

SmartThings Device SDK(STDK)

1.0.10



Generated by Doxygen 1.8.16

1 Overview	2
1.1 Introduction	2
1.2 IoT Core Device Library	2
1.3 Porting Layer	3
1.4 Device Application	4
2 Data Structure Index	5
2.1 Data Structures	5
3 File Index	7
3.1 File List	7
4 Data Structure Documentation	9
4.1 noti_data_raw_t::_quota Struct Reference	9
4.1.1 Field Documentation	10
4.2 noti_data_raw_t::_rate_limit Struct Reference	10
4.2.1 Field Documentation	11
4.3 iot_bsp_fs_handle_t Struct Reference	12
4.3.1 Field Documentation	12
4.4 iot_cap_cmd_data_t Struct Reference	13
4.4.1 Detailed Description	13
4.4.2 Field Documentation	13
4.5 iot_cap_cmd_set Struct Reference	14
4.5.1 Detailed Description	15
4.5.2 Field Documentation	16
4.6 iot_cap_cmd_set_list Struct Reference	16
4.6.1 Detailed Description	18
4.6.2 Field Documentation	18
4.7 iot_cap_evt_data_t Struct Reference	18
4.7.1 Detailed Description	19
4.7.2 Field Documentation	19
4.8 iot_cap_handle Struct Reference	20
4.8.1 Detailed Description	21
4.8.2 Field Documentation	21
4.9 iot_cap_handle_list Struct Reference	22
4.9.1 Detailed Description	22
4.9.2 Field Documentation	22
4.10 iot_cap_unit_t Struct Reference	23
4.10.1 Detailed Description	23
4.10.2 Field Documentation	23
4.11 iot_cap_val_t Struct Reference	24
4.11.1 Detailed Description	25
4.11.2 Field Documentation	25

4.12 iot_cloud_prov_data Struct Reference	26
4.12.1 Detailed Description	26
4.12.2 Field Documentation	26
4.13 iot_command Struct Reference	27
4.13.1 Detailed Description	28
4.13.2 Field Documentation	28
4.14 iot_context Struct Reference	28
4.14.1 Detailed Description	30
4.14.2 Field Documentation	31
4.15 iot_crypto_cipher_info Struct Reference	35
4.15.1 Detailed Description	35
4.15.2 Field Documentation	36
4.16 iot_crypto_ecdh_params Struct Reference	37
4.16.1 Detailed Description	37
4.16.2 Field Documentation	37
4.17 iot_crypto_ed25519_keypair Struct Reference	38
4.17.1 Detailed Description	39
4.17.2 Field Documentation	39
4.18 iot_crypto_keypair Struct Reference	39
4.18.1 Detailed Description	40
4.18.2 Field Documentation	40
4.19 iot_crypto_pk_context Struct Reference	40
4.19.1 Detailed Description	41
4.19.2 Field Documentation	41
4.20 iot_crypto_pk_funcs Struct Reference	41
4.20.1 Detailed Description	42
4.20.2 Field Documentation	42
4.21 iot_crypto_pk_info Struct Reference	43
4.21.1 Detailed Description	43
4.21.2 Field Documentation	43
4.22 iot_devconf_prov_data Struct Reference	44
4.22.1 Detailed Description	45
4.22.2 Field Documentation	45
4.23 iot_device_info Struct Reference	46
4.23.1 Detailed Description	47
4.23.2 Field Documentation	47
4.24 iot_device_prov_data Struct Reference	47
4.24.1 Detailed Description	48
4.24.2 Field Documentation	48
4.25 iot_easysetup_payload Struct Reference	49
4.25.1 Detailed Description	49
4.25.2 Field Documentation	50

4.26 iot_mac Struct Reference	50
4.26.1 Detailed Description	51
4.26.2 Field Documentation	51
4.27 iot_mqtt_ctx Struct Reference	51
4.27.1 Detailed Description	53
4.27.2 Field Documentation	53
4.28 iot_net_connection Struct Reference	54
4.28.1 Detailed Description	55
4.28.2 Field Documentation	55
4.29 iot_net_interface Struct Reference	56
4.29.1 Detailed Description	58
4.29.2 Field Documentation	58
4.30 iot_net_platform_context Struct Reference	59
4.30.1 Detailed Description	60
4.30.2 Field Documentation	60
4.31 iot_noti_data_t Struct Reference	61
4.31.1 Detailed Description	62
4.31.2 Field Documentation	62
4.32 iot_os_mutex Struct Reference	63
4.32.1 Detailed Description	63
4.32.2 Field Documentation	63
4.33 iot_pin_t Struct Reference	64
4.33.1 Detailed Description	64
4.33.2 Field Documentation	64
4.34 iot_registered_data Struct Reference	65
4.34.1 Detailed Description	65
4.34.2 Field Documentation	65
4.35 iot_state_data Struct Reference	66
4.35.1 Detailed Description	66
4.35.2 Field Documentation	66
4.36 iot_uuid Struct Reference	67
4.36.1 Detailed Description	67
4.36.2 Field Documentation	67
4.37 iot_wifi_conf Struct Reference	68
4.37.1 Detailed Description	68
4.37.2 Field Documentation	68
4.38 iot_wifi_prov_data Struct Reference	69
4.38.1 Detailed Description	70
4.38.2 Field Documentation	70
4.39 iot_wifi_scan_result_t Struct Reference	71
4.39.1 Detailed Description	71
4.39.2 Field Documentation	71

4.40 MessageData Struct Reference	72
4.40.1 Field Documentation	73
4.41 MQTTClient::MessageHandlers Struct Reference	74
4.41.1 Field Documentation	74
4.42 MQTTClient Struct Reference	75
4.42.1 Field Documentation	76
4.43 MQTTConnackData Struct Reference	78
4.43.1 Field Documentation	78
4.44 MQTTConnackFlags Union Reference	79
4.44.1 Field Documentation	79
4.45 MQTTConnectFlags Union Reference	80
4.45.1 Field Documentation	80
4.46 MQTTHeader Union Reference	82
4.46.1 Detailed Description	82
4.46.2 Field Documentation	82
4.47 MQTTLenString Struct Reference	83
4.47.1 Field Documentation	83
4.48 MQTTMessage Struct Reference	84
4.48.1 Field Documentation	84
4.49 MQTTPacket_connectData Struct Reference	85
4.49.1 Field Documentation	87
4.50 MQTTPacket_willOptions Struct Reference	88
4.50.1 Detailed Description	89
4.50.2 Field Documentation	89
4.51 MQTTString Struct Reference	90
4.51.1 Field Documentation	90
4.52 MQTTSubackData Struct Reference	91
4.52.1 Field Documentation	91
4.53 MQTTTransport Struct Reference	91
4.53.1 Field Documentation	92
4.54 noti_data_raw_t Union Reference	93
4.54.1 Detailed Description	93
4.54.2 Field Documentation	93
4.55 url_parse_t Struct Reference	94
4.55.1 Detailed Description	94
4.55.2 Field Documentation	94
5 File Documentation	95
5.1 st-device-sdk-c/doc/mainpage.dox File Reference	95
5.2 st-device-sdk-c/src/crypto/iot_crypto_ed25519.c File Reference	95
5.2.1 Function Documentation	96
5.3 st-device-sdk-c/src/crypto/mbedtls/iot_crypto_mbedtls.c File Reference	98

5.3.1 Function Documentation	99
5.4 st-device-sdk-c/src/crypto/ss/iot_crypto_ss.c File Reference	104
5.4.1 Function Documentation	104
5.5 st-device-sdk-c/src/easysetup/http/iot_easysetup_http.c File Reference	106
5.5.1 Macro Definition Documentation	107
5.5.2 Function Documentation	107
5.5.3 Variable Documentation	111
5.6 st-device-sdk-c/src/easysetup/iot_easysetup_crypto.c File Reference	111
5.6.1 Function Documentation	112
5.7 st-device-sdk-c/src/easysetup/iot_easysetup_d2d.c File Reference	113
5.7.1 Macro Definition Documentation	114
5.7.2 Function Documentation	114
5.8 st-device-sdk-c/src/easysetup/iot_easysetup_st_mqtt.c File Reference	116
5.8.1 Function Documentation	117
5.9 st-device-sdk-c/src/include/bsp/iot_bsp_debug.h File Reference	118
5.9.1 Function Documentation	119
5.10 st-device-sdk-c/src/include/bsp/iot_bsp_fs.h File Reference	120
5.10.1 Enumeration Type Documentation	121
5.10.2 Function Documentation	121
5.11 st-device-sdk-c/src/include/bsp/iot_bsp_nv_data.h File Reference	126
5.11.1 Function Documentation	127
5.12 st-device-sdk-c/src/include/bsp/iot_bsp_random.h File Reference	128
5.12.1 Function Documentation	129
5.13 st-device-sdk-c/src/include/bsp/iot_bsp_system.h File Reference	129
5.13.1 Macro Definition Documentation	130
5.13.2 Function Documentation	131
5.14 st-device-sdk-c/src/include/bsp/iot_bsp_wifi.h File Reference	132
5.14.1 Macro Definition Documentation	134
5.14.2 Enumeration Type Documentation	134
5.14.3 Function Documentation	135
5.15 st-device-sdk-c/src/include/iot_capability.h File Reference	137
5.15.1 Typedef Documentation	139
5.15.2 Enumeration Type Documentation	139
5.16 st-device-sdk-c/src/include/iot_crypto.h File Reference	140
5.16.1 Macro Definition Documentation	143
5.16.2 Typedef Documentation	146
5.16.3 Enumeration Type Documentation	146
5.16.4 Function Documentation	147
5.17 st-device-sdk-c/src/include/iot_crypto_internal.h File Reference	157
5.18 st-device-sdk-c/src/include/iot_debug.h File Reference	157
5.18.1 Macro Definition Documentation	158
5.18.2 Enumeration Type Documentation	160

5.18.3 Function Documentation	161
5.19 st-device-sdk-c/src/include/iot_easysetup.h File Reference	161
5.19.1 Macro Definition Documentation	163
5.19.2 Enumeration Type Documentation	166
5.19.3 Function Documentation	167
5.20 st-device-sdk-c/src/include/iot_error.h File Reference	169
5.20.1 Typedef Documentation	170
5.20.2 Enumeration Type Documentation	170
5.21 st-device-sdk-c/src/include/iot_internal.h File Reference	171
5.21.1 Macro Definition Documentation	173
5.21.2 Function Documentation	174
5.22 st-device-sdk-c/src/include/iot_jwt.h File Reference	189
5.22.1 Macro Definition Documentation	190
5.22.2 Function Documentation	190
5.23 st-device-sdk-c/src/include/iot_main.h File Reference	191
5.23.1 Macro Definition Documentation	193
5.23.2 Typedef Documentation	194
5.23.3 Enumeration Type Documentation	194
5.24 st-device-sdk-c/src/include/iot_net.h File Reference	196
5.24.1 Typedef Documentation	197
5.24.2 Function Documentation	197
5.25 st-device-sdk-c/src/include/iot_nv_data.h File Reference	198
5.25.1 Enumeration Type Documentation	199
5.25.2 Function Documentation	200
5.26 st-device-sdk-c/src/include/iot_util.h File Reference	213
5.26.1 Function Documentation	215
5.27 st-device-sdk-c/src/include/iot_uuid.h File Reference	218
5.27.1 Function Documentation	219
5.28 st-device-sdk-c/src/include/mqtt/iot_mqtt_client.h File Reference	220
5.28.1 Macro Definition Documentation	222
5.28.2 Typedef Documentation	223
5.28.3 Enumeration Type Documentation	224
5.28.4 Function Documentation	224
5.29 st-device-sdk-c/src/include/mqtt/iot_mqtt_connect.h File Reference	234
5.29.1 Macro Definition Documentation	235
5.29.2 Enumeration Type Documentation	235
5.29.3 Function Documentation	236
5.30 st-device-sdk-c/src/include/mqtt/iot_mqtt_format.h File Reference	240
5.30.1 Function Documentation	241
5.31 st-device-sdk-c/src/include/mqtt/iot_mqtt_packet.h File Reference	243
5.31.1 Macro Definition Documentation	244
5.31.2 Enumeration Type Documentation	245

5.31.3 Function Documentation	245
5.32 st-device-sdk-c/src/include/mqtt/iot_mqtt_publish.h File Reference	253
5.32.1 Macro Definition Documentation	254
5.32.2 Function Documentation	254
5.33 st-device-sdk-c/src/include/mqtt/iot_mqtt_stacktrace.h File Reference	257
5.33.1 Macro Definition Documentation	258
5.34 st-device-sdk-c/src/include/mqtt/iot_mqtt_subscribe.h File Reference	260
5.34.1 Macro Definition Documentation	260
5.34.2 Function Documentation	260
5.35 st-device-sdk-c/src/include/mqtt/iot_mqtt_unsubscribe.h File Reference	264
5.35.1 Macro Definition Documentation	265
5.35.2 Function Documentation	265
5.36 st-device-sdk-c/src/include/os/iot_os_util.h File Reference	268
5.36.1 Macro Definition Documentation	269
5.36.2 Typedef Documentation	270
5.36.3 Function Documentation	270
5.36.4 Variable Documentation	281
5.37 st-device-sdk-c/src/include/st_dev.h File Reference	282
5.37.1 Macro Definition Documentation	284
5.37.2 Typedef Documentation	284
5.37.3 Enumeration Type Documentation	286
5.37.4 Function Documentation	287
5.38 st-device-sdk-c/src/include/st_dev_version.h File Reference	295
5.38.1 Macro Definition Documentation	295
5.39 st-device-sdk-c/src/iot_api.c File Reference	296
5.39.1 Function Documentation	297
5.40 st-device-sdk-c/src/iot_capability.c File Reference	305
5.40.1 Macro Definition Documentation	306
5.40.2 Function Documentation	306
5.41 st-device-sdk-c/src/iot_crypto.c File Reference	313
5.41.1 Function Documentation	314
5.42 st-device-sdk-c/src/iot_jwt.c File Reference	316
5.42.1 Function Documentation	317
5.43 st-device-sdk-c/src/iot_main.c File Reference	318
5.43.1 Macro Definition Documentation	319
5.43.2 Function Documentation	319
5.44 st-device-sdk-c/src/iot_mqtt.c File Reference	321
5.44.1 Function Documentation	322
5.45 st-device-sdk-c/src/iot_nv_data.c File Reference	327
5.45.1 Macro Definition Documentation	328
5.45.2 Function Documentation	329
5.46 st-device-sdk-c/src/iot_util.c File Reference	344

5.46.1 Function Documentation	345
5.47 st-device-sdk-c/src/iot_uuid.c File Reference	349
5.47.1 Function Documentation	350
5.48 st-device-sdk-c/src/mqtt/client/iot_mqtt_client.c File Reference	351
5.48.1 Function Documentation	352
5.49 st-device-sdk-c/src/mqtt/packet/iot_mqtt_connect_client.c File Reference	365
5.49.1 Function Documentation	365
5.50 st-device-sdk-c/src/mqtt/packet/iot_mqtt_connect_server.c File Reference	370
5.50.1 Macro Definition Documentation	370
5.50.2 Function Documentation	370
5.51 st-device-sdk-c/src/mqtt/packet/iot_mqtt_deserialize_publish.c File Reference	372
5.51.1 Macro Definition Documentation	372
5.51.2 Function Documentation	372
5.52 st-device-sdk-c/src/mqtt/packet/iot_mqtt_format.c File Reference	373
5.52.1 Function Documentation	374
5.52.2 Variable Documentation	376
5.53 st-device-sdk-c/src/mqtt/packet/iot_mqtt_packet.c File Reference	376
5.53.1 Macro Definition Documentation	377
5.53.2 Function Documentation	377
5.54 st-device-sdk-c/src/mqtt/packet/iot_mqtt_serialize_publish.c File Reference	384
5.54.1 Function Documentation	385
5.55 st-device-sdk-c/src/mqtt/packet/iot_mqtt_subscribe_client.c File Reference	389
5.55.1 Function Documentation	389
5.56 st-device-sdk-c/src/mqtt/packet/iot_mqtt_subscribe_server.c File Reference	392
5.56.1 Function Documentation	392
5.57 st-device-sdk-c/src/mqtt/packet/iot_mqtt_unsubscribe_client.c File Reference	394
5.57.1 Function Documentation	394
5.58 st-device-sdk-c/src/mqtt/packet/iot_mqtt_unsubscribe_server.c File Reference	396
5.58.1 Function Documentation	397
5.59 st-device-sdk-c/src/port/net/mbedtls/iot_net_mbedtls.c File Reference	398
5.59.1 Function Documentation	399
5.60 st-device-sdk-c/src/port/net/mbedtls/iot_net_platform.h File Reference	399
5.60.1 Typedef Documentation	400
5.61 st-device-sdk-c/src/port/net/openssl/iot_net_platform.h File Reference	400
5.61.1 Typedef Documentation	401
5.62 st-device-sdk-c/src/port/net/openssl/iot_net_openssl.c File Reference	401
5.62.1 Function Documentation	401

1 Overview

1.1 Introduction

The SmartThings Device SDK(for short STDK) aims to make it easier to develop IoT devices that can be securely connected to the SmartThings Cloud by providing the IoT Core Device Library and the Reference git repositories. That is, the IoT Core Device Library can be adopted to the existing original chipset vendor's SDK easily or the existing original chipset vendor's SDK can also be adopted to the Reference git.

For this use scenarios, the IoT Core Device library and the Reference was separated into two git repositories as follows:

- [IoT Core Device Library](#)
- [Reference](#)

You can choose to only download the Reference repository from git, if you use a chipset that has already been ported. In this case, the IoT Core Device Library can be easily downloaded as a submodule in the Reference through the predefined setup.sh script.

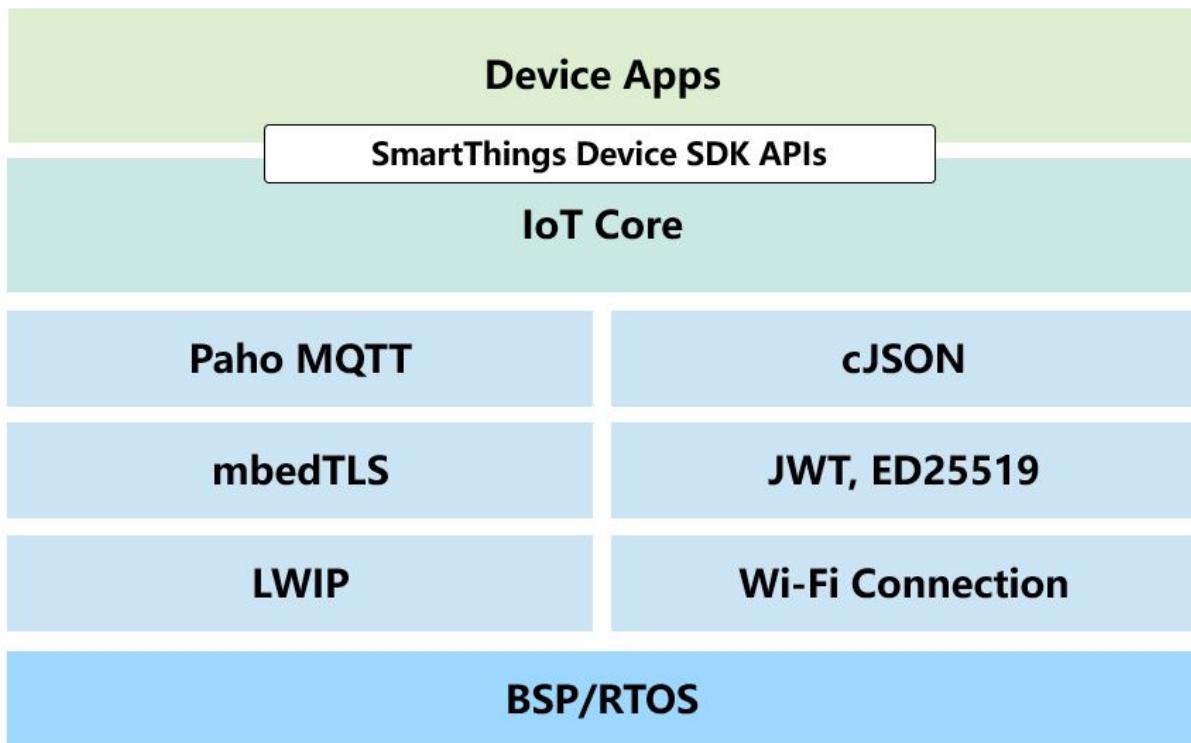


Figure 1 Architecture of SmartThings Device SDK

1.2 IoT Core Device Library

This library provides APIs to be used in device applications.

These APIs can be classified as follows to facilitate the basic behavior scenario of IoT devices and these have been implemented to make a device application as simple as possible.

For more detail, please click the link of each API.

- connection management
 - `st_conn_init()`
 - `st_conn_set_noti_cb()`
 - `st_conn_start()`
 - `st_conn_cleanup()`
 - `st_conn_ownership_confirm()`
- capability management
 - `st_cap_handle_init()`
 - `st_cap_cmd_set_cb()`
 - `st_cap_attr_create_int()`
 - `st_cap_attr_create_number()`
 - `st_cap_attr_create_string()`
 - `st_cap_attr_create_string_array()`
 - `st_cap_attr_free()`
 - `st_cap_attr_send()`

1.3 Porting Layer

The IoT core device library has a porting layer to support the use of the same APIs in the device application, even if the platform changes.

In the current STDK version, the platform-dependent directories that must be ported are present in the `src/port/bsp`, `src/port/net` and `src/port/os` of the IoT core device library.

- BSP(Board Support Package)

The BSP is the well-defined set of functions that the IoT core device library invokes in order to interact with a platform's specific networking, file IO, random number generator, and system feature.

- File system
 - * `iot_bsp_fs_init()`
 - * `iot_bsp_fs_open()`
 - * `iot_bsp_fs_read()`
 - * `iot_bsp_fs_write()`
 - * `iot_bsp_fs_close()`
 - * `iot_bsp_fs_remove()`
 - * `iot_bsp_fs_deinit()`
- Non-volatile storage
 - * `iot_bsp_nv_get_data_path()`
- Random number
 - * `iot_bsp_random()`
- System
 - * `iot_bsp_system_get_time_in_sec()`
 - * `iot_bsp_system_get_uniqueid()`
 - * `iot_bsp_system_poweroff()`
 - * `iot_bsp_system_reboot()`
 - * `iot_bsp_system_set_time_in_sec()`

- Wifi
 - * `iot_bsp_wifi_init()`
 - * `iot_bsp_wifi_get_freq()`
 - * `iot_bsp_wifi_get_mac()`
 - * `iot_bsp_wifi_get_scan_result()`
 - * `iot_bsp_wifi_set_mode()`
- Debug
 - * `iot_bsp_debug()`
 - * `iot_bsp_debug_check_heap()`

- Network

The STDK has a network layer with TLS communication using various TLS/SSL libraries. SmartThings does not support non-TLS communication.

- `iot_net_init()`
- OS

These APIs are related to operating system.

- `iot_os_delay()`
- `iot_os_eventgroup_clear_bits()`
- `iot_os_eventgroup_create()`
- `iot_os_eventgroup_delete()`
- `iot_os_eventgroup_set_bits()`
- `iot_os_eventgroup_wait_bits()`
- `iot_os_mutex_init()`
- `iot_os_mutex_lock()`
- `iot_os_mutex_unlock()`
- `iot_os_queue_create()`
- `iot_os_queue_delete()`
- `iot_os_queue_receive()`
- `iot_os_queue_reset()`
- `iot_os_queue_send()`
- `iot_os_thread_create()`
- `iot_os_thread_delete()`
- `iot_os_thread_yield()`
- `iot_os_timer_init()`
- `iot_os_timer_count_ms()`
- `iot_os_timer_isexpired()`
- `iot_os_timer_left_ms()`
- `iot_os_timer_destroy()`

1.4 Device Application

A device application is developed using the APIs provided by the IoT Core Device Library. We recommend reuse of the pre-supplied example applications, like `st_switch`. This allows for rapid development as you begin to develop a your new device. Please refer to the API references related to the IoT core device library as shown before

2 Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

noti_data_raw_t::quota	9
noti_data_raw_t::rate_limit	10
iot_bsp_fs_handle_t	12
iot_cap_cmd_data_t Contains data for "command" payload	13
iot_cap_cmd_set Contains user command callback function data	14
iot_cap_cmd_set_list Linked list for command callback function data	16
iot_cap_evt_data_t Contains data for "deviceEvent" payload	18
iot_cap_handle Contains data for capability handle	20
iot_cap_handle_list Linked list for capability handle	22
iot_cap_unit_t Contains a "unit" data	23
iot_cap_val_t Contains a various type of data which can be int, double, string and string array	24
iot_cloud_prov_data Contains "cloud provisioning" data	26
iot_command Contains "internal command" data	27
iot_context Contains "iot core's main context" data	28
iot_crypto_cipher_info Contains cipher informations	35
iot_crypto_ecdh_params Contains parameters to share a key between Things and ST apps	37
iot_crypto_ed25519_keypair Contains Ed25519 and Curve25519 key pairs	38
iot_crypto_keypair Contains public and private keys for Ed25519	39
iot_crypto_pk_context Contains key pair information and public key based function lists	40

iot_crypto_pk_funcs	Contains public key based function lists	41
iot_crypto_pk_info	Contains information of key pair	43
iot_devconf_prov_data	Contains "device configuration" data	44
iot_device_info	Contains "device's information" data	46
iot_device_prov_data	Contains "all device's provisioning" data	47
iot_easysetup_payload	Contains "easy-setup payload" data	49
iot_mac	Contains "wifi mac" data	50
iot_mqtt_ctx	Contains "mqtt handling context" data	51
iot_net_connection	Contains server related information	54
iot_net_interface	Contains "network management structure" data	56
iot_net_platform_context	Contains connection context	59
iot_noti_data_t	Contains data for notification data	61
iot_os_mutex	Contains a mutex data	63
iot_pin_t	Contains a pin values for pin type onboarding process	64
iot_registered_data	Contains "registration message" data	65
iot_state_data	Contains "iot core's main state" data	66
iot_uuid	Contains "uuid" data	67
iot_wifi_conf	Contains a "wifi stack configuration" data	68
iot_wifi_prov_data	Contains "wifi provisioning" data	69
iot_wifi_scan_result_t	Contains a "wifi scan" data	71
MessageData		72

MQTTClient::MessageHandlers	74
MQTTClient	75
MQTTConnackData	78
MQTTConnackFlags	79
MQTTConnectFlags	80
MQTTHeader	82
MQTTLenString	83
MQTTMessage	84
MQTTPacket_connectData	85
MQTTPacket_willOptions	88
MQTTString	90
MQTTSubackData	91
MQTTTransport	91
noti_data_raw_t Contains data for raw data of each notification	93
url_parse_t Contains a "url parse" data	94

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

st-device-sdk-c/src/iot_api.c	296
st-device-sdk-c/src/iot_capability.c	305
st-device-sdk-c/src/iot_crypto.c	313
st-device-sdk-c/src/iot_jwt.c	316
st-device-sdk-c/src/iot_main.c	318
st-device-sdk-c/src/iot_mqtt.c	321
st-device-sdk-c/src/iot_nv_data.c	327
st-device-sdk-c/src/iot_util.c	344
st-device-sdk-c/src/iot_uuid.c	349
st-device-sdk-c/src/crypto/iot_crypto_ed25519.c	95
st-device-sdk-c/src/crypto/mbedtls/iot_crypto_mbedtls.c	98

st-device-sdk-c/src/crypto/ss/ iot_crypto_ss.c	104
st-device-sdk-c/src/easysetup/ iot_easysetup_crypto.c	111
st-device-sdk-c/src/easysetup/ iot_easysetup_d2d.c	113
st-device-sdk-c/src/easysetup/ iot_easysetup_st_mqtt.c	116
st-device-sdk-c/src/easysetup/http/ iot_easysetup_http.c	106
st-device-sdk-c/src/include/ iot_capability.h	137
st-device-sdk-c/src/include/ iot_crypto.h	140
st-device-sdk-c/src/include/ iot_crypto_internal.h	157
st-device-sdk-c/src/include/ iot_debug.h	157
st-device-sdk-c/src/include/ iot_easysetup.h	161
st-device-sdk-c/src/include/ iot_error.h	169
st-device-sdk-c/src/include/ iot_internal.h	171
st-device-sdk-c/src/include/ iot_jwt.h	189
st-device-sdk-c/src/include/ iot_main.h	191
st-device-sdk-c/src/include/ iot_net.h	196
st-device-sdk-c/src/include/ iot_nv_data.h	198
st-device-sdk-c/src/include/ iot_util.h	213
st-device-sdk-c/src/include/ iot_uuid.h	218
st-device-sdk-c/src/include/ st_dev.h	282
st-device-sdk-c/src/include/ st_dev_version.h	295
st-device-sdk-c/src/include/bsp/ iot_bsp_debug.h	118
st-device-sdk-c/src/include/bsp/ iot_bsp_fs.h	120
st-device-sdk-c/src/include/bsp/ iot_bsp_nv_data.h	126
st-device-sdk-c/src/include/bsp/ iot_bsp_random.h	128
st-device-sdk-c/src/include/bsp/ iot_bsp_system.h	129
st-device-sdk-c/src/include/bsp/ iot_bsp_wifi.h	132
st-device-sdk-c/src/include/mqtt/ iot_mqtt_client.h	220
st-device-sdk-c/src/include/mqtt/ iot_mqtt_connect.h	234
st-device-sdk-c/src/include/mqtt/ iot_mqtt_format.h	240
st-device-sdk-c/src/include/mqtt/ iot_mqtt_packet.h	243
st-device-sdk-c/src/include/mqtt/ iot_mqtt_publish.h	253
st-device-sdk-c/src/include/mqtt/ iot_mqtt_stacktrace.h	257

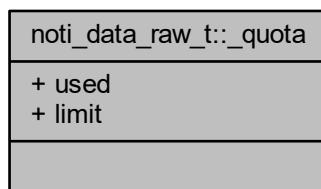
st-device-sdk-c/src/include/mqtt/ iot_mqtt_subscribe.h	260
st-device-sdk-c/src/include/mqtt/ iot_mqtt_unsubscribe.h	264
st-device-sdk-c/src/include/os/ iot_os_util.h	268
st-device-sdk-c/src/mqtt/client/ iot_mqtt_client.c	351
st-device-sdk-c/src/mqtt/packet/ iot_mqtt_connect_client.c	365
st-device-sdk-c/src/mqtt/packet/ iot_mqtt_connect_server.c	370
st-device-sdk-c/src/mqtt/packet/ iot_mqtt_deserialize_publish.c	372
st-device-sdk-c/src/mqtt/packet/ iot_mqtt_format.c	373
st-device-sdk-c/src/mqtt/packet/ iot_mqtt_packet.c	376
st-device-sdk-c/src/mqtt/packet/ iot_mqtt_serialize_publish.c	384
st-device-sdk-c/src/mqtt/packet/ iot_mqtt_subscribe_client.c	389
st-device-sdk-c/src/mqtt/packet/ iot_mqtt_subscribe_server.c	392
st-device-sdk-c/src/mqtt/packet/ iot_mqtt_unsubscribe_client.c	394
st-device-sdk-c/src/mqtt/packet/ iot_mqtt_unsubscribe_server.c	396
st-device-sdk-c/src/port/net/mbedtls/ iot_net_mbedtls.c	398
st-device-sdk-c/src/port/net/mbedtls/ iot_net_platform.h	399
st-device-sdk-c/src/port/net/openssl/ iot_net_openssl.c	401
st-device-sdk-c/src/port/net/openssl/ iot_net_platform.h	400

4 Data Structure Documentation

4.1 `noti_data_raw_t::_quota` Struct Reference

```
#include <st_dev.h>
```

Collaboration diagram for `noti_data_raw_t::_quota`:



Data Fields

- int **used**
Current used data usage in bytes.
- int **limit**
Current data limit in bytes.

4.1.1 Field Documentation

4.1.1.1 **limit** int limit

Current data limit in bytes.

4.1.1.2 **used** int used

Current used data usage in bytes.

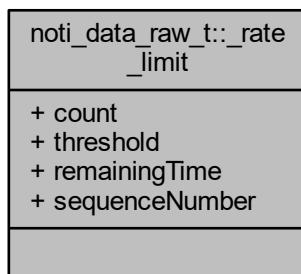
The documentation for this struct was generated from the following file:

- st-device-sdk-c/src/include/[st_dev.h](#)

4.2 noti_data_raw_t::rate_limit Struct Reference

```
#include <st_dev.h>
```

Collaboration diagram for noti_data_raw_t::rate_limit:



Data Fields

- int **count**
Current rate limit count.
- int **threshold**
Current rate limit threshold.
- int **remainingTime**
How much time remains for rate limit releasing.
- int **sequenceNumber**
Sequence number of event that triggered rate limit.

4.2.1 Field Documentation

4.2.1.1 **count** int count

Current rate limit count.

4.2.1.2 **remainingTime** int remainingTime

How much time remains for rate limit releasing.

4.2.1.3 **sequenceNumber** int sequenceNumber

Sequence number of event that triggered rate limit.

4.2.1.4 **threshold** int threshold

Current rate limit threshold.

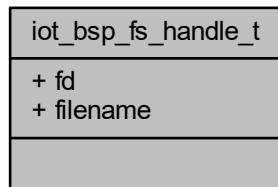
The documentation for this struct was generated from the following file:

- st-device-sdk-c/src/include/st_dev.h

4.3 iot_bsp_fs_handle_t Struct Reference

```
#include <iot_bsp_fs.h>
```

Collaboration diagram for iot_bsp_fs_handle_t:



Data Fields

- int [fd](#)
- char [filename](#) [128]

4.3.1 Field Documentation

4.3.1.1 [fd](#) int fd

4.3.1.2 [filename](#) char filename[128]

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

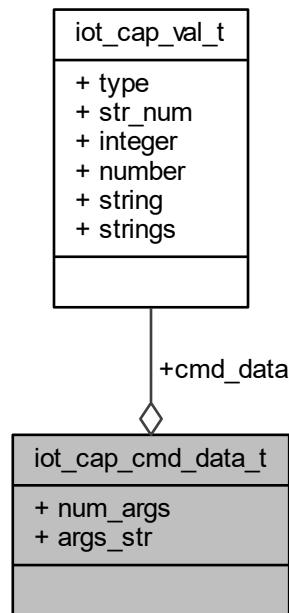
- st-device-sdk-c/src/include/bsp/[iot_bsp_fs.h](#)

4.4 iot_cap_cmd_data_t Struct Reference

Contains data for "command" payload.

```
#include <st_dev.h>
```

Collaboration diagram for iot_cap_cmd_data_t:



Data Fields

- `uint8_t num_args`
Number of arguments.
- `char * args_str [MAX_CAP_ARG]`
Name of each argument.
- `iot_cap_val_t cmd_data [MAX_CAP_ARG]`
Value of each arguments.

4.4.1 Detailed Description

Contains data for "command" payload.

4.4.2 Field Documentation

4.4.2.1 args_str `char* args_str[MAX_CAP_ARG]`

Name of each argument.

Note

This is used only if there is more than one argument.

4.4.2.2 cmd_data `iot_cap_val_t cmd_data[MAX_CAP_ARG]`

Value of each arguments.

4.4.2.3 num_args `uint8_t num_args`

Number of arguments.

Note

Usually 1, but if commands type is 'json object', it could be more than 1. (See colorControl capability.)

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

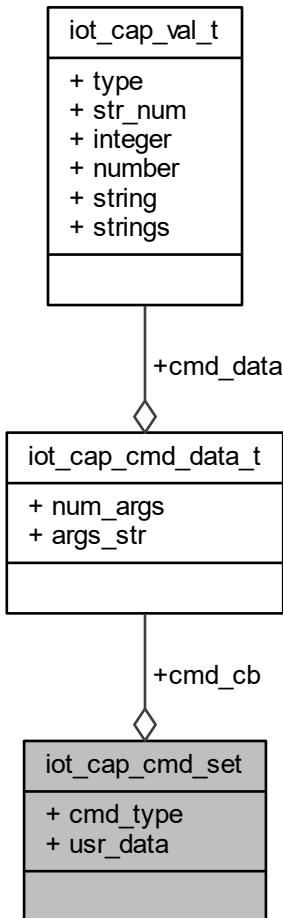
- st-device-sdk-c/src/include/[st_dev.h](#)

4.5 iot_cap_cmd_set Struct Reference

Contains user command callback function data.

```
#include <iot_capability.h>
```

Collaboration diagram for iot_cap_cmd_set:



Data Fields

- `const char * cmd_type`
NULL-terminated string, which is name of commands.
- `st_cap_cmd_cb cmd_cb`
User callback function.
- `void * usr_data`
User data for cmd_cb.

4.5.1 Detailed Description

Contains user command callback function data.

4.5.2 Field Documentation

4.5.2.1 cmd_cb [st_cap_cmd_cb](#) cmd_cb

User callback function.

4.5.2.2 cmd_type const char* cmd_type

NULL-terminated string, which is name of commands.

4.5.2.3 usr_data void* usr_data

User data for cmd_cb.

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

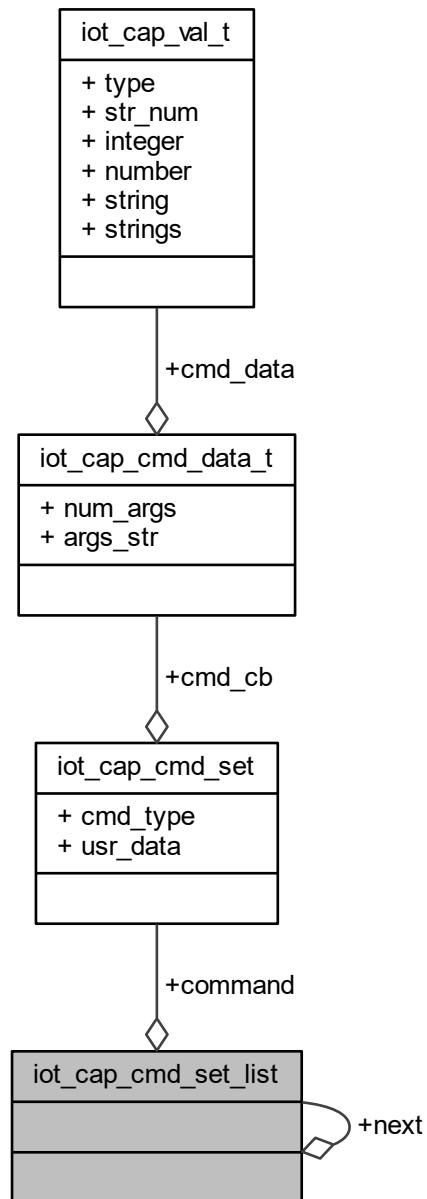
- st-device-sdk-c/src/include/[iot_capability.h](#)

4.6 iot_cap_cmd_set_list Struct Reference

linked list for command callback function data

```
#include <iot_capability.h>
```

Collaboration diagram for iot_cap_cmd_set_list:



Data Fields

- struct `iot_cap_cmd_set` * `command`
a pointer to a command data
- struct `iot_cap_cmd_set_list` * `next`
a pointer to a next list

4.6.1 Detailed Description

linked list for command callback function data

4.6.2 Field Documentation

4.6.2.1 **command** struct iot_cap_cmd_set* command

a pointer to a command data

4.6.2.2 **next** struct iot_cap_cmd_set_list* next

a pointer to a next list

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

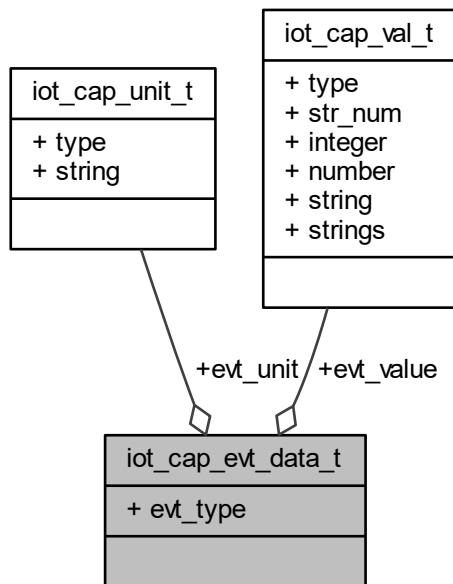
- st-device-sdk-c/src/include/[iot_capability.h](#)

4.7 iot_cap_evt_data_t Struct Reference

Contains data for "deviceEvent" payload.

```
#include <iot_capability.h>
```

Collaboration diagram for iot_cap_evt_data_t:



Data Fields

- const char * **evt_type**
NULL-terminated string, which is name of attributes.
- **iot_cap_val_t evt_value**
'value' data for deviceEvent.
- **iot_cap_unit_t evt_unit**
'unit' data for deviceEvent.

4.7.1 Detailed Description

Contains data for "deviceEvent" payload.

4.7.2 Field Documentation

4.7.2.1 **evt_type** const char* evt_type

NULL-terminated string, which is name of attributes.

4.7.2.2 **evt_unit** [iot_cap_unit_t](#) evt_unit

'unit' data for deviceEvent.

4.7.2.3 **evt_value** [iot_cap_val_t](#) evt_value

'value' data for deviceEvent.

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

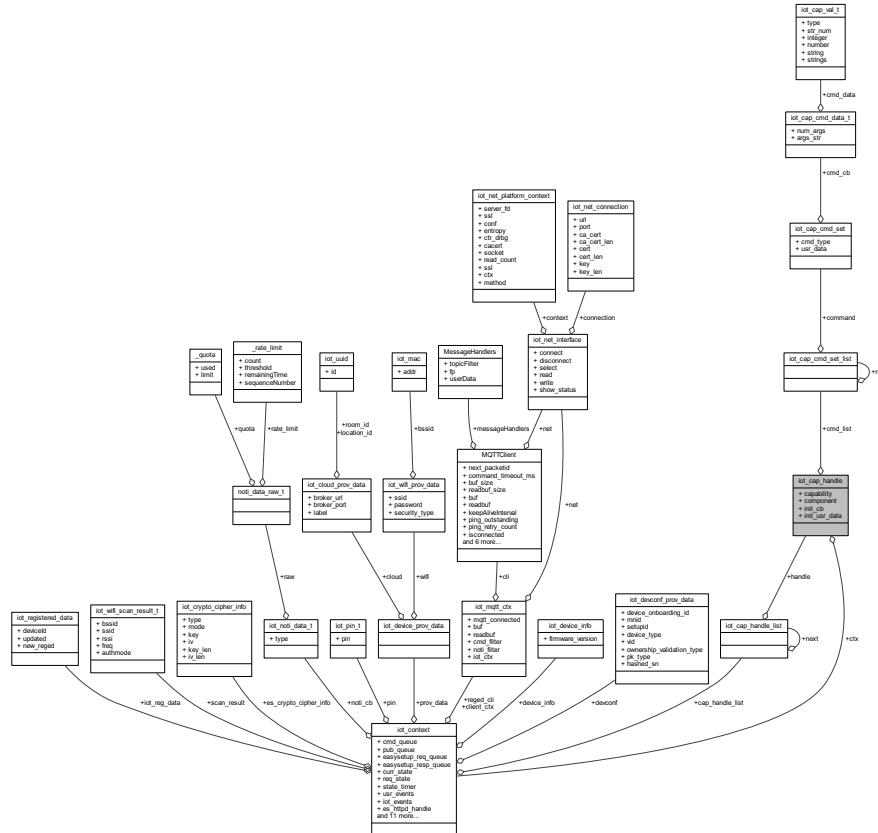
-
- st-device-sdk-c/src/include/[iot_capability.h](#)

4.8 iot_cap_handle Struct Reference

Contains data for capability handle.

```
#include <iot_capability.h>
```

Collaboration diagram for iot_cap_handle:



Data Fields

- const char * **capability**
NULL-terminated string, which is name of capability.
- const char * **component**
NULL-terminated string, which is name of component.
- struct **iot_cap_cmd_set_list** * **cmd_list**
List of command data.
- st_cap_init_cb **init_cb**
User callback function for init device state.
- void * **init_usr_data**
User data for init_cb.
- struct **iot_context** * **ctx**
ctx

4.8.1 Detailed Description

Contains data for capability handle.

4.8.2 Field Documentation

4.8.2.1 capability const char* capability

NULL-terminated string, which is name of capability.

Use capability id for this variable. e.g. "switchLevel"

4.8.2.2 cmd_list struct iot_cap_cmd_set_list* cmd_list

List of command data.

4.8.2.3 component const char* component

NULL-terminated string, which is name of component.

4.8.2.4 ctx struct iot_context* ctx

ctx

4.8.2.5 init_cb st_cap_init_cb init_cb

User callback function for init device state.

4.8.2.6 init_usr_data void* init_usr_data

User data for init_cb.

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

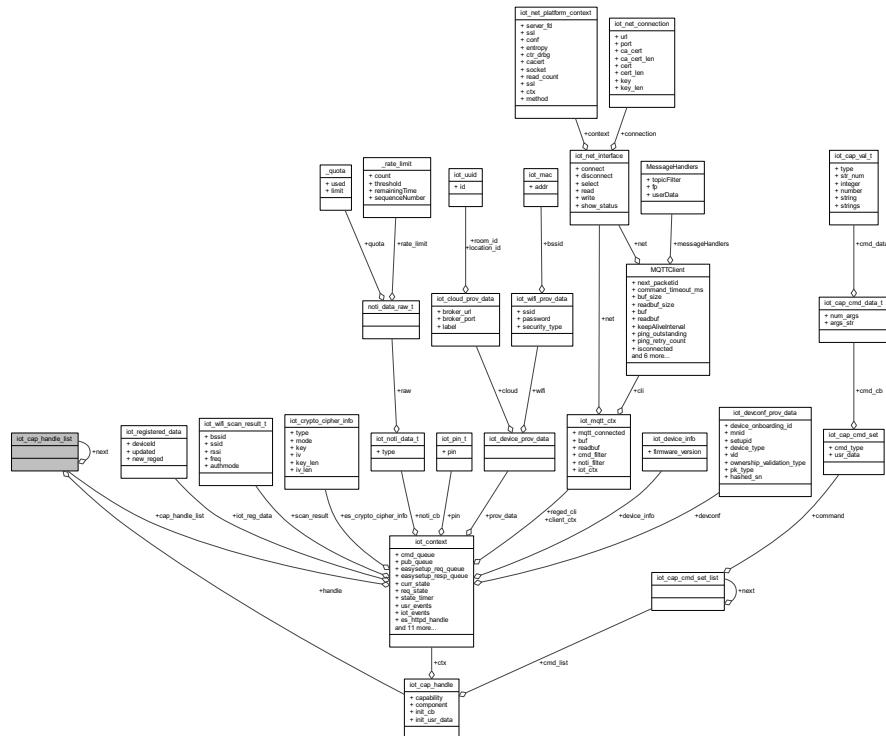
- st-device-sdk-c/src/include/[iot_capability.h](#)

4.9 iot_cap_handle_list Struct Reference

linked list for capability handle

```
#include <iot_capability.h>
```

Collaboration diagram for iot_cap_handle_list:



Data Fields

- struct **iot_cap_handle** * **handle**
a pointer to a capability handle
- struct **iot_cap_handle_list** * **next**
a pointer to a next list

4.9.1 Detailed Description

linked list for capability handle

4.9.2 Field Documentation

4.9.2.1 handle struct iot_cap_handle* handle

a pointer to a capability handle

4.9.2.2 next struct iot_cap_handle_list* next

a pointer to a next list

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

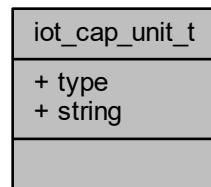
- st-device-sdk-c/src/include/[iot_capability.h](#)

4.10 iot_cap_unit_t Struct Reference

Contains a "unit" data.

```
#include <iot_capability.h>
```

Collaboration diagram for iot_cap_unit_t:



Data Fields

- `uint8_t type`
Unused or string.
- `char * string`
NULL-terminated string.

4.10.1 Detailed Description

Contains a "unit" data.

4.10.2 Field Documentation

4.10.2.1 **string** `char* string`

NULL-terminated string.

4.10.2.2 **type** `uint8_t type`

Unused or string.

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

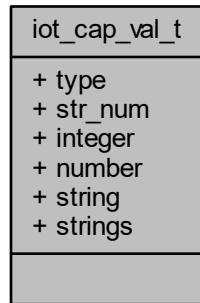
- st-device-sdk-c/src/include/[iot_capability.h](#)

4.11 **iot_cap_val_t** Struct Reference

Contains a various type of data which can be int, double, string and string array.

```
#include <st_dev.h>
```

Collaboration diagram for iot_cap_val_t:



Data Fields

- [iot_cap_val_type_t type](#)
Data type to notify valid data.
- [uint8_t str_num](#)
Number of strings. Only used for sting array.
- [int integer](#)
Integer.
- [double number](#)
Float number.
- [char * string](#)
NULL-terminated string.
- [char ** strings](#)
Array of NULL-terminated strings.

4.11.1 Detailed Description

Contains a various type of data which can be int, double, string and string array.

4.11.2 Field Documentation

4.11.2.1 **integer** int integer

Integer.

4.11.2.2 **number** double number

Float number.

4.11.2.3 **str_num** uint8_t str_num

Number of strings. Only used for sting array.

4.11.2.4 **string** char* string

NULL-terminated string.

4.11.2.5 **strings** char** strings

Array of NULL-terminated strings.

4.11.2.6 **type** iot_cap_val_type_t type

Data type to notify valid data.

Note

Even though there are 4 different type of data (integer, number, string, strings) in this structure, only one type of data is used.

Type of capability's data.

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

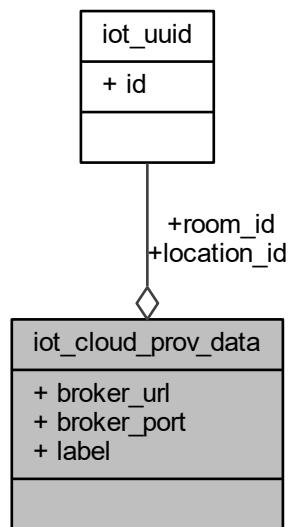
- st-device-sdk-c/src/include/[st_dev.h](#)

4.12 iot_cloud_prov_data Struct Reference

Contains "cloud provisioning" data.

```
#include <iot_main.h>
```

Collaboration diagram for iot_cloud_prov_data:



Data Fields

- char * **broker_url**
broker url for mqtt
- int **broker_port**
broker port num for mqtt
- struct **iot_uuid location_id**
location id for ST(server) management
- struct **iot_uuid room_id**
room id for ST(server) management
- char * **label**
device name, got from the mobile

4.12.1 Detailed Description

Contains "cloud provisioning" data.

4.12.2 Field Documentation

4.12.2.1 broker_port int broker_port

broker port num for mqtt

4.12.2.2 broker_url char* broker_url

broker url for mqtt

4.12.2.3 label char* label

device name, got from the mobile

4.12.2.4 location_id struct iot_uuid location_id

location id for ST(server) management

4.12.2.5 room_id struct iot_uuid room_id

room id for ST(server) management

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

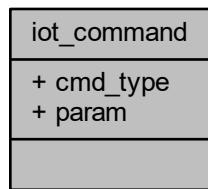
- st-device-sdk-c/src/include/[iot_main.h](#)

4.13 iot_command Struct Reference

Contains "internal command" data.

```
#include <iot_main.h>
```

Collaboration diagram for iot_command:



Data Fields

- enum [iot_command_type](#) cmd_type
command type to handle device
- void * [param](#)
additional inform for each command

4.13.1 Detailed Description

Contains "internal command" data.

4.13.2 Field Documentation

4.13.2.1 cmd_type enum [iot_command_type](#) cmd_type

command type to handle device

4.13.2.2 param void* param

additional inform for each command

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

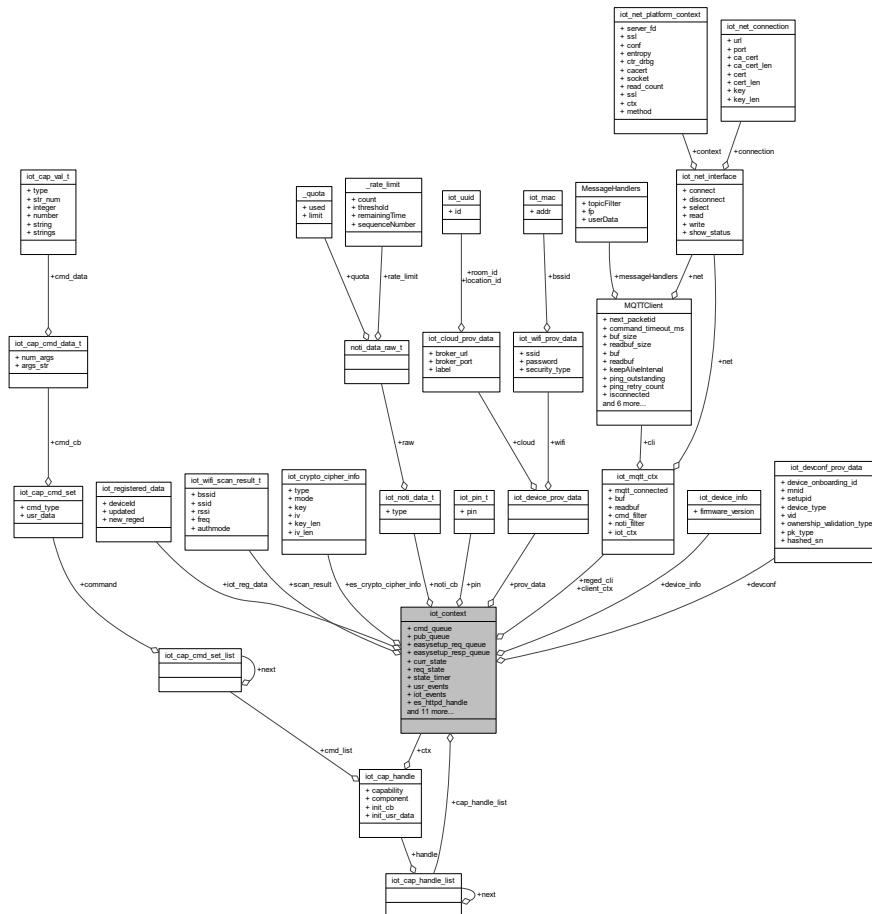
- st-device-sdk-c/src/include/[iot_main.h](#)

4.14 iot_context Struct Reference

Contains "iot core's main context" data.

```
#include <iot_main.h>
```

Collaboration diagram for iot_context:



Data Fields

- `iot_os_queue * cmd_queue`
iot core's internal command queue
- `iot_os_queue * pub_queue`
iot core's event publish queue
- `iot_os_queue * easysetup_req_queue`
request queue for easy-setup process
- `iot_os_queue * easysetup_resp_queue`
response queue for easy-setup process
- `iot_state_t curr_state`
reflect current iot_state
- `iot_state_t req_state`
reflect requested iot_state
- `iot_timer state_timer`
state checking timer for each iot_state
- `iot_os_eventgroup * user_events`
User level handling events.
- `iot_os_eventgroup * iot_events`
Internal handling events.

- `iot_cap_handle_list_t * cap_handle_list`
allocated capability handle lists
- `struct iot_mqtt_ctx * client_ctx`
mqtt context ref. for registration
- `struct iot_mqtt_ctx * reged_cli`
mqtt context ref. for connection
- `struct iot_device_prov_data prov_data`
allocated device provisioning data
- `struct iot_devconf_prov_data devconf`
allocated device configuration data
- `struct iot_device_info device_info`
allocated device information data
- `iot_crypto_cipher_info_t * es_crypto_cipher_info`
cipher context ref. for easy-setup process
- `struct iot_registered_data iot_reg_data`
allocated registration data from server
- `void * es_httpd_handle`
httpd handler for easy-setup process
- `uint16_t scan_num`
number of wifi ap scan result
- `iot_wifi_scan_result_t * scan_result`
actual data lists of each wifi ap scan result
- `char * lookup_id`
device's lookup id for server & mobile side notification
- `st_cap_noti_cb noti_cb`
notification handling callback for each capability
- `void * noti_usr_data`
notification handling callback data for user
- `st_status_cb status_cb`
iot core status handling callback for user
- `iot_status_t status_maps`
iot status callback maps to check it call or not
- `unsigned int reported_stat`
iot status callback checking flag to check it reported or not
- `void * status_usr_data`
iot core status handling callback data for user
- `int curr_otm_feature`
current device's supported onboarding process validation type
- `iot_pin_t * pin`
current device's PIN values for PIN type otm
- `unsigned int cmd_err`
current command handling error checking value
- `unsigned int cmd_status`
current command status
- `uint16_t cmd_count [IOT_COMMAND_TYPE_MAX]`
current queued command counts

4.14.1 Detailed Description

Contains "iot core's main context" data.

4.14.2 Field Documentation

4.14.2.1 cap_handle_list `iot_cap_handle_list_t*` cap_handle_list

allocated capability handle lists

4.14.2.2 client_ctx `struct iot_mqtt_ctx*` client_ctx

mqtt context ref. for registration

4.14.2.3 cmd_count `uint16_t cmd_count[IOT_COMMAND_TYPE_MAX]`

current queued command counts

4.14.2.4 cmd_err `unsigned int cmd_err`

current command handling error checking value

4.14.2.5 cmd_queue `iot_os_queue* cmd_queue`

iot core's internal command queue

4.14.2.6 cmd_status `unsigned int cmd_status`

current command status

4.14.2.7 curr_otm_feature `int curr_otm_feature`

current device's supported onboarding process validation type

4.14.2.8 curr_state `iot_state_t curr_state`

reflect current iot_state

4.14.2.9 devconf `struct iot_devconf_prov_data devconf`

allocated device configuration data

4.14.2.10 device_info `struct iot_device_info device_info`

allocated device information data

4.14.2.11 easysetup_req_queue `iot_os_queue* easysetup_req_queue`

request queue for easy-setup process

4.14.2.12 easysetup_resp_queue `iot_os_queue* easysetup_resp_queue`

response queue for easy-setup process

4.14.2.13 es_crypto_cipher_info `iot_crypto_cipher_info_t* es_crypto_cipher_info`

cipher context ref. for easy-setup process

4.14.2.14 es_httpd_handle `void* es_httpd_handle`

httpd handler for easy-setup process

4.14.2.15 iot_events `iot_os_eventgroup* iot_events`

Internal handling events.

4.14.2.16 iot_reg_data struct iot_registered_data iot_reg_data

allocated registration data from server

4.14.2.17 lookup_id char* lookup_id

device's lookup id for server & mobile side notification

4.14.2.18 noti_cb st_cap_noti_cb noti_cb

notification handling callback for each capability

4.14.2.19 noti_usr_data void* noti_usr_data

notification handling callback data for user

4.14.2.20 pin iot_pin_t* pin

current device's PIN values for PIN type otm

4.14.2.21 prov_data struct iot_device_prov_data prov_data

allocated device provisioning data

4.14.2.22 pub_queue iot_os_queue* pub_queue

iot core's event publish queue

4.14.2.23 reged_cli struct iot_mqtt_ctxt* reged_cli

mqtt context ref. for connection

4.14.2.24 reported_stat `unsigned int reported_stat`

iot status callback checking flag to check it reported or not

4.14.2.25 req_state `iot_state_t req_state`

reflect requested iot_state

4.14.2.26 scan_num `uint16_t scan_num`

number of wifi ap scan result

4.14.2.27 scan_result `iot_wifi_scan_result_t* scan_result`

actual data lists of each wifi ap scan result

4.14.2.28 state_timer `iot_os_timer state_timer`

state checking timer for each iot_state

4.14.2.29 status_cb `st_status_cb status_cb`

iot core status handling callback for user

4.14.2.30 status_maps `iot_status_t status_maps`

iot status callback maps to check it call or not

4.14.2.31 status_usr_data `void* status_usr_data`

iot core status handling callback data for user

4.14.2.32 usr_events iot_os_eventgroup* usr_events

User level handling events.

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

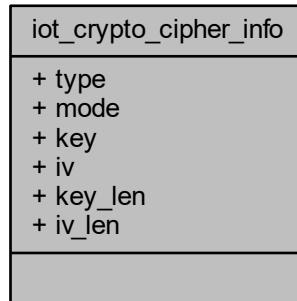
- st-device-sdk-c/src/include/iot_main.h

4.15 iot_crypto_cipher_info Struct Reference

Contains cipher informations.

```
#include <iot_crypto.h>
```

Collaboration diagram for iot_crypto_cipher_info:



Data Fields

- `iot_crypto_cipher_type_t type`
type of cipher algorithm
- `iot_crypto_cipher_mode_t mode`
mode of cipher operation
- `unsigned char * key`
a pointer to a shared key
- `unsigned char * iv`
a pointer to a IV for AES cipher
- `size_t key_len`
a length of a shared key
- `size_t iv_len`
a length of a IV

4.15.1 Detailed Description

Contains cipher informations.

4.15.2 Field Documentation

4.15.2.1 iv `unsigned char* iv`

a pointer to a IV for AES cipher

4.15.2.2 iv_len `size_t iv_len`

a length of a IV

4.15.2.3 key `unsigned char* key`

a pointer to a shared key

4.15.2.4 key_len `size_t key_len`

a length of a shared key

4.15.2.5 mode `iot_crypto_cipher_mode_t mode`

mode of cipher operation

4.15.2.6 type `iot_crypto_cipher_type_t type`

type of cipher algorithm

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

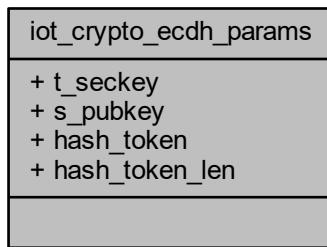
- st-device-sdk-c/src/include/[iot_crypto.h](#)

4.16 iot_crypto_ecdh_params Struct Reference

Contains parameters to share a key between Things and ST apps.

```
#include <iot_crypto.h>
```

Collaboration diagram for iot_crypto_ecdh_params:



Data Fields

- `unsigned char * t_seckey`
a pointer to a things secret key based on curve25519
- `unsigned char * s_pubkey`
a pointer to a server public key based on curve25519
- `unsigned char * hash_token`
a pointer to a random token as a salt
- `size_t hash_token_len`
a length of random token

4.16.1 Detailed Description

Contains parameters to share a key between Things and ST apps.

4.16.2 Field Documentation

4.16.2.1 hash_token `unsigned char* hash_token`

a pointer to a random token as a salt

4.16.2.2 hash_token_len size_t hash_token_len

a length of random token

4.16.2.3 s_pubkey unsigned char* s_pubkey

a pointer to a server public key based on curve25519

4.16.2.4 t_seckey unsigned char* t_seckey

a pointer to a things secret key based on curve25519

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

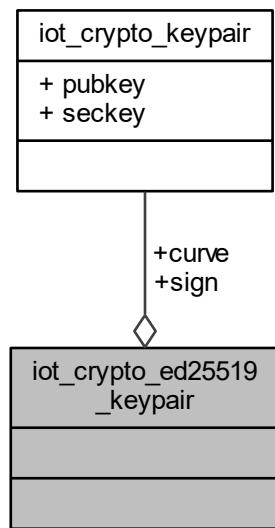
- st-device-sdk-c/src/include/[iot_crypto.h](#)

4.17 iot_crypto_ed25519_keypair Struct Reference

Contains Ed25519 and Curve25519 key pairs.

```
#include <iot_crypto.h>
```

Collaboration diagram for iot_crypto_ed25519_keypair:



Data Fields

- struct [iot_crypto_keypair sign](#)
a pointer to a Ed25519 key pair for signature
- struct [iot_crypto_keypair curve](#)
a pointer to a curve25519 key pair for encryption/decryption

4.17.1 Detailed Description

Contains Ed25519 and Curve25519 key pairs.

4.17.2 Field Documentation

4.17.2.1 **curve** struct [iot_crypto_keypair](#) curve

a pointer to a curve25519 key pair for encryption/decryption

4.17.2.2 **sign** struct [iot_crypto_keypair](#) sign

a pointer to a Ed25519 key pair for signature

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

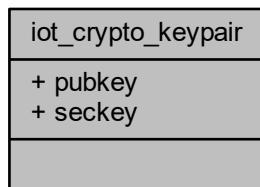
- st-device-sdk-c/src/include/[iot_crypto.h](#)

4.18 iot_crypto_keypair Struct Reference

Contains public and private keys for Ed25519.

```
#include <iot_crypto.h>
```

Collaboration diagram for iot_crypto_keypair:



Data Fields

- unsigned char * **pubkey**
- unsigned char * **seckey**
a pointer to a public key

4.18.1 Detailed Description

Contains public and private keys for Ed25519.

4.18.2 Field Documentation

4.18.2.1 **pubkey** unsigned char* pubkey

4.18.2.2 **seckey** unsigned char* seckey

a pointer to a public key

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

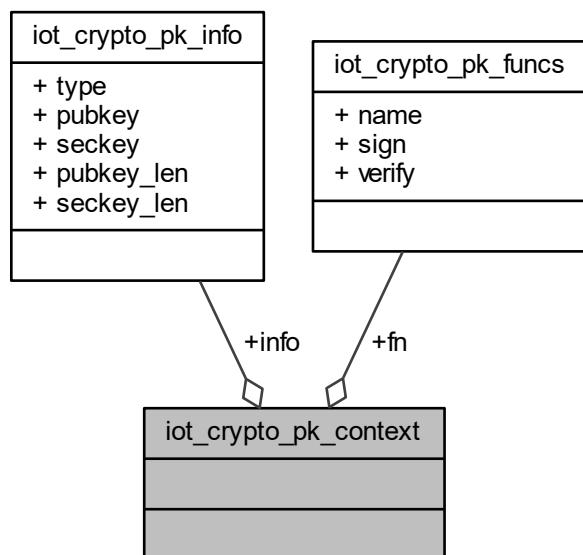
- st-device-sdk-c/src/include/[iot_crypto.h](#)

4.19 iot_crypto_pk_context Struct Reference

Contains key pair information and public key based function lists.

```
#include <iot_crypto.h>
```

Collaboration diagram for iot_crypto_pk_context:



Data Fields

- `iot_crypto_pk_info_t * info`
- `const iot_crypto_pk_funcs_t * fn`
a pointer to a key pair info

4.19.1 Detailed Description

Contains key pair information and public key based function lists.

4.19.2 Field Documentation

4.19.2.1 `fn const iot_crypto_pk_funcs_t* fn`

a pointer to a key pair info

4.19.2.2 `info iot_crypto_pk_info_t* info`

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

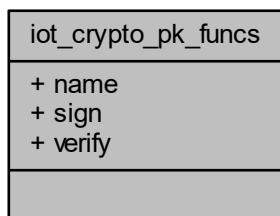
- st-device-sdk-c/src/include/[iot_crypto.h](#)

4.20 iot_crypto_pk_funcs Struct Reference

Contains public key based function lists.

```
#include <iot_crypto.h>
```

Collaboration diagram for iot_crypto_pk_funcs:



Data Fields

- const char * **name**
- **iot_error_t**(*** sign**)(**iot_crypto_pk_context_t** *ctx, unsigned char *input, size_t ilen, unsigned char *sig, size_t *slen)
string name to know this
- **iot_error_t**(*** verify**)(**iot_crypto_pk_context_t** *ctx, unsigned char *input, size_t ilen, unsigned char *sig, size_t *slen)
a pointer to a function to verify a signature

4.20.1 Detailed Description

Contains public key based function lists.

4.20.2 Field Documentation

4.20.2.1 **name** const char* name

4.20.2.2 **sign** **iot_error_t**(*** sign**) (**iot_crypto_pk_context_t** *ctx, unsigned char *input, size_t ilen, unsigned char *sig, size_t *slen)

string name to know this

a pointer to a function to create a signature

4.20.2.3 **verify** **iot_error_t**(*** verify**) (**iot_crypto_pk_context_t** *ctx, unsigned char *input, size_t ilen, unsigned char *sig, size_t *slen)

a pointer to a function to verify a signature

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

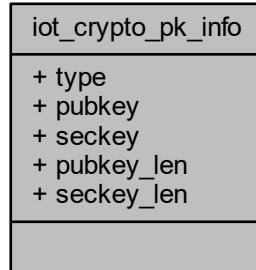
- st-device-sdk-c/src/include/**iot_crypto.h**

4.21 iot_crypto_pk_info Struct Reference

Contains information of key pair.

```
#include <iot_crypto.h>
```

Collaboration diagram for iot_crypto_pk_info:



Data Fields

- `iot_crypto_pk_type_t type`
type of key pair
- `unsigned char * pubkey`
public key of key pair
- `unsigned char * seckey`
private key of key pair
- `size_t pubkey_len`
length of public key
- `size_t seckey_len`
length of private key

4.21.1 Detailed Description

Contains information of key pair.

4.21.2 Field Documentation

4.21.2.1 pubkey `unsigned char* pubkey`

type of key pair

4.21.2.2 pubkey_len size_t pubkey_len

private key of key pair

4.21.2.3 seckey unsigned char* seckey

public key of key pair

4.21.2.4 seckey_len size_t seckey_len

length of public key

4.21.2.5 type iot_crypto_pk_type_t type

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

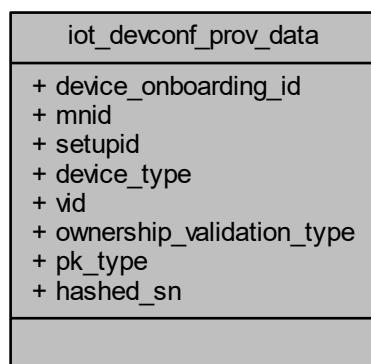
- st-device-sdk-c/src/include/[iot_crypto.h](#)

4.22 iot_devconf_prov_data Struct Reference

Contains "device configuration" data.

```
#include <iot_main.h>
```

Collaboration diagram for iot_devconf_prov_data:



Data Fields

- char * [device_onboarding_id](#)
onboarding id, determined in devworks
- char * [mnid](#)
mnid, determined in devworks
- char * [setupid](#)
setupid, determined in devworks
- char * [device_type](#)
device_type, determined in devworks
- char * [vid](#)
vid, determined in devworks
- int [ownership_validation_type](#)
onboarding process validation type, JUSTWORKS, BUTTON, PIN, QR
- [iot_crypto_pk_type_t](#) [pk_type](#)
Authentication type, determined in devworks.
- char * [hashed_sn](#)
hashed serial, self-generating values during onboarding process

4.22.1 Detailed Description

Contains "device configuration" data.

4.22.2 Field Documentation

4.22.2.1 [device_onboarding_id](#) char* device_onboarding_id

onboarding id, determined in devworks

4.22.2.2 [device_type](#) char* device_type

device_type, determined in devworks

4.22.2.3 [hashed_sn](#) char* hashed_sn

hashed serial, self-generating values during onboarding process

4.22.2.4 mnid `char* mnid`

mnid, determined in devworks

4.22.2.5 ownership_validation_type `int ownership_validation_type`

onboarding process validation type, JUSTWORKS, BUTTON, PIN, QR

4.22.2.6 pk_type `iot_crypto_pk_type_t pk_type`

Authentication type, determined in devworks.

4.22.2.7 setupid `char* setupid`

setupid, determined in devworks

4.22.2.8 vid `char* vid`

vid, determined in devworks

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

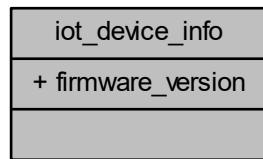
- st-device-sdk-c/src/include/[iot_main.h](#)

4.23 iot_device_info Struct Reference

Contains "device's information" data.

```
#include <iot_main.h>
```

Collaboration diagram for iot_device_info:



Data Fields

- char * **firmware_version**
device's binary/firmware version

4.23.1 Detailed Description

Contains "device's information" data.

4.23.2 Field Documentation

4.23.2.1 **firmware_version** char* firmware_version

device's binary/firmware version

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

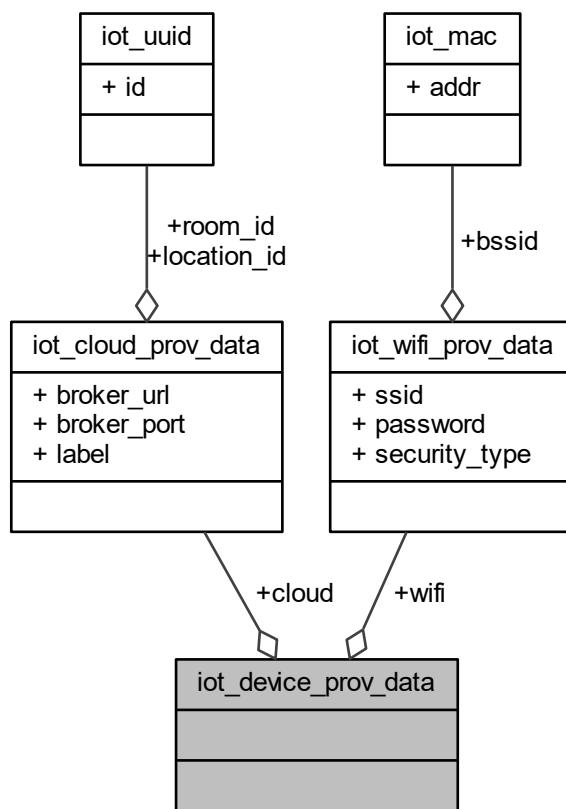
- st-device-sdk-c/src/include/[iot_main.h](#)

4.24 iot_device_prov_data Struct Reference

Contains "all device's provisioning" data.

```
#include <iot_main.h>
```

Collaboration diagram for iot_device_prov_data:



Data Fields

- struct [iot_wifi_prov_data](#) wifi
wifi provisionig data, refer to [iot_wifi_prov_data](#) struct
- struct [iot_cloud_prov_data](#) cloud
cloud provisionig data, refer to [iot_cloud_prov_data](#) struct

4.24.1 Detailed Description

Contains "all device's provisioning" data.

