

OTA TCP ESP32

Generated by Doxygen 1.9.1

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 crypt_buffer_t Struct Reference	5
3.1.1 Member Data Documentation	5
3.1.1.1 len	5
3.1.1.2 val	5
3.2 state_machine_params_t Struct Reference	5
3.2.1 Member Data Documentation	6
3.2.1.1 firmware_bytes_read	6
3.2.1.2 firmware_size	6
3.2.1.3 hash	6
3.2.1.4 semaphore	6
3.2.1.5 state	6
4 File Documentation	7
4.1 components/auth_hmac/auth_hmac.c File Reference	7
4.1.1 Macro Definition Documentation	8
4.1.1.1 HMAC_SHA256_LEN	8
4.1.2 Function Documentation	8
4.1.2.1 auth_hmac_generate_nonce()	8
4.1.2.2 auth_hmac_set_hmac_psk()	8
4.1.2.3 auth_hmac_verify_response()	8
4.1.3 Variable Documentation	9
4.1.3.1 is_set	9
4.1.3.2 len	9
4.1.3.3 val	9
4.2 components/auth_hmac/include/auth_hmac.h File Reference	10
4.2.1 Macro Definition Documentation	11
4.2.1.1 AUTH_HMAC_MAX_BUFFER_LEN	11
4.2.1.2 AUTH_HMAC_NONCE_LEN	11
4.2.2 Function Documentation	11
4.2.2.1 auth_hmac_generate_nonce()	11
4.2.2.2 auth_hmac_set_hmac_psk()	11
4.2.2.3 auth_hmac_verify_response()	12
4.3 components/msg_parser/include/msg_parser.h File Reference	12
4.3.1 Macro Definition Documentation	13
4.3.1.1 MSG_PARSER_BUF_LEN_BYTES	13
4.3.2 Function Documentation	13

4.3.2.1 msg_parser_build_firmware_ack()	14
4.3.2.2 msg_parser_build_ota_ack()	14
4.3.2.3 msg_parser_clean()	14
4.3.2.4 msg_parser_init()	15
4.3.2.5 msg_parser_run()	15
4.4 components/msg_parser/msg_parser.c File Reference	15
4.4.1 Macro Definition Documentation	16
4.4.1.1 FIRMWARE_ACK_SIZE_IN_BYTES	16
4.4.1.2 FIRMWARE_LEN_SIZE_IN_BYTES	16
4.4.1.3 HASH_SIZE_IN_BYTES	17
4.4.1.4 HEADER_SIZE_IN_BYTES	17
4.4.1.5 OTA_ACK_ERR_SIZE_IN_BYTE	17
4.4.1.6 OTA_ACK_FAIL_CODE	17
4.4.1.7 OTA_ACK_LEN_SIZE_IN_BYTES	17
4.4.1.8 OTA_ACK_OK_CODE	17
4.4.1.9 OTA_ACK_SIZE_IN_BYTES	17
4.4.2 Enumeration Type Documentation	17
4.4.2.1 msg_parser_states_e	17
4.4.3 Function Documentation	18
4.4.3.1 msg_parser_build_firmware_ack()	18
4.4.3.2 msg_parser_build_ota_ack()	18
4.4.3.3 msg_parser_clean()	19
4.4.3.4 msg_parser_init()	19
4.4.3.5 msg_parser_run()	19
4.5 components/ota_manager/include/ota_manager.h File Reference	20
4.5.1 Function Documentation	21
4.5.1.1 ota_check_rollback()	21
4.5.1.2 ota_process_end()	21
4.5.1.3 ota_process_init()	22
4.5.1.4 ota_process_write_block()	22
4.6 components/ota_manager/ota_manager.c File Reference	22
4.6.1 Macro Definition Documentation	23
4.6.1.1 HASH_SIZE_IN_BYTES	23
4.6.2 Function Documentation	23
4.6.2.1 ota_check_rollback()	24
4.6.2.2 ota_process_end()	24
4.6.2.3 ota_process_init()	24
4.6.2.4 ota_process_write_block()	25
4.7 components/sys_feedback/include/sys_feedback.h File Reference	25
4.7.1 Enumeration Type Documentation	26
4.7.1.1 sys_feedback_mode_t	26
4.7.2 Function Documentation	27

4.7.2.1 sys_feedback_init()	27
4.7.2.2 sys_feedback_set_normal_mode()	27
4.7.2.3 sys_feedback_set_update_mode()	27
4.7.2.4 sys_feedback_whoiam()	27
4.8 components/sys_feedback/sys_feedback.c File Reference	28
4.8.1 Macro Definition Documentation	28
4.8.1.1 DELAY_FEEDBACK_TASK_MS	28
4.8.1.2 FEEDBACK_GPIO	29
4.8.1.3 FEEDBACK_QUEUE_LEN	29
4.8.1.4 PINNED_CORE	29
4.8.2 Function Documentation	29
4.8.2.1 sys_feedback_init()	29
4.8.2.2 sys_feedback_set_normal_mode()	29
4.8.2.3 sys_feedback_set_update_mode()	29
4.8.2.4 sys_feedback_whoiam()	29
4.9 components/sys_initializer/include/sys_initializer.h File Reference	30
4.9.1 Function Documentation	31
4.9.1.1 sys_initializer_init()	31
4.10 components/sys_initializer/sys_initializer.c File Reference	31
4.10.1 Function Documentation	31
4.10.1.1 sys_initializer_init()	32
4.11 components/tcp_tls/include/tcp_tls.h File Reference	32
4.11.1 Macro Definition Documentation	33
4.11.1.1 TCP_TLS_MAX_BUFFER_LEN	33
4.11.2 Function Documentation	33
4.11.2.1 tcp_tls_init()	33
4.11.2.2 tcp_tls_set_server_cert()	33
4.11.2.3 tcp_tls_set_server_key()	34
4.12 components/tcp_tls/tcp_tls.c File Reference	34
4.12.1 Macro Definition Documentation	35
4.12.1.1 COUNT_NEEDED_TO_START_TCP_SOCKET	35
4.12.1.2 DELAY_AFTER_UPDATE_MS	35
4.12.1.3 KEEPCOUNT	35
4.12.1.4 KEEPIDLE_TIME_SEC	35
4.12.1.5 KEEPINTERVAL_SEC	36
4.12.1.6 PINNED_CORE	36
4.12.1.7 RX_TIMEOUT_SEC	36
4.12.1.8 RX_TIMEOUT_USEC	36
4.12.1.9 TCP_BUFFER_LEN_BYTES	36
4.12.2 Function Documentation	36
4.12.2.1 tcp_tls_init()	36
4.12.2.2 tcp_tls_set_server_cert()	36

4.12.2.3 tcp_tls_set_server_key()	37
4.13 components/types/types.h File Reference	37
4.13.1 Enumeration Type Documentation	38
4.13.1.1 types_error_code_e	38
4.14 components/wifi_ap/include/wifi_ap.h File Reference	38
4.14.1 Macro Definition Documentation	39
4.14.1.1 WIFI_AP_PASSWORD_MAX_LEN	39
4.14.1.2 WIFI_AP_SSID_MAX_LEN	39
4.14.2 Function Documentation	39
4.14.2.1 wifi_ap_init()	40
4.14.2.2 wifi_ap_set_password()	40
4.14.2.3 wifi_ap_set_ssid()	40
4.15 components/wifi_ap/wifi_ap.c File Reference	40
4.15.1 Macro Definition Documentation	41
4.15.1.1 MAX_CLIENTS	41
4.15.1.2 PASSWORD_MIN_LEN	41
4.15.1.3 WIFI_CHANNEL	41
4.15.2 Function Documentation	42
4.15.2.1 wifi_ap_init()	42
4.15.2.2 wifi_ap_set_password()	42
4.15.2.3 wifi_ap_set_ssid()	42
4.16 main/main.c File Reference	42
4.16.1 Macro Definition Documentation	43
4.16.1.1 RESET_DELAY_MS	43
4.16.1.2 VERSION_MAJOR	43
4.16.1.3 VERSION_MINOR	43
4.16.1.4 VERSION_PATCH	44
4.16.2 Function Documentation	44
4.16.2.1 app_main()	44

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

crypt_buffer_t	5
state_machine_params_t	5

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

components/auth_hmac/auth_hmac.c	7
components/auth_hmac/include/auth_hmac.h	10
components/msg_parser/msg_parser.c	15
components/msg_parser/include/msg_parser.h	12
components/ota_manager/ota_manager.c	22
components/ota_manager/include/ota_manager.h	20
components/sys_feedback/sys_feedback.c	28
components/sys_feedback/include/sys_feedback.h	25
components/sys_initializer/sys_initializer.c	31
components/sys_initializer/include/sys_initializer.h	30
components/tcp_tls/tcp_tls.c	34
components/tcp_tls/include/tcp_tls.h	32
components/types/types.h	37
components/wifi_ap/wifi_ap.c	40
components/wifi_ap/include/wifi_ap.h	38
main/main.c	42

