qcom_power_set_parameters (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT16 idlePeriod,	Period in ms for the device to remain awake after frame transmit/receive is complete and before entering SLEEP state
	A_UINT16 psPollNum,	The number of PS-poll messages sent by the device before notifying the AP it is awake
	A_UINT16 dtimPolicy,	WMI DTIM policy 1 = Ignore DTIM. The device does not listen to any contents after Beacon (CAB). 2 = Normal DTIM. DTIM period follows the listen interval. For example, if the listen interval is 4 and the DTIM period is 2, the device wakes up every fourth beacon. 3 = Stick DTIM. The device attempts to receive all CAB traffic. For example, if the DTIM period is 2 and the listen interval is 4, the device wakes up every second beacon. 4 = Auto DTIM (decided by device)
	A_INT16 tx_wakeup_policy,	WMI Tx wakeup policy 1= Wakeup upon uplink traffic. The number of uplink frames in a Beacon interval to trigger the transition to awake is determined by NUM_TX_TO_WAKEUP. 2 = Not wakeup upon uplink traffic
	A_UINT16 num_tx_to_wakeup,	
	A_UINT16 ps_fail_event_policy)	Power save fail event policy 1 = Any null frame with PM = 1. Tx failure is notified by the WMI_TARGET_PM_ERR_FAIL event. 2 = Ignore null frame with PM = 1 (Tx failures during scan).
Return Value	A_OK / A_ERROR	

qcom_wlan_power_wakeup_start

int qcom_wlan_power_wakeup_start(A_UINT32 ms);

qcom_wlan_power_wakeup_stop

int qcom_wlan_power_wakeup_stop(void);

qcom_wlan_wakeup_gpio_handler

void qcom_wlan_wakeup_gpio_handler(void *pcontext);

qcom_wlan_wakeup_gpio_initconfig

A_STATUS qcom_wlan_wakeup_gpio_initconfig(void);

NVRAM API

qcom_board_type_get

int qcom_board_type_get(void);

qcom_board_cfg_get

int qcom_board_cfg_get(void);

qcom_nvram_mode_get

A_INT32 qcom_nvram_mode_get(void);

qcom_nvram_size_get

A_INT32 qcom_nvram_size_get();

qcom_nvram_init

void qcom_nvram_init(void);

qcom_nvram_fini

void qcom_nvram_fini(void);

qcom_nvram_read

A_STATUS qcom_nvram_read(A_UINT32 offset, A_UCHAR *buf, A_UINT32 size);

qcom_nvram_erase

A_STATUS qcom_nvram_erase(A_UINT32 offset, A_UINT32 size);

qcom_nvram_write

A_STATUS qcom_nvram_write(A_UINT32 offset, A_UCHAR *buf, A_UINT32 size);

qcom_nvram_select

A_INT32 qcom_nvram_select(A_INT32 partition);

qcom_nvram_read_partition_word

A_INT32 qcom_nvram_read_partition_word(int partition, unsigned int offset, A_UINT32 *buf);

qcom_nvram_read_partition_age

A_INT32 qcom_nvram_read_partition_age(int partition,A_UINT32 *buf);

Encryption API

qcom_sec_md5_init

void qcom_sec_md5_init();

qcom_sec_md5_update

void qcom_sec_md5_update(unsigned char *data, int len);

qcom_sec_md5_final

void qcom_sec_md5_final(char md5_val[]);

qcom_aes_encrypt_init

void qcom_aes_encrypt_init(const unsigned char *key, int len);

qcom_aes_encrypt

void qcom_aes_encrypt(void *ctx, const unsigned char *plain, unsigned char *crypt);

qcom_aes_encrypt_deinit

void qcom_aes_encrypt_deinit(void *ctx);

qcom_aes_decrypt_init

void qcom_aes_decrypt_init(const unsigned char *key, int len);

qcom_aes_decrypt

void qcom_aes_decrypt(void *ctx, const unsigned char *crypt, unsigned char *plain);

qcom_aes_decrypt_deinit

void qcom_aes_decrypt_deinit(void *ctx);

Watchdog API

qcom_sys_reset

Definition	Reset the system.	
Prototype	A_STATUS qcom_sys_reset ()	
Return Value	A_OK / A_ERROR	

qcom_watchdog_feed

Definition	Reset watchdog timer.	
Prototype	void qcom_watchdog_feed ()	
Return Value	void	

qcom_watchdog

Definition	Enable/disable a watchdog timer.	
Prototype	int qcom_watchdog (	
	int enable,	APP_WDT_DISABLE – disable watchdog APP_WDT_ENABLE – enable watchdog
	int timeout)	Watidog timer expiry timeout in seconds
Description	If the watchdog timer is enabled, it will be reset by an internal maintainence timer	

	every 5 seconds. The developer can use the qcom_watchdog_feed() API to reset the timer from application, if desired.
Return Value	0

MISC API

qcom_mac_get

Definition	Get QCA4010/QCA4012 MAC address.	
Prototype	A_STATUS qcom_mac_get (	
	A_UINT8 device_id,	Interface ID (default is 0)
	A_UINT8 *pmac)	A pointer to MAC address
Return Value	A_OK / A_ERROR	

qcom_free_heap_size_get

A_STATUS qcom_free_heap_size_get(A_UINT32 *pSize);

qcom_buffer_info_get

Definition	Get the buffer statistics for system debug.	
Prototype	A_STATUS qcom_buffer_info_get(	
	A_UINT8 *pmac_cnt,	Rx free buffer count
	A_UINT8 *pmac_reap,	Rx used buffer count
	A_UINT8 *phtc_cnt,	Tx free buffer count
	A_UINT8 *phtc_reap,	Tx used buffer count
	A_UINT8 *pfw_cnt,	Internal free buffer count
	A_UINT8 *pfw_reap,	Internal used buffer count
	A_UINT8 *pfree_cnt,	Buffer left in the buffer pool
	A_UINT8 *phtc_get,	Tx buffer allocation statistics
	A_UINT8 *phtc_put)	Unused
Return Value	A_OK / A_ERROR	

qcom_promiscuous_enable

Definition	Enable promiscuous mode.	
Prototype	A_STATUS qcom_promiscuous_enable (	
	A_BOOL promiscuousEn)	TRUE = Enable promiscuous mode. FALSE = Disable promiscuous mode.
Return Value	A_OK / A_ERROR	

qcom_set_promiscuous_rx_cb

Definition	Set the call back function for promiscuous mode.	
Prototype	A_STATUS qcom_set_promiscuous_rx_cb (	
	WLAN_PROMISCUOUS_CB cb,	typedef void (*WLAN_PROMISCUOUS_CB)(unsigned char *buf, int length); cb is a hook function to process received packets of type WLAN_PROMISCOUS_CB. Since the hook function runs in the target domain, user must not call ThreadX API or Driver APIs in hook function.
Return Value	A_OK	

qcom_firmware_version_get

Definition	Get the firmware version.	
Prototype	A_STATUS qcom_firmware_get(	
	A_UINT32 *psize	A pointer to the free heap size
Return Value	A_OK / A_ERROR	

DataSet API

qcom_dset_create

Definition	Create data set on indicated media.	
Prototype	A_STATUS qcom_dset_create (	
	DSET_HANDLE *dset_handle,	Storage address for return data set handle. Default = 0.
	DSET_ID dset_id,	Data set ID
	uint32_t length,	Data set length
	uint32_t flags,	Media ID on which the data set is to be created. Default = 0 Media IDs: DSET_MEDIA_HOST 1 //host mode only DSET_MEDIA_RAM 2 DSET_MEDIA_NVRAM 3 DSET_MEDIA_OTP 4 //very small, one-time programming DSET_MEDIA_ROM 5 DSET_MEDIA_MEM 6 // The combination of (DSET_MEDIA_RAM + DSET_MEDIA_ROM) DEST_MEDIA_AORAM 7
	DSET_CB create_cb,	Asynchronous callback function. May be NULL.
	void *cb_arg)	Callback function parameter. May be NULL.
Return Value	A_OK / A_ERROR	

qcom_dset_open

Definition	Open data set for access.	
Prototype	A_STATUS qcom_dset_open (	
	DSET_HANDLE *dset_handle,	Storage address for return data set handle. Default = 0.
	DSET_ID dset_id,	Data set ID
	uint32_t flags,	If the flag is greater or less than 0, this is the start media ID the open operation will be performed on. Otherwise the open operation will traverse across all

		supported media. Media ID and traverse order: DSET_MEDIA_HOST 1 //host mode only DSET_MEDIA_RAM 2 DSET_MEDIA_NVRAM 3 DSET_MEDIA_OTP 4 //very small, one-time programming DSET_MEDIA_ROM 5 DSET_MEDIA_MEM 6 // The combination of (DSET_MEDIA_RAM + DSET_MEDIA_ROM) DEST_MEDIA_AORAM 7
	DSET_CB open_cb,	Asynchronous callback function. May be NULL.
	void *cb_arg)	Callback function parameter. May be NULL.
Return Value	A_OK / A_ERROR	

qcom_dset_read

Definition	Read data from data set to buffer.	
Prototype	A_STATUS qcom_dset_read (	
	DSET_HANDLE *dset_handle,	Data set handle
	uint8_t *buffer,	Data buffer
	uint32_t length,	Data length to read
	uint32_t offset,	Offset in data set to start reading. Default = 0
	DSET_CB read_cb,	Asynchronous callback function. May be NULL.
	void *cb_arg	Callback function parameter. May be NULL.
Return Value	A_OK = Read operation complete. Data in buffer ready for use. A_FAIL = Read operation failed. Data in buffer invalid. A_PENDING = Asynchronous operation. Data in buffer is not ready for use.	