4.24.2 Field Documentation

4.24.2.1 cloud struct [iot_cloud_prov_data](#) cloud

cloud provisionig data, refer to [iot_cloud_prov_data](#) struct

4.24.2.2 wifi struct [iot_wifi_prov_data](#) wifi

wifi provisionig data, refer to [iot_wifi_prov_data](#) struct

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

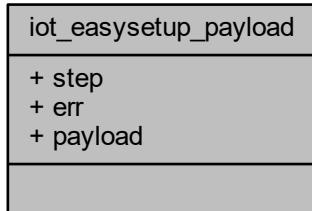
- st-device-sdk-c/src/include/[iot_main.h](#)

4.25 iot_easysetup_payload Struct Reference

Contains "easy-setup payload" data.

```
#include <iot_main.h>
```

Collaboration diagram for iot_easysetup_payload:



Data Fields

- enum [iot_easysetup_step](#) **step**
reflect easy-setup process step
- [iot_error_t](#) **err**
error status for each step
- char * **payload**
actual payload for each step

4.25.1 Detailed Description

Contains "easy-setup payload" data.

4.25.2 Field Documentation

4.25.2.1 **err** `iot_error_t err`

error status for each step

4.25.2.2 **payload** `char* payload`

actual payload for each step

4.25.2.3 **step** `enum iot_easystep_step step`

reflect easy-setup process step

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

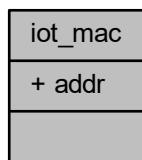
- st-device-sdk-c/src/include/[iot_main.h](#)

4.26 iot_mac Struct Reference

Contains "wifi mac" data.

```
#include <iot_bsp_wifi.h>
```

Collaboration diagram for iot_mac:



Data Fields

- `unsigned char addr [IOT_WIFI_MAX_BSSID_LEN]`
wifi mac address

4.26.1 Detailed Description

Contains "wifi mac" data.

4.26.2 Field Documentation

4.26.2.1 **addr** unsigned char addr[IOT_WIFI_MAX_BSSID_LEN]

wifi mac address

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

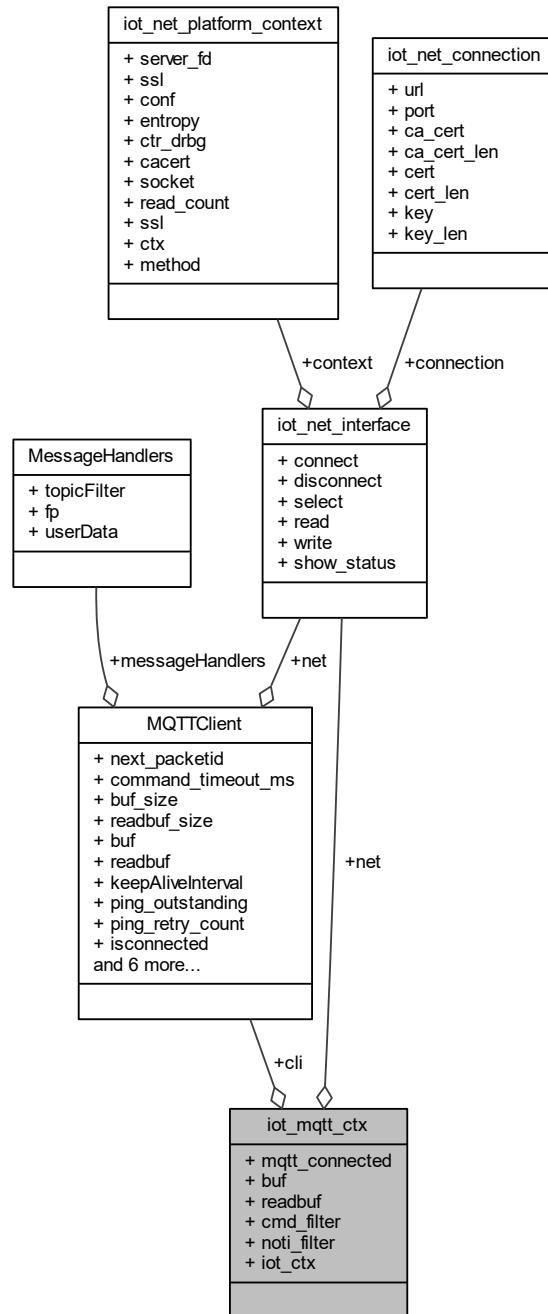
- st-device-sdk-c/src/include/bsp/[iot_bsp_wifi.h](#)

4.27 iot_mqtt_ctx Struct Reference

Contains "mqtt handling context" data.

```
#include <iot_main.h>
```

Collaboration diagram for iot_mqtt_ctx:



Data Fields

- `iot_net_interface_t net`
network management handle for mqtt
- `MQTTClient cli`
mqtt client handle for iot_core
- `bool mqtt_connected`

mqtt connected status

- unsigned char `buf` [`IOT_BUF_TX_SIZE`]
mqtt buffer for sending
- unsigned char `readbuf` [`IOT_BUF_RX_SIZE`]
mqtt buffer for receiving
- const char * `cmd_filter`
mqtt command topic filter string
- const char * `noti_filter`
mqtt notification topic filter string
- void * `iot_ctx`
iot main context ref. used for mqtt message callback

4.27.1 Detailed Description

Contains "mqtt handling context" data.

4.27.2 Field Documentation

4.27.2.1 `buf` unsigned char `buf`[`IOT_BUF_TX_SIZE`]

mqtt buffer for sending

4.27.2.2 `cli` `MQTTCClient` `cli`

mqtt client handle for `iot_core`

4.27.2.3 `cmd_filter` const char* `cmd_filter`

mqtt command topic filter string

4.27.2.4 `iot_ctx` void* `iot_ctx`

iot main context ref. used for mqtt message callback

4.27.2.5 `mqtt_connected` bool `mqtt_connected`

mqtt connected status

4.27.2.6 net iot_net_interface_t net

network management handle for mqtt

4.27.2.7 noti_filter const char* noti_filter

mqtt notification topic filter string

4.27.2.8 readbuf unsigned char readbuf[IOT_BUF_RX_SIZE]

mqtt buffer for receiving

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

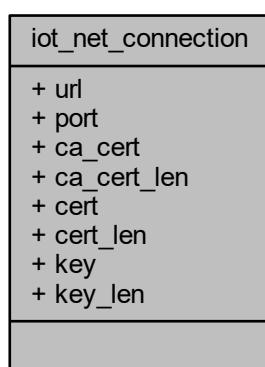
- st-device-sdk-c/src/include/[iot_main.h](#)

4.28 iot_net_connection Struct Reference

Contains server related information.

```
#include <iot_net.h>
```

Collaboration diagram for iot_net_connection:



Data Fields

- char * **url**
server address
- int **port**
server port
- const unsigned char * **ca_cert**
a pointer to a CA certificate
- unsigned int **ca_cert_len**
a size of CA certificate
- const unsigned char * **cert**
a pointer to a device certificate
- unsigned int **cert_len**
a size of device certificate
- const unsigned char * **key**
a pointer to a private key
- unsigned int **key_len**
a size of private key

4.28.1 Detailed Description

Contains server related information.

This structure has address, port and certificate of server.

4.28.2 Field Documentation

4.28.2.1 **ca_cert** const unsigned char* ca_cert

a pointer to a CA certificate

4.28.2.2 **ca_cert_len** unsigned int ca_cert_len

a size of CA certificate

4.28.2.3 **cert** const unsigned char* cert

a pointer to a device certificate

4.28.2.4 cert_len unsigned int cert_len

a size of device certificate

4.28.2.5 key const unsigned char* key

a pointer to a private key

4.28.2.6 key_len unsigned int key_len

a size of private key

4.28.2.7 port int port

server port

4.28.2.8 url char* url

server address

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

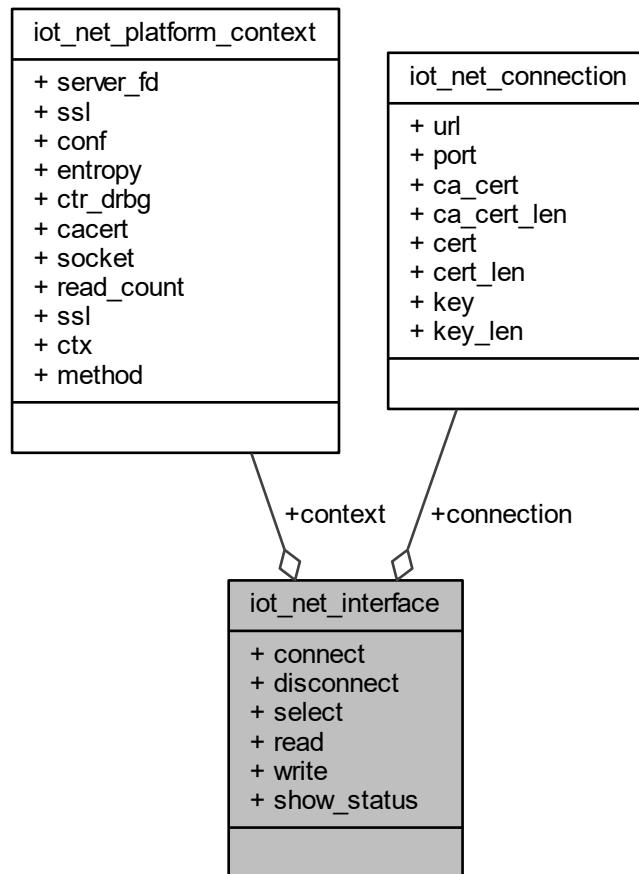
- st-device-sdk-c/src/include/iot_net.h

4.29 iot_net_interface Struct Reference

Contains "network management structure" data.

```
#include <iot_net.h>
```

Collaboration diagram for iot_net_interface:



Data Fields

- `iot_net_connection_t connection`
server connection informations
- `iot_net_platform_context_t context`
connect to server
- `iot_error_t(* connect)(iot_net_interface_t *)`
disconnect the server connection
- `void(* disconnect)(iot_net_interface_t *)`
check network socket status
- `int(* select)(iot_net_interface_t *, unsigned int)`
read from network
- `int(* read)(iot_net_interface_t *, unsigned char *, int, iot_os_timer)`
write to network
- `int(* write)(iot_net_interface_t *, unsigned char *, int, iot_os_timer)`
show socket status on console
- `void(* show_status)(iot_net_interface_t *)`

4.29.1 Detailed Description

Contains "network management structure" data.

4.29.2 Field Documentation

4.29.2.1 **connect** `iot_error_t (* connect) (iot_net_interface_t *)`

disconnect the server connection

4.29.2.2 **connection** `iot_net_connection_t connection`

server connection informations

<

contains connection context that depend to net library

4.29.2.3 **context** `iot_net_platform_context_t context`

connect to server

4.29.2.4 **disconnect** `void(* disconnect) (iot_net_interface_t *)`

check network socket status

4.29.2.5 **read** `int(* read) (iot_net_interface_t *, unsigned char *, int, iot_os_timer)`

write to network

4.29.2.6 **select** `int(* select) (iot_net_interface_t *, unsigned int)`

read from network

4.29.2.7 show_status void(* show_status) ([iot_net_interface_t](#) *)**4.29.2.8 write** int(* write) ([iot_net_interface_t](#) *, unsigned char *, int, [iot_os_timer](#))

show socket status on console

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

- st-device-sdk-c/src/include/[iot_net.h](#)

4.30 iot_net_platform_context Struct Reference

Contains connection context.

```
#include <iot_net_platform.h>
```

Collaboration diagram for iot_net_platform_context:

**Data Fields**

- mbedtls_net_context [server_fd](#)
 - mbedtls_ssl_context [ssl](#)
 - mbedtls_ssl_config [conf](#)
 - mbedtls_entropy_context [entropy](#)
 - mbedtls_ctr_drbg_context [ctr_drbg](#)
 - mbedtls_x509_crt [cacert](#)
 - int [socket](#)
- socket handle*

- int `read_count`
number of read data
- SSL * `ssl`
SSL Handle.
- SSL_CTX * `ctx`
set SSL context
- const SSL_METHOD * `method`
set SSL method

4.30.1 Detailed Description

Contains connection context.

4.30.2 Field Documentation

4.30.2.1 `cacert` mbedtls_x509_crt cacert

4.30.2.2 `conf` mbedtls_ssl_config conf

4.30.2.3 `ctr_drbg` mbedtls_ctr_drbg_context ctr_drbg

4.30.2.4 `ctx` SSL_CTX* ctx

set SSL context

4.30.2.5 `entropy` mbedtls_entropy_context entropy

4.30.2.6 `method` const SSL_METHOD* method

set SSL method

4.30.2.7 read_count int read_count

number of read data

4.30.2.8 server_fd mbedtls_net_context server_fd**4.30.2.9 socket** int socket

socket handle

4.30.2.10 ssl [1/2] SSL* ssl

SSL Handle.

4.30.2.11 ssl [2/2] mbedtls_ssl_context ssl

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

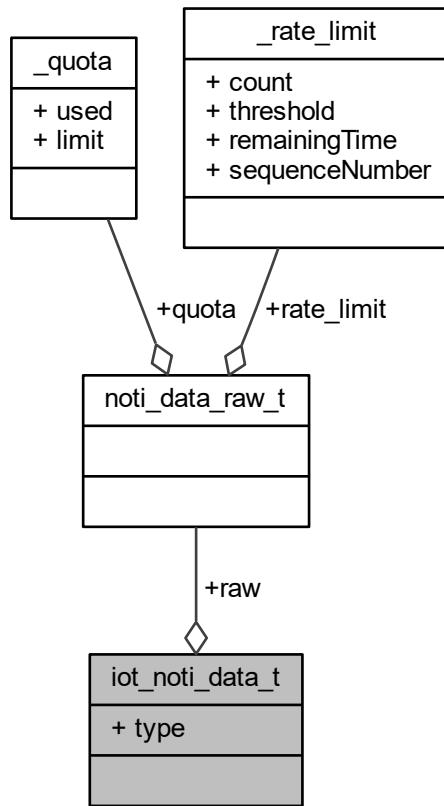
- st-device-sdk-c/src/port/net/mbedtls/iot_net_platform.h

4.31 iot_noti_data_t Struct Reference

Contains data for notification data.

```
#include <st_dev.h>
```

Collaboration diagram for iot_noti_data_t:



Data Fields

- `iot_noti_type_t type`
Type of notification's data.
- `noti_data_raw_t raw`
Raw data of each notification.

4.31.1 Detailed Description

Contains data for notification data.

4.31.2 Field Documentation

4.31.2.1 raw noti_data_raw_t raw

Raw data of each notification.

4.31.2.2 type iot_noti_type_t type

Type of notification's data.

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

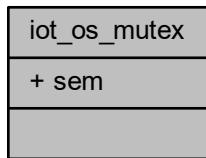
- st-device-sdk-c/src/include/[st_dev.h](#)

4.32 iot_os_mutex Struct Reference

Contains a mutex data.

```
#include <iot_os_util.h>
```

Collaboration diagram for iot_os_mutex:



Data Fields

- [iot_os_sem * sem](#)
semaphore

4.32.1 Detailed Description

Contains a mutex data.

4.32.2 Field Documentation

4.32.2.1 sem `iot_os_sem*` sem

semaphore

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

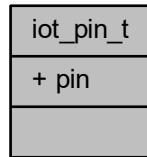
- st-device-sdk-c/src/include/os/[iot_os_util.h](#)

4.33 iot_pin_t Struct Reference

Contains a pin values for pin type onboarding process.

```
#include <st_dev.h>
```

Collaboration diagram for iot_pin_t:



Data Fields

- `unsigned char pin [8]`
actual pin values

4.33.1 Detailed Description

Contains a pin values for pin type onboarding process.

4.33.2 Field Documentation

4.33.2.1 pin `unsigned char pin[8]`

actual pin values

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

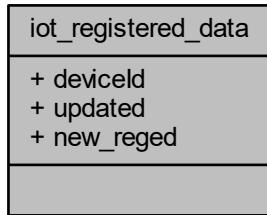
- st-device-sdk-c/src/include/[st_dev.h](#)

4.34 iot_registered_data Struct Reference

Contains "registration message" data.

```
#include <iot_main.h>
```

Collaboration diagram for iot_registered_data:



Data Fields

- char **deviceld** [IOT_REG_UUID_STR_LEN+1]
device Id, allocated from server
- bool **updated**
reflect getting device id
- bool **new_reged**
reflect that it is new registration process or not

4.34.1 Detailed Description

Contains "registration message" data.

4.34.2 Field Documentation

4.34.2.1 **deviceld** char deviceId[IOT_REG_UUID_STR_LEN+1]

device Id, allocated from server

4.34.2.2 **new_reged** bool new_reged

reflect that it is new registration process or not

4.34.2.3 **updated** bool updated

reflect getting device id

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

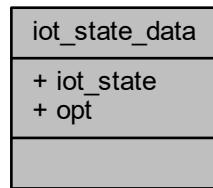
- st-device-sdk-c/src/include/[iot_main.h](#)

4.35 iot_state_data Struct Reference

Contains "iot core's main state" data.

```
#include <iot_main.h>
```

Collaboration diagram for iot_state_data:



Data Fields

- [iot_state_t](#) **iot_state**
current iot core's state
- int **opt**
additional option for each state

4.35.1 Detailed Description

Contains "iot core's main state" data.

4.35.2 Field Documentation

4.35.2.1 **iot_state** [iot_state_t](#) iot_state

current iot core's state

4.35.2.2 opt int opt

additional option for each state

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

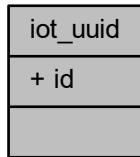
- st-device-sdk-c/src/include/[iot_main.h](#)

4.36 iot_uuid Struct Reference

Contains "uuid" data.

```
#include <iot_main.h>
```

Collaboration diagram for iot_uuid:



Data Fields

- unsigned char [id](#) [16]
actual uuid values, 16 octet

4.36.1 Detailed Description

Contains "uuid" data.

4.36.2 Field Documentation

4.36.2.1 id unsigned char id[16]

actual uuid values, 16 octet

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

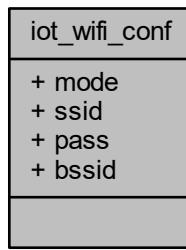
- st-device-sdk-c/src/include/[iot_main.h](#)

4.37 iot_wifi_conf Struct Reference

Contains a "wifi stack configuration" data.

```
#include <iot_bsp_wifi.h>
```

Collaboration diagram for iot_wifi_conf:



Data Fields

- `iot_wifi_mode_t mode`
wifi operation mode
- `char ssid [IOT_WIFI_MAX_SSID_LEN+1]`
wifi SSID string
- `char pass [IOT_WIFI_MAX_PASS_LEN+1]`
wifi password string
- `uint8_t bssid [IOT_WIFI_MAX_BSSID_LEN]`
wifi mac address

4.37.1 Detailed Description

Contains a "wifi stack configuration" data.

4.37.2 Field Documentation

4.37.2.1 `bssid uint8_t bssid[IOT_WIFI_MAX_BSSID_LEN]`

wifi mac address

4.37.2.2 mode `iot_wifi_mode_t mode`

wifi operation mode

4.37.2.3 pass `char pass[IOT_WIFI_MAX_PASS_LEN+1]`

wifi password string

4.37.2.4 ssid `char ssid[IOT_WIFI_MAX_SSID_LEN+1]`

wifi SSID string

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

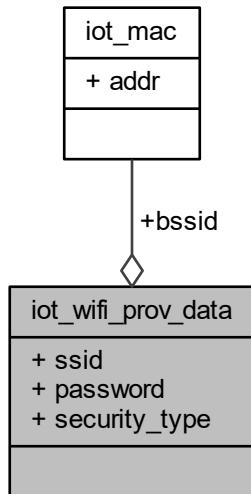
- st-device-sdk-c/src/include/bsp/[iot_bsp_wifi.h](#)

4.38 iot_wifi_prov_data Struct Reference

Contains "wifi provisioning" data.

```
#include <iot_main.h>
```

Collaboration diagram for iot_wifi_prov_data:



Data Fields

- char `ssid` [`IOT_WIFI_PROV_SSID_LEN`]
wifi SSID string
- char `password` [`IOT_WIFI_PROV_PASSWORD_LEN`]
wifi password string
- struct `iot_mac bssid`
wifi mac addresss struct
- `iot_wifi_auth_mode_t security_type`
wifi security type such as WEP, PSK2..

4.38.1 Detailed Description

Contains "wifi provisioning" data.

4.38.2 Field Documentation

4.38.2.1 `bssid` struct `iot_mac bssid`

wifi mac addresss struct

4.38.2.2 `password` char `password[IOT_WIFI_PROV_PASSWORD_LEN]`

wifi password string

4.38.2.3 `security_type` `iot_wifi_auth_mode_t security_type`

wifi security type such as WEP, PSK2..

4.38.2.4 `ssid` char `ssid[IOT_WIFI_PROV_SSID_LEN]`

wifi SSID string

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

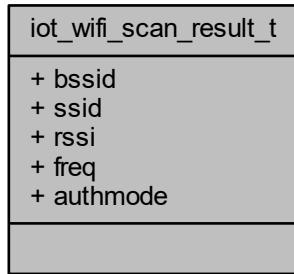
- st-device-sdk-c/src/include/`iot_main.h`

4.39 iot_wifi_scan_result_t Struct Reference

Contains a "wifi scan" data.

```
#include <iot_bsp_wifi.h>
```

Collaboration diagram for iot_wifi_scan_result_t:



Data Fields

- uint8_t **bssid** [IOT_WIFI_MAX_BSSID_LEN]
wifi mac address
- uint8_t **ssid** [IOT_WIFI_MAX_SSID_LEN+1]
wifi SSID string
- int8_t **rssi**
wifi signal strength
- uint16_t **freq**
wifi operation channel
- iot_wifi_auth_mode_t **authmode**
wifi authentication mode

4.39.1 Detailed Description

Contains a "wifi scan" data.

4.39.2 Field Documentation

4.39.2.1 authmode iot_wifi_auth_mode_t authmode

wifi authentication mode

4.39.2.2 bssid uint8_t bssid[IOT_WIFI_MAX_BSSID_LEN]

wifi mac address

4.39.2.3 freq uint16_t freq

wifi operation channel

4.39.2.4 rssi int8_t rssi

wifi signal strength

4.39.2.5 ssid uint8_t ssid[IOT_WIFI_MAX_SSID_LEN+1]

wifi SSID string

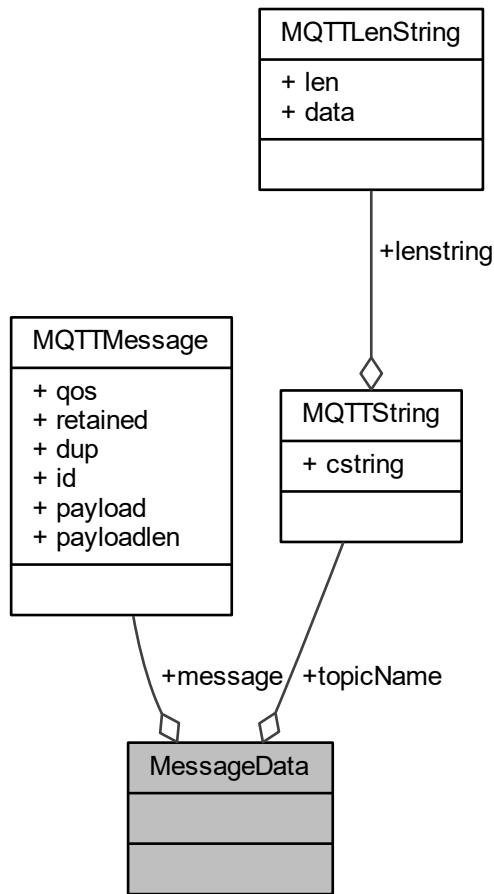
The documentation for this struct was generated from the following file:

- st-device-sdk-c/src/include/bsp/iot_bsp_wifi.h

4.40 MessageData Struct Reference

```
#include <iot_mqtt_client.h>
```

Collaboration diagram for MessageData:



Data Fields

- `MQTTMessage * message`
- `MQTTString * topicName`

4.40.1 Field Documentation

4.40.1.1 `message` `MQTTMessage*` `message`

4.40.1.2 **topicName** `MQTTString*` `topicName`

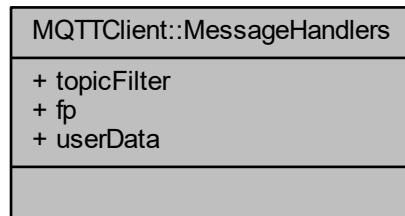
The documentation for this struct was generated from the following file:

- st-device-sdk-c/src/include/mqtt/[iot_mqtt_client.h](#)

4.41 MQTTClient::MessageHandlers Struct Reference

```
#include <iot_mqtt_client.h>
```

Collaboration diagram for MQTTClient::MessageHandlers:



Data Fields

- `const char * topicFilter`
- `void(* fp)(MessageData *, void *)`
- `void * userData`

4.41.1 Field Documentation

4.41.1.1 **fp** `void(* fp)(MessageData *, void *)`

4.41.1.2 **topicFilter** `const char* topicFilter`

4.41.1.3 **userData** `void* userData`

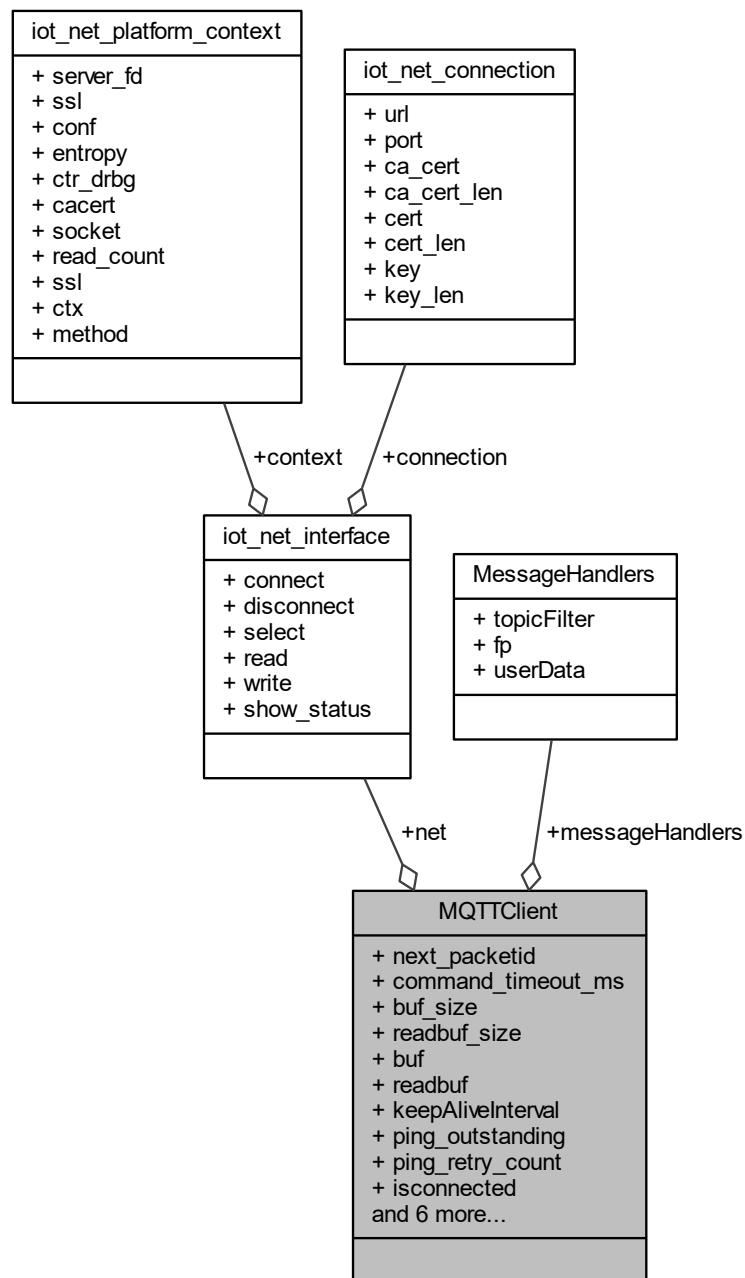
The documentation for this struct was generated from the following file:

- st-device-sdk-c/src/include/mqtt/[iot_mqtt_client.h](#)

4.42 MQTTClient Struct Reference

```
#include <iot_mqtt_client.h>
```

Collaboration diagram for MQTTClient:



Data Structures

- struct [MessageHandlers](#)

Data Fields

- unsigned int `next_packetid`
- unsigned int `command_timeout_ms`
- size_t `buf_size`
- size_t `readbuf_size`
- unsigned char * `buf`
- unsigned char * `readbuf`
- unsigned int `keepAliveInterval`
- char `ping_outstanding`
- int `ping_retry_count`
- int `isconnected`
- int `cleansession`
- struct `MQTTClient::MessageHandlers messageHandlers [MAX_MESSAGE_HANDLERS]`
- void(* `defaultMessageHandler`) (`MessageData` *, void *)
- void * `defaultUserData`
- `iot_net_interface_t * net`
- `iot_os_timer last_sent`
- `iot_os_timer last_received`
- `iot_os_timer ping_wait`

4.42.1 Field Documentation

4.42.1.1 buf unsigned char* `buf`

4.42.1.2 buf_size size_t `buf_size`

4.42.1.3 cleansession int `cleansession`

4.42.1.4 command_timeout_ms unsigned int `command_timeout_ms`

4.42.1.5 defaultMessageHandler void(* `defaultMessageHandler`) (`MessageData` *, void *)

4.42.1.6 defaultUserData void* `defaultUserData`

4.42.1.7 isconnected int isconnected

4.42.1.8 keepAliveInterval unsigned int keepAliveInterval

4.42.1.9 last_received iot_os_timer last_received

4.42.1.10 last_sent iot_os_timer last_sent

4.42.1.11 messageHandlers struct MQTTClient::MessageHandlers messageHandlers[MAX_MESSAGE_HANDLERS]

4.42.1.12 net iot_net_interface_t* net

4.42.1.13 next_packetid unsigned int next_packetid

4.42.1.14 ping_outstanding char ping_outstanding

4.42.1.15 ping_retry_count int ping_retry_count

4.42.1.16 ping_wait iot_os_timer ping_wait

4.42.1.17 readbuf unsigned char * readbuf

4.42.1.18 `readbuf_size` `size_t readbuf_size`

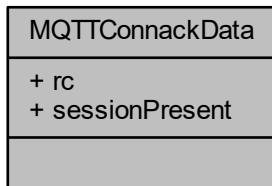
The documentation for this struct was generated from the following file:

- st-device-sdk-c/src/include/mqtt/[iot_mqtt_client.h](#)

4.43 MQTTConnackData Struct Reference

```
#include <iot_mqtt_client.h>
```

Collaboration diagram for MQTTConnackData:



Data Fields

- `unsigned char rc`
- `unsigned char sessionPresent`

4.43.1 Field Documentation

4.43.1.1 `rc` `unsigned char rc`

4.43.1.2 `sessionPresent` `unsigned char sessionPresent`

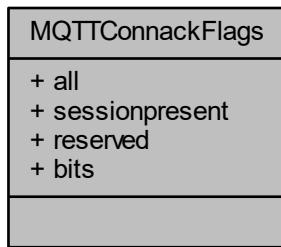
The documentation for this struct was generated from the following file:

- st-device-sdk-c/src/include/mqtt/[iot_mqtt_client.h](#)

4.44 MQTTConnackFlags Union Reference

```
#include <iot_mqtt_connect.h>
```

Collaboration diagram for MQTTConnackFlags:



Data Fields

- `unsigned char all`
- `struct {`
 - `unsigned int sessionpresent: 1`
 - `unsigned int reserved: 7``} bits`

4.44.1 Field Documentation

4.44.1.1 `all` `unsigned char all`

all connack flags

4.44.1.2 `bits` `struct { ... } bits`

4.44.1.3 `reserved` `unsigned int reserved`

unused

4.44.1.4 sessionpresent `unsigned int sessionpresent`

session present flag

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

- `st-device-sdk-c/src/include/mqtt/iot_mqtt_connect.h`

4.45 MQTTConnectFlags Union Reference

```
#include <iot_mqtt_connect.h>
```

Collaboration diagram for MQTTConnectFlags:



Data Fields

- `unsigned char all`
- `struct {`
 - `unsigned int: 1`
 - `unsigned int cleansession: 1`
 - `unsigned int will: 1`
 - `unsigned int willQoS: 2`
 - `unsigned int willRetain: 1`
 - `unsigned int password: 1`
 - `unsigned int username: 1``}` `bits`

4.45.1 Field Documentation

4.45.1.1 all unsigned char all

all connect flags

4.45.1.2 bits struct { ... } bits**4.45.1.3 cleansession** unsigned int cleansession

cleansession flag

4.45.1.4 int unsigned int

unused

4.45.1.5 password unsigned int password

3.1 password

4.45.1.6 username unsigned int username

3.1 user name

4.45.1.7 will unsigned int will

will flag

4.45.1.8 willQoS unsigned int willQoS

will QoS value

4.45.1.9 willRetain unsigned int willRetain

will retain setting

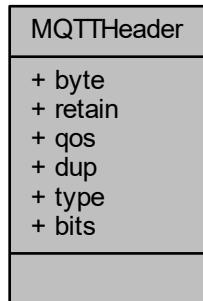
The documentation for this union was generated from the following file:

- st-device-sdk-c/src/include/mqtt/iot_mqtt_connect.h

4.46 MQTTHeader Union Reference

```
#include <iot_mqtt_packet.h>
```

Collaboration diagram for MQTTHeader:



Data Fields

- `unsigned char byte`
- `struct {`
 - `unsigned int retain: 1`
 - `unsigned int qos: 2`
 - `unsigned int dup: 1`
 - `unsigned int type: 4``}` **bits**

4.46.1 Detailed Description

Bitfields for the MQTT header byte.

4.46.2 Field Documentation

4.46.2.1 **bits** `struct { ... } bits`

4.46.2.2 **byte** `unsigned char byte`

the whole byte

4.46.2.3 dup unsigned int dup

DUP flag bit

4.46.2.4 qos unsigned int qos

QoS value, 0, 1 or 2

4.46.2.5 retain unsigned int retain

retained flag bit

4.46.2.6 type unsigned int type

message type nibble

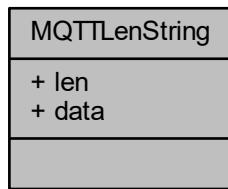
The documentation for this union was generated from the following file:

- st-device-sdk-c/src/include/mqtt/[iot_mqtt_packet.h](#)

4.47 MQTTLenString Struct Reference

```
#include <iot_mqtt_packet.h>
```

Collaboration diagram for MQTTLenString:



Data Fields

- int [len](#)
- char * [data](#)

4.47.1 Field Documentation

4.47.1.1 data char* data**4.47.1.2 len** int len

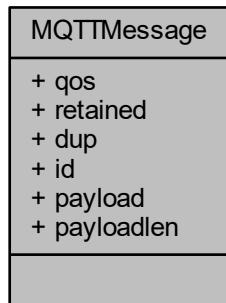
The documentation for this struct was generated from the following file:

- st-device-sdk-c/src/include/mqtt/[iot_mqtt_packet.h](#)

4.48 MQTTMessage Struct Reference

```
#include <iot_mqtt_client.h>
```

Collaboration diagram for MQTTMessage:



Data Fields

- enum [QoS qos](#)
- unsigned char [retained](#)
- unsigned char [dup](#)
- unsigned short [id](#)
- void * [payload](#)
- size_t [payloadlen](#)

4.48.1 Field Documentation

4.48.1.1 dup unsigned char dup

4.48.1.2 id unsigned short id

4.48.1.3 payload void* payload

4.48.1.4 payloadlen size_t payloadlen

4.48.1.5 qos enum QoS qos

4.48.1.6 retained unsigned char retained

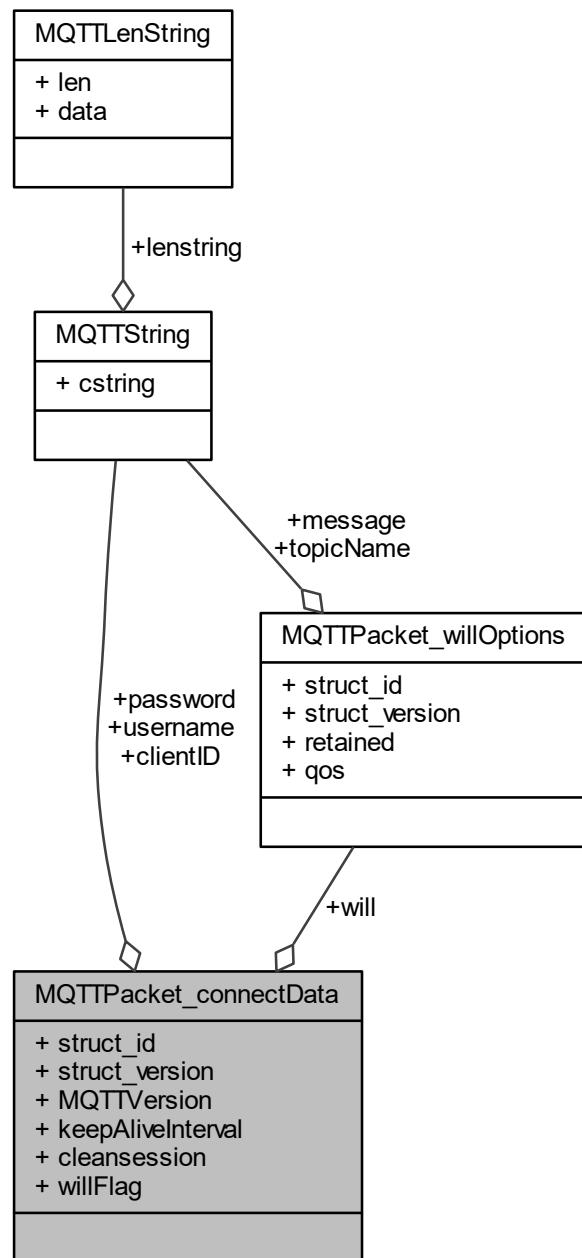
The documentation for this struct was generated from the following file:

- st-device-sdk-c/src/include/mqtt/[iot_mqtt_client.h](#)

4.49 MQTTPacket_connectData Struct Reference

```
#include <iot_mqtt_connect.h>
```

Collaboration diagram for MQTTPacket_connectData:



Data Fields

- char `struct_id` [4]
- int `struct_version`
- unsigned char `MQTTVersion`
- `MQTTString clientID`
- unsigned short `keepAliveInterval`

- unsigned char `cleansession`
- unsigned char `willFlag`
- `MQTTPacket_willOptions will`
- `MQTTString username`
- `MQTTString password`

4.49.1 Field Documentation

4.49.1.1 `cleansession` unsigned char `cleansession`

4.49.1.2 `clientID` `MQTTString clientID`

4.49.1.3 `keepAliveInterval` unsigned short `keepAliveInterval`

4.49.1.4 `MQTTVersion` unsigned char `MQTTVersion`

Version of MQTT to be used. 3 = 3.1 4 = 3.1.1

4.49.1.5 `password` `MQTTString password`

4.49.1.6 `struct_id` char `struct_id[4]`

The eyecatcher for this structure. must be MQTC.

4.49.1.7 `struct_version` int `struct_version`

The version number of this structure. Must be 0

4.49.1.8 `username` `MQTTString username`

4.49.1.9 `will` `MQTTPacket_willOptions will`

4.49.1.10 willFlag `unsigned char willFlag`

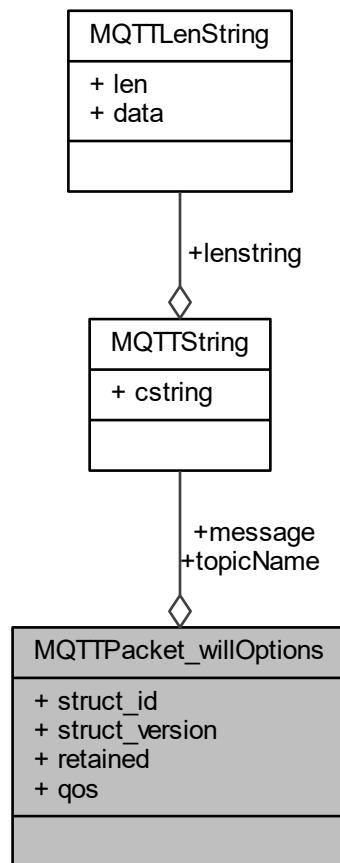
The documentation for this struct was generated from the following file:

- st-device-sdk-c/src/include/mqtt/iot_mqtt_connect.h

4.50 MQTTPacket_willOptions Struct Reference

```
#include <iot_mqtt_connect.h>
```

Collaboration diagram for MQTTPacket_willOptions:



Data Fields

- char `struct_id` [4]
- int `struct_version`
- `MQTTString topicName`
- `MQTTString message`
- unsigned char `retained`
- char `qos`

4.50.1 Detailed Description

Defines the MQTT "Last Will and Testament" (LWT) settings for the connect packet.

4.50.2 Field Documentation

4.50.2.1 **message** [MQTTString](#) message

The LWT payload.

4.50.2.2 **qos** char qos

The quality of service setting for the LWT message (see [MQTTAsync_message.qos](#) and [qos](#)).

4.50.2.3 **retained** unsigned char retained

The retained flag for the LWT message (see [MQTTAsync_message.retained](#)).

4.50.2.4 **struct_id** char struct_id[4]

The eyecatcher for this structure. must be MQTW.

4.50.2.5 **struct_version** int struct_version

The version number of this structure. Must be 0

4.50.2.6 **topicName** [MQTTString](#) topicName

The LWT topic to which the LWT message will be published.

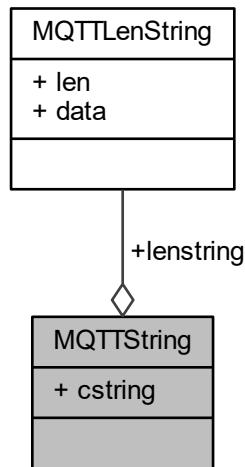
The documentation for this struct was generated from the following file:

- st-device-sdk-c/src/include/mqtt/[iot_mqtt_connect.h](#)

4.51 MQTTString Struct Reference

```
#include <iot_mqtt_packet.h>
```

Collaboration diagram for MQTTString:



Data Fields

- `char * cstring`
- `MQTTLenString lenstring`

4.51.1 Field Documentation

4.51.1.1 `cstring` `char* cstring`

4.51.1.2 `lenstring` `MQTTLenString lenstring`

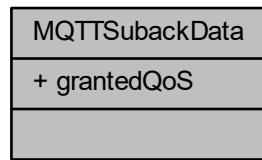
The documentation for this struct was generated from the following file:

- st-device-sdk-c/src/include/mqtt/[iot_mqtt_packet.h](#)

4.52 MQTTSubackData Struct Reference

```
#include <iot_mqtt_client.h>
```

Collaboration diagram for MQTTSubackData:



Data Fields

- enum [QoS](#) grantedQoS

4.52.1 Field Documentation

4.52.1.1 grantedQoS enum [QoS](#) grantedQoS

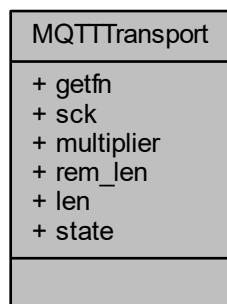
The documentation for this struct was generated from the following file:

- st-device-sdk-c/src/include/mqtt/[iot_mqtt_client.h](#)

4.53 MQTTTransport Struct Reference

```
#include <iot_mqtt_packet.h>
```

Collaboration diagram for MQTTTransport:



Data Fields

- int(* [getfn](#))(void *, unsigned char *, int)
- void * [sck](#)
- int [multiplier](#)
- int [rem_len](#)
- int [len](#)
- char [state](#)

4.53.1 Field Documentation

4.53.1.1 [getfn](#) int(* getfn) (void *, unsigned char *, int)

4.53.1.2 [len](#) int len

4.53.1.3 [multiplier](#) int multiplier

4.53.1.4 [rem_len](#) int rem_len

4.53.1.5 [sck](#) void* sck

4.53.1.6 [state](#) char state

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

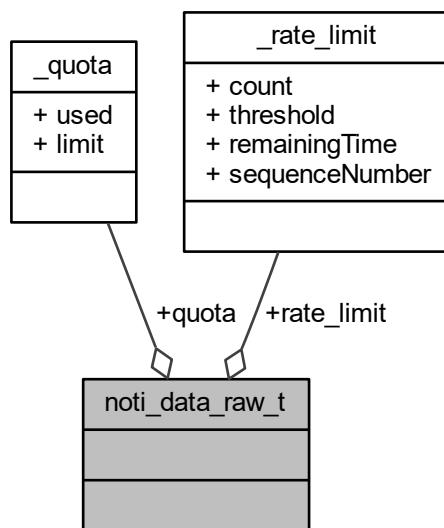
- st-device-sdk-c/src/include/mqtt/[iot_mqtt_packet.h](#)

4.54 noti_data_raw_t Union Reference

Contains data for raw data of each notification.

```
#include <st_dev.h>
```

Collaboration diagram for noti_data_raw_t:



Data Structures

- struct [_quota](#)
- struct [_rate_limit](#)

Data Fields

- struct [noti_data_raw_t::rate_limit](#) [rate_limit](#)
- struct [noti_data_raw_t::quota](#) [quota](#)

4.54.1 Detailed Description

Contains data for raw data of each notification.

4.54.2 Field Documentation

4.54.2.1 **quota** struct `noti_data_raw_t::_quota` quota

4.54.2.2 **rate_limit** struct `noti_data_raw_t::_rate_limit` rate_limit

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

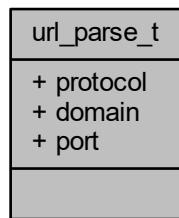
- st-device-sdk-c/src/include/[st_dev.h](#)

4.55 url_parse_t Struct Reference

Contains a "url parse" data.

```
#include <iot_util.h>
```

Collaboration diagram for url_parse_t:



Data Fields

- `char * protocol`
broker url's protocol part such as "ssl", "https"
- `char * domain`
broker url's domain part such as "test.example.com"
- `int port`
broker url's port number part such as 443, 8883'

4.55.1 Detailed Description

Contains a "url parse" data.

4.55.2 Field Documentation

4.55.2.1 domain char* domain

broker url's domain part such as "test.example.com"

4.55.2.2 port int port

broker url's port number part such as 443, 8883'

4.55.2.3 protocol char* protocol

broker url's protocol part such as "ssl", "https"

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

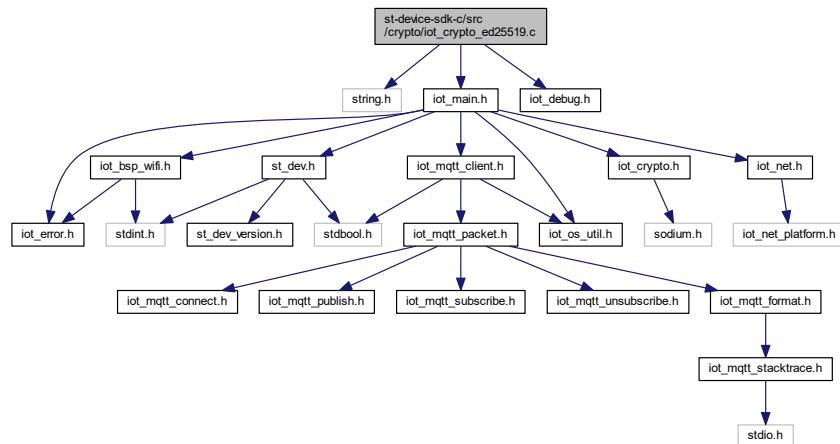
- st-device-sdk-c/src/include/iot_util.h

5 File Documentation

5.1 st-device-sdk-c/doc/mainpage.dox File Reference

5.2 st-device-sdk-c/src/crypto/iot_crypto_ed25519.c File Reference

```
#include <string.h>
#include "iot_main.h"
#include "iot_debug.h"
Include dependency graph for iot_crypto_ed25519.c:
```



Functions

- `iot_error_t iot_crypto_ed25519_init_keypair (iot_crypto_ed25519_keypair_t *kp)`
Prepare a buffer to store the ed25519 and curve25519 keypair.
- `void iot_crypto_ed25519_free_keypair (iot_crypto_ed25519_keypair_t *kp)`
Cleanup the buffer that allocated for keypair.
- `iot_error_t iot_crypto_ed25519_convert_keypair (iot_crypto_ed25519_keypair_t *kp)`
Converts an ed25519 keypair to an x25519 keypair.
- `iot_error_t iot_crypto_ed25519_convert_pubkey (unsigned char *ed25519_key, unsigned char *curve25519_key)`
Converts an ed25519 public key to an x25519 public key.
- `iot_error_t iot_crypto_ed25519_convert_seckey (unsigned char *ed25519_key, unsigned char *curve25519_key)`
Converts an ed25519 secret key to an x25519 secret key.

5.2.1 Function Documentation

5.2.1.1 `iot_crypto_ed25519_convert_keypair()` `iot_error_t iot_crypto_ed25519_convert_keypair (iot_crypto_ed25519_keypair_t * kp)`

Converts an ed25519 keypair to an x25519 keypair.

Parameters

in	<code>kp</code>	a pointer to a structure of the keypair buffers ed25519 keypair should be prepared
----	-----------------	--

Return values

<code>IOT_ERROR_NONE</code>	the buffer is sucessfully allocated
<code>IOT_ERROR_CRYPTO_ED_KEY_CONVERT</code>	failed to convert to x25519

5.2.1.2 `iot_crypto_ed25519_convert_pubkey()` `iot_error_t iot_crypto_ed25519_convert_pubkey (unsigned char * ed25519_key, unsigned char * curve25519_key)`

Converts an ed25519 public key to an x25519 public key.

Parameters

in	<code>ed25519_key</code>	a pointer to a public key buffer
out	<code>curve25519_key</code>	a pointer to a buffer to store converted x25519 public key

Return values

<i>IOT_ERROR_NONE</i>	the buffer is sucessfully allocated
<i>IOT_ERROR_CRYPTO_ED_KEY_CONVERT</i>	failed to convert to x25519

5.2.1.3 iot_crypto_ed25519_convert_seckey() *iot_error_t* iot_crypto_ed25519_convert_seckey (

```
    unsigned char * ed25519_key,
    unsigned char * curve25519_key )
```

Converts an ed25519 secret key to an x25519 secret key.

Parameters

in	<i>ed25519_key</i>	a pointer to a secret key buffer
out	<i>curve25519_key</i>	a pointer to a buffer to store converted x25519 secret key

Return values

<i>IOT_ERROR_NONE</i>	the buffer is sucessfully allocated
<i>IOT_ERROR_CRYPTO_ED_KEY_CONVERT</i>	failed to convert to x25519

5.2.1.4 iot_crypto_ed25519_free_keypair() *void* iot_crypto_ed25519_free_keypair (

```
    iot_crypto_ed25519_keypair_t * kp )
```

Cleanup the buffer that allocated for keypair.

Parameters

in	<i>kp</i>	a pointer to a structure of the keypair to cleanup
----	-----------	--

5.2.1.5 iot_crypto_ed25519_init_keypair() *iot_error_t* iot_crypto_ed25519_init_keypair (

```
    iot_crypto_ed25519_keypair_t * kp )
```

Prepare a buffer to store the ed25519 and curve25519 keypair.

Parameters

in	<i>kp</i>	a pointer to a structure of the keypair buffers
----	-----------	---

Return values

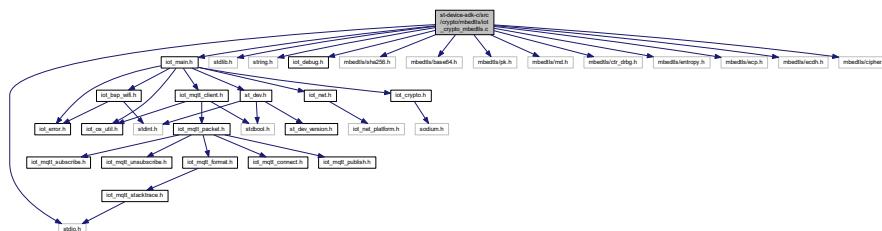
Return values

<i>IOT_ERROR_NONE</i>	the buffer is sucessfully allocated
<i>IOT_ERROR_MEM_ALLOC</i>	failed to alloc buffer for keypair

5.3 st-device-sdk-c/src/crypto/mbedtls/iot_crypto_mbedtls.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "iot_main.h"
#include "iot_debug.h"
#include "mbedtls/sha256.h"
#include "mbedtls/base64.h"
#include "mbedtls/pk.h"
#include "mbedtls/md.h"
#include "mbedtls/ctr_drbg.h"
#include "mbedtls/entropy.h"
#include "mbedtls/ecp.h"
#include "mbedtls/ecdh.h"
#include "mbedtls/cipher.h"

Include dependency graph for iot_crypto_mbedtls.c:
```



Functions

- **iot_error_t iot_crypto_base64_encode** (unsigned char *src, size_t src_len, unsigned char *dst, size_t dst_len, size_t *out_len)

Encode a string as a base64 string.
- **iot_error_t iot_crypto_base64_decode** (unsigned char *src, size_t src_len, unsigned char *dst, size_t dst_len, size_t *out_len)

Decode a base64 string as a string.
- **iot_error_t iot_crypto_base64_encode_urlsafe** (unsigned char *src, size_t src_len, unsigned char *dst, size_t dst_len, size_t *out_len)

Encode a string as a urlsafe base64 string.
- **iot_error_t iot_crypto_base64_decode_urlsafe** (unsigned char *src, size_t src_len, unsigned char *dst, size_t dst_len, size_t *out_len)

Decode a urlsafe base64 string as a string.
- **iot_error_t iot_crypto_sha256** (unsigned char *src, size_t src_len, unsigned char *dst)

Generate a digest by sha256 hash.

- `iot_error_t iot_crypto_ecdh_gen_master_secret` (`unsigned char *master, size_t mlen, iot_crypto_ecdh_params_t *params`)
Generate a master secret.
- `size_t iot_crypto_cipher_get_align_size` (`iot_crypto_cipher_type_t type, size_t size`)
Calculate the required align size that contains the encrypted data for the input size.
- `iot_error_t iot_crypto_cipher_aes` (`iot_crypto_cipher_info_t *info, unsigned char *input, size_t ilen, unsigned char *out, size_t *olen, size_t osize`)
Generic encryption/decryption function.

5.3.1 Function Documentation

5.3.1.1 iot_crypto_base64_decode() `iot_error_t iot_crypto_base64_decode (`
 `unsigned char * src,`
 `size_t src_len,`
 `unsigned char * dst,`
 `size_t dst_len,`
 `size_t * out_len)`

Decode a base64 string as a string.

Parameters

in	<code>src</code>	a pointer to a buffer to decode
in	<code>src_len</code>	the size of buffer pointed by src in bytes
out	<code>dst</code>	a pointer to a buffer to store base64 string
in	<code>dst_len</code>	the size of buffer pointed by dst in bytes
out	<code>out_len</code>	the bytes written to dst

Return values

<code>IOT_ERROR_NONE</code>	the string is sucessfully decoded
<code>IOT_ERROR_CRYPTO_BASE64</code>	failed to decode the string

5.3.1.2 iot_crypto_base64_decode_urlsafe() `iot_error_t iot_crypto_base64_decode_urlsafe (`
 `unsigned char * src,`
 `size_t src_len,`
 `unsigned char * dst,`
 `size_t dst_len,`
 `size_t * out_len)`

Decode a urlsafe base64 string as a string.

Parameters

in	<code>src</code>	a pointer to a buffer to decode
----	------------------	---------------------------------

Parameters

in	<i>src_len</i>	the size of buffer pointed by src in bytes
out	<i>dst</i>	a pointer to a buffer to store base64 string
in	<i>dst_len</i>	the size of buffer pointed by dst in bytes
out	<i>out_len</i>	the bytes written to dst

Return values

<i>IOT_ERROR_NONE</i>	the string is sucessfully decoded
<i>IOT_ERROR_CRYPTO_BASE64</i>	failed to decode the string
<i>IOT_ERROR_CRYPTO_BASE64_URLSAFE</i>	failed to encode the string as urlsafe

5.3.1.3 iot_crypto_base64_encode() `iot_error_t iot_crypto_base64_encode (`

```
    unsigned char * src,
    size_t src_len,
    unsigned char * dst,
    size_t dst_len,
    size_t * out_len )
```

Encode a string as a base64 string.

Parameters

in	<i>src</i>	a pointer to a buffer to encode
in	<i>src_len</i>	the size of buffer pointed by src in bytes
out	<i>dst</i>	a pointer to a buffer to store base64 string
in	<i>dst_len</i>	the size of buffer pointed by dst in bytes
out	<i>out_len</i>	the bytes written to dst

Return values

<i>IOT_ERROR_NONE</i>	the string is sucessfully encoded
<i>IOT_ERROR_CRYPTO_BASE64</i>	failed to encode the string

5.3.1.4 iot_crypto_base64_encode_urlsafe() `iot_error_t iot_crypto_base64_encode_urlsafe (`

```
    unsigned char * src,
    size_t src_len,
    unsigned char * dst,
    size_t dst_len,
    size_t * out_len )
```

Encode a string as a urlsafe base64 string.

This function replaces url unsafe characters ('+', '/') to url safe character ('-', '_')

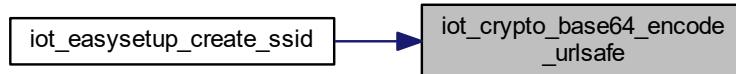
Parameters

in	<i>src</i>	a pointer to a buffer to encode
in	<i>src_len</i>	the size of buffer pointed by <i>src</i> in bytes
out	<i>dst</i>	a pointer to a buffer to store base64 string
in	<i>dst_len</i>	the size of buffer pointed by <i>dst</i> in bytes
out	<i>out_len</i>	the bytes written to <i>dst</i>

Return values

<i>IOT_ERROR_NONE</i>	the string is sucessfully encoded
<i>IOT_ERROR_CRYPTO_BASE64</i>	failed to encode the string in 3rd lib
<i>IOT_ERROR_CRYPTO_BASE64_URLSAFE</i>	failed to encode the string as urlsafe

Here is the caller graph for this function:



```

5.3.1.5 iot_crypto_cipher_aes() iot_error_t iot_crypto_cipher_aes (
    iot_crypto_cipher_info_t * info,
    unsigned char * input,
    size_t ilen,
    unsigned char * out,
    size_t * olen,
    size_t osize )

```

Generic encryption/decryption function.

Supported cipher algorithms is AES-256 CBC mode

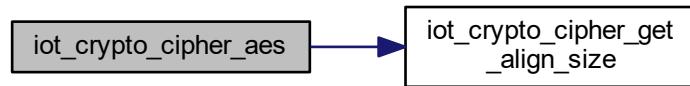
Parameters

in	<i>info</i>	a pointer to a buffer containing cipher context
in	<i>input</i>	a pointer to a buffer to encrypt/decrypt
in	<i>ilen</i>	the size of buffer pointed by <i>input</i> in bytes
out	<i>out</i>	a pointer to a buffer to store a signature
out	<i>olen</i>	the bytes written to <i>out</i>
in	<i>osize</i>	the size of buffer pointed by <i>out</i> in bytes

Return values

<i>IOT_ERROR_NONE</i>	encryption/decryption is a success
<i>IOT_ERROR_CRYPTO_CIPHER_UNKNOWN_TYPE</i>	not supported cipher is requested.
<i>IOT_ERROR_CRYPTO_CIPHER</i>	failed while encrypting/decrypting

Here is the call graph for this function:



5.3.1.6 *iot_crypto_cipher_get_align_size()* *size_t iot_crypto_cipher_get_align_size (*
iot_crypto_cipher_type_t type,
size_t size)

Calculate the required align size that contains the encrypted data for the input size.

the size of data to encrypt should be aligned when AES is used because AES is operated based on block.

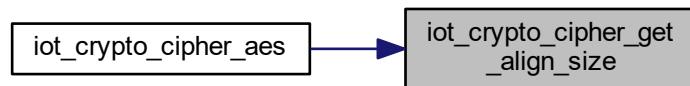
Parameters

in	<i>type</i>	cipher type
in	<i>size</i>	the size of data to encrypt

Return values

<i>return</i>	the aligned size
<i>return</i>	'0' when error occurs

Here is the caller graph for this function:



```
5.3.1.7 iot_crypto_ecdh_gen_master_secret() iot_error_t iot_crypto_ecdh_gen_master_secret (
    unsigned char * master,
    size_t mlen,
    iot_crypto_ecdh_params_t * params )
```

Generate a master secret.

This function generates a master secret by combining the shared key and the hashed token. The shared key is generated by ECDH using device's x25519 private key and peer's x25519 public key. The hashed token is shared by peer.

Parameters

out	<i>master</i>	a pointer to a buffer to store master secret
in	<i>mlen</i>	the size of buffer pointed by master in bytes
in	<i>params</i>	a pointer to a buffer containing the private key, public key and hashed token to generate master secret

Return values

<i>IOT_ERROR_NONE</i>	the master secret is sucessfully generated
<i>IOT_ERROR_INVALID_ARGS</i>	params has invalid data
<i>IOT_ERROR_MEM_ALLOC</i>	failed to alloc buffer for shared key
<i>IOT_ERROR_CRYPTO_PK_ECDH</i>	failed in ECDH processing
<i>IOT_ERROR_CRYPTO_ED_KEY_CONVERT</i>	failed to convert to x25519

```
5.3.1.8 iot_crypto_sha256() iot_error_t iot_crypto_sha256 (
    unsigned char * src,
    size_t src_len,
    unsigned char * dst )
```

Generate a digest by sha256 hash.

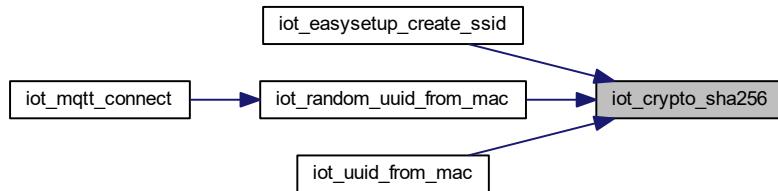
Parameters

in	<i>src</i>	a pointer to a buffer to generate a digest
in	<i>src_len</i>	the size of buffer pointed by src in bytes
out	<i>dst</i>	a pointer to a buffer to store a digest

Return values

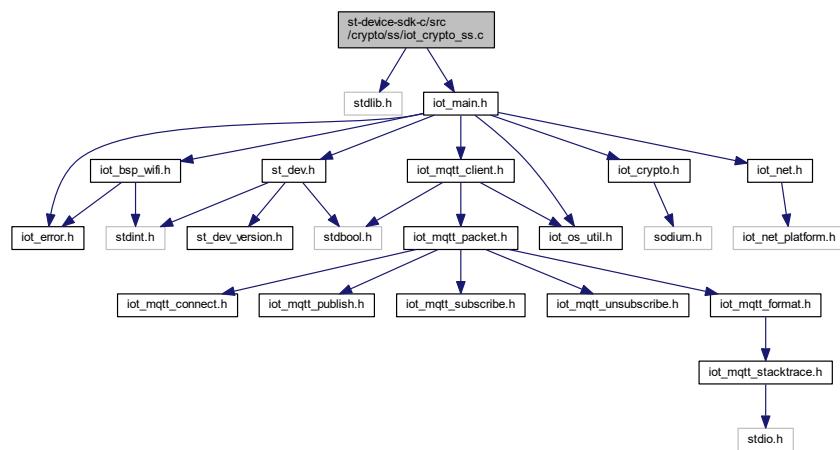
<i>IOT_ERROR_NONE</i>	a digest is sucessfully generated
<i>IOT_ERROR_CRYPTO_SHA256</i>	failed to generate a digest

Here is the caller graph for this function:



5.4 st-device-sdk-c/src/crypto/ss/iot_crypto_ss.c File Reference

```
#include <stdlib.h>
#include "iot_main.h"
Include dependency graph for iot_crypto_ss.c:
```



Functions

- [iot_error_t iot_crypto_ss_encrypt](#) (unsigned char *input, size_t ilen, unsigned char **output, size_t *olen)
Encryption of the input data.
- [iot_error_t iot_crypto_ss_decrypt](#) (unsigned char *input, size_t ilen, unsigned char **output, size_t *olen)
Decryption of the input data.

5.4.1 Function Documentation

```
5.4.1.1 iot_crypto_ss_decrypt() iot_error_t iot_crypto_ss_decrypt (
    unsigned char * input,
    size_t ilen,
    unsigned char ** output,
    size_t * olen )
```

Decryption of the input data.

The decryption key is generated from device unique value

Parameters

in	<i>input</i>	a pointer to a buffer to decrypt
in	<i>ilen</i>	the size of buffer pointed by input in bytes
out	<i>output</i>	a pointer of pointer to a buffer to store the decrypted data
out	<i>olen</i>	the bytes written to output buffer

Return values

<i>IOT_ERROR_NONE</i>	decryption is success
<i>IOT_ERROR_MEM_ALLOC</i>	mem alloc failed for output buffer
<i>IOT_ERROR_CRYPTO_CIPHER_ALIGN</i>	failed to get align size of the input size for output buffer
<i>IOT_ERROR_CRYPTO_CIPHER_UNKNOWN_TYPE</i>	cipher is requested with not supported algorithm
<i>IOT_ERROR_CRYPTO_CIPHER</i>	cipher operation is failed
<i>IOT_ERROR_CRYPTO_SS_KDF</i>	failed during derivate the key

```
5.4.1.2 iot_crypto_ss_encrypt() iot_error_t iot_crypto_ss_encrypt (
    unsigned char * input,
    size_t ilen,
    unsigned char ** output,
    size_t * olen )
```

Encryption of the input data.

The encryption key is generated from device unique value

Parameters

in	<i>input</i>	a pointer to a buffer to encrypt
in	<i>ilen</i>	the size of buffer pointed by input in bytes
out	<i>output</i>	a pointer of pointer to a buffer to store the encrypted data
out	<i>olen</i>	the bytes written to output buffer

Return values

<i>IOT_ERROR_NONE</i>	encryption is success
<i>IOT_ERROR_MEM_ALLOC</i>	mem alloc failed for output buffer
<i>IOT_ERROR_CRYPTO_CIPHER_ALIGN</i>	failed to get align size of the input size for output buffer

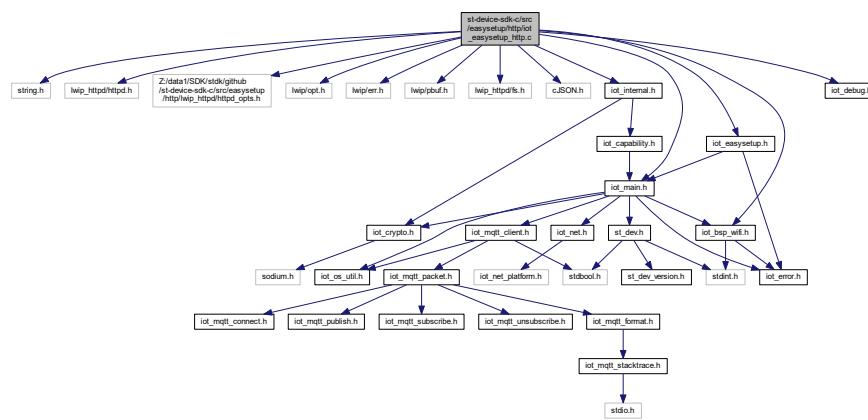
Return values

<code>IOT_ERROR_CRYPTO_CIPHER</code>	cipher operation is failed
<code>IOT_ERROR_CRYPTO_SS_KDF</code>	failed during derivate the key

5.5 st-device-sdk-c/src/easysetup/http/iot_easysetup_http.c File Reference

```
#include <string.h>
#include "lwip_httpd/httpd.h"
#include "lwip_httpd/fs.h"
#include "cJSON.h"
#include "iot_main.h"
#include "iot_internal.h"
#include "iot_debug.h"
#include "iot_easysetup.h"
#include "iot_bsp_wifi.h"

Include dependency graph for iot_easysetup_http.c:
```

**Macros**

- `#define MAX_PAYLOAD_LENGTH 1024`
- `#define ARRAY_SIZE(x) (int)(sizeof(x)/sizeof(x[0]))`
- `#define LWIP_HTTPD_POST_MAX_PAYLOAD_LEN 512`

Functions

- `iot_error_t _iot_easysetup_gen_get_payload (struct iot_context *ctx, const char *cmd, char **out_payload)`
http GET method payload handler
- `iot_error_t _iot_easysetup_gen_post_payload (struct iot_context *ctx, const char *cmd, char *in_payload, char **out_payload)`
http POST method payload handler
- `err_t httpd_post_begin (void *connection, const char *uri, const char *http_request, u16_t http_request_len, int content_len, char *response_uri, u16_t response_uri_len, u8_t *post_auto_wnd)`
- `err_t httpd_post_receive_data (void *connection, struct pbuf *p)`
- `void httpd_post_finished (void *connection, char *response_uri, u16_t response_uri_len)`

- void `httpd_cgi_handler` (const char *uri, int iNumParams, char **pcParam, char **pcValue, void *connection_state)
- int `fs_open_custom` (struct fs_file *file, const char *name)
- void `fs_close_custom` (struct fs_file *file)
- void * `fs_state_init` (struct fs_file *file, const char *name)
- void `fs_state_free` (struct fs_file *file, void *state)
- `iot_error_t iot_easysetup_init` (struct `iot_context` *ctx)

Start eayssetup device-to-device sequence.
- void `iot_easysetup_deinit` (struct `iot_context` *ctx)

Stop eayssetup device-to-device sequence.

Variables

- const char * `post_cgi_cmds` []
- const char * `get_cgi_cmds` []

5.5.1 Macro Definition Documentation

5.5.1.1 ARRAY_SIZE #define ARRAY_SIZE(
 x) (int)(sizeof(x)/sizeof(x[0]))

5.5.1.2 LWIP_HTTPD_POST_MAX_PAYLOAD_LEN #define LWIP_HTTPD_POST_MAX_PAYLOAD_LEN 512

5.5.1.3 MAX_PAYLOAD_LENGTH #define MAX_PAYLOAD_LENGTH 1024

5.5.2 Function Documentation

5.5.2.1 _iot_easysetup_gen_get_payload() `iot_error_t _iot_easysetup_gen_get_payload (`
 `struct iot_context * ctx,`
 `const char * cmd,`
 `char ** out_payload)`

http GET method payload handler

This function handle GET method cgi request

Parameters

in	<code>ctx</code>	<code>iot_context</code> handle
in	<code>cmd</code>	GET method uri
out	<code>out_payload</code>	output payload buffer for GET method. caller has full responsibility to free this memory.

Returns

```
iot_error_t
```

Return values

<i>IOT_ERROR_NONE</i>	success
-----------------------	---------

5.5.2.2 *_iot_easysetup_gen_post_payload()* *iot_error_t* *_iot_easysetup_gen_post_payload* (

```
    struct iot_context * ctx,
    const char * cmd,
    char * in_payload,
    char ** out_payload )
```

http POST method payload handler

This function handle POST method cgi request

Parameters

in	<i>ctx</i>	<i>iot_context</i> handle
in	<i>cmd</i>	POST method uri
in	<i>in_payload</i>	client updated payload via POST method has given to here. this shouldn't be freed inside of this function.
out	<i>out_payload</i>	output payload for http response. caller has full responsibility to free this memory

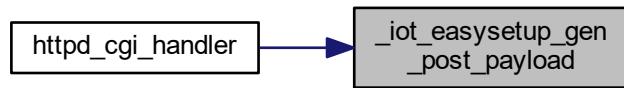
Returns

