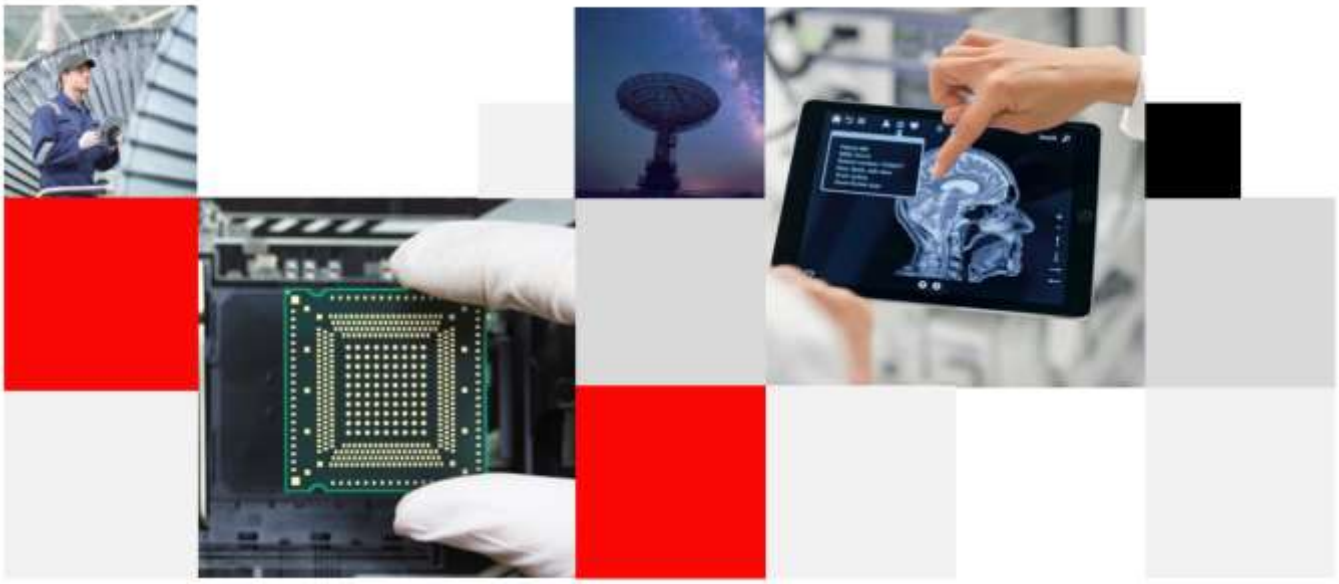
EC2x&EG9x&EG25-G Series
QuecOpen Wi-Fi API Reference Manual

LTE Standard Module Series

Version: 1.0

Date: 2021-03-15

Status: Released



Build a Smarter World

Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company headquarters:

Quectel Wireless Solutions Co., Ltd.
Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local office. For more information, please visit:
http://www.quectel.com/support/sales.htm.

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/technical.htm
Or email to support@quectel.com.

General Notes

Quectel offers the information as a service to its customers. The information provided is based upon customers' requirements. Quectel makes every effort to ensure the quality of the information it makes available. Quectel does not make any warranty as to the information contained herein, and does not accept any liability for any injury, loss or damage of any kind incurred by use of or reliance upon the information. All information supplied herein is subject to change without prior notice.

Disclaimer

While Quectel has made efforts to ensure that the functions and features under development are free from errors, it is possible that these functions and features could contain errors, inaccuracies and omissions. Unless otherwise provided by valid agreement, Quectel makes no warranties of any kind, implied or express, with respect to the use of features and functions under development. To the maximum extent permitted by law, Quectel excludes all liability for any loss or damage suffered in connection with the use of the functions and features under development, regardless of whether such loss or damage may have been foreseeable.

Duty of Confidentiality

The Receiving Party shall keep confidential all documentation and information provided by Quectel, except when the specific permission has been granted by Quectel. The Receiving Party shall not access or use Quectel's documentation and information for any purpose except as expressly provided herein. Furthermore, the Receiving Party shall not disclose any of the Quectel's documentation and information to any third party without the prior written consent by Quectel. For any noncompliance to the above requirements, unauthorized use, or other illegal or malicious use of the documentation and information, Quectel will reserve the right to take legal action.

Copyright

About the Document

Revision History

Version	Date	Author	Description
-	2021-01-27	Arthur CHEN	Creation of the document
1.0	2021-03-15	Arthur CHEN	First official release

Contents

Table Index

1 Introduction

Quectel LTE standard EC2x series and EG25-G modules support QuecOpen® solution. QuecOpen® is an open-source embedded development platform based on Linux system. It is intended to simplify the design and development of IoT applications. For more information, see ***document [1]***.

