ESP8266_NONOS_MESH_API

V1.0.0

Generated by Doxygen 1.8.10

Wed Feb 3 2016 17:06:45

Contents

1 Module Index **1**

1.1 Modules 1

2 Data Structure Index **3**

2.1 Data Structures 3

3 Module Documentation **5**

3.1 mesh APIs 5

3.1.1 Detailed Description 7

3.1.2 Function Documentation 7

3.1.2.1 espconn_mesh_add_option(struct mesh_header_format ∗head, struct mesh_↩ header_option_format ∗option) 7

3.1.2.2 espconn_mesh_connect(struct espconn ∗usr_esp) 7

3.1.2.3 espconn_mesh_create_option(uint8_t otype, uint8_t ∗ovalue, uint8_t val_len) . . 7

3.1.2.4 espconn_mesh_create_packet(uint8_t ∗dst_addr, uint8_t ∗src_addr, bool p2p, bool piggyback_cr, enum mesh_usr_proto_type proto, uint16_t data_len, bool option, uint16_t ot_len, bool frag, enum mesh_option_type frag_type, bool mf, uint16_t frag_idx, uint16_t frag_id) 8

3.1.2.5 espconn_mesh_deauth_all() 8

3.1.2.6 espconn_mesh_disable(espconn_mesh_callback disable_cb) 9

3.1.2.7 espconn_mesh_disconnect(struct espconn ∗usr_esp) 9

3.1.2.8 espconn_mesh_disp_route_table() 9

3.1.2.9 espconn_mesh_enable(espconn_mesh_callback enable_cb, enum mesh_type type) 10

3.1.2.10 espconn_mesh_encrypt_init(AUTH_MODE mode, uint8_t ∗passwd, uint8_t pw↩ _len) 10

3.1.2.11 espconn_mesh_get_dst_addr(struct mesh_header_format ∗head, uint8_↩ t ∗∗dst_addr) 10

3.1.2.12 espconn_mesh_get_max_hops() 10

3.1.2.13 espconn_mesh_get_node_info(enum mesh_node_type type, uint8_t ∗∗info, uint16_t ∗count) 11

3.1.2.14 espconn_mesh_get_option(struct mesh_header_format ∗head, enum mesh_↩ option_type otype, uint16_t oidx, struct mesh_header_option_format ∗∗option) . 11

3.1.2.15 espconn_mesh_get_router(struct station_config ∗router) 11

3.1.2.16 espconn_mesh_get_src_addr(struct mesh_header_format ∗head, uint8_↩
t ∗∗src_addr) 12
3.1.2.17 espconn_mesh_get_status() 12
3.1.2.18 espconn_mesh_get_sub_dev_count() 12
3.1.2.19 espconn_mesh_get_usr_data(struct mesh_header_format ∗head, uint8_↩
t ∗∗usr_data, uint16_t ∗data_len) 13
3.1.2.20 espconn_mesh_get_usr_data_proto(struct mesh_header_format ∗head, enum
mesh_usr_proto_type ∗proto) 14
3.1.2.21 espconn_mesh_group_id_init(uint8_t ∗grp_id, uint16_t gid_len) 14
3.1.2.22 espconn_mesh_layer(struct ip_addr ∗ip) 14
3.1.2.23 espconn_mesh_local_addr(struct ip_addr ∗ip) 15
3.1.2.24 espconn_mesh_print_ver() 15
3.1.2.25 espconn_mesh_regist_conn_ready_cb(espconn_mesh_usr_callback cb) 15
3.1.2.26 espconn_mesh_regist_usr_cb(espconn_mesh_usr_callback cb) 16
3.1.2.27 espconn_mesh_release_congest() 17
3.1.2.28 espconn_mesh_scan(struct mesh_scan_para_type ∗para) 17
3.1.2.29 espconn_mesh_sent(struct espconn ∗usr_esp, uint8 ∗pdata, uint16 len) 17
3.1.2.30 espconn_mesh_server_init(struct ip_addr ∗ip, uint16_t port) 18
3.1.2.31 espconn_mesh_set_dst_addr(struct mesh_header_format ∗head, uint8_t ∗dst↩
_addr) 18
3.1.2.32 espconn_mesh_set_max_hops(uint8_t max_hops) 18
3.1.2.33 espconn_mesh_set_router(struct station_config ∗router) 19
3.1.2.34 espconn_mesh_set_scan_retries(uint8_t retries) 19
3.1.2.35 espconn_mesh_set_src_addr(struct mesh_header_format ∗head, uint8_t ∗src↩
_addr) 19
3.1.2.36 espconn_mesh_set_ssid_prefix(uint8_t ∗prefix, uint8_t prefix_len) 20
3.1.2.37 espconn_mesh_set_usr_data(struct mesh_header_format ∗head, uint8_t ∗usr↩
_data, uint16_t data_len) 20
3.1.2.38 espconn_mesh_set_usr_data_proto(struct mesh_header_format ∗head, enum
mesh_usr_proto_type proto) 20
3.1.2.39 espconn_mesh_setup_timer(os_timer_t ∗timer, uint32_t time, os_timer_func_↩
t cb, void ∗arg, bool repeat) 20