Chapter 3

Class Documentation

3.1 crypt_buffer_t Struct Reference

Public Attributes

- uint8_t [val](#) [[TCP_TLS_MAX_BUFFER_LEN](#)]
- size_t [len](#)

3.1.1 Member Data Documentation

3.1.1.1 len

```
size_t crypt_buffer_t::len
```

3.1.1.2 val

```
uint8_t crypt_buffer_t::val [TCP\_TLS\_MAX\_BUFFER\_LEN]
```

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

- [components/tcp_tls/tcp_tls.c](#)

3.2 state_machine_params_t Struct Reference

Public Attributes

- [msg_parser_states_e](#) [state](#)
- uint32_t [firmware_size](#)
- uint32_t [firmware_bytes_read](#)
- uint8_t [hash](#) [[HASH_SIZE_IN_BYTES](#)]
- SemaphoreHandle_t [semaphore](#)

3.2.1 Member Data Documentation

3.2.1.1 firmware_bytes_read

`uint32_t state_machine_params_t::firmware_bytes_read`

3.2.1.2 firmware_size

`uint32_t state_machine_params_t::firmware_size`

3.2.1.3 hash

`uint8_t state_machine_params_t::hash[HASH_SIZE_IN_BYTES]`

3.2.1.4 semaphore

`SemaphoreHandle_t state_machine_params_t::semaphore`

3.2.1.5 state

`msg_parser_states_e state_machine_params_t::state`

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

- [components/msg_parser/msg_parser.c](#)

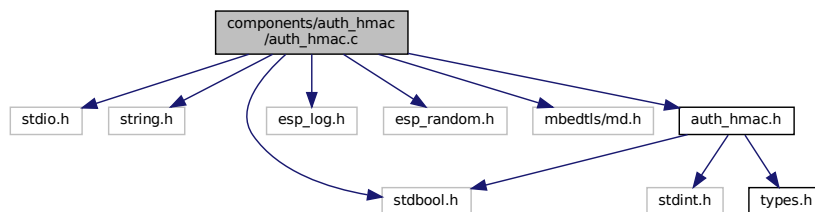
Chapter 4

File Documentation

4.1 components/auth_hmac/auth_hmac.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#include "esp_log.h"
#include "esp_random.h"
#include "mbedtls/md.h"
#include "auth_hmac.h"
```

Include dependency graph for auth_hmac.c:



Macros

- `#define HMAC_SHA256_LEN (32U)`

Functions

- `types_error_code_e auth_hmac_set_hmac_psk (const uint8_t *key, const size_t len)`
pre-shared key setter
- `void auth_hmac_generate_nonce (uint8_t *nonce, size_t len)`
Generate a random nonce for HMAC authentication.
- `bool auth_hmac_verify_response (const uint8_t *nonce, size_t nonce_len, const uint8_t *received_hmac, size_t received_len)`
Verify the HMAC response using the nonce and shared key.

4.1.1 Macro Definition Documentation

4.1.1.1 HMAC_SHA256_LEN

```
#define HMAC_SHA256_LEN (32U)
```

4.1.2 Function Documentation

4.1.2.1 auth_hmac_generate_nonce()

```
void auth_hmac_generate_nonce (
    uint8_t * nonce,
    size_t len )
```

Generate a random nonce for HMAC authentication.

Parameters

<i>nonce</i>	[out]: Pointer to the buffer where the nonce will be stored
<i>len</i>	[in]: Length of the nonce in bytes

4.1.2.2 auth_hmac_set_hmac_psk()

```
types_error_code_e auth_hmac_set_hmac_psk (
    const uint8_t * key,
    const size_t len )
```

pre-shared key setter

Parameters

<i>key</i>	[in]: pre-shared key
<i>len</i>	[in]: pre-shared key length in bytes

4.1.2.3 auth_hmac_verify_response()

```
bool auth_hmac_verify_response (
    const uint8_t * nonce,
```

```
size_t nonce_len,  
const uint8_t * received_hmac,  
size_t received_len )
```

Verify the HMAC response using the nonce and shared key.

Parameters

<i>nonce</i>	[in]: Pointer to the nonce used for HMAC generation
<i>nonce_len</i>	[in]: Length of the nonce in bytes
<i>received_hmac</i>	[in]: Pointer to the received HMAC response
<i>received_len</i>	[in]: Length of the received HMAC response in bytes

Returns

true if the HMAC is valid, false otherwise

4.1.3 Variable Documentation

4.1.3.1 is_set

```
bool is_set
```

4.1.3.2 len

```
size_t len
```

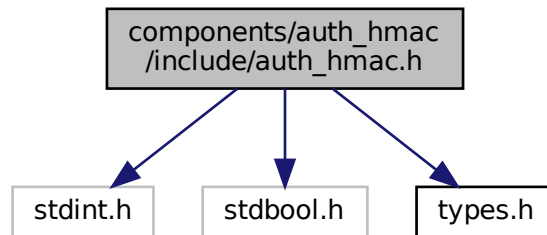
4.1.3.3 val

```
uint8_t val[AUTH_HMAC_MAX_BUFFER_LEN]
```

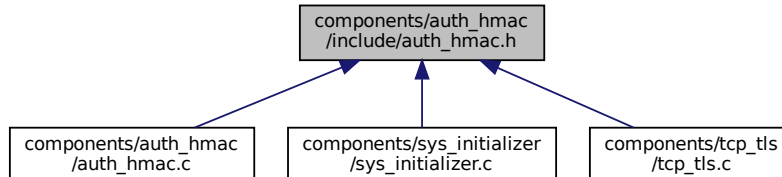
4.2 components/auth_hmac/include/auth_hmac.h File Reference

```
#include <stdint.h>
#include <stdbool.h>
#include "types.h"
```

Include dependency graph for auth_hmac.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define [AUTH_HMAC_MAX_BUFFER_LEN](#) (48U)
- #define [AUTH_HMAC_NONCE_LEN](#) (16U)

Functions

- [types_error_code_e auth_hmac_set_hmac_psk](#) (const uint8_t *key, const size_t len)
pre-shared key setter
- void [auth_hmac_generate_nonce](#) (uint8_t *nonce, size_t len)
Generate a random nonce for HMAC authentication.
- bool [auth_hmac_verify_response](#) (const uint8_t *nonce, size_t nonce_len, const uint8_t *received_hmac, size_t received_len)
Verify the HMAC response using the nonce and shared key.

4.2.1 Macro Definition Documentation

4.2.1.1 AUTH_HMAC_MAX_BUFFER_LEN

```
#define AUTH_HMAC_MAX_BUFFER_LEN (48U)
```

4.2.1.2 AUTH_HMAC_NONCE_LEN

```
#define AUTH_HMAC_NONCE_LEN (16U)
```

4.2.2 Function Documentation

4.2.2.1 auth_hmac_generate_nonce()

```
void auth_hmac_generate_nonce (
    uint8_t * nonce,
    size_t len )
```

Generate a random nonce for HMAC authentication.

Parameters

<i>nonce</i>	[out]: Pointer to the buffer where the nonce will be stored
<i>len</i>	[in]: Length of the nonce in bytes

4.2.2.2 auth_hmac_set_hmac_psk()

```
types_error_code_e auth_hmac_set_hmac_psk (
    const uint8_t * key,
    const size_t len )
```

pre-shared key setter

Parameters

<i>key</i>	[in]: pre-shared key
<i>len</i>	[in]: pre-shared key length in bytes

4.2.2.3 auth_hmac_verify_response()

```
bool auth_hmac_verify_response (
    const uint8_t * nonce,
    size_t nonce_len,
    const uint8_t * received_hmac,
    size_t received_len )
```

Verify the HMAC response using the nonce and shared key.

Parameters

<i>nonce</i>	[in]: Pointer to the nonce used for HMAC generation
<i>nonce_len</i>	[in]: Length of the nonce in bytes
<i>received_hmac</i>	[in]: Pointer to the received HMAC response
<i>received_len</i>	[in]: Length of the received HMAC response in bytes

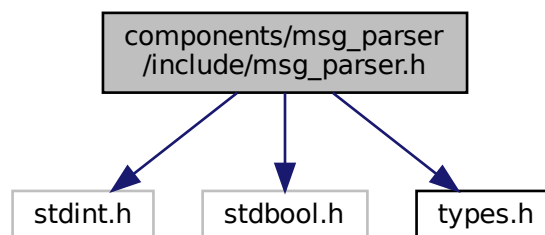
Returns

true if the HMAC is valid, false otherwise

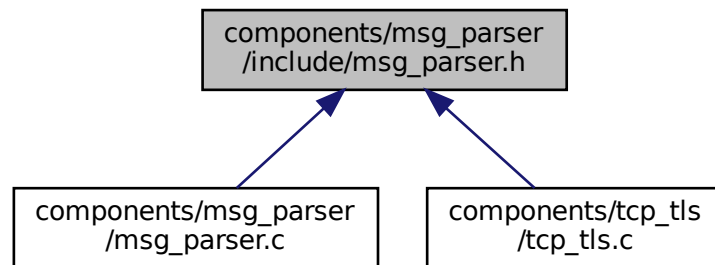
4.3 components/msg_parser/include/msg_parser.h File Reference