This document introduces Wi-Fi related APIs and examples for Quectel EC2x series, EG9x series and EG25-G QuecOpen® modules.

The Wi-Fi function on Quectel EC2x series, EG9x series and EG25-G QuecOpen® modules is designed to be used with Quectel Wi-Fi & Bluetooth FC20 series and FC21 modules. For detailed information about the two modules, see *Quectel_FC20_Series_Hardware_Design and Quectel_FC21_Hardware_Design* respectively. Quectel FC20 series and FC21 modules support four working modes, namely:

1) AP Mode

In AP mode, FC20 series and FC21 only enable one hotspot, supporting 2.4 GHz or 5 GHz frequency bands, but only one of the frequency bands is activated at a time.

2) STA Mode

In STA mode, FC20 series and FC21 work as a wireless network card and connect to hotspots in the 2.4 GHz and 5 GHz frequency bands.

3) AP-STA Coexistence Mode

In AP-STA coexistence mode, FC20 series and FC21 run AP and STA modes simultaneously.

4) AP-AP Mode

In AP-AP mode, FC20 series and FC21 run two hotspots simultaneously. The most common scenario is to set the same SSID, one with 2.4 GHz frequency band and the other with 5 GHz frequency band.

1.1. Applicable Modules

Table 1: Applicable Modules

Module Series	Module
EC2x series	EC25 series

	EC21 series
	EC20 R2.1
EG9x series	EG95 series
	EG91 series
EG25-G	EG25-G

2 Wi-Fi APIs

2.1. Header File

The Wi-Fi API header file *ql_wifi.h* is located in the directory of *ql-ol-crosstool/sysroots/armv7a-vfp-neon-oe-linux-gnueabi/usr/include/quectel-openlinux-sdk* in SDK.

Table 2: API Summary

APIs	Description
ql_wifi_register_handle	Register the callback function for Wi-Fi status reporting
ql_wifi_unregister_handle	Unregister Wi-Fi callback function
ql_wifi_enable	Enable Wi-Fi
ql_wifi_disable	Disable Wi-Fi
ql_wifi_enable_async	Enable Wi-Fi asynchronously
ql_wifi_disable_async	Disable Wi-Fi asynchronously
ql_wifi_module_enable	Load the Wi-Fi driver according to the currently set working mode
ql_wifi_module_disable	Unload the Wi-Fi driver according to the currently set working mode
ql_wifi_work_mode_set	Set current working mode
ql_wifi_work_mode_get	Get current working mode
ql_wifi_ap_ssid_set	Set the SSID in AP mode
ql_wifi_ap_ssid_get	Get the SSID in AP mode
ql_wifi_ap_ssid_hide_set	Set to hide SSID in AP mode
ql_wifi_ap_ssid_hide_get	Get the SSID hiding status in AP mode

ql_wifi_ap_mode_set	Set working protocol mode in AP mode
ql_wifi_ap_mode_get	Set working protocol mode in AP mode
ql_wifi_ap_bandwidth_set	Set the bandwidth in AP mode
ql_wifi_ap_bandwidth_get	Get the bandwidth in AP mode
ql_wifi_ap_channel_set	Set the channel in AP mode
ql_wifi_ap_channel_get	Get the channel in AP mode
ql_wifi_ap_auth_set	Set the security authentication in AP mode
ql_wifi_ap_auth_get	Get the security authentication in AP mode
ql_wifi_ap_max_sta_set	Set the maximum number of connections for terminals in AP mode
ql_wifi_ap_max_sta_get	Get the maximum number of connections for terminals in AP mode
ql_wifi_ap_sta_info_get	Get the client information in AP mode
ql_wifi_ap_start	Enable the upper application hostapd in AP mode
ql_wifi_ap_stop	Disable the upper application hostapd in AP mode
ql_wifi_ap_restart	Restart the upper application hostapd in AP mode
ql_wifi_status_get	Get current Wi-Fi status
ql_wifi_sta_ssid_set	Set the SSID in STA mode
ql_wifi_sta_ssid_get	Get the SSID in STA mode
ql_wifi_sta_auth_set	Set the security authentication in STA mode
ql_wifi_sta_auth_get	Get the security authentication in STA mode
ql_wifi_sta_connect	Enable the upper application wpa_supplicant in STA mode
ql_wifi_sta_disconnect	Disable the upper application wpa_supplicant in STA mode
ql_wifi_sta_status	Get STA mode connection status

NOTE

Unless otherwise specified, you shall not call several Wi-Fi APIs simultaneously.

2.2. Wi-Fi API Description

2.2.1. ql_wifi_register_handle

This function registers the callback function for Wi-Fi status reporting.

- **Prototype**

```
int ql_wifi_register_handle(wifi_event_handle event_handle, void *arg);
```

- **Parameter**

event_handle:
[In] Define the reported callback function.

arg:
[In] User-defined parameter.