4 Data Structure Documentation 23

4.1 mesh_header_format Struct Reference 23
4.1.1 Field Documentation 23
4.1.1.1 cp 23
4.1.1.2 cr 23
4.1.1.3 d 23
4.1.1.4 dst_addr 23
4.1.1.5 len 24
4.1.1.6 oe 24
4.1.1.7 option 24

4.1.1.8 p2p . . . 24
4.1.1.9 protocol . . . 24
4.1.1.10 rsv . . . 24
4.1.1.11 src_addr . . . 24
4.1.1.12 ver . . . 24
4.2 mesh_header_option_format Struct Reference . . . 24
4.2.1 Field Documentation . . . 24
4.2.1.1 olen . . . 24
4.2.1.2 otype . . . 25
4.2.1.3 ovalue . . . 25
4.3 mesh_header_option_frag_format Struct Reference . . . 25
4.3.1 Field Documentation . . . 25
4.3.1.1 id . . . 25
4.3.1.2 idx . . . 25
4.3.1.3 mf . . . 25
4.3.1.4 resv . . . 25
4.4 mesh_header_option_header_type Struct Reference . . . 25
4.4.1 Field Documentation . . . 26
4.4.1.1 olist . . . 26
4.4.1.2 ot_len . . . 26
4.5 mesh_scan_para_type Struct Reference . . . 26
4.5.1 Field Documentation . . . 26
4.5.1.1 grp_id . . . 26
4.5.1.2 grp_set . . . 26
4.5.1.3 usr_scan_cb . . . 26
4.6 mesh_sub_node_info Struct Reference . . . 26
4.6.1 Field Documentation . . . 26
4.6.1.1 mac . . . 26
4.6.1.2 sub_count . . . 27

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

mesh APIs 5

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

mesh_header_format 23
mesh_header_option_format 24
mesh_header_option_frag_format 25
mesh_header_option_header_type 25
mesh_scan_para_type 26
mesh_sub_node_info 26

Chapter 3

Module Documentation

3.1 mesh APIs

Mesh APIs.

Functions

- void ∗ espconn_mesh_create_packet (uint8_t ∗dst_addr, uint8_t ∗src_addr, bool p2p, bool piggyback_cr, enum mesh_usr_proto_type proto, uint16_t data_len, bool option, uint16_t ot_len, bool frag, enum mesh_↩ option_type frag_type, bool mf, uint16_t frag_idx, uint16_t frag_id)

    The function is used to create mesh packet.
- void ∗ espconn_mesh_create_option (uint8_t otype, uint8_t ∗ovalue, uint8_t val_len)

    The function is used to create mesh option.
- bool espconn_mesh_add_option (struct mesh_header_format ∗head, struct mesh_header_option_format ∗option)

    The function is used to add mesh option in mesh packet.
- bool espconn_mesh_get_option (struct mesh_header_format ∗head, enum mesh_option_type otype, uint16_t oidx, struct mesh_header_option_format ∗∗option)

    The function is used to get mesh option in mesh packet..
- bool espconn_mesh_get_usr_data (struct mesh_header_format ∗head, uint8_t ∗∗usr_data, uint16_t ∗data↩ _len)