```
#include <stdint.h>
#include <stdbool.h>
#include "types.h"
```

Include dependency graph for msg_parser.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define MSG_PARSER_BUF_LEN_BYTES (10U)`

Functions

- `types_error_code_e msg_parser_init (void)`
Initialize the msg_parser component.
- `types_error_code_e msg_parser_run (const uint8_t *p_data, const uint16_t len, uint32_t *p_out_bytes_read)`
Msg_parser state machine, responsible for parse the incoming messagens.
- `void msg_parser_clean (void)`
Reset msg_parser state machine parameters.
- `types_error_code_e msg_parser_build_firmware_ack (uint8_t *p_buffer, const uint8_t len, uint8_t *p_out_↵ len)`
Build the firmware ack message.
- `types_error_code_e msg_parser_build_ota_ack (uint8_t *p_buffer, const uint8_t len, const bool status, const uint32_t bytes_read, uint8_t *p_out_len)`

4.3.1 Macro Definition Documentation

4.3.1.1 MSG_PARSER_BUF_LEN_BYTES

```
#define MSG_PARSER_BUF_LEN_BYTES (10U)
```

4.3.2 Function Documentation

4.3.2.1 msg_parser_build_firmware_ack()

```
types_error_code_e msg_parser_build_firmware_ack (
    uint8_t * p_buffer,
    const uint8_t len,
    uint8_t * p_out_len )
```

Build the firmware ack message.

Parameters

<i>p_buffer</i>	[in]: Message data buffer
<i>len</i>	[in]: Message data buffer length
<i>p_out_len</i>	[out]: Built frame length

Returns

types_error_code_e

4.3.2.2 msg_parser_build_ota_ack()

```
types_error_code_e msg_parser_build_ota_ack (
    uint8_t * p_buffer,
    const uint8_t len,
    const bool status,
    const uint32_t bytes_read,
    uint8_t * p_out_len )
```

Parameters

<i>p_buffer</i>	[in]: Message data buffer
<i>len</i>	[in]: Message data buffer length
<i>status</i>	[in]: True for success e false for fail
<i>bytes_read</i>	[in]: Number of bytes read
<i>p_out_len</i>	[out]: Built frame length

Returns

types_error_code_e

4.3.2.3 msg_parser_clean()

```
void msg_parser_clean (
    void )
```

Reset msg_parser state machine parameters.

4.3.2.4 msg_parser_init()

```
types_error_code_e msg_parser_init (
    void )
```

Initialize the msg_parser component.

Returns

types_error_code_e

4.3.2.5 msg_parser_run()

```
types_error_code_e msg_parser_run (
    const uint8_t * p_data,
    const uint16_t len,
    uint32_t * p_out_bytes_read )
```

Msg_parser state machine, responsible for parse the incoming messagens.

Parameters

<i>p_data</i>	[in]: Message data buffer
<i>len</i>	[in]: Message data buffer length
<i>p_out_bytes_read</i>	[out]: Number os bytes read

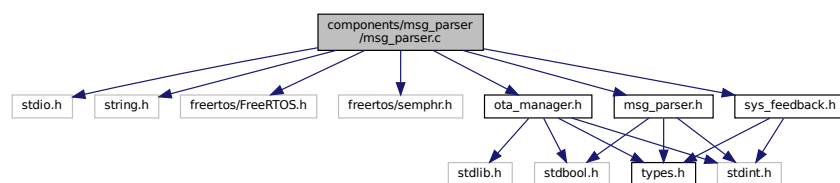
Returns

types_error_code_e

4.4 components/msg_parser/msg_parser.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/semphr.h"
#include "ota_manager.h"
#include "sys_feedback.h"
#include "msg_parser.h"
```

Include dependency graph for msg_parser.c:



Classes

- struct [state_machine_params_t](#)

Macros

- `#define FIRMWARE_LEN_SIZE_IN_BYTES (4U)`
- `#define HASH_SIZE_IN_BYTES (32U)`
- `#define HEADER_SIZE_IN_BYTES (FIRMWARE_LEN_SIZE_IN_BYTES + HASH_SIZE_IN_BYTES)`
- `#define OTA_ACK_SIZE_IN_BYTES (6U)`
- `#define OTA_ACK_LEN_SIZE_IN_BYTES (4U)`
- `#define OTA_ACK_ERR_SIZE_IN_BYTE (2U)`
- `#define OTA_ACK_OK_CODE (100U)`
- `#define OTA_ACK_FAIL_CODE (100U)`
- `#define FIRMWARE_ACK_SIZE_IN_BYTES (4U)`

Enumerations

- enum [msg_parser_states_e](#) { [READ_HEADER](#) , [START_OTA](#) , [WRITE_FIRMWARE](#) }

Functions

- [types_error_code_e msg_parser_init](#) (void)
Initialize the msg_parser component.
- [types_error_code_e msg_parser_run](#) (const uint8_t *p_data, const uint16_t len, uint32_t *p_out_bytes_read)
Msg_parser state machine, responsible for parse the incoming messagens.
- void [msg_parser_clean](#) (void)
Reset msg_parser state machine parameters.
- [types_error_code_e msg_parser_build_firmware_ack](#) (uint8_t *p_buffer, const uint8_t len, uint8_t *p_out_↔ len)
Build the firmware ack message.
- [types_error_code_e msg_parser_build_ota_ack](#) (uint8_t *p_buffer, const uint8_t len, const bool status, const uint32_t bytes_read, uint8_t *p_out_len)

4.4.1 Macro Definition Documentation

4.4.1.1 FIRMWARE_ACK_SIZE_IN_BYTES

```
#define FIRMWARE_ACK_SIZE_IN_BYTES (4U)
```

4.4.1.2 FIRMWARE_LEN_SIZE_IN_BYTES

```
#define FIRMWARE_LEN_SIZE_IN_BYTES (4U)
```

4.4.1.3 HASH_SIZE_IN_BYTES

```
#define HASH_SIZE_IN_BYTES (32U)
```

4.4.1.4 HEADER_SIZE_IN_BYTES

```
#define HEADER_SIZE_IN_BYTES (FIRMWARE_LEN_SIZE_IN_BYTES + HASH_SIZE_IN_BYTES)
```

4.4.1.5 OTA_ACK_ERR_SIZE_IN_BYTE

```
#define OTA_ACK_ERR_SIZE_IN_BYTE (2U)
```

4.4.1.6 OTA_ACK_FAIL_CODE

```
#define OTA_ACK_FAIL_CODE (100U)
```

4.4.1.7 OTA_ACK_LEN_SIZE_IN_BYTES

```
#define OTA_ACK_LEN_SIZE_IN_BYTES (4U)
```

4.4.1.8 OTA_ACK_OK_CODE

```
#define OTA_ACK_OK_CODE (100U)
```

4.4.1.9 OTA_ACK_SIZE_IN_BYTES

```
#define OTA_ACK_SIZE_IN_BYTES (6U)
```

4.4.2 Enumeration Type Documentation

4.4.2.1 msg_parser_states_e

```
enum msg_parser_states_e
```

Enumerator

READ_HEADER	
START_OTA	
WRITE_FIRMWARE	

4.4.3 Function Documentation

4.4.3.1 msg_parser_build_firmware_ack()

```
types_error_code_e msg_parser_build_firmware_ack (
    uint8_t * p_buffer,
    const uint8_t len,
    uint8_t * p_out_len )
```

Build the firmware ack message.

Parameters

<i>p_buffer</i>	[in]: Message data buffer
<i>len</i>	[in]: Message data buffer length
<i>p_out_len</i>	[out]: Built frame length

Returns

types_error_code_e

4.4.3.2 msg_parser_build_ota_ack()

```
types_error_code_e msg_parser_build_ota_ack (
    uint8_t * p_buffer,
    const uint8_t len,
    const bool status,
    const uint32_t bytes_read,
    uint8_t * p_out_len )
```

Parameters

<i>p_buffer</i>	[in]: Message data buffer
<i>len</i>	[in]: Message data buffer length
<i>status</i>	[in]: True for success e false for fail
<i>bytes_read</i>	[in]: Number of bytes read
<i>p_out_len</i>	[out]: Built frame length

Returns

types_error_code_e

4.4.3.3 msg_parser_clean()

```
void msg_parser_clean (  
    void )
```

Reset msg_parser state machine parameters.

4.4.3.4 msg_parser_init()

```
types_error_code_e msg_parser_init (  
    void )
```

Initialize the msg_parser component.

Returns

types_error_code_e

4.4.3.5 msg_parser_run()

```
types_error_code_e msg_parser_run (  
    const uint8_t * p_data,  
    const uint16_t len,  
    uint32_t * p_out_bytes_read )
```

Msg_parser state machine, responsible for parse the incoming messagens.