```
iot_error_t
```

Return values

<i>IOT_ERROR_NONE</i>	success
-----------------------	---------

Here is the caller graph for this function:



5.5.2.3 `fs_close_custom()` void fs_close_custom (struct fs_file * file)

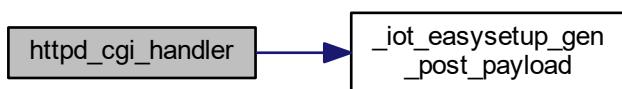
5.5.2.4 `fs_open_custom()` int fs_open_custom (struct fs_file * file, const char * name)

5.5.2.5 `fs_state_free()` void fs_state_free (struct fs_file * file, void * state)

5.5.2.6 `fs_state_init()` void* fs_state_init (struct fs_file * file, const char * name)

5.5.2.7 `httpd_cgi_handler()` void httpd_cgi_handler (const char * uri, int iNumParams, char ** pcParam, char ** pcValue, void * connection_state)

Here is the call graph for this function:



5.5.2.8 `httpd_post_begin()` err_t httpd_post_begin (void * connection, const char * uri, const char * http_request, u16_t http_request_len, int content_len, char * response_uri, u16_t response_uri_len, u8_t * post_auto_wnd)

5.5.2.9 httpd_post_finished() void httpd_post_finished (void * connection, char * response_uri, u16_t response_uri_len)

5.5.2.10 httpd_post_receive_data() err_t httpd_post_receive_data (void * connection, struct pbuf * p)

5.5.2.11 iot_easysetup_deinit() void iot_easysetup_deinit (struct iot_context * ctx)

Stop eayssetup device-to-device sequence.

This function stops httpd working

Parameters

in	ctx	iot_context handle
----	-----	--------------------

Returns

void

5.5.2.12 iot_easysetup_init() iot_error_t iot_easysetup_init (struct iot_context * ctx)

Start eayssetup device-to-device sequence.

This function makes wifi mode as soft-ap and starts httpd

Parameters

in	ctx	iot_context handle
----	-----	--------------------

Returns

iot_error_t

Return values

IOT_ERROR_NONE	success
IOT_ERROR_UNINITIALIZED	error

5.5.3 Variable Documentation

5.5.3.1 get_cgi_cmds const char* get_cgi_cmds[]

Initial value:

```
=
{
    IOT_ES_URI_GET_DEVICEINFO,
    IOT_ES_URI_GET_WIFISCANINFO,
    IOT_ES_URI_GET_POST_RESPONSE,
    IOT_ES_URI_GET_LOGS_SYSTEMINFO,
    IOT_ES_URI_GET_LOGS_DUMP,
}
```

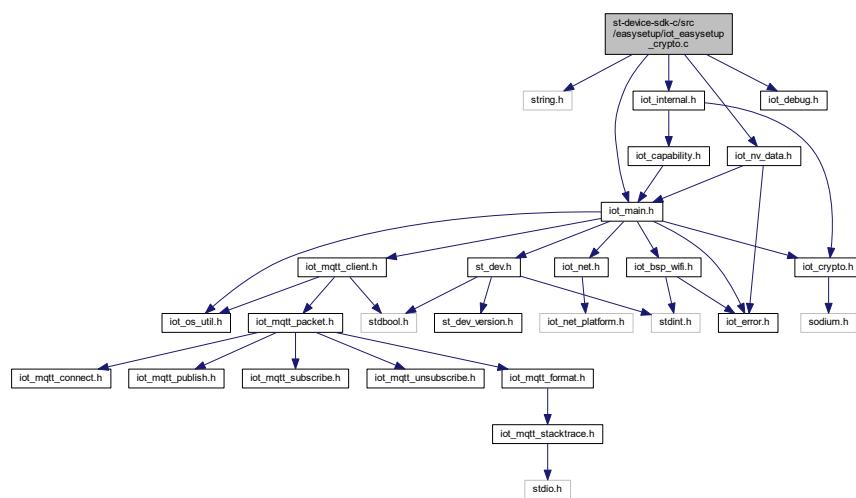
5.5.3.2 post_cgi_cmds const char* post_cgi_cmds[]

Initial value:

```
=
{
    IOT_ES_URI_POST_KEYINFO,
    IOT_ES_URI_POST_CONFIRMINFO,
    IOT_ES_URI_POST_CONFIRM,
    IOT_ES_URI_POST_WIFIPROVISIONINGINFO,
    IOT_ES_URI_POST_SETUPCOMPLETE,
    IOT_ES_URI_POST_LOGS,
}
```

5.6 st-device-sdk-c/src/easysetup/iot_easysetup_crypto.c File Reference

```
#include <string.h>
#include "iot_main.h"
#include "iot_internal.h"
#include "iot_nv_data.h"
#include "iot_debug.h"
Include dependency graph for iot_easysetup_crypto.c:
```



Functions

- void `iot_es_crypto_init_pk` (`iot_crypto_pk_info_t` *`pk_info`, `iot_crypto_pk_type_t` `type`)
initialize the buffer for pubkey information
- `iot_error_t iot_es_crypto_load_pk` (`iot_crypto_pk_info_t` *`pk_info`)
easy setup crypto to load pubkey
- void `iot_es_crypto_free_pk` (`iot_crypto_pk_info_t` *`pk_info`)
easy setup crypto to free pubkey

5.6.1 Function Documentation

5.6.1.1 `iot_es_crypto_free_pk()` void `iot_es_crypto_free_pk` (
 `iot_crypto_pk_info_t` * `pk_info`)

easy setup crypto to free pubkey

this function frees the loaded pubkey information

Parameters

in	<code>pk_info</code>	loaded pubkey information pointer
----	----------------------	-----------------------------------

5.6.1.2 `iot_es_crypto_init_pk()` void `iot_es_crypto_init_pk` (
 `iot_crypto_pk_info_t` * `pk_info`,
 `iot_crypto_pk_type_t` `type`)

initialize the buffer for pubkey information

this function uses to initialize

Parameters

in	<code>pk_info</code>	pubkey information structure to initialize
in	<code>type</code>	pubkey type, RSA or ED25519

5.6.1.3 `iot_es_crypto_load_pk()` `iot_error_t iot_es_crypto_load_pk` (
 `iot_crypto_pk_info_t` * `pk_info`)

easy setup crypto to load pubkey

this function loads pubkey information

Parameters

in	<i>pk_info</i>	pubkey information pointer for loading
----	----------------	--

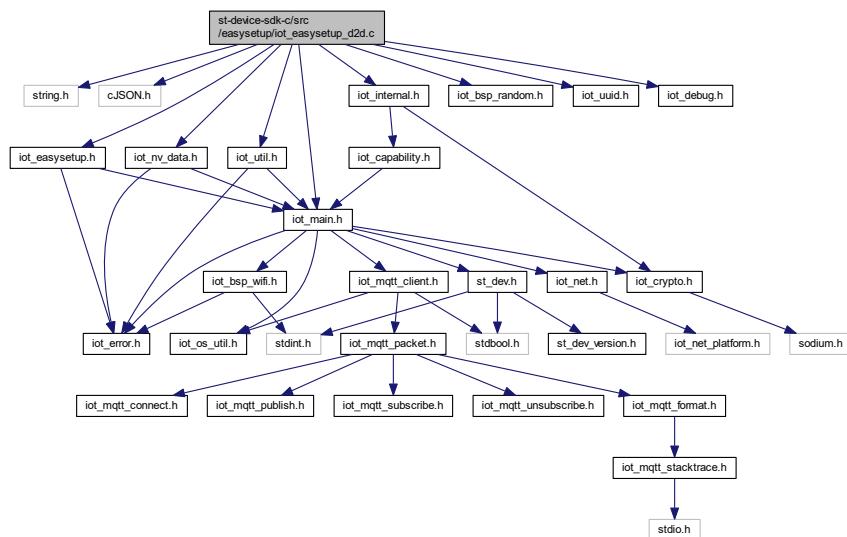
Return values

<i>IOT_ERROR_NONE</i>	success.
-----------------------	----------

5.7 st-device-sdk-c/src/easysetup/iot_easysetup_d2d.c File Reference

```
#include <string.h>
#include "cJSON.h"
#include "iot_main.h"
#include "iot_bsp_random.h"
#include "iot_easysetup.h"
#include "iot_internal.h"
#include "iot_nv_data.h"
#include "iot_util.h"
#include "iot_uuid.h"
#include "iot_debug.h"

Include dependency graph for iot_easysetup_d2d.c:
```



Macros

- #define HASH_SIZE (4)
- #define PIN_SIZE 8
- #define MAC_ADDR_BUFFER_SIZE 20
- #define URL_BUFFER_SIZE 64
- #define WIFIINFO_BUFFER_SIZE 20
- #define ES_CONFIRM_MAX_DELAY 10000
- #define JSON_H cJSON

Functions

- `iot_error_t iot_easysetup_create_ssid (struct iot_devconf_prov_data *devconf, char *ssid, size_t ssid_len)`
Create E4 type SSID.
- `void st_conn_ownership_confirm (IOT_CTX *iot_ctx, bool confirm)`
easysetup user confirm report function
- `iot_error_t iot_easysetup_request_handler (struct iot_context *ctx, struct iot_easysetup_payload request)`
easysetup cgi request handler

5.7.1 Macro Definition Documentation

5.7.1.1 ES_CONFIRM_MAX_DELAY `#define ES_CONFIRM_MAX_DELAY 10000`

5.7.1.2 HASH_SIZE `#define HASH_SIZE (4)`

5.7.1.3 JSON_H `#define JSON_H cJSON`

5.7.1.4 MAC_ADDR_BUFFER_SIZE `#define MAC_ADDR_BUFFER_SIZE 20`

5.7.1.5 PIN_SIZE `#define PIN_SIZE 8`

5.7.1.6 URL_BUFFER_SIZE `#define URL_BUFFER_SIZE 64`

5.7.1.7 WIFIINFO_BUFFER_SIZE `#define WIFIINFO_BUFFER_SIZE 20`

5.7.2 Function Documentation

5.7.2.1 iot_easysetup_create_ssid() `iot_error_t iot_easysetup_create_ssid (`
 `struct iot_devconf_prov_data * devconf,`
 `char * ssid,`
 `size_t ssid_len)`

Create E4 type SSID.

This function create E4 type soft-ap SSID for this device

Parameters

in	<i>devconf</i>	things static information
out	<i>ssid</i>	created ssid
in	<i>ssid_len</i>	ssid buffer length including null termination

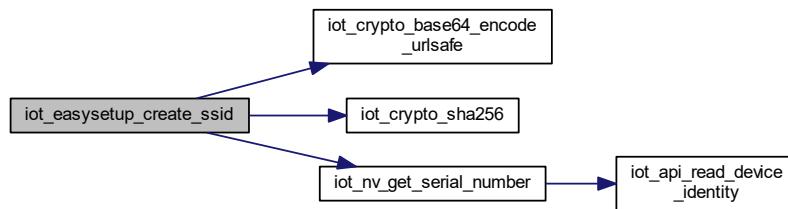
Returns

iot_state_t

Return values

<i>IOT_ERROR_NONE</i>	success
-----------------------	---------

Here is the call graph for this function:



5.7.2.2 iot_easysetup_request_handler() `iot_error_t iot_easysetup_request_handler (struct iot_context * ctx, struct iot_easysetup_payload request)`

easysetup cgi request handler

This function runs from iot-task by executing actual cgi payload manipulation.
result will be transferred to httpd task (iT) as easysetup response queue parameter.

Parameters

in	<i>ctx</i>	<code>iot_context</code> handle
in	<i>request</i>	easysetup payload as input

Returns

iot_error_t

Return values

<code>IOT_ERROR_NONE</code>	success
-----------------------------	---------

5.7.2.3 `st_conn_ownership_confirm()` `void st_conn_ownership_confirm (`
 `IOT_CTX * iot_ctx,`
 `bool confirm)`

easysetup user confirm report function

This function reports the user confirmation to easysetup

Parameters

in	<code>iot_ctx</code>	<code>iot_context</code> handle generated by <code>iot_main_init()</code>
in	<code>confirm</code>	user confirmation result

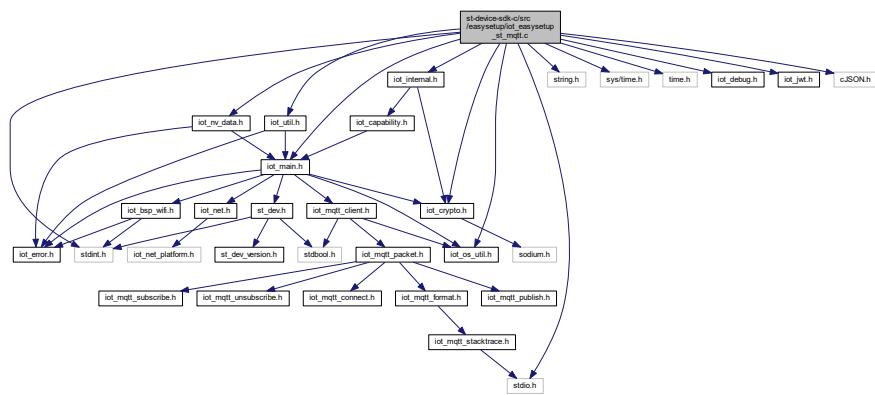
Here is the call graph for this function:



5.8 st-device-sdk-c/src/easysetup/iot_easysetup_st_mqtt.c File Reference

```
#include <stdint.h>
#include <stdio.h>
#include <string.h>
#include <sys/time.h>
#include <time.h>
#include "iot_main.h"
#include "iot_internal.h"
#include "iot_util.h"
#include "iot_nv_data.h"
#include "iot_debug.h"
#include "iot_jwt.h"
#include "iot_crypto.h"
#include "iot_os_util.h"
#include "cJSON.h"
```

Include dependency graph for iot_easysetup_st_mqtt.c:



Functions

- `iot_error_t iot_es_connect (struct iot_context *ctx, int conn_type)`
easy setup connect
- `iot_error_t iot_es_disconnect (struct iot_context *ctx, int conn_type)`
easy setup disconnect

5.8.1 Function Documentation

5.8.1.1 iot_es_connect() `iot_error_t iot_es_connect (`

```
    struct iot_context * ctx,
    int conn_type )
```

easy setup connect

this function tries to connect server for registration or communication process

Parameters

in	<code>ctx</code>	iot-core context
in	<code>conn_type</code>	set connection type. registration or communication with server

Return values

<code>IOT_ERROR_NONE</code>	success.
-----------------------------	----------

5.8.1.2 iot_es_disconnect() `iot_error_t iot_es_disconnect (`

```
struct iot_context * ctx,
int conn_type )
```

easy setup disconnect

this function tries to disconnect server for registration or communication process

Parameters

in	<i>ctx</i>	iot-core context
in	<i>conn_type</i>	set connection type. registration or communication with server

Return values

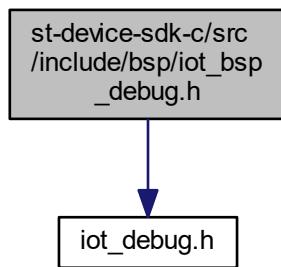
<i>IOT_ERROR_NONE</i>	success.
-----------------------	----------

Here is the caller graph for this function:



5.9 st-device-sdk-c/src/include/bsp/iot_esp_debug.h File Reference

```
#include <iot_debug.h>
Include dependency graph for iot_esp_debug.h:
```



Functions

- void `iot_bsp_debug` (`iot_debug_level_t` level, const char *tag, const char *fmt,...)
Write message into the log.
- void `iot_bsp_debug_check_heap` (const char *tag, const char *func, const int line, const char *fmt,...)
Check memory(heap) status.

5.9.1 Function Documentation

5.9.1.1 `iot_bsp_debug()` void `iot_bsp_debug` (
`iot_debug_level_t` level,
 const char * tag,
 const char * fmt,
 ...)

Write message into the log.

This function is not intended to be used directly. Instead, use one of IOT_ERROR, IOT_WARN, IOT_INFO, IOT_DEBUG macros.

Parameters

in	<i>level</i>	log level
		<ul style="list-style-type: none"> • IOT_DEBUG_LEVEL_NONE • IOT_DEBUG_LEVEL_ERROR • IOT_DEBUG_LEVEL_WARN • IOT_DEBUG_LEVEL_INFO • IOT_DEBUG_LEVEL_DEBUG
in	<i>func</i>	caller function
in	<i>line</i>	line number
in	<i>fmt</i>	user friendly string

5.9.1.2 `iot_bsp_debug_check_heap()` void `iot_bsp_debug_check_heap` (
 const char * tag,
 const char * func,
 const int line,
 const char * fmt,
 ...)

Check memory(heap) status.

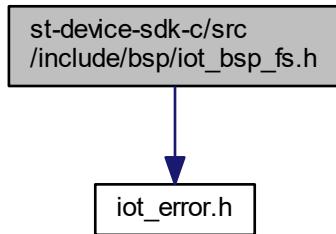
This function check memory(heap) status

Parameters

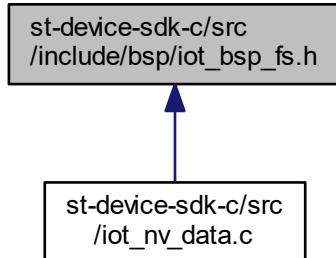
in	<i>tag</i>	tag
in	<i>func</i>	caller function
in	<i>line</i>	line number
in	<i>fmt</i>	user friendly string

5.10 st-device-sdk-c/src/include/bsp/iot_bsp_fs.h File Reference

```
#include "iot_error.h"
Include dependency graph for iot_bsp_fs.h:
```



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



Data Structures

- struct `iot_bsp_fs_handle_t`

iot_bsp_fs_open_mode_t

file system open mode.

- enum `iot_bsp_fs_open_mode_t` { `FS_READONLY`, `FS_READWRITE` }
- `iot_error_t iot_bsp_fs_init ()`
Initialize a file system.
- `iot_error_t iot_bsp_fs_deinit ()`
Deinitialize a file system.
- `iot_error_t iot_bsp_fs_open (const char *filename, iot_bsp_fs_open_mode_t mode, iot_bsp_fs_handle_t *handle)`
Open a file.
- `iot_error_t iot_bsp_fs_open_from_stnv (const char *filename, iot_bsp_fs_handle_t *handle)`
Open a file from stnv partition.
- `iot_error_t iot_bsp_fs_read (iot_bsp_fs_handle_t handle, char *buffer, unsigned int length)`
Read a file.
- `iot_error_t iot_bsp_fs_write (iot_bsp_fs_handle_t handle, const char *data, unsigned int length)`
Write a file.
- `iot_error_t iot_bsp_fs_close (iot_bsp_fs_handle_t handle)`
Close a file.
- `iot_error_t iot_bsp_fs_remove (const char *filename)`
Remove a file.

5.10.1 Enumeration Type Documentation**5.10.1.1 iot_bsp_fs_open_mode_t enum iot_bsp_fs_open_mode_t**

Enumerator

<code>FS_READONLY</code>	
<code>FS_READWRITE</code>	

5.10.2 Function Documentation**5.10.2.1 iot_bsp_fs_close() iot_error_t iot_bsp_fs_close (iot_bsp_fs_handle_t handle)**

Close a file.

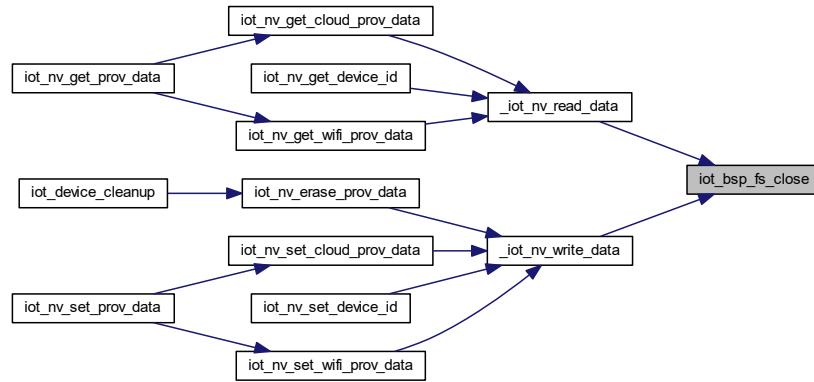
Parameters

<code>in</code>	<code>handle</code>	This is <code>iot_bsp_fs_handle_t</code> handle from <code>iot_bsp_fs_open()</code> .
-----------------	---------------------	---

Return values

<i>IOT_ERROR_NONE</i>	File close successful.
-----------------------	------------------------

Here is the caller graph for this function:



5.10.2.2 `iot_bsp_fs_deinit()` *iot_error_t iot_bsp_fs_deinit ()*

Deinitialize a file system.

Return values

<i>IOT_ERROR_NONE</i>	File-system deinit successful.
<i>IOT_ERROR_DEINIT_FAIL</i>	File-system deinit failed.

Here is the caller graph for this function:



5.10.2.3 `iot_bsp_fs_init()` *iot_error_t iot_bsp_fs_init ()*

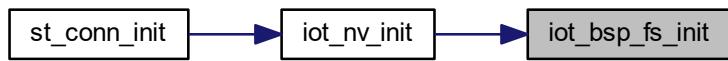
Initialize a file system.

This function initializes the file-system of the device. You must call this function before using file-management function (open, read, write, etc.)

Return values

<i>IOT_ERROR_NONE</i>	File-system init successful.
<i>IOT_ERROR_INIT_FAIL</i>	File-system init failed.

Here is the caller graph for this function:



5.10.2.4 *iot_bsp_fs_open()* *iot_error_t* *iot_bsp_fs_open* (
 const char * *filename*,
iot_bsp_fs_open_mode_t *mode*,
iot_bsp_fs_handle_t * *handle*)

Open a file.

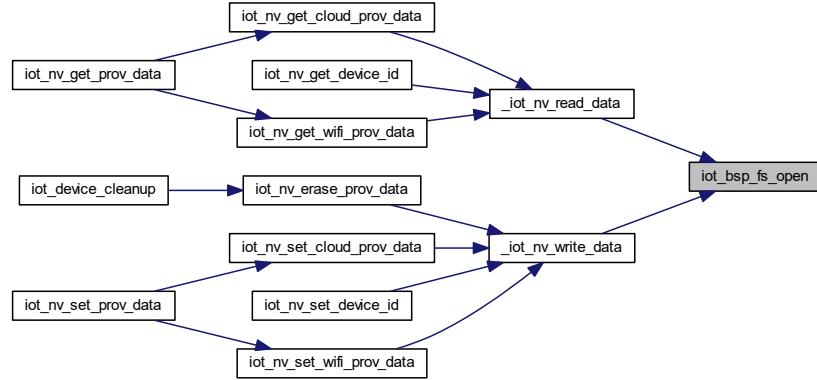
Parameters

in	<i>filename</i>	File name.
in	<i>mode</i>	File open mode. FS_READONLY or FS_READWRITE.
out	<i>handle</i>	A pointer to the file handle.

Return values

<i>IOT_ERROR_NONE</i>	File open successful.
<i>IOT_ERROR_FS_OPEN_FAIL</i>	File open failed.

Here is the caller graph for this function:



5.10.2.5 iot_bsp_fs_open_from_stnv() `iot_error_t iot_bsp_fs_open_from_stnv (const char * filename, iot_bsp_fs_handle_t * handle)`

Open a file from stnv partition.

This function will return the read-only file-system handle. You can access the stnv partition's file though this handle.

Parameters

in	<i>filename</i>	File name.
out	<i>handle</i>	A pointer to the file handle.

Return values

<i>IOT_ERROR_NONE</i>	File open successful.
<i>IOT_ERROR_FS_OPEN_FAIL</i>	File open failed.

5.10.2.6 iot_bsp_fs_read() `iot_error_t iot_bsp_fs_read (iot_bsp_fs_handle_t handle, char * buffer, unsigned int length)`

Read a file.

Parameters

in	<i>handle</i>	This is <code>iot_bsp_fs_handle_t</code> handle from <code>iot_bsp_fs_open()</code> .
----	---------------	---

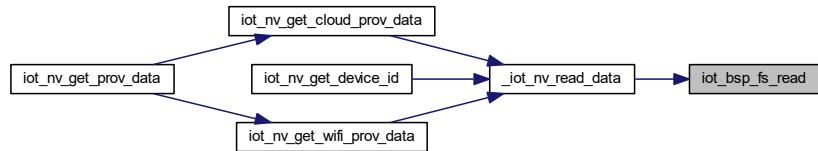
Parameters

out	<i>buffer</i>	A pointer to buffer array to store the read data from the file.
in	<i>length</i>	The size of buffer.

Return values

<i>IOT_ERROR_NONE</i>	File read successful.
<i>IOT_ERROR_FS_READ_FAIL</i>	File read failed.
<i>IOT_ERROR_FS_NO_FILE</i>	No file.

Here is the caller graph for this function:



5.10.2.7 `iot_bsp_fs_remove()` `iot_error_t iot_bsp_fs_remove (`
`const char * filename)`

Remove a file.

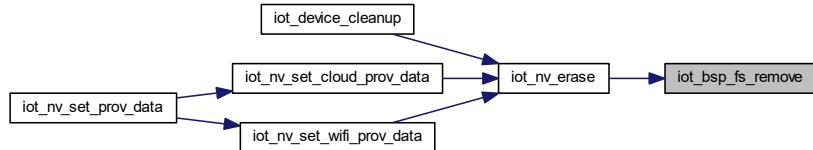
Parameters

in	<i>filename</i>	File name.
----	-----------------	------------

Return values

<i>IOT_ERROR_NONE</i>	File remove successful.
<i>IOT_ERROR_INVALID_ARGS</i>	Invalid filename
<i>IOT_ERROR_FS_REMOVE_FAIL</i>	File remove failed.

Here is the caller graph for this function:



5.10.2.8 iot_bsp_fs_write()

```

iot_error_t iot_bsp_fs_write (
    iot_bsp_fs_handle_t handle,
    const char * data,
    unsigned int length )
  
```

Write a file.

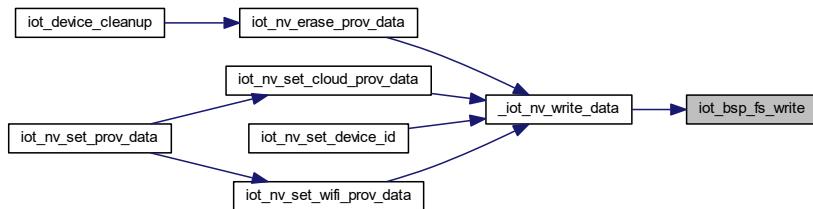
Parameters

in	<i>handle</i>	This is iot_bsp_fs_handle_t handle from iot_bsp_fs_open() .
in	<i>data</i>	A pointer to data array to write to the file.
in	<i>length</i>	The size of data.

Return values

<i>IOT_ERROR_NONE</i>	File write successful.
<i>IOT_ERROR_FS_WRITE_FAIL</i>	File write failed.

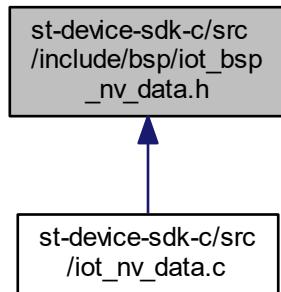
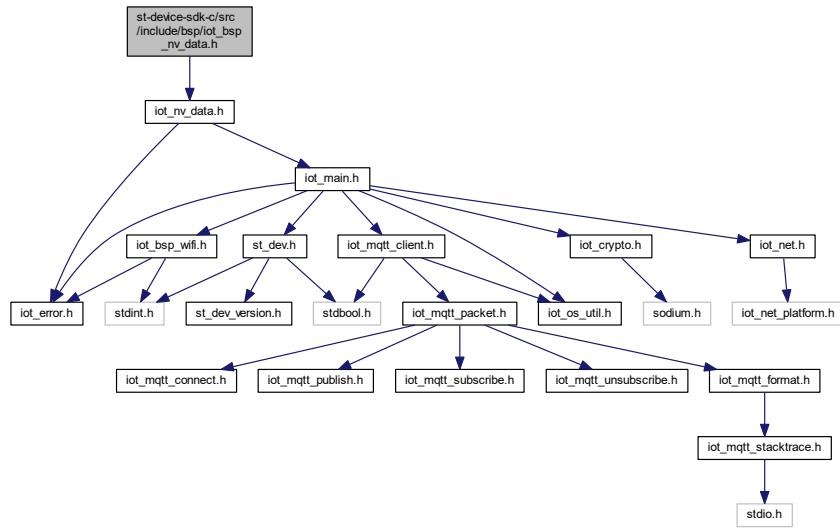
Here is the caller graph for this function:



5.11 st-device-sdk-c/src/include/bsp/iot_bsp_nv_data.h File Reference

```
#include "iot_nv_data.h"
```

Include dependency graph for iot_bsp_nv_data.h:



Functions

- const char * [iot_bsp_nv_get_data_path \(iot_nvd_t nv_type\)](#)
Get a nv data path.

5.11.1 Function Documentation

```
5.11.1.1 iot_bsp_nv_get_data_path() const char* iot_bsp_nv_get_data_path (
    iot_nvd_t nv_type )
```

Get a nv data path.

This function will return the nv data path. The detail of data path depends on the file-system type. ex : espressif nvs-system - nvs key ("nvlItemKey") linux file-system - file path ("/stnv/nvlItemKey")

Parameters

in	<i>nv_type</i>	The type of nv data. declaration is in "iot_nv_data.h"
----	----------------	--

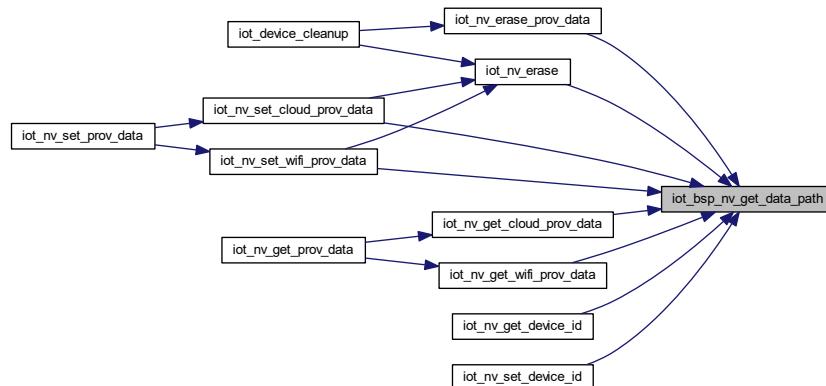
Returns

nv data path.

See also

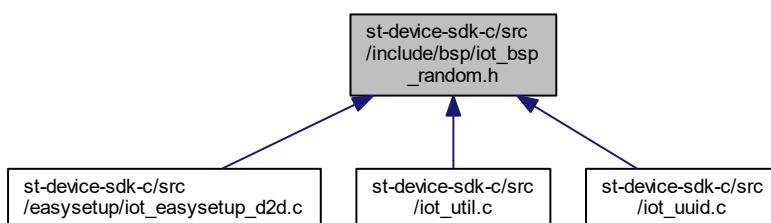
[iot_nv_data.h](#)

Here is the caller graph for this function:



5.12 st-device-sdk-c/src/include/bsp/iot_bsp_random.h File Reference

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



Functions

- `unsigned int iot_bsp_random ()`

Generate random number.

5.12.1 Function Documentation

5.12.1.1 `iot_bsp_random()` `unsigned int iot_bsp_random ()`

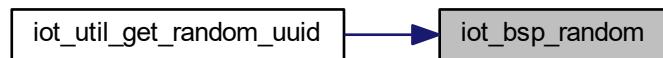
Generate random number.

This function generates and returns a random number.

Returns

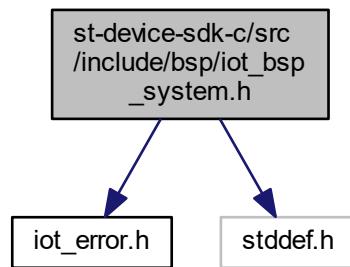
`unsigned int` random number

Here is the caller graph for this function:

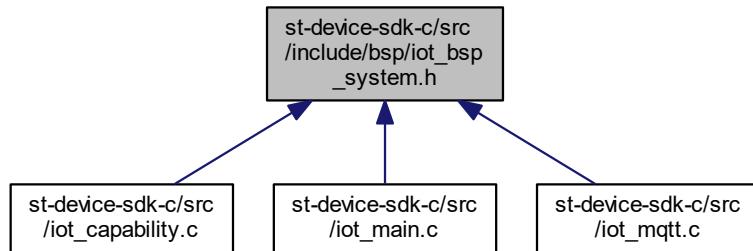


5.13 st-device-sdk-c/src/include/bsp/iot_bsp_system.h File Reference

```
#include "iot_error.h"
#include <stddef.h>
Include dependency graph for iot_bsp_system.h:
```



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



Macros

- `#define IOT_REBOOT() iot_esp_system_reboot()`
Macro to use iot_esp_system_reboot function.
- `#define IOT_POWEROFF() iot_esp_system_poweroff()`
Macro to use iot_esp_system_poweroff function.

Functions

- `void iot_esp_system_reboot ()`
Restart system.
- `void iot_esp_system_poweroff ()`
Shutdown system.
- `iot_error_t iot_esp_system_get_time_in_sec (char *buf, unsigned int buf_len)`
Get system time in second.
- `iot_error_t iot_esp_system_set_time_in_sec (const char *time_in_sec)`
Set system time in second.
- `iot_error_t iot_esp_system_get_uniqueid (unsigned char **uid, size_t *olen)`
Get device unique value.

5.13.1 Macro Definition Documentation

5.13.1.1 IOT_POWEROFF `#define IOT_POWEROFF() iot_esp_system_poweroff()`

Macro to use iot_esp_system_poweroff function.

5.13.1.2 IOT_REBOOT `#define IOT_REBOOT() iot_esp_system_reboot()`

Macro to use iot_esp_system_reboot function.

5.13.2 Function Documentation

5.13.2.1 iot_bsp_system_get_time_in_sec() `iot_error_t iot_bsp_system_get_time_in_sec (`

```
    char * buf,
    unsigned int buf_len )
```

Get system time in second.

Parameters

out	<i>buf</i>	A pointer to data array to store the system time in second.
in	<i>buf_len</i>	The length of buffer.

Return values

<i>IOT_ERROR_NONE</i>	Set time successful.
<i>IOT_ERROR_INVALID_ARGS</i>	Invalid argument.

5.13.2.2 iot_bsp_system_get_uniqueid() `iot_error_t iot_bsp_system_get_uniqueid (`

```
    unsigned char ** uid,
    size_t * olen )
```

Get device unique value.

The source of unique value

Parameters

out	<i>uid</i>	a pointer of pointer to a unique id buffer
out	<i>olen</i>	the bytes written to unique id buffer

Returns

`iot_error_t`

Return values

<i>IOT_ERROR_NONE</i>	success
<i>IOT_ERROR_MEM_ALLOC</i>	alloc failed for unique id buffer
<i>IOT_ERROR_NOT_IMPLEMENTED</i>	no way to make unique id

5.13.2.3 iot_bsp_system_poweroff() `void iot_bsp_system_poweroff ()`

Shutdown system.

This function shuts down system.

5.13.2.4 iot_bsp_system_reboot() `void iot_bsp_system_reboot ()`

Restart system.

This function restarts system.

5.13.2.5 iot_bsp_system_set_time_in_sec() `iot_error_t iot_bsp_system_set_time_in_sec (const char * time_in_sec)`

Set system time in second.

Parameters

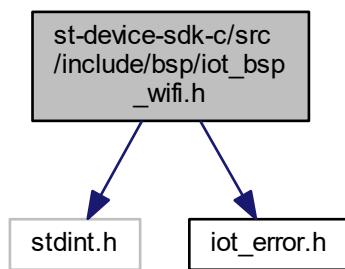
in	<i>time_in_sec</i>	A time value in second from struct timeval's tv_sec (ex : 1546300800)
----	--------------------	---

Return values

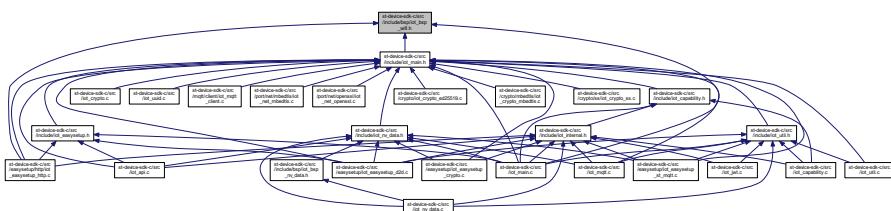
<i>IOT_ERROR_NONE</i>	Set time successful.
<i>IOT_ERROR_INVALID_ARGS</i>	Invalid argument.

5.14 st-device-sdk-c/src/include/bsp/iot_bsp_wifi.h File Reference

```
#include <stdint.h>
#include "iot_error.h"
Include dependency graph for iot_bsp_wifi.h:
```



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



Data Structures

- struct [iot_wifi_conf](#)
Contains a "wifi stack configuration" data.
- struct [iot_wifi_scan_result_t](#)
Contains a "wifi scan" data.
- struct [iot_mac](#)
Contains "wifi mac" data.

Macros

- #define IOT_WIFI_MAX_SSID_LEN (32)
- #define IOT_WIFI_MAX_PASS_LEN (64)
- #define IOT_WIFI_MAX_BSSID_LEN (6)
- #define IOT_WIFI_MAX_SCAN_RESULT (20)
- #define IOT_SOFT_AP_CHANNEL (1)
- #define IOT_WIFI_CMD_TIMEOUT 5000

Enumerations

- enum [iot_wifi_mode_t](#){
IOT_WIFI_MODE_OFF = 0, IOT_WIFI_MODE_SCAN, IOT_WIFI_MODE_STATION, IOT_WIFI_MODE_SOFTAP,
IOT_WIFI_MODE_P2P, IOT_WIFI_MODE_UNDEFINED = 0x20 }
- enum [iot_wifi_freq_t](#){ IOT_WIFI_FREQ_2_4G_ONLY = 0, IOT_WIFI_FREQ_5G_ONLY, IOT_WIFI_FREQ_2_4G_5G_BOTH }
- enum [iot_wifi_auth_mode_t](#){
IOT_WIFI_AUTH_OPEN = 0, IOT_WIFI_AUTH_WEP, IOT_WIFI_AUTH_WPA_PSK, IOT_WIFI_AUTH_WPA2_PSK,
IOT_WIFI_AUTH_WPA_WPA2_PSK, IOT_WIFI_AUTH_WPA2_ENTERPRISE, IOT_WIFI_AUTH_MAX }

Functions

- [iot_error_t iot_bsp_wifi_init \(\)](#)
Initialize Wi-Fi function.
- [iot_error_t iot_bsp_wifi_set_mode \(iot_wifi_conf *conf\)](#)
Set the Wi-Fi mode.
- [uint16_t iot_bsp_wifi_get_scan_result \(iot_wifi_scan_result_t *scan_result\)](#)
Get the AP scan result.
- [iot_error_t iot_bsp_wifi_get_mac \(struct iot_mac *wifi_mac\)](#)
Get the Wi-Fi MAC.
- [iot_wifi_freq_t iot_bsp_wifi_get_freq \(void\)](#)
Get the Wi-Fi Frequency band.

5.14.1 Macro Definition Documentation

5.14.1.1 IOT_SOFT_AP_CHANNEL #define IOT_SOFT_AP_CHANNEL (1)

5.14.1.2 IOT_WIFI_CMD_TIMEOUT #define IOT_WIFI_CMD_TIMEOUT 5000

5.14.1.3 IOT_WIFI_MAX_BSSID_LEN #define IOT_WIFI_MAX_BSSID_LEN (6)

5.14.1.4 IOT_WIFI_MAX_PASS_LEN #define IOT_WIFI_MAX_PASS_LEN (64)

5.14.1.5 IOT_WIFI_MAX_SCAN_RESULT #define IOT_WIFI_MAX_SCAN_RESULT (20)

5.14.1.6 IOT_WIFI_MAX_SSID_LEN #define IOT_WIFI_MAX_SSID_LEN (32)

5.14.2 Enumeration Type Documentation

5.14.2.1 iot_wifi_auth_mode_t enum iot_wifi_auth_mode_t

Enumerator

IOT_WIFI_AUTH_OPEN	
IOT_WIFI_AUTH_WEP	
IOT_WIFI_AUTH_WPA_PSK	
IOT_WIFI_AUTH_WPA2_PSK	
IOT_WIFI_AUTH_WPA_WPA2_PSK	
IOT_WIFI_AUTH_WPA2_ENTERPRISE	
IOT_WIFI_AUTH_MAX	

5.14.2.2 iot_wifi_freq_t enum iot_wifi_freq_t

Enumerator

IOT_WIFI_FREQ_2_4G_ONLY	
IOT_WIFI_FREQ_5G_ONLY	
IOT_WIFI_FREQ_2_4G_5G_BOTH	

5.14.2.3 iot_wifi_mode_t enum iot_wifi_mode_t

Enumerator

IOT_WIFI_MODE_OFF	
IOT_WIFI_MODE_SCAN	
IOT_WIFI_MODE_STATION	
IOT_WIFI_MODE_SOFTAP	
IOT_WIFI_MODE_P2P	
IOT_WIFI_MODE_UNDEFINED	

5.14.3 Function Documentation

5.14.3.1 iot_bsp_wifi_get_freq() iot_wifi_freq_t iot_bsp_wifi_get_freq (void)

Get the Wi-Fi Frequency band.

This function get the Wi-Fi Frequency band

Returns

IOT_WIFI_FREQ_2_4G_ONLY : 2.4GHz only supported
 IOT_WIFI_FREQ_5G_ONLY : 5GHz only supported
 IOT_WIFI_FREQ_2_4G_5G_BOTH : 2.4GHz and 5GHz both supported

5.14.3.2 iot_bsp_wifi_get_mac() iot_error_t iot_bsp_wifi_get_mac (struct iot_mac * wifi_mac)

Get the Wi-Fi MAC.

This function get the Wi-Fi MAC

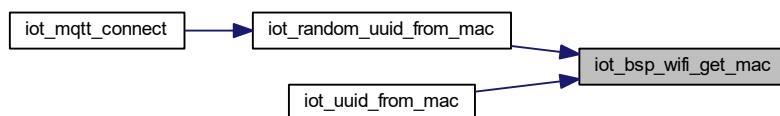
Parameters

out	<i>iot_mac</i>	array to save Wi-Fi MAC
-----	----------------	-------------------------

Returns

IOT_ERROR_NONE : Success IOT_ERROR_READ_FAIL

Here is the caller graph for this function:



5.14.3.3 `iot_bsp_wifi_get_scan_result()` `uint16_t iot_bsp_wifi_get_scan_result (iot_wifi_scan_result_t * scan_result)`

Get the AP scan result.

This function get the scan result

Parameters

out	<i>iot_wifi_scan_result_t</i>	array to save AP list
-----	-------------------------------	-----------------------

Returns

number of APs found

5.14.3.4 `iot_bsp_wifi_init()` `iot_error_t iot_bsp_wifi_init ()`

Initialize Wi-Fi function.

This function initializes Wi-Fi

Returns

IOT_ERROR_NONE : succeed

Here is the caller graph for this function:



5.14.3.5 iot_bsp_wifi_set_mode() `iot_error_t iot_bsp_wifi_set_mode (iot_wifi_conf * conf)`

Set the Wi-Fi mode.

This function set the wifi operating mode as scan, station and softap

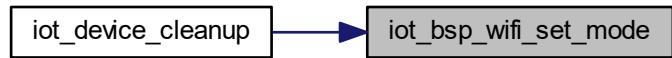
Parameters

in	<i>mode</i>	Wi-Fi operation mode
----	-------------	----------------------

Returns

IOT_ERROR_NONE : succeed

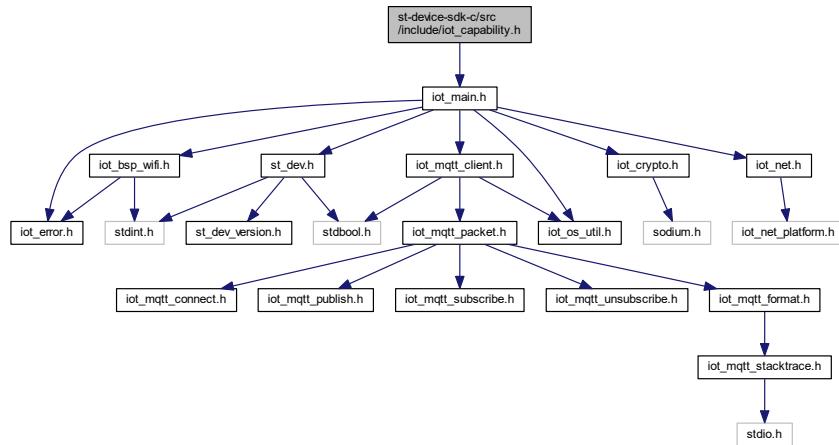
Here is the caller graph for this function:



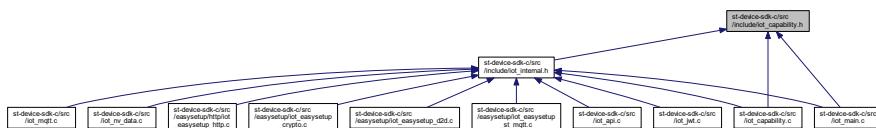
5.15 st-device-sdk-c/src/include/iot_capability.h File Reference

```
#include "iot_main.h"
```

Include dependency graph for iot_capability.h:



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



Data Structures

- struct `iot_cap_unit_t`
Contains a "unit" data.
- struct `iot_cap_evt_data_t`
Contains data for "deviceEvent" payload.
- struct `iot_cap_cmd_set`
Contains user command callback function data.
- struct `iot_cap_cmd_set_list`
linked list for command callback function data
- struct `iot_cap_handle`
Contains data for capability handle.
- struct `iot_cap_handle_list`
linked list for capability handle

Typedefs

- typedef struct `iot_cap_evt_data_t` `iot_cap_evt_data_t`
Contains data for "deviceEvent" payload.
- typedef struct `iot_cap_cmd_set` `iot_cap_cmd_set_t`
Contains user command callback function data.
- typedef struct `iot_cap_cmd_set_list` `iot_cap_cmd_set_list_t`
linked list for command callback function data

Enumerations

- enum `iot_cap_unit_type` { `IOT_CAP_UNIT_TYPE_UNUSED`, `IOT_CAP_UNIT_TYPE_STRING` }

5.15.1 Typedef Documentation

5.15.1.1 `iot_cap_cmd_set_list_t` `typedef struct iot_cap_cmd_set_list iot_cap_cmd_set_list_t`

linked list for command callback function data

5.15.1.2 `iot_cap_cmd_set_t` `typedef struct iot_cap_cmd_set iot_cap_cmd_set_t`

Contains user command callback function data.

5.15.1.3 `iot_cap_evt_data_t` `typedef struct iot_cap_evt_data_t iot_cap_evt_data_t`

Contains data for "deviceEvent" payload.

5.15.2 Enumeration Type Documentation

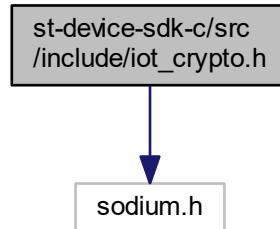
5.15.2.1 `iot_cap_unit_type` `enum iot_cap_unit_type`

Enumerator

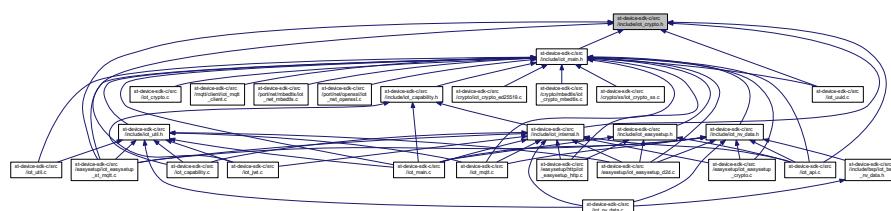
<code>IOT_CAP_UNIT_TYPE_UNUSED</code>	
<code>IOT_CAP_UNIT_TYPE_STRING</code>	

5.16 st-device-sdk-c/src/include/iot_crypto.h File Reference

```
#include "sodium.h"
Include dependency graph for iot_crypto.h:
```



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



Data Structures

- struct [iot_crypto_pk_info](#)
Contains information of key pair.
- struct [iot_crypto_pk_funcs](#)
Contains public key based function lists.
- struct [iot_crypto_pk_context](#)
Contains key pair information and public key based function lists.
- struct [iot_crypto_keypair](#)
Contains public and private keys for Ed25519.
- struct [iot_crypto_ed25519_keypair](#)
Contains Ed25519 and Curve25519 key pairs.
- struct [iot_crypto_ecdh_params](#)
Contains parameters to share a key between Things and ST apps.
- struct [iot_crypto_cipher_info](#)
Contains cipher informations.

Macros

- #define IOT_ERROR_CRYPTO_BASE64 (IOT_ERROR_CRYPTO_BASE - 1)
- #define IOT_ERROR_CRYPTO_BASE64_URLSAFE (IOT_ERROR_CRYPTO_BASE - 2)
- #define IOT_ERROR_CRYPTO_SHA256 (IOT_ERROR_CRYPTO_BASE - 10)
- #define IOT_ERROR_CRYPTO_PK_SIGN (IOT_ERROR_CRYPTO_BASE - 20)
- #define IOT_ERROR_CRYPTO_PK_VERIFY (IOT_ERROR_CRYPTO_BASE - 21)
- #define IOT_ERROR_CRYPTO_PK_PARSEKEY (IOT_ERROR_CRYPTO_BASE - 22)
- #define IOT_ERROR_CRYPTO_PK_INVALID_ARG (IOT_ERROR_CRYPTO_BASE - 23)
- #define IOT_ERROR_CRYPTO_PK_INVALID_CTX (IOT_ERROR_CRYPTO_BASE - 24)
- #define IOT_ERROR_CRYPTO_PK_INVALID_KEYLEN (IOT_ERROR_CRYPTO_BASE - 25)
- #define IOT_ERROR_CRYPTO_PK_UNKNOWN_KEYTYPE (IOT_ERROR_CRYPTO_BASE - 26)
- #define IOT_ERROR_CRYPTO_PK_NULL_FUNC (IOT_ERROR_CRYPTO_BASE - 27)
- #define IOT_ERROR_CRYPTO_PK_ECDH (IOT_ERROR_CRYPTO_BASE - 28)
- #define IOT_ERROR_CRYPTO_ED_KEY_CONVERT (IOT_ERROR_CRYPTO_BASE - 40)
- #define IOT_ERROR_CRYPTO_CIPHER (IOT_ERROR_CRYPTO_BASE - 60)
- #define IOT_ERROR_CRYPTO_CIPHER_UNKNOWN_TYPE (IOT_ERROR_CRYPTO_BASE - 61)
- #define IOT_ERROR_CRYPTO_CIPHER_UNKNOWN_MODE (IOT_ERROR_CRYPTO_BASE - 62)
- #define IOT_ERROR_CRYPTO_CIPHER_KEYLEN (IOT_ERROR_CRYPTO_BASE - 63)
- #define IOT_ERROR_CRYPTO_CIPHER_IVLEN (IOT_ERROR_CRYPTO_BASE - 64)
- #define IOT_ERROR_CRYPTO_CIPHER_OUTSIZE (IOT_ERROR_CRYPTO_BASE - 65)
- #define IOT_ERROR_CRYPTO_CIPHER_ALIGN (IOT_ERROR_CRYPTO_BASE - 66)
- #define IOT_ERROR_CRYPTO_SS_KDF (IOT_ERROR_CRYPTO_BASE - 80)
- #define IOT_CRYPTO_PK_TYPE_RSA "RSA"
- #define IOT_CRYPTO_PK_TYPE_ED25519 "ED25519"
- #define IOT_CRYPTO_ED25519_LEN crypto_sign_PUBLICKEYBYTES
- #define IOT_CRYPTO_SECRET_LEN 32
- #define IOT_CRYPTO_IV_LEN 16
- #define IOT_CRYPTO_SHA256_LEN 32
- #define IOT_CRYPTO_SIGNATURE_LEN 256
- #define IOT_CRYPTO_CAL_B64_LEN(x) (((x) + 2) / 3) * 4 + 1
- #define IOT_CRYPTO_ALIGN_B64_LEN(x) ((x + 3) & ~3)

Typedefs

- typedef struct `iot_crypto_pk_context` `iot_crypto_pk_context_t`
- typedef struct `iot_crypto_pk_info` `iot_crypto_pk_info_t`

Contains information of key pair.
- typedef struct `iot_crypto_pk_funcs` `iot_crypto_pk_funcs_t`

Contains public key based function lists.
- typedef struct `iot_crypto_ed25519_keypair` `iot_crypto_ed25519_keypair_t`

Contains Ed25519 and Curve25519 key pairs.
- typedef struct `iot_crypto_ecdh_params` `iot_crypto_ecdh_params_t`

Contains parameters to share a key between Things and ST apps.
- typedef struct `iot_crypto_cipher_info` `iot_crypto_cipher_info_t`

Contains cipher informations.

Enumerations

- enum `iot_crypto_pk_type_t` { `IOT_CRYPTO_PK_RSA`, `IOT_CRYPTO_PK_ED25519`, `IOT_CRYPTO_PK_UNKNOWN` }

Types of registered public key.

- enum `iot_crypto_cipher_type_t` { `IOT_CRYPTO_CIPHER_AES256`, `IOT_CRYPTO_CIPHER_UNKNOWN` }

Types of supported cipher algorithms.

- enum `iot_crypto_cipher_mode_t` { `IOT_CRYPTO_CIPHER_DECRYPT`, `IOT_CRYPTO_CIPHER_ENCRYPT` }

Types of cipher operations.

Functions

- `iot_error_t iot_crypto_base64_encode` (`unsigned char *src`, `size_t src_len`, `unsigned char *dst`, `size_t dst_len`, `size_t *out_len`)
Encode a string as a base64 string.
- `iot_error_t iot_crypto_base64_decode` (`unsigned char *src`, `size_t src_len`, `unsigned char *dst`, `size_t dst_len`, `size_t *out_len`)
Decode a base64 string as a string.
- `iot_error_t iot_crypto_base64_encode_urlsafe` (`unsigned char *src`, `size_t src_len`, `unsigned char *dst`, `size_t dst_len`, `size_t *out_len`)
Encode a string as a urlsafe base64 string.
- `iot_error_t iot_crypto_base64_decode_urlsafe` (`unsigned char *src`, `size_t src_len`, `unsigned char *dst`, `size_t dst_len`, `size_t *out_len`)
Decode a urlsafe base64 string as a string.
- `iot_error_t iot_crypto_sha256` (`unsigned char *src`, `size_t src_len`, `unsigned char *dst`)
Generate a digest by sha256 hash.
- `iot_error_t iot_crypto_pk_init` (`iot_crypto_pk_context_t *ctx`, `iot_crypto_pk_info_t *info`)
Initialize a context by passed private key data.
- `void iot_crypto_pk_free` (`iot_crypto_pk_context_t *ctx`)
Cleanup the context.
- `iot_error_t iot_crypto_pk_sign` (`iot_crypto_pk_context_t *ctx`, `unsigned char *input`, `size_t ilen`, `unsigned char *sig`, `size_t *slen`)
Generate a signature.
- `iot_error_t iot_crypto_pk_verify` (`iot_crypto_pk_context_t *ctx`, `unsigned char *input`, `size_t ilen`, `unsigned char *sig`, `size_t *slen`)
Verify the signature.
- `iot_error_t iot_crypto_ed25519_init_keypair` (`iot_crypto_ed25519_keypair_t *kp`)
Prepare a buffer to store the ed25519 and curve25519 keypair.
- `void iot_crypto_ed25519_free_keypair` (`iot_crypto_ed25519_keypair_t *kp`)
Cleanup the buffer that allocated for keypair.
- `iot_error_t iot_crypto_ed25519_convert_keypair` (`iot_crypto_ed25519_keypair_t *kp`)
Converts an ed25519 keypair to an x25519 keypair.
- `iot_error_t iot_crypto_ed25519_convert_pubkey` (`unsigned char *ed25519_key`, `unsigned char *curve25519_key`)
Converts an ed25519 public key to an x25519 public key.
- `iot_error_t iot_crypto_ed25519_convert_seckey` (`unsigned char *ed25519_key`, `unsigned char *curve25519_key`)
Converts an ed25519 secret key to an x25519 secret key.
- `iot_error_t iot_crypto_ecdh_gen_master_secret` (`unsigned char *master`, `size_t mlen`, `iot_crypto_ecdh_params_t *params`)
Generate a master secret.

- size_t `iot_crypto_cipher_get_align_size` (`iot_crypto_cipher_type_t` type, `size_t` size)
Calculate the required align size that contains the encrypted data for the input size.
- `iot_error_t iot_crypto_cipher_aes` (`iot_crypto_cipher_info_t` *info, `unsigned char` *input, `size_t` ilen, `unsigned char` *out, `size_t` *olen, `size_t` osize)
Generic encryption/decryption function.
- `iot_error_t iot_crypto_ss_encrypt` (`unsigned char` *input, `size_t` ilen, `unsigned char` **output, `size_t` *olen)
Encryption of the input data.
- `iot_error_t iot_crypto_ss_decrypt` (`unsigned char` *input, `size_t` ilen, `unsigned char` **output, `size_t` *olen)
Decryption of the input data.

5.16.1 Macro Definition Documentation

5.16.1.1 IOT_CRYPTO_ALIGN_B64_LEN `#define IOT_CRYPTO_ALIGN_B64_LEN(`
 `x) ((x + 3) & ~3)`

5.16.1.2 IOT_CRYPTO_CAL_B64_LEN `#define IOT_CRYPTO_CAL_B64_LEN(`
 `x) (((x) + 2) / 3) * 4 + 1)`

5.16.1.3 IOT_CRYPTO_ED25519_LEN `#define IOT_CRYPTO_ED25519_LEN crypto_sign_PUBLICKEYBYTES`

5.16.1.4 IOT_CRYPTO_IV_LEN `#define IOT_CRYPTO_IV_LEN 16`

5.16.1.5 IOT_CRYPTO_PK_TYPE_ED25519 `#define IOT_CRYPTO_PK_TYPE_ED25519 "ED25519"`

5.16.1.6 IOT_CRYPTO_PK_TYPE_RSA `#define IOT_CRYPTO_PK_TYPE_RSA "RSA"`

5.16.1.7 IOT_CRYPTO_SECRET_LEN `#define IOT_CRYPTO_SECRET_LEN 32`

5.16.1.8 IOT_CRYPTO_SHA256_LEN #define IOT_CRYPTO_SHA256_LEN 32

5.16.1.9 IOT_CRYPTO_SIGNATURE_LEN #define IOT_CRYPTO_SIGNATURE_LEN 256

5.16.1.10 IOT_ERROR_CRYPTO_BASE64 #define IOT_ERROR_CRYPTO_BASE64 (IOT_ERROR_CRYPTO_BASE - 1)

5.16.1.11 IOT_ERROR_CRYPTO_BASE64_URLSAFE #define IOT_ERROR_CRYPTO_BASE64_URLSAFE (IOT_ERROR_CRYPTO_BASE - 2)

5.16.1.12 IOT_ERROR_CRYPTO_CIPHER #define IOT_ERROR_CRYPTO_CIPHER (IOT_ERROR_CRYPTO_BASE - 60)

5.16.1.13 IOT_ERROR_CRYPTO_CIPHER_ALIGN #define IOT_ERROR_CRYPTO_CIPHER_ALIGN (IOT_ERROR_CRYPTO_BASE - 66)

5.16.1.14 IOT_ERROR_CRYPTO_CIPHER_IVLEN #define IOT_ERROR_CRYPTO_CIPHER_IVLEN (IOT_ERROR_CRYPTO_BASE - 64)

5.16.1.15 IOT_ERROR_CRYPTO_CIPHER_KEYLEN #define IOT_ERROR_CRYPTO_CIPHER_KEYLEN (IOT_ERROR_CRYPTO_BASE - 63)

5.16.1.16 IOT_ERROR_CRYPTO_CIPHER_OUTSIZE #define IOT_ERROR_CRYPTO_CIPHER_OUTSIZE (IOT_ERROR_CRYPTO_BASE - 65)

5.16.1.17 IOT_ERROR_CRYPTO_CIPHER_UNKNOWN_MODE #define IOT_ERROR_CRYPTO_CIPHER_UNKN← OWN_MODE (IOT_ERROR_CRYPTO_BASE - 62)

5.16.1.18 IOT_ERROR_CRYPTO_CIPHER_UNKNOWN_TYPE #define IOT_ERROR_CRYPTO_CIPHER_UNKNO←
WN_TYPE (IOT_ERROR_CRYPTO_BASE - 61)

5.16.1.19 IOT_ERROR_CRYPTO_ED_KEY_CONVERT #define IOT_ERROR_CRYPTO_ED_KEY_CONVERT (IOT_ERROR_CRYPTO_BA←
- 40)

5.16.1.20 IOT_ERROR_CRYPTO_PK_ECDH #define IOT_ERROR_CRYPTO_PK_ECDH (IOT_ERROR_CRYPTO_BA←
- 28)

5.16.1.21 IOT_ERROR_CRYPTO_PK_INVALID_ARG #define IOT_ERROR_CRYPTO_PK_INVALID_ARG (IOT_ERROR_CRYPTO_BA←
- 23)

5.16.1.22 IOT_ERROR_CRYPTO_PK_INVALID_CTX #define IOT_ERROR_CRYPTO_PK_INVALID_CTX (IOT_ERROR_CRYPTO_BA←
- 24)

5.16.1.23 IOT_ERROR_CRYPTO_PK_INVALID_KEYLEN #define IOT_ERROR_CRYPTO_PK_INVALID_KEYLEN (IOT_ERROR_CRYPTO_BA←
- 25)

5.16.1.24 IOT_ERROR_CRYPTO_PK_NULL_FUNC #define IOT_ERROR_CRYPTO_PK_NULL_FUNC (IOT_ERROR_CRYPTO_BA←
- 27)

5.16.1.25 IOT_ERROR_CRYPTO_PK_PARSEKEY #define IOT_ERROR_CRYPTO_PK_PARSEKEY (IOT_ERROR_CRYPTO_BA←
- 22)

5.16.1.26 IOT_ERROR_CRYPTO_PK_SIGN #define IOT_ERROR_CRYPTO_PK_SIGN (IOT_ERROR_CRYPTO_BA←
- 20)

5.16.1.27 IOT_ERROR_CRYPTO_PK_UNKNOWN_KEYTYPE #define IOT_ERROR_CRYPTO_PK_UNKNOWN_K←
EYTYPE (IOT_ERROR_CRYPTO_BASE - 26)

5.16.1.28 IOT_ERROR_CRYPTO_PK_VERIFY #define IOT_ERROR_CRYPTO_PK_VERIFY (IOT_ERROR_CRYPTO_BASE - 21)

5.16.1.29 IOT_ERROR_CRYPTO_SHA256 #define IOT_ERROR_CRYPTO_SHA256 (IOT_ERROR_CRYPTO_BASE - 10)

5.16.1.30 IOT_ERROR_CRYPTO_SS_KDF #define IOT_ERROR_CRYPTO_SS_KDF (IOT_ERROR_CRYPTO_BASE - 80)

5.16.2 Typedef Documentation

5.16.2.1 iot_crypto_cipher_info_t typedef struct iot_crypto_cipher_info iot_crypto_cipher_info_t
Contains cipher informations.

5.16.2.2 iot_crypto_ecdh_params_t typedef struct iot_crypto_ecdh_params iot_crypto_ecdh_params_t
Contains parameters to share a key between Things and ST apps.

5.16.2.3 iot_crypto_ed25519_keypair_t typedef struct iot_crypto_ed25519_keypair iot_crypto_ed25519_keypair_t
Contains Ed25519 and Curve25519 key pairs.

5.16.2.4 iot_crypto_pk_context_t typedef struct iot_crypto_pk_context iot_crypto_pk_context_t

5.16.2.5 iot_crypto_pk_funcs_t typedef struct iot_crypto_pk_funcs iot_crypto_pk_funcs_t
Contains public key based function lists.

5.16.2.6 iot_crypto_pk_info_t typedef struct iot_crypto_pk_info iot_crypto_pk_info_t
Contains information of key pair.

5.16.3 Enumeration Type Documentation

5.16.3.1 iot_crypto_cipher_mode_t enum iot_crypto_cipher_mode_t
Types of cipher operations.

Enumerator

IOT_CRYPTO_CIPHER_DECRYPT	
IOT_CRYPTO_CIPHER_ENCRYPT	

5.16.3.2 iot_crypto_cipher_type_t enum iot_crypto_cipher_type_t

Types of supported cipher algorithms.

Enumerator

IOT_CRYPTO_CIPHER_AES256	
IOT_CRYPTO_CIPHER_UNKNOWN	

5.16.3.3 iot_crypto_pk_type_t enum iot_crypto_pk_type_t

Types of registered public key.

Enumerator

IOT_CRYPTO_PK_RSA	
IOT_CRYPTO_PK_ED25519	
IOT_CRYPTO_PK_UNKNOWN	

5.16.4 Function Documentation

```
5.16.4.1 iot_crypto_base64_decode() iot_error_t iot_crypto_base64_decode (
    unsigned char * src,
    size_t src_len,
    unsigned char * dst,
    size_t dst_len,
    size_t * out_len )
```

Decode a base64 string as a string.

Parameters

in	<i>src</i>	a pointer to a buffer to decode
in	<i>src_len</i>	the size of buffer pointed by src in bytes
out	<i>dst</i>	a pointer to a buffer to store base64 string
in	<i>dst_len</i>	the size of buffer pointed by dst in bytes
out	<i>out_len</i>	the bytes written to dst

Return values

<i>IOT_ERROR_NONE</i>	the string is sucessfully decoded
<i>IOT_ERROR_CRYPTO_BASE64</i>	failed to decode the string

5.16.4.2 iot_crypto_base64_decode_urlsafe() *iot_error_t* iot_crypto_base64_decode_urlsafe (

```
    unsigned char * src,
    size_t src_len,
    unsigned char * dst,
    size_t dst_len,
    size_t * out_len )
```

Decode a urlsafe base64 string as a string.

Parameters

in	<i>src</i>	a pointer to a buffer to decode
in	<i>src_len</i>	the size of buffer pointed by src in bytes
out	<i>dst</i>	a pointer to a buffer to store base64 string
in	<i>dst_len</i>	the size of buffer pointed by dst in bytes
out	<i>out_len</i>	the bytes written to dst

Return values

<i>IOT_ERROR_NONE</i>	the string is sucessfully decoded
<i>IOT_ERROR_CRYPTO_BASE64</i>	failed to decode the string
<i>IOT_ERROR_CRYPTO_BASE64_URLSAFE</i>	failed to encode the string as urlsafe

5.16.4.3 iot_crypto_base64_encode() *iot_error_t* iot_crypto_base64_encode (

```
    unsigned char * src,
    size_t src_len,
    unsigned char * dst,
    size_t dst_len,
    size_t * out_len )
```

Encode a string as a base64 string.

Parameters

in	<i>src</i>	a pointer to a buffer to encode
in	<i>src_len</i>	the size of buffer pointed by src in bytes
out	<i>dst</i>	a pointer to a buffer to store base64 string
in	<i>dst_len</i>	the size of buffer pointed by dst in bytes
out	<i>out_len</i>	the bytes written to dst

Return values

<i>IOT_ERROR_NONE</i>	the string is sucessfully encoded
<i>IOT_ERROR_CRYPTO_BASE64</i>	failed to encode the string

5.16.4.4 iot_crypto_base64_encode_urlsafe() *iot_error_t* iot_crypto_base64_encode_urlsafe (

```
    unsigned char * src,
    size_t src_len,
    unsigned char * dst,
    size_t dst_len,
    size_t * out_len )
```

Encode a string as a urlsafe base64 string.

This function replaces url unsafe characters ('+', '/') to url safe character ('-', '_')

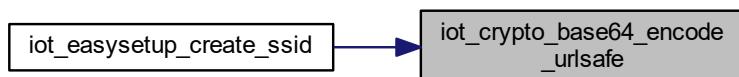
Parameters

in	<i>src</i>	a pointer to a buffer to encode
in	<i>src_len</i>	the size of buffer pointed by src in bytes
out	<i>dst</i>	a pointer to a buffer to store base64 string
in	<i>dst_len</i>	the size of buffer pointed by dst in bytes
out	<i>out_len</i>	the bytes written to dst

Return values

<i>IOT_ERROR_NONE</i>	the string is sucessfully encoded
<i>IOT_ERROR_CRYPTO_BASE64</i>	failed to encode the string in 3rd lib
<i>IOT_ERROR_CRYPTO_BASE64_URLSAFE</i>	failed to encode the string as urlsafe

Here is the caller graph for this function:

**5.16.4.5 iot_crypto_cipher_aes()** *iot_error_t* iot_crypto_cipher_aes (

```
    iot_crypto_cipher_info_t * info,
    unsigned char * input,
```

```
size_t ilen,
unsigned char * out,
size_t * olen,
size_t osize )
```

Generic encryption/decryption function.

Supported cipher algorithms is AES-256 CBC mode

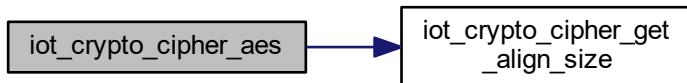
Parameters

in	<i>info</i>	a pointer to a buffer containing cipher context
in	<i>input</i>	a pointer to a buffer to encrypt/decrypt
in	<i>ilen</i>	the size of buffer pointed by input in bytes
out	<i>out</i>	a pointer to a buffer to store a signature
out	<i>olen</i>	the bytes written to out
in	<i>osize</i>	the size of buffer pointed by out in bytes

Return values

<i>IOT_ERROR_NONE</i>	encryption/decryption is a success
<i>IOT_ERROR_CRYPTO_CIPHER_UNKNOWN_TYPE</i>	not supported cipher is requested.
<i>IOT_ERROR_CRYPTO_CIPHER</i>	failed while encrypting/decrypting

Here is the call graph for this function:



5.16.4.6 iot_crypto_cipher_get_align_size() `size_t iot_crypto_cipher_get_align_size (`
 `iot_crypto_cipher_type_t type,`
 `size_t size)`

Calculate the required align size that contains the encrypted data for the input size.

the size of data to encrypt should be aligned when AES is used because AES is operated based on block.

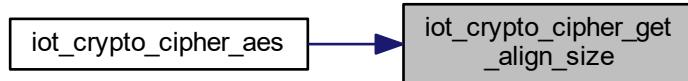
Parameters

in	<i>type</i>	cipher type
in	<i>size</i>	the size of data to encrypt

Return values

<i>return</i>	the aligned size
<i>return</i>	'0' when error occurs

Here is the caller graph for this function:



5.16.4.7 iot_crypto_ecdh_gen_master_secret() `iot_error_t iot_crypto_ecdh_gen_master_secret (`
 `unsigned char * master,`
 `size_t mlen,`
 `iot_crypto_ecdh_params_t * params)`

Generate a master secret.

This function generates a master secret by combining the shared key and the hashed token. The shared key is generated by ECDH using device's x25519 private key and peer's x25519 public key. The hashed token is shared by peer.

Parameters

<i>out</i>	<i>master</i>	a pointer to a buffer to store master secret
<i>in</i>	<i>mlen</i>	the size of buffer pointed by master in bytes
<i>in</i>	<i>params</i>	a pointer to a buffer containing the private key, public key and hashed token to generate master secret

Return values

<i>IOT_ERROR_NONE</i>	the master secret is sucessfully generated
<i>IOT_ERROR_INVALID_ARGS</i>	params has invalid data
<i>IOT_ERROR_MEM_ALLOC</i>	failed to alloc buffer for shared key
<i>IOT_ERROR_CRYPTO_PK_ECDH</i>	failed in ECDH processing
<i>IOT_ERROR_CRYPTO_ED_KEY_CONVERT</i>	failed to convert to x25519

5.16.4.8 iot_crypto_ed25519_convert_keypair() `iot_error_t iot_crypto_ed25519_convert_keypair (`
 `iot_crypto_ed25519_keypair_t * kp)`

Converts an ed25519 keypair to an x25519 keypair.

Parameters

in	<i>kp</i>	a pointer to a structure of the keypair buffers ed25519 keypair should be prepared
----	-----------	--

Return values

<i>IOT_ERROR_NONE</i>	the buffer is sucessfully allocated
<i>IOT_ERROR_CRYPTO_ED_KEY_CONVERT</i>	failed to convert to x25519

5.16.4.9 iot_crypto_ed25519_convert_pubkey() *iot_error_t* `iot_crypto_ed25519_convert_pubkey (`
 *unsigned char * ed25519_key,*
 *unsigned char * curve25519_key)*

Converts an ed25519 public key to an x25519 public key.

Parameters

in	<i>ed25519_key</i>	a pointer to a public key buffer
out	<i>curve25519_key</i>	a pointer to a buffer to store converted x25519 public key

Return values

<i>IOT_ERROR_NONE</i>	the buffer is sucessfully allocated
<i>IOT_ERROR_CRYPTO_ED_KEY_CONVERT</i>	failed to convert to x25519

5.16.4.10 iot_crypto_ed25519_convert_seckey() *iot_error_t* `iot_crypto_ed25519_convert_seckey (`
 *unsigned char * ed25519_key,*
 *unsigned char * curve25519_key)*

Converts an ed25519 secret key to an x25519 secret key.

Parameters

in	<i>ed25519_key</i>	a pointer to a secret key buffer
out	<i>curve25519_key</i>	a pointer to a buffer to store converted x25519 secret key

Return values

<i>IOT_ERROR_NONE</i>	the buffer is sucessfully allocated
<i>IOT_ERROR_CRYPTO_ED_KEY_CONVERT</i>	failed to convert to x25519