    The function is used to get user data in mesh packet..
- bool espconn_mesh_set_usr_data (struct mesh_header_format ∗head, uint8_t ∗usr_data, uint16_t data_len)

    The function is used to set user data in mesh packet..
- bool espconn_mesh_get_src_addr (struct mesh_header_format ∗head, uint8_t ∗∗src_addr)

    The function is used to get source address of mesh packet.
- bool espconn_mesh_get_dst_addr (struct mesh_header_format ∗head, uint8_t ∗∗dst_addr)

    The function is used to get destination address of mesh packet.
- bool espconn_mesh_set_src_addr (struct mesh_header_format ∗head, uint8_t ∗src_addr)

    The function is used to set source address of mesh packet.
- bool espconn_mesh_set_dst_addr (struct mesh_header_format ∗head, uint8_t ∗dst_addr)

    The function is used to set destination address of mesh packet.
- bool espconn_mesh_get_usr_data_proto (struct mesh_header_format ∗head, enum mesh_usr_proto_type ∗proto)

    The function is used to get protocol used by user data in mesh packet.
- bool espconn_mesh_set_usr_data_proto (struct mesh_header_format ∗head, enum mesh_usr_proto_type proto)

    The function is used to set protocol used by user data in mesh packet.

- bool espconn_mesh_local_addr (struct ip_addr ∗ip)

  Check whether the IP address is mesh local IP address or not.
- bool espconn_mesh_get_node_info (enum mesh_node_type type, uint8_t ∗∗info, uint16_t ∗count)

  The function is used to get the information of mesh node..
- bool espconn_mesh_get_router (struct station_config ∗router)

  The function is used to get router AP information used by mesh node.
- bool espconn_mesh_set_router (struct station_config ∗router)

  The function is used to set router AP information for mesh node.
- bool espconn_mesh_encrypt_init (AUTH_MODE mode, uint8_t ∗passwd, uint8_t pw_len)

  The function is used to init encrypt algorithm and password for mesh AP.
- bool espconn_mesh_group_id_init (uint8_t ∗grp_id, uint16_t gid_len)

  The function is used to init group id for mesh node.
- bool espconn_mesh_regist_conn_ready_cb (espconn_mesh_usr_callback cb)

  The function is used to register user callback. If TCP connection with parent node is ready, mesh will call the user callback..
- bool espconn_mesh_regist_usr_cb (espconn_mesh_usr_callback cb)

  The function is used to register user callback. If child node joins parent, parent will trigger the callback to indicate user.
- bool espconn_mesh_server_init (struct ip_addr ∗ip, uint16_t port)

  The function is used to set server ip and port for mesh node.
- bool espconn_mesh_set_max_hops (uint8_t max_hops)

  The function is used to set max_hops for mesh network..
- bool espconn_mesh_set_ssid_prefix (uint8_t ∗prefix, uint8_t prefix_len)

  The function is used to set SSID prefix for mesh AP.
- bool espconn_mesh_set_scan_retries (uint8_t retries)

  The function is used to set scan retries if no available AP to been found.
- int8_t espconn_mesh_connect (struct espconn ∗usr_esp)

  Try to establish mesh connection to server.
- int8_t espconn_mesh_disconnect (struct espconn ∗usr_esp)

  Disconnect a mesh connection.
- int8_t espconn_mesh_get_status ()

  The function is used to get current status of mesh node.
- int8_t espconn_mesh_sent (struct espconn ∗usr_esp, uint8 ∗pdata, uint16 len)

  Send data through mesh network.
- uint8_t espconn_mesh_get_max_hops ()

  The function is used to get current max hop of mesh network.
- uint8_t espconn_mesh_layer (struct ip_addr ∗ip)

  The function is used to get current max hop of mesh network.
- uint16_t espconn_mesh_get_sub_dev_count ()

  The function is used to get the number of all the sub nodes of current node.
- void espconn_mesh_enable (espconn_mesh_callback enable_cb, enum mesh_type type)

  To enable mesh network.
- void espconn_mesh_disable (espconn_mesh_callback disable_cb)

  To disable mesh network.
- void espconn_mesh_deauth_all ()

  The function is used to reject all the child node.
- void espconn_mesh_disp_route_table ()

  The function is used to display route table of current node.
- void espconn_mesh_print_ver ()

  The function is used to print version of mesh.
- void espconn_mesh_release_congest ()

TThe function is used to discard all the packet to parent.