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.2. ql_wifi_unregister_handle

This function unregisters Wi-Fi callback function.

- **Prototype**

```
void ql_wifi_unregister_handle(void);
```

- **Parameter**

N/A

- **Return Value**

N/A

2.2.3. ql_wifi_enable

This function enables Wi-Fi.

- **Prototype**

```
int ql_wifi_enable(void);
```

- **Parameter**

N/A

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.4. ql_wifi_disable

This function disables Wi-Fi.

- **Prototype**

```
int ql_wifi_disable(void);
```

- **Parameter**

N/A

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.5. ql_wifi_enable_async

This function enables Wi-Fi asynchronously.

- **Prototype**

```
int ql_wifi_enable_async(void);
```

- **Parameter**

N/A

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.6. ql_wifi_disable_async

This function disables Wi-Fi asynchronously.

- **Prototype**

```
int ql_wifi_disable_async(void);
```

- **Parameter**

N/A

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.7. ql_wifi_module_enable

This function loads the Wi-Fi driver according to the currently set working mode.

- **Prototype**

```
int ql_wifi_module_enable(void);
```

- **Parameter**

N/A

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.8. ql_wifi_module_disable

This function unloads the Wi-Fi driver according to the currently set working mode.

- **Prototype**

```
int ql_wifi_module_disable(void);
```

- **Parameter**

N/A

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.9. ql_wifi_work_mode_set

This function sets current working mode.

- **Prototype**

```
int ql_wifi_work_mode_set(ql_wifi_work_mode_e mode);
```

- **Parameter**

mode:
[Out] Gets current working mode.

QL_WIFI_WORK_MODE_STA	STA mode
QL_WIFI_WORK_MODE_AP0	AP mode
QL_WIFI_WORK_MODE_AP0_STA	AP-STA mode
QL_WIFI_WORK_MODE_AP0_AP1	AP-AP mode

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.10. ql_wifi_work_mode_get

This function gets current working mode.

- **Prototype**

```
int ql_wifi_work_mode_get(ql_wifi_work_mode_e *mode);
```

- **Parameter**

mode:
[Out] Gets current working mode.

QL_WIFI_WORK_MODE_STA	STA mode
QL_WIFI_WORK_MODE_AP0	AP mode
QL_WIFI_WORK_MODE_AP0_STA	AP-STA mode
QL_WIFI_WORK_MODE_AP0_AP1	AP-AP mode

- **Return Value**

0	This function is executed successfully.
-1	It fails to be executed.

2.2.11. ql_wifi_ap_ssid_set

This function sets the SSID in AP mode.

- **Prototype**

```
int ql_wifi_ap_ssid_set(ql_wifi_ap_index_e idx, char *ssid);
```

- **Parameter**

idx:
[In] AP mode index.

QL_WIFI_AP_INDEX_AP0	Index of hotspot 0
QL_WIFI_AP_INDEX_AP1	Index of hotspot 1

ssid:
[In] The set SSID. It should be within 32 bytes and is not empty.

- **Return Value**

0	This function is executed successfully.
-1	It fails to be executed.

2.2.12. ql_wifi_ap_ssid_get

This function gets the SSID in AP mode.

- **Prototype**

```
int ql_wifi_ap_ssid_get(ql_wifi_ap_index_e idx, char *ssid);
```

- **Parameter**

idx:
[In] AP mode index.
QL_WIFI_AP_INDEX_AP0 Index of hotspot 0
QL_WIFI_AP_INDEX_AP1 Index of hotspot 1

ssid:
[Out] The obtained SSID.

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.13. ql_wifi_ap_ssid_hide_set

This function sets to hide SSID in AP mode.

- **Prototype**

```
int ql_wifi_ap_ssid_hide_set(ql_wifi_ap_index_e idx, bool hide);
```

- **Parameter**

idx:
[In] AP mode index.
QL_WIFI_AP_INDEX_AP0 Index of hotspot 0
QL_WIFI_AP_INDEX_AP1 Index of hotspot 1

hide:
[In] Sets whether to hide SSID in AP mode.
true Enable to hide SSID
false Disable to hide SSID

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.14. ql_wifi_ap_ssid_hide_get

This function gets the SSID hiding status in AP mode.

- **Prototype**

```
int ql_wifi_ap_ssid_hide_get(ql_wifi_ap_index_e idx, bool *hide);
```

- **Parameter**

idx:
[In] AP mode index.
QL_WIFI_AP_INDEX_AP0 Index of hotspot 0
QL_WIFI_AP_INDEX_AP1 Index of hotspot 1

hide:
[Out] The obtained SSID hiding status.

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.15. ql_wifi_ap_mode_set

This function sets the working protocol mode in AP mode.