```
5.16.4.11 iot_crypto_ed25519_free_keypair() void iot_crypto_ed25519_free_keypair (
    iot_crypto_ed25519_keypair_t * kp )
```

Cleanup the buffer that allocated for keypair.

Parameters

in	kp	a pointer to a structure of the keypair to cleanup
----	----	--

```
5.16.4.12 iot_crypto_ed25519_init_keypair() iot_error_t iot_crypto_ed25519_init_keypair (
    iot_crypto_ed25519_keypair_t * kp )
```

Prepare a buffer to store the ed25519 and curve25519 keypair.

Parameters

in	kp	a pointer to a structure of the keypair buffers
----	----	---

Return values

IOT_ERROR_NONE	the buffer is sucessfully allocated
IOT_ERROR_MEM_ALLOC	failed to alloc buffer for keypair

```
5.16.4.13 iot_crypto_pk_free() void iot_crypto_pk_free (
    iot_crypto_pk_context_t * ctx )
```

Cleanup the context.

Parameters

in	ctx	a pointer to a buffer to cleanup
----	-----	----------------------------------

```
5.16.4.14 iot_crypto_pk_init() iot_error_t iot_crypto_pk_init (
    iot_crypto_pk_context_t * ctx,
    iot_crypto_pk_info_t * info )
```

Initialize a context by passed private key data.

Parameters

in	ctx	a pointer to a buffer to handle crypto context
in	info	a pointer to a buffer containing private key data

Return values

<i>IOT_ERROR_NONE</i>	context is sucessfully initialized
<i>IOT_ERROR_CRYPTO_PK_INVALID_CTX</i>	ctx is null
<i>IOT_ERROR_CRYPTO_PK_INVALID_ARG</i>	info is null
<i>IOT_ERROR_CRYPTO_PK_UNKNOWN_KEYTYPE</i>	private key data has not supported algorithm

5.16.4.15 iot_crypto_pk_sign() *iot_error_t* iot_crypto_pk_sign (

```
    iot_crypto_pk_context_t * ctx,
    unsigned char * input,
    size_t ilen,
    unsigned char * sig,
    size_t * slen )
```

Generate a signature.

Parameters

in	<i>ctx</i>	a pointer to a buffer containing crypto context
in	<i>input</i>	a pointer to a buffer to generate a signature
in	<i>ilen</i>	the size of buffer pointed by input in bytes
out	<i>sig</i>	a pointer to a buffer to store a signature
out	<i>slen</i>	the bytes written to sig

Return values

<i>IOT_ERROR_NONE</i>	the signature is sucessfully generated
<i>IOT_ERROR_CRYPTO_PK_PARSEKEY</i>	failed to parse device certificate
<i>IOT_ERROR_CRYPTO_PK_INVALID_CTX</i>	ctx is null
<i>IOT_ERROR_CRYPTO_PK_INVALID_ARG</i>	info is null
<i>IOT_ERROR_CRYPTO_PK_INVALID_KEYLEN</i>	a length of private key is wrong
<i>IOT_ERROR_CRYPTO_PK_SIGN</i>	failed to generate signature

5.16.4.16 iot_crypto_pk_verify() *iot_error_t* iot_crypto_pk_verify (

```
    iot_crypto_pk_context_t * ctx,
    unsigned char * input,
    size_t ilen,
    unsigned char * sig,
    size_t slen )
```

Verify the signature.

Parameters

in	<i>ctx</i>	a pointer to a buffer containing crypto context
----	------------	---

Parameters

in	<i>input</i>	a pointer to a buffer to generate a signature
in	<i>ilen</i>	the size of buffer pointed by input in bytes
in	<i>sig</i>	a pointer to a buffer containing the signature
in	<i>slen</i>	the size of buffer pointed by sig in bytes

Return values

<i>IOT_ERROR_NONE</i>	the signature is sucessfully verified
<i>IOT_ERROR_CRYPTO_PK_NULL_FUNC</i>	verify function is not implemented
<i>IOT_ERROR_CRYPTO_PK_PARSEKEY</i>	failed to parse device certificate
<i>IOT_ERROR_CRYPTO_PK_INVALID_CTX</i>	ctx is null
<i>IOT_ERROR_CRYPTO_PK_INVALID_ARG</i>	info is null
<i>IOT_ERROR_CRYPTO_PK_INVALID_KEYLEN</i>	a length of private key is wrong
<i>IOT_ERROR_CRYPTO_PK_VERIFY</i>	the signature is mismatched

5.16.4.17 *iot_crypto_sha256()* *iot_error_t* *iot_crypto_sha256* (

```
    unsigned char * src,
    size_t src_len,
    unsigned char * dst )
```

Generate a digest by sha256 hash.

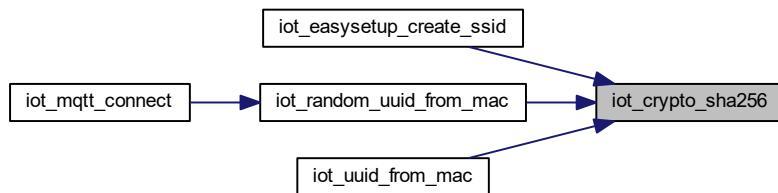
Parameters

in	<i>src</i>	a pointer to a buffer to generate a digest
in	<i>src_len</i>	the size of buffer pointed by src in bytes
out	<i>dst</i>	a pointer to a buffer to store a digest

Return values

<i>IOT_ERROR_NONE</i>	a digest is sucessfully generated
<i>IOT_ERROR_CRYPTO_SHA256</i>	failed to generate a digest

Here is the caller graph for this function:



```
5.16.4.18 iot_crypto_ss_decrypt() iot_error_t iot_crypto_ss_decrypt (
    unsigned char * input,
    size_t ilen,
    unsigned char ** output,
    size_t * olen )
```

Decryption of the input data.

The decryption key is generated from device unique value

Parameters

in	<i>input</i>	a pointer to a buffer to decrypt
in	<i>ilen</i>	the size of buffer pointed by <i>input</i> in bytes
out	<i>output</i>	a pointer of pointer to a buffer to store the decrypted data
out	<i>olen</i>	the bytes written to <i>output</i> buffer

Return values

<i>IOT_ERROR_NONE</i>	decryption is success
<i>IOT_ERROR_MEM_ALLOC</i>	mem alloc failed for output buffer
<i>IOT_ERROR_CRYPTO_CIPHER_ALIGN</i>	failed to get align size of the input size for output buffer
<i>IOT_ERROR_CRYPTO_CIPHER_UNKNOWN_TYPE</i>	cipher is requested with not supported algorithm
<i>IOT_ERROR_CRYPTO_CIPHER</i>	cipher operation is failed
<i>IOT_ERROR_CRYPTO_SS_KDF</i>	failed during derive the key

```
5.16.4.19 iot_crypto_ss_encrypt() iot_error_t iot_crypto_ss_encrypt (
    unsigned char * input,
    size_t ilen,
    unsigned char ** output,
    size_t * olen )
```

Encryption of the input data.

The encryption key is generated from device unique value

Parameters

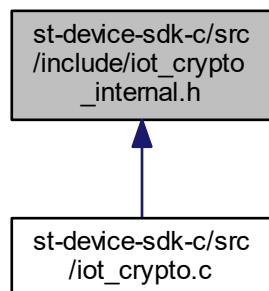
in	<i>input</i>	a pointer to a buffer to encrypt
in	<i>ilen</i>	the size of buffer pointed by input in bytes
out	<i>output</i>	a pointer of pointer to a buffer to store the encrypted data
out	<i>olen</i>	the bytes written to output buffer

Return values

<i>IOT_ERROR_NONE</i>	encryption is success
<i>IOT_ERROR_MEM_ALLOC</i>	mem alloc failed for output buffer
<i>IOT_ERROR_CRYPTO_CIPHER_ALIGN</i>	failed to get align size of the input size for output buffer
<i>IOT_ERROR_CRYPTO_CIPHER</i>	cipher operation is failed
<i>IOT_ERROR_CRYPTO_SS_KDF</i>	failed during derivate the key

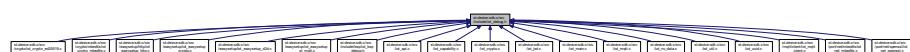
5.17 st-device-sdk-c/src/include/iot_crypto_internal.h File Reference

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



5.18 st-device-sdk-c/src/include/iot_debug.h File Reference

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



iot_debug_level_t

internal debug level.

- #define IOT_DEBUG_PREFIX "[IoT]"
• #define COLOR_CYAN "\033[0;36m"
• #define COLOR_END "\033[0;m"
• #define IOT_ERROR(fmt, args...)
Error level logging macro.
• #define IOT_WARN(fmt, args...)
Warning level logging macro.
• #define IOT_INFO(fmt, args...)
Info level logging macro.
• #define IOT_DEBUG(fmt, args...)
Debug level logging macro.
• #define HIT()
• #define ENTER()
• #define LEAVE()
• #define IOT_MEM_CHECK(fmt, args...)
Memory(heap) checking macro.
• #define IOT_ERROR_CHECK(condition, ret, fmt, args...)
Condition checking macro.
• #define IOT_WARN_CHECK(condition, ret, fmt, args...)
• #define IOT_DEBUG_CHECK(condition, ret, fmt, args...)
• enum iot_debug_level_t {
 IOT_DEBUG_LEVEL_NONE = 0, IOT_DEBUG_LEVEL_ERROR, IOT_DEBUG_LEVEL_WARN, IOT_DEBUG_LEVEL_INFO,
 IOT_DEBUG_LEVEL_DEBUG, IOT_DEBUG_LEVEL_MAX }
• void iot_bsp_debug (iot_debug_level_t level, const char *tag, const char *fmt,...)
• void iot_bsp_debug_check_heap (const char *tag, const char *func, const int line, const char *fmt,...)

5.18.1 Macro Definition Documentation

5.18.1.1 COLOR_CYAN #define COLOR_CYAN "\033[0;36m"

5.18.1.2 COLOR_END #define COLOR_END "\033[0;m"

5.18.1.3 ENTER #define ENTER()

5.18.1.4 HIT #define HIT()

```
5.18.1.5 IOT_DEBUG #define IOT_DEBUG(\
    fmt, \
    args... )
```

Debug level logging macro.

Macro to use log function

```
5.18.1.6 IOT_DEBUG_CHECK #define IOT_DEBUG_CHECK(\
    condition, \
    ret, \
    fmt, \
    args... )
```

Value:

```
do { \
    if ((condition)) { \
        IOT_DEBUG(fmt, ##args); \
        return (ret); \
    } \
} while (0)
```

```
5.18.1.7 IOT_DEBUG_PREFIX #define IOT_DEBUG_PREFIX "[IoT]"
```

```
5.18.1.8 IOT_ERROR #define IOT_ERROR(\
    fmt, \
    args... )
```

Error level logging macro.

Macro to use log function

```
5.18.1.9 IOT_ERROR_CHECK #define IOT_ERROR_CHECK(\
    condition, \
    ret, \
    fmt, \
    args... )
```

Value:

```
do { \
    if ((condition)) { \
        IOT_ERROR(fmt, ##args); \
        return (ret); \
    } \
} while (0)
```

Condition checking macro.

Macro to check condition

5.18.1.10 IOT_INFO #define IOT_INFO(
fmt,
args...)

Info level logging macro.

Macro to use log function

5.18.1.11 IOT_MEM_CHECK #define IOT_MEM_CHECK(
fmt,
args...)

Memory(heap) checking macro.

Macro to check memory(heap)

5.18.1.12 IOT_WARN #define IOT_WARN(
fmt,
args...)

Warning level logging macro.

Macro to use log function

5.18.1.13 IOT_WARN_CHECK #define IOT_WARN_CHECK(
condition,
ret,
fmt,
args...)

Value:

```
do { \
    if ((condition)) { \
        IOT_WARN(fmt, ##args); \
        return (ret); \
    } \
} while (0)
```

5.18.1.14 LEAVE #define LEAVE()

5.18.2 Enumeration Type Documentation

5.18.2.1 iot_debug_level_t enum iot_debug_level_t

Enumerator

IOT_DEBUG_LEVEL_NONE	
IOT_DEBUG_LEVEL_ERROR	
IOT_DEBUG_LEVEL_WARN	
IOT_DEBUG_LEVEL_INFO	
IOT_DEBUG_LEVEL_DEBUG	
IOT_DEBUG_LEVEL_MAX	

5.18.3 Function Documentation

5.18.3.1 iot_bsp_debug() `void iot_bsp_debug (`

```
iot_debug_level_t level,
const char * tag,
const char * fmt,
... )
```

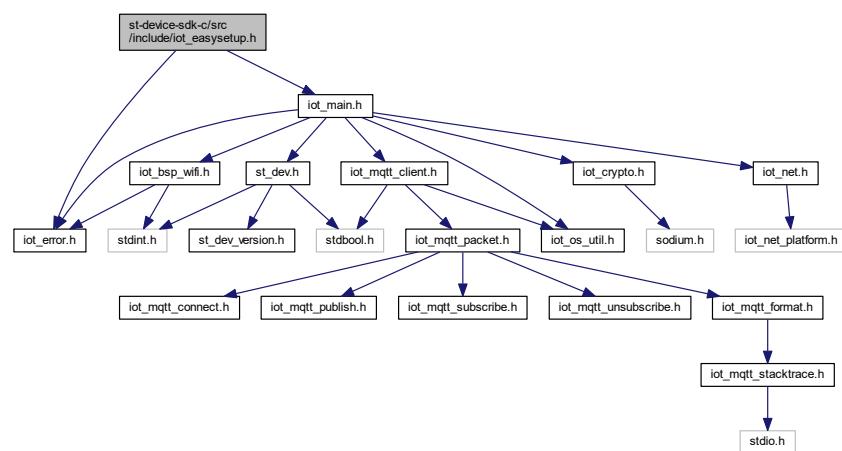
5.18.3.2 iot_bsp_debug_check_heap() `void iot_bsp_debug_check_heap (`

```
const char * tag,
const char * func,
const int line,
const char * fmt,
... )
```

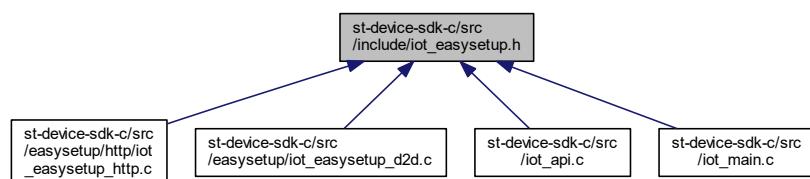
5.19 st-device-sdk-c/src/include/iot_easysetup.h File Reference

```
#include "iot_error.h"
#include "iot_main.h"
```

Include dependency graph for iot_easysetup.h:



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



Macros

- #define IOT_OVF_TYPE JUSTWORKS (1 << OVF_BIT JUSTWORKS)
- #define IOT_OVF_TYPE_BUTTON (1 << OVF_BIT_BUTTON)
- #define IOT_OVF_TYPE_PIN (1 << OVF_BIT_PIN)
- #define IOT_OVF_TYPE_QR (1 << OVF_BIT_QR)
- #define IOT_ES_URI_POST_KEYINFO "keyinfo"
- #define IOT_ES_URI_POST_CONFIRMINFO "confirminfo"
- #define IOT_ES_URI_POST_CONFIRM "confirm"
- #define IOT_ES_URI_POST_WIFIPOWERINGINFO "wifiprovisioninginfo"
- #define IOT_ES_URI_POST_SETUPCOMPLETE "setupcomplete"
- #define IOT_ES_URI_POST_LOGS "logs"
- #define IOT_ES_URI_GET_DEVICEINFO "/deviceinfo"
- #define IOT_ES_URI_GET_WIFISCANINFO "/wifiscaninfo"
- #define IOT_ES_URI_GET_POST_RESPONSE "/post_response"
- #define IOT_ES_URI_GET_LOGS_SYSTEMINFO "/logs/systeminfo"
- #define IOT_ES_URI_GET_LOGS_DUMP "/logs/dump"
- #define IOT_ERROR_EASYSETUP_INVALID_REQUEST (IOT_ERROR_EASYSETUP_BASE - 1)
- #define IOT_ERROR_EASYSETUP_INVALID_SEQUENCE (IOT_ERROR_EASYSETUP_BASE - 2)
- #define IOT_ERROR_EASYSETUP_NOT_SUPPORTED (IOT_ERROR_EASYSETUP_BASE - 3)
- #define IOT_ERROR_EASYSETUP_INVALID_PIN (IOT_ERROR_EASYSETUP_BASE - 4)
- #define IOT_ERROR_EASYSETUP_INVALID_QR (IOT_ERROR_EASYSETUP_BASE - 5)
- #define IOT_ERROR_EASYSETUP_INVALID_BROKER_URL (IOT_ERROR_EASYSETUP_BASE - 6)
- #define IOT_ERROR_EASYSETUP_INVALID_PAYLOAD (IOT_ERROR_EASYSETUP_BASE - 7)
- #define IOT_ERROR_EASYSETUP_INTERNAL_SERVER_ERROR (IOT_ERROR_EASYSETUP_BASE - 50)
- #define IOT_ERROR_EASYSETUP_WIFI_SCANLIST_NOT_FOUND (IOT_ERROR_EASYSETUP_BASE - 51)
- #define IOT_ERROR_EASYSETUP_SERIAL_NOT_FOUND (IOT_ERROR_EASYSETUP_BASE - 52)
- #define IOT_ERROR_EASYSETUP_CONFIRM_TIMEOUT (IOT_ERROR_EASYSETUP_BASE - 53)
- #define IOT_ERROR_EASYSETUP_CONFIRM_DENIED (IOT_ERROR_EASYSETUP_BASE - 54)
- #define IOT_ERROR_EASYSETUP_SHARED_KEY_CREATION_FAIL (IOT_ERROR_EASYSETUP_BASE - 55)
- #define IOT_ERROR_EASYSETUP_RPK_NOT_FOUND (IOT_ERROR_EASYSETUP_BASE - 56)
- #define IOT_ERROR_EASYSETUP_CERTIFICATE_NOT_FOUND (IOT_ERROR_EASYSETUP_BASE - 57)
- #define IOT_ERROR_EASYSETUP_PIN_NOT_FOUND (IOT_ERROR_EASYSETUP_BASE - 58)
- #define IOT_ERROR_EASYSETUP_SPUB_DECODE_ERROR (IOT_ERROR_EASYSETUP_BASE - 59)
- #define IOT_ERROR_EASYSETUP RAND DECODE_ERROR (IOT_ERROR_EASYSETUP_BASE - 60)
- #define IOT_ERROR_EASYSETUP_INTERNAL_WARNING (IOT_ERROR_EASYSETUP_BASE - 97)
- #define IOT_ERROR_EASYSETUP_INTERNAL_CRITICAL_ERROR (IOT_ERROR_EASYSETUP_BASE - 98)
- #define IOT_ERROR_EASYSETUP_400_BASE IOT_ERROR_EASYSETUP_INVALID_REQUEST
- #define IOT_ERROR_EASYSETUP_500_BASE IOT_ERROR_EASYSETUP_INTERNAL_SERVER_ERROR
- #define IOT_ERROR_EASYSETUP_MAX (IOT_ERROR_EASYSETUP_BASE - 99)

Enumerations

- enum ownership_validation_feature {
 OVF_BIT JUSTWORKS = 0, OVF_BIT_QR, OVF_BIT_BUTTON, OVF_BIT_PIN,
 OVF_BIT_MAX_FEATURE
 }

Functions

- `iot_error_t iot_easystatus_request_handler (struct iot_context *ctx, struct iot_easystatus_payload request)`
easystatus cgi request handler
- `iot_error_t iot_easystatus_create_ssid (struct iot_devconf_prov_data *devconf, char *ssid, size_t ssid_len)`
Create E4 type SSID.
- `iot_error_t iot_easystatus_init (struct iot_context *ctx)`
Start easystatus device-to-device sequence.
- `void iot_easystatus_deinit (struct iot_context *ctx)`
Stop easystatus device-to-device sequence.

5.19.1 Macro Definition Documentation

5.19.1.1 IOT_ERROR_EASYSETUP_400_BASE `#define IOT_ERROR_EASYSETUP_400_BASE IOT_ERROR_EASYSETUP_INVALID_R...`

5.19.1.2 IOT_ERROR_EASYSETUP_500_BASE `#define IOT_ERROR_EASYSETUP_500_BASE IOT_ERROR_EASYSETUP_INTERNAL...`

5.19.1.3 IOT_ERROR_EASYSETUP_CERTIFICATE_NOT_FOUND `#define IOT_ERROR_EASYSETUP_CERTIFICATE_NOT_FOUND (IOT_ERROR_EASYSETUP_BASE - 57)`

5.19.1.4 IOT_ERROR_EASYSETUP_CONFIRM_DENIED `#define IOT_ERROR_EASYSETUP_CONFIRM_DENIED (IOT_ERROR_EASYSETUP_BASE - 54)`

5.19.1.5 IOT_ERROR_EASYSETUP_CONFIRM_TIMEOUT `#define IOT_ERROR_EASYSETUP_CONFIRM_TIMEOUT (IOT_ERROR_EASYSETUP_BASE - 53)`

5.19.1.6 IOT_ERROR_EASYSETUP_INTERNAL_CRITICAL_ERROR `#define IOT_ERROR_EASYSETUP_INTERNAL_CRITICAL_ERROR (IOT_ERROR_EASYSETUP_BASE - 98)`

5.19.1.7 IOT_ERROR_EASYSETUP_INTERNAL_SERVER_ERROR `#define IOT_ERROR_EASYSETUP_INTERNAL_SERVER_ERROR (IOT_ERROR_EASYSETUP_BASE - 50)`

5.19.1.8 IOT_ERROR_EASYSETUP_INTERNAL_WARNING #define IOT_ERROR_EASYSETUP_INTERNAL_WARNING (IOT_ERROR_EASYSETUP_BASE - 97)

5.19.1.9 IOT_ERROR_EASYSETUP_INVALID_BROKER_URL #define IOT_ERROR_EASYSETUP_INVALID_BROKER_URL (IOT_ERROR_EASYSETUP_BASE - 6)

5.19.1.10 IOT_ERROR_EASYSETUP_INVALID_PAYLOAD #define IOT_ERROR_EASYSETUP_INVALID_PAYLOAD (IOT_ERROR_EASYSETUP_BASE - 7)

5.19.1.11 IOT_ERROR_EASYSETUP_INVALID_PIN #define IOT_ERROR_EASYSETUP_INVALID_PIN (IOT_ERROR_EASYSETUP_BASE - 4)

5.19.1.12 IOT_ERROR_EASYSETUP_INVALID_QR #define IOT_ERROR_EASYSETUP_INVALID_QR (IOT_ERROR_EASYSETUP_BASE - 5)

5.19.1.13 IOT_ERROR_EASYSETUP_INVALID_REQUEST #define IOT_ERROR_EASYSETUP_INVALID_REQUEST (IOT_ERROR_EASYSETUP_BASE - 1)

5.19.1.14 IOT_ERROR_EASYSETUP_INVALID_SEQUENCE #define IOT_ERROR_EASYSETUP_INVALID_SEQUENCE (IOT_ERROR_EASYSETUP_BASE - 2)

5.19.1.15 IOT_ERROR_EASYSETUP_MAX #define IOT_ERROR_EASYSETUP_MAX (IOT_ERROR_EASYSETUP_BASE - 99)

5.19.1.16 IOT_ERROR_EASYSETUP_NOT_SUPPORTED #define IOT_ERROR_EASYSETUP_NOT_SUPPORTED (IOT_ERROR_EASYSETUP_BASE - 3)

5.19.1.17 IOT_ERROR_EASYSETUP_PIN_NOT_FOUND #define IOT_ERROR_EASYSETUP_PIN_NOT_FOUND (IOT_ERROR_EASYSETUP_BASE - 58)

5.19.1.18 IOT_ERROR_EASYSETUP_RAND_DECODE_ERROR #define IOT_ERROR_EASYSETUP_RAND_DECODE_ERROR (IOT_ERROR_EASYSETUP_BASE - 60)

5.19.1.19 IOT_ERROR_EASYSETUP_RPK_NOT_FOUND #define IOT_ERROR_EASYSETUP_RPK_NOT_FOUND (IOT_ERROR_EASYSETUP_BASE - 56)

5.19.1.20 IOT_ERROR_EASYSETUP_SERIAL_NOT_FOUND #define IOT_ERROR_EASYSETUP_SERIAL_NOT_FOUND (IOT_ERROR_EASYSETUP_BASE - 52)

5.19.1.21 IOT_ERROR_EASYSETUP_SHARED_KEY_CREATION_FAIL #define IOT_ERROR_EASYSETUP_SHARED_KEY_CREATION_FAIL (IOT_ERROR_EASYSETUP_BASE - 55)

5.19.1.22 IOT_ERROR_EASYSETUP_SPUB_DECODE_ERROR #define IOT_ERROR_EASYSETUP_SPUB_DECODE_ERROR (IOT_ERROR_EASYSETUP_BASE - 59)

5.19.1.23 IOT_ERROR_EASYSETUP_WIFI_SCANLIST_NOT_FOUND #define IOT_ERROR_EASYSETUP_WIFI_SCANLIST_NOT_FOUND (IOT_ERROR_EASYSETUP_BASE - 51)

5.19.1.24 IOT_ES_URI_GET_DEVICEINFO #define IOT_ES_URI_GET_DEVICEINFO "/deviceinfo"

5.19.1.25 IOT_ES_URI_GET_LOGS_DUMP #define IOT_ES_URI_GET_LOGS_DUMP "/logs/dump"

5.19.1.26 IOT_ES_URI_GET_LOGS_SYSTEMINFO #define IOT_ES_URI_GET_LOGS_SYSTEMINFO "/logs/systeminfo"

5.19.1.27 IOT_ES_URI_GET_POST_RESPONSE #define IOT_ES_URI_GET_POST_RESPONSE "/post-response"

5.19.1.28 IOT_ES_URI_GET_WIFISCANINFO #define IOT_ES_URI_GET_WIFISCANINFO "/wifiscaninfo"

5.19.1.29 IOT_ES_URI_POST_CONFIRM #define IOT_ES_URI_POST_CONFIRM "confirm"

5.19.1.30 IOT_ES_URI_POST_CONFIRMINFO #define IOT_ES_URI_POST_CONFIRMINFO "confirminfo"

5.19.1.31 IOT_ES_URI_POST_KEYINFO #define IOT_ES_URI_POST_KEYINFO "keyinfo"

5.19.1.32 IOT_ES_URI_POST_LOGS #define IOT_ES_URI_POST_LOGS "logs"

5.19.1.33 IOT_ES_URI_POST_SETUPCOMPLETE #define IOT_ES_URI_POST_SETUPCOMPLETE "setupcomplete"

5.19.1.34 IOT_ES_URI_POST_WIFIPROVISIONINGINFO #define IOT_ES_URI_POST_WIFIPROVISIONINGINFO "wifiprovisioninginfo"

5.19.1.35 IOT_OVF_TYPE_BUTTON #define IOT_OVF_TYPE_BUTTON (1 << OVF_BIT_BUTTON)

5.19.1.36 IOT_OVF_TYPE_JUSTWORKS #define IOT_OVF_TYPE_JUSTWORKS (1 << OVF_BIT_JUSTWORKS)

5.19.1.37 IOT_OVF_TYPE_PIN #define IOT_OVF_TYPE_PIN (1 << OVF_BIT_PIN)

5.19.1.38 IOT_OVF_TYPE_QR #define IOT_OVF_TYPE_QR (1 << OVF_BIT_QR)

5.19.2 Enumeration Type Documentation

5.19.2.1 ownership_validation_feature enum ownership_validation_feature

Enumerator

OVF_BIT_JUSTWORKS
OVF_BIT_QR
OVF_BIT_BUTTON
OVF_BIT_PIN
OVF_BIT_MAX_FEATURE

5.19.3 Function Documentation

5.19.3.1 iot_easystatus_create_ssid() `iot_error_t iot_easystatus_create_ssid (`
 `struct iot_devconf_prov_data * devconf,`
 `char * ssid,`
 `size_t ssid_len)`

Create E4 type SSID.

This function create E4 type soft-ap SSID for this device

Parameters

in	<i>devconf</i>	things static information
out	<i>ssid</i>	created ssid
in	<i>ssid_len</i>	ssid buffer length including null termination

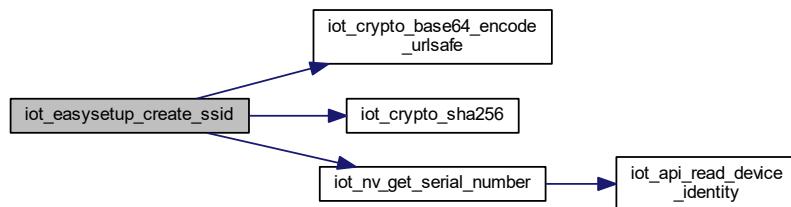
Returns

`iot_state_t`

Return values

<code>IOT_ERROR_NONE</code>	success
-----------------------------	---------

Here is the call graph for this function:



5.19.3.2 iot_easysetup_deinit() `void iot_easysetup_deinit (`
 `struct iot_context * ctx)`

Stop eayssetup device-to-device sequence.

This function stops httpd working

Parameters

in	<code>ctx</code>	<code>iot_context</code> handle
----	------------------	---------------------------------

Returns

`void`

5.19.3.3 iot_easysetup_init() `iot_error_t iot_easysetup_init (`
 `struct iot_context * ctx)`

Start eayssetup device-to-device sequence.

This function makes wifi mode as soft-ap and starts httpd

Parameters

in	<code>ctx</code>	<code>iot_context</code> handle
----	------------------	---------------------------------

Returns

`iot_error_t`

Return values

<code>IOT_ERROR_NONE</code>	success
<code>IOT_ERROR_UNINITIALIZED</code>	error

5.19.3.4 iot_easysetup_request_handler() `iot_error_t iot_easysetup_request_handler (`
 `struct iot_context * ctx,`
 `struct iot_easysetup_payload request)`

easysetup cgi request handler

This function runs from iot-task by executing actual cgi payload manipulation.
result will be transferred to httpd task (tiT) as easysetup response queue parameter.

Parameters

in	<i>ctx</i>	iot_context handle
in	<i>request</i>	easysetup payload as input

Returns

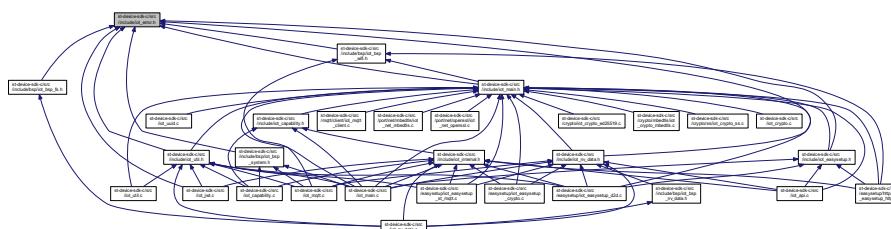
iot_error_t

Return values

IOT_ERROR_NONE	success
----------------	---------

5.20 st-device-sdk-c/src/include/iot_error.h File Reference

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

**iot_error_t**

internal error codes.

- enum `iot_error_t` {

 IOT_ERROR_NONE = 0, IOT_ERROR_BAD_REQ = -1, IOT_ERROR_INVALID_ARGS = -2, IOT_ERROR_WRITE_FAIL = -3,

 IOT_ERROR_READ_FAIL = -4, IOT_ERROR_UNINITIALIZED = -5, IOT_ERROR_MEM_ALLOC = -6,

 IOT_ERROR_TIMEOUT = -7,

 IOT_ERROR_REG_UPDATED = -10, IOT_ERROR_NOT_IMPLEMENTED = -11, IOT_ERROR_INIT_FAIL = -12, IOT_ERROR_DEINIT_FAIL = -13,

 IOT_ERROR_NV_DATA_ERROR = -14, IOT_ERROR_NV_DATA_NOT_EXIST = -15, IOT_ERROR_FS_OPEN_FAIL = -16, IOT_ERROR_FS_READ_FAIL = -17,

 IOT_ERROR_FS_WRITE_FAIL = -18, IOT_ERROR_FS_REMOVE_FAIL = -19, IOT_ERROR_FS_CLOSE_FAIL = -20, IOT_ERROR_FS_NO_FILE = -21,

 IOT_ERROR_FS_ENCRYPT_INIT = -22, IOT_ERROR_FS_ENCRYPT_FAIL = -23, IOT_ERROR_FS_DECRYPT_FAIL = -24, IOT_ERROR_UUID_FAIL = -25,

 IOT_ERROR_MQTT_NETCONN_FAIL = -200, IOT_ERROR_MQTT_CONNECT_FAIL = -201, IOT_ERROR_MQTT_SERVER_FAIL = -202, IOT_ERROR_MQTT_PUBLISH_FAIL = -203,

 IOT_ERROR_MQTT_REJECT_CONNECT = -204, IOT_ERROR_NET_INVALID_INTERFACE = -300, IOT_ERROR_NET_CONNECT = -301, IOT_ERROR_CRYPTO_BASE = -1000,

 IOT_ERROR_JWT_BASE = -2000, IOT_ERROR_PROV_FAIL = -2998, IOT_ERROR_CONNECT_FAIL = -2999, IOT_ERROR_EASYSETUP_BASE = -3000 }

• typedef enum `iot_error_t` `iot_error_t`

5.20.1 Typedef Documentation

5.20.1.1 `iot_error_t` typedef enum iot_error_t iot_error_t

5.20.2 Enumeration Type Documentation

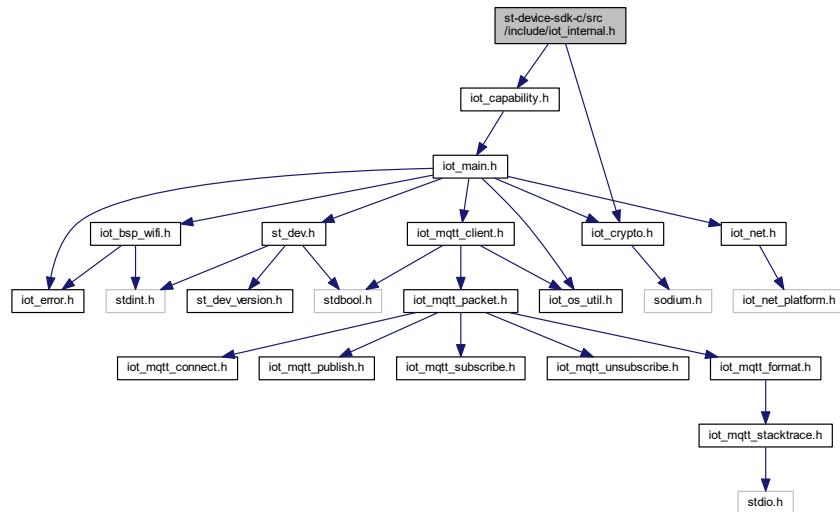
5.20.2.1 `iot_error_t` enum iot_error_t

Enumerator

IOT_ERROR_NONE	
IOT_ERROR_BAD_REQ	
IOT_ERROR_INVALID_ARGS	
IOT_ERROR_WRITE_FAIL	
IOT_ERROR_READ_FAIL	
IOT_ERROR_UNINITIALIZED	
IOT_ERROR_MEM_ALLOC	
IOT_ERROR_TIMEOUT	
IOT_ERROR_REG_UPDATED	
IOT_ERROR_NOT_IMPLEMENTED	
IOT_ERROR_INIT_FAIL	
IOT_ERROR_DEINIT_FAIL	
IOT_ERROR_NV_DATA_ERROR	
IOT_ERROR_NV_DATA_NOT_EXIST	
IOT_ERROR_FS_OPEN_FAIL	
IOT_ERROR_FS_READ_FAIL	
IOT_ERROR_FS_WRITE_FAIL	
IOT_ERROR_FS_REMOVE_FAIL	
IOT_ERROR_FS_CLOSE_FAIL	
IOT_ERROR_FS_NO_FILE	
IOT_ERROR_FS_ENCRYPT_INIT	
IOT_ERROR_FS_ENCRYPT_FAIL	
IOT_ERROR_FS_DECRYPT_FAIL	
IOT_ERROR_UUID_FAIL	
IOT_ERROR_MQTT_NETCONN_FAIL	
IOT_ERROR_MQTT_CONNECT_FAIL	
IOT_ERROR_MQTT_SERVER_UNAVAIL	
IOT_ERROR_MQTT_PUBLISH_FAIL	
IOT_ERROR_MQTT_REJECT_CONNECT	
IOT_ERROR_NET_INVALID_INTERFACE	
IOT_ERROR_NET_CONNECT	
IOT_ERROR_CRYPTO_BASE	
IOT_ERROR_JWT_BASE	
IOT_ERROR_PROV_FAIL	
IOT_ERROR_CONNECT_FAIL	
IOT_ERROR_EASYSETUP_BASE	

5.21 st-device-sdk-c/src/include/iot_internal.h File Reference

```
#include "iot_capability.h"
#include "iot_crypto.h"
Include dependency graph for iot_internal.h:
```



Macros

- #define IOT_TASK_NAME "iot-task"
- #define IOT_TASK_STACK_SIZE (1024*5)
- #define IOT_TASK_PRIORITY (4)
- #define IOT_QUEUE_LENGTH (10)
- #define IOT_PUB_QUEUE_LENGTH (10)
- #define IOT_TOPIC_SIZE (100)
- #define IOT_PAYLOAD_SIZE (1024)
- #define IOT_PUB_TOPIC_REGISTRATION "/v1/registrations"
- #define IOT_SUB_TOPIC_REGISTRATION "/v1/registrations/notification/%s"
- #define IOT_PUB_TOPIC_EVENT "/v1/deviceEvents/%s"
- #define IOT_SUB_TOPIC_COMMAND "/v1/commands/%s"
- #define IOT_SUB_TOPIC_NOTIFICATION "/v1/notifications/%s"
- #define IOT_DEFAULT_TIMEOUT 12000 /* milli-seconds */
- #define IOT_MQTT_KEEPALIVE_INTERVAL 120 /* seconds */

Functions

- `iot_error_t iot_command_send` (struct `iot_context` *ctx, enum `iot_command_type` cmd_type, const void *param, int param_size)

send command to iot main task
- `iot_error_t iot_state_update` (struct `iot_context` *ctx, `iot_state_t` new_state, int need_interact)

update iot state
- `iot_error_t iot_easysetup_request` (struct `iot_context` *ctx, enum `iot_easysetup_step` step, const void *payload)

send easysetup cgi payload manipulation request
- `iot_error_t iot_api_onboarding_config_load` (unsigned char *onboarding_config, unsigned int onboarding_config_len, struct `iot_devconf_prov_data` *devconf)

load "onboarding_config.json" from application source directory
- `iot_error_t iot_api_device_info_load` (unsigned char *device_info, unsigned int device_info_len, struct `iot_device_info` *info)

load "device_info.json" from application source directory
- `void iot_api_onboarding_config_mem_free` (struct `iot_devconf_prov_data` *devconf)

free onboarding config memory
- `void iot_api_device_info_mem_free` (struct `iot_device_info` *info)

free device info memory
- `void iot_api_prov_data_mem_free` (struct `iot_device_prov_data` *prov)

free prov data memory
- `iot_error_t iot_api_read_device_identity` (unsigned char *device_nv_info, unsigned int device_nv_info_len, const char *object, char **nv_data)

Extract required data from "device_info.json" which is located in application source directory.
- `iot_error_t iot_device_cleanup` (struct `iot_context` *ctx)

device cleanup
- `iot_error_t iot_es_connect` (struct `iot_context` *ctx, int conn_type)

easy setup connect
- `iot_error_t iot_es_disconnect` (struct `iot_context` *ctx, int conn_type)

easy setup disconnect
- `void iot_es_crypto_init_pk` (`iot_crypto_pk_info_t` *pk_info, `iot_crypto_pk_type_t` type)

initialize the buffer for pubkey information
- `iot_error_t iot_es_crypto_load_pk` (`iot_crypto_pk_info_t` *pk_info)

easy setup crypto to load pubkey
- `void iot_es_crypto_free_pk` (`iot_crypto_pk_info_t` *pk_info)

easy setup crypto to free pubkey
- `iot_error_t iot_mqtt_connect` (struct `iot_mqtt_ctx` *target_cli, char *username, char *sign_data)

mqtt connect
- `iot_error_t iot_mqtt_publish` (struct `iot_context` *ctx, void *payload)

publish mqtt message
- `iot_error_t iot_mqtt_subscribe` (struct `iot_mqtt_ctx` *mqtt_ctx)

subscribe mqtt topic
- `iot_error_t iot_mqtt_unsubscribe` (struct `iot_mqtt_ctx` *mqtt_ctx)

unsubscribe mqtt topic
- `void iot_mqtt_disconnect` (struct `iot_mqtt_ctx` *target_cli)

mqtt disconnect
- `iot_error_t iot_mqtt_registration` (struct `iot_mqtt_ctx` *mqtt_ctx)

register device to ST server
- `void iot_cap_sub_cb` (`iot_cap_handle_list_t` *cap_handle_list, char *payload)

callback for mqtt command msg
- `void iot_noti_sub_cb` (struct `iot_context` *ctx, char *payload)

- callback for mqtt noti msg*
- void `iot_cap_call_init_cb (iot_cap_handle_list_t *cap_handle_list)`
call init callback
 - `iot_error_t iot_get_time_in_sec (char *buf, size_t buf_len)`
get time data by sec
 - `iot_error_t iot_get_time_in_ms (char *buf, size_t buf_len)`
get time data in msec

5.21.1 Macro Definition Documentation

5.21.1.1 IOT_DEFAULT_TIMEOUT `#define IOT_DEFAULT_TIMEOUT 12000 /* milli-seconds */`

5.21.1.2 IOT_MQTT_KEEPALIVE_INTERVAL `#define IOT_MQTT_KEEPALIVE_INTERVAL 120 /* seconds */`

5.21.1.3 IOT_PAYLOAD_SIZE `#define IOT_PAYLOAD_SIZE (1024)`

5.21.1.4 IOT_PUB_QUEUE_LENGTH `#define IOT_PUB_QUEUE_LENGTH (10)`

5.21.1.5 IOT_PUB_TOPIC_EVENT `#define IOT_PUB_TOPIC_EVENT "/v1/deviceEvents/%s"`

5.21.1.6 IOT_PUB_TOPIC_REGISTRATION `#define IOT_PUB_TOPIC_REGISTRATION "/v1/registrations"`

5.21.1.7 IOT_QUEUE_LENGTH `#define IOT_QUEUE_LENGTH (10)`

5.21.1.8 IOT_SUB_TOPIC_COMMAND `#define IOT_SUB_TOPIC_COMMAND "/v1/commands/%s"`

5.21.1.9 IOT_SUB_TOPIC_NOTIFICATION #define IOT_SUB_TOPIC_NOTIFICATION "/v1/notifications/%s"

5.21.1.10 IOT_SUB_TOPIC_REGISTRATION #define IOT_SUB_TOPIC_REGISTRATION "/v1/registrations/notification/%s"

5.21.1.11 IOT_TASK_NAME #define IOT_TASK_NAME "iot-task"

5.21.1.12 IOT_TASK_PRIORITY #define IOT_TASK_PRIORITY (4)

5.21.1.13 IOT_TASK_STACK_SIZE #define IOT_TASK_STACK_SIZE (1024*5)

5.21.1.14 IOT_TOPIC_SIZE #define IOT_TOPIC_SIZE (100)

5.21.2 Function Documentation

5.21.2.1 iot_api_device_info_load() iot_error_t iot_api_device_info_load (unsigned char * device_info, unsigned int device_info_len, struct iot_device_info * info)

load "device_info.json" from application source directory

"device_info.json" should be updated by application developer

This function parses downloaded "device_info.json" to be used for EasySetup

Only firmwareVersion will be parsed by this api. others are handled by another api

Parameters

in	device_info	start pointer of json data
in	device_info_len	json data length
out	info	"device_info.json" will be parsed and mapped to this internal structure

Return values

IOT_ERROR_NONE	success.
IOT_ERROR_UNINITIALIZED	invalid json value.

Return values

<i>IOT_ERROR_MEM_ALLOC</i>	memory allocation failure.
----------------------------	----------------------------

example

```
{
    "deviceInfo": {
        "firmwareVersion": "FwVer0011A",
        ...
    }
}
```

Here is the caller graph for this function:



5.21.2.2 iot_api_device_info_mem_free()

```
void iot_api_device_info_mem_free (
    struct iot_device_info * info )
```

free device info memory

this function frees the loaded device's information

Parameters

in	<i>info</i>	loaded device's information
----	-------------	-----------------------------

5.21.2.3 iot_api_onboarding_config_load()

```
iot_error_t iot_api_onboarding_config_load (
    unsigned char * onboarding_config,
    unsigned int onboarding_config_len,
    struct iot_devconf_prov_data * devconf )
```

load "onboarding_config.json" from application source directory

"onboarding_config.json" can be downloaded from SmartThings Developer Workspace
This function parses downloaded "onboarding_config.json" to be used for EasySetup

Parameters

in	<i>onboarding_config</i>	start pointer of json data
in	<i>onboarding_config_len</i>	json data length
out	<i>devconf</i>	"onboarding_config.json" will be parsed and mapped to this internal structure

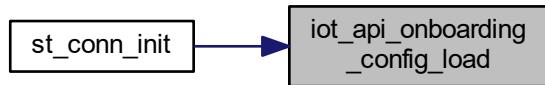
Return values

<i>IOT_ERROR_NONE</i>	success.
<i>IOT_ERROR_UNINITIALIZED</i>	invalid json value.
<i>IOT_ERROR_MEM_ALLOC</i>	memory allocation failure.
<i>IOT_ERROR_CRYPTO_SHA256</i>	sha256 error.
<i>IOT_ERROR_CRYPTO_BASE64</i>	base64 error.
<i>IOT_ERROR_CRYPTO_BASE64_URLSAFE</i>	base64 urlsafe error.

example

```
{
    "onboardingConfig": {
        "deviceOnboardingID": "NAME", // max. 13 character. this will be prefix of soft-ap ssid.
        "mnId": "MNID", // mnId for developer and/or manufacturer. "MNID" shouldn't be used.
        "setupId": "999", // 3-digit Device onboarding ID for this device.
        "vid": "VID", // VID(Vendor ID) for this profile.
        "deviceTypeID": "TYPE", // Device type which is selected from Developer Workspace.
        "ownershipValidationType": [ "JUSTWORKS", "BUTTON", "PIN", "QR" ],
        // "JUSTWORKS" for confirming without user interaction.
        // "BUTTON" for confirming by pressing builtin button.
        // "PIN" for confirming by matching 8-digit number PIN
        // "QR" for confirming by scanning a QR code by SmartThings app.
        "identityType": "ED25519 or CERTIFICATE" // ED25519 or X.509 CERTIFICATE
    }
}
```

Here is the caller graph for this function:



5.21.2.4 iot_api_onboarding_config_mem_free() void iot_api_onboarding_config_mem_free (struct iot_devconf_prov_data * devconf)

free onboarding config memory

this function frees the loaded onboarding configuration

Parameters

in	<i>devconf</i>	loaded onboarding configuration
----	----------------	---------------------------------

5.21.2.5 iot_api_prov_data_mem_free() void iot_api_prov_data_mem_free (

```
    struct iot_device_prov_data * prov )
```

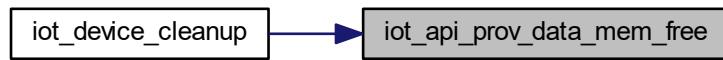
free prov data memory

this function frees the loaded provisioning data

Parameters

in	<i>prov</i>	loaded provisioning data
----	-------------	--------------------------

Here is the caller graph for this function:



5.21.2.6 iot_api_read_device_identity() `iot_error_t iot_api_read_device_identity (`
 `unsigned char * device_nv_info,`
 `unsigned int device_nv_info_len,`
 `const char * object,`
 `char ** nv_data)`

Extract required data from "device_info.json" which is located in application source directory.

"device_info.json" should be updated by application developer

This function parses downloaded "device_info.json" to be used for EasySetup

Parameters

in	<i>device_nv_info</i>	starting pointer of json data
in	<i>device_nv_info_len</i>	json data length
in	<i>object</i>	object name for searching json data.
out	<i>nv_data</i>	"device_info.json" will be parsed by "object" and mapped to this pointer

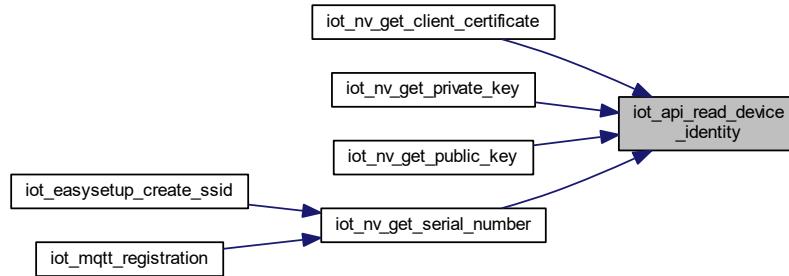
Return values

<i>IOT_ERROR_NONE</i>	success.
<i>IOT_ERROR_UNINITIALIZED</i>	invalid json value.
<i>IOT_ERROR_MEM_ALLOC</i>	memory allocation failure.

example

```
{
  "nvProfile": {
    "privateKey": "privateKey", // Client (= Device) Private key
    "publicKey": "publicKey", // Client (= Device) Public key
    "serialNumber": "serialNumber" // Device Serial Number
  }
}
```

Here is the caller graph for this function:



5.21.2.7 `iot_cap_call_init_cb()` `void iot_cap_call_init_cb (`
 `iot_cap_handle_list_t * cap_handle_list)`

call init callback

this function is used to call all allocated capability callbacks when target is connected

Parameters

in	<code>cap_handle_list</code>	allocated capability handle list
----	------------------------------	----------------------------------

5.21.2.8 `iot_cap_sub_cb()` `void iot_cap_sub_cb (`
 `iot_cap_handle_list_t * cap_handle_list,`
 `char * payload)`

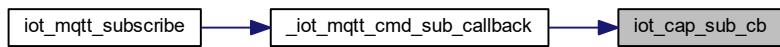
callback for mqtt command msg

this function is used to handle command message from server

Parameters

in	<code>cap_handle_list</code>	allocated capability handle list
in	<code>payload</code>	received raw message from server

Here is the caller graph for this function:



5.21.2.9 iot_command_send() `iot_error_t iot_command_send (`
 `struct iot_context * ctx,`
 `enum iot_command_type cmd_type,`
 `const void * param,`
 `int param_size)`

send command to iot main task

this function sends specific command to iot-task via queue

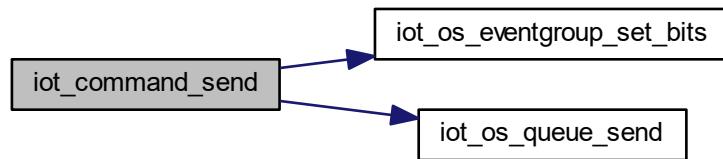
Parameters

in	<i>ctx</i>	iot-core context
in	<i>cmd_type</i>	actual specific command type
in	<i>param</i>	additional parameter data for each command
in	<i>param_size</i>	additional parameter size

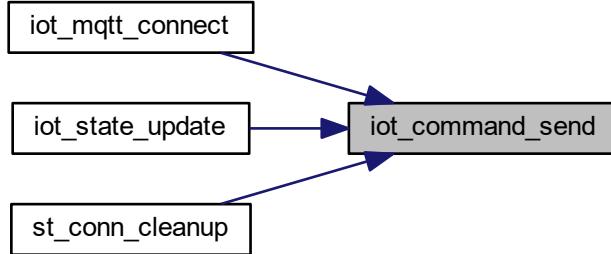
Return values

<i>IOT_ERROR_NONE</i>	success.
<i>IOT_ERROR_MEM_ALLOC</i>	memory allocation failed
<i>IOT_ERROR_BAD_REQ</i>	queue send error

Here is the call graph for this function:



Here is the caller graph for this function:



5.21.2.10 iot_device_cleanup() `iot_error_t iot_device_cleanup (struct iot_context * ctx)`

device cleanup

this function triggers clean-up process. All registered data will be removed

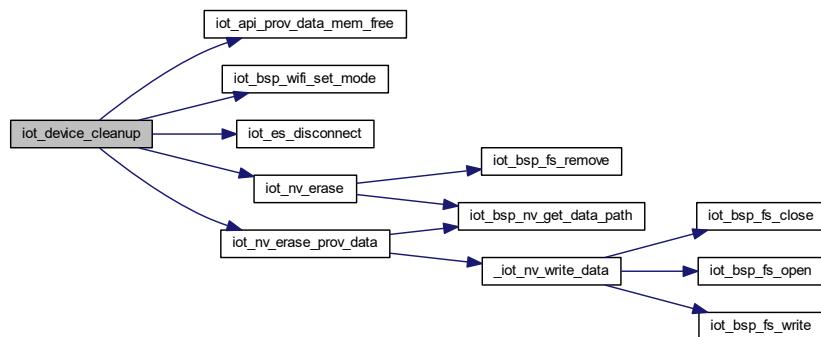
Parameters

in	<code>ctx</code>	iot-core context
----	------------------	------------------

Return values

<code>IOT_ERROR_NONE</code>	success.
-----------------------------	----------

Here is the call graph for this function:



```
5.21.2.11 iot_easysetup_request() iot_error_t iot_easysetup_request (
    struct iot_context * ctx,
    enum iot_easysetup_step step,
    const void * payload )
```

send easysetup cgi payload manipulation request

easysetup cgi payload manipulation should be done at iot-task. This function sends payload to iot-task via queue

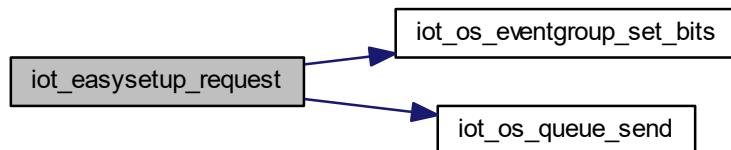
Parameters

in	<i>ctx</i>	iot-core context
in	<i>step</i>	indicates which uri(command) is dealing with
in	<i>payload</i>	payload data - mostly json data

Return values

<i>IOT_ERROR_NONE</i>	success.
<i>IOT_ERROR_BAD_REQ</i>	queue send error

Here is the call graph for this function:



```
5.21.2.12 iot_es_connect() iot_error_t iot_es_connect (
    struct iot_context * ctx,
    int conn_type )
```

easy setup connect

this function tries to connect server for registration or communication process

Parameters

in	<i>ctx</i>	iot-core context
in	<i>conn_type</i>	set connection type. registration or communication with server

Return values

<i>IOT_ERROR_NONE</i>	success.
-----------------------	----------

5.21.2.13 *iot_es_crypto_free_pk()* `void iot_es_crypto_free_pk (`
`iot_crypto_pk_info_t * pk_info)`

easy setup crypto to free pubkey

this function frees the loaded pubkey information

Parameters

<i>in</i>	<i>pk_info</i>	loaded pubkey information pointer
-----------	----------------	-----------------------------------

5.21.2.14 *iot_es_crypto_init_pk()* `void iot_es_crypto_init_pk (`
`iot_crypto_pk_info_t * pk_info,`
`iot_crypto_pk_type_t type)`

initialize the buffer for pubkey information

this function uses to initialize

Parameters

<i>in</i>	<i>pk_info</i>	pubkey information structure to initialize
<i>in</i>	<i>type</i>	pubkey type, RSA or ED25519

5.21.2.15 *iot_es_crypto_load_pk()* `iot_error_t iot_es_crypto_load_pk (`
`iot_crypto_pk_info_t * pk_info)`

easy setup crypto to load pubkey

this function loads pubkey information

Parameters

<i>in</i>	<i>pk_info</i>	pubkey information pointer for loading
-----------	----------------	--

Return values

<i>IOT_ERROR_NONE</i>	success.
-----------------------	----------

```
5.21.2.16 iot_es_disconnect() iot_error_t iot_es_disconnect (
    struct iot_context * ctx,
    int conn_type )
```

easy setup disconnect

this function tries to disconnect server for registration or communication process

Parameters

in	<i>ctx</i>	iot-core context
in	<i>conn_type</i>	set connection type. registration or communication with server

Return values

<i>IOT_ERROR_NONE</i>	success.
-----------------------	----------

Here is the caller graph for this function:



```
5.21.2.17 iot_get_time_in_ms() iot_error_t iot_get_time_in_ms (
    char * buf,
    size_t buf_len )
```

get time data in msec

this function tries to get time value in millisecond by string

Parameters

in	<i>buf</i>	buffer point to contain millisecond based string value
in	<i>buf_len</i>	size of allocated buffer for string

Return values

<i>IOT_ERROR_NONE</i>	success.
-----------------------	----------

```
5.21.2.18 iot_get_time_in_sec() iot_error_t iot_get_time_in_sec (
    char * buf,
    size_t buf_len )
```

get time data by sec

this function tries to get time value in second by string

Parameters

in	<i>buf</i>	buffer point to contain second based string value
in	<i>buf_len</i>	size of allocated buffer for string

Return values

<i>IOT_ERROR_NONE</i>	success.
-----------------------	----------

```
5.21.2.19 iot_mqtt_connect() iot_error_t iot_mqtt_connect (
    struct iot_mqtt_ctx * target_cli,
    char * username,
    char * sign_data )
```

mqtt connect

this function connects ST server by mqtt

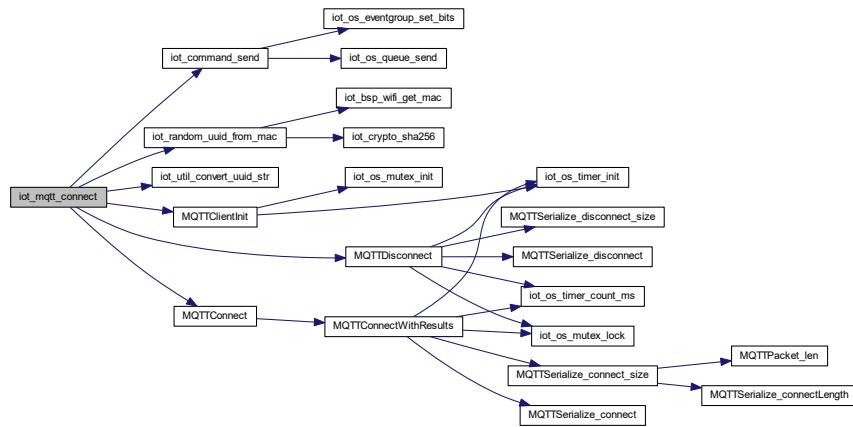
Parameters

in	<i>target_cli</i>	mqtt handling context
in	<i>username</i>	username to connect ST server based on mqtt protocol
in	<i>sign_data</i>	specific password that was jwt token-based to connect ST server

Return values

<i>IOT_ERROR_NONE</i>	success.
-----------------------	----------

Here is the call graph for this function:



5.21.2.20 iot_mqtt_disconnect() void iot_mqtt_disconnect (struct iot_mqtt_ctx * target_cli)

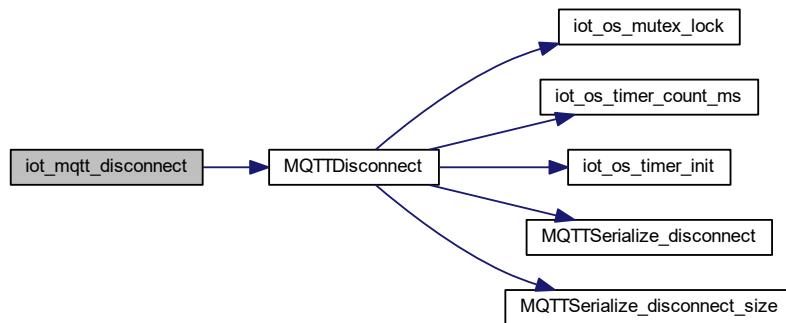
mqtt disconnect

this function is used to disconnect from server

Parameters

in	<i>target_cli</i>	mqtt handling context
----	-------------------	-----------------------

Here is the call graph for this function:



5.21.2.21 iot_mqtt_publish() `iot_error_t iot_mqtt_publish (`
 `struct iot_context * ctx,`
 `void * payload)`

publish mqtt message

this function is used to publish command & notification message

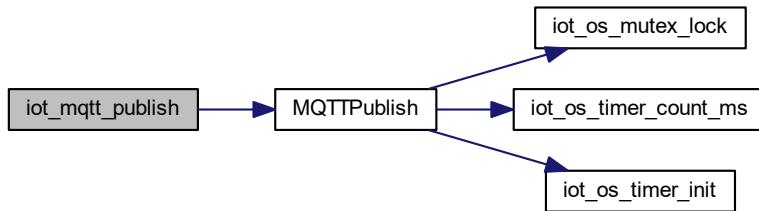
Parameters

in	<i>ctx</i>	iot-core context
in	<i>payload</i>	raw message sending to server

Return values

<i>IOT_ERROR_NONE</i>	success.
-----------------------	----------

Here is the call graph for this function:



5.21.2.22 iot_mqtt_registration() `iot_error_t iot_mqtt_registration (`
 `struct iot_mqtt_ctx * mqtt_ctx)`

register device to ST server

this function is used to handle the registration process

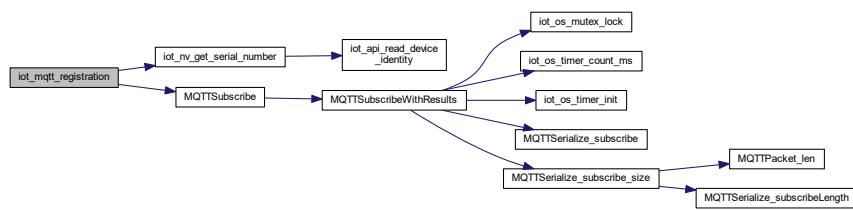
Parameters

in	<i>mqtt_ctx</i>	mqtt handling context
----	-----------------	-----------------------

Return values

<i>IOT_ERROR_NONE</i>	success.
-----------------------	----------

Here is the call graph for this function:



5.21.2.23 iot_mqtt_subscribe() *iot_error_t iot_mqtt_subscribe (struct iot_mqtt_ctx * mqtt_ctx)*

subscribe mqtt topic

this function is used to subscribe command & notification topics

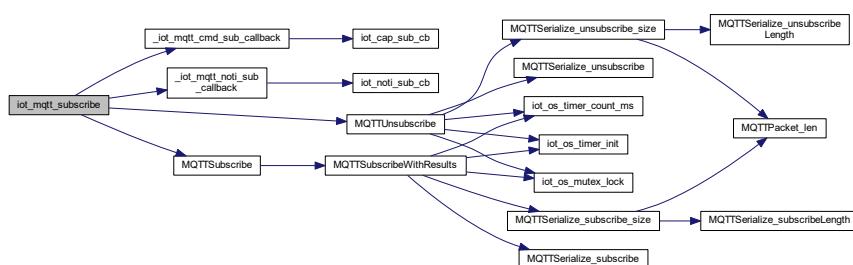
Parameters

in	<i>mqtt_ctx</i>	mqtt handling context
----	-----------------	-----------------------

Return values

<i>IOT_ERROR_NONE</i>	success.
-----------------------	----------

Here is the call graph for this function:



5.21.2.24 iot_mqtt_unsubscribe() *iot_error_t iot_mqtt_unsubscribe (struct iot_mqtt_ctx * mqtt_ctx)*

unsubscribe mqtt topic

this function is used to unsubscribe command & notification topics

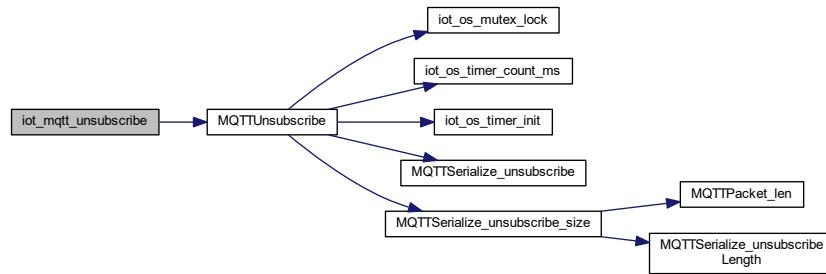
Parameters

in	<i>mqtt_ctx</i>	mqtt handling context
----	-----------------	-----------------------

Return values

<i>IOT_ERROR_NONE</i>	success.
-----------------------	----------

Here is the call graph for this function:



5.21.2.25 `iot_noti_sub_cb()` `void iot_noti_sub_cb (`
 `struct iot_context * ctx,`
 `char * payload)`

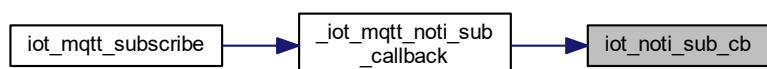
callback for mqtt noti msg

this function is used to handle notification message from server

Parameters

in	<i>ctx</i>	iot-core context
in	<i>payload</i>	received raw message from server

Here is the caller graph for this function:



```
5.21.2.26 iot_state_update() iot_error_t iot_state_update (
    struct iot_context * ctx,
    iot_state_t new_state,
    int need_interact )
```

update iot state

this function tries to update iot-state using iot_command_send internally

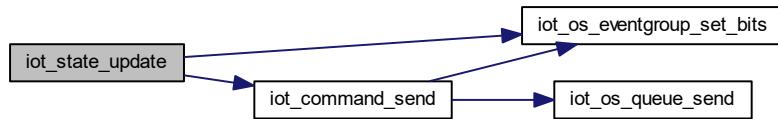
Parameters

in	<i>ctx</i>	iot-core context
in	<i>new_state</i>	new iot-state to update
in	<i>need_interact</i>	additional parameter data for each command

Return values

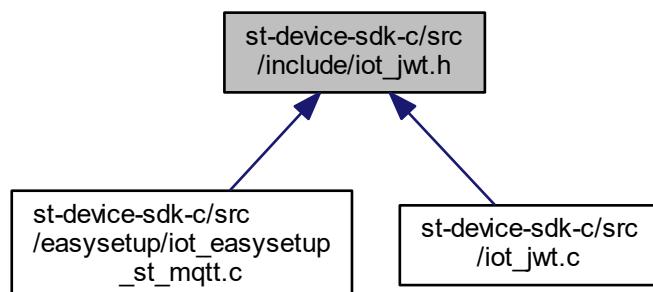
<i>IOT_ERROR_NONE</i>	success.
<i>IOT_ERROR_MEM_ALLOC</i>	memory allocation failed
<i>IOT_ERROR_BAD_REQ</i>	queue send error

Here is the call graph for this function:



5.22 st-device-sdk-c/src/include/iot_jwt.h File Reference

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



Macros

- #define IOT_ERROR_JWT_INVALID_ARG (IOT_ERROR_JWT_BASE - 0)
- #define IOT_ERROR_JWT_MALLOC (IOT_ERROR_JWT_BASE - 1)
- #define IOT_ERROR_JWT_CJSON (IOT_ERROR_JWT_BASE - 2)

Functions

- `iot_error_t iot_jwt_create (char **token, const char *sn, iot_crypto_pk_info_t *pk_info)`
Create a JWT as proof of the device's identity.

5.22.1 Macro Definition Documentation

5.22.1.1 IOT_ERROR_JWT_CJSON #define IOT_ERROR_JWT_CJSON (IOT_ERROR_JWT_BASE - 2)

5.22.1.2 IOT_ERROR_JWT_INVALID_ARG #define IOT_ERROR_JWT_INVALID_ARG (IOT_ERROR_JWT_BASE - 0)

5.22.1.3 IOT_ERROR_JWT_MALLOC #define IOT_ERROR_JWT_MALLOC (IOT_ERROR_JWT_BASE - 1)

5.22.2 Function Documentation

5.22.2.1 `iot_jwt_create()` `iot_error_t iot_jwt_create (` `char ** token,` `const char * sn,` `iot_crypto_pk_info_t * pk_info)`

Create a JWT as proof of the device's identity.

This function makes a JWT string to connect to ST Cloud. Pass the JWT as password. Supported key types are RS256 and ED25519.

Parameters

out	<code>token</code>	a pointer of buffer to store a formatted and signed string
in	<code>sn</code>	device serial number as user name
in	<code>pk_info</code>	private key data

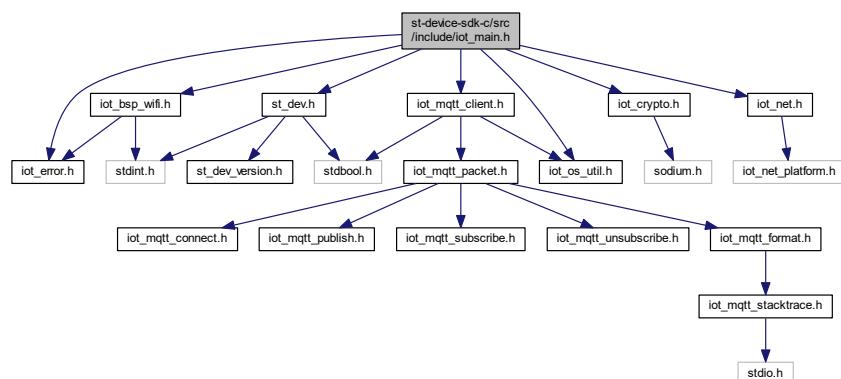
Return values

<i>IOT_ERROR_NONE</i>	JWT is sucessfully generated
<i>IOT_ERROR_JWT_MALLOC</i>	no more available heap memory
<i>IOT_ERROR_JWT_CJSON</i>	failed to make json

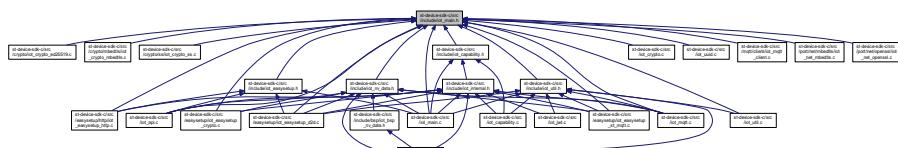
5.23 st-device-sdk-c/src/include/iot_main.h File Reference

```
#include "st_dev.h"
#include "iot_error.h"
#include "iot_bsp_wifi.h"
#include "iot_os_util.h"
#include "iot_crypto.h"
#include "iot_net.h"
#include "iot_mqtt_client.h"