- void espconn_mesh_scan (struct mesh_scan_para_type ∗para)

  The function is used to scan AP around current node.

- void espconn_mesh_setup_timer (os_timer_t ∗timer, uint32_t time, os_timer_func_t cb, void ∗arg, bool repeat)

  The function is used setup timer with callback function.

3.1.1 Detailed Description

Mesh APIs.

3.1.2 Function Documentation

3.1.2.1 bool espconn_mesh_add_option (struct mesh_header_format ∗ *head,* struct mesh_header_option_format ∗ *option*)

The function is used to add mesh option in mesh packet.

Parameters

struct	mesh_header_format ∗head : mesh packet header
struct	mesh_header_option_format ∗option : option

Returns

true : success
false : fail

3.1.2.2 int8_t espconn_mesh_connect (struct espconn ∗ *usr_esp*)

Try to establish mesh connection to server.

Attention

1. If espconn_mesh_connect fail, returns non-0 value, there is no connection, so it won't enter any espconn callback.

Parameters

struct	espconn ∗usr_esp : the network connection structure, the usr_esp to listen to the connection

Returns

0 : succeed
Non-0 : error code

- ESPCONN_RTE - Routing Problem
- ESPCONN_MEM - Out of memory
- ESPCONN_ISCONN - Already connected
- ESPCONN_ARG - Illegal argument, can't find the corresponding connection according to structure espconn

3.1.2.3 void∗ espconn_mesh_create_option (uint8_t *otype,* uint8_t ∗ *ovalue,* uint8_t *val_len*)

The function is used to create mesh option.

Parameters

uint8_t	otype : option type
uint8_t	∗ovalue : option value
uint8_t	val_len : length of option value

Returns

NULL: create mesh option fail.
addr: the start address of option.

3.1.2.4 void∗ espconn_mesh_create_packet (uint8_t ∗ *dst_addr,* uint8_t ∗ *src_addr,* bool *p2p,* bool *piggyback_cr,* enum mesh_usr_proto_type *proto,* uint16_t *data_len,* bool *option,* uint16_t *ot_len,* bool *frag,* enum mesh_option_type *frag_type,* bool *mf,* uint16_t *frag_idx,* uint16_t *frag_id*)

The function is used to create mesh packet.

Attention

1. If the destination of packet is server or mobile, the dst_addr is the combination of IP address and port of server or mobile.
2. If the destination of packet is node, the dst_addr is the mac address of destination device.
3. If mobile or server try to sent packet to device, mobile or server needs to fill the src_addr with combination of its IP address and port. Mobile or server can fill src_addr with default value which is zero.
4. If the packet is produced by device, device need to fill src_addr with its mac address.
5. Device and mobile should set the piggyback_cr.

Parameters

uint8_t	∗dst_addr : destination address (6 Bytes)
uint8_t	∗src_addr : source address (6 Bytes)
bool	p2p : node-to-node packet
bool	piggyback_cr : piggyback flow request
enum	mesh_usr_proto_type proto : protocol used by user data
uint16_t	data_len : length of user data
bool	option : option flag
uint16_t	ot_len : option total length
bool	frag : fragmentation flag
enum	mesh_option_type frag_type : fragmentation type
bool	mf : more fragmentation
uint16_t	frag_idx : fragmentation index
uint16_t	frag_id : fragmentation id

Returns

NULL: create mesh packet fail.
addr: the start address of mesh packet.

3.1.2.5 void espconn_mesh_deauth_all ( )

The function is used to reject all the child node.

Parameters

null	

Returns

null

3.1.2.6 void espconn_mesh_disable (espconn_mesh_callback *disable_cb*)

To disable mesh network.

Attention

When mesh network is disabed, the system will trigger disable_cb.