- **Prototype**

```
int ql_wifi_ap_mode_set(ql_wifi_ap_index_e idx, ql_wifi_mode_type_e mode);
```

- **Parameter**

idx:
[In] AP mode index.
QL_WIFI_AP_INDEX_AP0 Index of hotspot 0
QL_WIFI_AP_INDEX_AP1 Index of hotspot 1

mode:
[In] The working protocol mode.

QL_WIFI_MODE_80211B	IEEE 802.11b (2.4 GHZ)
QL_WIFI_MODE_80211BG	IEEE 802.11b/g (2.4 GHZ)
QL_WIFI_MODE_80211BGN	IEEE 802.11b/g/n (2.4 GHz)
QL_WIFI_MODE_80211A	IEEE 802.11a (5 GHz)
QL_WIFI_MODE_80211AN	IEEE 802.11a/n (5 GHz)
QL_WIFI_MODE_80211AC	IEEE 802.11a/c (5 GHz)

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.16. ql_wifi_ap_mode_get

This function gets the working protocol mode in AP mode.

- **Prototype**

```
int ql_wifi_ap_mode_get(ql_wifi_ap_index_e idx, ql_wifi_mode_type_e *mode);
```

- **Parameter**

idx:
[In] AP mode index.

QL_WIFI_AP_INDEX_AP0	Index of hotspot 0
QL_WIFI_AP_INDEX_AP1	Index of hotspot 1

mode:
[Out] The working protocol mode.

QL_WIFI_MODE_80211B	IEEE 802.11b (2.4 GHZ)
QL_WIFI_MODE_80211BG	IEEE 802.11b/g (2.4 GHZ)
QL_WIFI_MODE_80211BGN	IEEE 802.11b/g/n (2.4 GHz)
QL_WIFI_MODE_80211A	IEEE 802.11a (5 GHz)
QL_WIFI_MODE_80211AN	IEEE 802.11a/n (5 GHz)
QL_WIFI_MODE_80211AC	IEEE 802.11a/c (5 GHz)

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.17. ql_wifi_ap_bandwidth_set

This function sets the bandwidth in AP mode.

- **Prototype**

```
int ql_wifi_ap_bandwidth_set(ql_wifi_ap_index_e idx, ql_wifi_bandwidth_type_e bandwidth);
```

- **Parameter**

idx:
[In] AP mode index.

QL_WIFI_AP_INDEX_AP0	Index of hotspot 0
QL_WIFI_AP_INDEX_AP1	Index of hotspot 1

bandwidth:
[In] The working bandwidth.

QL_WIFI_BANDWIDTH_HT20	20 MHz
QL_WIFI_BANDWIDTH_HT40	40 MHz
QL_WIFI_BANDWIDTH_HT80	80 MHz

- **Return Value**

0	This function is executed successfully.
-1	It fails to be executed.

2.2.18. ql_wifi_ap_bandwidth_get

This function gets the bandwidth in AP mode.

- **Prototype**

```
int ql_wifi_ap_bandwidth_get(ql_wifi_ap_index_e idx, ql_wifi_bandwidth_type_e bandwidth);
```

- **Parameter**

idx:
[In] AP mode index.

QL_WIFI_AP_INDEX_AP0	Index of hotspot 0
QL_WIFI_AP_INDEX_AP1	Index of hotspot 1

bandwidth:
[Out] The working bandwidth.

QL_WIFI_BANDWIDTH_HT20	20 MHz
QL_WIFI_BANDWIDTH_HT40	40 MHz
QL_WIFI_BANDWIDTH_HT80	80 MHz

- **Return Value**

0	This function is executed successfully.
-1	It fails to be executed.

2.2.19. ql_wifi_ap_channel_set

This function sets the channel in AP mode.

- **Prototype**

```
int ql_wifi_ap_channel_set(ql_wifi_ap_index_e idx, int channel);
```

- **Parameter**

idx:
[In] AP mode index.

QL_WIFI_AP_INDEX_AP0	Index of hotspot 0
QL_WIFI_AP_INDEX_AP1	Index of hotspot 1

channel:
[In] Set to 2.4 G or 5 G channel.

0/1/2/3/4/5/6/7/8/9/10/11/12/13/14	2.4 G channel
36/40/44/48/52/56/60/64/100/104/108/112/116/120/124/128/132/136/140/144/149/153/157/161/165/175/181	5 G channel

- **Return Value**

0	This function is executed successfully.
-1	It fails to be executed.

2.2.20. ql_wifi_ap_channel_get

This function gets the channel in AP mode.

- **Prototype**

```
int ql_wifi_ap_channel_get(ql_wifi_ap_index_e idx, int *channel);
```

- **Parameter**

idx:
[In] AP mode index.