qcom_dset_write

Definition	Write data to data set.	
Prototype	A_STATUS qcom_dset_write (	
	DSET_HANDLE *dset_handle,	Data set handle
	uint8_t *buffer,	Data buffer

	uint32_t length,	Data length to write
	uint32_t offset,	Offset in data set to start writing. Default = 0
	uint32_t flags,	Only support circular operation. Default = 0
	DSET_CB write_cb,	Asynchronous callback function. May be NULL.
	void *cb_arg	Callback function parameter. May be NULL.
Return Value	A_OK, A_FAIL, A_PENDING	

qcom_dset_close

Definition	Close the opened data set and release resource.	
Prototype	A_STATUS qcom_dset_close (	
	DSET_HANDLE *dset_handle,	Data set handle
	DSET_CB close_cb,	Asynchronous callback function. May be NULL.
	void *cb_arg)	Callback function parameter. May be NULL.
Return Value	A_OK, A_FAIL, A_PENDING	

qcom_dset_commit

Definition	Commit cached data after data set is created on media.	
Prototype	A_STATUS qcom_dset_commit (	
	DSET_HANDLE *dset_handle,	Data set handle
	DSET_CB commit_cb,	Asynchronous callback function. May be NULL.
	void *cb_arg)	Callback function parameter. May be NULL.
Return Value	A_OK, A_FAIL, A_PENDING	

qcom_dset_delete

Definition	Delete data set from media.	
Prototype	A_STATUS qcom_dset_delete (	
	DSET_ID dset_id,	Data set ID
	A_UINT32 flags,	The media ID that the data set resides on. Media ID and traverse order:

		DSET_MEDIA_HOST 1 //host mode only DSET_MEDIA_RAM 2 DSET_MEDIA_NVRAM 3 DSET_MEDIA_OTP 4 //very small, one-time programming DSET_MEDIA_ROM 5 DSET_MEDIA_MEM 6 // The combination of (DSET_MEDIA_RAM+ DSET_MEDIA_ROM) DEST_MEDIA_AORAM 7
	DSET_CB delete_cb,	Asynchronous callback function. May be NULL.
	void *cb_arg)	Callback function parameter. May be NULL.
Return Value	A_OK, A_FAIL, A_PENDING	

qcom_dset_size

Definition	Return data set size.	
Prototype	A_UINT32 qcom_dset_size (	
	DSET_HANDLE *dset_handle)	Data set handle
Return Value	Data set size	

qcom_dset_media

Definition	Get media ID that the data set resides on.	
Prototype	A_UINT32 qcom_dset_media (	
	DSET_HANDLE *dset_handle)	Data set handle
Return Value	Data set media ID.	

Crypto API

qcom_crypto_obj_usage_restrict

A_CRYPTO_STATUS qcom_crypto_obj_usage_restrict(qcom_crypto_obj_hdl_t hdl, A_UINT32 obj_usage);

qcom_crypto_transient_obj_reset

A_CRYPTO_STATUS qcom_crypto_transient_obj_reset(qcom_crypto_obj_hdl_t hdl);

qcom_crypto_obj_info_get

A_CRYPTO_STATUS qcom_crypto_obj_info_get(qcom_crypto_obj_hdl_t hdl, qcom_crypto_obj_info_t *info);

qcom_crypto_transient_obj_alloc

A_CRYPTO_STATUS qcom_crypto_transient_obj_alloc(A_UINT32 obj_type, A_UINT32 max_key_size, qcom_crypto_obj_hdl_t *obj);

qcom_crypto_transient_obj_keygen

A_CRYPTO_STATUS qcom_crypto_transient_obj_keygen(qcom_crypto_obj_hdl_t hdl, A_UINT32 key_size,
qcom_crypto_attrib_t *attrs, A_UINT32 attr_count);

qcom_crypto_transient_obj_populate

A_CRYPTO_STATUS qcom_crypto_transient_obj_populate(qcom_crypto_obj_hdl_t hdl, qcom_crypto_attrib_t *attrs, A_UINT32 attr_count);

qcom_crypto_transient_obj_free

A_CRYPTO_STATUS qcom_crypto_transient_obj_free(qcom_crypto_obj_hdl_t hdl);

qcom_crypto_op_info_get

A_CRYPTO_STATUS qcom_crypto_op_info_get(qcom_crypto_obj_hdl_t hdl, qcom_crypto_op_info_t *info);

qcom_crypto_op_alloc

A_CRYPTO_STATUS qcom_crypto_op_alloc(A_UINT32 alg, A_UINT32 mode, A_UINT32 max_key_size, qcom_crypto_op_hdl_t *hdl);

qcom_crypto_op_free

A_CRYPTO_STATUS qcom_crypto_op_free(qcom_crypto_op_hdl_t hdl);

qcom_crypto_op_reset

A_CRYPTO_STATUS qcom_crypto_op_reset(qcom_crypto_op_hdl_t hdl);

qcom_crypto_op_key_set

A_CRYPTO_STATUS qcom_crypto_op_key_set(qcom_crypto_op_hdl_t op_hdl, qcom_crypto_obj_hdl_t obj_hdl);

qcom_crypto_op_copy

A_CRYPTO_STATUS qcom_crypto_op_copy(qcom_crypto_op_hdl_t dst_hdl, qcom_crypto_op_hdl_t src_hdl);

qcom_crypto_op_verify_digest

A_CRYPTO_STATUS qcom_crypto_op_verify_digest(qcom_crypto_op_hdl_t hdl, qcom_crypto_attrib_t *params, A_UINT32 param_count,
void *digest, A_UINT32 digest_len, void *sign, A_UINT32 sign_len);

qcom_crypto_op_sign_digest

A_CRYPTO_STATUS qcom_crypto_op_sign_digest(qcom_crypto_op_hdl_t hdl, qcom_crypto_attrib_t *params, A_UINT32 param_count,
void *digest, A_UINT32 digest_len, void *sign, A_UINT32 *sign_len);

qcom_crypto_op_cipher_init

A_CRYPTO_STATUS qcom_crypto_op_cipher_init(qcom_crypto_op_hdl_t hdl, void *IV, A_UINT32 IVLen);

qcom_crypto_op_cipher_update

A_CRYPTO_STATUS qcom_crypto_op_cipher_update(qcom_crypto_op_hdl_t hdl, void *srcData, A_UINT32 srcLen,

void *destData, A_UINT32 *destLen);

qcom_crypto_op_cipher_dofinal

A_CRYPTO_STATUS qcom_crypto_op_cipher_dofinal(qcom_crypto_op_hdl_t hdl, void *srcData, A_UINT32 srcLen, void *destData, A_UINT32 *destLen);

qcom_crypto_obj_buf_attrib_get

A_CRYPTO_STATUS qcom_crypto_obj_buf_attrib_get(qcom_crypto_obj_hdl_t hdl, A_UINT32 attrib_id, void *buffer, A_UINT32 size);

qcom_crypto_obj_val_attrib_get

A_CRYPTO_STATUS qcom_crypto_obj_val_attrib_get(qcom_crypto_obj_hdl_t hdl, A_UINT32 attrib_id, A_UINT32 *a, A_UINT32 *b);

qcom_crypto_rng_get

A_CRYPTO_STATUS qcom_crypto_rng_get(void *buffer, A_UINT16 len);

Network API

qcom_ipconfig

Definition	Set/get IPv4 parameters, or trigger DHCP client.	
Prototype	A_STATUS qcom_ipconfig (	
	A_UINT8 device_id,	Interface ID, 0 or 1 (default 0)
	A_INT32 mode,	Command mode. Possible values- IP_CONFIG_QUERY (0) – get IP v4 parameters IP_CONFIG_SET (1) – assign static IP v4 parameters IP_CONFIG_DHCP (2) – Run DHCP client
	A_UINT32 *address,	IPv4 address
	A_UINT32 *submask,	IPv4 subnet mask