Parameters

<i>p_data</i>	[in]: Message data buffer
<i>len</i>	[in]: Message data buffer length
<i>p_out_bytes_read</i>	[out]: Number os bytes read

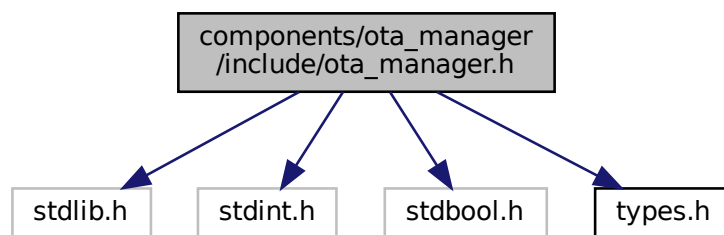
Returns

types_error_code_e

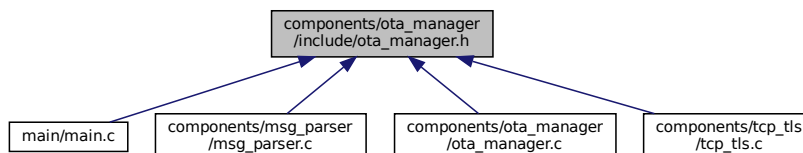
4.5 components/ota_manager/include/ota_manager.h File Reference

```
#include <stdlib.h>
#include <stdint.h>
#include <stdbool.h>
#include "types.h"
```

Include dependency graph for ota_manager.h:



This graph shows which files directly or indirectly include this file:



Functions

- [types_error_code_e ota_process_init](#) (const size_t, const uint8_t *)

Initializes an Over-The-Air (OTA) update process by setting the firmware size, copying the hash, selecting the next OTA partition, and preparing the partition for writing. It also initializes a SHA-256 context for hash computation and returns an appropriate error code based on the success or failure of the initialization steps.

- [types_error_code_e ota_process_write_block](#) (const uint8_t *, const size_t)

Writes a block of data to an ongoing Over-The-Air (OTA) update process, verifies the integrity of the data using SHA-256 hashing, and checks the firmware size against the expected size. It returns an error code indicating the status of the operation, such as success, failure, or in-progress, and handles errors like mismatched firmware size or hash computation failures.

- [types_error_code_e ota_process_end](#) (bool)

Concludes an ongoing OTA (Over-The-Air) update process, ensuring proper cleanup and validation. It checks if the system is healthy, frees allocated resources, finalizes the OTA update, sets the new boot partition, and returns an appropriate error code based on the operation's success or failure.

- void `ota_check_rollback` (bool)

Evaluates the health of the system and manages OTA rollback behavior based on the firmware state. If the system is unhealthy or the firmware verification fails, it triggers a rollback and reboot; otherwise, it marks the firmware as valid and cancels the rollback.

4.5.1 Function Documentation

4.5.1.1 `ota_check_rollback()`

```
void ota_check_rollback (
    bool is_healthy )
```

Evaluates the health of the system and manages OTA rollback behavior based on the firmware state. If the system is unhealthy or the firmware verification fails, it triggers a rollback and reboot; otherwise, it marks the firmware as valid and cancels the rollback.

Parameters

<code>is_healthy</code>	true when system is healthy, false otherwise
-------------------------	--

4.5.1.2 `ota_process_end()`

```
types_error_code_e ota_process_end (
    bool is_healthy )
```

Concludes an ongoing OTA (Over-The-Air) update process, ensuring proper cleanup and validation. It checks if the system is healthy, frees allocated resources, finalizes the OTA update, sets the new boot partition, and returns an appropriate error code based on the operation's success or failure.

Parameters

<code>is_healthy</code>	true when writing process was successful
-------------------------	--

Returns

`types_error_code_e`

4.5.1.3 ota_process_init()

```
types_error_code_e ota_process_init (
    const size_t img_size,
    const uint8_t * hash )
```

Initializes an Over-The-Air (OTA) update process by setting the firmware size, copying the hash, selecting the next OTA partition, and preparing the partition for writing. It also initializes a SHA-256 context for hash computation and returns an appropriate error code based on the success or failure of the initialization steps.

Parameters

<i>img_size</i>	Firmware size to be updated
<i>hash</i>	Received hash

Returns

types_error_code_e

4.5.1.4 ota_process_write_block()

```
types_error_code_e ota_process_write_block (
    const uint8_t * data,
    const size_t data_len )
```

Writes a block of data to an ongoing Over-The-Air (OTA) update process, verifies the integrity of the data using SHA-256 hashing, and checks the firmware size against the expected size. It returns an error code indicating the status of the operation, such as success, failure, or in-progress, and handles errors like mismatched firmware size or hash computation failures.

Parameters

<i>data</i>	Firmware block
<i>data_len</i>	Firmware block size

Returns

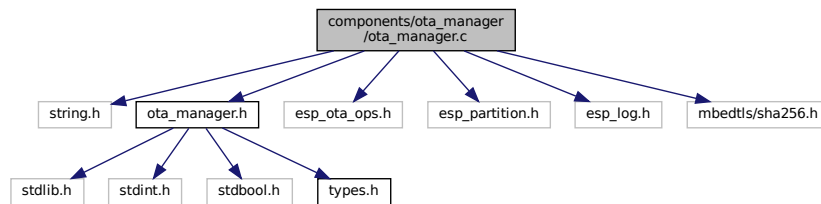
types_error_code_e

4.6 components/ota_manager/ota_manager.c File Reference

```
#include <string.h>
#include "ota_manager.h"
#include "esp_ota_ops.h"
#include "esp_partition.h"
#include "esp_log.h"
```

```
#include "mbedtls/sha256.h"
```

Include dependency graph for ota_manager.c:



Macros

- `#define HASH_SIZE_IN_BYTES (32U)`

Functions

- `types_error_code_e ota_process_init` (const size_t img_size, const uint8_t *hash)
Initializes an Over-The-Air (OTA) update process by setting the firmware size, copying the hash, selecting the next OTA partition, and preparing the partition for writing. It also initializes a SHA-256 context for hash computation and returns an appropriate error code based on the success or failure of the initialization steps.
- `types_error_code_e ota_process_write_block` (const uint8_t *data, const size_t data_len)
Writes a block of data to an ongoing Over-The-Air (OTA) update process, verifies the integrity of the data using SHA-256 hashing, and checks the firmware size against the expected size. It returns an error code indicating the status of the operation, such as success, failure, or in-progress, and handles errors like mismatched firmware size or hash computation failures.
- `types_error_code_e ota_process_end` (bool is_healthy)
Concludes an ongoing OTA (Over-The-Air) update process, ensuring proper cleanup and validation. It checks if the system is healthy, frees allocated resources, finalizes the OTA update, sets the new boot partition, and returns an appropriate error code based on the operation's success or failure.
- void `ota_check_rollback` (bool is_healthy)
Evaluates the health of the system and manages OTA rollback behavior based on the firmware state. If the system is unhealthy or the firmware verification fails, it triggers a rollback and reboot; otherwise, it marks the firmware as valid and cancels the rollback.

4.6.1 Macro Definition Documentation

4.6.1.1 HASH_SIZE_IN_BYTES

```
#define HASH_SIZE_IN_BYTES (32U)
```

4.6.2 Function Documentation

4.6.2.1 ota_check_rollback()

```
void ota_check_rollback (
    bool is_healthy )
```

Evaluates the health of the system and manages OTA rollback behavior based on the firmware state. If the system is unhealthy or the firmware verification fails, it triggers a rollback and reboot; otherwise, it marks the firmware as valid and cancels the rollback.

Parameters

<i>is_healthy</i>	true when system is healthy, false otherwise
-------------------	--

4.6.2.2 ota_process_end()

```
types_error_code_e ota_process_end (
    bool is_healthy )
```

Concludes an ongoing OTA (Over-The-Air) update process, ensuring proper cleanup and validation. It checks if the system is healthy, frees allocated resources, finalizes the OTA update, sets the new boot partition, and returns an appropriate error code based on the operation's success or failure.

Parameters

<i>is_healthy</i>	true when writing process was successful
-------------------	--

Returns

types_error_code_e

4.6.2.3 ota_process_init()

```
types_error_code_e ota_process_init (
    const size_t img_size,
    const uint8_t * hash )
```

Initializes an Over-The-Air (OTA) update process by setting the firmware size, copying the hash, selecting the next OTA partition, and preparing the partition for writing. It also initializes a SHA-256 context for hash computation and returns an appropriate error code based on the success or failure of the initialization steps.

Parameters

<i>img_size</i>	Firmware size to be updated
<i>hash</i>	Received hash

Returns

types_error_code_e

4.6.2.4 ota_process_write_block()

```
types_error_code_e ota_process_write_block (
    const uint8_t * data,
    const size_t data_len )
```

Writes a block of data to an ongoing Over-The-Air (OTA) update process, verifies the integrity of the data using SHA-256 hashing, and checks the firmware size against the expected size. It returns an error code indicating the status of the operation, such as success, failure, or in-progress, and handles errors like mismatched firmware size or hash computation failures.