QL_WIFI_AP_INDEX_AP0	Index of hotspot 0
QL_WIFI_AP_INDEX_AP1	Index of hotspot 1

channel:
[Out] Set to 2.4 G or 5 G channel.

0/1/2/3/4/5/6/7/8/9/10/11/12/13/14	2.4 G channel
36/40/44/48/52/56/60/64/100/104/108/112/116/120/124/128/132/136/140/144/149/153/157/161/165/175/181	5 G channel

- **Return Value**

0	This function is executed successfully.
-1	It fails to be executed.

2.2.21. ql_wifi_ap_auth_set

This function sets the security authentication in AP mode.

- **Prototype**

```
int ql_wifi_ap_auth_set(ql_wifi_ap_index_e idx, ql_wifi_ap_auth_s *auth);
```

- **Parameter**

idx:
[In] AP mode index.

QL_WIFI_AP_INDEX_AP0	Index of hotspot 0
QL_WIFI_AP_INDEX_AP1	Index of hotspot 1

auth:
[In] Security authentication. See ***Chapter 2.2.22.1*** for details.

- **Return Value**

0	This function is executed successfully.
-1	It fails to be executed.

2.2.22. ql_wifi_ap_auth_get

This function gets the security authentication in AP mode.

- **Prototype**

```
int ql_wifi_ap_auth_get(ql_wifi_ap_index_e idx, ql_wifi_ap_auth_s *auth);
```

- **Parameter**

idx:
[In] AP mode index.

QL_WIFI_AP_INDEX_AP0	Index of hotspot 0
QL_WIFI_AP_INDEX_AP1	Index of hotspot 1

auth:
[Out] Security authentication. See ***Chapter 2.2.22.1*** for details.

- **Return Value**

0	This function is executed successfully.
-1	It fails to be executed.

2.2.22.1. ql_wifi_ap_auth_s

The security authentication type is defined as follows, usually it only uses wpa_psk in union:

```
typedef struct {
    int auth;
    union {
        struct {
            int default_index;
            char passwd[4][64];
        } wep;

        struct {
            short pairwise;
            char passwd[64];
            int group_rekey;
        } wpa_psk;
    };
} ql_wifi_ap_auth_s;;
```

- **Parameters**

Type	Parameter	Description
int	*auth*	Security authentication type
int	*default_index*	The configured index of WEP
char	*passwd*	The password of WEP
short	*pairwise*	Encryption
char	*passwd*	The password of WPA/WPA2
int	*group_rekey*	Group key update

2.2.23. ql_wifi_ap_max_sta_set

This function sets the maximum number of connections for terminals in AP mode.

- **Prototype**

```
int ql_wifi_ap_max_sta_set(ql_wifi_ap_index_e idx, int max_sta_num);
```

- **Parameter**

idx:
[In] AP mode index
QL_WIFI_AP_INDEX_AP0 Index of hotspot 0
QL_WIFI_AP_INDEX_AP1 Index of hotspot 1

max_sta_num:
[In] Set the maximum number of connections for terminals in AP mode. Maximum quantity: 16

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.24. ql_wifi_ap_max_sta_get

This function gets the maximum number of connections for terminals in AP mode.

- **Prototype**

```
int ql_wifi_ap_max_sta_get(ql_wifi_ap_index_e idx, int *max_sta_num);
```

- **Parameter**

idx:
[In] AP mode index.

QL_WIFI_AP_INDEX_AP0	Index of hotspot 0
QL_WIFI_AP_INDEX_AP1	Index of hotspot 1

max_sta_num:
[Out] Get the maximum number of connections for terminals in AP mode.

- **Return Value**

0	This function is executed successfully.
-1	It fails to be executed.

2.2.25. ql_wifi_ap_sta_info_get

This function gets the client information in AP mode.

- **Prototype**

```
int ql_wifi_ap_sta_info_get(ql_wifi_ap_index_e idx, ql_wifi_ap_sta_info_s *sta_info);
```

- **Parameter**

idx:
[In] AP mode index.

QL_WIFI_AP_INDEX_AP0	Index of hotspot 0
QL_WIFI_AP_INDEX_AP1	Index of hotspot 1

sta_info:
[Out] The obtained client information. See ***Chapter 2.2.25.1*** for details.

- **Return Value**

0	This function is executed successfully.
-1	It fails to be executed.

2.2.25.1. ql_wifi_ap_sta_info_s

The client information is defined as follows:

```
typedef struct {
    int num;
    struct {
        unsigned char mac[6];
        in_addr_t addr;
        char hostname[33];
    } sta[16];
} ql_wifi_ap_sta_info_s
```

- **Parameters**

Type	Parameter	Description
int	*num*	Number of client devices
unsigned char	*mac*	MAC address of the device
in_addr_t	*addr*	IP address of the device
char	*hostname*	Host name

2.2.26. ql_wifi_ap_start

This function enables the upper application hostapd in AP mode to make the new configuration information effective.