Parameters

espconn_↩ mesh_callback	disable_cb : callback function of mesh-disable

Returns

null

3.1.2.7 int8_t espconn_mesh_disconnect (struct espconn ∗ *usr_esp*)

Disconnect a mesh connection.

Attention

Do not call this API in any espconn callback. If needed, please use system task to trigger espconn_mesh_↩ disconnect.

Parameters

struct	espconn ∗usr_esp : the network connection structure

Returns

0 : succeed
Non-0 : error code

- ESPCONN_ARG - illegal argument, can't find the corresponding TCP connection according to structure espconn

3.1.2.8 void espconn_mesh_disp_route_table ( )

The function is used to display route table of current node.

Parameters

null.	

Returns

null.

3.1.2.9 void espconn_mesh_enable (espconn_mesh_callback *enable_cb,* enum mesh_type *type*)

To enable mesh network.

Attention

1. the function should be called in user_init.
2. Therefore, after enable mesh, user should wait for the enable_cb to be triggered.

Parameters

espconn_↩ mesh_callback	enable_cb : callback function of mesh-enable
enum	mesh_type type : type of mesh, local or online.

Returns

null

3.1.2.10 bool espconn_mesh_encrypt_init (AUTH_MODE *mode,* uint8_t ∗ *passwd,* uint8_t *pw_len*)

The function is used to init encrypt algorithm and password for mesh AP.

Attention

1. The API should be called before enable mesh..

Parameters

AUTH_MODE	mode : encrypt algorithm (WPA_PSK, WPA2_PSK, WPA_WPA2_PSK)
uint8_t	∗passwd : password
uint8_t	pw_len : length of password

Returns

true : success
false : fail

3.1.2.11 bool espconn_mesh_get_dst_addr (struct mesh_header_format ∗ *head,* uint8_t ∗∗ *dst_addr*)

The function is used to get destination address of mesh packet.

Parameters

struct	mesh_header_format ∗head : mesh packet header
uint8_t	∗∗dst_addr : destination address

Returns

true : success
false : fail

3.1.2.12 uint8_t espconn_mesh_get_max_hops ( )

The function is used to get current max hop of mesh network.

Parameters

null.	

Returns

the current max hop of mesh

3.1.2.13 bool espconn_mesh_get_node_info (enum mesh_node_type *type,* uint8_t ∗∗ *info,* uint16_t ∗ *count*)

The function is used to get the information of mesh node..

Attention

1. Before enable mesh, you must not use the API.
2. If type is MESH_NODE_PARENT, count is the number of parent (always 1), info is the mac address of paren.
3. If type is MESH_NODE_CHILD, count is the number of children whose hop away from current node is one. Info is the collection of sub node information

Parameters

enum	mesh_node_type type : mesh node type
uint8_t	∗∗info : the information will be saved in ∗info
uint16_t	∗count : the node count in ∗inf

Returns

true : success
false : fail

3.1.2.14 bool espconn_mesh_get_option (struct mesh_header_format ∗ *head,* enum mesh_option_type *otype,* uint16_t *oidx,* struct mesh_header_option_format ∗∗ *option*)

The function is used to get mesh option in mesh packet..

Parameters

struct	mesh_header_format ∗head : mesh packet header
enum	mesh_option_type otype : option type
uint16_t	oidx : option index
struct	mesh_header_option_format ∗∗option : option

Returns

true : sucess, option is pointered to the destination option
false : fail

3.1.2.15 bool espconn_mesh_get_router (struct station_config ∗ *router*)

The function is used to get router AP information used by mesh node.

Attention

1. The API should be called after user receives the first user packet from parent.
2. User should provide the router buffer to save router AP information

Parameters

struct	station_config ∗router : if success, the router AP information will be saved in router

Returns

true : success
false : fail

3.1.2.16 bool espconn_mesh_get_src_addr (struct mesh_header_format ∗ *head*, uint8_t ∗∗ *src_addr*)

The function is used to get source address of mesh packet.

Parameters

struct	mesh_header_format ∗head : mesh packet header
uint8_t	∗∗src_addr : source address

Returns

true : success
false : fail

3.1.2.17 int8_t espconn_mesh_get_status ( )

The function is used to get current status of mesh node.