	A_UINT32 *gateway)	IPv4 gateway address
Return Value	A_OK / A_ERROR	

qcom_dns_server_address_get

Definition	Get DNS address from DHCP result.	
Prototype	A_STATUS qcom_dns_server_address_get (	
	A_UINT32 *pdns	Pointer to array where the DNS addresses are to be copied. Caller must ensure that the array size is equal to or greater than 3.
	A_UINT32* number	Number of addresses
Return Value	A_OK / A_ERROR	

qcom_ping

Definition	Send an IPv4 ping. The Host is blocked until a result is received from the Target.	
Prototype	A_STATUS qcom_ping (	
	A_UINT32 host,	IPv4 destination address
	A_UINT32 size)	Size of ping payload in bytes
Return Value	A_OK / A_ERROR	

qcom_ping6

Definition	Send an IPv6 ping. The Host is blocked until a result is received from the Target.	
Prototype	A_STATUS qcom_ping6 (	
	A_UINT8 *host,	IPv6 destination address
	A_UINT32 size)	Size of ping payload in bytes
Return Value	A_OK / A_ERROR	

qcom_ip6config_router_prefix

Definition	Set the IPv6 router prefix (SoftAP mode only)	
Prototype	A_STATUS qcom_ip6config_router_prefix (	
	A_UINT8 device_id,	Interface ID, 0 or 1 (default 0)
	A_UINT8* addr,	Prefix
	A_INT32 prefix_length,	Length of prefix
	A_INT32 prefix_lifetime,	Lifetime of prefix
	A_INT32 valid_lifetime)	
Return Value	A_OK / A_ERROR	

qcom_tcp_set_exp_backoff

Definition	Set the maximum number of TCP backff retries.	
Prototype	A_STATUS qcom_tcp_set_exp_backoff (	
	A_UINT8 device_id,	Interface ID, 0 or 1 (default 0)
	A_INT32 retries)	Number of retires (4 – 12)
Return Value	A_OK / A_ERROR	

qcom_ip4_route

Definition	Add, delete or query IPv4 route.	
Prototype	A_STATUS qcom_ip4_route (	
	A_UINT8 device_id,	Unused (default 0)
	A_UINT32 cmd,	Cmd – ROUTE_ADD (0) ROUTE_DEL (1) ROUTE_SHOW (2)
	A_UINT32* addr,	Pointer to IPv4 Address
	A_UINT32* subnet,	Pointer to subnet mask
	A_UINT32* gw,	Pointer to Gateway IPv4 address
	A_UINT32* ifindex,	Pointer to interface index
	IPV4_ROUTE_LIST_T* rtlist)	List of routes, returned with ROUTE_SHOW option
Return Value	A_OK / A_ERROR	

qcom_ip6_route

Definition	Add, delete or query IPv6 route.	
Prototype	A_STATUS qcom_ip6_route (	
	A_UINT8 device_id,	Unused (default 0)
	A_UINT32 cmd,	Cmd – ROUTE_ADD (0) ROUTE_DEL (1) ROUTE_SHOW (2)
	A_UINT8* addr,	Pointer to IPv6 Address
	A_UINT32* prefixLen,	Prefix Length
	A_UINT32* gw,	Pointer to Gateway IPv6 address
	A_UINT32* ifindex,	Pointer to interface index (used with ROUTE_DEL option)
	IPV6_ROUTE_LIST_T* rtlist)	List of routes, returned with ROUTE_SHOW option
Return Value	A_OK / A_ERROR	

DHCP API

qcom_dhcpc_enable

A_STATUS qcom_dhcpc_enable(A_UINT8 device_id, A_BOOL enable);

qcom_dhcps_enable

qcom_dhcps_set_pool

Definition	Set DHCP pool parameters.	
Prototype	A_STATUS qcom_dhcps_set_pool(	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT32 startip,	The first IP address of DHCP server IP pool

	A_UINT32 endip,	The last IP address of DHCP server IP pool
	A_UINT32 leasetime)	Lease time
Return Value	A_OK / A_ERROR	

qcom_dhcps_get_pool

A_STATUS qcom_dhcps_get_pool(A_UINT8 device_id, A_UINT32 *pstartip, A_UINT32 *pendip, A_UINT32 *pleasetime);

qcom_dhcps_release_pool

Definition	Release DHCP address.	
Prototype	A_STATUS qcom_dhcps_release_pool(	
	A_UINT8 device_id,)	Interface ID, 0 or 1 (default 0)
Return Value	A_OK / A_ERROR	

Bridge mode API

qcom_bridge_mode_enable

Definition	Enable/Disable bridging	
Prototype	A_STATUS qcom_bridge_mode_enable (	
	A_UINT16 bridgemode)	1 = Enable; 0 = Disable
Return Value	A_OK / A_ERROR	

HTTP API

qcom_http_server

Definition	Start/Stop HTTP/HTTPS server

Prototype	int qcom_http_server(	
	A_INT32 enable)	Enable: 0 = Stop HTTP server 1 = Start HTTP server 2 = Stop HTTPS server 3 = Start HTTPS server
Return Value	0 / -1	

qcom_http_server_method

Definition	Get/post the values from/to the HTTP server. For GET operation, make sure sufficient memory is allocated before calling this API.	
Prototype	A_INT32 qcom_http_server_method(	
	A_INT32 command,	0 = POST; 1 = GET
	A_UINT8 *pagename,	Page name from/to which the data is to get/post
	A_UINT8 *objname,	Name of the object in HTML file
	A_INT32 objtype,	Type of the object (string/integer/Boolean)
	A_INT32 objlen,	Object length 1 = Boolean; 4 = Integer; String length = string
	A_UINT8 * value)	Value of the object
Return Value	0 / -1	

qcom_http_client_method

Definition	Configure HTTP client	
Prototype	A_STATUS qcom_http_client_method(	
	A_UINT32 cmd,	0 = HTTPC_CONNECT_CMD 1 = HTTPC_GET_CMD 2 = HTTPC_POST_CMD 3 = HTTPC_DATA_CMD 4 = HTTPC_DISCONNECT_CMD 5 = HTTPC_CONNECT_SSL_CMD
	A_UINT8 *url,	Page name
	A_UINT8 *data,	Object name
	A_UINT8 * value)	Output of the HTTP client request
Return Value	A_OK / A_ERROR	

DNS API

qcom_dnsc_enable

Definition	Enable/Disable DNS client function.	
Prototype	A_STATUS qcom_dnsc_enable (	
	A_BOOL enable)	1 = Enable, 0 = Disable
Return Value	A_OK / A_ERROR	

qcom_dnsc_add_server_address

Definition	Add DNS server IP address.	
Prototype	A_STATUS qcom_dnsc_add_server_address (	
	A_UINT8* ipaddress	IP address of DNS server
	A_UINT8 type)	Type = AF_INET (IPv4) = AF_INET6 (IPv6)
Return Value	A_OK / A_ERROR	

qcom_dnsc_del_server_address

Definition	Delete DNS server IP address.	
Prototype	A_STATUS qcom_dnsc_del_server_address (	
	A_UINT8* ipaddress	IP address of DNS server
	A_UINT8 type)	Type = AF_INET (IPv4) = AF_INET6 (IPv6)
Return Value	A_OK / A_ERROR	

地址：深圳市福田区侨香路裕和大厦八楼　　邮码（POST）：518040
Harmony Building,Qiaoxiang Rd,Futian District,Shenzhen,Guangdong,China.
电话（TEL）：0086-755-82531616　传真（FAX）：0086-755-82531717

qcom_dnsc_get_host_by_name

Definition	Get IP address by URL.	
Prototype	A_STATUS qcom_dnsc_get_host_by_name (	
	A_CHAR *pname,	A pointer to URL
	A_UINT32 *pipaddress)	A pointer to IP address
Return Value	A_OK / A_ERROR	

qcom_dnss_enable

Definition	Enable or disable DNS server	
Prototype	A_STATUS qcom_dnss_enable(	
	A_BOOL enable)	1 = Enable; 0 = Disable
Return Value	A_OK / A_ERROR	

qcom_dns_local_domain

Definition	Set DNS local domain name.	
Prototype	A_STATUS qcom_dns_local_domain(	
	A_CHAR *local_domain)	Local domain name
Return Value	A_OK / A_ERROR	

qcom_dns_entry_create

Definition	Create a DNS server entry (IPv4).	
Prototype	A_STATUS qcom_dns_entry_create (	
	A_CHAR *hostname	The point of domain name
	A_UINT32 ipaddress)	IPv4 address
Return Value	A_OK / A_ERROR	

qcom_dns_entry_delete

Definition	Delete a DNS server entry (IPv4).	
Prototype	A_STATUS qcom_dns_entry_delete (	
	A_CHAR *hostname	The point of domain name
	A_UINT32 ipaddress)	IPv4 address
Return Value	A_OK / A_ERROR	

qcom_dns_6entry_create

Definition	Create a DNS server entry (IPv6).	
Prototype	A_STATUS qcom_dns_6entry_create (	
	A_CHAR *hostname	The point of domain name
	A_UINT8 ipaddress)	IPv6 address
Return Value	A_OK / A_ERROR	

qcom_dns_6entry_delete

Definition	Delete a DNS server entry (IPv6).	
Prototype	A_STATUS qcom_dns_6entry_delete (	
	A_CHAR *hostname	The point of domain name
	A_UINT8 ipaddress)	IPv6 address
Return Value	A_OK / A_ERROR	

qcom_dnsc_get_host_by_name2

Definition	Get IP address by URL.	
Prototype	A_STATUS qcom_dnsc_get_host_by_name2 (	
	A_CHAR *pname,	A pointer to URL
	A_UINT32 *pipaddress,	A pointer to IP address
	A_INT32 domain,	2 = IPv4; 3 = IPv6
	A_UINT32 mode)	1 = gethostbyname2;