- **Prototype**

```
int ql_wifi_ap_start(ql_wifi_ap_index_e idx);
```

- **Parameter**

idx:
[In] AP mode index.
QL_WIFI_AP_INDEX_AP0 Index of hotspot 0
QL_WIFI_AP_INDEX_AP1 Index of hotspot 1

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.27. ql_wifi_ap_stop

This function disables the upper application hostapd in AP mode.

- **Prototype**

```
int ql_wifi_ap_stop(ql_wifi_ap_index_e idx);
```

- **Parameter**

idx:
[In] AP mode index.
QL_WIFI_AP_INDEX_AP0 Index of hotspot 0
QL_WIFI_AP_INDEX_AP1 Index of hotspot 1

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.28. ql_wifi_ap_restart

This function restarts the upper application hostapd in AP mode to make the new configuration information effective.

- **Prototype**

```
int ql_wifi_ap_restart(ql_wifi_ap_index_e idx);
```

- **Parameter**

idx:
[In] AP mode index.
QL_WIFI_AP_INDEX_AP0 Index of hotspot 0
QL_WIFI_AP_INDEX_AP1 Index of hotspot 1

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.29. ql_wifi_status_get

This function gets current Wi-Fi status.

- **Prototype**

```
int ql_wifi_status_get(ql_wifi_status_e *status);
```

- **Parameter**

status：
[Out] Current Wi-Fi status.

QL_WIFI_STATUS_ENABLE	Currently Wi-Fi is enabled
QL_WIFI_STATUS_DISABLE	Currently Wi-Fi is disabled
QL_WIFI_STATUS_ERR_DRIVER	Driver error
QL_WIFI_STATUS_ERR_SOFTWARE	Software error

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.30. ql_wifi_sta_ssid_set

This function sets the connected SSID in STA mode.

- **Prototype**

```
int ql_wifi_sta_ssid_set(char *ssid);
```

- **Parameter**

ssid:
[In] The set SSID. It should be within 32 bytes and is not empty.

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.31. ql_wifi_sta_ssid_get

This function gets the connected SSID in STA mode.

- **Prototype**

```
int ql_wifi_sta_ssid_get(char *ssid);
```

- **Parameter**

ssid:
[Out] The obtained SSID.

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.32. ql_wifi_sta_auth_set

This function sets the security authentication in STA mode.

- **Prototype**

```
int ql_wifi_sta_auth_set(ql_wifi_sta_auth_s *auth);
```

- **Parameter**

auth:
[In] Set the security authentication. See ***Chapter 2.2.33.1*** for details.

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.33. ql_wifi_sta_auth_get

This function gets the security authentication in STA mode.

- **Prototype**

```
int ql_wifi_sta_auth_get(ql_wifi_sta_auth_s *auth);
```

- **Parameter**

auth:
[Out] The obtained security authentication. See ***Chapter 2.2.33.1*** for details.

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.33.1. ql_wifi_sta_auth_s

The security authentication is defined as follows:

```
typedef struct {
    int auth;
    union {
        struct {
            char passwd[64];
        } wep;

        struct {
            short pairwise;
            char passwd[64];
        } wpa_psk;
    };
} ql_wifi_sta_auth_s;
```

- **Parameters**

Type	Parameter	Description
int	*auth*	Security authentication type
char	*passwd*	The password of WEP
short	*pairwise*	Encryption
char	*passwd*	The password of WPA/WPA2

2.2.34. ql_wifi_sta_connect

This function enables the upper application wpa_supplicant in STA mode to make the new configuration information effective.

- **Prototype**

```
int ql_wifi_sta_connect(void);
```

- **Parameter**

N/A

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.35. ql_wifi_sta_disconnect

This function disables the upper application wpa_supplicant in STA mode.

- **Prototype**

```
int ql_wifi_sta_disconnect(void);
```

- **Parameter**

N/A

- **Return Value**

0 This function is executed successfully.
-1 It fails to be executed.

2.2.36. ql_wifi_sta_status

This function gets STA mode connection status.

- **Prototype**

```
int ql_wifi_sta_status(ql_wifi_station_status_e *status);
```

- **Parameter**

status:
[Out] The connection status in STA mode.