Parameters

<i>data</i>	Firmware block
<i>data_len</i>	Firmware block size

Returns

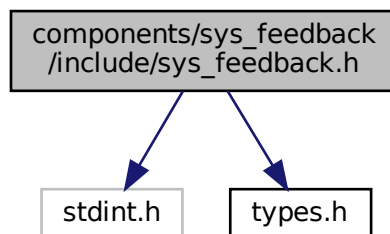
types_error_code_e

4.7 components/sys_feedback/include/sys_feedback.h File Reference

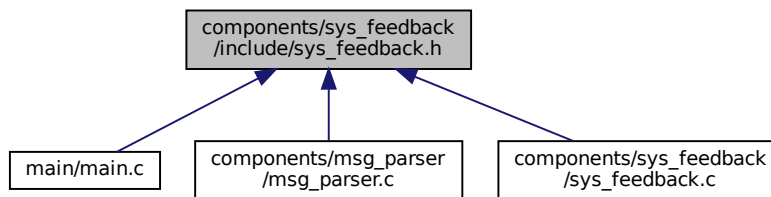
```
#include <stdint.h>
```

```
#include "types.h"
```

Include dependency graph for sys_feedback.h:



This graph shows which files directly or indirectly include this file:



Enumerations

- enum `sys_feedback_mode_t` { `SYS_FEEDBACK_MODE_UPDATE` , `SYS_FEEDBACK_MODE_NORMAL` }
Sys_feedback modes.

Functions

- `types_error_code_e sys_feedback_init` (void)
Initialize sys_feedback component.
- void `sys_feedback_set_update_mode` (void)
Update sys_feedback to update mode.
- void `sys_feedback_set_normal_mode` (void)
Update sys_feedback to normal mode.
- void `sys_feedback_whoiam` (const uint8_t major, const uint8_t minor, const uint8_t patch)
Print information about the firmware.

4.7.1 Enumeration Type Documentation

4.7.1.1 sys_feedback_mode_t

```
enum sys_feedback_mode_t
```

`Sys_feedback` modes.

Values used to indicate the system mode

Enumerator

<code>SYS_FEEDBACK_MODE_UPDATE</code>	
<code>SYS_FEEDBACK_MODE_NORMAL</code>	

4.7.2 Function Documentation

4.7.2.1 sys_feedback_init()

```
types_error_code_e sys_feedback_init (  
    void )
```

Initialize sys_feedback component.

Returns

types_error_code_e

4.7.2.2 sys_feedback_set_normal_mode()

```
void sys_feedback_set_normal_mode (  
    void )
```

Update sys_feedback to normal mode.

4.7.2.3 sys_feedback_set_update_mode()

```
void sys_feedback_set_update_mode (  
    void )
```

Update sys_feedback to update mode.

4.7.2.4 sys_feedback_whoiam()

```
void sys_feedback_whoiam (  
    const uint8_t major,  
    const uint8_t minor,  
    const uint8_t patch )
```

Print information about the firmware.

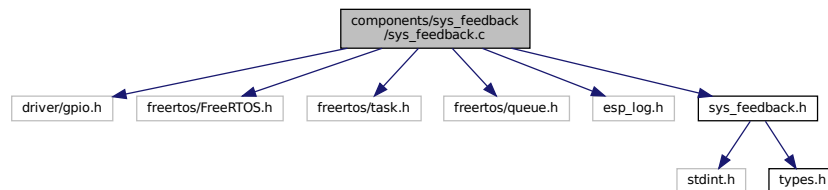
Parameters

<i>major</i>	[in]: Major version
<i>minor</i>	[in]: Minor version
<i>patch</i>	[in]: Patch version

4.8 components/sys_feedback/sys_feedback.c File Reference

```
#include "driver/gpio.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "esp_log.h"
#include "sys_feedback.h"
```

Include dependency graph for sys_feedback.c:



Macros

- `#define FEEDBACK_GPIO` (GPIO_NUM_2)
- `#define PINNED_CORE` (1)
- `#define FEEDBACK_QUEUE_LEN` (5)
- `#define DELAY_FEEDBACK_TASK_MS` (100U)

Functions

- `types_error_code_e sys_feedback_init` (void)
Initialize sys_feedback component.
- `void sys_feedback_set_update_mode` (void)
Update sys_feedback to update mode.
- `void sys_feedback_set_normal_mode` (void)
Update sys_feedback to normal mode.
- `void sys_feedback_whoiam` (const uint8_t major, const uint8_t minor, const uint8_t patch)
Print information about the firmware.

4.8.1 Macro Definition Documentation

4.8.1.1 DELAY_FEEDBACK_TASK_MS

```
#define DELAY_FEEDBACK_TASK_MS (100U)
```

4.8.1.2 FEEDBACK_GPIO

```
#define FEEDBACK_GPIO (GPIO_NUM_2)
```

4.8.1.3 FEEDBACK_QUEUE_LEN

```
#define FEEDBACK_QUEUE_LEN (5)
```

4.8.1.4 PINNED_CORE

```
#define PINNED_CORE (1)
```

4.8.2 Function Documentation

4.8.2.1 sys_feedback_init()

```
types_error_code_e sys_feedback_init (  
    void )
```

Initialize sys_feedback component.

Returns

types_error_code_e

4.8.2.2 sys_feedback_set_normal_mode()

```
void sys_feedback_set_normal_mode (  
    void )
```

Update sys_feedback to normal mode.

4.8.2.3 sys_feedback_set_update_mode()

```
void sys_feedback_set_update_mode (  
    void )
```

Update sys_feedback to update mode.

4.8.2.4 sys_feedback_whoiam()

```
void sys_feedback_whoiam (  
    const uint8_t major,  
    const uint8_t minor,  
    const uint8_t patch )
```

Print information about the firmware.

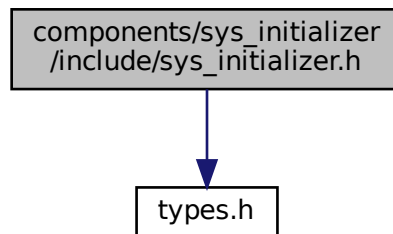
Parameters

<i>major</i>	[in]: Major version
<i>minor</i>	[in]: Minor version
<i>patch</i>	[in]: Patch version

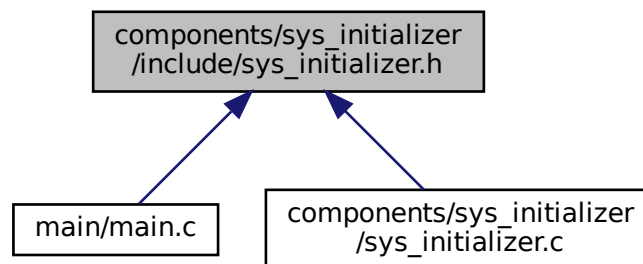
4.9 components/sys_initializer/include/sys_initializer.h File Reference

```
#include "types.h"
```

Include dependency graph for sys_initializer.h:



This graph shows which files directly or indirectly include this file:



Functions

- [types_error_code_e sys_initializer_init](#) (void)
Initialize the sys_initializer component.

4.9.1 Function Documentation

4.9.1.1 sys_initializer_init()

```
types_error_code_e sys_initializer_init (
    void )
```

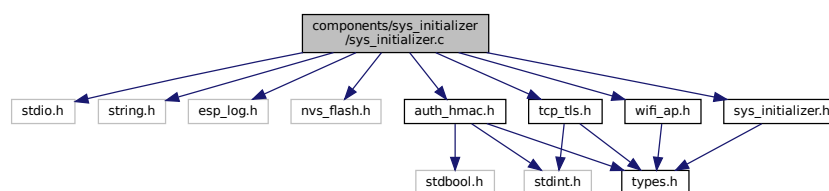
Initialize the sys_initializer component.