Include dependency graph for iot_main.h:
```



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



Data Structures

- struct `iot_uuid`
Contains "uuid" data.
- struct `iot_wifi_prov_data`
Contains "wifi provisioning" data.
- struct `iot_cloud_prov_data`
Contains "cloud provisioning" data.

- struct `iot_devconf_prov_data`
Contains "device configuration" data.
- struct `iot_device_prov_data`
Contains "all device's provisioning" data.
- struct `iot_command`
Contains "internal command" data.
- struct `iot_easysetup_payload`
Contains "easy-setup payload" data.
- struct `iot_registered_data`
Contains "registration message" data.
- struct `iot_mqtt_ctx`
Contains "mqtt handling context" data.
- struct `iot_device_info`
Contains "device's information" data.
- struct `iot_state_data`
Contains "iot core's main state" data.
- struct `iot_context`
Contains "iot core's main context" data.

Macros

- `#define IOT_WIFI_PROV_SSID_LEN (31 + 1)`
- `#define IOT_WIFI_PROV_PASSWORD_LEN (63 + 1)`
- `#define IOT_BUF_TX_SIZE 3500`
- `#define IOT_BUF_RX_SIZE 512`
- `#define IOT_EVENT_BIT_COMMAND (1 << 0)`
- `#define IOT_EVENT_BIT_CAPABILITY (1 << 1)`
- `#define IOT_EVENT_BIT_EASYSETUP_REQ (1 << 2)`
- `#define IOT_EVENT_BIT_EASYSETUP_RESP (1 << 3)`
- `#define IOT_EVENT_BIT_EASYSETUP_CONFIRM (1 << 4)`
- `#define IOT_EVENT_BIT_ALL (IOT_EVENT_BIT_COMMAND | IOT_EVENT_BIT_CAPABILITY | IOT_EVENT_BIT_EASYSETUP_REQ)`
- `#define IOT_REG_UUID_STR_LEN (36)`

Typedefs

- `typedef enum iot_state_type iot_state_t`
- `typedef struct iot_cap_handle_list iot_cap_handle_list_t`

Enumerations

- `enum _iot_noti_type {
 _IOT_NOTI_TYPE_UNKNOWN = IOT_NOTI_TYPE_UNKNOWN, _IOT_NOTI_TYPE_DEV_DELETED =
 IOT_NOTI_TYPE_DEV_DELETED, _IOT_NOTI_TYPE_RATE_LIMIT = IOT_NOTI_TYPE_RATE_LIMIT,
 _IOT_NOTI_TYPE_QUOTA_REACHED = IOT_NOTI_TYPE_QUOTA_REACHED,
 _IOT_NOTI_TYPE_JWT_EXPIRED }`
- `enum iot_command_type {
 IOT_COMMAND_READY_TO_CTL, IOT_COMMAND_NETWORK_MODE, IOT_COMMAND_CHECK_PROV_STATUS,
 IOT_COMMAND_SELF_CLEANUP,
 IOT_COMMAND_CHECK_CLOUD_STATE, IOT_COMMAND_CLOUD_REGISTERING, IOT_COMMAND_CLOUD_REGISTE
 IOT_COMMAND_CLOUD_CONNECTING,
 IOT_COMMAND_NOTIFICATION_RECEIVED, IOT_COMMAND_TYPE_MAX, IOT_CMD_STATE_HANDLE
}`

- enum `iot_easysetup_step` {
 IOT_EASYSETUP_STEP_DEVICEINFO, IOT_EASYSETUP_STEP_KEYINFO, IOT_EASYSETUP_STEP_CONFIRMINFO,
 IOT_EASYSETUP_STEP_CONFIRM,
 IOT_EASYSETUP_STEP_WIFISCANINFO, IOT_EASYSETUP_STEP_WIFIPROVISIONINGINFO, IOT_EASYSETUP_STEP_S
 IOT_EASYSETUP_STEP_LOG_SYSTEMINFO,
 IOT_EASYSETUP_STEP_LOG_CREATE_DUMP, IOT_EASYSETUP_STEP_LOG_GET_DUMP }
- enum `iot_connect_type` { IOT_CONNECT_TYPE_REGISTRATION, IOT_CONNECT_TYPE_COMMUNICATION }
- enum `iot_state_type` {
 IOT_STATE_CHANGE_FAILED = -2, IOT_STATE_UNKNOWN = -1, IOT_STATE_INITIALIZED = 0,
 IOT_STATE_PROV_ENTER,
 IOT_STATE_PROV_CONFIRMING, IOT_STATE_PROV_DONE, IOT_STATE_CLOUD_DISCONNECTED,
 IOT_STATE_CLOUD_REGISTERING,
 IOT_STATE_CLOUD_CONNECTING, IOT_STATE_CLOUD_CONNECTED }
- enum `iot_state_opt` { IOT_STATE_OPT_NONE, IOT_STATE_OPT_NEED_INTERACT }

5.23.1 Macro Definition Documentation

5.23.1.1 IOT_BUF_RX_SIZE #define IOT_BUF_RX_SIZE 512

5.23.1.2 IOT_BUF_TX_SIZE #define IOT_BUF_TX_SIZE 3500

5.23.1.3 IOT_EVENT_BIT_ALL #define IOT_EVENT_BIT_ALL (IOT_EVENT_BIT_COMMAND | IOT_EVENT_BIT_CAPABILITY
| IOT_EVENT_BIT_EASYSETUP_REQ)

5.23.1.4 IOT_EVENT_BIT_CAPABILITY #define IOT_EVENT_BIT_CAPABILITY (1 << 1)

5.23.1.5 IOT_EVENT_BIT_COMMAND #define IOT_EVENT_BIT_COMMAND (1 << 0)

5.23.1.6 IOT_EVENT_BIT_EASYSETUP_CONFIRM #define IOT_EVENT_BIT_EASYSETUP_CONFIRM (1 <<
4)

5.23.1.7 IOT_EVENT_BIT_EASYSETUP_REQ #define IOT_EVENT_BIT_EASYSETUP_REQ (1 << 2)

5.23.1.8 IOT_EVENT_BIT_EASYSETUP_RESP #define IOT_EVENT_BIT_EASYSETUP_RESP (1 << 3)

5.23.1.9 IOT_REG_UUID_STR_LEN #define IOT_REG_UUID_STR_LEN (36)

5.23.1.10 IOT_WIFI_PROV_PASSWORD_LEN #define IOT_WIFI_PROV_PASSWORD_LEN (63 + 1)

5.23.1.11 IOT_WIFI_PROV_SSID_LEN #define IOT_WIFI_PROV_SSID_LEN (31 + 1)

5.23.2 Typedef Documentation

5.23.2.1 iot_cap_handle_list_t typedef struct iot_cap_handle_list iot_cap_handle_list_t

5.23.2.2 iot_state_t typedef enum iot_state_type iot_state_t

5.23.3 Enumeration Type Documentation

5.23.3.1 _iot_noti_type enum _iot_noti_type

Enumerator

_IOT_NOTI_TYPE_UNKNOWN
_IOT_NOTI_TYPE_DEV_DELETED
_IOT_NOTI_TYPE_RATE_LIMIT
_IOT_NOTI_TYPE_QUOTA_REACHED
_IOT_NOTI_TYPE_JWT_EXPIRED

5.23.3.2 iot_command_type enum iot_command_type

Enumerator

IOT_COMMAND_READY_TO_CTL	
IOT_COMMAND_NETWORK_MODE	
IOT_COMMAND_CHECK_PROV_STATUS	
IOT_COMMAND_SELF_CLEANUP	
IOT_COMMAND_CHECK_CLOUD_STATE	
IOT_COMMAND_CLOUD_REGISTERING	
IOT_COMMAND_CLOUD_REGISTERED	
IOT_COMMAND_CLOUD_CONNECTING	
IOT_COMMAND_NOTIFICATION_RECEIVED	
IOT_COMMAND_TYPE_MAX	
IOT_CMD_STATE_HANDLE	

5.23.3.3 iot_connect_type enum `iot_connect_type`

Enumerator

IOT_CONNECT_TYPE_REGISTRATION	
IOT_CONNECT_TYPE_COMMUNICATION	

5.23.3.4 iot_easysetup_step enum `iot_easysetup_step`

Enumerator

IOT_EASYSETUP_STEP_DEVICEINFO	
IOT_EASYSETUP_STEP_KEYINFO	
IOT_EASYSETUP_STEP_CONFIRMINFO	
IOT_EASYSETUP_STEP_CONFIRM	
IOT_EASYSETUP_STEP_WIFISCANINFO	
IOT_EASYSETUP_STEP_WIFIPROVISIONINGINFO	
IOT_EASYSETUP_STEP_SETUPCOMPLETE	
IOT_EASYSETUP_STEP_LOG_SYSTEMINFO	
IOT_EASYSETUP_STEP_LOG_CREATE_DUMP	
IOT_EASYSETUP_STEP_LOG_GET_DUMP	

5.23.3.5 iot_state_opt enum `iot_state_opt`

Enumerator

IOT_STATE_OPT_NONE	
IOT_STATE_OPT_NEED_INTERACT	

5.23.3.6 iot_state_type enum iot_state_type

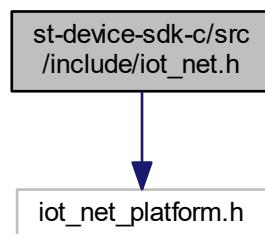
Enumerator

IOT_STATE_CHANGE_FAILED
IOT_STATE_UNKNOWN
IOT_STATE_INITIALIZED
IOT_STATE_PROV_ENTER
IOT_STATE_PROV_CONFIRMING
IOT_STATE_PROV_DONE
IOT_STATE_CLOUD_DISCONNECTED
IOT_STATE_CLOUD_REGISTERING
IOT_STATE_CLOUD_CONNECTING
IOT_STATE_CLOUD_CONNECTED

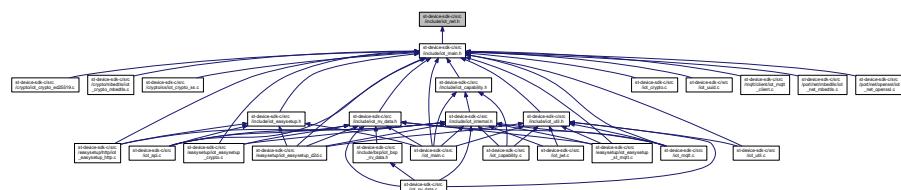
5.24 st-device-sdk-c/src/include/iot_net.h File Reference

```
#include "iot_net_platform.h"
```

Include dependency graph for iot_net.h:



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



Data Structures

- struct `iot_net_connection`
Contains server related information.
- struct `iot_net_interface`
Contains "network management structure" data.

TypeDefs

- typedef struct `iot_net_interface` `iot_net_interface_t`
Contains "network management structure" data.
- typedef struct `iot_net_connection` `iot_net_connection_t`
Contains server related information.

Functions

- `iot_error_t iot_net_init (iot_net_interface_t *net)`
Initialize the network structure for SSL connection.

5.24.1 TypeDef Documentation

5.24.1.1 `iot_net_connection_t` `typedef struct iot_net_connection iot_net_connection_t`

Contains server related information.

This structure has address, port and certificate of server.

5.24.1.2 `iot_net_interface_t` `typedef struct iot_net_interface iot_net_interface_t`

Contains "network management structure" data.

5.24.2 Function Documentation

5.24.2.1 `iot_net_init()` `iot_error_t iot_net_init (` `iot_net_interface_t * net)`

Initialize the network structure for SSL connection.

Parameters

<code>n</code>	- <code>iot_net_interface</code> structure
----------------	--

Returns

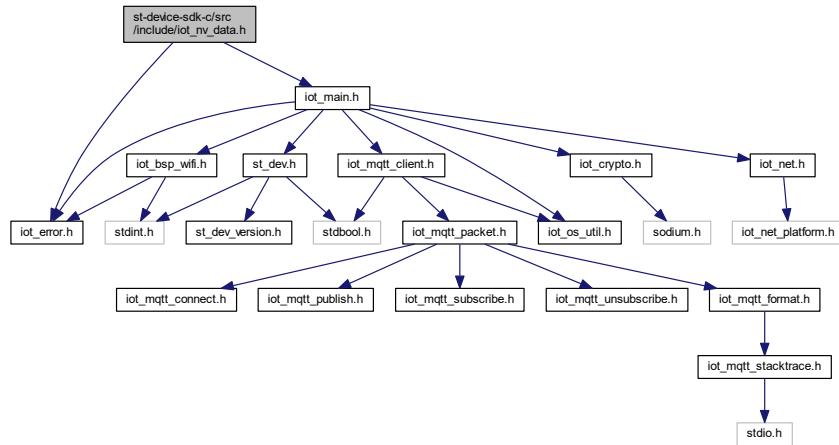
```
iot_error_t
```

Return values

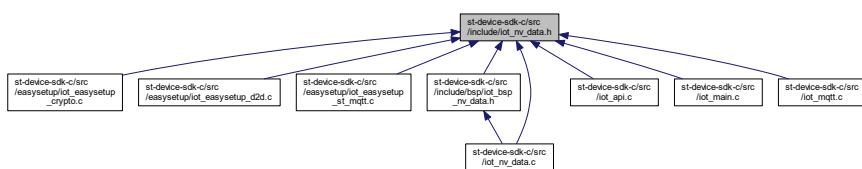
<i>IOT_ERROR_NONE</i>	success
<i>IOT_ERROR_NET_INVALID_INTERFACE</i>	error

5.25 st-device-sdk-c/src/include/iot_nv_data.h File Reference

```
#include "iot_error.h"
#include "iot_main.h"
Include dependency graph for iot_nv_data.h:
```



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



iot_nvd_t

internal nv data codes.

- enum `iot_nvd_t` {
 `IOT_NVD_WIFI_PROV_STATUS` = 0, `IOT_NVD_AP_SSID`, `IOT_NVD_AP_PASS`, `IOT_NVD_AP_BSSID`, `IOT_NVD_AP_AUTH_TYPE`, `IOT_NVD_CLOUD_PROV_STATUS`, `IOT_NVD_SERVER_URL`, `IOT_NVD_SERVER_PORT`, `IOT_NVD_LOCATION_ID`, `IOT_NVD_ROOM_ID`, `IOT_NVD_LABEL`, `IOT_NVD_DEVICE_ID`, `IOT_NVD_PRIVATE_KEY`, `IOT_NVD_PUBLIC_KEY`, `IOT_NVD_CA_CERT`, `IOT_NVD_SUB_CERT`, `IOT_NVD_SERIAL_NUM`, `IOT_NVD_MAX` }

- `iot_error_t iot_nv_init` (unsigned char *device_info, unsigned int device_info_len)
Initialize a nv file-system.
- `iot_error_t iot_nv_deinit` ()
Deinitialize a nv file-system.
- `iot_error_t iot_nv_get_prov_data` (struct `iot_device_prov_data` *prov_data)
Get provisioning data from the nv file-system.
- `iot_error_t iot_nv_set_prov_data` (struct `iot_device_prov_data` *prov_data)
Set provisioning data to the nv file-system.
- `iot_error_t iot_nv_erase_prov_data` ()
Erase wifi/cloud provisioning data.
- `iot_error_t iot_nv_get_wifi_prov_data` (struct `iot_wifi_prov_data` *wifi_prov)
Get wifi provisioning data from the nv file-system.
- `iot_error_t iot_nv_set_wifi_prov_data` (struct `iot_wifi_prov_data` *wifi_prov)
Set wifi provisioning data to the nv file-system.
- `iot_error_t iot_nv_get_cloud_prov_data` (struct `iot_cloud_prov_data` *cloud_prov)
Get cloud provisioning data from the nv file-system.
- `iot_error_t iot_nv_set_cloud_prov_data` (struct `iot_cloud_prov_data` *cloud_prov)
Set cloud provisioning data to the nv file-system.
- `iot_error_t iot_nv_get_private_key` (char **key, unsigned int *len)
Get a private key from the nv file-system.
- `iot_error_t iot_nv_get_public_key` (char **key, unsigned int *len)
Get a public key from the nv file-system.
- `iot_error_t iot_nv_get_root_certificate` (char **cert, unsigned int *len)
Get a root cert from the nv file-system.
- `iot_error_t iot_nv_get_client_certificate` (char **cert, unsigned int *len)
Get a client cert from the nv file-system.
- `iot_error_t iot_nv_get_device_id` (char **device_id, unsigned int *len)
Get a device id from the nv file-system.
- `iot_error_t iot_nv_set_device_id` (const char *device_id)
Set a device id to the nv file-system.
- `iot_error_t iot_nv_get_serial_number` (char **sn, unsigned int *len)
Get a serial number from the nv file-system.
- `iot_error_t iot_nv_erase` (`iot_nvd_t` nv_type)
Erase a nv data.

5.25.1 Enumeration Type Documentation

5.25.1.1 `iot_nvd_t` enum `iot_nvd_t`

Enumerator

IOT_NVD_WIFI_PROV_STATUS	
IOT_NVD_AP_SSID	
IOT_NVD_AP_PASS	
IOT_NVD_AP_BSSID	
IOT_NVD_AP_AUTH_TYPE	
IOT_NVD_CLOUD_PROV_STATUS	
IOT_NVD_SERVER_URL	

Enumerator

IOT_NVD_SERVER_PORT	
IOT_NVD_LOCATION_ID	
IOT_NVD_ROOM_ID	
IOT_NVD_LABEL	
IOT_NVD_DEVICE_ID	
IOT_NVD_PRIVATE_KEY	
IOT_NVD_PUBLIC_KEY	
IOT_NVD_CA_CERT	
IOT_NVD_SUB_CERT	
IOT_NVD_SERIAL_NUM	
IOT_NVD_MAX	

5.25.2 Function Documentation

5.25.2.1 `iot_nv_deinit()` `iot_error_t iot_nv_deinit()`

Deinitialize a nv file-system.

Return values

<code>IOT_ERROR_NONE</code>	NV file-system deinit successful.
<code>IOT_ERROR_DEINIT_FAIL</code>	NV file-system deinit failed.

See also

[iot_bsp_fs_deinit](#)

Here is the call graph for this function:

5.25.2.2 `iot_nv_erase()` `iot_error_t iot_nv_erase(`
`iot_nvd_t nv_type)`

Erase a nv data.

This function erase the nv data completely in File-system.

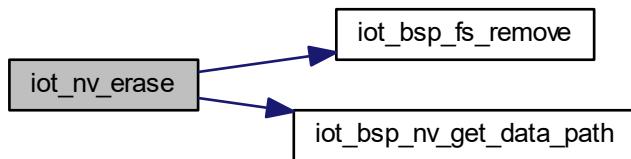
Parameters

in	<i>nv_type</i>	The type of nv data to erase.
----	----------------	-------------------------------

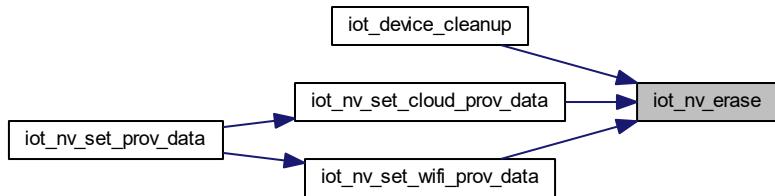
Return values

<i>IOT_ERROR_NONE</i>	NV data erase successful.
<i>IOT_ERROR_INVALID_ARGS</i>	Invalid nv type.
<i>IOT_ERROR_NV_DATA_ERROR</i>	NV data erase failed.
<i>IOT_ERROR_NV_DATA_NOT_EXIST</i>	NV data does not exist.

Here is the call graph for this function:



Here is the caller graph for this function:



5.25.2.3 iot_nv_erase_prov_data() *iot_error_t* iot_nv_erase_prov_data ()

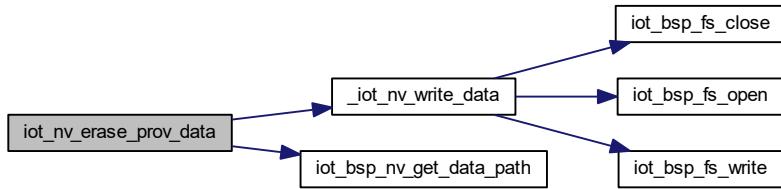
Erase wifi/cloud provisioning data.

This function erases the wifi/cloud provisioning data in File-system.

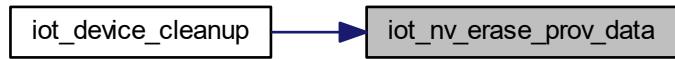
Return values

<i>IOT_ERROR_NONE</i>	NV data erase successful.
<i>IOT_ERROR_NV_DATA_ERROR</i>	NV data erase failed.

Here is the call graph for this function:



Here is the caller graph for this function:



5.25.2.4 `iot_nv_get_client_certificate()` `iot_error_t iot_nv_get_client_certificate (`
`char ** cert,`
`unsigned int * len)`

Get a client cert from the nv file-system.

Parameters

<code>out</code>	<code>cert</code>	A pointer to data array to store the client cert from the nv file-system.
<code>out</code>	<code>len</code>	The length of the nv data.

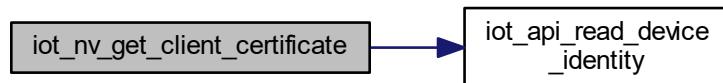
Return values

<code>IOT_ERROR_NONE</code>	Get nv data successful.
<code>IOT_ERROR_INVALID_ARGS</code>	Invalid argument.
<code>IOT_ERROR_NV_DATA_ERROR</code>	Get nv data failed.

Warning

The caller is always responsible to free the allocated pointer after using the data.

Here is the call graph for this function:



5.25.2.5 iot_nv_get_cloud_prov_data() `iot_error_t iot_nv_get_cloud_prov_data (struct iot_cloud_prov_data * cloud_prov)`

Get cloud provisioning data from the nv file-system.

Parameters

<code>out</code>	<code>cloud_prov</code>	A pointer to data structure to store the cloud provisioning data from the nv file-system.
------------------	-------------------------	---

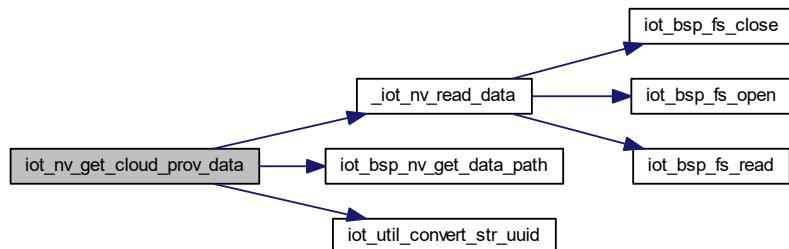
Return values

<code>IOT_ERROR_NONE</code>	Get nv data successful.
<code>IOT_ERROR_INVALID_ARGS</code>	Invalid argument.
<code>IOT_ERROR_NV_DATA_ERROR</code>	Get nv data failed.

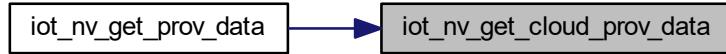
Warning

The caller is always responsible to free the allocated pointer after using the data.

Here is the call graph for this function:



Here is the caller graph for this function:



5.25.2.6 iot_nv_get_device_id() `iot_error_t iot_nv_get_device_id (`
 `char ** device_id,`
 `unsigned int * len)`

Get a device id from the nv file-system.

Parameters

<code>out</code>	<code>device_id</code>	A pointer to data array to store the device id from the nv file-system.
<code>out</code>	<code>len</code>	The length of the nv data.

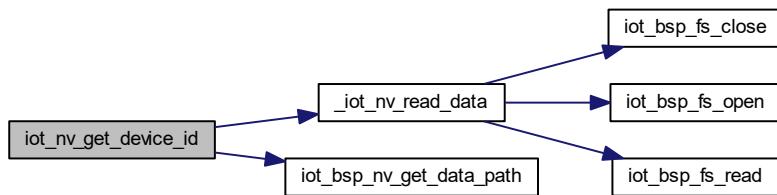
Return values

<code>IOT_ERROR_NONE</code>	Get nv data successful.
<code>IOT_ERROR_INVALID_ARGS</code>	Invalid argument.
<code>IOT_ERROR_NV_DATA_ERROR</code>	Get nv data failed.

Warning

The caller is always responsible to free the allocated pointer after using the data.

Here is the call graph for this function:



5.25.2.7 iot_nv_get_private_key() `iot_error_t iot_nv_get_private_key (`
 `char ** key,`
 `unsigned int * len)`

Get a private key from the nv file-system.

Parameters

out	<i>key</i>	A pointer to data array to store the private key from the nv file-system.
out	<i>len</i>	The length of the nv data.

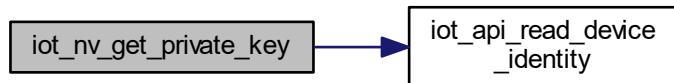
Return values

<i>IOT_ERROR_NONE</i>	Get nv data successful.
<i>IOT_ERROR_INVALID_ARGS</i>	Invalid argument.
<i>IOT_ERROR_NV_DATA_ERROR</i>	Get nv data failed.

Warning

The caller is always responsible to free the allocated pointer after using the data.

Here is the call graph for this function:



5.25.2.8 iot_nv_get_prov_data() `iot_error_t iot_nv_get_prov_data (`
 `struct iot_device_prov_data * prov_data)`

Get provisioning data from the nv file-system.

Parameters

out	<i>prov_data</i>	A pointer to data structure to store the provisioning data from the nv file-system.
-----	------------------	---

Return values

<i>IOT_ERROR_NONE</i>	Get nv data successful.
<i>IOT_ERROR_INVALID_ARGS</i>	Invalid argument.
<i>IOT_ERROR_NV_DATA_ERROR</i>	Get nv data failed.

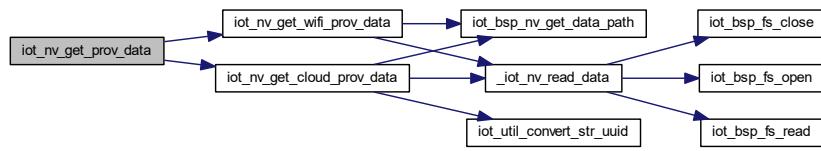
Warning

The caller is always responsible to free the allocated pointer after using the data.

See also

[iot_nv_get_wifi_prov_data](#)
[iot_nv_get_cloud_prov_data](#)

Here is the call graph for this function:



5.25.2.9 iot_nv_get_public_key() *iot_error_t* iot_nv_get_public_key (

```

    char ** key,
    unsigned int * len
  )

```

Get a public key from the nv file-system.

Parameters

out	<i>key</i>	A pointer to data array to store the public key from the nv file-system.
out	<i>len</i>	The length of the nv data.

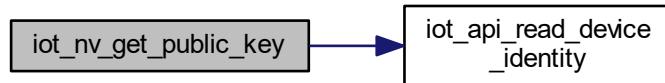
Return values

<i>IOT_ERROR_NONE</i>	Get nv data successful.
<i>IOT_ERROR_INVALID_ARGS</i>	Invalid argument.
<i>IOT_ERROR_NV_DATA_ERROR</i>	Get nv data failed.

Warning

The caller is always responsible to free the allocated pointer after using the data.

Here is the call graph for this function:



5.25.2.10 iot_nv_get_root_certificate() `iot_error_t iot_nv_get_root_certificate (`
 `char ** cert,`
 `unsigned int * len)`

Get a root cert from the nv file-system.

Parameters

<code>out</code>	<code>cert</code>	A pointer to data array to store the root cert from the nv file-system.
<code>out</code>	<code>len</code>	The length of the nv data.

Return values

<code>IOT_ERROR_NONE</code>	Get nv data successful.
<code>IOT_ERROR_INVALID_ARGS</code>	Invalid argument.
<code>IOT_ERROR_NV_DATA_ERROR</code>	Get nv data failed.

Warning

The caller is always responsible to free the allocated pointer after using the data.

5.25.2.11 iot_nv_get_serial_number() `iot_error_t iot_nv_get_serial_number (`
 `char ** sn,`
 `unsigned int * len)`

Get a serial number from the nv file-system.

Parameters

<code>out</code>	<code>sn</code>	A pointer to data array to store the serial number from the nv file-system.
<code>out</code>	<code>len</code>	The length of the nv data.

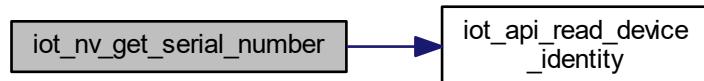
Return values

<code>IOT_ERROR_NONE</code>	Get nv data successful.
<code>IOT_ERROR_INVALID_ARGS</code>	Invalid argument.
<code>IOT_ERROR_NV_DATA_ERROR</code>	Get nv data failed.

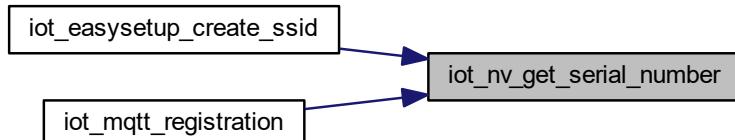
Warning

The caller is always responsible to free the allocated pointer after using the data.

Here is the call graph for this function:



Here is the caller graph for this function:



5.25.2.12 `iot_nv_get_wifi_prov_data()` `iot_error_t iot_nv_get_wifi_prov_data (struct iot_wifi_prov_data * wifi_prov)`

Get wifi provisioning data from the nv file-system.

Parameters

<code>out</code>	<code>wifi_prov</code>	A pointer to data structure to store the wifi provisioning data from the nv file-system.
------------------	------------------------	--

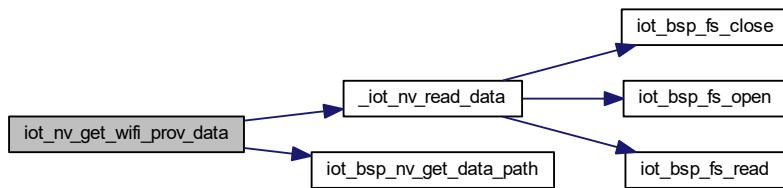
Return values

<code>IOT_ERROR_NONE</code>	Get nv data successful.
<code>IOT_ERROR_INVALID_ARGS</code>	Invalid argument.
<code>IOT_ERROR_NV_DATA_ERROR</code>	Get nv data failed.

Warning

The caller is always responsible to free the allocated pointer after using the data.

Here is the call graph for this function:



Here is the caller graph for this function:



5.25.2.13 `iot_nv_init()` `iot_error_t iot_nv_init (`
`unsigned char * device_info,`
`unsigned int device_info_len)`

Initialize a nv file-system.

This function initializes the nv file-system of the device. You must call this function before using nv management function.

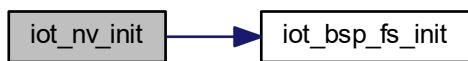
Return values

<i>IOT_ERROR_NONE</i>	NV file-system init successful.
<i>IOT_ERROR_INIT_FAILED</i>	NV file-system init failed.

See also

[iot_bsp_fs_init](#)

Here is the call graph for this function:



Here is the caller graph for this function:



5.25.2.14 *iot_nv_set_cloud_prov_data()* *iot_error_t* *iot_nv_set_cloud_prov_data* (
*struct iot_cloud_prov_data * cloud_prov*)

Set cloud provisioning data to the nv file-system.

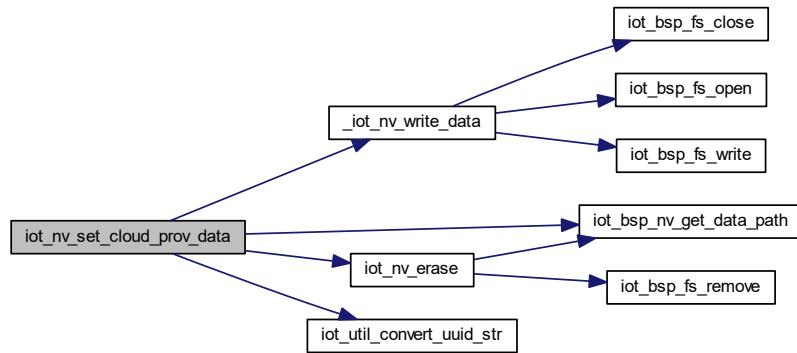
Parameters

<i>in</i>	<i>cloud_prov</i>	A pointer to cloud provisioning data structure.
-----------	-------------------	---

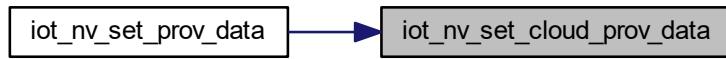
Return values

<i>IOT_ERROR_NONE</i>	Set nv data successful.
<i>IOT_ERROR_INVALID_ARGS</i>	Invalid argument.
<i>IOT_ERROR_NV_DATA_ERROR</i>	Set nv data failed.

Here is the call graph for this function:



Here is the caller graph for this function:



5.25.2.15 `iot_nv_set_device_id()` `iot_error_t iot_nv_set_device_id (const char * device_id)`

Set a device id to the nv file-system.

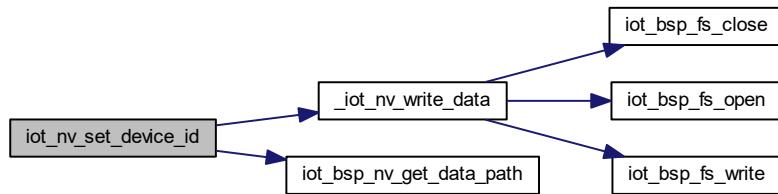
Parameters

in	<code>device_id</code>	The string of device id from MQTT Server.
----	------------------------	---

Return values

<code>IOT_ERROR_NONE</code>	Set nv data successful.
<code>IOT_ERROR_INVALID_ARGS</code>	Invalid argument.
<code>IOT_ERROR_NV_DATA_ERROR</code>	Set nv data failed.

Here is the call graph for this function:



5.25.2.16 iot_nv_set_prov_data() `iot_error_t iot_nv_set_prov_data (struct iot_device_prov_data * prov_data)`

Set provisioning data to the nv file-system.

Parameters

in	<code>prov_data</code>	A pointer to provisioning data structure.
----	------------------------	---

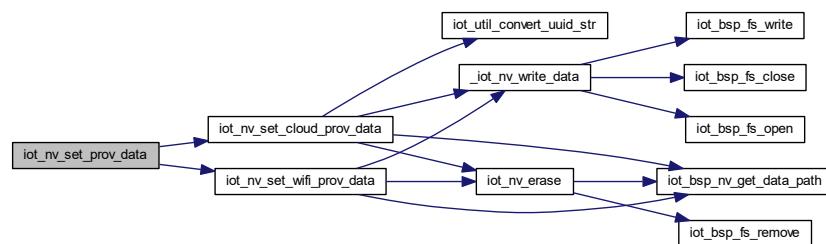
Return values

<code>IOT_ERROR_NONE</code>	Set nv data successful.
<code>IOT_ERROR_INVALID_ARGS</code>	Invalid argument.
<code>IOT_ERROR_NV_DATA_ERROR</code>	Set nv data failed.

See also

[iot_nv_set_wifi_prov_data](#)
[iot_nv_set_cloud_prov_data](#)

Here is the call graph for this function:



5.25.2.17 iot_nv_set_wifi_prov_data() `iot_error_t iot_nv_set_wifi_prov_data (struct iot_wifi_prov_data * wifi_prov)`

Set wifi provisioning data to the nv file-system.

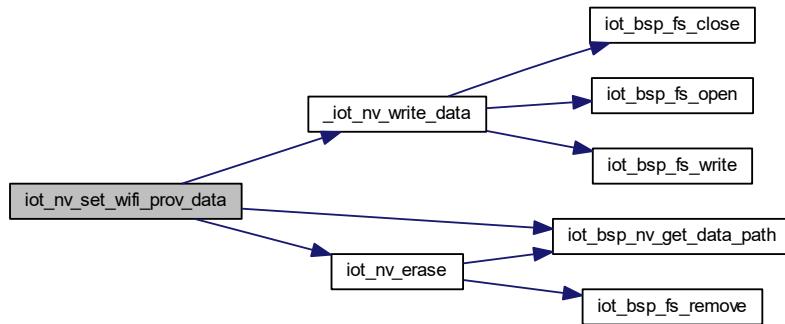
Parameters

in	<code>wifi_prov</code>	A pointer to wifi provisioning data structure.
----	------------------------	--

Return values

<code>IOT_ERROR_NONE</code>	Set nv data successful.
<code>IOT_ERROR_INVALID_ARGS</code>	Invalid argument.
<code>IOT_ERROR_NV_DATA_ERROR</code>	Set nv data failed.

Here is the call graph for this function:



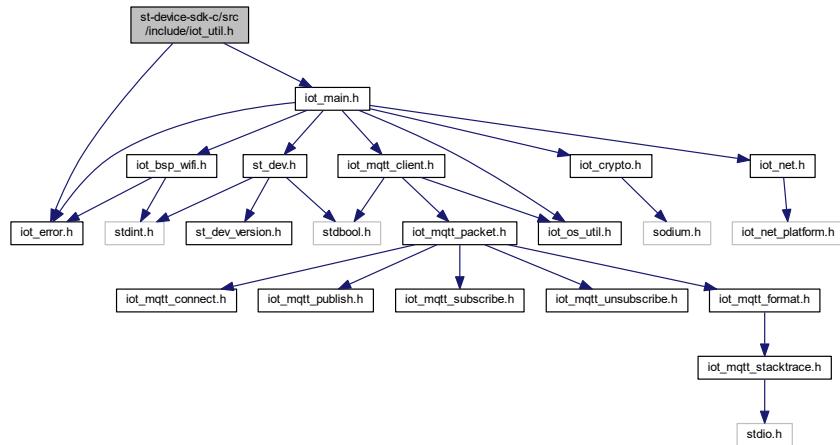
Here is the caller graph for this function:



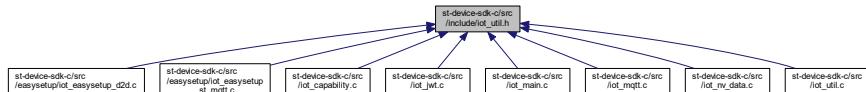
5.26 st-device-sdk-c/src/include/iot_util.h File Reference

```
#include "iot_error.h"
#include "iot_main.h"
```

Include dependency graph for iot_util.h:



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



Data Structures

- struct [url_parse_t](#)
Contains a "url parse" data.

Functions

- [iot_error_t iot_util_url_parse \(char *url, url_parse_t *output\)](#)
parse url with protocol, domain, port number parts for st-iot-core
- [iot_error_t iot_util_convert_str_uuid \(const char *str, struct iot_uuid *uuid\)](#)
uuid type string to iot_uuid struct converting function for st-iot-core
- [iot_error_t iot_util_convert_uuid_str \(struct iot_uuid *uuid, char *str, int max_sz\)](#)
iot_uuid struct based value to uuid type string converting function for st-iot-core
- [iot_error_t iot_util_get_random_uuid \(struct iot_uuid *uuid\)](#)
To get random uuid based on iot_uuid struct.
- [iot_error_t iot_util_convert_str_mac \(char *str, struct iot_mac *mac\)](#)
To convert WIFI mac string into iot_mac struct value.
- [iot_error_t iot_util_convert_mac_str \(struct iot_mac *mac, char *str, int max_sz\)](#)
To convert iot_mac value into WIFI mac string.
- [uint16_t iot_util_convert_channel_freq \(uint8_t channel\)](#)
To convert Wi-Fi channel into frequency value.

5.26.1 Function Documentation

5.26.1.1 iot_util_convert_channel_freq() `uint16_t iot_util_convert_channel_freq (uint8_t channel)`

To convert Wi-Fi channel into frequency value.

This function tries to convert from the channel to frequency

Parameters

in	Wi-Fi	channel
----	-------	---------

Returns

Wi-Fi frequency

5.26.1.2 iot_util_convert_mac_str() `iot_error_t iot_util_convert_mac_str (`

```
    struct iot_mac * mac,
    char * str,
    int max_sz )
```

To convert `iot_mac` value into WiFi mac string.

This function tries to convert from the `iot_mac` struct value to string

Parameters

in	<i>mac</i>	converting wanted <code>iot_mac</code> struct value pointer
in	<i>str</i>	allocated memory pointer for converted WiFi mac string
in	<i>max_sz</i>	max size of allocated memory pointer

Returns

`iot_error_t`

Return values

<code>IOT_ERROR_NONE</code>	success
<code>IOT_ERROR_INVALID_ARGS</code>	invalid arguments

5.26.1.3 iot_util_convert_str_mac() `iot_error_t iot_util_convert_str_mac (`
 `char * str,`
 `struct iot_mac * mac)`

To convert WIFI mac string into `iot_mac` struct value.

This function tries to convert from the string to `iot_mac` struct value

Parameters

in	<code>str</code>	WIFI mac string pointer such as 'xx:xx:xx:xx:xx:xx'
in	<code>mac</code>	allocated <code>iot_mac</code> struct pointer to get <code>iot_mac</code> value from str

Returns

`iot_error_t`

Return values

<code>IOT_ERROR_NONE</code>	success
<code>IOT_ERROR_INVALID_ARGS</code>	invalid arguments

5.26.1.4 iot_util_convert_str_uuid() `iot_error_t iot_util_convert_str_uuid (`
 `const char * str,`
 `struct iot_uuid * uuid)`

uuid type string to `iot_uuid` struct converting function for st-iot-core

This function tries to convert from uuid type string to `iot_uuid` struct

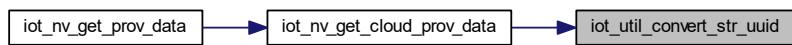
Parameters

in	<code>str</code>	uuid type string pointer such as 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
in	<code>uuid</code>	allocated <code>iot_uuid</code> struct pointer to get <code>iot_uuid</code> value from str

Returns

return `IOT_ERROR_NONE` on success, or `iot_error_t` errors if it fails

Here is the caller graph for this function:



```
5.26.1.5 iot_util_convert_uuid_str() iot_error_t iot_util_convert_uuid_str (
    struct iot_uuid * uuid,
    char * str,
    int max_sz )
```

iot_uuid struct based value to uuid type string converting function for st-iot-core

This function tries to convert from *iot_uuid* struct based value to uuid type string

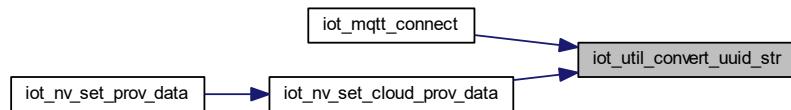
Parameters

in	<i>uuid</i>	converting wanted <i>iot_uuid</i> struct value pointer
in	<i>str</i>	allocated memory pointer for converted uuid type string
in	<i>max_sz</i>	max size of allocated memory pointer

Returns

return IOT_ERROR_NONE on success, or *iot_error_t* errors if it fails

Here is the caller graph for this function:



```
5.26.1.6 iot_util_get_random_uuid() iot_error_t iot_util_get_random_uuid (
    struct iot_uuid * uuid )
```

To get random uuid based on *iot_uuid* struct.

This function tries to make random *iot_uuid* values

Parameters

in	<i>uuid</i>	allocated <i>iot_uuid</i> struct pointer to get random <i>iot_uuid</i> value
----	-------------	--

Returns

```
return IOT_ERROR_NONE on success, or iot_error_t errors if it fails
```

Here is the call graph for this function:



5.26.1.7 iot_util_url_parse() `iot_error_t iot_util_url_parse (char * url, url_parse_t * output)`

parse url with protocol, domain, port number parts for st-iot-core

This function parse give url protocol, domain, port number parts

Parameters

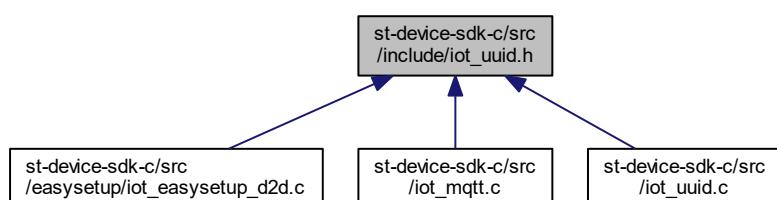
in	<code>url</code>	null-terminated url string like "https://example.sample.com:1234"
out	<code>output</code>	parsed output with <code>url_parse_t</code> type

Returns

```
return IOT_ERROR_NONE on success, or iot_error_t errors if it fails
```

5.27 st-device-sdk-c/src/include/iot_uuid.h File Reference

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



Functions

- `iot_error_t iot_random_uuid_from_mac (struct iot_uuid *uuid)`
Change mac to random uuid This function changes mac to random uuid.
- `iot_error_t iot_uuid_from_mac (struct iot_uuid *uuid)`
Change mac to uuid This function changes mac to the uuid.

5.27.1 Function Documentation

5.27.1.1 `iot_random_uuid_from_mac()` `iot_error_t iot_random_uuid_from_mac (struct iot_uuid * uuid)`

Change mac to random uuid This function changes mac to random uuid.

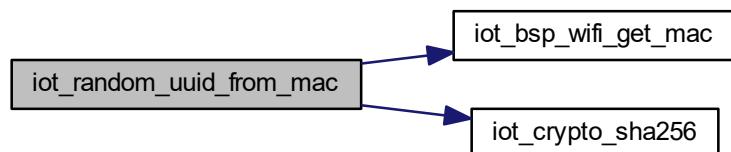
Parameters

<code>out</code>	<code>uuid</code>	created random uuid
------------------	-------------------	---------------------

Returns

`IOT_ERROR_NONE` on success

Here is the call graph for this function:



Here is the caller graph for this function:



```
5.27.1.2 iot_uuid_from_mac() iot_error_t iot_uuid_from_mac (
    struct iot_uuid * uuid )
```

Change mac to uuid This function changes mac to the uuid.

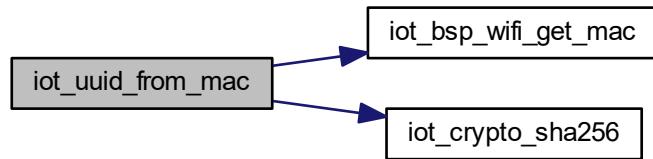
Parameters

out	uuid	created uuid
-----	------	--------------

Returns

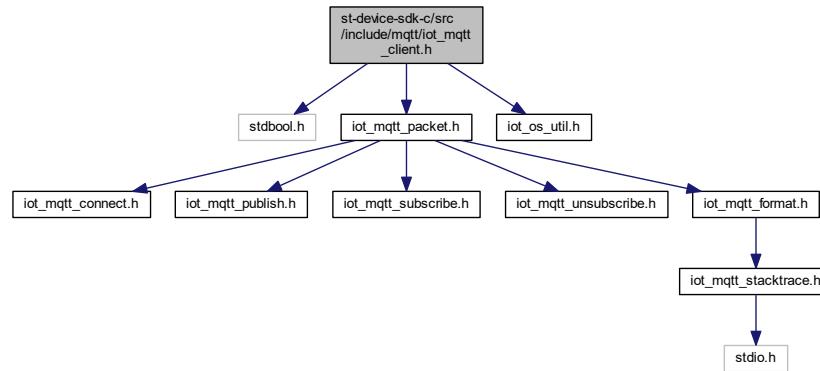
IOT_ERROR_NONE on success

Here is the call graph for this function:

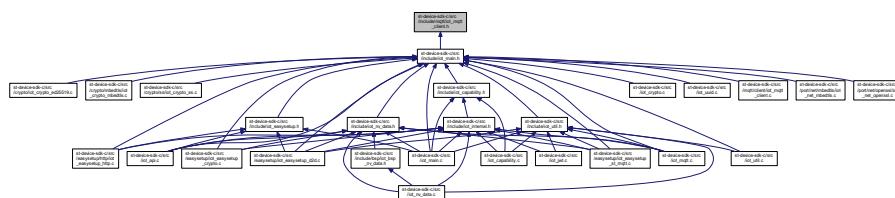


5.28 st-device-sdk-c/src/include/mqtt/iot_mqtt_client.h File Reference

```
#include <stdbool.h>
#include "iot_mqtt_packet.h"
#include "iot_os_util.h"
Include dependency graph for iot_mqtt_client.h:
```



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



Data Structures

- struct [MQTTMessage](#)
- struct [MessageData](#)
- struct [MQTTConnackData](#)
- struct [MQTTSubackData](#)
- struct [MQTTClient](#)
- struct [MQTTClient::MessageHandlers](#)

Macros

- #define [DLLImport](#)
- #define [DLLExport](#)
- #define [MAX_PACKET_ID](#) 65535 /* according to the MQTT specification - do not change! */
- #define [IOT_MQTT_STACK_SIZE](#) 2048
- #define [IOT_MQTT_PRIORITY](#) 4
- #define [CONFIG_STDK_MQTT_SEND_BUFFER](#) 2048
- #define [CONFIG_STDK_MQTT_RECV_BUFFER](#) 2048
- #define [CONFIG_STDK_MQTT_SEND_CYCLE](#) 30000
- #define [CONFIG_STDK_MQTT_PUBLISH_RETRY](#) 3
- #define [CONFIG_STDK_MQTT_PING_RETRY](#) 3
- #define [CONFIG_STDK_MQTT_RECV_CYCLE](#) 100
- #define [CONFIG_STDK_MQTT_DYNAMIC_BUFFER](#) 1
- #define [MAX_MESSAGE_HANDLERS](#) 5 /* redefinable - how many subscriptions do you want? */
- #define [DefaultClient](#) {0, 0, 0, 0, NULL, NULL, 0, 0, 0}

Typedefs

- typedef struct [MQTTMessage](#) [MQTTMessage](#)
- typedef struct [MQTTClient](#) [MQTTClient](#)
- typedef struct [MessageData](#) [MessageData](#)
- typedef struct [MQTTConnackData](#) [MQTTConnackData](#)
- typedef struct [MQTTSubackData](#) [MQTTSubackData](#)
- typedef void(*) [messageHandler](#) ([MessageData](#) *, void *)

Enumerations

- enum [QoS](#) { [QOS0](#), [QOS1](#), [QOS2](#), [SUBFAIL](#) = 0x80 }
- enum [returnCode](#) { [MQTT_DISCONNECTED](#) = -3, [MQTT_BUFFER_OVERFLOW](#) = -2, [MQTT_FAILURE](#) = -1, [MQTT_SUCCESS](#) = 0 }

Functions

- `DLLEXport bool MQTTClientInit (MQTTClient *client, iot_net_interface_t *network, unsigned int command←_timeout_ms, unsigned char *sendbuf, size_t sendbuf_size, unsigned char *readbuf, size_t readbuf_size)`
- `DLLEXport int MQTTConnectWithResults (MQTTClient *client, MQTTPacket_connectData *options, MQTTConnackData *data)`
- `DLLEXport int MQTTConnect (MQTTClient *client, MQTTPacket_connectData *options)`
- `DLLEXport int MQTTPublish (MQTTClient *client, const char *, MQTTMessage *)`
- `DLLEXport int MQTTSetMessageHandler (MQTTClient *c, const char *topicFilter, messageHandler messageHandler, void *userData)`
- `DLLEXport int MQTTSubscribe (MQTTClient *client, const char *topicFilter, enum QoS, messageHandler, void *userData)`
- `DLLEXport int MQTTSubscribeWithResults (MQTTClient *client, const char *topicFilter, enum QoS, messageHandler, MQTTSubackData *data, void *userData)`
- `DLLEXport int MQTTUnsubscribe (MQTTClient *client, const char *topicFilter)`
- `DLLEXport int MQTTDisconnect (MQTTClient *client)`
- `DLLEXport int MQTTYield (MQTTClient *client, int time)`

5.28.1 Macro Definition Documentation

5.28.1.1 `CONFIG_STDK_MQTT_DYNAMIC_BUFFER` `#define CONFIG_STDK_MQTT_DYNAMIC_BUFFER 1`

5.28.1.2 `CONFIG_STDK_MQTT_PING_RETRY` `#define CONFIG_STDK_MQTT_PING_RETRY 3`

5.28.1.3 `CONFIG_STDK_MQTT_PUBLISH_RETRY` `#define CONFIG_STDK_MQTT_PUBLISH_RETRY 3`

5.28.1.4 `CONFIG_STDK_MQTT_RECV_BUFFER` `#define CONFIG_STDK_MQTT_RECV_BUFFER 2048`

5.28.1.5 `CONFIG_STDK_MQTT_RECV_CYCLE` `#define CONFIG_STDK_MQTT_RECV_CYCLE 100`

5.28.1.6 `CONFIG_STDK_MQTT_SEND_BUFFER` `#define CONFIG_STDK_MQTT_SEND_BUFFER 2048`

5.28.1.7 CONFIG_STDK_MQTT_SEND_CYCLE #define CONFIG_STDK_MQTT_SEND_CYCLE 30000

5.28.1.8 DefaultClient #define DefaultClient {0, 0, 0, 0, NULL, NULL, 0, 0, 0}

5.28.1.9 DLLExport #define DLLExport

5.28.1.10 DLLImport #define DLLImport

5.28.1.11 IOT_MQTT_PRIORITY #define IOT_MQTT_PRIORITY 4

5.28.1.12 IOT_MQTT_STACK_SIZE #define IOT_MQTT_STACK_SIZE 2048

5.28.1.13 MAX_MESSAGE_HANDLERS #define MAX_MESSAGE_HANDLERS 5 /* redefinable - how many subscriptions do you want? */

5.28.1.14 MAX_PACKET_ID #define MAX_PACKET_ID 65535 /* according to the MQTT specification - do not change! */

5.28.2 Typedef Documentation

5.28.2.1 MessageData typedef struct [MessageData](#) MessageData

5.28.2.2 messageHandler typedef void(* messageHandler) ([MessageData](#) *, void *)

5.28.2.3 MQTTClient `typedef struct MQTTClient MQTTClient`

5.28.2.4 MQTTConnackData `typedef struct MQTTConnackData MQTTConnackData`

5.28.2.5 MQTTMessage `typedef struct MQTTMessage MQTTMessage`

5.28.2.6 MQTTSubackData `typedef struct MQTTSubackData MQTTSubackData`

5.28.3 Enumeration Type Documentation

5.28.3.1 QoS `enum QoS`

Enumerator

QOS0
QOS1
QOS2
SUBFAIL

5.28.3.2 returnCode `enum resultCode`

Enumerator

MQTT_DISCONNECTED
MQTT_BUFFER_OVERFLOW
MQTT_FAILURE
MQTT_SUCCESS

5.28.4 Function Documentation

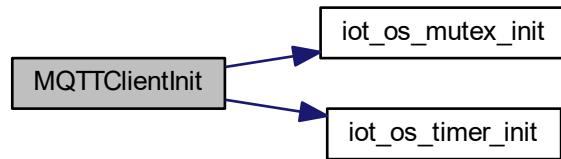
```
5.28.4.1 MQTTClientInit() DLLExport bool MQTTClientInit (
    MQTTClient * client,
    iot_net_interface_t * network,
    unsigned int command_timeout_ms,
    unsigned char * sendbuf,
    size_t sendbuf_size,
    unsigned char * readbuf,
    size_t readbuf_size )
```

Create an MQTT client object

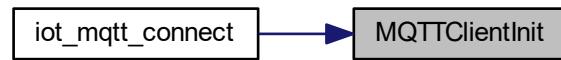
Parameters

<i>client</i>	
<i>network</i>	
<i>command_timeout_ms</i>	

Here is the call graph for this function:



Here is the caller graph for this function:



```
5.28.4.2 MQTTConnect() DLLExport int MQTTConnect (
    MQTTClient * client,
    MQTTPacket_connectData * options )
```

MQTT Connect - send an MQTT connect packet down the network and wait for a Connack. The nework object must be connected to the network endpoint before calling this

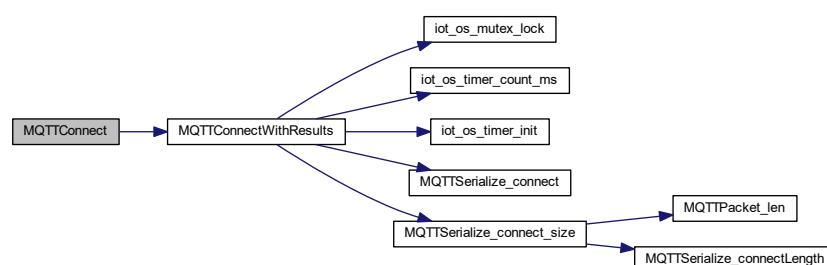
Parameters

<i>options</i>	- connect options
----------------	-------------------

Returns

success code

Here is the call graph for this function:



Here is the caller graph for this function:



```

5.28.4.3 MQTTConnectWithResults() DLLExport int MQTTConnectWithResults (
    MQTTClient * client,
    MQTTPacket_connectData * options,
    MQTTConnackData * data )
  
```

MQTT Connect - send an MQTT connect packet down the network and wait for a Connack. The nework object must be connected to the network endpoint before calling this

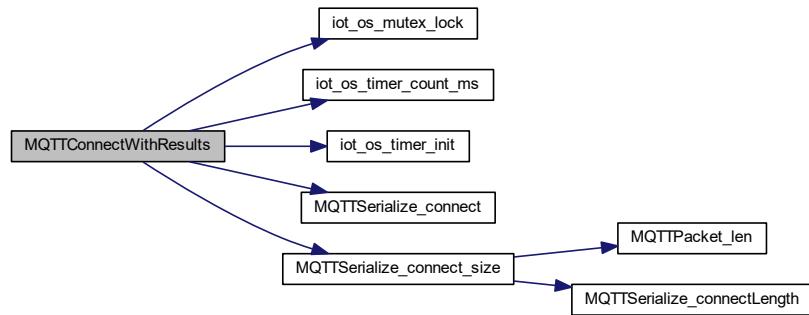
Parameters

<i>options</i>	- connect options
----------------	-------------------

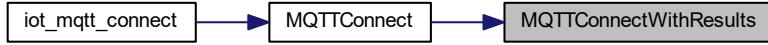
Returns

success code

Here is the call graph for this function:



Here is the caller graph for this function:



5.28.4.4 `MQTTDisconnect()` `DLLExport int MQTTDisconnect (`
`MQTTClient * client)`

MQTT Disconnect - send an MQTT disconnect packet and close the connection

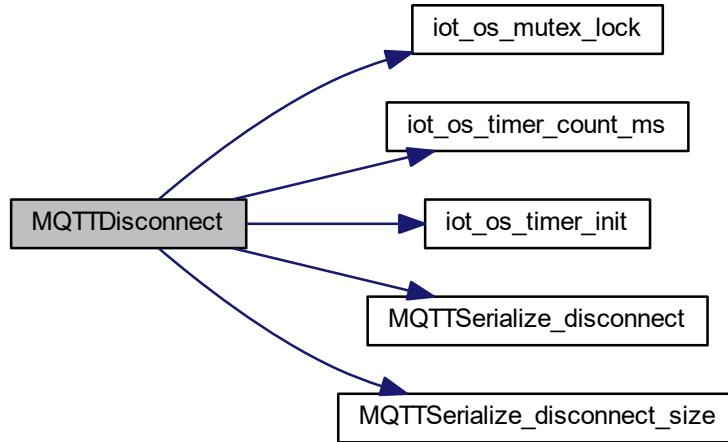
Parameters

<code>client</code>	- the client object to use
---------------------	----------------------------

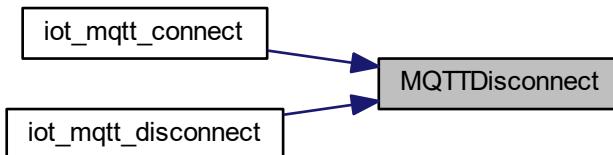
Returns

success code

Here is the call graph for this function:



Here is the caller graph for this function:



5.28.4.5 MQTTPublish() `DLLEXPORT int MQTTPublish (`
`MQTTClient * client,`
`const char * ,`
`MQTTMessage *)`

MQTT Publish - send an MQTT publish packet and wait for all acks to complete for all QoSs

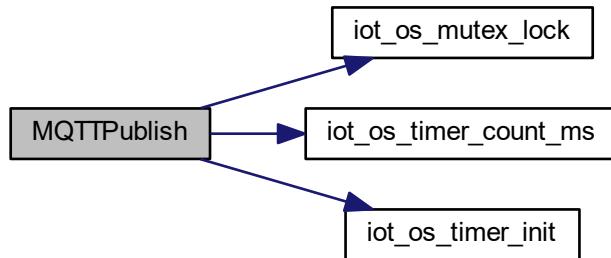
Parameters

<code>client</code>	- the client object to use
<code>topic</code>	- the topic to publish to
<code>message</code>	- the message to send

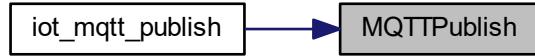
Returns

success code

Here is the call graph for this function:



Here is the caller graph for this function:

**5.28.4.6 MQTTSetMessageHandler()** `DLLExport int MQTTSetMessageHandler (`

```

    MQTTClient * c,
    const char * topicFilter,
    messageHandler messageHandler,
    void * userData )
  
```

MQTT SetMessageHandler - set or remove a per topic message handler

Parameters

<code>client</code>	- the client object to use
<code>topicFilter</code>	- the topic filter set the message handler for
<code>messageHandler</code>	- pointer to the message handler function or NULL to remove

Returns

success code

```
5.28.4.7 MQTTSubscribe() DLLExport int MQTTSubscribe (
    MQTTClient * client,
    const char * topicFilter,
    enum QoS,
    messageHandler ,
    void * userData )
```

MQTT Subscribe - send an MQTT subscribe packet and wait for suback before returning.

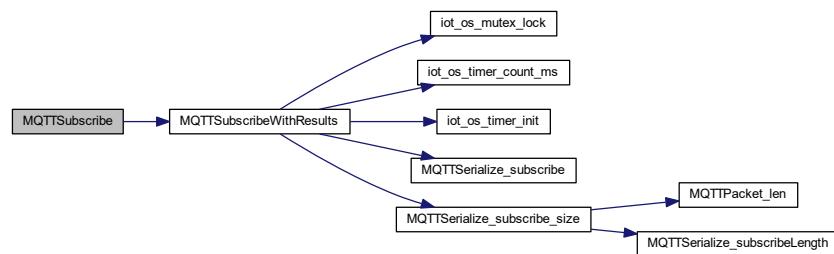
Parameters

<i>client</i>	- the client object to use
<i>topicFilter</i>	- the topic filter to subscribe to
<i>message</i>	- the message to send

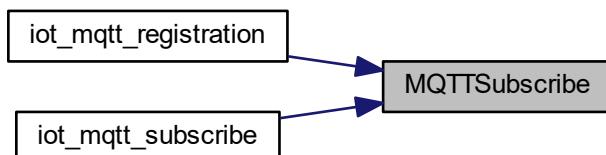
Returns

success code

Here is the call graph for this function:



Here is the caller graph for this function:



```
5.28.4.8 MQTTSubscribeWithResults() DLLExport int MQTTSubscribeWithResults (
    MQTTClient * client,
    const char * topicFilter,
    enum QoS,
    messageHandler ,
    MQTTSubackData * data,
    void * userData )
```

MQTT Subscribe - send an MQTT subscribe packet and wait for suback before returning.

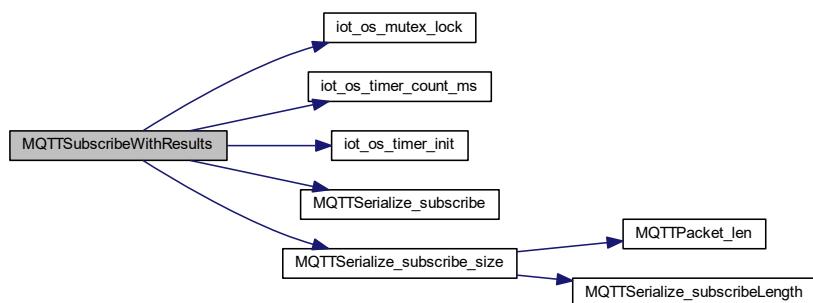
Parameters

<i>client</i>	- the client object to use
<i>topicFilter</i>	- the topic filter to subscribe to
<i>message</i>	- the message to send
<i>data</i>	- suback granted QoS returned

Returns

success code

Here is the call graph for this function:



Here is the caller graph for this function:



5.28.4.9 MQTTUnsubscribe() `DLLExport int MQTTUnsubscribe (`
`MQTTClient * client,`
`const char * topicFilter)`

MQTT Subscribe - send an MQTT unsubscribe packet and wait for unsuback before returning.

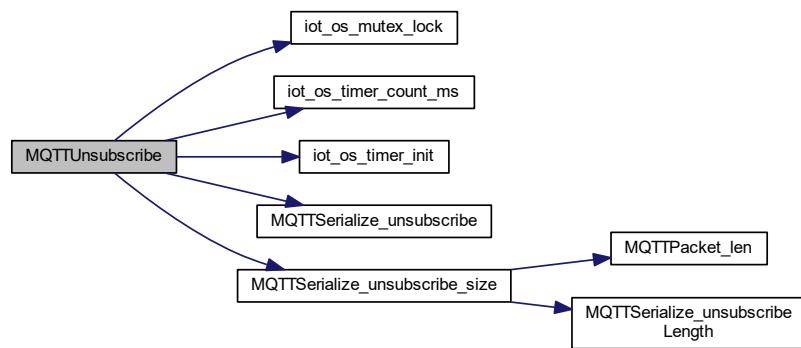
Parameters

<i>client</i>	- the client object to use
<i>topicFilter</i>	- the topic filter to unsubscribe from

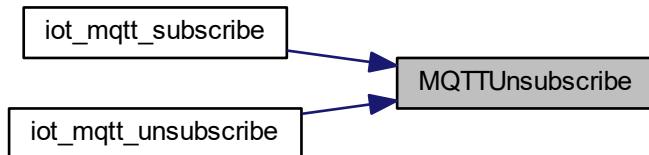
Returns

success code

Here is the call graph for this function:



Here is the caller graph for this function:



```

5.28.4.10 MQTTYield() DLLExport int MQTTYield (
    MQTTClient * client,
    int time )
  
```

MQTT Yield - MQTT background

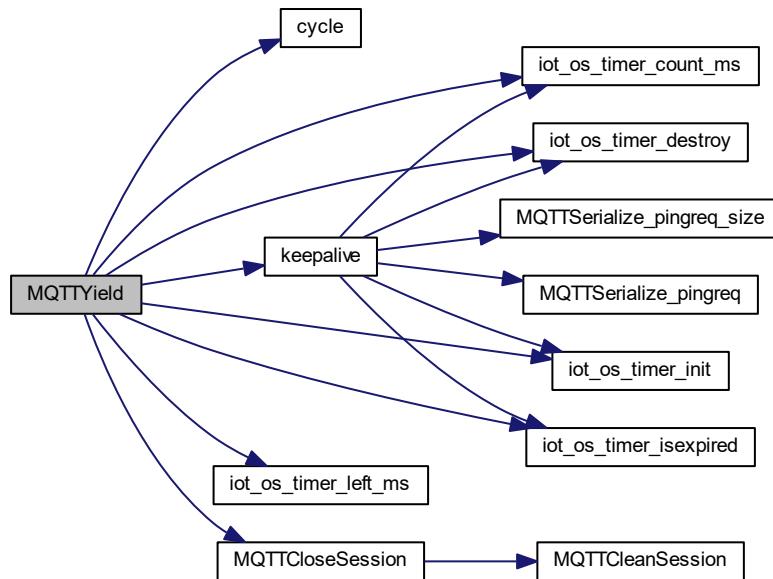
Parameters

<code>client</code>	- the client object to use
<code>time</code>	- the time, in milliseconds, to yield for

Returns

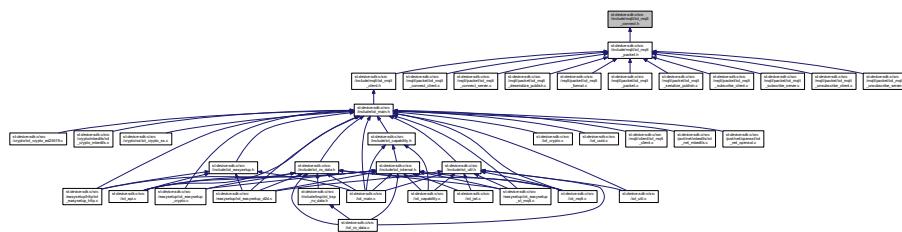
success code

Here is the call graph for this function:



5.29 st-device-sdk-c/src/include/mqtt/iot_mqtt_connect.h File Reference

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



Data Structures

- union [MQTTConnectFlags](#)
- struct [MQTTPacket_willOptions](#)
- struct [MQTTPacket_connectData](#)
- union [MQTTConnackFlags](#)

Macros

- `#define DLLImport`
- `#define DLLExport`
- `#define MQTTPacket_willOptions_initializer { {'M', 'Q', 'T', 'W'}, 0, {NULL, {0, NULL}}, {NULL, {0, NULL}}, 0, 0 }`
- `#define MQTTPacket_connectData_initializer`

Enumerations

- enum `connack_return_codes` {

 `MQTT_CONNECTION_ACCEPTED` = 0, `MQTT_UNNACCEPTABLE_PROTOCOL` = 1, `MQTT_CLIENTID_REJECTED` = 2, `MQTT_SERVER_UNAVAILABLE` = 3,

 `MQTT_BAD_USERNAME_OR_PASSWORD` = 4, `MQTT_NOT_AUTHORIZED` = 5 }

Functions

- `DLLExport int MQTTSerialize_connect (unsigned char *buf, int buflen, MQTTPacket_connectData *options)`
- `DLLExport int MQTTDeserialize_connect (MQTTPacket_connectData *data, unsigned char *buf, int len)`
- `DLLExport int MQTTSerialize_connect_size (MQTTPacket_connectData *options)`
- `DLLExport int MQTTSerialize_connack (unsigned char *buf, int buflen, unsigned char connack_rc, unsigned char sessionPresent)`
- `DLLExport int MQTTDeserialize_connack (unsigned char *sessionPresent, unsigned char *connack_rc, unsigned char *buf, int buflen)`
- `DLLExport int MQTTSerialize_disconnect (unsigned char *buf, int buflen)`
- `DLLExport int MQTTSerialize_pingreq (unsigned char *buf, int buflen)`
- `DLLExport int MQTTSerialize_disconnect_size ()`
- `DLLExport int MQTTSerialize_pingreq_size ()`

5.29.1 Macro Definition Documentation

5.29.1.1 **DLLExport** `#define DLLExport`

5.29.1.2 **DLLImport** `#define DLLImport`

5.29.1.3 **MQTTPacket_connectData_initializer** `#define MQTTPacket_connectData_initializer`

Value:

```
{ {'M', 'Q', 'T', 'C'}, 0, 4, {NULL, {0, NULL}}, 60, 1, 0, \
MQTTPacket_willOptions_initializer, {NULL, {0, NULL}}, {NULL, {0, NULL}} }
```

5.29.1.4 **MQTTPacket_willOptions_initializer** `#define MQTTPacket_willOptions_initializer { {'M', 'Q', 'T', 'W'}, 0, {NULL, {0, NULL}}, {NULL, {0, NULL}}, 0, 0 }`

5.29.2 Enumeration Type Documentation

5.29.2.1 **connack_return_codes** `enum connack_return_codes`

Enumerator

MQTT_CONNECTION_ACCEPTED
MQTT_UNACCEPTABLE_PROTOCOL
MQTT_CLIENTID_REJECTED
MQTT_SERVER_UNAVAILABLE
MQTT_BAD_USERNAME_OR_PASSWORD
MQTT_NOT_AUTHORIZED

5.29.3 Function Documentation

5.29.3.1 MQTTDeserialize_connack() [DLLExport](#) int MQTTDeserialize_connack (

```
    unsigned char * sessionPresent,
    unsigned char * connack_rc,
    unsigned char * buf,
    int buflen )
```

Deserializes the supplied (wire) buffer into connack data - return code

Parameters

<i>sessionPresent</i>	the session present flag returned (only for MQTT 3.1.1)
<i>connack_rc</i>	returned integer value of the connack return code
<i>buf</i>	the raw buffer data, of the correct length determined by the remaining length field
<i>len</i>	the length in bytes of the data in the supplied buffer

Returns

error code. 1 is success, 0 is failure

5.29.3.2 MQTTDeserialize_connect() [DLLExport](#) int MQTTDeserialize_connect (

```
    MQTTPacket_connectData * data,
    unsigned char * buf,
    int len )
```

Deserializes the supplied (wire) buffer into connect data structure

Parameters

<i>data</i>	the connect data structure to be filled out
<i>buf</i>	the raw buffer data, of the correct length determined by the remaining length field
<i>len</i>	the length in bytes of the data in the supplied buffer

Returns

error code. 1 is success, 0 is failure

5.29.3.3 MQTTSerialize_connack() `DLLEXPORT int MQTTSerialize_connack (`

```
    unsigned char * buf,
    int buflen,
    unsigned char connack_rc,
    unsigned char sessionPresent )
```

Serializes the connack packet into the supplied buffer.

Parameters

<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer
<i>connack_rc</i>	the integer connack return code to be used
<i>sessionPresent</i>	the MQTT 3.1.1 sessionPresent flag

Returns

serialized length, or error if 0

5.29.3.4 MQTTSerialize_connect() `DLLEXPORT int MQTTSerialize_connect (`

```
    unsigned char * buf,
    int buflen,
    MQTTPacket_connectData * options )
```

Serializes the connect options into the buffer.

Parameters

<i>buf</i>	the buffer into which the packet will be serialized
<i>len</i>	the length in bytes of the supplied buffer
<i>options</i>	the options to be used to build the connect packet

Returns

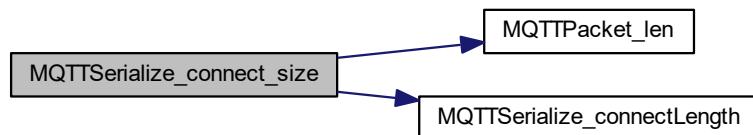
serialized length, or error if 0

Here is the caller graph for this function:



5.29.3.5 MQTTSerialize_connect_size() `DLLEXPORT int MQTTSerialize_connect_size (MQTTPacket_connectData * options)`

Here is the call graph for this function:



Here is the caller graph for this function:



5.29.3.6 MQTTSerialize_disconnect() `DLLEXPORT int MQTTSerialize_disconnect (unsigned char * buf, int buflen)`

Serializes a disconnect packet into the supplied buffer, ready for writing to a socket

Parameters

<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer, to avoid overruns

Returns

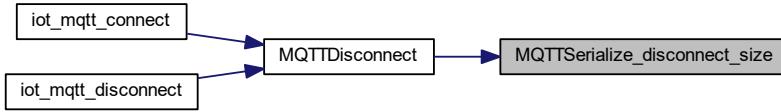
serialized length, or error if 0

Here is the caller graph for this function:



5.29.3.7 MQTTSerialize_disconnect_size() DLLExport int MQTTSerialize_disconnect_size ()

Here is the caller graph for this function:



5.29.3.8 MQTTSerialize_pingreq() DLLExport int MQTTSerialize_pingreq (

```

unsigned char * buf,
int buflen )
  
```

Serializes a disconnect packet into the supplied buffer, ready for writing to a socket

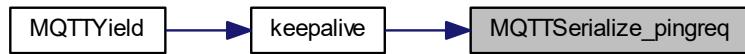
Parameters

<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer, to avoid overruns

Returns

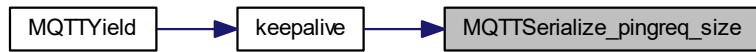
serialized length, or error if 0

Here is the caller graph for this function:



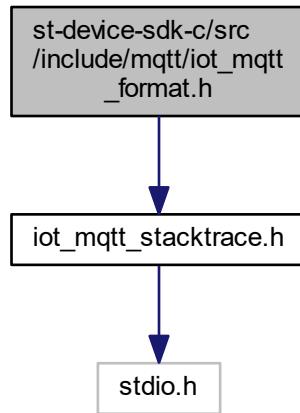
5.29.3.9 MQTTSerialize_pingreq_size() DLLExport int MQTTSerialize_pingreq_size ()

Here is the caller graph for this function:

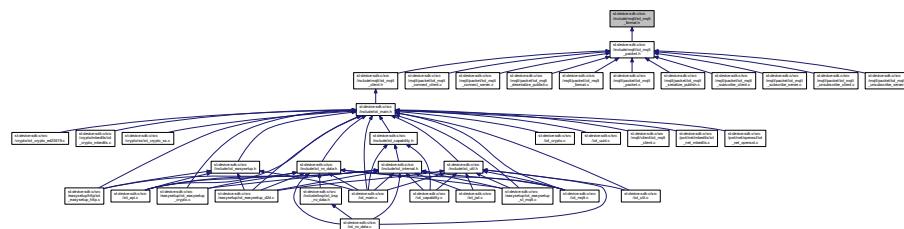


5.30 st-device-sdk-c/src/include/mqtt/iot_mqtt_format.h File Reference