QL_WIFI_STATION_DISABLE	STA mode is disabled
QL_WIFI_STATION_CONNECTED	STA mode is connected
QL_WIFI_STATION_DISCONNECTED	STA mode is disconnected

- **Return Value**

0	This function is executed successfully.
-1	It fails to be executed.

3 Wi-Fi API Example

3.1. STA Mode Configuration

See example *example/wifi/example_wifi.c* for details. Run the program as follows:

```
Please input case: 7
Please input WiFi handle register(0-register/1-unregister): 0      //Register the callback function.

case 1: Configure WiFi
case 2: Enable WiFi
case 3: Activate Hostapd
case 4: Activate WPA Supplicant
case 5: Get WPA Supplicant status
case 6: Get WiFi configuraction
case 7: Register Event Callback
case 8: Enable Module WiFi
case 9: Get WiFi status
case 100: exit
Please input case: 1
Please input WiFi work mode(0-STA/1-AP0/2-AP0+STA/3-AP0+AP1): 0 //Set to STA mode.

Please input STA ssid(maximun size 32): quectel_test        //Set SSID.

Please input STA type of authentication(0-OPEN/2-WPA PSK/3-WPA2 PSK/4-WPA2&WPA): 4    //Set
the authentication.

Please input STA WPA PSK pairwise(0-AUTH/1-TKIP/2-AES): 0      //Set the encryption.

Please input STA WPA PSK password: 12345678              //Set the password.

case 1: Configure WiFi
case 2: Enable WiFi
case 3: Activate Hostapd
case 4: Activate WPA Supplicant
case 5: Get WPA Supplicant status
case 6: Get WiFi configuraction
```

```
case 7: Register Event Callback
case 8: Enable Module WiFi
case 9: Get WiFi status
case 100: exit
Please input case: 2
Please input WiFi status(0-Disable/1-Enable/2-Async Enable/3-Async Disable): 2        //Enable Wi-Fi.
asynchronously
```

3.2. AP Mode Configuration

See example *example/wifi/example_wifi.c* for details. Run the program as follows:

```
Please input case: 7
Please input WiFi handle register(0-register/1-unregister): 0      // Register the callback function.

case 1: Configure WiFi
case 2: Enable WiFi
case 3: Activate Hostapd
case 4: Activate WPA Supplicant
case 5: Get WPA Supplicant status
case 6: Get WiFi configuraction
case 7: Register Event Callback
case 8: Enable Module WiFi
case 9: Get WiFi status
case 100: exit
Please input case: 1

Please input WiFi work mode(0-STA/1-AP0/2-AP0+STA/3-AP0+AP1): 1        //Set to AP mode.

Please input WiFi index 0 ssid(maximun size 32): quectel_test              //Set SSID.

Please input WiFi index 0 IEEE 802.11 mode(0-b/1-bg/2-bgn/3-a/4-an/5-ac): 2     //Set protocol mode.

Please input WiFi index 0 Bandwidth(0-20MHz/1-40MHz/2-80MHz): 0     //Set the bandwidth.

Please input WiFi index 0 channel: 6       //Set the channel

Please input WiFi index 0 maximun station(1-16):16    //Set maximum number of connections

Please input WiFi index 0 type of authentication(0-OPEN/2-WPA PSK/3-WPA2
PSK/4-WPA-PSK&WPA2-PSK): 4    //Set the authentication
```

```
Please input WiFi index 0 WPA PSK pairwise(0-AUTH/1-TKIP/2-AES): 0       //Set the encryption.

Please input WiFi index 0 WPA PSK password: 12345678        //Set the password

case 1: Configure WiFi
case 2: Enable WiFi
case 3: Activate Hostapd
case 4: Activate WPA Supplicant
case 5: Get WPA Supplicant status
case 6: Get WiFi configuraction
case 7: Register Event Callback
case 8: Enable Module WiFi
case 9: Get WiFi status
case 100: exit
Please input case: 2

Please input WiFi status(0-Disable/1-Enable/2-Async Enable/3-Async Disable): 2        //Enable Wi-Fi.
asynchronously
```