		2 = resolvehostname
Return Value	A_OK / A_ERROR	

OTA API

qcom_ota_request_daemon

void qcom_ota_request_daemon(void);

qcom_ota_daemon_start

int qcom_ota_daemon_start(void);

qcom_ota_daemon_stop

int qcom_ota_daemon_stop(void);

qcom_ota_upgrade

Definition	This API allows user to either upgrade target (QCA4010/QCA4012) or host from an external server.	
Prototype	A_INT32 qcom_ota_upgrade(	
	A_UINT8 device_id,	Device ID
	A_UINT32 serverip,	Server IP address
	A_CHAR* filename,	Filename
	A_UINT8 mode,	0 = Firmware OTA download 1 = Host OTA download
	A_UINT8 preserve_last,	0 = OTA can overwrite the current partition. 1 = OTA is not allowed to overwrite the current partition. The preserve_last parameter is mainly used when only one alternative partition is available (which happens to be the current one). Unsuccessfully updating that partition results in falling back to the default partition which could be old.

	A_UINT8 protocol,	0 = TFTP
	A_UINT32* resp_code,	OTA Service will update resp_code and size of the downloaded OTA image.
	A_UIN32 *length)	
Return Value	A_OK / A_ERROR	

qcom_read_ota_area

Definition	Read data (in bytes) from the offset in OTA area.	
Prototype	A_STATUS qcom_read_ota_area(	
	A_UINT32 offset,	Offset to the OTA area
	A_UINT32 size,	Data size in bytes
	A_UCHAR *buffer,	The buffer to store the read content
	A_UNIT32 *ret_len)	The return data length
Return Value	A_OK / A_ERROR	

qcom_ota_done

Definition	Terminate or close the OTA Service.	
Prototype	A_INT32 qcom_ota_done()	
Return Value	A_OK / A_ERROR	

SNTP API

qcom_enable_sntp_client

Definition	Enable/disable SNTP at run time.	
Prototype	void qcom_enable_sntp_client (	
	A_BOOL enable)	Enable/disabe SNTP
Return Value	0 / -1	

qcom_sntp_srvr_addr

Definition	Configure SNTP Server address.	
Prototype	void qcom_sntp_srvr_addr(	
	A_INT32 flag,	Flag to add/delete the address 1 – Add server address 2- Delete server address
	char *srvr_addr)	Pointer to SNTP Server address
Return Value	0 / -1	

qcom_sntp_zone

Definition	Configure SNTP time zone and enable/disable day light saving.	
Prototype	void qcom_sntp_zone(	
	A_INT32 hour,	Time stored in hours
	A_INT32 min,	Time stored in minutes
	A_BOOL add_sub,	Time to add or subtract to get the appropriate time zone
	A_BOOL enable	Enable or disable day light saving
Return Value	0 / -1	

qcom_sntp_get_time

Definition	Get SNTP time. The “time parameter will contain the current date time information.	
Prototype	void qcom_sntp_get_time(	
	A_UINT8 device_id,	Interface ID, 0 or 1 (default 0)
	tSntpTime *time)	Pointer to SNTP time structure
Return Value	0 / -1	

qcom_sntp_show_config

void qcom_sntp_show_config();

qcom_sntp_get_time_of_day

Definition	Get SNTP time of day in seconds and milliseconds.	
Prototype	void qcom_sntp_get_time_of_day(	
	A_UINT8 device_id,	Interface ID, 0 or 1 (default 0)
	tSntpTime *time)	Pointer to SNTP structure that contains time in seconds and milliseconds
Return Value	0 / -1	

qcom_sntp_query_srvr_address

Definition	Query the SNTP address.	
Prototype	void qcom_sntp_query_srvr_address(	
	A_UINT8 device_id,	Interface ID (default 0)
	SNTP_QUERY_SRVR_ADDRESS *addr	Pointer to SNTP server address structure
Return Value	0 / -1	

SSL API

qcom_SSL_ctx_new

Definition	Create an SSL context for use by one or more SSL sessions.	
Prototype	SSL_CTX* qcom_SSL_CTX_new (	
	SSL_ROLE_T role,	Role = 1 (Server); 2 (Client)
	A_INT32 inbufSize,	Initial input buffer size
	A_INT32 outbufSize,	Initial output buffer size
	A_INT32 reserved)	
Description	This function must be called before using any other SSL function.	
Return Value	Pointer to the new context on success; null on error	

qcom_SSL_ctx_free

Definition	Release SSL context.	
Prototype	void qcom_SSL_CTX_free (	
	SSL_CTX * ctx)	SSL context to release
Return Value	SSL_CLNT_OK on success; error code on error	

qcom_SSL_new

Definition	Create an SSL connection object for use with an SSL session.	
Prototype	SSL* qcom_SSL_new (	
	SSL_CTX * ctx)	Pointer to SSL context
Description	Close the connection object with qcom_SSL_shutdown after transaction is completed.	
Return Value	Pointer to SSL structure on success; null on error	

qcom_SSL_addCert

Definition	Add a certificate to the SSL object.	
Prototype	A_INT32 qcom_SSL_addCert(	
	SSL_CTX *ctx,	Pointer to SSL context
	A_UINT8* cert,	Address of array of binary data
	A_UINT32 size)	Size of array
Description	SSL object in server mode requires at least one certificate.	
Return Value	1 on success, negative error code on error	

qcom_SSL_setCaList

Definition	Set CA list for the SSL object	
Prototype	A_INT32 qcom_SSL_setCaList(	
	SSL_CTX *ctx,	Pointer to SSL context
	A_UINT8* cert,	Reference to SSL CA LIST
	A_UINT32 size)	Size of array

Description	SSL performs certificate validation based on the peer certificate. Only one CA list is allowed, thus the CA list must include all root certificates required for the session.
Return Value	1 on success, negative error code on error

qcom_SSL_addCert

Definition	Add a certificate to the SSL object.	
Prototype	A_INT32 qcom_SSL_addCert(	
	SSL_CTX *ctx,	Pointer to SSL context
	A_UINT8* cert,	Address of array of binary data
	A_UINT32 size)	Size of array
Description	SSL object in server mode requires at least one certificate.	
Return Value	1 on success, negative error code on error	

qcom_SSL_storeCert

Definition	Store a certificate or CA list to flash.	
Prototype	A_INT32 qcom_SSL_storeCert(	
	A_CHAR *name,	Name of the certificate
	A_UINT8* cert,	Address of array of binary data
	A_UINT32 size)	Size of array
Return Value	1 on success, negative error code on error	

qcom_SSL_loadCert

Definition	Load certificate from flash	
Prototype	A_INT32 qcom_SSL_loadCert(	
	SSL_CTX *ctx,	Pointer to SSL context
	SSL_CERT_TYPE_T type	type - Certificate or CA List
	Char *name)	Name of the certificate
Description	Load a certificate or CA list from flash and store it in the SSL object.	
Return Value	1 on success, negative error code on error	

qcom_SSL_listCert

Definition	List the certificates stored in the flash	
Prototype	A_INT32 qcom_SSL_listCert(	
	SSL_FILE_NAME_LIST *fileNames)	Pointer to array containing names of certificate files stored in flash.
Description	Return the list of certificates stored in the flash	
Return Value	1 on success, negative error code on error	

qcom_SSL_shutdown

Definition	Shut down data flow for an SSL session.	
Prototype	int qcom_SSL_shutdown (	
	SSL * ssl)	Pointer to SSL session
Return Value	0 = The SSL_shutdown function is first issued from the application. Continue issuing the SSL_shutdown function until the code 1 is received, which means the remote application is shutdown. 1 = Both client and server applications have issued the SSL_shutdown function (in SSL version 3 and TLS version 1). -1 = error	

qcom_SSL_configure

Definition	Configure the SSL connection.	
Prototype	A_INT32 qcom_SSL_configure(	
	SSL *ssl,	Pointer to SSL session
	SSL_CONFIG *cfg)	Pointer to the structure that contains the SSL configuration
Description	Provide information on Cipher, verification policy etc.	
Return Value	1 on success, negative error code on error	

qcom_SSL_set_fd

Definition	Assign a socket to an SSL session.	
Prototype	int qcom_SSL_set_fd (	
	SSL * ssl,	Pointer to SSL session
	int fd)	File descriptor of the socket
Return Value	1 on success, negative error code on error	

qcom_SSL_accept

Definition	Accept an SSL session connection request from a remote client application.	
Prototype	int qcom_SSL_accept (	
	SSL * ssl)	Pointer to SSL session
Return Value	1 on success, negative error code on error	

qcom_SSL_connect

Definition	Start an SSL session with a remote SSL server.	
Prototype	int qcom_SSL_connect (	
	SSL * ssl)	Pointer to SSL session
Return Value	1 on success, negative error code on error	

qcom_SSL_read

Definition	Read application data from an SSL session.	
Prototype	int qcom_SSL_read (	
	SSL * ssl,	Pointer to SSL session
	char * buf,	Pointer to the buffer to read
	int num)	Maximum bytes of data to read
Return Value	Byte count of the read data on sucess; error code on error	

qcom_SSL_write

Definition	Write application data over an SSL session.	
Prototype	int qcom_SSL_write (	
	SSL * ssl,	Pointer to SSL session
	char * buf,	Pointer to the data to send
	int num)	Number of bytes of data to send
Return Value	Byte count of the sent data on sucess; error code on error	