Attention

1. The API should be called after enable mesh.

Parameters

null	

Returns

MESH_DISABLE: mesh is disabled.
MESH_WIFI_CONN: node is trying to connect parent WIFI AP.
MESH_NET_CONN: node has got its IP address and tries to establish TCP connect with parent.
MESH_ONLINE_AVAIL: online mesh is available.
MESH_ONLINE_AVAIL: online mesh is available.
MESH_LOCAL_AVAIL: local mesh is available.

3.1.2.18 uint16_t espconn_mesh_get_sub_dev_count ( )

The function is used to get the number of all the sub nodes of current node.

Parameters

null.	

Returns

the number of sub nodes

3.1.2.19 bool espconn_mesh_get_usr_data (struct mesh_header_format * *head*, uint8_t ** *usr_data*, uint16_t * *data_len*)

The function is used to get user data in mesh packet..

Parameters

struct	mesh_header_format ∗head : mesh packet header
uint8_t	∗∗usr_data : user data
uint16_t	∗data_len : length of user data

Returns

true : success
false : fail

3.1.2.20 bool espconn_mesh_get_usr_data_proto (struct mesh_header_format ∗ *head,* enum mesh_usr_proto_type ∗ *proto*)

The function is used to get protocol used by user data in mesh packet.

Parameters

struct	mesh_header_format ∗head : mesh packet header
enum	mesh_usr_proto_type ∗proto : protocol of user data

Returns

true : success
false : fail

3.1.2.21 bool espconn_mesh_group_id_init (uint8_t ∗ *grp_id,* uint16_t *gid_len*)

The function is used to init group id for mesh node.

Attention

1. The API should be called before enable mesh.
2. The current group id length must be 6.

Parameters

uint8_t	∗grp_id : group id
uint16_t	gid_len : length of group id

Returns

true : success
false : fail

3.1.2.22 uint8_t espconn_mesh_layer (struct ip_addr ∗ *ip*)

The function is used to get current max hop of mesh network.

Attention

1. If ip is not local IP address, the layer will be one..
2. ip should not be NULL. If the ip is NULL, the layer will be one.

Parameters

struct	ip_addr ∗ip : IP address

Returns

hop away from router

3.1.2.23 bool espconn_mesh_local_addr (struct ip_addr ∗ *ip*)

Check whether the IP address is mesh local IP address or not.

Attention

1. The range of mesh local IP address is 2.255.255.∗ ∼ max_hop.255.255.∗.
2. IP pointer should not be NULL. If the IP pointer is NULL, it will return false.

Parameters

struct	ip_addr ∗ip : IP address

Returns

true : the IP address is mesh local IP address
false : the IP address is not mesh local IP address

3.1.2.24 void espconn_mesh_print_ver ( )

The function is used to print version of mesh.

Parameters

null.	

Returns

null.

3.1.2.25 bool espconn_mesh_regist_conn_ready_cb (espconn_mesh_usr_callback *cb*)

The function is used to register user callback. If TCP connection with parent node is ready, mesh will call the user callback..

Parameters

espconn_↩ mesh_usr_↩ callback	cb : user callback function

Returns

true : success
false : fail

3.1.2.26 bool espconn_mesh_regist_usr_cb (espconn_mesh_usr_callback *cb*)

The function is used to register user callback. If child node joins parent, parent will trigger the callback to indicate user.

/**

Parameters

espconn_↩ mesh_usr_↩ callback	cb : user callback function

Returns

true : success
false : fail

3.1.2.27 void espconn_mesh_release_congest ( )

TThe function is used to discard all the packet to parent.

Parameters

null.	

Returns

null.

3.1.2.28 void espconn_mesh_scan (struct mesh_scan_para_type ∗ *para*)

The function is used to scan AP around current node.

Attention

1. user can scan all the AP or mesh node AP.
2. If you plan to scan all the AP, please clear grp_id and grp_set in para.
3. If you just plan to scan mesh node AP, you should set grp_id and grp_set in para.

Parameters

struct	mesh_scan_para_type ∗para : parameter of scan

Returns

null.