```
#include "iot_mqtt_stacktrace.h"
Include dependency graph for iot_mqtt_format.h:
```



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



Functions

- const char * [MQTTPacket_getName](#) (unsigned short packetid)
- int [MQTTStringFormat_connect](#) (char *strbuf, int strbuflen, [MQTTPacket_connectData](#) *data)
- int [MQTTStringFormat_connack](#) (char *strbuf, int strbuflen, unsigned char connack_rc, unsigned char sessionPresent)
- int [MQTTStringFormat_publish](#) (char *strbuf, int strbuflen, unsigned char dup, int qos, unsigned char retained, unsigned short packetid, [MQTTString](#) topicName, unsigned char *payload, int payloadlen)
- int [MQTTStringFormat_ack](#) (char *strbuf, int strbuflen, unsigned char packettype, unsigned char dup, unsigned short packetid)
- int [MQTTStringFormat_subscribe](#) (char *strbuf, int strbuflen, unsigned char dup, unsigned short packetid, int count, [MQTTString](#) topicFilters[], int requestedQoSs[])
- int [MQTTStringFormat_suback](#) (char *strbuf, int strbuflen, unsigned short packetid, int count, int *grantedQoSs)
- int [MQTTStringFormat_unsubscribe](#) (char *strbuf, int strbuflen, unsigned char dup, unsigned short packetid, int count, [MQTTString](#) topicFilters[])
- char * [MQTTFormat_toClientString](#) (char *strbuf, int strbuflen, unsigned char *buf, int buflen)
- char * [MQTTFormat_toServerString](#) (char *strbuf, int strbuflen, unsigned char *buf, int buflen)

5.30.1 Function Documentation

5.30.1.1 [MQTTFormat_toClientString\(\)](#)

```
char* MQTTFormat_toClientString (
    char * strbuf,
    int strbuflen,
    unsigned char * buf,
    int buflen )
```

5.30.1.2 [MQTTFormat_toServerString\(\)](#)

```
char* MQTTFormat_toServerString (
    char * strbuf,
    int strbuflen,
    unsigned char * buf,
    int buflen )
```

5.30.1.3 MQTTPacket_getName() const char* MQTTPacket_getName (unsigned short packetid)

5.30.1.4 MQTTStringFormat_ack() int MQTTStringFormat_ack (char * strbuf, int strbuflen, unsigned char packettype, unsigned char dup, unsigned short packetid)

5.30.1.5 MQTTStringFormat_connack() int MQTTStringFormat_connack (char * strbuf, int strbuflen, unsigned char connack_rc, unsigned char sessionPresent)

5.30.1.6 MQTTStringFormat_connect() int MQTTStringFormat_connect (char * strbuf, int strbuflen, MQTTPacket_connectData * data)

5.30.1.7 MQTTStringFormat_publish() int MQTTStringFormat_publish (char * strbuf, int strbuflen, unsigned char dup, int qos, unsigned char retained, unsigned short packetid, MQTTString topicName, unsigned char * payload, int payloadlen)

5.30.1.8 MQTTStringFormat_suback() int MQTTStringFormat_suback (char * strbuf, int strbuflen, unsigned short packetid, int count, int * grantedQoSs)

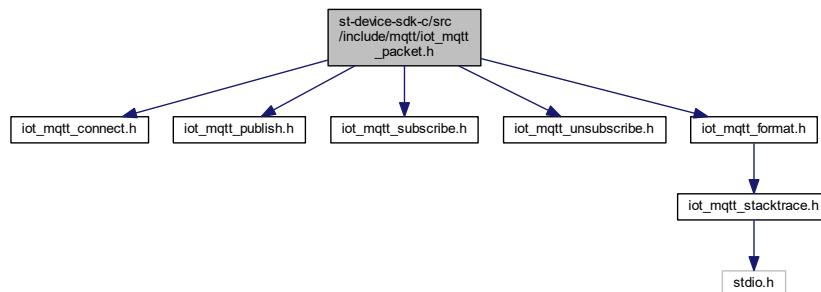
```
5.30.1.9 MQTTStringFormat_subscribe() int MQTTStringFormat_subscribe (
    char * strbuf,
    int strbuflen,
    unsigned char dup,
    unsigned short packetid,
    int count,
    MQTTString topicFilters[],
    int requestedQoSs[] )
```

```
5.30.1.10 MQTTStringFormat_unsubscribe() int MQTTStringFormat_unsubscribe (
    char * strbuf,
    int strbuflen,
    unsigned char dup,
    unsigned short packetid,
    int count,
    MQTTString topicFilters[] )
```

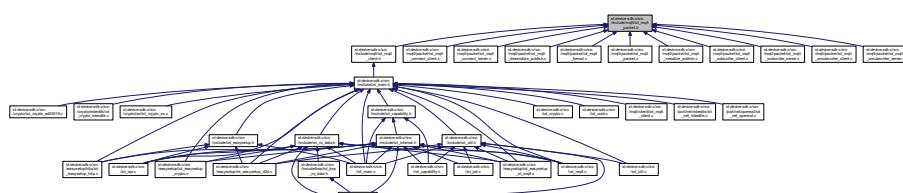
5.31 st-device-sdk-c/src/include/mqtt/iot_mqtt_packet.h File Reference

```
#include "iot_mqtt_connect.h"
#include "iot_mqtt_publish.h"
#include "iot_mqtt_subscribe.h"
#include "iot_mqtt_unsubscribe.h"
#include "iot_mqtt_format.h"
```

Include dependency graph for iot_mqtt_packet.h:



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



Data Structures

- union `MQTTHeader`
- struct `MQTTLenString`
- struct `MQTTString`
- struct `MQTTTransport`

Macros

- `#define DLLImport`
- `#define DLLExport`
- `#define MQTTString_initializer {NULL, {0, NULL}}`

Enumerations

- enum `errors` { `MQTTPACKET_BUFFER_TOO_SHORT` = -2, `MQTTPACKET_READ_ERROR` = -1, `MQTTPACKET_READ_COMPLETE` }
- enum `msgTypes` {
`CONNECT` = 1, `CONNACK`, `PUBLISH`, `PUBACK`,
`PUBREC`, `PUBREL`, `PUBCOMP`, `SUBSCRIBE`,
`SUBACK`, `UNSUBSCRIBE`, `UNSUBACK`, `PINGREQ`,
`PINGRESP`, `DISCONNECT` }

Functions

- int `MQTTstrlen` (`MQTTString` mqttstring)
- `DLLExport` int `MQTTSerialize_ack` (unsigned char *buf, int buflen, unsigned char type, unsigned char dup, unsigned short packetid)
- `DLLExport` int `MQTTDeserialize_ack` (unsigned char *packettype, unsigned char *dup, unsigned short *packetid, unsigned char *buf, int buflen)
- `DLLExport` int `MQTTSerialize_ack_size` ()
- int `MQTTPacket_len` (int rem_len)
- `DLLExport` int `MQTTPacket_equals` (`MQTTString` *a, char *b)
- `DLLExport` int `MQTTPacket_encode` (unsigned char *buf, int length)
- int `MQTTPacket_decode` (int(*getcharfn)(unsigned char *, int), int *value)
- int `MQTTPacket_decodeBuf` (unsigned char *buf, int *value)
- int `readInt` (unsigned char **pptr)
- char `readChar` (unsigned char **pptr)
- void `writeChar` (unsigned char **pptr, char c)
- void `writelnt` (unsigned char **pptr, int anInt)
- int `readMQTTLenString` (`MQTTString` *mqttstring, unsigned char **pptr, unsigned char *enddata)
- void `writeCString` (unsigned char **pptr, const char *string)
- void `writeMQTTString` (unsigned char **pptr, `MQTTString` mqttstring)
- `DLLExport` int `MQTTPacket_read` (unsigned char *buf, int buflen, int(*getfn)(unsigned char *, int))
- int `MQTTPacket_readnb` (unsigned char *buf, int buflen, `MQTTTransport` *trp)
- const char * `MQTTPacket_msgTypesToString` (enum `msgTypes`)

5.31.1 Macro Definition Documentation

5.31.1.1 DLLExport #define DLLExport**5.31.1.2 DLLImport** #define DLLImport**5.31.1.3 MQTTString_initializer** #define MQTTString_initializer {NULL, {0, NULL}}**5.31.2 Enumeration Type Documentation****5.31.2.1 errors** enum errors

Enumerator

MQTTPACKET_BUFFER_TOO_SHORT	
MQTTPACKET_READ_ERROR	
MQTTPACKET_READ_COMPLETE	

5.31.2.2 msgTypes enum msgTypes

Enumerator

CONNECT	
CONNACK	
PUBLISH	
PUBACK	
PUBREC	
PUBREL	
PUBCOMP	
SUBSCRIBE	
SUBACK	
UNSUBSCRIBE	
UNSUBACK	
PINGREQ	
PINGRESP	
DISCONNECT	

5.31.3 Function Documentation

5.31.3.1 MQTTDeserialize_ack() `DLLExport int MQTTDeserialize_ack (`
 `unsigned char * packettype,`
 `unsigned char * dup,`
 `unsigned short * packetid,`
 `unsigned char * buf,`
 `int buflen)`

Deserializes the supplied (wire) buffer into an ack

Parameters

<code>packettype</code>	returned integer - the MQTT packet type
<code>dup</code>	returned integer - the MQTT dup flag
<code>packetid</code>	returned integer - the MQTT packet identifier
<code>buf</code>	the raw buffer data, of the correct length determined by the remaining length field
<code>buflen</code>	the length in bytes of the data in the supplied buffer

Returns

error code. 1 is success, 0 is failure

5.31.3.2 MQTTPacket_decode() `int MQTTPacket_decode (`
 `int (*) (unsigned char *, int) getcharfn,`
 `int * value)`

Decodes the message length according to the MQTT algorithm

Parameters

<code>getcharfn</code>	pointer to function to read the next character from the data source
<code>value</code>	the decoded length returned

Returns

the number of bytes read from the socket

5.31.3.3 MQTTPacket_decodeBuf() `int MQTTPacket_decodeBuf (`
 `unsigned char * buf,`
 `int * value)`

5.31.3.4 MQTTPacket_encode() `DLLExport int MQTTPacket_encode (`
 `unsigned char * buf,`
 `int length)`

Encodes the message length according to the MQTT algorithm

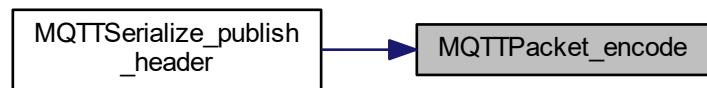
Parameters

<i>buf</i>	the buffer into which the encoded data is written
<i>length</i>	the length to be encoded

Returns

the number of bytes written to buffer

Here is the caller graph for this function:



5.31.3.5 MQTTPacket_equals() `DLLEXPORT int MQTTpacket_equals (`
`MQTTString * a,`
`char * bptr)`

Compares an [MQTTString](#) to a C string

Parameters

<i>a</i>	the MQTTString to compare
<i>bptr</i>	the C string to compare

Returns

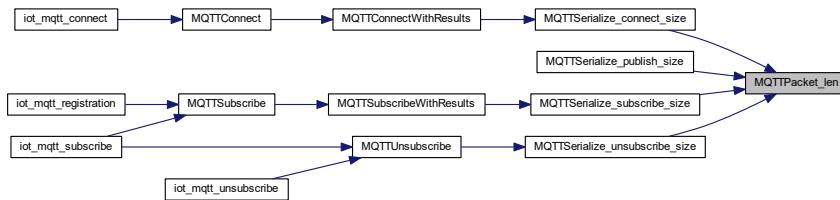
boolean - equal or not

Here is the caller graph for this function:



5.31.3.6 MQTTPacket_len() `int MQTTPacket_len (`
 `int rem_len)`

Here is the caller graph for this function:



5.31.3.7 MQTTPacket_msgTypesToString() `const char* MQTTPacket_msgTypesToString (`
 `enum msgTypes)`

5.31.3.8 MQTTPacket_read() `DLLExport int MQTTPacket_read (`
 `unsigned char * buf,`
 `int buflen,`
 `int(*)(unsigned char *, int) getfn)`

Helper function to read packet data from some source into a buffer

Parameters

<code>buf</code>	the buffer into which the packet will be serialized
<code>buflen</code>	the length in bytes of the supplied buffer
<code>getfn</code>	pointer to a function which will read any number of bytes from the needed source

Returns

integer MQTT packet type, or -1 on error

Note

the whole message must fit into the caller's buffer

5.31.3.9 MQTTPacket_readnb() `int MQTTPacket_readnb (`
 `unsigned char * buf,`
 `int buflen,`
 `MQTTTransport * trp)`

Helper function to read packet data from some source into a buffer, non-blocking

Parameters

<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer
<i>trp</i>	pointer to a transport structure holding what is needed to solve getting data from it

Returns

integer MQTT packet type, 0 for call again, or -1 on error

Note

the whole message must fit into the caller's buffer

5.31.3.10 MQTTSerialize_ack() `DLLExport int MQTTSerialize_ack (`

```
    unsigned char * buf,
    int buflen,
    unsigned char packettype,
    unsigned char dup,
    unsigned short packetid )
```

Serializes the ack packet into the supplied buffer.

Parameters

<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer
<i>type</i>	the MQTT packet type
<i>dup</i>	the MQTT dup flag
<i>packetid</i>	the MQTT packet identifier

Returns

serialized length, or error if 0

5.31.3.11 MQTTSerialize_ack_size() `DLLExport int MQTTSerialize_ack_size ()`**5.31.3.12 MQTTstrlen()** `int MQTTstrlen (`
`MQTTString mqttsstring)`

Return the length of the MQTTstring - C string if there is one, otherwise the length delimited string

Parameters

<i>mqttstring</i>	the string to return the length of
-------------------	------------------------------------

Returns

the length of the string

5.31.3.13 `readChar()` `char readChar (`
`unsigned char ** pptr)`

Reads one character from the input buffer.

Parameters

<i>pptr</i>	pointer to the input buffer - incremented by the number of bytes used & returned
-------------	--

Returns

the character read

5.31.3.14 `readInt()` `int readInt (`
`unsigned char ** pptr)`

Calculates an integer from two bytes read from the input buffer

Parameters

<i>pptr</i>	pointer to the input buffer - incremented by the number of bytes used & returned
-------------	--

Returns

the integer value calculated

5.31.3.15 `readMQTTLenString()` `int readMQTTLenString (`
`MQTTString * mqttstring,`
`unsigned char ** pptr,`
`unsigned char * enddata)`

Parameters

<i>mqttstring</i>	the MQTTString structure into which the data is to be read
<i>pptr</i>	pointer to the output buffer - incremented by the number of bytes used & returned
<i>enddata</i>	pointer to the end of the data: do not read beyond

Returns

1 if successful, 0 if not

5.31.3.16 writeChar() `void writeChar (`
 `unsigned char ** pptr,`
 `char c)`

Writes one character to an output buffer.

Parameters

<i>pptr</i>	pointer to the output buffer - incremented by the number of bytes used & returned
<i>c</i>	the character to write

Here is the caller graph for this function:



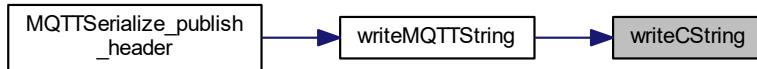
5.31.3.17 writeCString() `void writeCString (`
 `unsigned char ** pptr,`
 `const char * string)`

Writes a "UTF" string to an output buffer. Converts C string to length-delimited.

Parameters

<i>pptr</i>	pointer to the output buffer - incremented by the number of bytes used & returned
<i>string</i>	the C string to write

Here is the caller graph for this function:



5.31.3.18 writeInt() void writeInt (
 unsigned char ** pptr,
 int anInt)

Writes an integer as 2 bytes to an output buffer.

Parameters

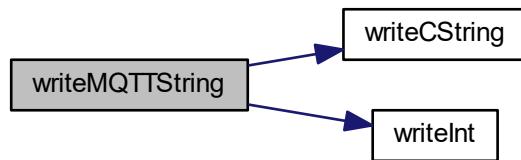
<i>pptr</i>	pointer to the output buffer - incremented by the number of bytes used & returned
<i>anInt</i>	the integer to write

Here is the caller graph for this function:

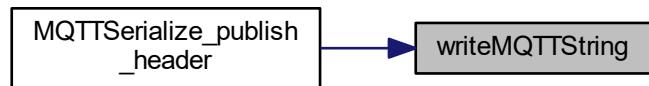


5.31.3.19 writeMQTTString() void writeMQTTString (
 unsigned char ** pptr,
 MQTTString mqttstring)

Here is the call graph for this function:

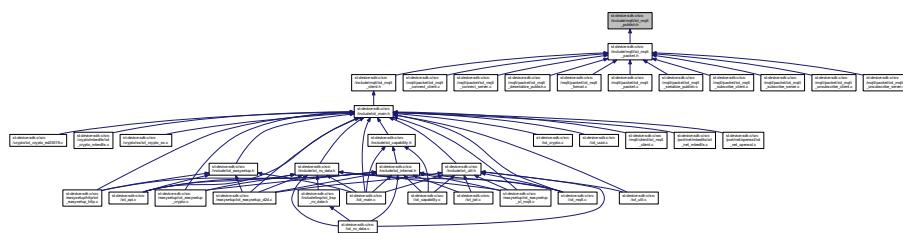


Here is the caller graph for this function:



5.32 st-device-sdk-c/src/include/mqtt/iot_mqtt_publish.h File Reference

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



Macros

- `#define DLLImport`
- `#define DLLExport`

Functions

- `DLLExport int MQTTSerialize_publish (unsigned char *buf, int buflen, unsigned char dup, int qos, unsigned char retained, unsigned short packetid, MQTTString topicName, unsigned char *payload, int payloadlen)`
- `DLLExport int MQTTSerialize_publish_size (int qos, MQTTString topicName, int payloadlen)`
- `DLLExport int MQTTSerialize_publish_header (unsigned char *buf, unsigned char dup, int qos, unsigned char retained, unsigned short packetid, MQTTString topicName, int payloadlen)`
- `DLLExport int MQTTDeserialize_publish (unsigned char *dup, int *qos, unsigned char *retained, unsigned short *packetid, MQTTString *topicName, unsigned char **payload, int *payloadlen, unsigned char *buf, int len)`
- `DLLExport int MQTTSerialize_puback (unsigned char *buf, int buflen, unsigned short packetid)`
- `DLLExport int MQTTSerialize_pubrel (unsigned char *buf, int buflen, unsigned char dup, unsigned short packetid)`
- `DLLExport int MQTTSerialize_pubcomp (unsigned char *buf, int buflen, unsigned short packetid)`

5.32.1 Macro Definition Documentation

5.32.1.1 `DLLExport` #define DLLExport

5.32.1.2 `DLLImport` #define DLLImport

5.32.2 Function Documentation

5.32.2.1 `MQTTDeserialize_publish()` `DLLExport int MQTTDeserialize_publish (`

```
    unsigned char * dup,
    int * qos,
    unsigned char * retained,
    unsigned short * packetid,
    MQTTString * topicName,
    unsigned char ** payload,
    int * payloadlen,
    unsigned char * buf,
    int buflen )
```

Deserializes the supplied (wire) buffer into publish data

Parameters

<code>dup</code>	returned integer - the MQTT dup flag
<code>qos</code>	returned integer - the MQTT QoS value
<code>retained</code>	returned integer - the MQTT retained flag
<code>packetid</code>	returned integer - the MQTT packet identifier
<code>topicName</code>	returned <code>MQTTString</code> - the MQTT topic in the publish
<code>payload</code>	returned byte buffer - the MQTT publish payload
<code>payloadlen</code>	returned integer - the length of the MQTT payload
<code>buf</code>	the raw buffer data, of the correct length determined by the remaining length field
<code>buflen</code>	the length in bytes of the data in the supplied buffer

Returns

error code. 1 is success

5.32.2.2 MQTTSerialize_puback() `DLLEXPORT int MQTTSerialize_puback (`

```
    unsigned char * buf,
    int buflen,
    unsigned short packetid )
```

Serializes a puback packet into the supplied buffer.

Parameters

<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer
<i>packetid</i>	integer - the MQTT packet identifier

Returns

serialized length, or error if 0

5.32.2.3 MQTTSerialize_pubcomp() `DLLEXPORT int MQTTSerialize_pubcomp (`

```
    unsigned char * buf,
    int buflen,
    unsigned short packetid )
```

Serializes a pubrel packet into the supplied buffer.

Parameters

<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer
<i>packetid</i>	integer - the MQTT packet identifier

Returns

serialized length, or error if 0

5.32.2.4 MQTTSerialize_publish() `DLLEXPORT int MQTTSerialize_publish (`

```
    unsigned char * buf,
    int buflen,
    unsigned char dup,
    int qos,
```

```

unsigned char retained,
unsigned short packetid,
MQTTString topicName,
unsigned char * payload,
int payloadlen )

```

Serializes the supplied publish data into the supplied buffer, ready for sending

Parameters

<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer
<i>dup</i>	integer - the MQTT dup flag
<i>qos</i>	integer - the MQTT QoS value
<i>retained</i>	integer - the MQTT retained flag
<i>packetid</i>	integer - the MQTT packet identifier
<i>topicName</i>	MQTTString - the MQTT topic in the publish
<i>payload</i>	byte buffer - the MQTT publish payload
<i>payloadlen</i>	integer - the length of the MQTT payload

Returns

the length of the serialized data. <= 0 indicates error

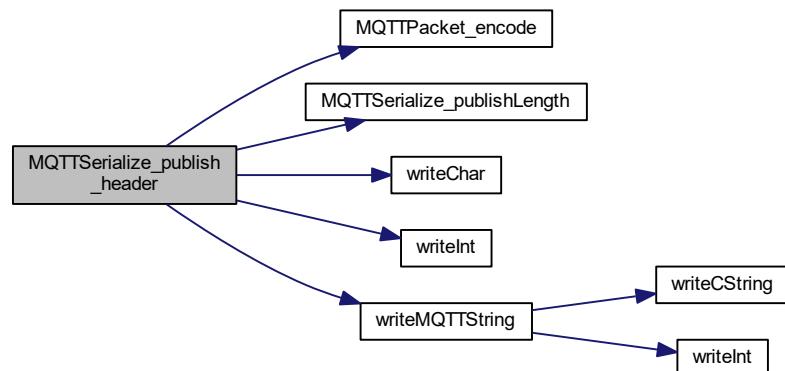
5.32.2.5 MQTTSerialize_publish_header()

```

unsigned char * buf,
unsigned char dup,
int qos,
unsigned char retained,
unsigned short packetid,
MQTTString topicName,
int payloadlen )

```

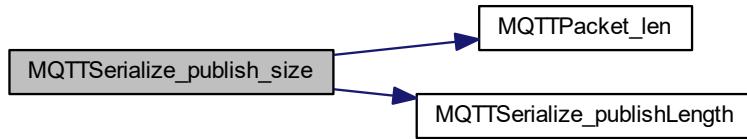
Here is the call graph for this function:



5.32.2.6 MQTTSerialize_publish_size()

```
DLLExport int MQTTSerialize_publish_size (
    int qos,
    MQTTString topicName,
    int payloadlen )
```

Here is the call graph for this function:

**5.32.2.7 MQTTSerialize_pubrel()**

```
DLLExport int MQTTSerialize_pubrel (
    unsigned char * buf,
    int buflen,
    unsigned char dup,
    unsigned short packetid )
```

Serializes a pubrel packet into the supplied buffer.

Parameters

<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer
<i>dup</i>	integer - the MQTT dup flag
<i>packetid</i>	integer - the MQTT packet identifier

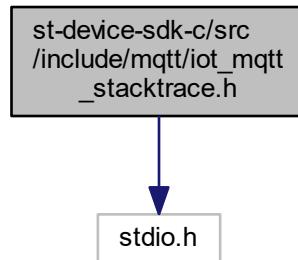
Returns

serialized length, or error if 0

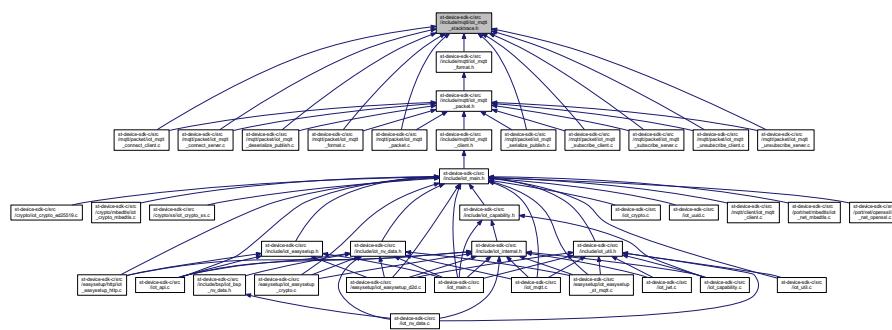
5.33 st-device-sdk-c/src/include/mqtt/iot_mqtt_stacktrace.h File Reference

```
#include <stdio.h>
```

Include dependency graph for iot_mqtt_stacktrace.h:



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



Macros

- `#define NOSTACKTRACE 1`
- `#define FUNC_ENTRY`
- `#define FUNC_ENTRY_NOLOG`
- `#define FUNC_ENTRY_MED`
- `#define FUNC_ENTRY_MAX`
- `#define FUNC_EXIT`
- `#define FUNC_EXIT_NOLOG`
- `#define FUNC_EXIT_MED`
- `#define FUNC_EXIT_MAX`
- `#define FUNC_EXIT_RC(x)`
- `#define FUNC_EXIT_MED_RC(x)`
- `#define FUNC_EXIT_MAX_RC(x)`

5.33.1 Macro Definition Documentation

5.33.1.1 FUNC_ENTRY #define FUNC_ENTRY

5.33.1.2 FUNC_ENTRY_MAX #define FUNC_ENTRY_MAX

5.33.1.3 FUNC_ENTRY_MED #define FUNC_ENTRY_MED

5.33.1.4 FUNC_ENTRY_NOLOG #define FUNC_ENTRY_NOLOG

5.33.1.5 FUNC_EXIT #define FUNC_EXIT

5.33.1.6 FUNC_EXIT_MAX #define FUNC_EXIT_MAX

5.33.1.7 FUNC_EXIT_MAX_RC #define FUNC_EXIT_MAX_RC(
 x)

5.33.1.8 FUNC_EXIT_MED #define FUNC_EXIT_MED

5.33.1.9 FUNC_EXIT_MED_RC #define FUNC_EXIT_MED_RC(
 x)

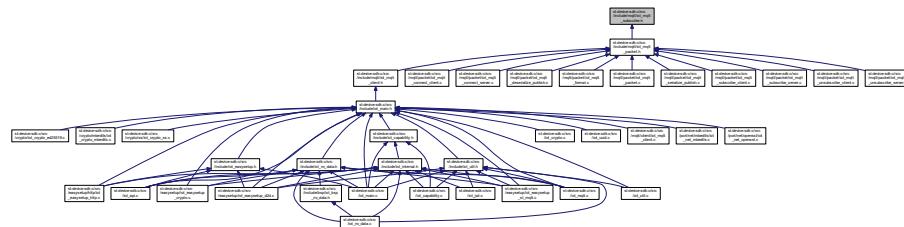
5.33.1.10 FUNC_EXIT_NOLOG #define FUNC_EXIT_NOLOG

5.33.1.11 FUNC_EXIT_RC #define FUNC_EXIT_RC(
 x)

5.33.1.12 NOSTACKTRACE #define NOSTACKTRACE 1

5.34 st-device-sdk-c/src/include/mqtt/iot_mqtt_subscribe.h File Reference

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



Macros

- `#define DLLImport`
- `#define DLLExport`

Functions

- `DLLExport int MQTTSerialize_subscribe (unsigned char *buf, int buflen, unsigned char dup, unsigned short packetid, int count, MQTTString topicFilters[], int requestedQoSs[])`
- `int MQTTSerialize_subscribe_size (int count, MQTTString topicFilters[])`
- `DLLExport int MQTTDeserialize_subscribe (unsigned char *dup, unsigned short *packetid, int maxcount, int *count, MQTTString topicFilters[], int requestedQoSs[], unsigned char *buf, int len)`
- `DLLExport int MQTTSerialize_suback (unsigned char *buf, int buflen, unsigned short packetid, int count, int *grantedQoSs)`
- `DLLExport int MQTTDeserialize_suback (unsigned short *packetid, int maxcount, int *count, int grantedQoSs[], unsigned char *buf, int len)`

5.34.1 Macro Definition Documentation

5.34.1.1 DLLExport #define DLLExport

5.34.1.2 DLLImport #define DLLImport

5.34.2 Function Documentation

```

5.34.2.1 MQTTDeserialize_suback() DLLExport int MQTTDeserialize_suback (
    unsigned short * packetid,
    int maxcount,
    int * count,
    int grantedQoSs[],
    unsigned char * buf,
    int buflen )
  
```

Deserializes the supplied (wire) buffer into suback data

Parameters

<i>packetid</i>	returned integer - the MQTT packet identifier
<i>maxcount</i>	- the maximum number of members allowed in the grantedQoSs array
<i>count</i>	returned integer - number of members in the grantedQoSs array
<i>grantedQoSs</i>	returned array of integers - the granted qualities of service
<i>buf</i>	the raw buffer data, of the correct length determined by the remaining length field
<i>buflen</i>	the length in bytes of the data in the supplied buffer

Returns

error code. 1 is success, 0 is failure

5.34.2.2 MQTTDeserialize_subscribe() `DLLEXPORT int MQTTDeserialize_subscribe (`

```
    unsigned char * dup,
    unsigned short * packetid,
    int maxcount,
    int * count,
    MQTTString topicFilters[],
    int requestedQoSs[],
    unsigned char * buf,
    int buflen )
```

Deserializes the supplied (wire) buffer into subscribe data

Parameters

<i>dup</i>	integer returned - the MQTT dup flag
<i>packetid</i>	integer returned - the MQTT packet identifier
<i>maxcount</i>	- the maximum number of members allowed in the topicFilters and requestedQoSs arrays
<i>count</i>	- number of members in the topicFilters and requestedQoSs arrays
<i>topicFilters</i>	- array of topic filter names
<i>requestedQoSs</i>	- array of requested QoS
<i>buf</i>	the raw buffer data, of the correct length determined by the remaining length field
<i>buflen</i>	the length in bytes of the data in the supplied buffer

Returns

the length of the serialized data. <= 0 indicates error

5.34.2.3 MQTTSerialize_suback() `DLLEXPORT int MQTTSerialize_suback (`

```
    unsigned char * buf,
    int buflen,
    unsigned short packetid,
```

```
int count,  
int * grantedQoSs )
```

Serializes the supplied suback data into the supplied buffer, ready for sending

Parameters

<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer
<i>packetid</i>	integer - the MQTT packet identifier
<i>count</i>	- number of members in the grantedQoSs array
<i>grantedQoSs</i>	- array of granted QoS

Returns

the length of the serialized data. <= 0 indicates error

5.34.2.4 MQTTSerialize_subscribe() [DLLExport](#) int MQTTSerialize_subscribe (

```
unsigned char * buf,
int buflen,
unsigned char dup,
unsigned short packetid,
int count,
MQTTString topicFilters[],
int requestedQoSs[] )
```

Serializes the supplied subscribe data into the supplied buffer, ready for sending

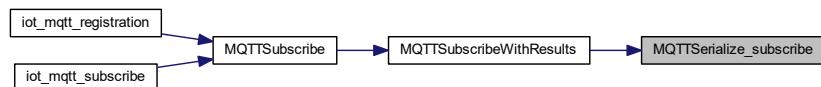
Parameters

<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer
<i>dup</i>	integer - the MQTT dup flag
<i>packetid</i>	integer - the MQTT packet identifier
<i>count</i>	- number of members in the topicFilters and reqQos arrays
<i>topicFilters</i>	- array of topic filter names
<i>requestedQoSs</i>	- array of requested QoS

Returns

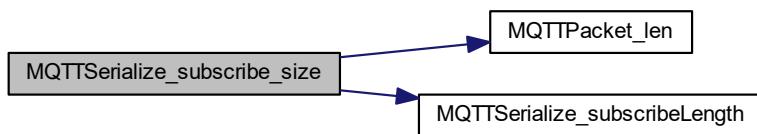
the length of the serialized data. <= 0 indicates error

Here is the caller graph for this function:

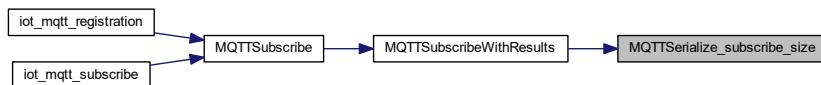


```
5.34.2.5 MQTTSerialize_subscribe_size() int MQTTSerialize_subscribe_size (
    int count,
    MQTTString topicFilters[] )
```

Here is the call graph for this function:

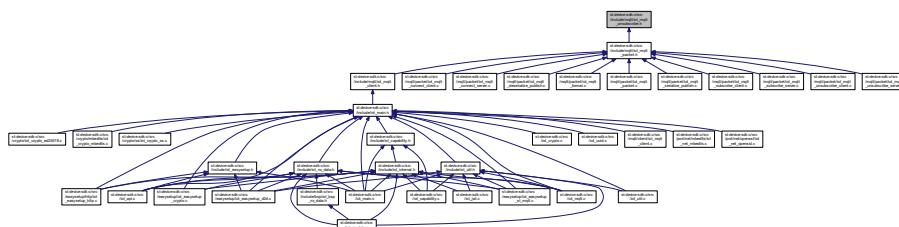


Here is the caller graph for this function:



5.35 st-device-sdk-c/src/include/mqtt/iot_mqtt_unsubscribe.h File Reference

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



Macros

- `#define DLLImport`
- `#define DLLExport`

Functions

- `DLLExport int MQTTSerialize_unsubscribe (unsigned char *buf, int buflen, unsigned char dup, unsigned short packetid, int count, MQTTString topicFilters[])`
- `DLLExport int MQTTSerialize_unsubscribe_size (int count, MQTTString topicFilters[])`
- `DLLExport int MQTTDeserialize_unsubscribe (unsigned char *dup, unsigned short *packetid, int max_count, int *count, MQTTString topicFilters[], unsigned char *buf, int len)`
- `DLLExport int MQTTSerialize_unsuback (unsigned char *buf, int buflen, unsigned short packetid)`
- `DLLExport int MQTTDeserialize_unsuback (unsigned short *packetid, unsigned char *buf, int len)`

5.35.1 Macro Definition Documentation

5.35.1.1 **DLLExport** #define DLLExport

5.35.1.2 **DLLImport** #define DLLImport

5.35.2 Function Documentation

5.35.2.1 MQTTDeserialize_unsuback() `DLLExport int MQTTDeserialize_unsuback (`
`unsigned short * packetid,`
`unsigned char * buf,`
`int buflen)`

Deserializes the supplied (wire) buffer into unsuback data

Parameters

<i>packetid</i>	returned integer - the MQTT packet identifier
<i>buf</i>	the raw buffer data, of the correct length determined by the remaining length field
<i>buflen</i>	the length in bytes of the data in the supplied buffer

Returns

error code. 1 is success, 0 is failure

5.35.2.2 MQTTDeserialize_unsubscribe() `DLLExport int MQTTDeserialize_unsubscribe (`
`unsigned char * dup,`
`unsigned short * packetid,`
`int maxcount,`
`int * count,`
`MQTTString topicFilters[],`
`unsigned char * buf,`
`int len)`

Deserializes the supplied (wire) buffer into unsubscribe data

Parameters

<i>dup</i>	integer returned - the MQTT dup flag
<i>packetid</i>	integer returned - the MQTT packet identifier

Parameters

<i>maxcount</i>	- the maximum number of members allowed in the topicFilters and requestedQoSs arrays
<i>count</i>	- number of members in the topicFilters and requestedQoSs arrays
<i>topicFilters</i>	- array of topic filter names
<i>buf</i>	the raw buffer data, of the correct length determined by the remaining length field
<i>buflen</i>	the length in bytes of the data in the supplied buffer

Returns

the length of the serialized data. <= 0 indicates error

5.35.2.3 MQTTSerialize_unsuback() `DLLExport int MQTTSerialize_unsuback (`

```
unsigned char * buf,
int buflen,
unsigned short packetid )
```

Serializes the supplied unsuback data into the supplied buffer, ready for sending

Parameters

<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer
<i>packetid</i>	integer - the MQTT packet identifier

Returns

the length of the serialized data. <= 0 indicates error

5.35.2.4 MQTTSerialize_unsubscribe() `DLLExport int MQTTSerialize_unsubscribe (`

```
unsigned char * buf,
int buflen,
unsigned char dup,
unsigned short packetid,
int count,
MQTTString topicFilters[] )
```

Serializes the supplied unsubscribe data into the supplied buffer, ready for sending

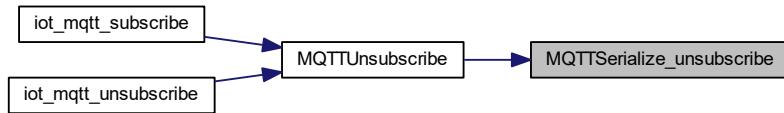
Parameters

<i>buf</i>	the raw buffer data, of the correct length determined by the remaining length field
<i>buflen</i>	the length in bytes of the data in the supplied buffer
<i>dup</i>	integer - the MQTT dup flag
<i>packetid</i>	integer - the MQTT packet identifier
<i>count</i>	- number of members in the topicFilters array
<i>topicFilters</i>	- array of topic filter names

Returns

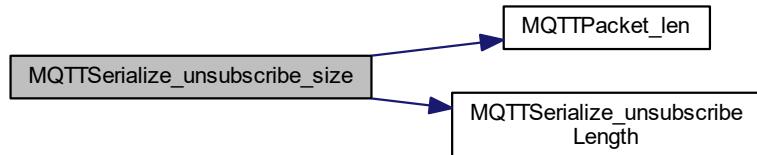
the length of the serialized data. <= 0 indicates error

Here is the caller graph for this function:

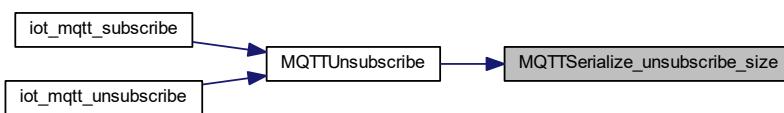


```
5.35.2.5 MQTTSerialize_unsubscribe_size() DLLExport int MQTTSerialize_unsubscribe_size (  
    int count,  
    MQTTString topicFilters[] )
```

Here is the call graph for this function:

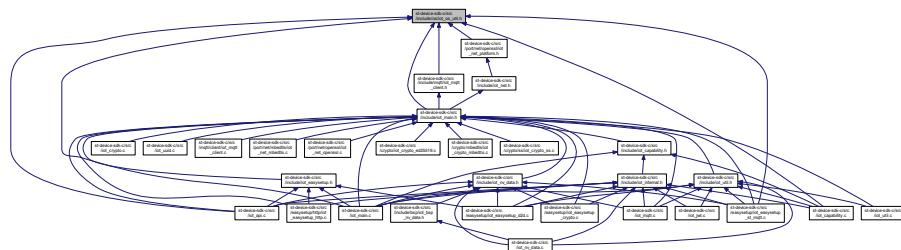


Here is the caller graph for this function:



5.36 st-device-sdk-c/src/include/os/iot_os_util.h File Reference

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



Data Structures

- struct `iot_os_mutex`
Contains a mutex data.

Macros

- #define `IOT_DELAY(x)` `iot_os_delay(x)`
- #define `IOT_OS_MAX_DELAY` `iot_os_max_delay`
- #define `IOT_OS_TRUE` `iot_os_true`
- #define `IOT_OS_FALSE` `iot_os_false`

Typedefs

- typedef void * `iot_os_thread`
- typedef void `iot_os_queue`
- typedef void `iot_os_eventgroup`
- typedef void `iot_os_sem`
- typedef void * `iot_os_timer`
- typedef struct `iot_os_mutex` `iot_os_mutex`

Contains a mutex data.

Functions

- int `iot_os_thread_create` (void *`thread_function`, const char *`name`, int `stack_size`, void *`data`, int `priority`, `iot_os_thread` *`thread_handle`)
create and run thread
- void `iot_os_thread_delete` (`iot_os_thread` `thread_handle`)
delete thread
- void `iot_os_thread_yield` ()
yield task
- `iot_os_queue` * `iot_os_queue_create` (int `queue_length`, int `item_size`)
create queue
- int `iot_os_queue_reset` (`iot_os_queue` *`queue_handle`)
reset queue
- void `iot_os_queue_delete` (`iot_os_queue` *`queue_handle`)

- `int iot_os_queue_send (iot_os_queue *queue_handle, void *data, unsigned int wait_time_ms)`
send message to the back of queue.
- `int iot_os_queue_receive (iot_os_queue *queue_handle, void *data, unsigned int wait_time_ms)`
receive message from the front of queue.
- `iot_os_eventgroup * iot_os_eventgroup_create (void)`
create eventgroup
- `void iot_os_eventgroup_delete (iot_os_eventgroup *eventgroup_handle)`
delete eventgroup
- `unsigned int iot_os_eventgroup_wait_bits (iot_os_eventgroup *eventgroup_handle, const unsigned int bits_to_wait_for, const int clear_on_exit, const int wait_for_all_bits, const unsigned int wait_time_ms)`
wait for bit of group of bits to become set
- `unsigned int iot_os_eventgroup_set_bits (iot_os_eventgroup *eventgroup_handle, const unsigned int bits_to_set)`
set bit/bits of eventgroup
- `unsigned int iot_os_eventgroup_clear_bits (iot_os_eventgroup *eventgroup_handle, const unsigned int bits_to_clear)`
clear bit/bits of eventgroup
- `int iot_os_mutex_init (iot_os_mutex *mutex)`
create mutex
- `int iot_os_mutex_lock (iot_os_mutex *mutex)`
mutex lock
- `int iot_os_mutex_unlock (iot_os_mutex *mutex)`
mutex unlock
- `void iot_os_delay (unsigned int delay_ms)`
delay a thread
- `void iot_os_timer_init (iot_os_timer *timer)`
init timer
- `char iot_os_timer_isexpired (iot_os_timer timer)`
check timer expired
- `void iot_os_timer_count_ms (iot_os_timer timer, unsigned int timeout_ms)`
set timer count
- `unsigned int iot_os_timer_left_ms (iot_os_timer timer)`
return remaining time in ms unit
- `void iot_os_timer_destroy (iot_os_timer *timer)`
destroy timer

Variables

- `const unsigned int iot_os_max_delay`
- `const unsigned int iot_os_true`
- `const unsigned int iot_os_false`

5.36.1 Macro Definition Documentation

5.36.1.1 IOT_DELAY #define IOT_DELAY(x) iot_os_delay(x)

5.36.1.2 IOT_OS_FALSE #define IOT_OS_FALSE iot_os_false

5.36.1.3 IOT_OS_MAX_DELAY #define IOT_OS_MAX_DELAY iot_os_max_delay

5.36.1.4 IOT_OS_TRUE #define IOT_OS_TRUE iot_os_true

5.36.2 Typedef Documentation

5.36.2.1 iot_os_eventgroup typedef void iot_os_eventgroup

5.36.2.2 iot_os_mutex typedef struct iot_os_mutex iot_os_mutex

Contains a mutex data.

5.36.2.3 iot_os_queue typedef void iot_os_queue

5.36.2.4 iot_os_sem typedef void iot_os_sem

5.36.2.5 iot_os_thread typedef void* iot_os_thread

5.36.2.6 iot_os_timer typedef void* iot_os_timer

5.36.3 Function Documentation

5.36.3.1 iot_os_delay() void iot_os_delay (unsigned int delay_ms)

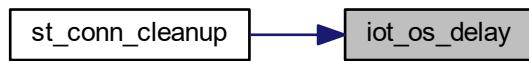
delay a thread

This function will delay thread for given time

Parameters

in	<i>delay</i>	time to wait
----	--------------	--------------

Here is the caller graph for this function:



5.36.3.2 iot_os_eventgroup_clear_bits() `unsigned int iot_os_eventgroup_clear_bits (`
 `iot_os_eventgroup * eventgroup_handle,`
 `const unsigned int bits_to_clear)`

clear bit/bits of eventgroup

This function will clear bit/bits of eventgroup

Parameters

in	<i>eventgroup_handle</i>	handle of eventgroup to clear bit/bits
in	<i>bits_to_clear</i>	bitwise value of bit/bits to clear ex) use 0b101 will clear bit0 and bit2 ex) use 0b100 will clear only bit2

Returns

return is bits of event BEFORE clear bit.

5.36.3.3 iot_os_eventgroup_create() `iot_os_eventgroup* iot_os_eventgroup_create (`
 `void)`

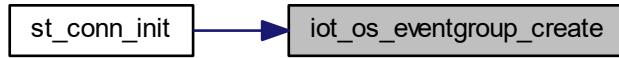
create eventgroup

This function create eventgroup and return eventgroup handle

Returns

return is eventgroup handle of eventgroup created by this function If eventgroup was not created, NULL is returned.

Here is the caller graph for this function:



5.36.3.4 iot_os_eventgroup_delete() `void iot_os_eventgroup_delete (`
 `iot_os_eventgroup * eventgroup_handle)`

delete eventgroup

This function delete eventgroup

Parameters

in	<code>eventgroup_handle</code>	eventgroup handle of eventgroup to delete
----	--------------------------------	---

5.36.3.5 iot_os_eventgroup_set_bits() `unsigned int iot_os_eventgroup_set_bits (`
 `iot_os_eventgroup * eventgroup_handle,`
 `const unsigned int bits_to_set)`

set bit/bits of eventgroup

This function will set bit/bits of eventgroup

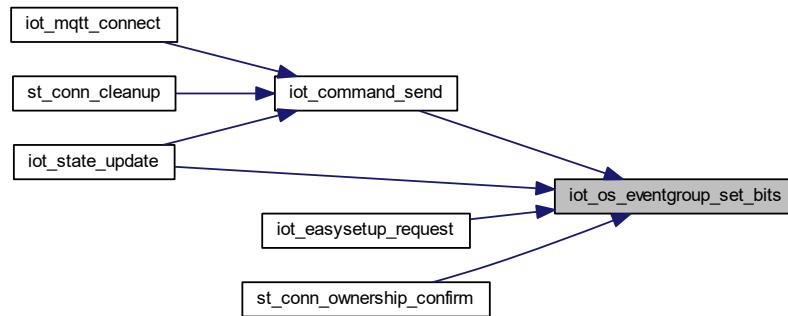
Parameters

in	<code>eventgroup_handle</code>	handle of eventgroup to set bit/bits
in	<code>bits_to_set</code>	bitwise value of bit/bits to set ex) use 0b101 will set bit0 and bit2 ex) use 0b100 will set only bit2

Returns

return is bits of event after set bit/bits. if you set 'clear_on_exit' to IOT_OS_TRUE, it can be 0 because of other task waiting for bits

Here is the caller graph for this function:



5.36.3.6 iot_os_eventgroup_wait_bits() `unsigned int iot_os_eventgroup_wait_bits (`
 `iot_os_eventgroup * eventgroup_handle,`
 `const unsigned int bits_to_wait_for,`
 `const int clear_on_exit,`
 `const int wait_for_all_bits,`
 `const unsigned int wait_time_ms)`

wait for bit of group of bits to become set

This function will wait for event bit

Parameters

in	<code>eventgroup_handle</code>	handle of eventgroup waiting for
in	<code>bits_to_wait_for</code>	bitwise value of bit/bits. ex) use 0b101 to wait bit 0 and/or bit 2
in	<code>clear_on_exit</code>	if this value is IOT_OS_TRUE, bits in 'bits_to_wait_for' will be cleared
in	<code>wait_for_all_bits</code>	if this value is IOT_OS_TRUE, wait for ""ALL"" bits in 'bits_to_wait_for'
in	<code>wait_time_ms</code>	maximum time to wait until all/one of bits are set

Returns

return is bits of eventgroup. if you wait for bit0 and bit3, can check with was setted, return will be 0b101 = 5

5.36.3.7 iot_os_mutex_init() `int iot_os_mutex_init (`
 `iot_os_mutex * mutex)`

create mutex

This function will create mutex

Parameters

<code>out</code>	<code>mutex</code>	handle of mutex created by this function
------------------	--------------------	--

Returns

`IOT_ERROR_NONE` : success otherwise : fail

Here is the caller graph for this function:



5.36.3.8 `iot_os_mutex_lock()` `int iot_os_mutex_lock (`
 `iot_os_mutex * mutex)`

mutex lock

This function will lock mutex before critical section

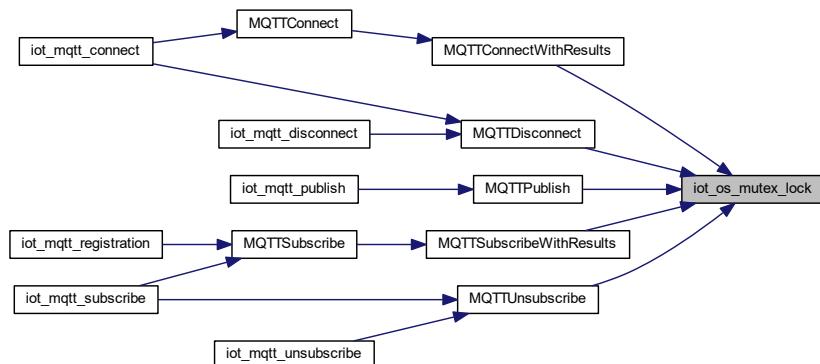
Parameters

<code>in</code>	<code>mutex</code>	handle of mutex
-----------------	--------------------	-----------------

Returns

`IOT_ERROR_NONE` : success otherwise : fail

Here is the caller graph for this function:



5.36.3.9 iot_os_mutex_unlock() `int iot_os_mutex_unlock (`
`iot_os_mutex * mutex)`

mutex unlock

This function will unlock mutex after critical section

Parameters

in	<i>mutex</i>	handle of mutex
----	--------------	-----------------

Returns

IOT_ERROR_NONE : success otherwise : fail

5.36.3.10 iot_os_queue_create() `iot_os_queue* iot_os_queue_create (`
`int queue_length,`
`int item_size)`

create queue

This function create queue and return queue handle

Parameters

in	<i>queue_length</i>	maximum number of queue item
in	<i>item_size</i>	size of item

Returns

return is queue handle of queue created by this function If queue was not created, NULL is returned.

Here is the caller graph for this function:



5.36.3.11 iot_os_queue_delete() void iot_os_queue_delete (
 iot_os_queue * queue_handle)

delete queue

This function delete queue

Parameters

in	queue_handle	handle of queue to be deleted.
----	--------------	--------------------------------

5.36.3.12 iot_os_queue_receive() int iot_os_queue_receive (
 iot_os_queue * queue_handle,
 void * data,
 unsigned int wait_time_ms)

receive message from the front of queue.

This function will receive item from the front of queue

Parameters

in	queue_handle	handle of queue to receive item
in	data	buffer for item received from queue
in	wait_time_ms	maximum time to wait until queue is not empty

Returns

IOT_OS_TRUE : success otherwise : fail

5.36.3.13 iot_os_queue_reset() int iot_os_queue_reset (
 iot_os_queue * queue_handle)

reset queue

This function reset queue

Parameters

in	queue_handle	handle of queue to reset.
----	--------------	---------------------------

Returns

IOT_OS_TRUE : success otherwise : fail

```
5.36.3.14 iot_os_queue_send() int iot_os_queue_send (
    iot_os_queue * queue_handle,
    void * data,
    unsigned int wait_time_ms )
```

send message to the back of queue.

This function will send item to the back of queue

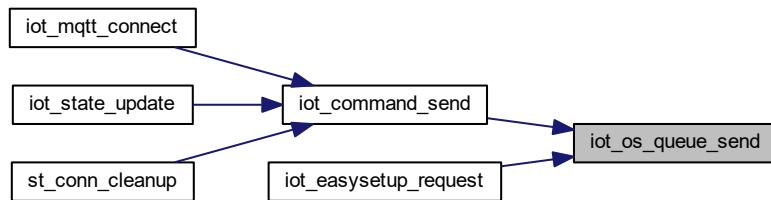
Parameters

in	<i>queue_handle</i>	handle of queue to save item
in	<i>data</i>	item to be saved in queue
in	<i>wait_time_ms</i>	maximum time to wait until queue is available

Returns

IOT_OS_TRUE : success otherwise : fail

Here is the caller graph for this function:



```
5.36.3.15 iot_os_thread_create() int iot_os_thread_create (
    void * thread_function,
    const char * name,
    int stack_size,
    void * data,
    int priority,
    iot_os_thread * thread_handle )
```

create and run thread

This function creates and runs thread

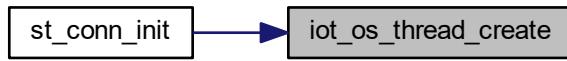
Parameters

in	<i>thread_function</i>	function pointer for thread loop
in	<i>name</i>	name of thread
in	<i>stack_size</i>	stack size of thread
in	<i>data</i>	data that will passed into the thread as the parameter
in	<i>priority</i>	priority of thread created by this function
out	<i>thread_handle</i>	thread handle of thread created by this function (optional, NULL is possible)

Returns

IOT_OS_TRUE : success otherwise : fail

Here is the caller graph for this function:



5.36.3.16 iot_os_thread_delete() void iot_os_thread_delete (
 iot_os_thread thread_handle)

delete thread

This function delete thread

Parameters

in	<i>thread_handle</i>	thread handle of thread to delete(if NULL is passed, delete current thread)
----	----------------------	---

5.36.3.17 iot_os_thread_yield() void iot_os_thread_yield ()

yield task

This function yields task

5.36.3.18 iot_os_timer_count_ms() void iot_os_timer_count_ms (
 iot_os_timer timer,
 unsigned int timeout_ms)

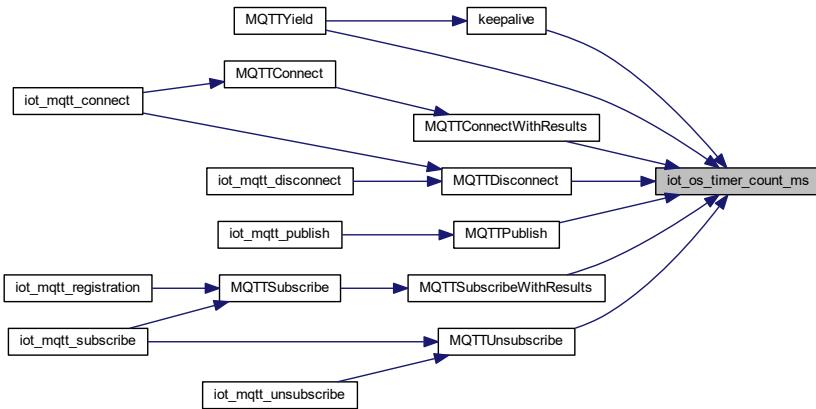
set timer count

This function will set timer count in ms unit.

Parameters

in	<i>timer</i>	timer handle
in	<i>count</i>	count to set in ms unit

Here is the caller graph for this function:



5.36.3.19 `iot_os_timer_destroy()` `void iot_os_timer_destroy (`
 `iot_os_timer * timer)`

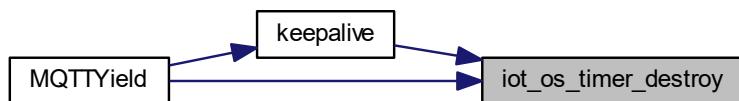
destroy timer

This function will destroy timer struct

Parameters

in	<i>timer</i>	pointer of timer to destroy
----	--------------	-----------------------------

Here is the caller graph for this function:



5.36.3.20 `iot_os_timer_init()` `void iot_os_timer_init (`
 `iot_os_timer * timer)`

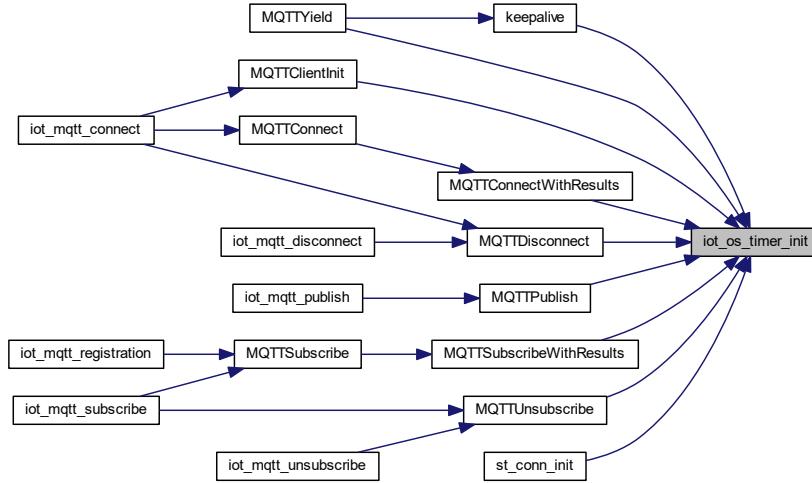
init timer

This function will init timer struct

Parameters

in	<i>timer</i>	pointer of timer to init
----	--------------	--------------------------

Here is the caller graph for this function:



5.36.3.21 `iot_os_timer_isexpired()` `char iot_os_timer_isexpired (`
 `iot_os_timer timer)`

check timer expired

This function will check if timer is expired

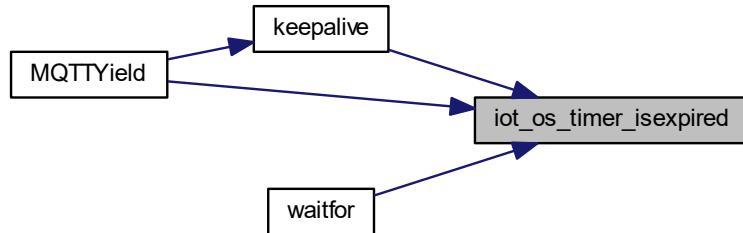
Parameters

in	<i>timer</i>	timer handle
----	--------------	--------------

Returns

1 : timer is expired 0 : timer is not expired

Here is the caller graph for this function:



5.36.3.22 iot_os_timer_left_ms() `unsigned int iot_os_timer_left_ms (iot_os_timer timer)`

return remaining time in ms unit

This function will return remaining time in ms unit

Parameters

in	<i>timer</i>	timer handle
----	--------------	--------------

Returns

0 : timer is expired non-zero : remaining time

Here is the caller graph for this function:



5.36.4 Variable Documentation

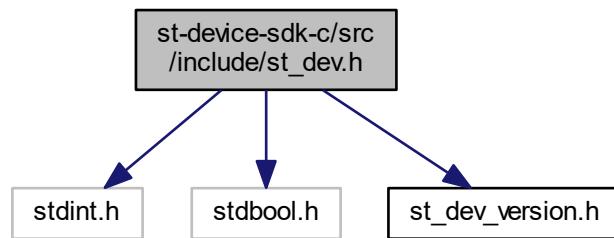
5.36.4.1 iot_os_false const unsigned int iot_os_false

5.36.4.2 iot_os_max_delay const unsigned int iot_os_max_delay

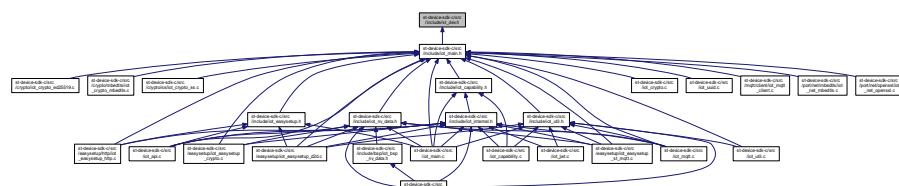
5.36.4.3 iot_os_true const unsigned int iot_os_true

5.37 st-device-sdk-c/src/include/st_dev.h File Reference

```
#include <stdint.h>
#include <stdbool.h>
#include "st_dev_version.h"
Include dependency graph for st_dev.h:
```



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



Data Structures

- struct `iot_pin_t`
Contains a pin values for pin type onboarding process.
- struct `iot_cap_val_t`
Contains a various type of data which can be int, double, string and string array.
- struct `iot_cap_cmd_data_t`

- union `noti_data_raw_t`

Contains data for "command" payload.
- struct `noti_data_raw_t::rate_limit`

Contains data for raw data of each notification.
- struct `noti_data_raw_t::quota`
- struct `iot_noti_data_t`

Contains data for notification data.

Macros

- `#define MAX_CAP_ARG (5)`

TypeDefs

- typedef enum `iot_status` `iot_status_t`

Contains a enumeration values for types of iot_status.
- typedef enum `iot_stat_lv` `iot_stat_lv_t`

Contains a enumeration values for types of iot_status level.
- typedef void * `IOT_CTX`
- typedef void * `IOT_CAP_HANDLE`
- typedef void * `IOT_EVENT`
- typedef struct `iot_pin_t` `iot_pin_t`

Contains a pin values for pin type onboarding process.
- typedef enum `iot_cap_val_type` `iot_cap_val_type_t`

Contains a enumeration values for types of capability.
- typedef enum `iot_noti_type` `iot_noti_type_t`

Contains a enumeration values for types of notification.
- typedef void(* `st_status_cb`) (`iot_status_t` `iot_status`, `iot_stat_lv_t` `stat_lv`, void *`usr_data`)
- typedef void(* `st_cap_init_cb`) (`IOT_CAP_HANDLE` *`cap_handle`, void *`init_usr_data`)
- typedef void(* `st_cap_noti_cb`) (`iot_noti_data_t` *`noti_data`, void *`noti_usr_data`)
- typedef void(* `st_cap_cmd_cb`) (`IOT_CAP_HANDLE` *`cap_handle`, `iot_cap_cmd_data_t` *`cmd_data`, void *`usr_data`)

Enumerations

- enum `iot_status` {

`IOT_STATUS_IDLE` = (1 << 0), `IOT_STATUS_PROVISIONING` = (1 << 1), `IOT_STATUS_NEED_INTERACT` = (1 << 2), `IOT_STATUS_CONNECTING` = (1 << 3),
`IOT_STATUS_ALL` }

Contains a enumeration values for types of iot_status.
 - enum `iot_stat_lv` { `IOT_STAT_LV_STAY`, `IOT_STAT_LV_START`, `IOT_STAT_LV_DONE`, `IOT_STAT_LV_FAIL` }
 - enum `iot_cap_val_type` {

`IOT_CAP_VAL_TYPE_UNKNOWN` = -1, `IOT_CAP_VAL_TYPE_INTEGER`, `IOT_CAP_VAL_TYPE_NUMBER`,
`IOT_CAP_VAL_TYPE_INT_OR_NUM`,
`IOT_CAP_VAL_TYPE_STRING`, `IOT_CAP_VAL_TYPE_STR_ARRAY` }

Contains a enumeration values for types of capability.
 - enum `iot_noti_type` { `IOT_NOTI_TYPE_UNKNOWN` = -1, `IOT_NOTI_TYPE_DEV_DELETED`, `IOT_NOTI_TYPE_RATE_LIMIT`,
`IOT_NOTI_TYPE_QUOTA_REACHED` }
- Contains a enumeration values for types of notification.*

Functions

- `IOT_EVENT * st_cap_attr_create_int (char *attribute, int integer, char *unit)`
Create IOT_EVENT data with integer value.
- `IOT_EVENT * st_cap_attr_create_number (char *attribute, double number, char *unit)`
Create IOT_EVENT data with real number(double) value.
- `IOT_EVENT * st_cap_attr_create_string (char *attribute, char *string, char *unit)`
Create IOT_EVENT data with string value.
- `IOT_EVENT * st_cap_attr_create_string_array (char *attribute, uint8_t str_num, char *string_array[], char *unit)`
Create IOT_EVENT data with string array value.
- `void st_cap_attr_free (IOT_EVENT *event)`
Free IOT_EVENT data.
- `int st_cap_attr_send (IOT_CAP_HANDLE *cap_handle, uint8_t evt_num, IOT_EVENT *event[])`
Request to publish deviceEvent.
- `IOT_CAP_HANDLE * st_cap_handle_init (IOT_CTX *iot_ctx, const char *component, const char *capability, st_cap_init_cb init_cb, void *init_usr_data)`
Create and initialize a capability handle.
- `int st_conn_set_noti_cb (IOT_CTX *iot_ctx, st_cap_noti_cb noti_cb, void *noti_usr_data)`
Register callback function for notification event.
- `int st_cap_cmd_set_cb (IOT_CAP_HANDLE *cap_handle, const char *cmd_type, st_cap_cmd_cb cmd_cb, void *usr_data)`
Register callback function for command message.
- `IOT_CTX * st_conn_init (unsigned char *onboarding_config, unsigned int onboarding_config_len, unsigned char *device_info, unsigned int device_info_len)`
st-iot-core initialize function
- `int st_conn_start (IOT_CTX *iot_ctx, st_status_cb status_cb, iot_status_t maps, void *usr_data, iot_pin_t *pin_num)`
st-iot-core server connection function
- `int st_conn_cleanup (IOT_CTX *iot_ctx, bool reboot)`
st-iot-core device clean-up function
- `void st_conn_ownership_confirm (IOT_CTX *iot_ctx, bool confirm)`
easysetup user confirm report function

5.37.1 Macro Definition Documentation

5.37.1.1 MAX_CAP_ARG #define MAX_CAP_ARG (5)

5.37.2 Typedef Documentation

5.37.2.1 IOT_CAP_HANDLE `typedef void* IOT_CAP_HANDLE`

5.37.2.2 iot_cap_val_type_t `typedef enum iot_cap_val_type iot_cap_val_type_t`

Contains a enumeration values for types of capability.

5.37.2.3 IOT_CTX `typedef void* IOT_CTX`**5.37.2.4 IOT_EVENT** `typedef void* IOT_EVENT`**5.37.2.5 iot_noti_type_t** `typedef enum iot_noti_type iot_noti_type_t`

Contains a enumeration values for types of notification.

5.37.2.6 iot_pin_t `typedef struct iot_pin_t iot_pin_t`

Contains a pin values for pin type onboarding process.

5.37.2.7 iot_stat_lv_t `typedef enum iot_stat_lv iot_stat_lv_t`

Contains a enumeration values for types of iot_status level.

5.37.2.8 iot_status_t `typedef enum iot_status iot_status_t`

Contains a enumeration values for types of iot_status.

5.37.2.9 st_cap_cmd_cb `typedef void(* st_cap_cmd_cb) (IOT_CAP_HANDLE *cap_handle, iot_cap_cmd_data_t *cmd_data, void *usr_data)`**5.37.2.10 st_cap_init_cb** `typedef void(* st_cap_init_cb) (IOT_CAP_HANDLE *cap_handle, void *init↔_usr_data)`

5.37.2.11 st_cap_noti_cb `typedef void(* st_cap_noti_cb) (iot_noti_data_t *noti_data, void *noti_usr_data)`

5.37.2.12 st_status_cb `typedef void(* st_status_cb) (iot_status_t iot_status, iot_stat_lv_t stat_lv, void *usr_data)`

5.37.3 Enumeration Type Documentation

5.37.3.1 iot_cap_val_type enum `iot_cap_val_type`

Contains a enumeration values for types of capability.

Enumerator

IOT_CAP_VAL_TYPE_UNKNOWN	For undefined type.
IOT_CAP_VAL_TYPE_INTEGER	For integer.
IOT_CAP_VAL_TYPE_NUMBER	For float number.
IOT_CAP_VAL_TYPE_INT_OR_NUM	For integer or float number.
IOT_CAP_VAL_TYPE_STRING	For NULL-terminated string.
IOT_CAP_VAL_TYPE_STR_ARRAY	For array of NULL-terminated strings.

5.37.3.2 iot_noti_type enum `iot_noti_type`

Contains a enumeration values for types of notification.

Enumerator

IOT_NOTI_TYPE_UNKNOWN	For undefined type.
IOT_NOTI_TYPE_DEV_DELETED	For device deleted event.
IOT_NOTI_TYPE_RATE_LIMIT	For rate limit event.
IOT_NOTI_TYPE_QUOTA_REACHED	For data quota reached event.

5.37.3.3 iot_stat_lv enum `iot_stat_lv`

Contains a enumeration values for types of iot_status level.

Enumerator

IOT_STAT_LV_STAY	meanings for staying level with each status
------------------	---

Enumerator

IOT_STAT_LV_START	meanings for start level with each status
IOT_STAT_LV_DONE	meanings for done level with each status
IOT_STAT_LV_FAIL	meanings for fail level with each status

5.37.3.4 iot_status enum iot_status

Contains a enumeration values for types of iot_status.

Enumerator

IOT_STATUS_IDLE	For idle status, not connected.
IOT_STATUS_PROVISIONING	For provisioning status. do onboarding process.
IOT_STATUS_NEED_INTERACT	For user interation status. need to interact with user.
IOT_STATUS_CONNECTING	For server connecting status. do connecting server.
IOT_STATUS_ALL	

5.37.4 Function Documentation

```
5.37.4.1 st_cap_attr_create_int() IOT_EVENT* st_cap_attr_create_int (
    char * attribute,
    int integer,
    char * unit )
```

Create IOT_EVENT data with integer value.

This function creates a new IOT_EVENT data with input parameters. Once it returns, user has full responsibility for deallocating event data by using [st_cap_attr_free](#).

Parameters

in	<i>attribute</i>	The attribute string of IOT_EVENT data.
in	<i>integer</i>	The integer to add to IOT_EVENT data.
in	<i>unit</i>	The unit string if needed. Otherwise NULL.

Returns

Pointer of IOT_EVENT which is used to publish device status.

Warning

Must call [st_cap_attr_free](#) to free IOT_EVENT data after using it.

See also

[st_cap_attr_send](#)

5.37.4.2 st_cap_attr_create_number() [IOT_EVENT*](#) st_cap_attr_create_number (

```
    char * attribute,  
    double number,  
    char * unit )
```

Create IOT_EVENT data with real number(double) value.

This function creates a new IOT_EVENT data with input parameters. Once it returns, user has full responsibility for deallocating event data by using [st_cap_attr_free](#).

Parameters

in	<i>attribute</i>	The attribute string of IOT_EVENT data.
in	<i>number</i>	The double number to add to IOT_EVENT data.
in	<i>unit</i>	The unit string if needed. Otherwise NULL.

Returns

Pointer of IOT_EVENT which is used to publish device status.

Warning

Must call [st_cap_attr_free](#) to free IOT_EVENT data after using it.

See also

[st_cap_attr_send](#)

5.37.4.3 st_cap_attr_create_string() [IOT_EVENT*](#) st_cap_attr_create_string (

```
    char * attribute,  
    char * string,  
    char * unit )
```

Create IOT_EVENT data with string value.

This function creates a new IOT_EVENT data with input parameters. Once it returns, user has full responsibility for deallocating event data by using [st_cap_attr_free](#).

Parameters

in	<i>attribute</i>	The attribute string of IOT_EVENT data.
in	<i>string</i>	The string to add to IOT_EVENT data.
in	<i>unit</i>	The unit string if needed. Otherwise NULL.

Returns

Pointer of IOT_EVENT which is used to publish device status.

Warning

Must call [st_cap_attr_free](#) to free IOT_EVENT data after using it.

See also

[st_cap_attr_send](#)

5.37.4.4 st_cap_attr_create_string_array() `IOT_EVENT* st_cap_attr_create_string_array (`

```
    char * attribute,
    uint8_t str_num,
    char * string_array[],
    char * unit )
```

Create IOT_EVENT data with string array value.

This function creates a new IOT_EVENT data with input parameters. Once it returns, user has full responsibility for deallocating event data by using [st_cap_attr_free](#).

Parameters

in	<i>attribute</i>	The attribute string of IOT_EVENT data.
in	<i>str_num</i>	The number of strings in the string_array.
in	<i>string_array</i>	The pointer of string array to add to IOT_EVENT data.
in	<i>unit</i>	The unit string if needed. Otherwise NULL.

Returns

Pointer of IOT_EVENT which is used to publish device status.

Warning

Must call [st_cap_attr_free](#) to free IOT_EVENT data after using it.

See also

[st_cap_attr_send](#)

5.37.4.5 st_cap_attr_free() void st_cap_attr_free (

```
IOT_EVENT * event )
```

Free IOT_EVENT data.

This function frees IOT_EVENT data.

Parameters

in	<i>event</i>	The IOT_EVENT data to free.
----	--------------	-----------------------------

5.37.4.6 st_cap_attr_send() int st_cap_attr_send (