WLAN API

qcom_op_set_mode

Definition	Set operation mode.	
Prototype	A_STATUS qcom_op_set_mode (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT32 mode)	0 = Station; 1 = AP
Return Value	A_OK / A_ERROR	

qcom_op_get_mode

Definition	Get operation mode.	
Prototype	A_STATUS qcom_op_set_mode (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT32 *pmode)	A pointer to operation mode
Return Value	A_OK / A_ERROR	

qcom_get_country_code

Definition	Get the country code in use.	
Prototype	A_STATUS qcom_get_country_code (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_CHAR *pcountry_code)	A pointer to country code.
Return Value	A_OK / A_ERROR	

qcom_disconnect

Definition	Disconnect from the currently associated AP, or abort the current connection process.	
Prototype	A_STATUS qcom_disconnect (	
	A_UINT8 device_id)	Interface ID (default 0)
Return Value	A_OK / A_ERROR	

qcom_set_phy_mode

Definition	Set PHY operating mode.	
Prototype	A_STATUS qcom_set_phy_mode (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT8 phymode)	PHY modes: QCOM_11A_MODE = 0x1, QCOM_11B_MODE = 0x2, QCOM_11G_MODE = 0x3, QCOM_11N_MODE = 0x4, QCOM_HT40_MODE = 0x5,
Return Value	A_OK / A_ERROR	

qcom_get_phy_mode

Definition	Get PHY operating mode.

Prototype	A_STATUS qcom_get_phy_mode (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT8 *pphymode)	A pointer to PHY mode
Return Value	A_OK / A_ERROR	

qcom_set_channel

Definition	Set channel ID.	
Prototype	A_STATUS qcom_set_channel (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT16 channel)	Channel number
Return Value	A_OK / A_ERROR	

qcom_get_channel

Definition	Get the current channel.	
Prototype	A_STATUS qcom_get_channel (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT16 *pchannel)	A pointer to channel
Return Value	A_OK / A_ERROR	

qcom_set_tx_power

Definition	Set Tx power level.	
Prototype	A_STATUS qcom_set_tx_power (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT32 dbm)	Desired transmit power (1-17 dBm)
Return Value	A_OK / A_ERROR	

qcom_get_tx_power

Definition	Get Tx power level.	
Prototype	A_STATUS qcom_get_tx_power (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT32 *pdbm)	A pointer to Tx power
Return Value	A_OK / A_ERROR	

qcom_allow_aggr_set_tid

Definition	Set TID to enable/disable aggregation.	
Prototype	A_STATUS qcom_allow_aggr_set_tid(	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT16 tx_allow_aggr,	16-bit mask. Bit position indicates TID. 1 = Allow aggregation at uplink.
	A_UINT16 rx_allow_aggr)	16 bit mask. Bit position indicates TID. 1 = Allow aggregation at downlink.
Return Value	A_OK / A_ERROR	

qcom_allow_aggr_get_tid

Definition	Get TID aggregation settings.	
Prototype	A_STATUS qcom_allow_aggr_get_tid(	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT16 *ptx_allow_aggr,	A pointer to Tx aggregation bit mask
	A_UINT16 *prx_allow_aggr)	A pointer to Rx aggregation bit mask
Return Value	A_OK / A_ERROR	

qcom_set_connect_callback

Definition	Set the call back function which will be invoked on connection state change.

Prototype	A_STATUS qcom_set_connect_callback (	
	A_UINT8 device_id,	Interface ID (default 0)
	void *callback)	Call back function. The prototype of the function is: void fn(A_UINT8 device_id, A_INT32 value) value = 1 (Successful connection) = 0 (Disconnect)
Return Value	A_OK / A_ERROR	

qcom_set_ssid

Definition	Set SSID.	
Prototype	A_STATUS qcom_set_ssid (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_CHAR *pssid)	A pointer to SSID (length < 32)
Description	Set the SSID. This API must be called before calling qcom_set_scan or qcom_commit.	
Return Value	A_OK / A_ERROR	

qcom_get_ssid

Definition	Get SSID.	
Prototype	A_STATUS qcom_get_ssid (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_CHAR *pssid)	A pointer to SSID (length < 32)
Return Value	A_OK / A_ERROR	

qcom_get_state

Definition	Get chip Wi-Fi link state.	
Prototype	A_STATUS qcom_get_state (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT8 *pstate)	Chip states: K_WLAN_LINK_STATE_DISCONNECTED_STATE

		K_WLAN_LINK_STATE_STARTING_STATE K_WLAN_LINK_STATE_CONNECTING_STATE K_WLAN_LINK_STATE_AUTHENTICATING_STATE K_WLAN_LINK_STATE_CONNECTED_STATE K_WLAN_LINK_STATE_DISCONNECTING_STATE K_WLAN_LINK_STATE_WEP_KEY_NOT_MATCH
Return Value	A_OK / A_ERROR	

qcom_get_disconnect_reason

Definition	Get disconnect reason code.	
Prototype	A_STATUS qcom_get_disconnect_reason (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT32 *preason)	Disconnect reason code
Return Value	0x0 = Connected state 0x01 = NO_NETWORK_AVAIL 0x02 = LOST_LINK (bmiss) 0x03 = DISCONNECT_CMD 0x04 = BSS_DISCONNECTED 0x05 = AUTH_FAILED 0x06 = ASSOC_FAILED 0x07 = NO_RESOURCES_AVAIL 0x08 = CSERV_DISCONNECT 0x0A = INVALID_PROFILE 0x0B = DOT11H_CHANNEL_SWITCH 0x0C = PROFILE_MISMATCH 0x0D = CONNECTION_EVICTED 0x0E = IBSS_MERGE 0x0F = EXCESS_TX_RETRY (TX frames failed after maximum retries) 0x10 = SEC_HS_TO_RECV_M1 (Security 4-way handshake timed out waiting for M1) 0x11 = SEC_HS_TO_RECV_M3 (Security 4-way handshake timed out waiting for M3) 0x12 = TKIP_COUNTERMEASURES 0xFD = CCX_TARGET_ROAMING_INDICATION (Hack for CCX AP-Assisted roaming) 0xFE = CCKM_ROAMING_INDICATION (Hack for CCKM fast roaming)	

qcom_set_bssid

A_STATUS qcom_set_bssid(A_UINT8 device_id, A_UINT8 *pbssid);

qcom_get_bssid

Definition	Get the BSSID of the device context for the given device ID.	
Prototype	A_STATUS qcom_get_bssid (	
	A_UINT8 device_id,	Interface ID, 0 or 1 (default 0)
	A_UINT8 pmode[ATH_MAC_LEN])	Pointer to the memory where the BSSID should be copied to.
Return Value	A_OK / A_ERROR	

qcom_set_lpl_enable

Definition	Enable or disable LPL function.	
Prototype	A_STATUS qcom_set_lpl_enable(	
	A_UINT8 device_id,	Interface ID
	A_UINT8 enable)	0 = Disable; 1 = Enable
Return Value	A_OK / A_ERROR	

qcom_set_gtx_enable

Definition	Enable or disable Green Tx function.	
Prototype	A_STATUS qcom_set_gtx_enable(	
	A_UINT8 device_id,	Interface ID
	A_UINT8 enable)	0 = Disable; 1 = Enable
Return Value	A_OK / A_ERROR	

qcom_set_rate

Definition	Set data rate.	
Prototype	A_STATUS qcom_set_rate(	
	A_UINT8 device_id,	Interface ID
	A_UINT8 mcs,	0 = Disable; 1 = Enable
	A_UINT8 rate)	If mcs is set to 0, rate can be set to 1, 2, 5, 11, 6, 9, 12, 18, 24, 36, 48, 54 Mbps. If mcs is set to 1, rate can be set to 0 to 7.
Return Value	A_OK / A_ERROR	