3.3. AP-AP Mode Configuration

See example *example/wifi/ap_ap_mode.c* for details. Run the program as follows:

```
ql_wifi_ap_auth_s auth_ap0;
    ql_wifi_ap_auth_s auth_ap1;

    /* setting AP0 */
    ql_wifi_work_mode_set(QL_WIFI_WORK_MODE_AP0_AP1); //Set to AP-AP mode
    ql_wifi_ap_ssid_set(QL_WIFI_AP_INDEX_AP0, "Quectel-WORK_MODE_AP0_AP1"); //Set SSID.
    ql_wifi_ap_mode_set(QL_WIFI_AP_INDEX_AP0, QL_WIFI_MODE_80211BGN); //Set to b/g/n.
    ql_wifi_ap_bandwidth_set(QL_WIFI_AP_INDEX_AP0, QL_WIFI_BANDWIDTH_HT20); //Set the.
bandwidth.
    ql_wifi_ap_channel_set(QL_WIFI_AP_INDEX_AP0, 11);    //Set to 2.4 G channel.
    auth_ap0.auth = QL_WIFI_AUTH_WPA_PSK;      //Set the authentication.
    auth_ap0.wpa_psk.pairwise = QL_WIFI_AUTH_WPA_PAIRWISE_AUTO; //Set the encryption.
    auth_ap0.wpa_psk.group_rekey = 3600; /* one hour */              //Set group key.
    strcpy(auth_ap0.wpa_psk.passwd, "12345678");  //Set the password.
    ql_wifi_ap_auth_set(QL_WIFI_AP_INDEX_AP0, &auth_ap0);

    /* setting AP1 */
    ql_wifi_ap_ssid_set(QL_WIFI_AP_INDEX_AP1, "Quectel-WORK_MODE_AP0_AP1"); //Set SSID.
    ql_wifi_ap_mode_set(QL_WIFI_AP_INDEX_AP1, QL_WIFI_MODE_80211AN);   //Set to a/n mode.
```

```
    ql_wifi_ap_bandwidth_set(QL_WIFI_AP_INDEX_AP1, QL_WIFI_BANDWIDTH_HT20); //Set the
bandwidth.
    ql_wifi_ap_channel_set(QL_WIFI_AP_INDEX_AP1, 149);        //Set to 5G channel.
    auth_ap1.auth = QL_WIFI_AUTH_WPA_PSK;      //Set the authentication
    auth_ap1.wpa_psk.pairwise = QL_WIFI_AUTH_WPA_PAIRWISE_AUTO;     //Set the encryption.
    auth_ap1.wpa_psk.group_rekey = 3600; /* one hour */              //Set group key.
    strcpy(auth_ap1.wpa_psk.passwd, "12345678");  //Set the password.
    ql_wifi_ap_auth_set(QL_WIFI_AP_INDEX_AP1, &auth_ap1);

    ql_wifi_enable();  //Enable Wi-Fi.
    return 0;
```

3.4. AP-STA Mode Configuration

See example *example/wifi/ap_sta_mode.c* for details. Run the program as follows:

```
ql_wifi_ap_auth_s ap_auth;
    ql_wifi_sta_auth_s sta_auth;

    /* setting AP0 */
    ql_wifi_work_mode_set(QL_WIFI_WORK_MODE_AP0_STA); //Set to AP-STA mode.
    ql_wifi_ap_ssid_set(QL_WIFI_AP_INDEX_AP0, "Quectel-WORK_MODE_AP0"); //Set SSID.
    ql_wifi_ap_mode_set(QL_WIFI_AP_INDEX_AP0, QL_WIFI_MODE_80211BGN); //Set to b/g/n
mode.
    ql_wifi_ap_bandwidth_set(QL_WIFI_AP_INDEX_AP0, QL_WIFI_BANDWIDTH_HT20); //Set the
bandwidth.
    ql_wifi_ap_channel_set(QL_WIFI_AP_INDEX_AP0, 11); // Set to 2.4G channel.

    ap_auth.auth = QL_WIFI_AUTH_WPA_PSK;       // Set the authentication.
    ap_auth.wpa_psk.pairwise = QL_WIFI_AUTH_WPA_PAIRWISE_AUTO; //Set the encryption.
    ap_auth.wpa_psk.group_rekey = 3600; /* one hour */  //Set group key.
    strcpy(ap_auth.wpa_psk.passwd, "12345678");    //Set the password.
    ql_wifi_ap_auth_set(QL_WIFI_AP_INDEX_AP0, &ap_auth);


    /* setting STA */
    ql_wifi_sta_ssid_set("Quectel-test");       //Set SSID.

    sta_auth.auth = QL_WIFI_AUTH_WPA_PSK;       //Set the authentication.
    sta_auth.wpa_psk.pairwise = QL_WIFI_AUTH_WPA_PAIRWISE_AUTO;      //Set the encryption.
    strcpy(sta_auth.wpa_psk.passwd, "12345678");   //Set the password.
    ql_wifi_sta_auth_set(&sta_auth);
```

```
ql_wifi_enable();  //Enable Wi-Fi.
return 0;
```

4 Appendix A References

Table 3: Related Documents

SN	Document Name	Remark
[1]	Quectel_EC2x&EG9x&EG25-G_Series_ QuecOpen_Quick_Start_Guide	Quick start guide for QuecOpen solution of EC2x series, EG9x series and EG25-G modules
[2]	Quectel_FC20_Series_Hardware_Design	FC20 Series Hardware Design
[3]	Quectel_FC21_Hardware_Design	FC21 Hardware Design

Table 4: Terms and Abbreviations

Abbreviation	Description
AP	Access Point
API	Application Programming Interface
SDK	Software Development Kit
SSID	Service Set Identifier
STA	Station
WEP	Wired Equivalent Privacy
Wi-Fi	Wireless Fidelity
WPA	Wi-Fi Protected Access