```
IOT_CAP_HANDLE * cap_handle,
uint8_t evt_num,
IOT_EVENT * event[] )
```

Request to publish deviceEvent.

This function creates a deviceEvent with the list of IOT_EVENT data, and requests to publish it. When there is no error, this function returns sequence number, which is unique value to identify the deviceEvent message.

Parameters

in	<i>cap_handle</i>	The IOT_CAP_HANDLE to publish a deviceEvent.
in	<i>evt_num</i>	The number of IOT_EVENT data in the event.
in	<i>event</i>	The IOT_EVENT data list to create the deviceEvent.

Returns

return sequence number(which is positive integer) if successful, negative integer for error case.

5.37.4.7 st_cap_cmd_set_cb() int st_cap_cmd_set_cb (

```
IOT_CAP_HANDLE * cap_handle,
const char * cmd_type,
st_cap_cmd_cb cmd_cb,
void * usr_data )
```

Register callback function for command message.

This function registers user callback for command message from ST server. If the capability(used to create handle) and cmd_type of command message are same with input arguments, the callback function will be called.

Parameters

in	<i>cap_handle</i>	The capability handle to register cb function.
in	<i>cmd_type</i>	The commands interested to process.
in	<i>cmd_cb</i>	The callback function invoked when command is received.
in	<i>usr_data</i>	User data for cmd_cb.

Returns

`return (0)` if it works successfully, non-zero for error case.

Warning

User callback must return because MQTT works after user callback has ended.

5.37.4.8 st_cap_handle_init() `IOT_CAP_HANDLE* st_cap_handle_init (`

```
    IOT_CTX * iot_ctx,
    const char * component,
    const char * capability,
    st_cap_init_cb init_cb,
    void * init_usr_data )
```

Create and initialize a capability handle.

This function creates a capability handle, and initializes it with input args.

Parameters

in	<i>iot_ctx</i>	The iot context.
in	<i>component</i>	Component string. Default component name is "main".
in	<i>capability</i>	Capability string. This should be matched with "id" value of capability definition json format.
in	<i>init_cb</i>	The function which is called to initialize device state.
in	<i>init_usr_data</i>	User data for init_cb.

Returns

Pointer of created capability handle.

5.37.4.9 st_conn_cleanup() `int st_conn_cleanup (`

```
    IOT_CTX * iot_ctx,
    bool reboot )
```

st-iot-core device clean-up function

This function cleans-up all DATA including provisioning & registered data

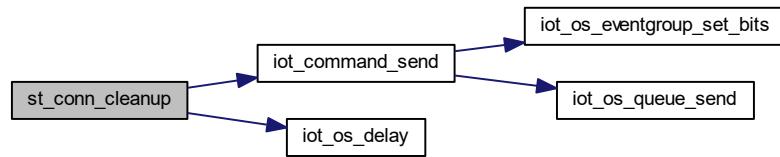
Parameters

in	<i>iot_ctx</i>	<code>iot_context</code> handle generated by <code>iot_main_init()</code>
in	<i>reboot</i>	boolean set true for auto-reboot of system, else false.

Returns

return (0) if it works successfully, non-zero for error case.

Here is the call graph for this function:



5.37.4.10 `st_conn_init()` `IOT_CTX* st_conn_init (`
 `unsigned char * onboardings_config,`
 `unsigned int onboardings_config_len,`
 `unsigned char * device_info,`
 `unsigned int device_info_len)`

st-iot-core initialize function

This function initializes st-iot-core for target

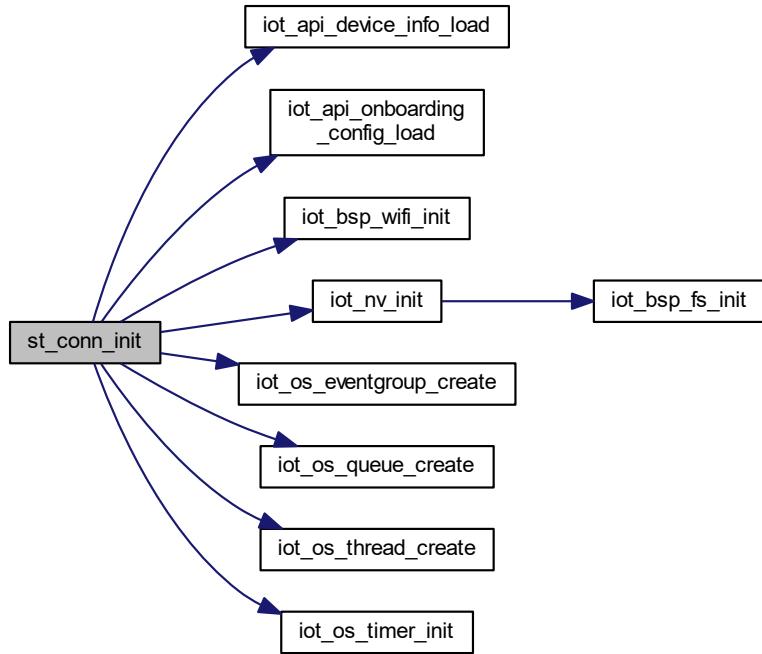
Parameters

in	<code>onboarding_config</code>	starting pointer of <code>onboarding_config.json</code> contents
in	<code>onboarding_config_len</code>	size of <code>onboarding_config.json</code> contents
in	<code>device_info</code>	starting pointer of <code>device_info.json</code> contents
in	<code>device_info_len</code>	size of <code>device_info.json</code> contents

Returns

return IOT_CTX handle(a pointer) if it succeeds, or NULL if it fails

Here is the call graph for this function:



```
5.37.4.11 st_conn_ownership_confirm() void st_conn_ownership_confirm (
    IOT_CTX * iot_ctx,
    bool confirm )
```

easysetup user confirm report function

This function reports the user confirmation to easysetup

Parameters

in	<code>iot_ctx</code>	<code>iot_context</code> handle generated by <code>iot_main_init()</code>
in	<code>confirm</code>	user confirmation result

Here is the call graph for this function:



5.37.4.12 st_conn_set_noti_cb() int st_conn_set_noti_cb (

```
    IOT_CTX * iot_ctx,
    st_cap_noti_cb noti_cb,
    void * noti_usr_data )
```

Register callback function for notification event.

This function registers user callback function which will be called when notification event occurs(such as rate limit, delete device).

Parameters

in	<i>iot_ctx</i>	The iot context.
in	<i>noti_cb</i>	The callback function which will be called when notification event occurs.
in	<i>noti_usr_data</i>	User data for noti_cb.

Returns

return (0) if it works successfully, non-zero for error case.

Warning

User callback must return because MQTT works after user callback has ended

5.37.4.13 st_conn_start() int st_conn_start (

```
    IOT_CTX * iot_ctx,
    st_status_cb status_cb,
    iot_status_t maps,
    void * usr_data,
    iot_pin_t * pin_num )
```

st-iot-core server connection function

This function tries to connect server

Parameters

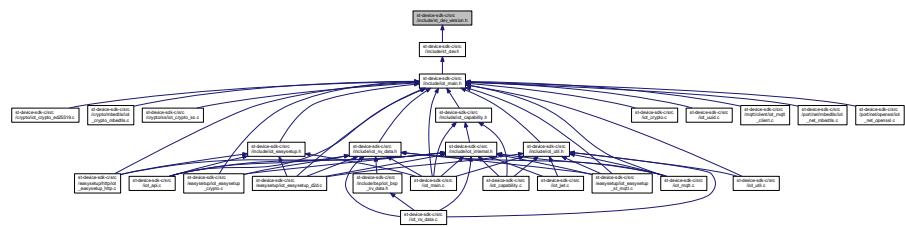
in	<i>iot_ctx</i>	iot_context handle generated by iot_main_init()
in	<i>status_cb</i>	user callback function to receive status of st-iot-core
in	<i>maps</i>	status of st-iot-core interested to receive through <i>status_cb</i>
in	<i>usr_data</i>	user data(a pointer) to use in <i>status_cb</i>
in	<i>pin_num</i>	if PIN ownership validation type used, valid 8 digit pin should be set. otherwise set null.

Returns

`return (0)` if it works successfully, non-zero for error case.

5.38 st-device-sdk-c/src/include/st_dev_version.h File Reference

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

**Macros**

- `#define VER_MAJOR (1)`
- `#define VER_MINOR (0)`
- `#define VER_PATCH (10)`
- `#define STDK_VERSION(a, b, c) (((a) << 16) + ((b) << 8) + (c))`
- `#define STDK_VERSION_CODE (STDK_VERSION(VER_MAJOR,VER_MINOR,VER_PATCH))`

5.38.1 Macro Definition Documentation

5.38.1.1 STDK_VERSION `#define STDK_VERSION(
 a,
 b,
 c) (((a) << 16) + ((b) << 8) + (c))`

5.38.1.2 STDK_VERSION_CODE `#define STDK_VERSION_CODE (STDK_VERSION(VER_MAJOR,VER_MINOR,VER_PATCH))`

5.38.1.3 VER_MAJOR #define VER_MAJOR (1)

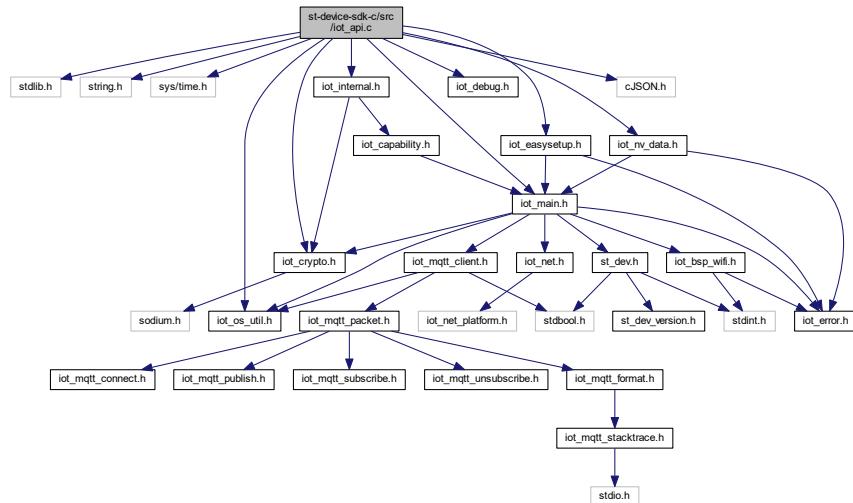
5.38.1.4 VER_MINOR #define VER_MINOR (0)

5.38.1.5 VER_PATCH #define VER_PATCH (10)

5.39 st-device-sdk-c/src/iot_api.c File Reference

```
#include <stdlib.h>
#include <string.h>
#include <sys/time.h>
#include "iot_main.h"
#include "iot_internal.h"
#include "iot_debug.h"
#include "iot_easysetup.h"
#include "iot_crypto.h"
#include "iot_nv_data.h"
#include "iot_os_util.h"
#include "cJSON.h"
```

Include dependency graph for iot_api.c:



Functions

- `iot_error_t iot_command_send (struct iot_context *ctx, enum iot_command_type new_cmd, const void *param, int param_size)`
send command to iot main task
- `iot_error_t iot_easysetup_request (struct iot_context *ctx, enum iot_easysetup_step step, const void *payload)`

- `iot_error_t iot_state_update (struct iot_context *ctx, iot_state_t new_state, int opt)`
update iot state
- `void iot_api_onboarding_config_mem_free (struct iot_devconf_prov_data *devconf)`
free onboarding config memory
- `iot_error_t iot_api_onboarding_config_load (unsigned char *onboarding_config, unsigned int onboarding_config_len, struct iot_devconf_prov_data *devconf)`
load "onboarding_config.json" from application source directory
- `iot_error_t iot_get_time_in_sec (char *buf, size_t buf_len)`
get time data by sec
- `iot_error_t iot_get_time_in_ms (char *buf, size_t buf_len)`
get time data in msec
- `void iot_api_device_info_mem_free (struct iot_device_info *device_info)`
free device info memory
- `iot_error_t iot_api_device_info_load (unsigned char *device_info, unsigned int device_info_len, struct iot_device_info *info)`
load "device_info.json" from application source directory
- `void iot_api_prov_data_mem_free (struct iot_device_prov_data *prov)`
free prov data memory
- `iot_error_t iot_api_read_device_identity (unsigned char *nv_prof, unsigned int nv_prof_len, const char *object, char **nv_data)`
Extract required data from "device_info.json" which is located in application source directory.
- `iot_error_t iot_device_cleanup (struct iot_context *ctx)`
device cleanup

5.39.1 Function Documentation

5.39.1.1 iot_api_device_info_load() `iot_error_t iot_api_device_info_load (`
`unsigned char * device_info,`
`unsigned int device_info_len,`
`struct iot_device_info * info)`

load "device_info.json" from application source directory

"device_info.json" should be updated by application developer

This function parses downloaded "device_info.json" to be used for EasySetup

Only firmwareVersion will be parsed by this api. others are handled by another api

Parameters

in	<code>device_info</code>	start pointer of json data
in	<code>device_info_len</code>	json data length
out	<code>info</code>	"device_info.json" will be parsed and mapped to this internal structure

Return values

<code>IOT_ERROR_NONE</code>	success.
<code>IOT_ERROR_UNINITIALIZED</code>	invalid json value.

Return values

<i>IOT_ERROR_MEM_ALLOC</i>	memory allocation failure.
----------------------------	----------------------------

example

```
{
    "deviceInfo": {
        "firmwareVersion": "FwVer0011A",
        ...
    }
}
```

Here is the caller graph for this function:



5.39.1.2 *iot_api_device_info_mem_free()* *void iot_api_device_info_mem_free (struct iot_device_info * info)*

free device info memory

this function frees the loaded device's information

Parameters

<i>in</i>	<i>info</i>	loaded device's information
-----------	-------------	-----------------------------

5.39.1.3 *iot_api_onboarding_config_load()* *iot_error_t iot_api_onboarding_config_load (unsigned char * onboarding_config, unsigned int onboarding_config_len, struct iot_devconf_prov_data * devconf)*

load "onboarding_config.json" from application source directory

"onboarding_config.json" can be downloaded from SmartThings Developer Workspace
This function parses downloaded "onboarding_config.json" to be used for EasySetup

Parameters

<i>in</i>	<i>onboarding_config</i>	start pointer of json data
<i>in</i>	<i>onboarding_config_len</i>	json data length
<i>out</i>	<i>devconf</i>	"onboarding_config.json" will be parsed and mapped to this internal structure

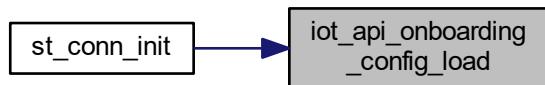
Return values

<i>IOT_ERROR_NONE</i>	success.
<i>IOT_ERROR_UNINITIALIZED</i>	invalid json value.
<i>IOT_ERROR_MEM_ALLOC</i>	memory allocation failure.
<i>IOT_ERROR_CRYPTO_SHA256</i>	sha256 error.
<i>IOT_ERROR_CRYPTO_BASE64</i>	base64 error.
<i>IOT_ERROR_CRYPTO_BASE64_URLSAFE</i>	base64 urlsafe error.

example

```
{
    "onboardingConfig": {
        "deviceOnboardingID": "NAME", // max. 13 character. this will be prefix of soft-ap ssid.
        "mnId": "MNID", // mnId for developer and/or manufacturer. "MNID" shouldn't be used.
        "setupId": "999", // 3-digit Device onboarding ID for this device.
        "vid": "VID", // VID(Vendor ID) for this profile.
        "deviceTypeID": "TYPE", // Device type which is selected from Developer Workspace.
        "ownershipValidationType": [ "JUSTWORKS", "BUTTON", "PIN", "QR" ],
        // "JUSTWORKS" for confirming without user interaction.
        // "BUTTON" for confirming by pressing builtin button.
        // "PIN" for confirming by matching 8-digit number PIN
        // "QR" for confirming by scanning a QR code by SmartThings app.
        "identityType": "ED25519 or CERTIFICATE" // ED25519 or X.509 CERTIFICATE
    }
}
```

Here is the caller graph for this function:



5.39.1.4 *iot_api_onboarding_config_mem_free()* void iot_api_onboarding_config_mem_free (struct *iot_devconf_prov_data* * *devconf*)

free onboarding config memory

this function frees the loaded onboarding configuration

Parameters

in	<i>devconf</i>	loaded onboarding configuration
----	----------------	---------------------------------

5.39.1.5 *iot_api_prov_data_mem_free()* void iot_api_prov_data_mem_free (

```
    struct iot_device_prov_data * prov )
```

free prov data memory

this function frees the loaded provisioning data

Parameters

in	<i>prov</i>	loaded provisioning data
----	-------------	--------------------------

Here is the caller graph for this function:



5.39.1.6 iot_api_read_device_identity() `iot_error_t iot_api_read_device_identity (`
 `unsigned char * device_nv_info,`
 `unsigned int device_nv_info_len,`
 `const char * object,`
 `char ** nv_data)`

Extract required data from "device_info.json" which is located in application source directory.

"device_info.json" should be updated by application developer

This function parses downloaded "device_info.json" to be used for EasySetup

Parameters

in	<i>device_nv_info</i>	starting pointer of json data
in	<i>device_nv_info_len</i>	json data length
in	<i>object</i>	object name for searching json data.
out	<i>nv_data</i>	"device_info.json" will be parsed by "object" and mapped to this pointer

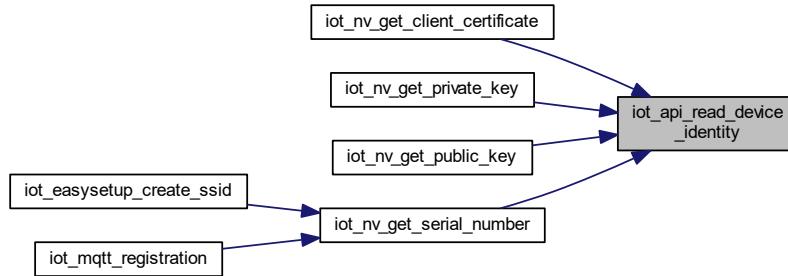
Return values

<i>IOT_ERROR_NONE</i>	success.
<i>IOT_ERROR_UNINITIALIZED</i>	invalid json value.
<i>IOT_ERROR_MEM_ALLOC</i>	memory allocation failure.

example

```
{
  "nvProfile": {
    "privateKey": "privateKey", // Client (= Device) Private key
    "publicKey": "publicKey", // Client (= Device) Public key
    "serialNumber": "serialNumber" // Device Serial Number
  }
}
```

Here is the caller graph for this function:



5.39.1.7 iot_command_send() iot_error_t iot_command_send (

```

    struct iot_context * ctx,
    enum iot_command_type cmd_type,
    const void * param,
    int param_size )
```

send command to iot main task

this function sends specific command to iot-task via queue

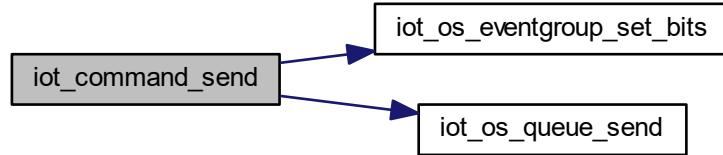
Parameters

in	<i>ctx</i>	iot-core context
in	<i>cmd_type</i>	actual specific command type
in	<i>param</i>	additional parameter data for each command
in	<i>param_size</i>	additional parameter size

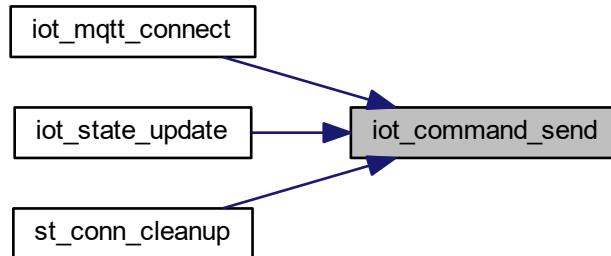
Return values

<i>IOT_ERROR_NONE</i>	success.
<i>IOT_ERROR_MEM_ALLOC</i>	memory allocation failed
<i>IOT_ERROR_BAD_REQ</i>	queue send error

Here is the call graph for this function:



Here is the caller graph for this function:



5.39.1.8 iot_device_cleanup() `iot_error_t iot_device_cleanup (struct iot_context * ctx)`

device cleanup

this function triggers clean-up process. All registered data will be removed

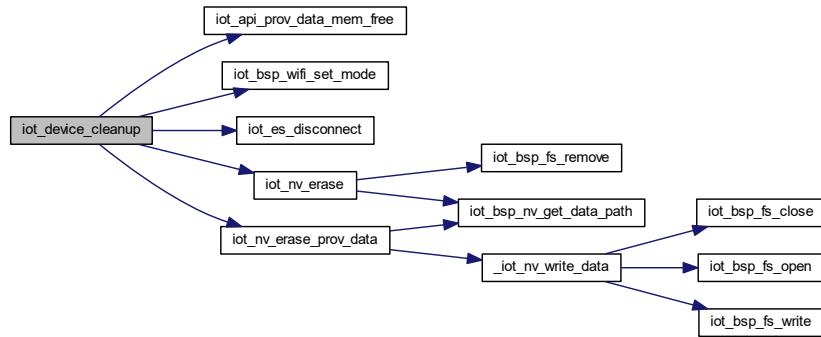
Parameters

in	<code>ctx</code>	iot-core context
----	------------------	------------------

Return values

<code>IOT_ERROR_NONE</code>	success.
-----------------------------	----------

Here is the call graph for this function:



5.39.1.9 `iot_easysetup_request()`

```

iot_error_t iot_easysetup_request (
    struct iot_context * ctx,
    enum iot_easysetup_step step,
    const void * payload )
  
```

send easysetup cgi payload manipulation request

easysetup cgi payload manipulation should be done at iot-task. This function sends payload to iot-task via queue

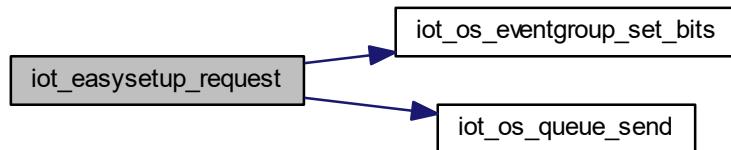
Parameters

in	<code>ctx</code>	iot-core context
in	<code>step</code>	indicates which uri(command) is dealing with
in	<code>payload</code>	payload data - mostly json data

Return values

<code>IOT_ERROR_NONE</code>	success.
<code>IOT_ERROR_BAD_REQ</code>	queue send error

Here is the call graph for this function:



```
5.39.1.10 iot_get_time_in_ms() iot_error_t iot_get_time_in_ms (
    char * buf,
    size_t buf_len )
```

get time data in msec

this function tries to get time value in millisecond by string

Parameters

in	<i>buf</i>	buffer point to contain millisecond based string value
in	<i>buf_len</i>	size of allocated buffer for string

Return values

<i>IOT_ERROR_NONE</i>	success.
-----------------------	----------

```
5.39.1.11 iot_get_time_in_sec() iot_error_t iot_get_time_in_sec (
    char * buf,
    size_t buf_len )
```

get time data by sec

this function tries to get time value in second by string

Parameters

in	<i>buf</i>	buffer point to contain second based string value
in	<i>buf_len</i>	size of allocated buffer for string

Return values

<i>IOT_ERROR_NONE</i>	success.
-----------------------	----------

```
5.39.1.12 iot_state_update() iot_error_t iot_state_update (
    struct iot_context * ctx,
    iot_state_t new_state,
    int need_interact )
```

update iot state

this function tries to update iot-state using iot_command_send internally

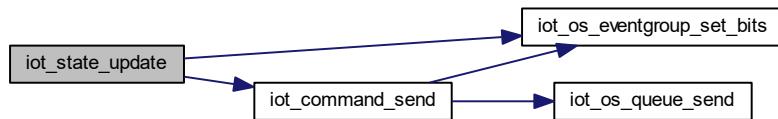
Parameters

in	<i>ctx</i>	iot-core context
in	<i>new_state</i>	new iot-state to update
in	<i>need_interact</i>	additional parameter data for each command

Return values

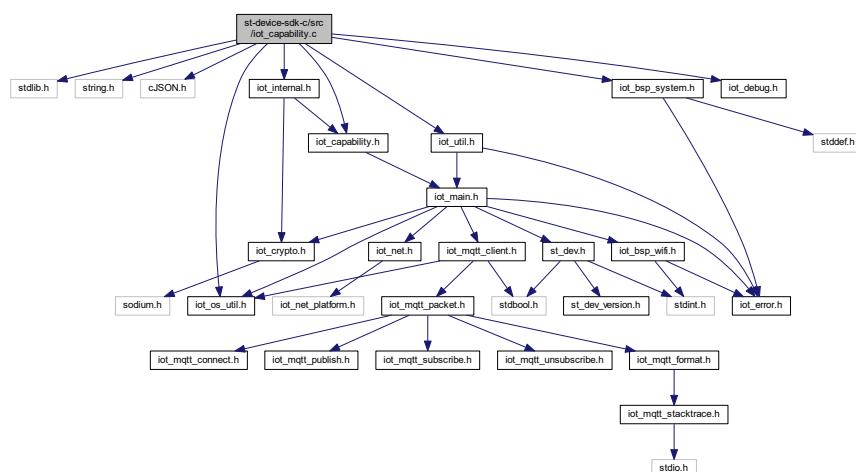
<i>IOT_ERROR_NONE</i>	success.
<i>IOT_ERROR_MEM_ALLOC</i>	memory allocation failed
<i>IOT_ERROR_BAD_REQ</i>	queue send error

Here is the call graph for this function:



5.40 st-device-sdk-c/src/iot_capability.c File Reference

```
#include <stdlib.h>
#include <string.h>
#include <cJSON.h>
#include "iot_internal.h"
#include "iot_util.h"
#include "iot_debug.h"
#include "iot_capability.h"
#include "iot_os_util.h"
#include "iot_bsp_system.h"
Include dependency graph for iot_capability.c:
```



Macros

- `#define MAX_SQNUM 0x7FFFFFFF`

Functions

- `IOT_EVENT * st_cap_attr_create_int (char *attribute, int integer, char *unit)`
Create IOT_EVENT data with integer value.
- `IOT_EVENT * st_cap_attr_create_number (char *attribute, double number, char *unit)`
Create IOT_EVENT data with real number(double) value.
- `IOT_EVENT * st_cap_attr_create_string (char *attribute, char *string, char *unit)`
Create IOT_EVENT data with string value.
- `IOT_EVENT * st_cap_attr_create_string_array (char *attribute, uint8_t str_num, char *string_array[], char *unit)`
Create IOT_EVENT data with string array value.
- `void st_cap_attr_free (IOT_EVENT *event)`
Free IOT_EVENT data.
- `IOT_CAP_HANDLE * st_cap_handle_init (IOT_CTX *iot_ctx, const char *component, const char *capability, st_cap_init_cb init_cb, void *init_usr_data)`
Create and initialize a capability handle.
- `int st_conn_set_noti_cb (IOT_CTX *iot_ctx, st_cap_noti_cb noti_cb, void *noti_usr_data)`
Register callback function for notification event.
- `int st_cap_cmd_set_cb (IOT_CAP_HANDLE *cap_handle, const char *cmd_type, st_cap_cmd_cb cmd_cb, void *usr_data)`
Register callback function for command message.
- `int st_cap_attr_send (IOT_CAP_HANDLE *cap_handle, uint8_t evt_num, IOT_EVENT *event[])`
Request to publish deviceEvent.
- `void iot_noti_sub_cb (struct iot_context *ctx, char *payload)`
callback for mqtt noti msg
- `void iot_cap_sub_cb (iot_cap_handle_list_t *cap_handle_list, char *payload)`
callback for mqtt command msg
- `void iot_cap_call_init_cb (iot_cap_handle_list_t *cap_handle_list)`
call init callback

5.40.1 Macro Definition Documentation

5.40.1.1 MAX_SQNUM `#define MAX_SQNUM 0x7FFFFFFF`

5.40.2 Function Documentation

5.40.2.1 iot_cap_call_init_cb() `void iot_cap_call_init_cb (` `iot_cap_handle_list_t * cap_handle_list)`

call init callback

this function is used to call all allocated capability callbacks when target is connected

Parameters

in	<i>cap_handle_list</i>	allocated capability handle list
----	------------------------	----------------------------------

```
5.40.2.2 iot_cap_sub_cb() void iot_cap_sub_cb (
    iot_cap_handle_list_t * cap_handle_list,
    char * payload )
```

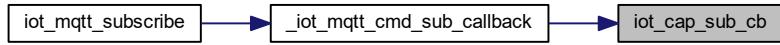
callback for mqtt command msg

this function is used to handle command message from server

Parameters

in	<i>cap_handle_list</i>	allocated capability handle list
in	<i>payload</i>	received raw message from server

Here is the caller graph for this function:



```
5.40.2.3 iot_noti_sub_cb() void iot_noti_sub_cb (
    struct iot_context * ctx,
    char * payload )
```

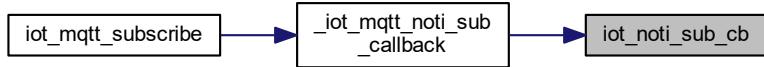
callback for mqtt noti msg

this function is used to handle notification message from server

Parameters

in	<i>ctx</i>	iot-core context
in	<i>payload</i>	received raw message from server

Here is the caller graph for this function:



5.40.2.4 `st_cap_attr_create_int()`

```
IOT_EVENT* st_cap_attr_create_int (
    char * attribute,
    int integer,
    char * unit )
```

Create IOT_EVENT data with integer value.

This function creates a new IOT_EVENT data with input parameters. Once it returns, user has full responsibility for deallocating event data by using [st_cap_attr_free](#).

Parameters

in	<i>attribute</i>	The attribute string of IOT_EVENT data.
in	<i>integer</i>	The integer to add to IOT_EVENT data.
in	<i>unit</i>	The unit string if needed. Otherwise NULL.

Returns

Pointer of IOT_EVENT which is used to publish device status.

Warning

Must call [st_cap_attr_free](#) to free IOT_EVENT data after using it.

See also

[st_cap_attr_send](#)

5.40.2.5 `st_cap_attr_create_number()`

```
IOT_EVENT* st_cap_attr_create_number (
    char * attribute,
    double number,
    char * unit )
```

Create IOT_EVENT data with real number(double) value.

This function creates a new IOT_EVENT data with input parameters. Once it returns, user has full responsibility for deallocating event data by using [st_cap_attr_free](#).

Parameters

in	<i>attribute</i>	The attribute string of IOT_EVENT data.
in	<i>number</i>	The double number to add to IOT_EVENT data.
in	<i>unit</i>	The unit string if needed. Otherwise NULL.

Returns

Pointer of IOT_EVENT which is used to publish device status.

Warning

Must call [st_cap_attr_free](#) to free IOT_EVENT data after using it.

See also

[st_cap_attr_send](#)

5.40.2.6 st_cap_attr_create_string() [`IOT_EVENT* st_cap_attr_create_string\(`](#)

```
    char * attribute,
    char * string,
    char * unit )
```

Create IOT_EVENT data with string value.

This function creates a new IOT_EVENT data with input parameters. Once it returns, user has full responsibility for deallocating event data by using [st_cap_attr_free](#).

Parameters

in	<i>attribute</i>	The attribute string of IOT_EVENT data.
in	<i>string</i>	The string to add to IOT_EVENT data.
in	<i>unit</i>	The unit string if needed. Otherwise NULL.

Returns

Pointer of IOT_EVENT which is used to publish device status.

Warning

Must call [st_cap_attr_free](#) to free IOT_EVENT data after using it.

See also

[st_cap_attr_send](#)

5.40.2.7 st_cap_attr_create_string_array() `IOT_EVENT* st_cap_attr_create_string_array (`

```
    char * attribute,
    uint8_t str_num,
    char * string_array[],
    char * unit )
```

Create IOT_EVENT data with string array value.

This function creates a new IOT_EVENT data with input parameters. Once it returns, user has full responsibility for deallocating event data by using [st_cap_attr_free](#).

Parameters

in	<i>attribute</i>	The attribute string of IOT_EVENT data.
in	<i>str_num</i>	The number of strings in the string_array.
in	<i>string_array</i>	The pointer of string array to add to IOT_EVENT data.
in	<i>unit</i>	The unit string if needed. Otherwise NULL.

Returns

Pointer of IOT_EVENT which is used to publish device status.

Warning

Must call [st_cap_attr_free](#) to free IOT_EVENT data after using it.

See also

[st_cap_attr_send](#)

5.40.2.8 st_cap_attr_free() `void st_cap_attr_free (`

```
    IOT_EVENT * event )
```

Free IOT_EVENT data.

This function frees IOT_EVENT data.

Parameters

in	<i>event</i>	The IOT_EVENT data to free.
----	--------------	-----------------------------

5.40.2.9 st_cap_attr_send() `int st_cap_attr_send (`

```
    IOT_CAP_HANDLE * cap_handle,
    uint8_t evt_num,
    IOT_EVENT * event[] )
```

Request to publish deviceEvent.

This function creates a deviceEvent with the list of IOT_EVENT data, and requests to publish it. When there is no error, this function returns sequence number, which is unique value to identify the deviceEvent message.

Parameters

in	<i>cap_handle</i>	The IOT_CAP_HANDLE to publish a deviceEvent.
in	<i>evt_num</i>	The number of IOT_EVENT data in the event.
in	<i>event</i>	The IOT_EVENT data list to create the deviceEvent.

Returns

return sequence number(which is positive integer) if successful, negative integer for error case.

```
5.40.2.10 st_cap_cmd_set_cb() int st_cap_cmd_set_cb (
    IOT_CAP_HANDLE * cap_handle,
    const char * cmd_type,
    st_cap_cmd_cb cmd_cb,
    void * usr_data )
```

Register callback function for command message.

This function registers user callback for command message from ST server. If the capability(used to create handle) and cmd_type of command message are same with input arguments, the callback function will be called.

Parameters

in	<i>cap_handle</i>	The capability handle to register cb function.
in	<i>cmd_type</i>	The commands interested to process.
in	<i>cmd_cb</i>	The callback function invoked when command is received.
in	<i>usr_data</i>	User data for cmd_cb.

Returns

return (0) if it works successfully, non-zero for error case.

Warning

User callback must return because MQTT works after user callback has ended.

```
5.40.2.11 st_cap_handle_init() IOT_CAP_HANDLE* st_cap_handle_init (
    IOT_CTX * iot_ctx,
    const char * component,
    const char * capability,
```

```
st_cap_init_cb init_cb,  
void * init_usr_data )
```

Create and initialize a capability handle.

This function creates a capability handle, and initializes it with input args.

Parameters

in	<i>iot_ctx</i>	The iot context.
in	<i>component</i>	Component string. Default component name is "main".
in	<i>capability</i>	Capability string. This should be matched with "id" value of capability definition json format.
in	<i>init_cb</i>	The function which is called to initialize device state.
in	<i>init_usr_data</i>	User data for init_cb.

Returns

Pointer of created capability handle.

5.40.2.12 st_conn_set_noti_cb() `int st_conn_set_noti_cb (`

```
    IOT_CTX * iot_ctx,
    st_cap_noti_cb noti_cb,
    void * noti_usr_data )
```

Register callback function for notification event.

This function registers user callback function which will be called when notification event occurs(such as rate limit, delete device).

Parameters

in	<i>iot_ctx</i>	The iot context.
in	<i>noti_cb</i>	The callback function which will be called when notification event occurs.
in	<i>noti_usr_data</i>	User data for noti_cb.

Returns

`return (0)` if it works successfully, non-zero for error case.

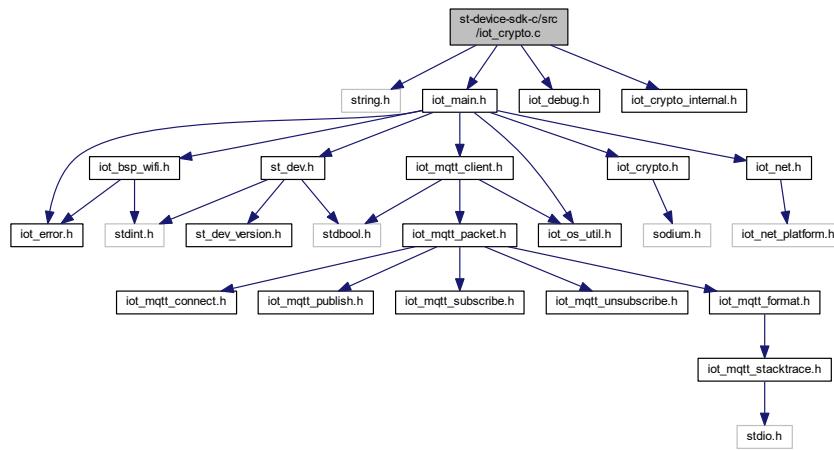
Warning

User callback must return because MQTT works after user callback has ended

5.41 st-device-sdk-c/src/iot_crypto.c File Reference

```
#include <string.h>
#include "iot_main.h"
#include "iot_debug.h"
```

```
#include "iot_crypto_internal.h"
Include dependency graph for iot_crypto.c:
```



Functions

- `iot_error_t iot_crypto_pk_init (iot_crypto_pk_context_t *ctx, iot_crypto_pk_info_t *info)`
Initialize a context by passed private key data.
- `void iot_crypto_pk_free (iot_crypto_pk_context_t *ctx)`
Cleanup the context.
- `iot_error_t iot_crypto_pk_sign (iot_crypto_pk_context_t *ctx, unsigned char *input, size_t ilen, unsigned char *sig, size_t *slen)`
Generate a signature.
- `iot_error_t iot_crypto_pk_verify (iot_crypto_pk_context_t *ctx, unsigned char *input, size_t ilen, unsigned char *sig, size_t *slen)`
Verify the signature.

5.41.1 Function Documentation

5.41.1.1 iot_crypto_pk_free() `void iot_crypto_pk_free (`
 `iot_crypto_pk_context_t * ctx)`

Cleanup the context.

Parameters

in	<code>ctx</code>	a pointer to a buffer to cleanup
----	------------------	----------------------------------

5.41.1.2 iot_crypto_pk_init() `iot_error_t iot_crypto_pk_init (`

```
    iot_crypto_pk_context_t * ctx,
    iot_crypto_pk_info_t * info )
```

Initialize a context by passed private key data.

Parameters

in	<i>ctx</i>	a pointer to a buffer to handle crypto context
in	<i>info</i>	a pointer to a buffer containing private key data

Return values

<i>IOT_ERROR_NONE</i>	context is sucessfully initialized
<i>IOT_ERROR_CRYPTO_PK_INVALID_CTX</i>	ctx is null
<i>IOT_ERROR_CRYPTO_PK_INVALID_ARG</i>	info is null
<i>IOT_ERROR_CRYPTO_PK_UNKNOWN_KEYTYPE</i>	private key data has not supported algorithm

5.41.1.3 **iot_crypto_pk_sign()** *iot_error_t* `iot_crypto_pk_sign(`

```
    iot_crypto_pk_context_t * ctx,
    unsigned char * input,
    size_t ilen,
    unsigned char * sig,
    size_t * slen )
```

Generate a signature.

Parameters

in	<i>ctx</i>	a pointer to a buffer containing crypto context
in	<i>input</i>	a pointer to a buffer to generate a signature
in	<i>ilen</i>	the size of buffer pointed by input in bytes
out	<i>sig</i>	a pointer to a buffer to store a signature
out	<i>slen</i>	the bytes written to sig

Return values

<i>IOT_ERROR_NONE</i>	the signature is sucessfully generated
<i>IOT_ERROR_CRYPTO_PK_PARSEKEY</i>	failed to parse device certificate
<i>IOT_ERROR_CRYPTO_PK_INVALID_CTX</i>	ctx is null
<i>IOT_ERROR_CRYPTO_PK_INVALID_ARG</i>	info is null
<i>IOT_ERROR_CRYPTO_PK_INVALID_KEYLEN</i>	a length of private key is wrong
<i>IOT_ERROR_CRYPTO_PK_SIGN</i>	failed to generate signature

5.41.1.4 **iot_crypto_pk_verify()** *iot_error_t* `iot_crypto_pk_verify(`

```
    iot_crypto_pk_context_t * ctx,
```

```
unsigned char * input,
size_t ilen,
unsigned char * sig,
size_t slen )
```

Verify the signature.

Parameters

in	<i>ctx</i>	a pointer to a buffer containing crypto context
in	<i>input</i>	a pointer to a buffer to generate a signature
in	<i>ilen</i>	the size of buffer pointed by input in bytes
in	<i>sig</i>	a pointer to a buffer containing the signature
in	<i>slen</i>	the size of buffer pointed by sig in bytes

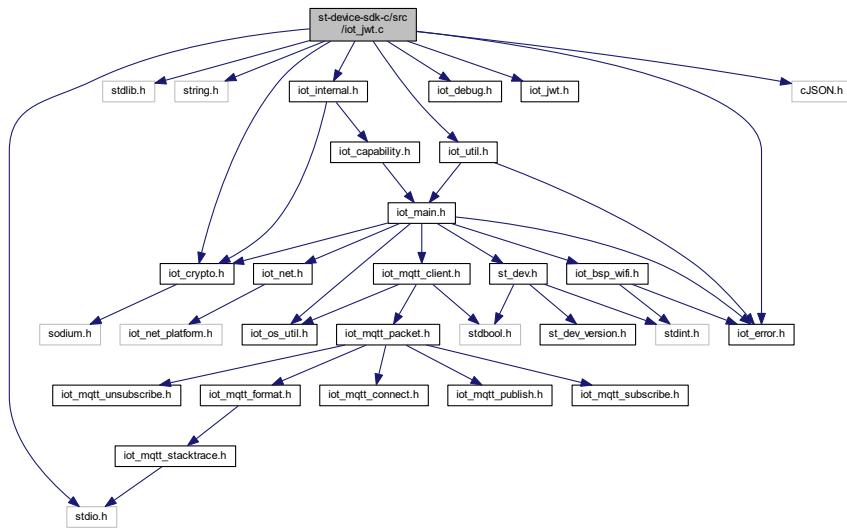
Return values

<i>IOT_ERROR_NONE</i>	the signature is sucessfully verified
<i>IOT_ERROR_CRYPTO_PK_NULL_FUNC</i>	verify function is not implemented
<i>IOT_ERROR_CRYPTO_PK_PARSEKEY</i>	failed to parse device certificate
<i>IOT_ERROR_CRYPTO_PK_INVALID_CTX</i>	ctx is null
<i>IOT_ERROR_CRYPTO_PK_INVALID_ARG</i>	info is null
<i>IOT_ERROR_CRYPTO_PK_INVALID_KEYLEN</i>	a length of private key is wrong
<i>IOT_ERROR_CRYPTO_PK_VERIFY</i>	the signature is mismatched

5.42 st-device-sdk-c/src/iot_jwt.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "iot_internal.h"
#include "iot_debug.h"
#include "iot_error.h"
#include "iot_crypto.h"
#include "iot_jwt.h"
#include "iot_util.h"
#include "cJSON.h"
```

Include dependency graph for iot_jwt.c:



Functions

- **`iot_error_t iot_jwt_create (char **token, const char *sn, iot_crypto_pk_info_t *pk_info)`**
Create a JWT as proof of the device's identity.

5.42.1 Function Documentation

5.42.1.1 `iot_jwt_create()`

```

iot_error_t iot_jwt_create (
    char ** token,
    const char * sn,
    iot_crypto_pk_info_t * pk_info )
```

Create a JWT as proof of the device's identity.

This function makes a JWT string to connect to ST Cloud. Pass the JWT as password. Supported key types are RS256 and ED25519.

Parameters

out	<code>token</code>	a pointer of buffer to store a formatted and signed string
in	<code>sn</code>	device serial number as user name
in	<code>pk_info</code>	private key data

Return values

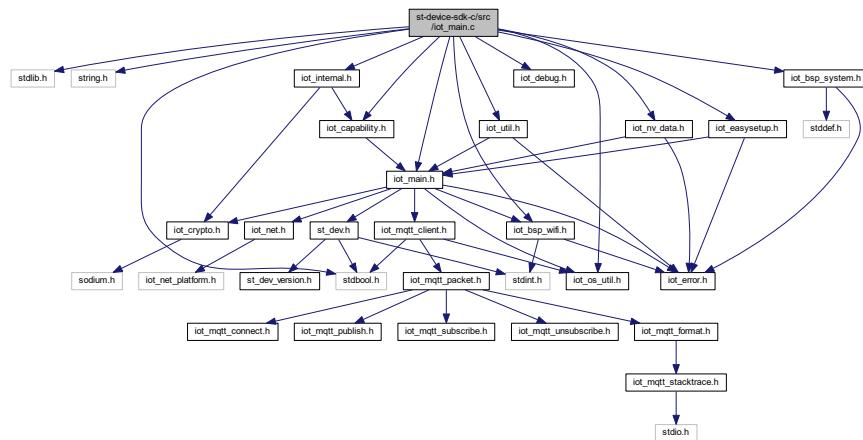
<code>IOT_ERROR_NONE</code>	JWT is sucessfully generated
<code>IOT_ERROR_JWT_MALLOC</code>	no more available heap memory

Return values

<code>IOT_ERROR_JWT_CJSON</code>	failed to make json
----------------------------------	---------------------

5.43 st-device-sdk-c/src/iot_main.c File Reference

```
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include "iot_main.h"
#include "iot_internal.h"
#include "iot_debug.h"
#include "iot_esp_wifi.h"
#include "iot_nv_data.h"
#include "iot_easysetup.h"
#include "iot_capability.h"
#include "iot_os_util.h"
#include "iot_util.h"
#include "iot_esp_system.h"
Include dependency graph for iot_main.c:
```



Macros

- `#define NEXT_STATE_TIMEOUT_MS (100000)`
- `#define EASYSETUP_TIMEOUT_MS (300000) /* 5 min */`

Functions

- `IOT_CTX * st_conn_init (unsigned char *onboarding_config, unsigned int onboarding_config_len, unsigned char *device_info, unsigned int device_info_len)`
st-iot-core initialize function
- `int st_conn_start (IOT_CTX *iot_ctx, st_status_cb status_cb, iot_status_t maps, void *usr_data, iot_pin_t *pin_num)`
st-iot-core server connection function
- `int st_conn_cleanup (IOT_CTX *iot_ctx, bool reboot)`
st-iot-core device clean-up function

5.43.1 Macro Definition Documentation

5.43.1.1 EASYSETUP_TIMEOUT_MS #define EASYSETUP_TIMEOUT_MS (300000) /* 5 min */

5.43.1.2 NEXT_STATE_TIMEOUT_MS #define NEXT_STATE_TIMEOUT_MS (100000)

5.43.2 Function Documentation

5.43.2.1 st_conn_cleanup() int st_conn_cleanup (
 IOT_CTX * iot_ctx,
 bool reboot)

st-iot-core device clean-up function

This function cleans-up all DATA including provisioning & registered data

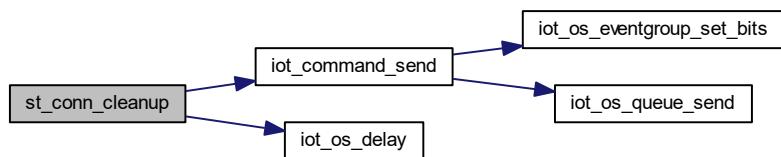
Parameters

in	<i>iot_ctx</i>	iot_context handle generated by iot_main_init()
in	<i>reboot</i>	boolean set true for auto-reboot of system, else false.

Returns

return (0) if it works successfully, non-zero for error case.

Here is the call graph for this function:



```
5.43.2.2 st_conn_init() IOT_CTX* st_conn_init (
    unsigned char * onboarding_config,
    unsigned int onboarding_config_len,
    unsigned char * device_info,
    unsigned int device_info_len )
```

st-iot-core initialize function

This function initializes st-iot-core for target

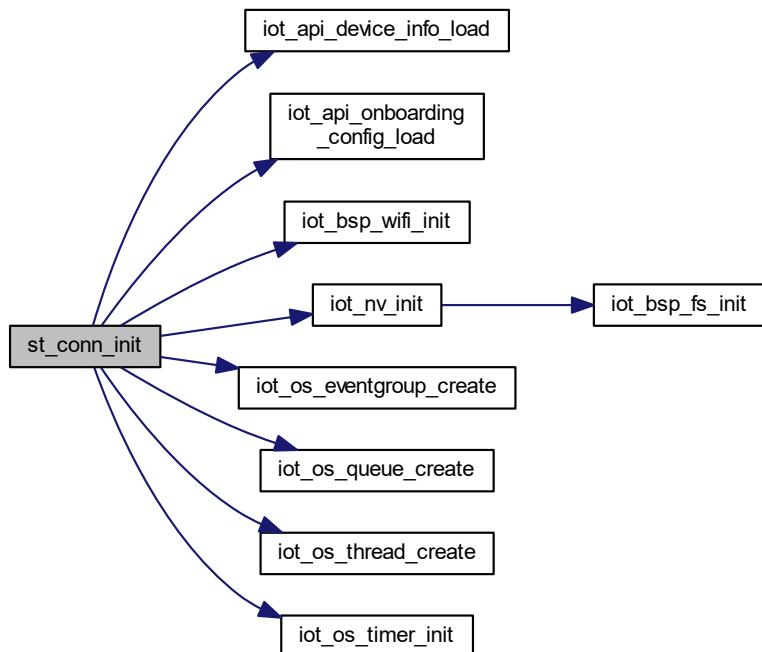
Parameters

in	<i>onboarding_config</i>	starting pointer of onboarding_config.json contents
in	<i>onboarding_config_len</i>	size of onboarding_config.json contents
in	<i>device_info</i>	starting pointer of device_info.json contents
in	<i>device_info_len</i>	size of device_info.json contents

Returns

return IOT_CTX handle(a pointer) if it succeeds, or NULL if it fails

Here is the call graph for this function:



```
5.43.2.3 st_conn_start() int st_conn_start (
    IOT_CTX * iot_ctx,
    st_status_cb status_cb,
    iot_status_t maps,
    void * usr_data,
    iot_pin_t * pin_num )
```

st-iot-core server connection function

This function tries to connect server

Parameters

in	<i>iot_ctx</i>	<code>iot_context</code> handle generated by <code>iot_main_init()</code>
in	<i>status_cb</i>	user callback function to receive status of st-iot-core
in	<i>maps</i>	status of st-iot-core interested to receive through <code>status_cb</code>
in	<i>usr_data</i>	user data(a pointer) to use in <code>status_cb</code>
in	<i>pin_num</i>	if PIN ownership validation type used, valid 8 digit pin should be set. otherwise set null.

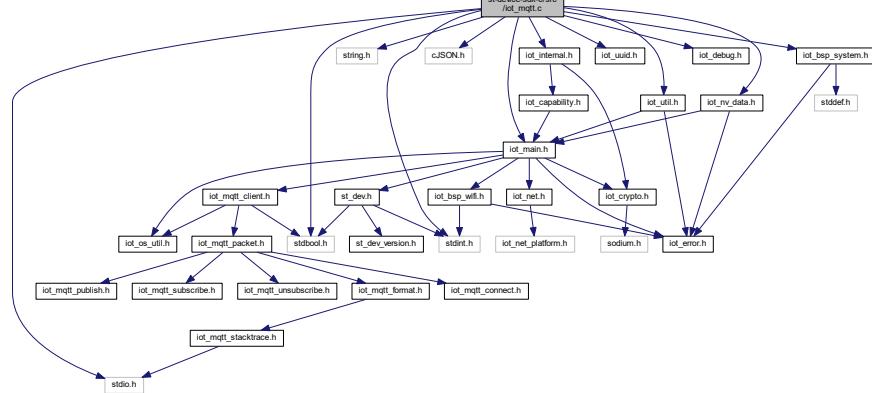
Returns

`return (0)` if it works successfully, non-zero for error case.

5.44 st-device-sdk-c/src/iot_mqtt.c File Reference

```
#include <stdint.h>
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#include <cJSON.h>
#include "iot_main.h"
#include "iot_internal.h"
#include "iot_uuid.h"
#include "iot_util.h"
#include "iot_debug.h"
#include "iot_nv_data.h"
#include "iot_bsp_system.h"
```

Include dependency graph for `iot_mqtt.c`:



Functions

- `iot_error_t iot_mqtt_connect (struct iot_mqtt_ctx *target_cli, char *username, char *sign_data)`
mqtt connect
- `iot_error_t iot_mqtt_publish (struct iot_context *ctx, void *payload)`
publish mqtt message
- `void _iot_mqtt_noti_sub_callback (MessageData *md, void *userData)`
- `void _iot_mqtt_cmd_sub_callback (MessageData *md, void *userData)`
- `iot_error_t iot_mqtt_subscribe (struct iot_mqtt_ctx *mqtt_ctx)`
subscribe mqtt topic
- `iot_error_t iot_mqtt_unsubscribe (struct iot_mqtt_ctx *mqtt_ctx)`
unsubscribe mqtt topic
- `void iot_mqtt_disconnect (struct iot_mqtt_ctx *target_cli)`
mqtt disconnect
- `iot_error_t iot_mqtt_registration (struct iot_mqtt_ctx *mqtt_ctx)`
register device to ST server

5.44.1 Function Documentation

5.44.1.1 `_iot_mqtt_cmd_sub_callback()` `void _iot_mqtt_cmd_sub_callback (`
`MessageData * md,`
`void * userData)`

Here is the call graph for this function:

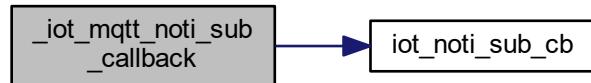


Here is the caller graph for this function:

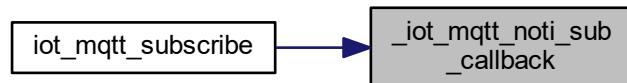


```
5.44.1.2 _iot_mqtt_noti_sub_callback() void _iot_mqtt_noti_sub_callback (
    MessageData * md,
    void * userData )
```

Here is the call graph for this function:



Here is the caller graph for this function:



```
5.44.1.3 iot_mqtt_connect() iot_error_t iot_mqtt_connect (
    struct iot_mqtt_ctx * target_cli,
    char * username,
    char * sign_data )
```

mqtt connect

this function connects ST server by mqtt

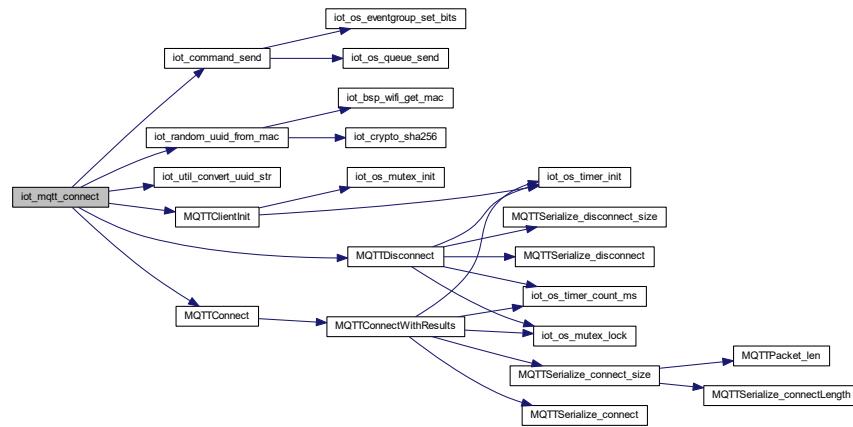
Parameters

in	<i>target_cli</i>	mqtt handling context
in	<i>username</i>	username to connect ST server based on mqtt protocol
in	<i>sign_data</i>	specific password that was jwt token-based to connect ST server

Return values

<i>IOT_ERROR_NONE</i>	success.
-----------------------	----------

Here is the call graph for this function:



5.44.1.4 iot_mqtt_disconnect() `void iot_mqtt_disconnect (struct iot_mqtt_ctx * target_cli)`

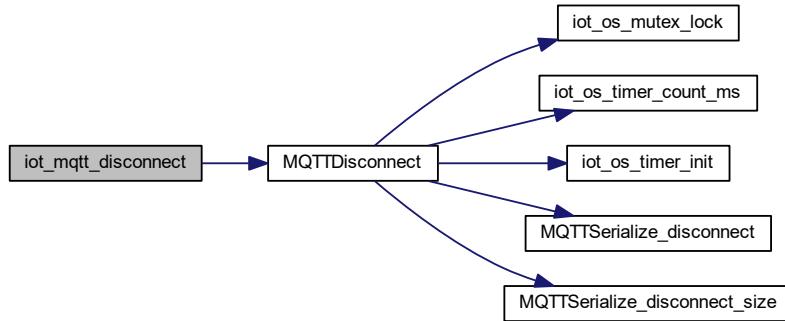
mqtt disconnect

this function is used to disconnect from server

Parameters

in	<i>target_cli</i>	mqtt handling context
----	-------------------	-----------------------

Here is the call graph for this function:



```
5.44.1.5 iot_mqtt_publish() iot_error_t iot_mqtt_publish (
    struct iot_context * ctx,
    void * payload )
```

publish mqtt message

this function is used to publish command & notification message

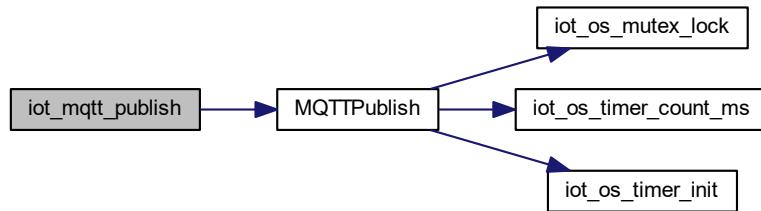
Parameters

in	<i>ctx</i>	iot-core context
in	<i>payload</i>	raw message sending to server

Return values

<i>IOT_ERROR_NONE</i>	success.
-----------------------	----------

Here is the call graph for this function:



```
5.44.1.6 iot_mqtt_registration() iot_error_t iot_mqtt_registration (
    struct iot_mqtt_ctx * mqtt_ctx )
```

register device to ST server

this function is used to handle the registration process

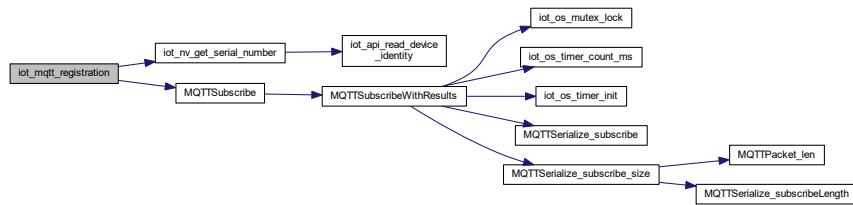
Parameters

in	<i>mqtt_ctx</i>	mqtt handling context
----	-----------------	-----------------------

Return values

<i>IOT_ERROR_NONE</i>	success.
-----------------------	----------

Here is the call graph for this function:



5.44.1.7 iot_mqtt_subscribe() *iot_error_t* `iot_mqtt_subscribe (struct iot_mqtt_ctx * mqtt_ctx)`

subscribe mqtt topic

this function is used to subscribe command & notification topics

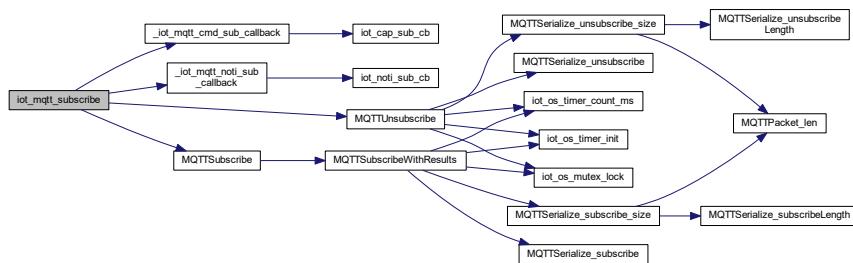
Parameters

in	<i>mqtt_ctx</i>	mqtt handling context
----	-----------------	-----------------------

Return values

<i>IOT_ERROR_NONE</i>	success.
-----------------------	----------

Here is the call graph for this function:



5.44.1.8 iot_mqtt_unsubscribe() *iot_error_t* `iot_mqtt_unsubscribe (struct iot_mqtt_ctx * mqtt_ctx)`

unsubscribe mqtt topic

this function is used to unsubscribe command & notification topics

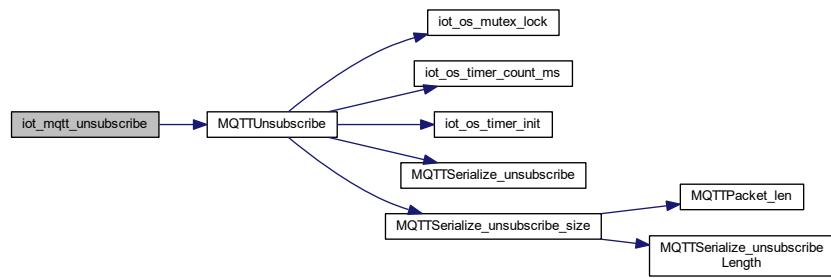
Parameters

in	<i>mqtt_ctx</i>	mqtt handling context
----	-----------------	-----------------------

Return values

<i>IOT_ERROR_NONE</i>	success.
-----------------------	----------

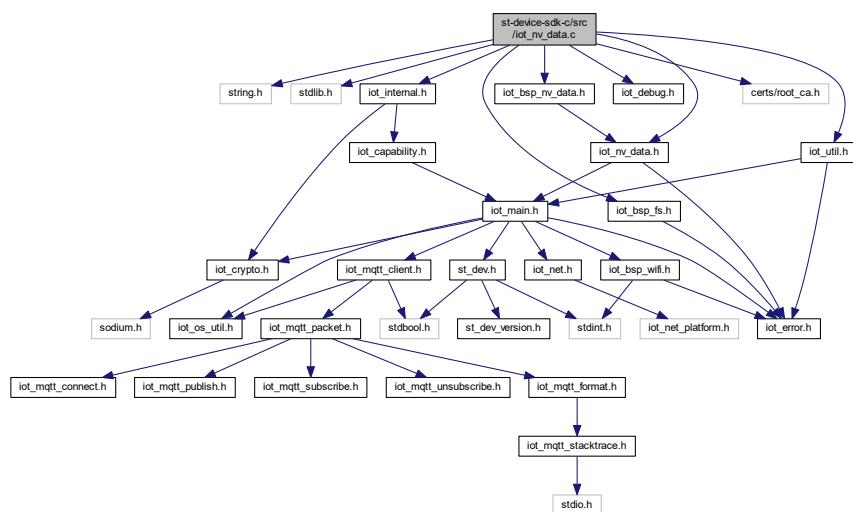
Here is the call graph for this function:



5.45 st-device-sdk-c/src/iot_nv_data.c File Reference

```

#include <string.h>
#include <stdlib.h>
#include "iot_nv_data.h"
#include "iot_bsp_fs.h"
#include "iot_bsp_nv_data.h"
#include "iot_debug.h"
#include "iot_util.h"
#include "certs/root_ca.h"
#include "iot_internal.h"
Include dependency graph for iot_nv_data.c:
  
```



Macros

- #define IOT_NVD_MAX_DATA_LEN (2048)
- #define IOT_NVD_MAX_ID_LEN (36)
- #define IOT_NVD_MAX_PW_LEN (64)
- #define IOT_NVD_MAX_BSSID_LEN (6)
- #define IOT_NVD_MAX_UID_LEN (128)

Functions

- iot_error_t iot_nv_read_data (const char *path, char *data, unsigned int size)
Get data from the nv file-system.
- iot_error_t iot_nv_write_data (const char *path, const char *data, unsigned int size)
Set data to the nv file-system.
- iot_error_t iot_nv_init (unsigned char *device_info, unsigned int device_info_len)
Initialize a nv file-system.
- iot_error_t iot_nv_deinit ()
Deinitialize a nv file-system.
- iot_error_t iot_nv_get_prov_data (struct iot_device_prov_data *prov_data)
Get provisioning data from the nv file-system.
- iot_error_t iot_nv_set_prov_data (struct iot_device_prov_data *prov_data)
Set provisioning data to the nv file-system.
- iot_error_t iot_nv_erase_prov_data ()
Erase wifi/cloud provisioning data.
- iot_error_t iot_nv_get_wifi_prov_data (struct iot_wifi_prov_data *wifi_prov)
Get wifi provisioning data from the nv file-system.
- iot_error_t iot_nv_set_wifi_prov_data (struct iot_wifi_prov_data *wifi_prov)
Set wifi provisioning data to the nv file-system.
- iot_error_t iot_nv_get_cloud_prov_data (struct iot_cloud_prov_data *cloud_prov)
Get cloud provisioning data from the nv file-system.
- iot_error_t iot_nv_set_cloud_prov_data (struct iot_cloud_prov_data *cloud_prov)
Set cloud provisioning data to the nv file-system.
- iot_error_t iot_nv_get_private_key (char **key, unsigned int *len)
Get a private key from the nv file-system.
- iot_error_t iot_nv_get_public_key (char **key, unsigned int *len)
Get a public key from the nv file-system.
- iot_error_t iot_nv_get_root_certificate (char **cert, unsigned int *len)
Get a root cert from the nv file-system.
- iot_error_t iot_nv_get_client_certificate (char **cert, unsigned int *len)
Get a client cert from the nv file-system.
- iot_error_t iot_nv_get_device_id (char **device_id, unsigned int *len)
Get a device id from the nv file-system.
- iot_error_t iot_nv_set_device_id (const char *device_id)
Set a device id to the nv file-system.
- iot_error_t iot_nv_get_serial_number (char **sn, unsigned int *len)
Get a serial number from the nv file-system.
- iot_error_t iot_nv_erase (iot_nvd_t nv_type)
Erase a nv data.

5.45.1 Macro Definition Documentation

5.45.1.1 IOT_NVD_MAX_BSSID_LEN #define IOT_NVD_MAX_BSSID_LEN (6)

5.45.1.2 IOT_NVD_MAX_DATA_LEN #define IOT_NVD_MAX_DATA_LEN (2048)

5.45.1.3 IOT_NVD_MAX_ID_LEN #define IOT_NVD_MAX_ID_LEN (36)

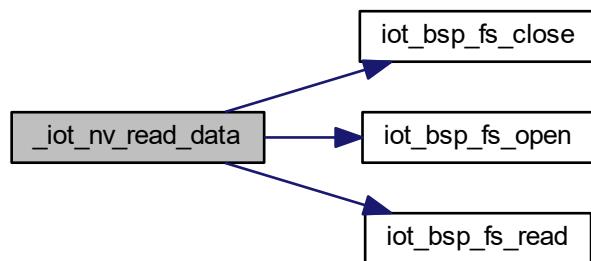
5.45.1.4 IOT_NVD_MAX_PW_LEN #define IOT_NVD_MAX_PW_LEN (64)

5.45.1.5 IOT_NVD_MAX_UID_LEN #define IOT_NVD_MAX_UID_LEN (128)

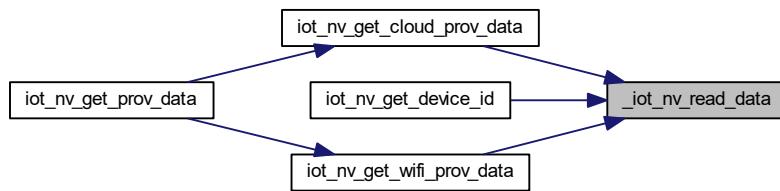
5.45.2 Function Documentation

5.45.2.1 _iot_nv_read_data() iot_error_t _iot_nv_read_data (const char * path, char * data, unsigned int size)

Here is the call graph for this function:

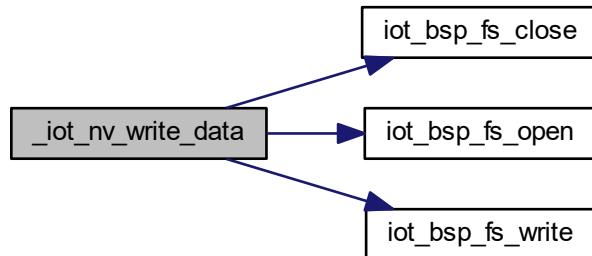


Here is the caller graph for this function:

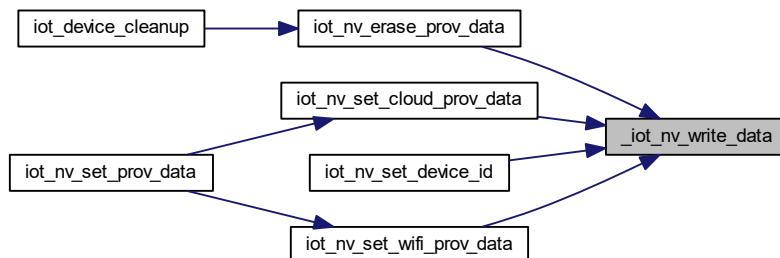


5.45.2.2 `_iot_nv_write_data()` `iot_error_t _iot_nv_write_data (`
`const char * path,`
`const char * data,`
`unsigned int size)`

Here is the call graph for this function:



Here is the caller graph for this function:



5.45.2.3 iot_nv_deinit() `iot_error_t iot_nv_deinit ()`

Deinitialize a nv file-system.

Return values

<code>IOT_ERROR_NONE</code>	NV file-system deinit successful.
<code>IOT_ERROR_DEINIT_FAIL</code>	NV file-system deinit failed.

See also

[iot_bsp_fs_deinit](#)

Here is the call graph for this function:



5.45.2.4 iot_nv_erase() `iot_error_t iot_nv_erase (` `iot_nvd_t nv_type)`

Erase a nv data.

This function erase the nv data completely in File-system.

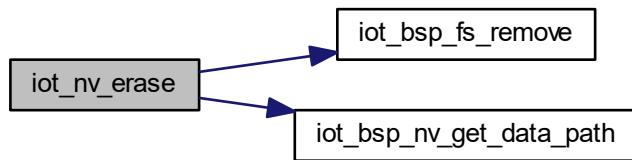
Parameters

in	<code>nv_type</code>	The type of nv data to erase.
----	----------------------	-------------------------------

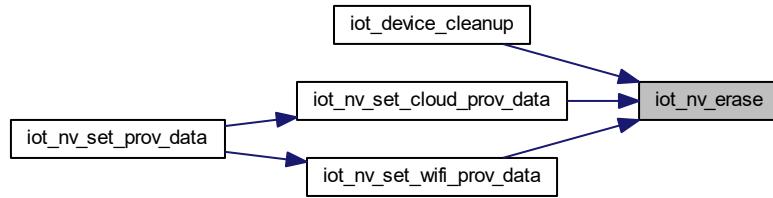
Return values

<code>IOT_ERROR_NONE</code>	NV data erase successful.
<code>IOT_ERROR_INVALID_ARGS</code>	Invalid nv type.
<code>IOT_ERROR_NV_DATA_ERROR</code>	NV data erase failed.
<code>IOT_ERROR_NV_DATA_NOT_EXIST</code>	NV data does not exist.

Here is the call graph for this function:



Here is the caller graph for this function:



5.45.2.5 `iot_nv_erase_prov_data()` `iot_error_t iot_nv_erase_prov_data ()`

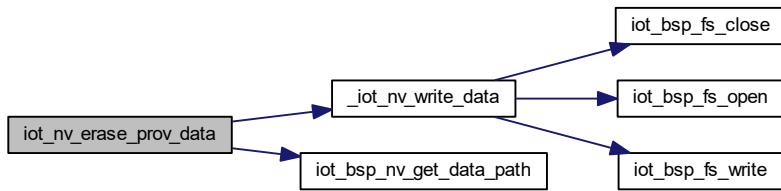
Erase wifi/cloud provisioning data.

This function erases the wifi/cloud provisioning data in File-system.

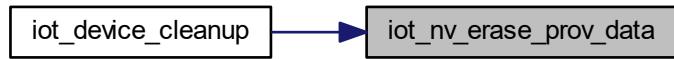
Return values

<code>IOT_ERROR_NONE</code>	NV data erase successful.
<code>IOT_ERROR_NV_DATA_ERROR</code>	NV data erase failed.

Here is the call graph for this function:



Here is the caller graph for this function:



5.45.2.6 `iot_nv_get_client_certificate()` `iot_error_t iot_nv_get_client_certificate (`
`char ** cert,`
`unsigned int * len)`

Get a client cert from the nv file-system.

Parameters

<code>out</code>	<code>cert</code>	A pointer to data array to store the client cert from the nv file-system.
<code>out</code>	<code>len</code>	The length of the nv data.

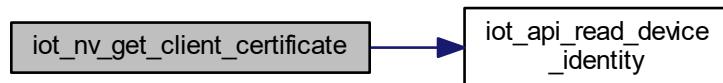
Return values

<code>IOT_ERROR_NONE</code>	Get nv data successful.
<code>IOT_ERROR_INVALID_ARGS</code>	Invalid argument.
<code>IOT_ERROR_NV_DATA_ERROR</code>	Get nv data failed.

Warning

The caller is always responsible to free the allocated pointer after using the data.

Here is the call graph for this function:



5.45.2.7 iot_nv_get_cloud_prov_data() `iot_error_t iot_nv_get_cloud_prov_data (struct iot_cloud_prov_data * cloud_prov)`

Get cloud provisioning data from the nv file-system.