WLAN Scan API

qcom_scan_set_mode

Definition	Set firmware scan behavior by enabling foreground or background scanning.	
Prototype	A_STATUS qcom_scan_set_mode (	
	A_UINT8 device_id,	Interface ID, 0 or 1 (default 0)
	A_UINT32 mode)	1 = Foreground 0 = Background
Return Value	A_OK / A_ERROR	

qcom_scan_get_mode

Definition	Get the scan mode.	
Prototype	A_STATUS qcom_scan_get_mode (	
	A_UINT8 device_id,	Interface ID, 0 or 1 (default 0)
	A_UINT32 *pmode)	A pointer to scan mode
Return Value	A_OK / A_ERROR	

qcom_scan_set_para

A_STATUS qcom_scan_set_para(A_UINT8 device_id, A_UINT16 max_dwell_time, A_UINT16 pass_dwell_time);

qcom_scan_bss_start

A_STATUS qcom_scan_bss_start(A_UINT8 device_id, A_CHAR *ssid);

qcom_scan_all_bss_start

A_STATUS qcom_scan_all_bss_start(A_UINT8 device_id);

qcom_scan_get_bss_number

A_STATUS qcom_scan_get_bss_number(A_UINT8 device_id, A_UINT32 *pnumber);

qcom_scan_get_bss_info

A_STATUS qcom_scan_get_bss_info(A_UINT8 device_id, A_UINT32 id, QCOM_BSS_SCAN_INFO *pbuf);

qcom_bss_packet_print

void qcom_bss_packet_print(A_UINT8 * buf, int length);

qcom_record_bss_info

void qcom_record_bss_info(A_UINT8 device_id, A_UINT8 * pBuf, int len);

qcom_record_bss_info

void qcom_bss_scan_result_show(A_UINT8 device_id);

qcom_get_bss_entry_by_ssid

Definition	Get detailed information by the SSID from scan result.	
Prototype	QCOM_BSS_SCAN_INFO *qcom_get_bss_entry_by_ssid(	
	A_UINT8 device_id,	Interface ID
	char ssid[])	SSID

Return Value	Information of SSID. Return NULL if no matched SSID.
	typedef struct _qcom_scan_info
	{
	unsigned char channel; //channel number.
	unsigned char ssid_len; //the length of ssid.
	unsigned char rssi; //received signal strength indicator.
	unsigned char security_enabled; //security enable/disable.
	unsigned short beacon_period; //beacon intervel.
	unsigned char preamble; //short/long preamble.
	unsigned char bss_type; //ESS/IBSS type network.
	unsigned char bssid[6]; //bssid information.
	unsigned char ssid[32]; //ssid information.
	unsigned char rsn_cipher; //TKIP/CCMP/WEP/NA
	unsigned char rsn_auth; //PSK/1X/NA
	unsigned char wpa_cipher; //TKIP/CCMP/WEP/NA
	unsigned char wpa_auth; //PSK/1X/NA
	unsigned short caps; //capability info
	unsigned char wep_support; //not used
	unsigned char reserved; //keeps struct on 4-byte boundary
	} QCOM_BSS_SCAN_INFO, * QCOM_BSS_SCAN_INFO_PTR;	

qcom_set_scan

Definition	Start a long or short channel scan.

Prototype	A_STATUS qcom_set_scan(	
	A_UINT8 device_id,	Interface ID

	qcom_start_scan_params_t *pScanParams)	typedef struct _qcom_start_scan_params
		{
		A_BOOL forceFgScan; // Forces foreground scan
		A_UINT32 homeDwellTimeInMs; // Maximum duration in the home channel (milliseconds)
		A_UINT32 forceScanIntervalInMs; // Time interval between scans (milliseconds)
		A_UINT8 scanType; // 0 = full scan, 1 = short scan
		A_UINT8 numChannels; // Number of channels
		A_UINT16 channelList[QCOM_START_SCAN_PARAMS_CHANNEL_LIST_MAX]; // Number of channels
		} qcom_start_scan_params_t;

Return Value	A_OK / A_ERROR	

qcom_get_scan

Definition	Get the scan results.	
Prototype	A_STATUS qcom_get_scan (	
	A_UINT8 device_id,	Interface ID (default 0)
	QCOM_BSS_SCAN_INFO** buf,	Pointer to buffer that will contain scan results
	A_INT16* numResults)	Pointer to number of results
Description	This API will block until scan is complete. User must not allocate or free “buf” parameter, it will point to the library’s scan buffer.	
Return Value	A_OK / A_ERROR	

WLAN Security API

qcom_sec_set_wepkey

Definition	Set WEP encryption key by index.

Prototype	A_STATUS qcom_sec_set_wepkey (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT32 keyindex,	WEP key index (0-3)
	A_CHAR *pkey)	WEP key (format “ABCDEF1234”)
Return Value	A_OK / A_ERROR	

qcom_sec_get_wepkey

Definition	Get WEP encryption key by index.	
Prototype	A_STATUS qcom_sec_get_wepkey (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT32 keyindex,	WEP key index (0-3)
	A_CHAR *pkey)	A pointer to WEP key
Return Value	A_OK / A_ERROR	

qcom_sec_set_wepkey_index

Definition	Set WEP key index.	
Prototype	A_STATUS qcom_sec_set_wepkey_index (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT32 keyindex)	WEP key index (0-3)
Return Value	A_OK / A_ERROR	

qcom_sec_get_wepkey_index

Definition	Get WEP key index.	
Prototype	A_STATUS qcom_sec_get_wepkey_index (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT32 keyindex)	WEP key index (0-3)
Return Value	A_OK / A_ERROR	

qcom_sec_set_wep_mode

Definition	Set WEP mode.	
Prototype	A_STATUS qcom_sec_set_wep_mode(	
	A_UINT8 device_id,	Interface ID
	A_UINT32 mode)	WEP mode 0 = Open; 1 = Shared
Return Value	A_OK / A_ERROR	

qcom_sec_get_wep_mode

Definition	Get WEP mode.	
Prototype	A_STATUS qcom_sec_get_wep_mode(	
	A_UINT8 device_id,	Interface ID
	A_UINT32 *mode)	A pointer to WEP mode 0 = Open; 1 = Shared
Return Value	A_OK / A_ERROR	

qcom_sec_set_auth_mode

Definition	Set authentication mode.	
Prototype	A_STATUS qcom_sec_set_auth_mode (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT32 mode)	Authentication modes: WLAN_AUTH_NONE WLAN_AUTH_WPA WLAN_AUTH_WPA2 WLAN_AUTH_WPA_PSK WLAN_AUTH_WPA2_PSK WLAN_AUTH_WPA_CCKM WLAN_AUTH_WPA2_CCKM WLAN_AUTH_WPA2_PSK_SHA256 WLAN_AUTH_INVALID
Return Value	A_OK / A_ERROR	

qcom_sec_get_auth_mode

Definition	Get authentication mode.	
Prototype	A_STATUS qcom_sec_get_auth_mode (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT32 *pmode)	A pointer to authentication mode
Return Value	A_OK / A_ERROR	

qcom_sec_set_encrypt_mode

Definition	Set encryption mode.	
Prototype	A_STATUS qcom_sec_set_encrypt_mode (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT32 mode)	Encryption modes: WLAN_CRYPT_NONE WLAN_CRYPT_WEP_CRYPT WLAN_CRYPT_TKIP_CRYPT WLAN_CRYPT_AES_CRYPT WLAN_CRYPT_BIP_CRYPT WLAN_CRYPT_KTK_CRYPT WLAN_CRYPT_INVALID
Return Value	A_OK / A_ERROR	

qcom_sec_get_encrypt_mode

Definition	Get encryption mode.	
Prototype	A_STATUS qcom_sec_get_encrypt_mode (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT32 *pmode)	A pointer to encryption mode
Return Value	A_OK / A_ERROR	

qcom_sec_set_passphrase

Definition	Set passphrase.	
Prototype	A_STATUS qcom_sec_set_passphrase (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_CHAR *passphrase)	A pointer to passphrase
Return Value	A_OK / A_ERROR	

qcom_sec_get_passphrase

Definition	Get passphrase.	
Prototype	A_STATUS qcom_sec_get_passphrase (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_CHAR *passphrase)	A pointer to passphrase
Return Value	A_OK / A_ERROR	

WLAN AP MODE API

qcom_ap_set_beacon_interval

Definition	Set Beacon interval (Soft AP mode).	
Prototype	A_STATUS qcom_ap_set_beacon_interval (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT32 beacon_interval)	Beacon interval in TUs (1 TU = 1024 μs)
Description	The default beacon interval is 100 TUs (102.4 ms).	
Return Value	A_OK / A_ERROR	

qcom_ap_get_beacon_interval

Definition	Get Beacon interval (Soft AP mode).