Returns

types_error_code_e

4.10 components/sys_initializer/sys_initializer.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "esp_log.h"
#include "nvs_flash.h"
#include "tcp_tls.h"
#include "wifi_ap.h"
#include "auth_hmac.h"
#include "sys_initializer.h"
Include dependency graph for sys_initializer.c:
```



Functions

- `types_error_code_e sys_initializer_init (void)`
Initialize the sys_initializer component.

4.10.1 Function Documentation

4.10.1.1 sys_initializer_init()

```
types_error_code_e sys_initializer_init (
    void )
```

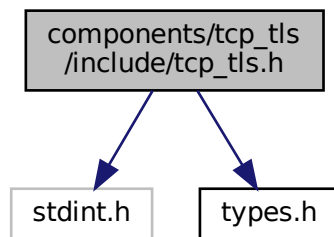
Initialize the sys_initializer component.

Returns

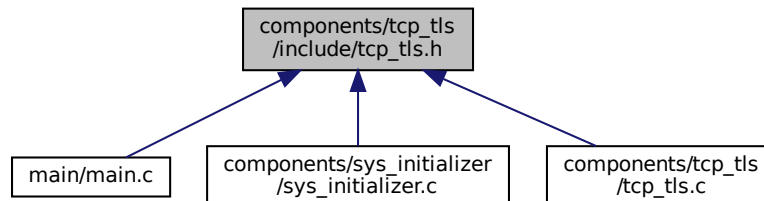
types_error_code_e

4.11 components/tcp_tls/include/tcp_tls.h File Reference

```
#include <stdint.h>
#include "types.h"
Include dependency graph for tcp_tls.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- `#define TCP_TLS_MAX_BUFFER_LEN (4198U)`

Functions

- [types_error_code_e tcp_tls_init](#) (void)
Initialize the tcp_tls component.
- [types_error_code_e tcp_tls_set_server_cert](#) (const uint8_t *crt, const size_t len)
Server_cert setter.
- [types_error_code_e tcp_tls_set_server_key](#) (const uint8_t *key, const size_t len)
Server_key setter.

4.11.1 Macro Definition Documentation

4.11.1.1 TCP_TLS_MAX_BUFFER_LEN

```
#define TCP_TLS_MAX_BUFFER_LEN (4198U)
```

4.11.2 Function Documentation

4.11.2.1 tcp_tls_init()

```
types_error_code_e tcp_tls_init (  
    void )
```

Initialize the tcp_tls component.

Returns

types_error_code_e

4.11.2.2 tcp_tls_set_server_cert()

```
types_error_code_e tcp_tls_set_server_cert (  
    const uint8_t * crt,  
    const size_t len )
```

Server_cert setter.

Parameters

<i>crt</i>	[in]: Server certificate
<i>len</i>	[in]: Server certificate length in bytes

Returns

types_error_code_e

4.11.2.3 tcp_tls_set_server_key()

```
types_error_code_e tcp_tls_set_server_key (
    const uint8_t * key,
    const size_t len )
```

Server_key setter.

Parameters

<i>key</i>	[in]: Server key
<i>len</i>	[in]: Server key length in bytes

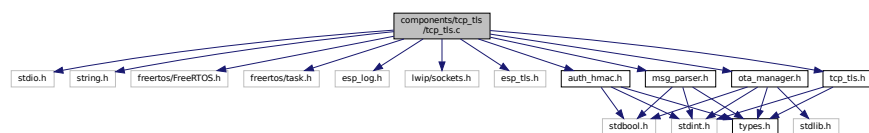
Returns

types_error_code_e

4.12 components/tcp_tls/tcp_tls.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_log.h"
#include "lwip/sockets.h"
#include "esp_tls.h"
#include "msg_parser.h"
#include "auth_hmac.h"
#include "ota_manager.h"
#include "tcp_tls.h"
```

Include dependency graph for tcp_tls.c:

**Classes**

- struct [crypt_buffer_t](#)

Macros

- `#define PINNED_CORE (1)`
- `#define COUNT_NEEDED_TO_START_TCP_SOCKET (2U)`
- `#define TCP_BUFFER_LEN_BYTES (2048U)`
- `#define KEEPIDLE_TIME_SEC (30)`
- `#define KEEPINTERVAL_SEC (5)`
- `#define KEEPCOUNT (2)`
- `#define RX_TIMEOUT_SEC (0)`
- `#define RX_TIMEOUT_USEC (500000)`
- `#define DELAY_AFTER_UPDATE_MS (200)`

Functions

- `types_error_code_e tcp_tls_init (void)`
Initialize the tcp_tls component.
- `types_error_code_e tcp_tls_set_server_cert (const uint8_t *cert, const size_t len)`
Server_cert setter.
- `types_error_code_e tcp_tls_set_server_key (const uint8_t *key, const size_t len)`
Server_key setter.

4.12.1 Macro Definition Documentation

4.12.1.1 COUNT_NEEDED_TO_START_TCP_SOCKET

```
#define COUNT_NEEDED_TO_START_TCP_SOCKET (2U)
```

4.12.1.2 DELAY_AFTER_UPDATE_MS

```
#define DELAY_AFTER_UPDATE_MS (200)
```

4.12.1.3 KEEPCOUNT

```
#define KEEPCOUNT (2)
```

4.12.1.4 KEEPIDLE_TIME_SEC

```
#define KEEPIDLE_TIME_SEC (30)
```

4.12.1.5 KEEPINTERVAL_SEC

```
#define KEEPINTERVAL_SEC (5)
```

4.12.1.6 PINNED_CORE

```
#define PINNED_CORE (1)
```

4.12.1.7 RX_TIMEOUT_SEC

```
#define RX_TIMEOUT_SEC (0)
```

4.12.1.8 RX_TIMEOUT_USEC

```
#define RX_TIMEOUT_USEC (500000)
```

4.12.1.9 TCP_BUFFER_LEN_BYTES

```
#define TCP_BUFFER_LEN_BYTES (2048U)
```

4.12.2 Function Documentation

4.12.2.1 tcp_tls_init()

```
types_error_code_e tcp_tls_init (  
    void )
```

Initialize the tcp_tls component.

Returns

types_error_code_e

4.12.2.2 tcp_tls_set_server_cert()

```
types_error_code_e tcp_tls_set_server_cert (  
    const uint8_t * crt,  
    const size_t len )
```

Server_cert setter.

Parameters

<i>crt</i>	[in]: Server certificate
<i>len</i>	[in]: Server certificate length in bytes

Returns

types_error_code_e

4.12.2.3 tcp_tls_set_server_key()

```
types_error_code_e tcp_tls_set_server_key (
    const uint8_t * key,
    const size_t len )
```

Server_key setter.

Parameters

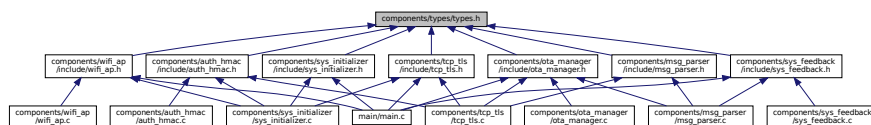
<i>key</i>	[in]: Server key
<i>len</i>	[in]: Server key length in bytes

Returns

types_error_code_e

4.13 components/types/types.h File Reference

This graph shows which files directly or indirectly include this file:



Enumerations

- enum `types_error_code_e` {
`ERR_CODE_OK` , `ERR_CODE_IN_PROGRESS` , `ERR_CODE_FAIL` , `ERR_CODE_INVALID_PARAM` ,
`ERR_CODE_INVALID_OP` , `ERR_CODE_NOT_ALLOWED` }

Error code types.

4.13.1 Enumeration Type Documentation

4.13.1.1 `types_error_code_e`

enum `types_error_code_e`

Error code types.

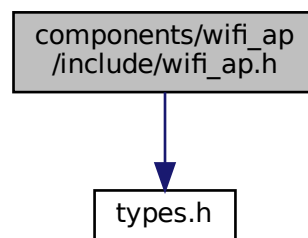
Enumerator

ERR_CODE_OK	
ERR_CODE_IN_PROGRESS	
ERR_CODE_FAIL	
ERR_CODE_INVALID_PARAM	
ERR_CODE_INVALID_OP	
ERR_CODE_NOT_ALLOWED	

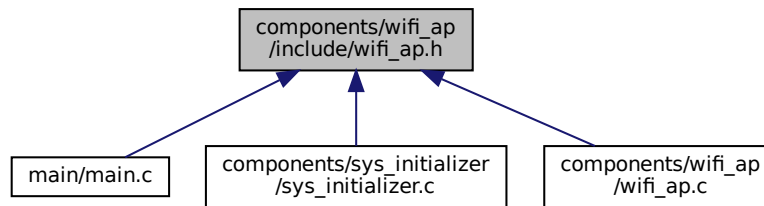
4.14 `components/wifi_ap/include/wifi_ap.h` File Reference

```
#include "types.h"
```

Include dependency graph for `wifi_ap.h`:



This graph shows which files directly or indirectly include this file:



Macros

- `#define WIFI_AP_SSID_MAX_LEN (32U)`
- `#define WIFI_AP_PASSWORD_MAX_LEN (64U)`

Functions

- `void wifi_ap_init (void)`
Initialize wifi_ap component.
- `types_error_code_e wifi_ap_set_ssid (char *ssid, const uint8_t len)`
SSID setter.
- `types_error_code_e wifi_ap_set_password (char *password, const uint8_t len)`
Password setter.

4.14.1 Macro Definition Documentation

4.14.1.1 WIFI_AP_PASSWORD_MAX_LEN

```
#define WIFI_AP_PASSWORD_MAX_LEN (64U)
```

4.14.1.2 WIFI_AP_SSID_MAX_LEN

```
#define WIFI_AP_SSID_MAX_LEN (32U)
```

4.14.2 Function Documentation

4.14.2.1 wifi_ap_init()