3.1.2.29 int8_t espconn_mesh_sent (struct espconn ∗ *usr_esp,* uint8 ∗ *pdata,* uint16 *len*)

Send data through mesh network.

Attention

Please call espconn_mesh_sent after espconn_sent_callback of the pre-packet.

Parameters

struct	espconn ∗usr_esp : the network connection structure

uint8	∗pdata : pointer of data
uint16	len : data length

Returns

0 : succeed
Non-0 : error code

- ESPCONN_MEM - out of memory
- ESPCONN_ARG - illegal argument, can't find the corresponding network transmission according to structure espconn
- ESPCONN_MAXNUM - buffer of sending data is full
- ESPCONN_IF - send UDP data fail

3.1.2.30 bool espconn_mesh_server_init (struct ip_addr ∗ *ip,* uint16_t *port*)

The function is used to set server ip and port for mesh node.

Attention

1. The API should be called before enable mesh.

Parameters

struct	ip_addr ∗ip : IP address
uint16_t	port : port

Returns

true : success
false : fail

3.1.2.31 bool espconn_mesh_set_dst_addr (struct mesh_header_format ∗ *head,* uint8_t ∗ *dst_addr*)

The function is used to set destination address of mesh packet.

Parameters

struct	mesh_header_format ∗head : mesh packet header
uint8_t	∗dst_addr : destination address

Returns

true : success
false : fail

3.1.2.32 bool espconn_mesh_set_max_hops (uint8_t *max_hops*)

The function is used to set max_hops for mesh network..

Attention

1. The API should be called before enable mesh.

Parameters

uint8_t	max_hops : max hops of mesh network

Returns

true : success
false : fail

3.1.2.33 bool espconn_mesh_set_router (struct station_config ∗ *router*)

The function is used to set router AP information for mesh node.

Attention

1. The API should be called before enable mesh..

Parameters

struct	station_config ∗router : router AP information (ssid, password, bssid (optional))

Returns

true : success
false : fail

3.1.2.34 bool espconn_mesh_set_scan_retries (uint8_t *retries*)

The function is used to set scan retries if no available AP to been found.

Attention

1. If no available AP mesh node, the scan retries will works.
2. If retries should be larger than zero (zero will be failed).
3. One scan will take about 15 ∗ retries seconds at most. If retries is 2, the scan will take 30 seconds at most.

Parameters

uint8_t	retries : scan retry count

Returns

true : success
false : fail

3.1.2.35 bool espconn_mesh_set_src_addr (struct mesh_header_format ∗ *head,* uint8_t ∗ *src_addr*)

The function is used to set source address of mesh packet.

Parameters

struct	mesh_header_format ∗head : mesh packet header

uint8_t	∗src_addr : source address

Returns

true : success
false : fail

3.1.2.36 bool espconn_mesh_set_ssid_prefix (uint8_t ∗ *prefix*, uint8_t *prefix_len*)

The function is used to set SSID prefix for mesh AP.

Attention

1. The API should be called before enable mesh.

Parameters

uint8_t	∗prefix : prefix of SSID
uint8_t	prefix_len : length of prefix

Returns

true : success
false : fail

3.1.2.37 bool espconn_mesh_set_usr_data (struct mesh_header_format ∗ *head*, uint8_t ∗ *usr_data*, uint16_t *data_len*)

The function is used to set user data in mesh packet..

Parameters

struct	mesh_header_format ∗head : mesh packet header
uint8_t	∗usr_data : user data
uint16_t	data_len : length of user data

Returns

true : success
false : fail

3.1.2.38 bool espconn_mesh_set_usr_data_proto (struct mesh_header_format ∗ *head*, enum mesh_usr_proto_type *proto*)

The function is used to set protocol used by user data in mesh packet.

Parameters

struct	mesh_header_format ∗head : mesh packet header
enum	mesh_usr_proto_type proto : protocol of user data

Returns

true : success
false : fail

3.1.2.39 void espconn_mesh_setup_timer (os_timer_t ∗ *timer*, uint32_t *time*, os_timer_func_t *cb*, void ∗ *arg*, bool *repeat*)

The function is used setup timer with callback function.