Prototype	A_STATUS qcom_ap_get_beacon_interval (	
	A_UINT8 device_id,	Interface ID
	A_UINT32 *pbeacon_interval)	A pointer to beacon interval
Return Value	A_OK / A_ERROR	

qcom_ap_hidden_mode_enable

Definition	Enable/Disable SSID hidden mode (Soft AP mode).	
Prototype	A_STATUS qcom_ap_hidden_mode_enable (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_BOOL enable)	1 = Enable; 0 = Disable
Return Value	A_OK / A_ERROR	

qcom_ap_set_max_station_number

Definition	Set the maximum number of stations allowed to connect.	
Prototype	A_STATUS qcom_ap_set_max_station_number (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT32 sta_num)	Maximum number of stations
Return Value	A_OK / A_ERROR	

qcom_ap_set_flag

Definition	Set connect control flags.	
Prototype	A_STATUS qcom_ap_set_flag (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT32 flg)	Control flag CONNECT_ASSOC_POLICY_USER=0x0001, CONNECT_SEND_REASSOC =0x0002, CONNECT_IGNORE_WPAx_GROUP_CIPHER = 0x0004, CONNECT_PROFILE_MATCH_DONE=0x0008, CONNECT_IGNORE_AAC_BEACON=0x0010, CONNECT_CSA_FOLLOW_BSS=0x0020,

		CONNECT_DO_WPA_OFFLOAD=0x0040, CONNECT_DO_NOT_DEAUTH=0x0080, CONNECT_WPS_FLAG=0x0100, CONNECT_IGNORE_BSSID_HINT=0x0200, CONNECT_STAY_AWAKE=0x0400, CONNECT_SAFE_MODE=0x0800, /* AP configuration flags */ AP_NO_DISASSOC_UPON_DEAUTH=0x10000, AP_HOSTPAL_SUPPORT=0x20000
Return Value	A_OK / A_ERROR	

qcom_ap_set_inact_time

Definition	Set inactive time (SoftAP mode).	
Prototype	A_STATUS qcom_ap_set_inact_time(	
	A_UINT8 device_id,	Interface ID
	A_UINT16 minutes)	Inactive time in minutes
Return Value	A_OK / A_ERROR	

qcom_ap_get_sta_info

Definition	Get information of the stations that are connected to the SoftAP.	
Prototype	A_STATUS qcom_ap_get_sta_info(	
	A_UINT8 device_id,	Interface ID
	A_UINT8 *pnum,	A pointer to the number of the stations which are connected to the AP
	A_UINT8 *pmac)	A pointer to the MAC of the stations which are connected to the AP
Return Value	A_OK / A_ERROR	

WLAN STA MODE API

qcom_sta_connect_with_scan

qcom_sta_connect_without_scan

qcom_sta_reconnect_start

Definition	Start reconnet action at an interval specified by user.	
Prototype	A_STATUS qcom_sta_reconnect_start (	
	A_UINT8 device_id,	Interface ID
	A_UINT32 seconds)	Reconnect interval in seconds
Return Value	A_OK / A_ERROR	

qcom_sta_reconnect_stop

Definition	Stop reconnect action.	
Prototype	A_STATUS qcom_sta_reconnect_stop (	
	A_UINT8 device_id)	Interface ID (default 0)
Return Value	A_OK / A_ERROR	

qcom_sta_get_rssi

Definition	Get RSSI value for link quality (SNR).	
Prototype	A_STATUS qcom_sta_get_rssi (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT8 *prssi)	Link quality (SNR in dB)
Return Value	A_OK / A_ERROR	

qcom_sta_set_listen_time

Definition	Set listen interval.	
Prototype	A_STATUS qcom_sta_set_listen_time (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT32 time)	Time interval in TUs for the device to wake up and listen for traffic. 1 TU = 1024 μs
Return Value	A_OK / A_ERROR	

qcom_sta_get_listen_time

Definition	Get listen interval.	
Prototype	A_STATUS qcom_sta_get_listen_time (	
	A_UINT8 device_id,	Interface ID (default 0)
	A_UINT32 *ptime)	A pointer to of listen time interval
Return Value	A_OK / A_ERROR	

qcom_sta_connect_event_wait

qcom_sta_connect_event_wakeup

qcom_connect_ap

qcom_connect_ap_scaned

qcom_commit

Definition	Connect to an AP or start soft AP mode with the security configuration by user.	
Prototype	A_STATUS qcom_commit (	
	A_UINT8 device_id)	Interface ID (default 0)
Description	This API will initiate the connect procedure. It can also be used to start a soft AP. The user must do the following before calling commit: 1. Set Callback function. 2. Set Mode (Station/ soft AP/Ad-Hoc). 3. Set SSID. 4. Set security parameters (optional).	
Return Value	A_OK / A_ERROR	

WLAN WPS API

qcom_wps_connect

Definition	Initiate WPS connection on the specific interface ID.	
Prototype	void qcom_wps_connect (	
	A_UINT8 device_id)	Interface ID, 0 or 1
Return Value	A_OK / A_ERROR	

qcom_wps_enable

Definition	Enable/ disable WPS function. This function must be invoked along with qcom_wps_resister_event_handler().	
Prototype	void qcom_wps_enable (	

	A_UINT8 device_id,	Interface ID, 0 or 1 (default 0)
	int enable)	1 = Enable; 0 = Disable
Return Value	N/A	

qcom_wps_start

Definition	Start WPS connection.	
Prototype	void qcom_wps_start(	
	A_UINT8 device_id,	Interface ID, 0 or 1 (default 0)
	int connect,	1 = After WPS procedure is finished, if the device is in enrollee mode, it connects registrar immediately. 0 = No action after WPS procedure is finished.
	int mode,	0 = PIN mode; 1 = PBC mode
	char *pin)	The address of PIN string when mode is PIN mode.
Return Value	N/A	

qcom_wps_stop

Definition	Stop WPS connection.	
Prototype	void qcom_wps_stop (	
	A_UINT8 device_id)	Interface ID, 0 or 1
Return Value	N/A	

qcom_wps_register_event_handler

Definition	Allow application to register a callback with WPS to indicate WPS error status and take necessary action for the event received. This function must be invoked whenever qcom_wps_enable() is called.	
Prototype	void qcom_wps_register_event_handler (	
	qcom_wps_event_handler_t handler	Event handler function pointer. The handler prototype information is provided in the WPS data structure section.
	void *pArg)	Application context pointer to be supplied in the

		callback
Return Value	N/A	

qcom_wps_set_credentials

Definition	Set the WPS credentials received over the WPS profile event.	
Prototype	A_STATUS qcom_wps_set_credentials (	
	A_UINT8 device_id	Interface ID, 0 or 1
	qcom_wps_credentials_t *pwps_prof)	Credentials structure pointer. Details of this data structure is provided in WPS data structure section
Return Value	A_OK / A_ERROR	

WLAN P2P API

qcom_p2p_func_init

Definition	Enable/disable P2P function.	
Prototype	void qcom_p2p_func_init (	
	A_UINT8 device_id	Interface ID (default 0)
	A_INT32 enable)	1 = Enable; 0 = Disable
Return Value	N/A	

qcom_p2p_func_cancel

Definition	Cancel the P2P function.	
Prototype	A_STATUS qcom_p2p_func_cancel (	
	A_UINT8 device_id)	Interface ID (default 0)
Return Value	A_OK / A_ERROR	

qcom_p2p_func_auth

Definition	Authorize the peer P2P device to connect itself.	
Prototype	A_STATUS qcom_p2p_func_auth (	
	A_UINT8 device_id	Interface ID (default 0)
	A_INT32 dev_auth	0 = Authorize; 1 = Deauthorize
	enum p2p_wps_method wps_method	WPS method: P2P_WPS_NOT_READY P2P_WPS_PIN_LABEL P2P_WPS_PIN_DISPLAY P2P_WPS_PIN_KEYPAD P2P_WPS_PBC
	A_UINT8 *peer_mac	The point of peer P2P device MAC address
	A_BOOL persistent)	persistent group TRUE: persistent; FALSE: non-persistent
Return Value	A_OK / A_ERROR	

qcom_p2p_func_connect

Definition	Connect to the peer P2P device.	
Prototype	A_STATUS qcom_p2p_func_connect (	
	A_UINT8 device_id	Interface ID (default 0)
	enum p2p_wps_method wps_method	WPS method: P2P_WPS_NOT_READY P2P_WPS_PIN_LABEL P2P_WPS_PIN_DISPLAY P2P_WPS_PIN_KEYPAD P2P_WPS_PBC
	A_UINT8 *peer_mac	The point of peer P2P device MAC address
	A_BOOL persistent)	persistent group TRUE: persistent; FALSE: non-persistent
Return Value	A_OK / A_ERROR	

qcom_p2p_func_invite

Definition	Invite the other P2P device to join the P2P group.	
Prototype	A_STATUS qcom_p2p_func_invite (	
	A_UINT8 device_id	Interface ID (default 0)
	A_CHAR *pssid	The point of ssid
	enum p2p_wps_method wps_method	WPS method: P2P_WPS_NOT_READY P2P_WPS_PIN_LABEL P2P_WPS_PIN_DISPLAY P2P_WPS_PIN_KEYPAD P2P_WPS_PBC
	A_UINT8 *pmac	The point of P2P device invited MAC address
	A_BOOL persistent	persistent group TRUE: persistent; FALSE: non-persistent
	P2P_INV_ROLE role)	P2P_INV_ROLE_GO, P2P_INV_ROLE_ACTIVE_GO, P2P_INV_ROLE_CLIENT,
Return Value	A_OK / A_ERROR	