```
void wifi_ap_init (
    void )
```

Initialize wifi_ap component.

4.14.2.2 wifi_ap_set_password()

```
types_error_code_e wifi_ap_set_password (
    char * password,
    const uint8_t len )
```

Password setter.

Parameters

<i>password</i>	[in]: Wi-Fi AP password
<i>len</i>	[in]: Password length

4.14.2.3 wifi_ap_set_ssid()

```
types_error_code_e wifi_ap_set_ssid (
    char * ssid,
    const uint8_t len )
```

SSID setter.

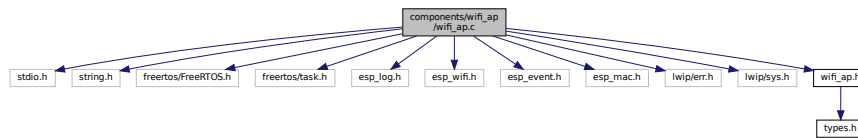
Parameters

<i>ssid</i>	[in]: Wi-Fi AP SSID
<i>len</i>	[in]: SSID length

4.15 components/wifi_ap/wifi_ap.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_log.h"
#include "esp_wifi.h"
#include "esp_event.h"
#include "esp_mac.h"
#include "lwip/err.h"
```

```
#include "lwip/sys.h"
#include "wifi_ap.h"
Include dependency graph for wifi_ap.c:
```



Macros

- `#define PASSWORD_MIN_LEN (8U) /* Min length of Wi-Fi API. Don't choose less than 8 bytes */`
- `#define MAX_CLIENTS (1U)`
- `#define WIFI_CHANNEL (1U)`

Functions

- `void wifi_ap_init (void)`
Initialize wifi_ap component.
- `types_error_code_e wifi_ap_set_ssid (char *ssid, const uint8_t len)`
SSID setter.
- `types_error_code_e wifi_ap_set_password (char *password, const uint8_t len)`
Password setter.

4.15.1 Macro Definition Documentation

4.15.1.1 MAX_CLIENTS

```
#define MAX_CLIENTS (1U)
```

4.15.1.2 PASSWORD_MIN_LEN

```
#define PASSWORD_MIN_LEN (8U) /* Min length of Wi-Fi API. Don't choose less than 8 bytes */
```

4.15.1.3 WIFI_CHANNEL

```
#define WIFI_CHANNEL (1U)
```

4.15.2 Function Documentation

4.15.2.1 wifi_ap_init()

```
void wifi_ap_init (
    void )
```

Initialize wifi_ap component.

4.15.2.2 wifi_ap_set_password()

```
types_error_code_e wifi_ap_set_password (
    char * password,
    const uint8_t len )
```

Password setter.

Parameters

<i>password</i>	[in]: Wi-Fi AP password
<i>len</i>	[in]: Password length

4.15.2.3 wifi_ap_set_ssid()

```
types_error_code_e wifi_ap_set_ssid (
    char * ssid,
    const uint8_t len )
```

SSID setter.

Parameters

<i>ssid</i>	[in]: Wi-Fi AP SSID
<i>len</i>	[in]: SSID length

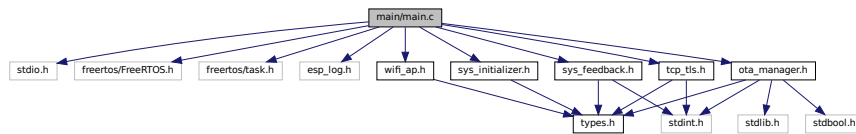
4.16 main/main.c File Reference

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
```



```
#include "esp_log.h"
#include "sys_initializer.h"
#include "wifi_ap.h"
#include "tcp_tls.h"
#include "ota_manager.h"
#include "sys_feedback.h"
```

Include dependency graph for main.c:



Macros

- #define [VERSION_MAJOR](#) (1U)
- #define [VERSION_MINOR](#) (0U)
- #define [VERSION_PATCH](#) (0U)
- #define [RESET_DELAY_MS](#) (1000)

Functions

- void [app_main](#) (void)
Main task.

4.16.1 Macro Definition Documentation

4.16.1.1 RESET_DELAY_MS

```
#define RESET_DELAY_MS (1000)
```

4.16.1.2 VERSION_MAJOR

```
#define VERSION_MAJOR (1U)
```

4.16.1.3 VERSION_MINOR

```
#define VERSION_MINOR (0U)
```

4.16.1.4 VERSION_PATCH

```
#define VERSION_PATCH (0U)
```

4.16.2 Function Documentation

4.16.2.1 app_main()