Parameters

os_timer_t	∗timer : timer
uint32_t	time: timeout time
os_timer_func↩ _t	cb : callback function
void	∗arg : argment
bool	repeat: repeat flag

Returns

null.

Chapter 4

Data Structure Documentation

4.1 mesh_header_format Struct Reference

Data Fields

- uint8_t ver:2
- uint8_t oe: 1
- uint8_t cp: 1
- uint8_t cr: 1
- uint8_t rsv:3
- struct {
      uint8_t d: 1
      uint8_t p2p:1
      uint8_t protocol:6
  } **proto**
- uint16_t len
- uint8_t dst_addr [ESP_MESH_ADDR_LEN]
- uint8_t src_addr [ESP_MESH_ADDR_LEN]
- struct mesh_header_option_header_type option [0]

4.1.1 Field Documentation

4.1.1.1 uint8_t cp

piggyback congest permit in packet

4.1.1.2 uint8_t cr

piggyback congest request in packet

4.1.1.3 uint8_t d

direction, 1:upwards, 0:downwards

4.1.1.4 uint8_t dst_addr[ESP_MESH_ADDR_LEN]

destination address

4.1.1.5 uint16_t len

packet total length (include mesh header)

4.1.1.6 uint8_t oe

option flag

4.1.1.7 struct **mesh_header_option_header_type** option[0]

mesh option

4.1.1.8 uint8_t p2p

node to node packet

4.1.1.9 uint8_t protocol

protocol used by user data;

4.1.1.10 uint8_t rsv

reserved

4.1.1.11 uint8_t src_addr[ESP_MESH_ADDR_LEN]

source address

4.1.1.12 uint8_t ver

version of mesh

The documentation for this struct was generated from the following file:

- include/espressif/mesh.h

4.2 mesh_header_option_format Struct Reference

Data Fields

- uint8_t otype
- uint8_t olen
- uint8_t ovalue [0]

4.2.1 Field Documentation

4.2.1.1 uint8_t olen

current option length

4.2.1.2 uint8_t otype

option type

4.2.1.3 uint8_t ovalue[0]

option value

The documentation for this struct was generated from the following file:

- include/espressif/mesh.h

4.3 mesh_header_option_frag_format Struct Reference

Data Fields

- uint16_t id
- struct {
    uint16_t resv:1
    uint16_t mf:1
    uint16_t idx:14
  } **offset**

4.3.1 Field Documentation

4.3.1.1 uint16_t id

identify of fragment

4.3.1.2 uint16_t idx

fragment offset

4.3.1.3 uint16_t mf

more fragment

4.3.1.4 uint16_t resv

reserved

The documentation for this struct was generated from the following file:

- include/espressif/mesh.h

4.4 mesh_header_option_header_type Struct Reference

Data Fields

- uint16_t ot_len
- struct mesh_header_option_format olist [0]

4.4.1 Field Documentation

4.4.1.1 struct **mesh_header_option_format** olist[0]

option list

4.4.1.2 uint16_t ot_len

option total length

The documentation for this struct was generated from the following file:

- include/espressif/mesh.h

4.5 mesh_scan_para_type Struct Reference

Data Fields

- espconn_mesh_scan_callback usr_scan_cb
- uint8_t grp_id [ESP_MESH_GROUP_ID_LEN]
- bool grp_set

4.5.1 Field Documentation

4.5.1.1 uint8_t grp_id[ESP_MESH_GROUP_ID_LEN]

group id

4.5.1.2 bool grp_set

group set

4.5.1.3 espconn_mesh_scan_callback usr_scan_cb

scan done callback

The documentation for this struct was generated from the following file:

- include/espressif/mesh.h

4.6 mesh_sub_node_info Struct Reference

Data Fields

- uint16_t sub_count
- uint8_t mac [ESP_MESH_ADDR_LEN]

4.6.1 Field Documentation

4.6.1.1 uint8_t mac[ESP_MESH_ADDR_LEN]

mac address of child

4.6.1.2 uint16_t sub_count

the count of sub-node

The documentation for this struct was generated from the following file:

- include/espressif/mesh.h