qcom_p2p_func_join

Definition	Join the P2P group.	
Prototype	A_STATUS qcom_p2p_func_join (	
	A_UINT8 device_id	Interface ID (default 0)
	enum p2p_wps_method wps_method	WPS method: P2P_WPS_NOT_READY P2P_WPS_PIN_LABEL P2P_WPS_PIN_DISPLAY P2P_WPS_PIN_KEYPAD P2P_WPS_PBC
	A_UINT8 *pmac	The point of P2P device invited MAC address
	char *ppin)	The point of pin string
Return Value	A_OK / A_ERROR	

qcom_p2p_func_prov

Definition	Start P2P provision negotiation to the peer P2P device.	
Prototype	A_STATUS qcom_p2p_func_prov (	
	A_UINT8 device_id	Interface ID (default 0)
	enum p2p_wps_method wps_method	WPS method: P2P_WPS_NOT_READY P2P_WPS_PIN_LABEL P2P_WPS_PIN_DISPLAY P2P_WPS_PIN_KEYPAD P2P_WPS_PBC
	A_UINT8 *pmac)	The point of P2P device invited MAC address
Return Value	A_OK / A_ERROR	

qcom_p2p_func_set_config

Definition	Set the P2P configuration.	
Prototype	A_STATUS qcom_p2p_func_set_config (	
	A_UINT8 device_id	Interface ID (default 0)
	A_UINT32 uigo_intend	Go intent value
	A_UINT32 uiclisten_ch	Listen channel
	A_UINT32 uiop_ch	Operation channel
	A_UINT32 uiage	Aging time
	A_UINT32 reg_class	Register domain
	A_UINT32 op_reg_class	Operating register domain
	A_UINT32 max_node_count)	Maximum node number (up to 5)
Return Value	A_OK / A_ERROR	

qcom_p2p_func_set_pass_ssid

Definition	Set the passphrase and ssid for P2P	
Prototype	A_STATUS qcom_p2p_func_set_pass_ssid	
	A_UINT8 device_id	Interface ID (default 0)
	A_CHAR *ppass	The point of passpharase

	A_CHAR *pssid	The point of SSID
Return Value	A_OK / A_ERROR	

qcom_p2p_func_get_pass_ssid

Definition	Get the passphrase and ssid for P2P	
Prototype	A_STATUS qcom_p2p_func_get_pass_ssid	
	A_UINT8 device_id	Interface ID (default 0)
	A_CHAR *ppass	The point of passpharase
	A_CHAR *pssid	The point of SSID
Return Value	A_OK / A_ERROR	

qcom_p2p_func_set_opps

Definition	Set OpPS parmeters for power save mode.	
Prototype	A_STATUS qcom_p2p_func_set_opps (	
	A_UINT8 device_id	Interface ID (default 0)
	A_UINT8 enable	1 = Enable; 0 = Disable
	A_UINT8 ctwin	Client traffic windows
Return Value	A_OK / A_ERROR	

qcom_p2p_func_set_noa

Definition	Set NOA parameters for power save mode.	
Prototype	A_STATUS qcom_p2p_func_set_noa (	
	A_UINT8 device_id	Interface ID (default 0)
	A_UINT8 uccount	The number of absence intervals
	A_UINT32 uistart	The start time for the schedule
	A_UINT32 uiduration	The maxinum duration
	A_UINT32 uiinterval	A_UINT32 uiinterval
Return Value	A_OK / A_ERROR	

qcom_p2p_func_find

Definition	Start to find other P2P devices.	
Prototype	A_STATUS qcom_p2p_func_find (	
	A_UINT8 device_id	Interface ID (default 0)
	P2P_DEV_CTXT *dev,	NULL, not useful
	P2P_DISC_TYPE type,	Find type: P2P_DISC_START_WITH_FULL P2P_DISC_ONLY_SOCIAL P2P_DISC_PROGRESSIVE
	A_UINT32 timeout	Timeout time, unit is second
Return Value	A_OK / A_ERROR	

qcom_p2p_func_stop_find

Definition	Stop P2P find action.	
Prototype	A_STATUS qcom_p2p_func_stop_find (	
	A_UINT8 device_id)	Interface ID (default 0)
Return Value	A_OK on success, error code on error	

qcom_p2p_func_set

Definition	Set the P2P configuration	
Prototype	A_STATUS qcom_p2p_func_set (	
	A_UINT8 device_id	Interface ID (default 0)
	P2P_DEV_CTXT *dev,	NULL, not useful
	P2P_CONF_ID config_id	Config ID: P2P_CONFIG_LISTEN_CHANNEL P2P_CONFIG_CROSS_CONNECT P2P_CONFIG_SSID_POSTFIX P2P_CONFIG_INTRA_BSS P2P_CONFIG_CONCURRENT_MODE P2P_CONFIG_GO_INTENT P2P_CONFIG_DEV_NAME

		P2P_CONFIG_P2P_OPMODE
	void *data	The point of specific data
	A_UINT32 data_length)	The length of data
Return Value	A_OK on success, error code on error	

qcom_p2p_func_get_node_list

Definition	Get the P2P node list. This API is blocking and should be called in the context of an application thread.	
Prototype	A_STATUS qcom_p2p_func_get_node_list (	
	A_UINT8 device_id	Interface ID (default 0)
	P2P_NODE_LIST_EVENT *app_buf	Application buffer to hold node list
	A_UINT32 buf_size)	Application buffer size
Return Value	A_OK on success, error code on error	

qcom_p2p_func_invite_auth

Definition	Determine join or reject to join the P2P group.	
Prototype	A_STATUS qcom_p2p_func_invite_auth (	
	A_UINT8 device_id	Interface ID (default 0)
	P2P_FW_INVITE_REQ_RSP_CMD *inv)	typedef PREPACK struct { A_UINT16 force_freq; A_UINT8 status; A_UINT8 dialog_token; A_UINT8 is_go; A_UINT8 group_bssid[MAC_LEN]; } P2P_FW_INVITE_REQ_RSP_CMD;
Return Value	A_OK on success, error code on error	

qcom_p2p_func_get_network_list

Definition	Get the P2P persistent network list. This API is blocking and should be called in the context of an application thread.

Prototype	A_STATUS qcom_p2p_func_get_network_list (	
	A_UINT8 device_id	Interface ID (default 0)
	P2P_PERSISTENT_MAC_LIST*app_buf	Application buffer to hold node list
	A_UINT32 buf_size)	Application buffer size
Return Value	A_OK on success, error code on error	

qcom_p2p_func_register_event_handler

Definition	Register application handler for P2P WMI events.	
Prototype	A_STATUS qcom_p2p_func_register_event_handler (	
	qcom_p2p_event_handler cb	Pointer to event handler in application
	void *arg	Argument to be passed when event handler is called
Return Value	A_OK on success, error code on error	

qcom_p2p_func_start_go

Definition	Start P2P GO (autonomous GO and after GO negotiation).	
Prototype	A_STATUS qcom_p2p_func_start_go (	
	A_UINT8 device_id	Interface ID (default 0)
	A_UINT8 *pssid	Pointer to SSID
	A_UINT8 *ppass	Pointer to passphrase
	A_INT32 channel	GO operating channel
	A_BOOL persistent)	persistent group TRUE: persistent; FALSE: non-persistent
Return Value	A_OK on success, error code on error	

qcom_p2p_func_wps_start_no_scan

Definition	Start WPS for client after GO negotiation is complete.	
Prototype	A_STATUS _qcom_p2p_func_wps_start_no_scan (	

	A_UINT8 device_id	Interface ID (default 0)
	A_BOOL is_go	TRUE: GO; FALSE: Client
	A_BOOL is_push_method	TRUE: WPS PBC; FALSE: Other
	A_UINT32 channel	Channel on which to start WPS
	A_UINT8 *pssid	Pointer to SSID
	A_UINT8 *pin	Pointer to PIN
	A_UINT8 *pmac)	Pointer to MAC address
Return Value	A_OK on success, error code on error	

qcom_p2p_func_get_role

Definition	Get the device role from WMI event buffer.	
Prototype	P2P_INV_ROLE qcom_p2p_func_get_role (	
	A_UINT32 role)	Role in WMI event buffer
Return Value	P2P_INV_ROLE	

qcom_p2p_func_listen

Definition	Have the P2P device listen to the listen channel.	
Prototype	A_STATUS _qcom_p2p_func_listen (	
	A_UINT8 device_id	Interface ID (default 0)
	A_UINT32 timeout)	Timeout in seconds
Return Value	A_OK on success, error code on error	