```
void app_main (  
    void )
```

Main task.

Index

app_main
 main.c, [44](#)

auth_hmac.c
 auth_hmac_generate_nonce, [8](#)
 auth_hmac_set_hmac_psk, [8](#)
 auth_hmac_verify_response, [8](#)
 HMAC_SHA256_LEN, [8](#)
 is_set, [9](#)
 len, [9](#)
 val, [9](#)

auth_hmac.h
 auth_hmac_generate_nonce, [11](#)
 AUTH_HMAC_MAX_BUFFER_LEN, [11](#)
 AUTH_HMAC_NONCE_LEN, [11](#)
 auth_hmac_set_hmac_psk, [11](#)
 auth_hmac_verify_response, [12](#)

auth_hmac_generate_nonce
 auth_hmac.c, [8](#)
 auth_hmac.h, [11](#)

AUTH_HMAC_MAX_BUFFER_LEN
 auth_hmac.h, [11](#)

AUTH_HMAC_NONCE_LEN
 auth_hmac.h, [11](#)

auth_hmac_set_hmac_psk
 auth_hmac.c, [8](#)
 auth_hmac.h, [11](#)

auth_hmac_verify_response
 auth_hmac.c, [8](#)
 auth_hmac.h, [12](#)

components/auth_hmac/auth_hmac.c, [7](#)
components/auth_hmac/include/auth_hmac.h, [10](#)
components/msg_parser/include/msg_parser.h, [12](#)
components/msg_parser/msg_parser.c, [15](#)
components/ota_manager/include/ota_manager.h, [20](#)
components/ota_manager/ota_manager.c, [22](#)
components/sys_feedback/include/sys_feedback.h, [25](#)
components/sys_feedback/sys_feedback.c, [28](#)
components/sys_initializer/include/sys_initializer.h, [30](#)
components/sys_initializer/sys_initializer.c, [31](#)
components/tcp_tls/include/tcp_tls.h, [32](#)
components/tcp_tls/tcp_tls.c, [34](#)
components/types/types.h, [37](#)
components/wifi_ap/include/wifi_ap.h, [38](#)
components/wifi_ap/wifi_ap.c, [40](#)
COUNT_NEEDED_TO_START_TCP_SOCKET
 tcp_tls.c, [35](#)
crypt_buffer_t, [5](#)
 len, [5](#)
 val, [5](#)

DELAY_AFTER_UPDATE_MS
 tcp_tls.c, [35](#)

DELAY_FEEDBACK_TASK_MS
 sys_feedback.c, [28](#)

ERR_CODE_FAIL
 types.h, [38](#)

ERR_CODE_IN_PROGRESS
 types.h, [38](#)

ERR_CODE_INVALID_OP
 types.h, [38](#)

ERR_CODE_INVALID_PARAM
 types.h, [38](#)

ERR_CODE_NOT_ALLOWED
 types.h, [38](#)

ERR_CODE_OK
 types.h, [38](#)

FEEDBACK_GPIO
 sys_feedback.c, [28](#)

FEEDBACK_QUEUE_LEN
 sys_feedback.c, [29](#)

FIRMWARE_ACK_SIZE_IN_BYTES
 msg_parser.c, [16](#)

firmware_bytes_read
 state_machine_params_t, [6](#)

FIRMWARE_LEN_SIZE_IN_BYTES
 msg_parser.c, [16](#)

firmware_size
 state_machine_params_t, [6](#)

hash
 state_machine_params_t, [6](#)

HASH_SIZE_IN_BYTES
 msg_parser.c, [16](#)
 ota_manager.c, [23](#)

HEADER_SIZE_IN_BYTES
 msg_parser.c, [17](#)

HMAC_SHA256_LEN
 auth_hmac.c, [8](#)

is_set
 auth_hmac.c, [9](#)

KEEPCOUNT
 tcp_tls.c, [35](#)

KEEPIDLE_TIME_SEC
 tcp_tls.c, [35](#)

KEEPINTERVAL_SEC
 tcp_tls.c, [35](#)

- len
 - auth_hmac.c, [9](#)
 - crypt_buffer_t, [5](#)
- main.c
 - app_main, [44](#)
 - RESET_DELAY_MS, [43](#)
 - VERSION_MAJOR, [43](#)
 - VERSION_MINOR, [43](#)
 - VERSION_PATCH, [43](#)
- main/main.c, [42](#)
- MAX_CLIENTS
 - wifi_ap.c, [41](#)
- msg_parser.c
 - FIRMWARE_ACK_SIZE_IN_BYTES, [16](#)
 - FIRMWARE_LEN_SIZE_IN_BYTES, [16](#)
 - HASH_SIZE_IN_BYTES, [16](#)
 - HEADER_SIZE_IN_BYTES, [17](#)
 - msg_parser_build_firmware_ack, [18](#)
 - msg_parser_build_ota_ack, [18](#)
 - msg_parser_clean, [19](#)
 - msg_parser_init, [19](#)
 - msg_parser_run, [19](#)
 - msg_parser_states_e, [17](#)
 - OTA_ACK_ERR_SIZE_IN_BYTE, [17](#)
 - OTA_ACK_FAIL_CODE, [17](#)
 - OTA_ACK_LEN_SIZE_IN_BYTES, [17](#)
 - OTA_ACK_OK_CODE, [17](#)
 - OTA_ACK_SIZE_IN_BYTES, [17](#)
 - READ_HEADER, [18](#)
 - START_OTA, [18](#)
 - WRITE_FIRMWARE, [18](#)
- msg_parser.h
 - MSG_PARSER_BUF_LEN_BYTES, [13](#)
 - msg_parser_build_firmware_ack, [13](#)
 - msg_parser_build_ota_ack, [14](#)
 - msg_parser_clean, [14](#)
 - msg_parser_init, [14](#)
 - msg_parser_run, [15](#)
- MSG_PARSER_BUF_LEN_BYTES
 - msg_parser.h, [13](#)
- msg_parser_build_firmware_ack
 - msg_parser.c, [18](#)
 - msg_parser.h, [13](#)
- msg_parser_build_ota_ack
 - msg_parser.c, [18](#)
 - msg_parser.h, [14](#)
- msg_parser_clean
 - msg_parser.c, [19](#)
 - msg_parser.h, [14](#)
- msg_parser_init
 - msg_parser.c, [19](#)
 - msg_parser.h, [14](#)
- msg_parser_run
 - msg_parser.c, [19](#)
 - msg_parser.h, [15](#)
- msg_parser_states_e
 - msg_parser.c, [17](#)
- OTA_ACK_ERR_SIZE_IN_BYTE
 - msg_parser.c, [17](#)
- OTA_ACK_FAIL_CODE
 - msg_parser.c, [17](#)
- OTA_ACK_LEN_SIZE_IN_BYTES
 - msg_parser.c, [17](#)
- OTA_ACK_OK_CODE
 - msg_parser.c, [17](#)
- OTA_ACK_SIZE_IN_BYTES
 - msg_parser.c, [17](#)
- ota_check_rollback
 - ota_manager.c, [23](#)
 - ota_manager.h, [21](#)
- ota_manager.c
 - HASH_SIZE_IN_BYTES, [23](#)
 - ota_check_rollback, [23](#)
 - ota_process_end, [24](#)
 - ota_process_init, [24](#)
 - ota_process_write_block, [25](#)
- ota_manager.h
 - ota_check_rollback, [21](#)
 - ota_process_end, [21](#)
 - ota_process_init, [21](#)
 - ota_process_write_block, [22](#)
- ota_process_end
 - ota_manager.c, [24](#)
 - ota_manager.h, [21](#)
- ota_process_init
 - ota_manager.c, [24](#)
 - ota_manager.h, [21](#)
- ota_process_write_block
 - ota_manager.c, [25](#)
 - ota_manager.h, [22](#)
- PASSWORD_MIN_LEN
 - wifi_ap.c, [41](#)
- PINNED_CORE
 - sys_feedback.c, [29](#)
 - tcp_tls.c, [36](#)
- READ_HEADER
 - msg_parser.c, [18](#)
- RESET_DELAY_MS
 - main.c, [43](#)
- RX_TIMEOUT_SEC
 - tcp_tls.c, [36](#)
- RX_TIMEOUT_USEC
 - tcp_tls.c, [36](#)
- semaphore
 - state_machine_params_t, [6](#)
- START_OTA
 - msg_parser.c, [18](#)
- state
 - state_machine_params_t, [6](#)
- state_machine_params_t, [5](#)
 - firmware_bytes_read, [6](#)
 - firmware_size, [6](#)
 - hash, [6](#)

- semaphore, 6
- state, 6
- sys_feedback.c
 - DELAY_FEEDBACK_TASK_MS, 28
 - FEEDBACK_GPIO, 28
 - FEEDBACK_QUEUE_LEN, 29
 - PINNED_CORE, 29
 - sys_feedback_init, 29
 - sys_feedback_set_normal_mode, 29
 - sys_feedback_set_update_mode, 29
 - sys_feedback_whoiam, 29
- sys_feedback.h
 - sys_feedback_init, 27
 - SYS_FEEDBACK_MODE_NORMAL, 26
 - sys_feedback_mode_t, 26
 - SYS_FEEDBACK_MODE_UPDATE, 26
 - sys_feedback_set_normal_mode, 27
 - sys_feedback_set_update_mode, 27
 - sys_feedback_whoiam, 27
- sys_feedback_init
 - sys_feedback.c, 29
 - sys_feedback.h, 27
- SYS_FEEDBACK_MODE_NORMAL
 - sys_feedback.h, 26
- sys_feedback_mode_t
 - sys_feedback.h, 26
- SYS_FEEDBACK_MODE_UPDATE
 - sys_feedback.h, 26
- sys_feedback_set_normal_mode
 - sys_feedback.c, 29
 - sys_feedback.h, 27
- sys_feedback_set_update_mode
 - sys_feedback.c, 29
 - sys_feedback.h, 27
- sys_feedback_whoiam
 - sys_feedback.c, 29
 - sys_feedback.h, 27
- sys_initializer.c
 - sys_initializer_init, 31
- sys_initializer.h
 - sys_initializer_init, 31
- sys_initializer_init
 - sys_initializer.c, 31
 - sys_initializer.h, 31
- TCP_BUFFER_LEN_BYTES
 - tcp_tls.c, 36
- tcp_tls.c
 - COUNT_NEEDED_TO_START_TCP_SOCKET, 35
 - DELAY_AFTER_UPDATE_MS, 35
 - KEEPCOUNT, 35
 - KEEPIDLE_TIME_SEC, 35
 - KEEPINTERVAL_SEC, 35
 - PINNED_CORE, 36
 - RX_TIMEOUT_SEC, 36
 - RX_TIMEOUT_USEC, 36
 - TCP_BUFFER_LEN_BYTES, 36
 - tcp_tls_init, 36
 - tcp_tls_set_server_cert, 36
 - tcp_tls_set_server_key, 37
- tcp_tls.h
 - tcp_tls_init, 33
 - TCP_TLS_MAX_BUFFER_LEN, 33
 - tcp_tls_set_server_cert, 33
 - tcp_tls_set_server_key, 34
- tcp_tls_init
 - tcp_tls.c, 36
 - tcp_tls.h, 33
- TCP_TLS_MAX_BUFFER_LEN
 - tcp_tls.h, 33
- tcp_tls_set_server_cert
 - tcp_tls.c, 36
 - tcp_tls.h, 33
- tcp_tls_set_server_key
 - tcp_tls.c, 37
 - tcp_tls.h, 34
- types.h
 - ERR_CODE_FAIL, 38
 - ERR_CODE_IN_PROGRESS, 38
 - ERR_CODE_INVALID_OP, 38
 - ERR_CODE_INVALID_PARAM, 38
 - ERR_CODE_NOT_ALLOWED, 38
 - ERR_CODE_OK, 38
 - types_error_code_e, 38
- types_error_code_e
 - types.h, 38
- val
 - auth_hmac.c, 9
 - crypt_buffer_t, 5
- VERSION_MAJOR
 - main.c, 43
- VERSION_MINOR
 - main.c, 43
- VERSION_PATCH
 - main.c, 43
- wifi_ap.c
 - MAX_CLIENTS, 41
 - PASSWORD_MIN_LEN, 41
 - wifi_ap_init, 42
 - wifi_ap_set_password, 42
 - wifi_ap_set_ssid, 42
 - WIFI_CHANNEL, 41
- wifi_ap.h
 - wifi_ap_init, 39
 - WIFI_AP_PASSWORD_MAX_LEN, 39
 - wifi_ap_set_password, 40
 - wifi_ap_set_ssid, 40
 - WIFI_AP_SSID_MAX_LEN, 39
- wifi_ap_init
 - wifi_ap.c, 42
 - wifi_ap.h, 39
- WIFI_AP_PASSWORD_MAX_LEN
 - wifi_ap.h, 39
- wifi_ap_set_password
 - wifi_ap.c, 42

- wifi_ap.h, [40](#)
- wifi_ap_set_ssid
 - wifi_ap.c, [42](#)
 - wifi_ap.h, [40](#)
- WIFI_AP_SSID_MAX_LEN
 - wifi_ap.h, [39](#)
- WIFI_CHANNEL
 - wifi_ap.c, [41](#)
- WRITE_FIRMWARE
 - msg_parser.c, [18](#)