Parameters

<code>out</code>	<code>cloud_prov</code>	A pointer to data structure to store the cloud provisioning data from the nv file-system.
------------------	-------------------------	---

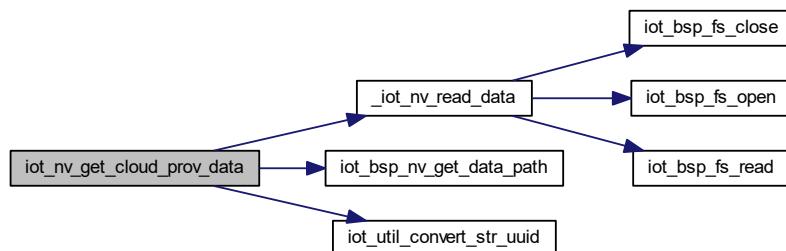
Return values

<code>IOT_ERROR_NONE</code>	Get nv data successful.
<code>IOT_ERROR_INVALID_ARGS</code>	Invalid argument.
<code>IOT_ERROR_NV_DATA_ERROR</code>	Get nv data failed.

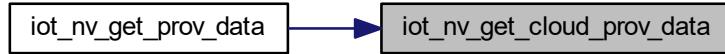
Warning

The caller is always responsible to free the allocated pointer after using the data.

Here is the call graph for this function:



Here is the caller graph for this function:



5.45.2.8 iot_nv_get_device_id() *iot_error_t* iot_nv_get_device_id (

```

    char ** device_id,
    unsigned int * len )

```

Get a device id from the nv file-system.

Parameters

<i>out</i>	<i>device_id</i>	A pointer to data array to store the device id from the nv file-system.
<i>out</i>	<i>len</i>	The length of the nv data.

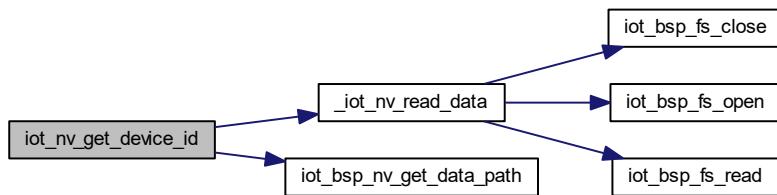
Return values

<i>IOT_ERROR_NONE</i>	Get nv data successful.
<i>IOT_ERROR_INVALID_ARGS</i>	Invalid argument.
<i>IOT_ERROR_NV_DATA_ERROR</i>	Get nv data failed.

Warning

The caller is always responsible to free the allocated pointer after using the data.

Here is the call graph for this function:



5.45.2.9 iot_nv_get_private_key() `iot_error_t iot_nv_get_private_key (`
 `char ** key,`
 `unsigned int * len)`

Get a private key from the nv file-system.

Parameters

<code>out</code>	<code>key</code>	A pointer to data array to store the private key from the nv file-system.
<code>out</code>	<code>len</code>	The length of the nv data.

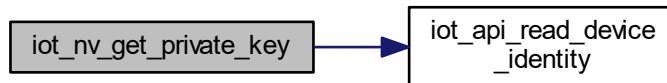
Return values

<code>IOT_ERROR_NONE</code>	Get nv data successful.
<code>IOT_ERROR_INVALID_ARGS</code>	Invalid argument.
<code>IOT_ERROR_NV_DATA_ERROR</code>	Get nv data failed.

Warning

The caller is always responsible to free the allocated pointer after using the data.

Here is the call graph for this function:



5.45.2.10 iot_nv_get_prov_data() `iot_error_t iot_nv_get_prov_data (`
 `struct iot_device_prov_data * prov_data)`

Get provisioning data from the nv file-system.

Parameters

<code>out</code>	<code>prov_data</code>	A pointer to data structure to store the provisioning data from the nv file-system.
------------------	------------------------	---

Return values

<code>IOT_ERROR_NONE</code>	Get nv data successful.
<code>IOT_ERROR_INVALID_ARGS</code>	Invalid argument.
<code>IOT_ERROR_NV_DATA_ERROR</code>	Get nv data failed.

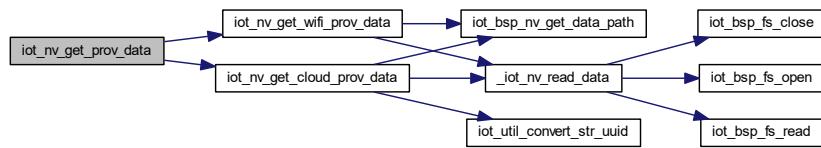
Warning

The caller is always responsible to free the allocated pointer after using the data.

See also

[iot_nv_get_wifi_prov_data](#)
[iot_nv_get_cloud_prov_data](#)

Here is the call graph for this function:



5.45.2.11 iot_nv_get_public_key() `iot_error_t iot_nv_get_public_key (`
 `char ** key,`
 `unsigned int * len)`

Get a public key from the nv file-system.

Parameters

out	<code>key</code>	A pointer to data array to store the public key from the nv file-system.
out	<code>len</code>	The length of the nv data.

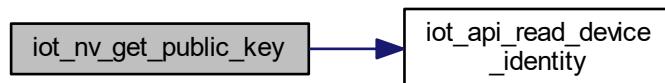
Return values

<code>IOT_ERROR_NONE</code>	Get nv data successful.
<code>IOT_ERROR_INVALID_ARGS</code>	Invalid argument.
<code>IOT_ERROR_NV_DATA_ERROR</code>	Get nv data failed.

Warning

The caller is always responsible to free the allocated pointer after using the data.

Here is the call graph for this function:



5.45.2.12 iot_nv_get_root_certificate() `iot_error_t iot_nv_get_root_certificate (`
`char ** cert,`
`unsigned int * len)`

Get a root cert from the nv file-system.

Parameters

<code>out</code>	<code>cert</code>	A pointer to data array to store the root cert from the nv file-system.
<code>out</code>	<code>len</code>	The length of the nv data.

Return values

<code>IOT_ERROR_NONE</code>	Get nv data successful.
<code>IOT_ERROR_INVALID_ARGS</code>	Invalid argument.
<code>IOT_ERROR_NV_DATA_ERROR</code>	Get nv data failed.

Warning

The caller is always responsible to free the allocated pointer after using the data.

5.45.2.13 iot_nv_get_serial_number() `iot_error_t iot_nv_get_serial_number (`
`char ** sn,`
`unsigned int * len)`

Get a serial number from the nv file-system.

Parameters

<code>out</code>	<code>sn</code>	A pointer to data array to store the serial number from the nv file-system.
<code>out</code>	<code>len</code>	The length of the nv data.

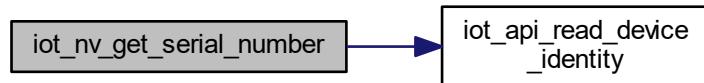
Return values

<code>IOT_ERROR_NONE</code>	Get nv data successful.
<code>IOT_ERROR_INVALID_ARGS</code>	Invalid argument.
<code>IOT_ERROR_NV_DATA_ERROR</code>	Get nv data failed.

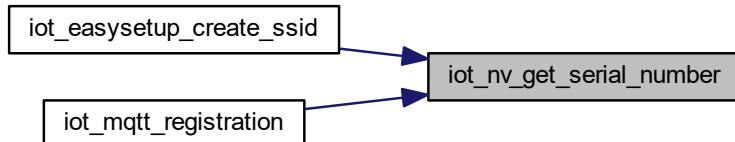
Warning

The caller is always responsible to free the allocated pointer after using the data.

Here is the call graph for this function:



Here is the caller graph for this function:



5.45.2.14 `iot_nv_get_wifi_prov_data()` `iot_error_t iot_nv_get_wifi_prov_data (struct iot_wifi_prov_data * wifi_prov)`

Get wifi provisioning data from the nv file-system.

Parameters

<code>out</code>	<code>wifi_prov</code>	A pointer to data structure to store the wifi provisioning data from the nv file-system.
------------------	------------------------	--

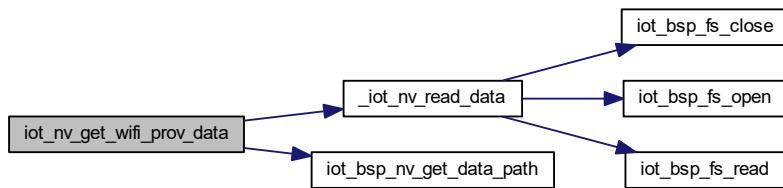
Return values

<code>IOT_ERROR_NONE</code>	Get nv data successful.
<code>IOT_ERROR_INVALID_ARGS</code>	Invalid argument.
<code>IOT_ERROR_NV_DATA_ERROR</code>	Get nv data failed.

Warning

The caller is always responsible to free the allocated pointer after using the data.

Here is the call graph for this function:



Here is the caller graph for this function:



5.45.2.15 `iot_nv_init()` `iot_error_t iot_nv_init (`
`unsigned char * device_info,`
`unsigned int device_info_len)`

Initialize a nv file-system.

This function initializes the nv file-system of the device. You must call this function before using nv management function.

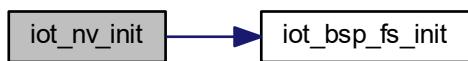
Return values

<i>IOT_ERROR_NONE</i>	NV file-system init successful.
<i>IOT_ERROR_INIT_FAILED</i>	NV file-system init failed.

See also

[iot_bsp_fs_init](#)

Here is the call graph for this function:



Here is the caller graph for this function:



5.45.2.16 *iot_nv_set_cloud_prov_data()* *iot_error_t* *iot_nv_set_cloud_prov_data* (
 *struct iot_cloud_prov_data * cloud_prov*)

Set cloud provisioning data to the nv file-system.

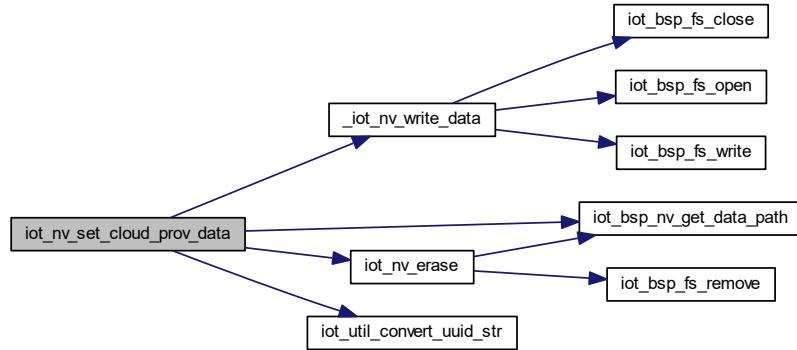
Parameters

<i>in</i>	<i>cloud_prov</i>	A pointer to cloud provisioning data structure.
-----------	-------------------	---

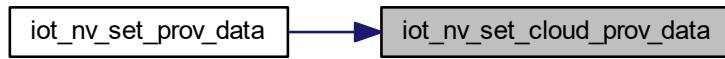
Return values

<i>IOT_ERROR_NONE</i>	Set nv data successful.
<i>IOT_ERROR_INVALID_ARGS</i>	Invalid argument.
<i>IOT_ERROR_NV_DATA_ERROR</i>	Set nv data failed.

Here is the call graph for this function:



Here is the caller graph for this function:



5.45.2.17 `iot_nv_set_device_id()` `iot_error_t iot_nv_set_device_id (const char * device_id)`

Set a device id to the nv file-system.

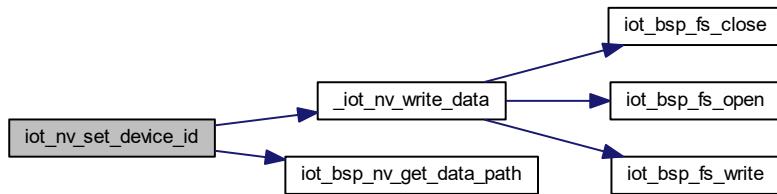
Parameters

in	<code>device_id</code>	The string of device id from MQTT Server.
----	------------------------	---

Return values

<code>IOT_ERROR_NONE</code>	Set nv data successful.
<code>IOT_ERROR_INVALID_ARGS</code>	Invalid argument.
<code>IOT_ERROR_NV_DATA_ERROR</code>	Set nv data failed.

Here is the call graph for this function:



5.45.2.18 iot_nv_set_prov_data() `iot_error_t iot_nv_set_prov_data (struct iot_device_prov_data * prov_data)`

Set provisioning data to the nv file-system.

Parameters

in	<code>prov_data</code>	A pointer to provisioning data structure.
----	------------------------	---

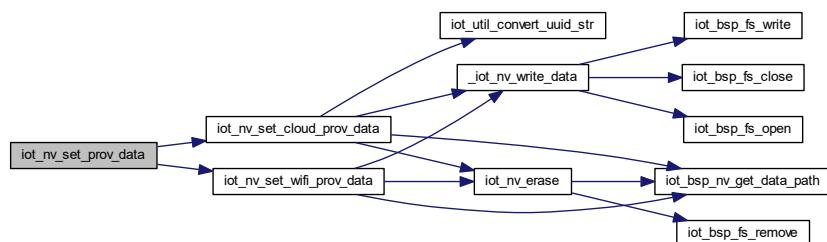
Return values

<code>IOT_ERROR_NONE</code>	Set nv data successful.
<code>IOT_ERROR_INVALID_ARGS</code>	Invalid argument.
<code>IOT_ERROR_NV_DATA_ERROR</code>	Set nv data failed.

See also

[iot_nv_set_wifi_prov_data](#)
[iot_nv_set_cloud_prov_data](#)

Here is the call graph for this function:



5.45.2.19 iot_nv_set_wifi_prov_data() `iot_error_t iot_nv_set_wifi_prov_data (struct iot_wifi_prov_data * wifi_prov)`

Set wifi provisioning data to the nv file-system.

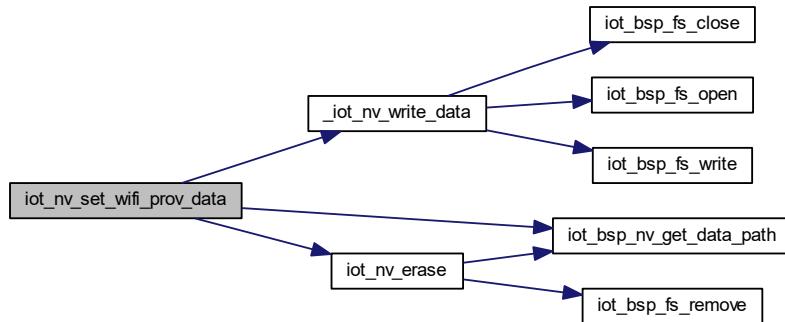
Parameters

in	<code>wifi_prov</code>	A pointer to wifi provisioning data structure.
----	------------------------	--

Return values

<code>IOT_ERROR_NONE</code>	Set nv data successful.
<code>IOT_ERROR_INVALID_ARGS</code>	Invalid argument.
<code>IOT_ERROR_NV_DATA_ERROR</code>	Set nv data failed.

Here is the call graph for this function:



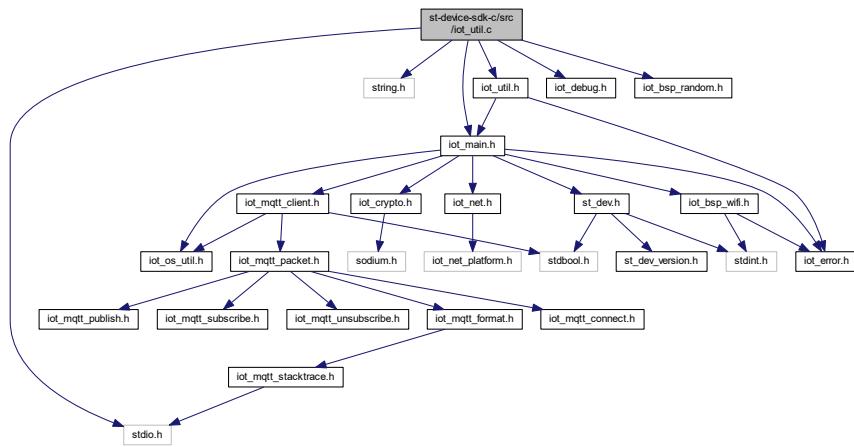
Here is the caller graph for this function:



5.46 st-device-sdk-c/src/iot_util.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "iot_main.h"
#include "iot_util.h"
```

```
#include "iot_debug.h"
#include "iot_bsp_random.h"
Include dependency graph for iot_util.c:
```



Functions

- `iot_error_t iot_util_convert_str_uuid (const char *str, struct iot_uuid *uuid)`
uuid type string to iot_uuid struct converting function for st-iot-core
- `iot_error_t iot_util_convert_uuid_str (struct iot_uuid *uuid, char *str, int max_sz)`
iot_uuid struct based value to uuid type string converting function for st-iot-core
- `iot_error_t iot_util_get_random_uuid (struct iot_uuid *uuid)`
To get random uuid based on iot_uuid struct.
- `iot_error_t iot_util_convert_str_mac (char *str, struct iot_mac *mac)`
To convert WiFi mac string into iot_mac struct value.
- `iot_error_t iot_util_convert_mac_str (struct iot_mac *mac, char *str, int max_sz)`
To convert iot_mac value into WiFi mac string.
- `uint16_t iot_util_convert_channel_freq (uint8_t channel)`
To convert Wi-Fi channel into frequency value.
- `iot_error_t iot_util_url_parse (char *url, url_parse_t *output)`
parse url with protocol, domain, port number parts for st-iot-core

5.46.1 Function Documentation

5.46.1.1 iot_util_convert_channel_freq() uint16_t iot_util_convert_channel_freq (uint8_t channel)

To convert Wi-Fi channel into frequency value.

This function tries to convert from the channel to frequency

Parameters

in	Wi-Fi	channel
----	-------	---------

Returns

Wi-Fi frequency

5.46.1.2 iot_util_convert_mac_str() `iot_error_t` `iot_util_convert_mac_str` (
 `struct iot_mac * mac,`
 `char * str,`
 `int max_sz)`

To convert `iot_mac` value intto WIFI mac string.

This function tries to convert from the `iot_mac` struct value to string

Parameters

in	<code>mac</code>	converting wanted <code>iot_mac</code> struct value pointer
in	<code>str</code>	allocated memory pointer for converted WIFI mac string
in	<code>max_sz</code>	max size of allocated memory pointer

Returns`iot_error_t`**Return values**

<code>IOT_ERROR_NONE</code>	success
<code>IOT_ERROR_INVALID_ARGS</code>	invalid arguments

5.46.1.3 iot_util_convert_str_mac() `iot_error_t` `iot_util_convert_str_mac` (
 `char * str,`
 `struct iot_mac * mac)`

To convert WIFI mac string into `iot_mac` struct value.

This function tries to convert from the string to `iot_mac` struct value

Parameters

in	<code>str</code>	WIFI mac string pointer such as 'xx:xx:xx:xx:xx:xx'
in	<code>mac</code>	allocated <code>iot_mac</code> struct pointer to get <code>iot_mac</code> value from str

Returns

`iot_error_t`

Return values

<code>IOT_ERROR_NONE</code>	success
<code>IOT_ERROR_INVALID_ARGS</code>	invalid arguments

```
5.46.1.4 iot_util_convert_str_uuid() iot_error_t iot_util_convert_str_uuid (
    const char * str,
    struct iot_uuid * uuid )
```

uuid type string to `iot_uuid` struct converting function for st-iot-core

This function tries to convert from uuid type string to `iot_uuid` struct

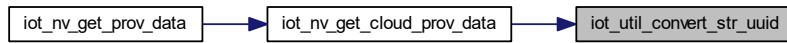
Parameters

in	<code>str</code>	uuid type string pointer such as 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
in	<code>uuid</code>	allocated <code>iot_uuid</code> struct pointer to get <code>iot_uuid</code> value from str

Returns

return `IOT_ERROR_NONE` on success, or `iot_error_t` errors if it fails

Here is the caller graph for this function:



```
5.46.1.5 iot_util_convert_uuid_str() iot_error_t iot_util_convert_uuid_str (
    struct iot_uuid * uuid,
    char * str,
    int max_sz )
```

`iot_uuid` struct based value to uuid type string converting function for st-iot-core

This function tries to convert from `iot_uuid` struct based value to uuid type string

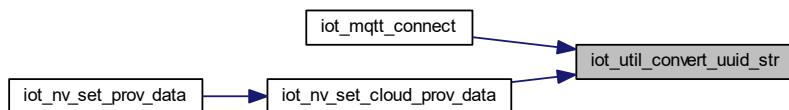
Parameters

in	<i>uuid</i>	converting wanted <code>iot_uuid</code> struct value pointer
in	<i>str</i>	allocated memory pointer for converted uuid type string
in	<i>max_sz</i>	max size of allocated memory pointer

Returns

return `IOT_ERROR_NONE` on success, or `iot_error_t` errors if it fails

Here is the caller graph for this function:



5.46.1.6 `iot_util_get_random_uuid()` `iot_error_t iot_util_get_random_uuid (struct iot_uuid * uuid)`

To get random uuid based on `iot_uuid` struct.

This function tries to make random `iot_uuid` values

Parameters

in	<i>uuid</i>	allocated <code>iot_uuid</code> struct pointer to get random <code>iot_uuid</code> value
----	-------------	--

Returns

return `IOT_ERROR_NONE` on success, or `iot_error_t` errors if it fails

Here is the call graph for this function:



```
5.46.1.7 iot_util_url_parse() iot_error_t iot_util_url_parse (
    char * url,
    url_parse_t * output )
```

parse url with protocol, domain, port number parts for st-iot-core

This function parse give url protocol, domain, port number parts

Parameters

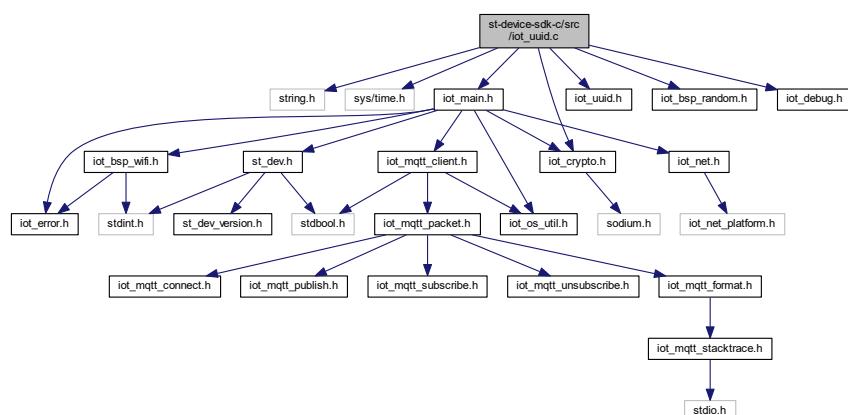
in	url	null-terminated url string like "https://example.sample.com:1234"
out	output	parsed output with url_parse_t type

Returns

return IOT_ERROR_NONE on success, or iot_error_t errors if it fails

5.47 st-device-sdk-c/src/iot_uuid.c File Reference

```
#include <string.h>
#include <sys/time.h>
#include "iot_main.h"
#include "iot_crypto.h"
#include "iot_uuid.h"
#include "iot_bsp_random.h"
#include "iot_debug.h"
Include dependency graph for iot_uuid.c:
```



Functions

- [iot_error_t iot_random_uuid_from_mac \(struct iot_uuid *uuid\)](#)
Change mac to random uuid This function changes mac to random uuid.
- [iot_error_t iot_uuid_from_mac \(struct iot_uuid *uuid\)](#)
Change mac to uuid This function changes mac to the uuid.

5.47.1 Function Documentation

5.47.1.1 iot_random_uuid_from_mac() `iot_error_t iot_random_uuid_from_mac (struct iot_uuid * uuid)`

Change mac to random uuid This function changes mac to random uuid.

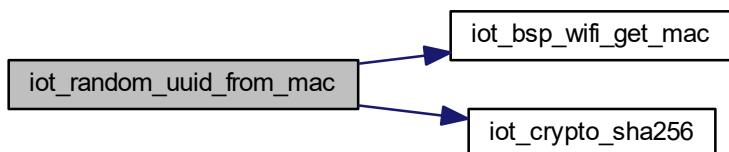
Parameters

out	<i>uuid</i>	created random uuid
-----	-------------	---------------------

Returns

`IOT_ERROR_NONE` on success

Here is the call graph for this function:



Here is the caller graph for this function:



5.47.1.2 iot_uuid_from_mac() `iot_error_t iot_uuid_from_mac (struct iot_uuid * uuid)`

Change mac to uuid This function changes mac to the uuid.

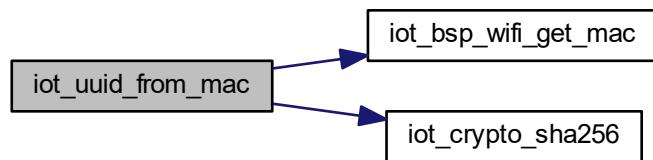
Parameters

<code>out</code>	<code>uuid</code>	<code>created uuid</code>
------------------	-------------------	---------------------------

Returns

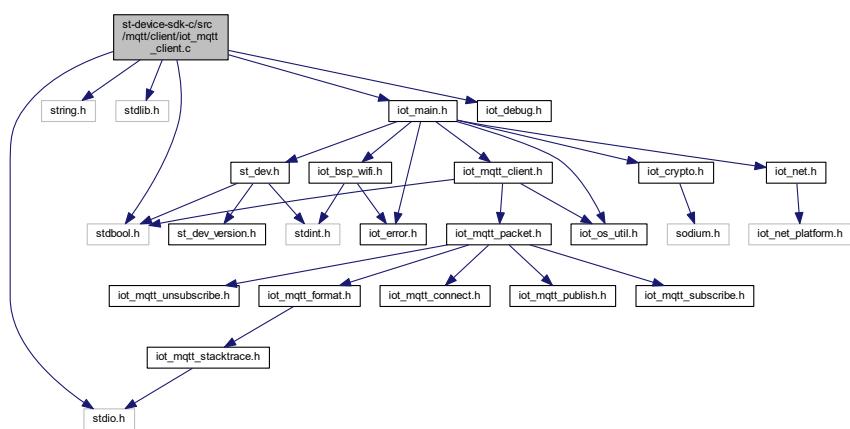
`IOT_ERROR_NONE` on success

Here is the call graph for this function:

**5.48 st-device-sdk-c/src/mqtt/client/iot_mqtt_client.c File Reference**

```

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>
#include "iot_main.h"
#include "iot_debug.h"
Include dependency graph for iot_mqtt_client.c:
  
```



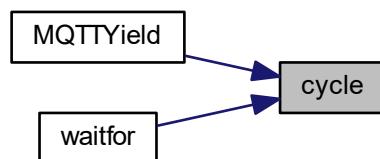
Functions

- bool `MQTTClientInit (MQTTClient *c, iot_net_interface_t *network, unsigned int command_timeout_ms, unsigned char *sendbuf, size_t sendbuf_size, unsigned char *readbuf, size_t readbuf_size)`
- int `deliverMessage (MQTTClient *c, MQTTString *topicName, MQTTMessage *message)`
- int `keepalive (MQTTClient *c)`
- void `MQTTCleanSession (MQTTClient *c)`
- void `MQTTCloseSession (MQTTClient *c)`
- int `cycle (MQTTClient *c, iot_os_timer timer)`
- int `MQTTYield (MQTTClient *c, int timeout_ms)`
- int `waitfor (MQTTClient *c, int packet_type, iot_os_timer timer)`
- int `MQTTConnectWithResults (MQTTClient *c, MQTTPacket_connectData *options, MQTTConnackData *data)`
- int `MQTTConnect (MQTTClient *c, MQTTPacket_connectData *options)`
- int `MQTTSetMessageHandler (MQTTClient *c, const char *topicFilter, messageHandler messageHandler, void *userData)`
- int `MQTTSubscribeWithResults (MQTTClient *c, const char *topicFilter, enum QoS qos, messageHandler messageHandler, MQTTSubackData *data, void *userData)`
- int `MQTTSubscribe (MQTTClient *c, const char *topicFilter, enum QoS qos, messageHandler messageHandler, void *userData)`
- int `MQTTUnsubscribe (MQTTClient *c, const char *topicFilter)`
- int `MQTTPublish (MQTTClient *c, const char *topicName, MQTTMessage *message)`
- int `MQTTDisconnect (MQTTClient *c)`

5.48.1 Function Documentation

```
5.48.1.1 cycle() int cycle (
    MQTTClient * c,
    iot_os_timer timer )
```

Here is the caller graph for this function:



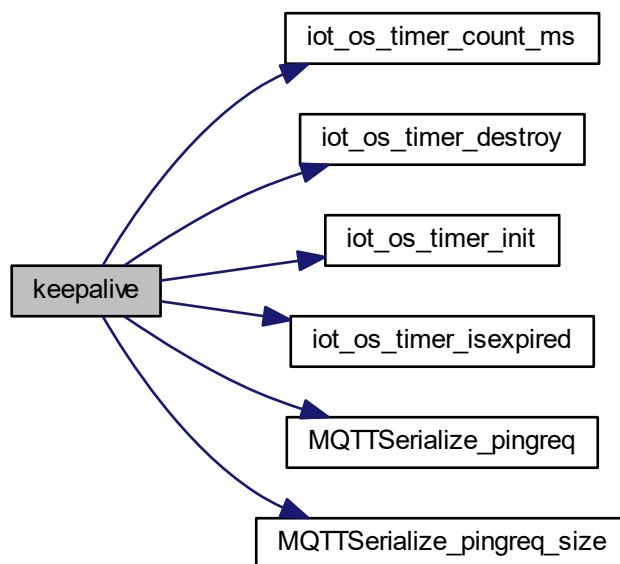
```
5.48.1.2 deliverMessage() int deliverMessage (
    MQTTClient * c,
    MQTTString * topicName,
    MQTTMessage * message )
```

Here is the call graph for this function:



```
5.48.1.3 keepalive() int keepalive (
    MQTTClient * c )
```

Here is the call graph for this function:

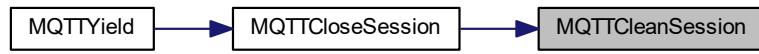


Here is the caller graph for this function:



5.48.1.4 MQTTCleanSession() `void MQTTCleanSession (`
 `MQTTClient * c)`

Here is the caller graph for this function:



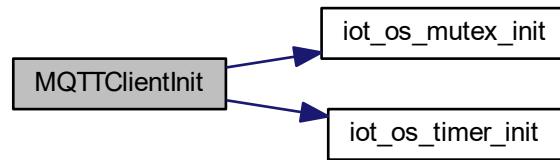
5.48.1.5 MQTTClientInit() `bool MQTTClientInit (`
 `MQTTClient * client,`
 `iot_net_interface_t * network,`
 `unsigned int command_timeout_ms,`
 `unsigned char * sendbuf,`
 `size_t sendbuf_size,`
 `unsigned char * readbuf,`
 `size_t readbuf_size)`

Create an MQTT client object

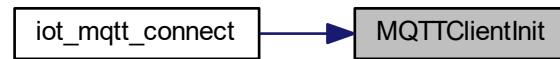
Parameters

<code>client</code>	
<code>network</code>	
<code>command_timeout_ms</code>	

Here is the call graph for this function:



Here is the caller graph for this function:

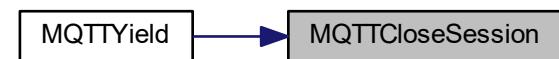


5.48.1.6 MQTTCloseSession() `void MQTTCloseSession (`
`MQTTClient * c)`

Here is the call graph for this function:



Here is the caller graph for this function:



```
5.48.1.7 MQTTConnect() int MQTTConnect (
    MQTTClient * client,
    MQTTPacket_connectData * options )
```

MQTT Connect - send an MQTT connect packet down the network and wait for a Connack The nework object must be connected to the network endpoint before calling this

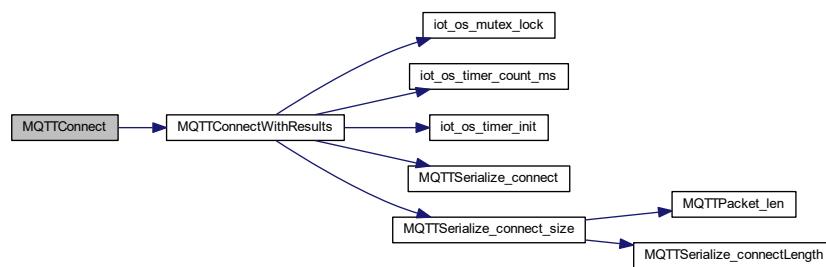
Parameters

<i>options</i>	- connect options
----------------	-------------------

Returns

success code

Here is the call graph for this function:



Here is the caller graph for this function:



```
5.48.1.8 MQTTConnectWithResults() int MQTTConnectWithResults (
    MQTTClient * client,
    MQTTPacket_connectData * options,
    MQTTConnackData * data )
```

MQTT Connect - send an MQTT connect packet down the network and wait for a Connack The nework object must be connected to the network endpoint before calling this

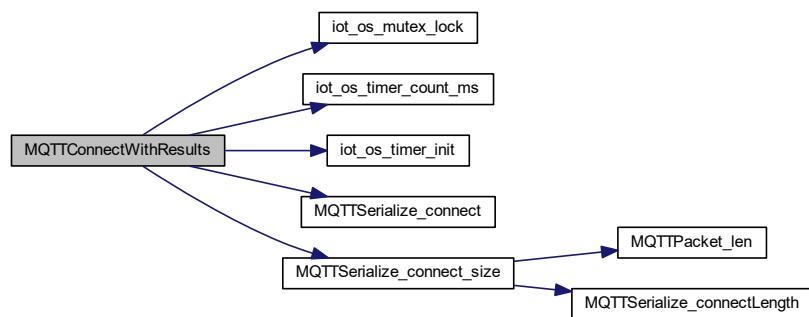
Parameters

<i>options</i>	- connect options
----------------	-------------------

Returns

success code

Here is the call graph for this function:



Here is the caller graph for this function:



5.48.1.9 MQTTDisconnect() `int MQTTDisconnect (MQTTClient * client)`

MQTT Disconnect - send an MQTT disconnect packet and close the connection

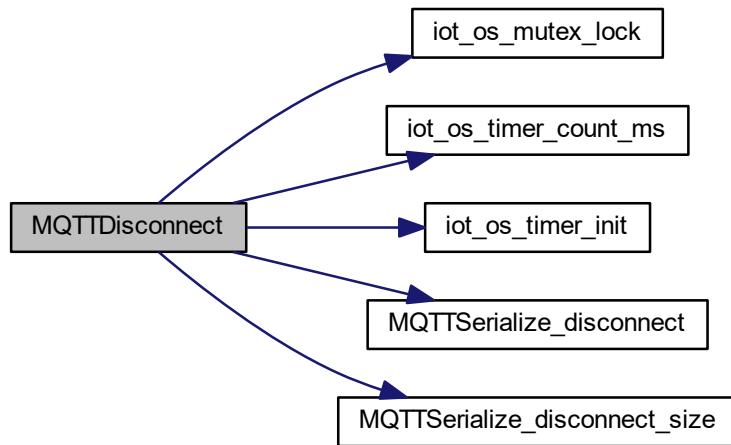
Parameters

<i>client</i>	- the client object to use
---------------	----------------------------

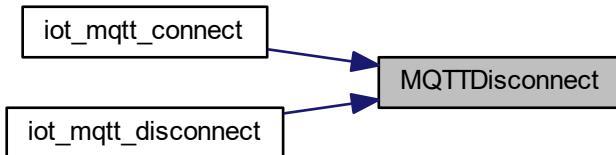
Returns

success code

Here is the call graph for this function:



Here is the caller graph for this function:



```

5.48.1.10 MQTTPublish() int MQTTPublish (
    MQTTClient * client,
    const char * ,
    MQTTMessage * )
  
```

MQTT Publish - send an MQTT publish packet and wait for all acks to complete for all QoSs

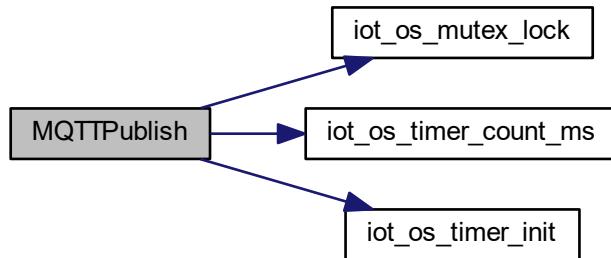
Parameters

<i>client</i>	- the client object to use
<i>topic</i>	- the topic to publish to
<i>message</i>	- the message to send

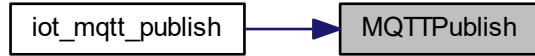
Returns

success code

Here is the call graph for this function:



Here is the caller graph for this function:

**5.48.1.11 MQTTSetMessageHandler()** `int MQTTSetMessageHandler (`

```

    MQTTClient * c,
    const char * topicFilter,
    messageHandler messageHandler,
    void * userData )
  
```

MQTT SetMessageHandler - set or remove a per topic message handler

Parameters

<code>client</code>	- the client object to use
<code>topicFilter</code>	- the topic filter set the message handler for
<code>messageHandler</code>	- pointer to the message handler function or NULL to remove

Returns

success code

```
5.48.1.12 MQTTSubscribe() int MQTTSubscribe (
    MQTTClient * client,
    const char * topicFilter,
    enum QoS,
    messageHandler ,
    void * userData )
```

MQTT Subscribe - send an MQTT subscribe packet and wait for suback before returning.

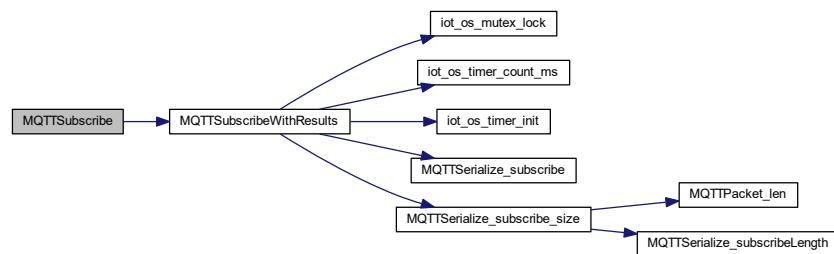
Parameters

<i>client</i>	- the client object to use
<i>topicFilter</i>	- the topic filter to subscribe to
<i>message</i>	- the message to send

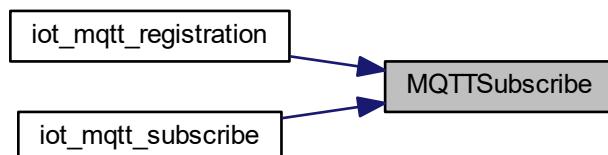
Returns

success code

Here is the call graph for this function:



Here is the caller graph for this function:



```
5.48.1.13 MQTTSubscribeWithResults() int MQTTSubscribeWithResults (
    MQTTClient * client,
    const char * topicFilter,
    enum QoS,
    messageHandler ,
    MQTTSubackData * data,
    void * userData )
```

MQTT Subscribe - send an MQTT subscribe packet and wait for suback before returning.

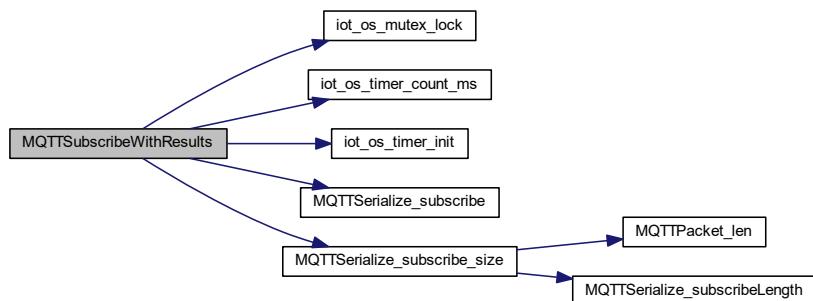
Parameters

<i>client</i>	- the client object to use
<i>topicFilter</i>	- the topic filter to subscribe to
<i>message</i>	- the message to send
<i>data</i>	- suback granted QoS returned

Returns

success code

Here is the call graph for this function:



Here is the caller graph for this function:



```
5.48.1.14 MQTTUnsubscribe() int MQTTUnsubscribe (  
    MQTTClient * client,  
    const char * topicFilter )
```

MQTT Subscribe - send an MQTT unsubscribe packet and wait for unsuback before returning.

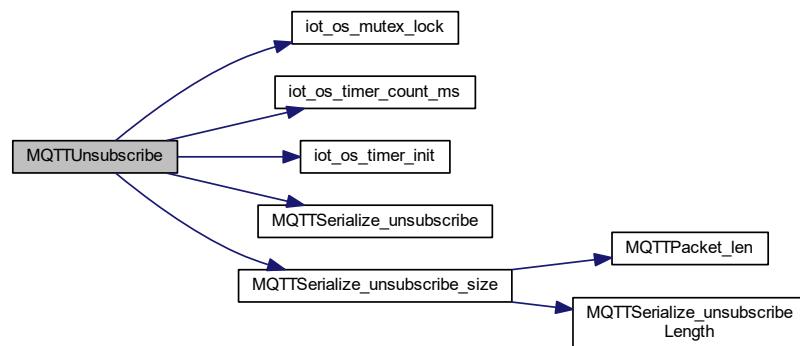
Parameters

<i>client</i>	- the client object to use
<i>topicFilter</i>	- the topic filter to unsubscribe from

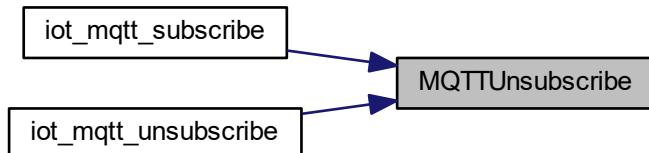
Returns

success code

Here is the call graph for this function:



Here is the caller graph for this function:



```

5.48.1.15 MQTYYield() int MQTYYield (
    MQTTClient * client,
    int time )
  
```

MQTT Yield - MQTT background

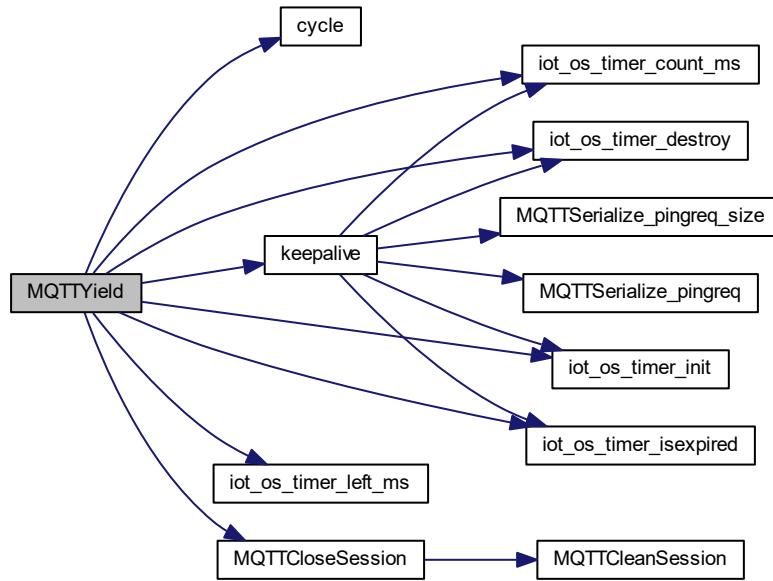
Parameters

<i>client</i>	- the client object to use
<i>time</i>	- the time, in milliseconds, to yield for

Returns

success code

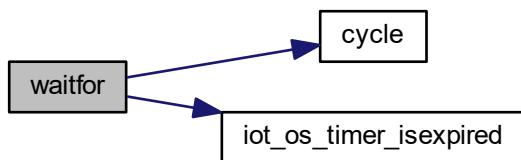
Here is the call graph for this function:



```

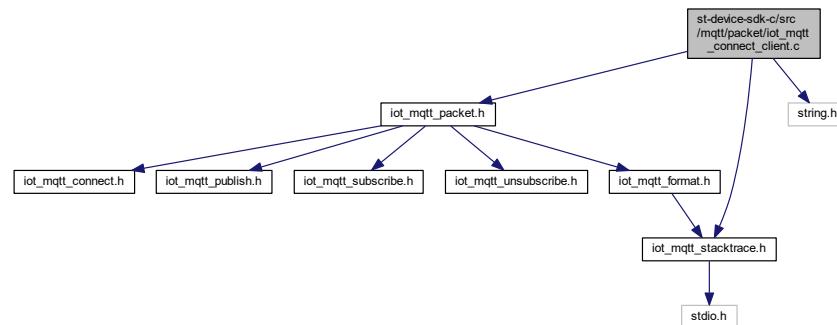
5.48.1.16 waitfor() int waitfor (
    MQTTClient * c,
    int packet_type,
    iot_os_timer timer )
  
```

Here is the call graph for this function:



5.49 st-device-sdk-c/src/mqtt/packet/iot_mqtt_connect_client.c File Reference

```
#include "iot_mqtt_packet.h"
#include "iot_mqtt_stacktrace.h"
#include <string.h>
Include dependency graph for iot_mqtt_connect_client.c:
```



Functions

- int `MQTTSerialize_connectLength (MQTTPacket_connectData *options)`
- int `MQTTSerialize_connect (unsigned char *buf, int buflen, MQTTPacket_connectData *options)`
- int `MQTTSerialize_connect_size (MQTTPacket_connectData *options)`
- int `MQTTDeserialize_connack (unsigned char *sessionPresent, unsigned char *connack_rc, unsigned char *buf, int buflen)`
- int `MQTTSerialize_zero (unsigned char *buf, int buflen, unsigned char packettype)`
- int `MQTTSerialize_disconnect (unsigned char *buf, int buflen)`
- int `MQTTSerialize_disconnect_size ()`
- int `MQTTSerialize_pingreq (unsigned char *buf, int buflen)`
- int `MQTTSerialize_pingreq_size ()`

5.49.1 Function Documentation

```
5.49.1.1 MQTTDeserialize_connack() int MQTTDeserialize_connack (
    unsigned char * sessionPresent,
    unsigned char * connack_rc,
    unsigned char * buf,
    int buflen )
```

Deserializes the supplied (wire) buffer into connack data - return code

Parameters

<code>sessionPresent</code>	the session present flag returned (only for MQTT 3.1.1)
<code>connack_rc</code>	returned integer value of the connack return code
<code>buf</code>	the raw buffer data, of the correct length determined by the remaining length field
<code>len</code>	the length in bytes of the data in the supplied buffer

Returns

error code. 1 is success, 0 is failure

```
5.49.1.2 MQTTSerialize_connect() int MQTTSerialize_connect (
    unsigned char * buf,
    int buflen,
    MQTTPacket_connectData * options )
```

Serializes the connect options into the buffer.

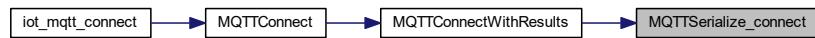
Parameters

<i>buf</i>	the buffer into which the packet will be serialized
<i>len</i>	the length in bytes of the supplied buffer
<i>options</i>	the options to be used to build the connect packet

Returns

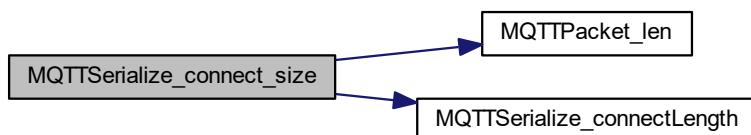
serialized length, or error if 0

Here is the caller graph for this function:



```
5.49.1.3 MQTTSerialize_connect_size() int MQTTSerialize_connect_size (
    MQTTPacket_connectData * options )
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.49.1.4 MQTTSanitize_connectLength() `int MQTTSanitize_connectLength (MQTTPacket_connectData * options)`

Determines the length of the MQTT connect packet that would be produced using the supplied connect options.

Parameters

<code>options</code>	the options to be used to build the connect packet
----------------------	--

Returns

the length of buffer needed to contain the serialized version of the packet

Here is the caller graph for this function:



5.49.1.5 MQTTSanitize_disconnect() `int MQTTSanitize_disconnect (unsigned char * buf, int buflen)`

Serializes a disconnect packet into the supplied buffer, ready for writing to a socket

Parameters

<code>buf</code>	the buffer into which the packet will be serialized
<code>buflen</code>	the length in bytes of the supplied buffer, to avoid overruns

Returns

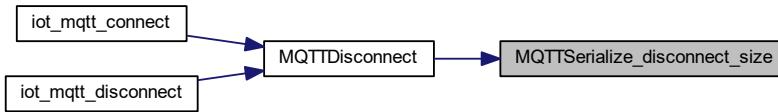
serialized length, or error if 0

Here is the caller graph for this function:



5.49.1.6 MQTTSerialize_disconnect_size() int MQTTSerialize_disconnect_size ()

Here is the caller graph for this function:



5.49.1.7 MQTTSerialize_pingreq() int MQTTSerialize_pingreq (

```

unsigned char * buf,
int buflen )
  
```

Serializes a disconnect packet into the supplied buffer, ready for writing to a socket

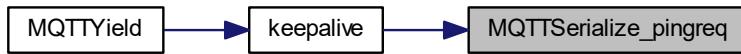
Parameters

<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer, to avoid overruns

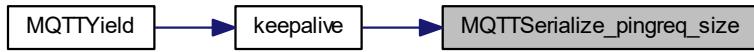
Returns

serialized length, or error if 0

Here is the caller graph for this function:

**5.49.1.8 MQTTSerialize_pingreq_size()** `int MQTTSerialize_pingreq_size ()`

Here is the caller graph for this function:

**5.49.1.9 MQTTSerialize_zero()** `int MQTTSerialize_zero (`

```
    unsigned char * buf,
    int buflen,
    unsigned char packettype )
```

Serializes a 0-length packet into the supplied buffer, ready for writing to a socket

Parameters

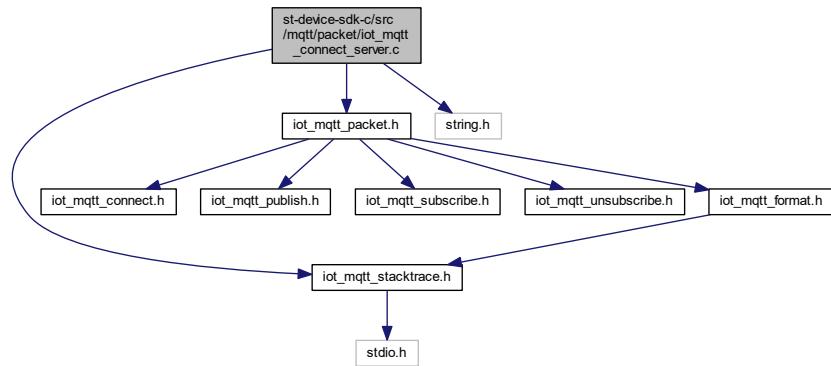
<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer, to avoid overruns
<i>packettype</i>	the message type

Returns

serialized length, or error if 0

5.50 st-device-sdk-c/src/mqtt/packet/iot_mqtt_connect_server.c File Reference

```
#include "iot_mqtt_stacktrace.h"
#include "iot_mqtt_packet.h"
#include <string.h>
Include dependency graph for iot_mqtt_connect_server.c:
```



Macros

- #define min(a, b) ((a < b) ? a : b)

Functions

- int **MQTTPacket_checkVersion** (MQTTString *protocol, int version)
- int **MQTTDeserialize_connect** (MQTTPacket_connectData *data, unsigned char *buf, int len)
- int **MQTTSerialize_connack** (unsigned char *buf, int buflen, unsigned char connack_rc, unsigned char sessionPresent)

5.50.1 Macro Definition Documentation

5.50.1.1 min #define min(

```

    a,
    b ) ((a < b) ? a : b)
```

5.50.2 Function Documentation

5.50.2.1 **MQTTDeserialize_connect()** int MQTTDeserialize_connect (

```

MQTTPacket_connectData * data,
unsigned char * buf,
int len )
```

Deserializes the supplied (wire) buffer into connect data structure

Parameters

<i>data</i>	the connect data structure to be filled out
<i>buf</i>	the raw buffer data, of the correct length determined by the remaining length field
<i>len</i>	the length in bytes of the data in the supplied buffer

Returns

error code. 1 is success, 0 is failure

5.50.2.2 MQTTPacket_checkVersion() `int MQTTPacket_checkVersion (`
`MQTTString * protocol,`
`int version)`

Validates MQTT protocol name and version combinations

Parameters

<i>protocol</i>	the MQTT protocol name as an MQTTString
<i>version</i>	the MQTT protocol version number, as in the connect packet

Returns

correct MQTT combination? 1 is true, 0 is false

5.50.2.3 MQTTSerialize_connack() `int MQTTSerialize_connack (`
`unsigned char * buf,`
`int buflen,`
`unsigned char connack_rc,`
`unsigned char sessionPresent)`

Serializes the connack packet into the supplied buffer.

Parameters

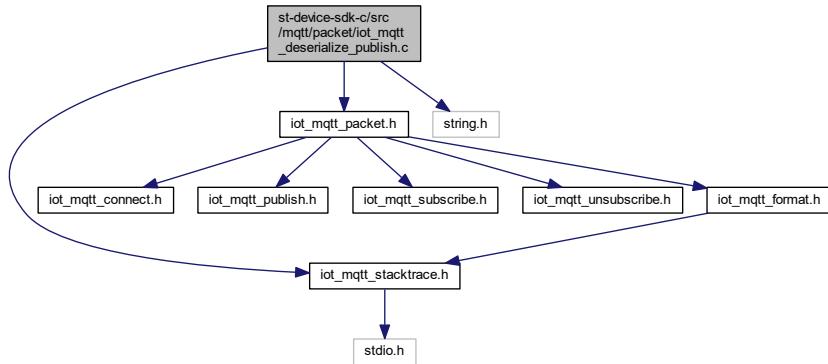
<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer
<i>connack_rc</i>	the integer connack return code to be used
<i>sessionPresent</i>	the MQTT 3.1.1 sessionPresent flag

Returns

serialized length, or error if 0

5.51 st-device-sdk-c/src/mqtt/packet/iot_mqtt_deserialize_publish.c File Reference

```
#include "iot_mqtt_stacktrace.h"
#include "iot_mqtt_packet.h"
#include <string.h>
Include dependency graph for iot_mqtt_deserialize_publish.c:
```



Macros

- #define min(a, b) ((a < b) ? 1 : 0)

Functions

- int **MQTTDeserialize_publish** (unsigned char *dup, int *qos, unsigned char *retained, unsigned short *packetid, **MQTTString** *topicName, unsigned char **payload, int *payloadlen, unsigned char *buf, int buflen)
- int **MQTTDeserialize_ack** (unsigned char *packettype, unsigned char *dup, unsigned short *packetid, unsigned char *buf, int buflen)

5.51.1 Macro Definition Documentation

```
5.51.1.1 min #define min(
    a,
    b ) ((a < b) ? 1 : 0)
```

5.51.2 Function Documentation

```
5.51.2.1 MQTTDeserialize_ack() int MQTTDeserialize_ack (
    unsigned char * packettype,
    unsigned char * dup,
    unsigned short * packetid,
    unsigned char * buf,
    int buflen )
```

Deserializes the supplied (wire) buffer into an ack

Parameters

<i>packettype</i>	returned integer - the MQTT packet type
<i>dup</i>	returned integer - the MQTT dup flag
<i>packetid</i>	returned integer - the MQTT packet identifier
<i>buf</i>	the raw buffer data, of the correct length determined by the remaining length field
<i>buflen</i>	the length in bytes of the data in the supplied buffer

Returns

error code. 1 is success, 0 is failure

5.51.2.2 MQTTDeserialize_publish() `int MQTTDeserialize_publish (`

```
unsigned char * dup,
int * qos,
unsigned char * retained,
unsigned short * packetid,
MQTTString * topicName,
unsigned char ** payload,
int * payloadlen,
unsigned char * buf,
int buflen )
```

Deserializes the supplied (wire) buffer into publish data

Parameters

<i>dup</i>	returned integer - the MQTT dup flag
<i>qos</i>	returned integer - the MQTT QoS value
<i>retained</i>	returned integer - the MQTT retained flag
<i>packetid</i>	returned integer - the MQTT packet identifier
<i>topicName</i>	returned MQTTString - the MQTT topic in the publish
<i>payload</i>	returned byte buffer - the MQTT publish payload
<i>payloadlen</i>	returned integer - the length of the MQTT payload
<i>buf</i>	the raw buffer data, of the correct length determined by the remaining length field
<i>buflen</i>	the length in bytes of the data in the supplied buffer

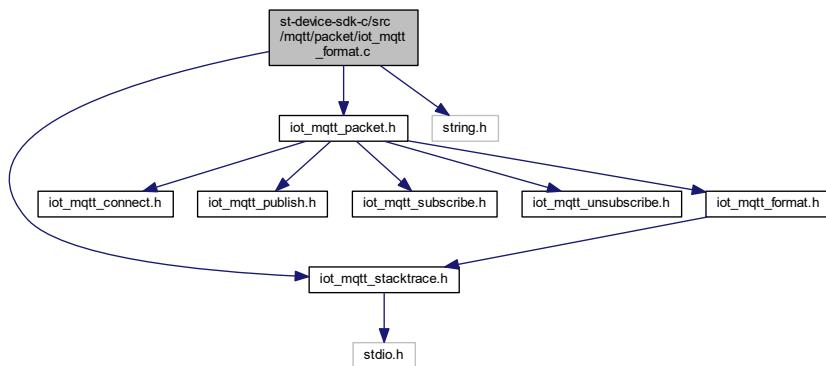
Returns

error code. 1 is success

5.52 st-device-sdk-c/src/mqtt/packet/iot_mqtt_format.c File Reference

```
#include "iot_mqtt_stacktrace.h"
#include "iot_mqtt_packet.h"
```

```
#include <string.h>
Include dependency graph for iot_mqtt_format.c:
```



Functions

- const char * `MQTTPacket_getName` (unsigned short packetid)
- int `MQTTStringFormat_connect` (char *strbuf, int strbuflen, `MQTTPacket_connectData` *data)
- int `MQTTStringFormat_connack` (char *strbuf, int strbuflen, unsigned char connack_rc, unsigned char sessionPresent)
- int `MQTTStringFormat_publish` (char *strbuf, int strbuflen, unsigned char dup, int qos, unsigned char retained, unsigned short packetid, `MQTTString` topicName, unsigned char *payload, int payloadlen)
- int `MQTTStringFormat_ack` (char *strbuf, int strbuflen, unsigned char packettype, unsigned char dup, unsigned short packetid)
- int `MQTTStringFormat_subscribe` (char *strbuf, int strbuflen, unsigned char dup, unsigned short packetid, int count, `MQTTString` topicFilters[], int requestedQoSs[])
- int `MQTTStringFormat_suback` (char *strbuf, int strbuflen, unsigned short packetid, int count, int *grantedQoSs)
- int `MQTTStringFormat_unsubscribe` (char *strbuf, int strbuflen, unsigned char dup, unsigned short packetid, int count, `MQTTString` topicFilters[])

Variables

- const char * `MQTTPacket_names` []

5.52.1 Function Documentation

5.52.1.1 `MQTTPacket_getName()` const char* `MQTTPacket_getName` (
 unsigned short `packetid`)

5.52.1.2 MQTTStringFormat_ack() int MQTTStringFormat_ack (

```
char * strbuf,
int strbuflen,
unsigned char packettype,
unsigned char dup,
unsigned short packetid )
```

5.52.1.3 MQTTStringFormat_connack() int MQTTStringFormat_connack (

```
char * strbuf,
int strbuflen,
unsigned char connack_rc,
unsigned char sessionPresent )
```

5.52.1.4 MQTTStringFormat_connect() int MQTTStringFormat_connect (

```
char * strbuf,
int strbuflen,
MQTTPacket_connectData * data )
```

5.52.1.5 MQTTStringFormat_publish() int MQTTStringFormat_publish (

```
char * strbuf,
int strbuflen,
unsigned char dup,
int qos,
unsigned char retained,
unsigned short packetid,
MQTTString topicName,
unsigned char * payload,
int payloadlen )
```

5.52.1.6 MQTTStringFormat_suback() int MQTTStringFormat_suback (

```
char * strbuf,
int strbuflen,
unsigned short packetid,
int count,
int * grantedQoSs )
```

5.52.1.7 MQTTStringFormat_subscribe() int MQTTStringFormat_subscribe (

```
char * strbuf,
int strbuflen,
unsigned char dup,
unsigned short packetid,
int count,
MQTTString topicFilters[],
int requestedQoSs[] )
```

```
5.52.1.8 MQTTStringFormat_unsubscribe() int MQTTStringFormat_unsubscribe (
    char * strbuf,
    int strbuflen,
    unsigned char dup,
    unsigned short packetid,
    int count,
    MQTTString topicFilters[] )
```

5.52.2 Variable Documentation

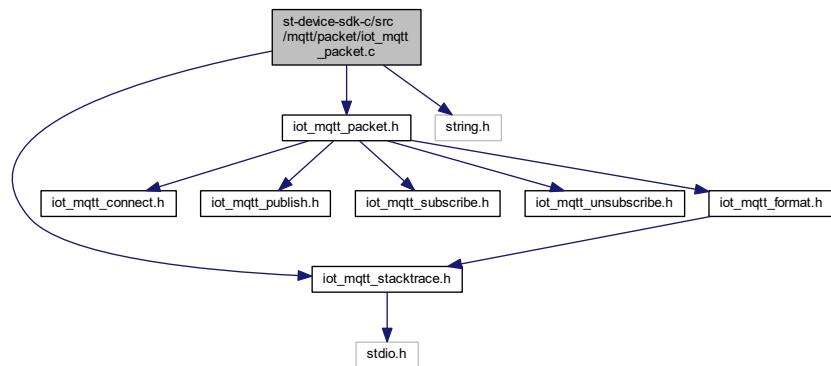
5.52.2.1 MQTTPacket_names const char* MQTTPacket_names[]

Initial value:

```
=
{
    "RESERVED", "CONNECT", "CONNACK", "PUBLISH", "PUBACK", "PUBREC", "PUBREL",
    "PUBCOMP", "SUBSCRIBE", "SUBACK", "UNSUBSCRIBE", "UNSUBACK",
    "PINGREQ", "PINGRESP", "DISCONNECT"
}
```

5.53 st-device-sdk-c/src/mqtt/packet/iot_mqtt_packet.c File Reference

```
#include "iot_mqtt_stacktrace.h"
#include "iot_mqtt_packet.h"
#include <string.h>
Include dependency graph for iot_mqtt_packet.c:
```



Macros

- #define MAX_NO_OF_REMAINING_LENGTH_BYTES 4

Functions

- int `MQTTPacket_encode` (unsigned char *buf, int length)
- int `MQTTPacket_decode` (int(*getcharfn)(unsigned char *, int), int *value)
- int `MQTTPacket_len` (int rem_len)
- int `bufchar` (unsigned char *c, int count)
- int `MQTTPacket_decodeBuf` (unsigned char *buf, int *value)
- int `readInt` (unsigned char **pptr)
- char `readChar` (unsigned char **pptr)
- void `writeChar` (unsigned char **pptr, char c)
- void `writelnt` (unsigned char **pptr, int anInt)
- void `writeCString` (unsigned char **pptr, const char *string)
- int `getLenStringLen` (char *ptr)
- void `writeMQTTString` (unsigned char **pptr, `MQTTString` mqttstring)
- int `readMQTTLenString` (`MQTTString` *mqttstring, unsigned char **pptr, unsigned char *enddata)
- int `MQTTstrlen` (`MQTTString` mqttstring)
- int `MQTTPacket_equals` (`MQTTString` *a, char *bptr)
- int `MQTTPacket_read` (unsigned char *buf, int buflen, int(*getfn)(unsigned char *, int))
- int `MQTTPacket_readnb` (unsigned char *buf, int buflen, `MQTTTransport` *trp)
- const char * `MQTTPacket_msgTypesToString` (enum `msgTypes` msgType)

5.53.1 Macro Definition Documentation

5.53.1.1 MAX_NO_OF_REMAINING_LENGTH_BYTES #define MAX_NO_OF_REMAINING_LENGTH_BYTES 4

5.53.2 Function Documentation

5.53.2.1 bufchar() int bufchar (

```
    unsigned char * c,
    int count )
```

5.53.2.2 getLenStringLen() int getLenStringLen (

```
    char * ptr )
```

5.53.2.3 MQTTPacket_decode() int MQTTPacket_decode (

```
    int(*)(unsigned char *, int) getcharfn,
    int * value )
```

Decodes the message length according to the MQTT algorithm

Parameters

<i>getcharfn</i>	pointer to function to read the next character from the data source
<i>value</i>	the decoded length returned

Returns

the number of bytes read from the socket

5.53.2.4 MQTTPacket_decodeBuf() `int MQTTPacket_decodeBuf (`

```
    unsigned char * buf,
    int * value )
```

5.53.2.5 MQTTPacket_encode() `int MQTTPacket_encode (`

```
    unsigned char * buf,
    int length )
```

Encodes the message length according to the MQTT algorithm

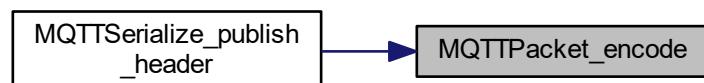
Parameters

<i>buf</i>	the buffer into which the encoded data is written
<i>length</i>	the length to be encoded

Returns

the number of bytes written to buffer

Here is the caller graph for this function:

**5.53.2.6 MQTTPacket_equals()** `int MQTTPacket_equals (`

```
    MQTTString * a,
    char * bptr )
```

Compares an [MQTTString](#) to a C string

Parameters

<i>a</i>	the MQTTString to compare
<i>bptr</i>	the C string to compare

Returns

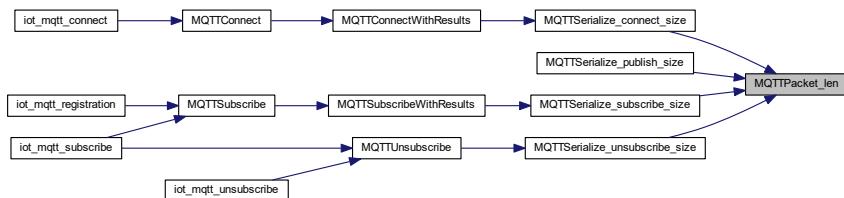
boolean - equal or not

Here is the caller graph for this function:



5.53.2.7 MQTTPacket_len() `int MQTTPacket_len (int rem_len)`

Here is the caller graph for this function:



5.53.2.8 MQTTPacket_msgTypesToString() `const char* MQTTPacket_msgTypesToString (enum msgTypes msgType)`

5.53.2.9 MQTTPacket_read() `int MQTTPacket_read (unsigned char * buf, int buflen, int(*)(unsigned char *, int) getfn)`

Helper function to read packet data from some source into a buffer

Parameters

<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer
<i>getfn</i>	pointer to a function which will read any number of bytes from the needed source

Returns

integer MQTT packet type, or -1 on error

Note

the whole message must fit into the caller's buffer

5.53.2.10 `MQTTPacket_readnb()` `int MQTTPacket_readnb (`
`unsigned char * buf,`
`int buflen,`
`MQTTTransport * trp)`

Helper function to read packet data from some source into a buffer, non-blocking

Parameters

<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer
<i>trp</i>	pointer to a transport structure holding what is needed to solve getting data from it

Returns

integer MQTT packet type, 0 for call again, or -1 on error

Note

the whole message must fit into the caller's buffer

5.53.2.11 `MQTTstrlen()` `int MQTTstrlen (`
`MQTTString mqttstring)`

Return the length of the MQTTstring - C string if there is one, otherwise the length delimited string

Parameters

<i>mqttstring</i>	the string to return the length of
-------------------	------------------------------------

Returns

the length of the string

```
5.53.2.12 readChar() char readChar (
    unsigned char ** pptr )
```

Reads one character from the input buffer.

Parameters

<i>pptr</i>	pointer to the input buffer - incremented by the number of bytes used & returned
-------------	--

Returns

the character read

```
5.53.2.13 readInt() int readInt (
    unsigned char ** pptr )
```

Calculates an integer from two bytes read from the input buffer

Parameters

<i>pptr</i>	pointer to the input buffer - incremented by the number of bytes used & returned
-------------	--

Returns

the integer value calculated

```
5.53.2.14 readMQTTLenString() int readMQTTLenString (
    MQTTString * mqtstring,
    unsigned char ** pptr,
    unsigned char * enddata )
```

Parameters

<i>mqtstring</i>	the MQTTString structure into which the data is to be read
<i>pptr</i>	pointer to the output buffer - incremented by the number of bytes used & returned
<i>enddata</i>	pointer to the end of the data: do not read beyond

Returns

1 if successful, 0 if not

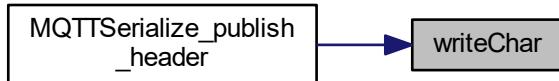
5.53.2.15 writeChar() void writeChar (unsigned char ** pptr, char c)

Writes one character to an output buffer.

Parameters

<i>pptr</i>	pointer to the output buffer - incremented by the number of bytes used & returned
<i>c</i>	the character to write

Here is the caller graph for this function:



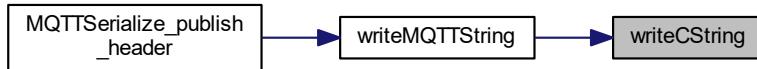
5.53.2.16 writeCString() void writeCString (unsigned char ** pptr, const char * string)

Writes a "UTF" string to an output buffer. Converts C string to length-delimited.

Parameters

<i>pptr</i>	pointer to the output buffer - incremented by the number of bytes used & returned
<i>string</i>	the C string to write

Here is the caller graph for this function:



5.53.2.17 writeInt() void writeInt (
 unsigned char ** pptr,
 int anInt)

Writes an integer as 2 bytes to an output buffer.

Parameters

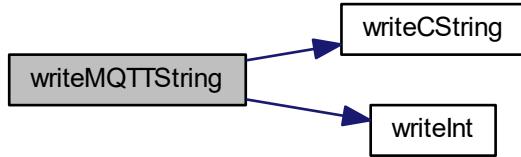
<i>pptr</i>	pointer to the output buffer - incremented by the number of bytes used & returned
<i>anInt</i>	the integer to write

Here is the caller graph for this function:

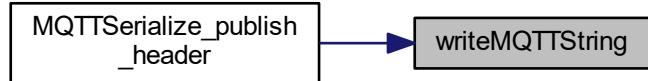


5.53.2.18 writeMQTTString() void writeMQTTString (
 unsigned char ** pptr,
 MQTTString mqttstring)

Here is the call graph for this function:

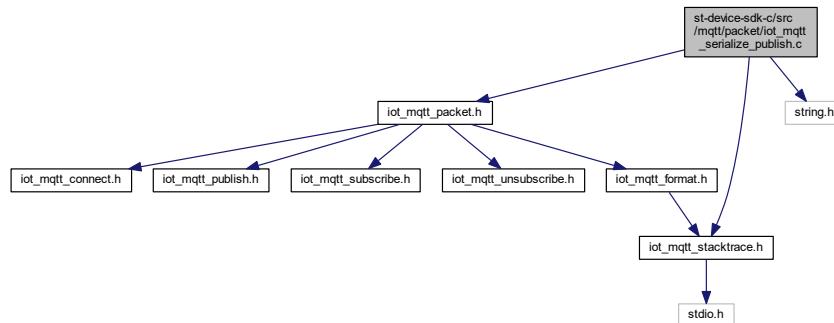


Here is the caller graph for this function:



5.54 st-device-sdk-c/src/mqtt/packet/iot_mqtt_serialize_publish.c File Reference

```
#include "iot_mqtt_packet.h"
#include "iot_mqtt_stacktrace.h"
#include <string.h>
Include dependency graph for iot_mqtt_serialize_publish.c:
```



Functions

- int [MQTTSerialize_publishLength](#) (int qos, [MQTTString](#) topicName, int payloadlen)

- int [MQTTSerialize_publish](#) (unsigned char *buf, int buflen, unsigned char dup, int qos, unsigned char retained, unsigned short packetid, [MQTTString](#) topicName, unsigned char *payload, int payloadlen)
- int [MQTTSerialize_publish_size](#) (int qos, [MQTTString](#) topicName, int payloadlen)
- int [MQTTSerialize_publish_header](#) (unsigned char *buf, unsigned char dup, int qos, unsigned char retained, unsigned short packetid, [MQTTString](#) topicName, int payloadlen)
- int [MQTTSerialize_ack](#) (unsigned char *buf, int buflen, unsigned char packettype, unsigned char dup, unsigned short packetid)
- int [MQTTSerialize_ack_size](#) ()
- int [MQTTSerialize_puback](#) (unsigned char *buf, int buflen, unsigned short packetid)
- int [MQTTSerialize_pubrel](#) (unsigned char *buf, int buflen, unsigned char dup, unsigned short packetid)
- int [MQTTSerialize_pubcomp](#) (unsigned char *buf, int buflen, unsigned short packetid)

5.54.1 Function Documentation

5.54.1.1 [MQTTSerialize_ack\(\)](#) int MQTTSerialize_ack (

```
    unsigned char * buf,
    int buflen,
    unsigned char packettype,
    unsigned char dup,
    unsigned short packetid )
```

Serializes the ack packet into the supplied buffer.

Parameters

<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer
<i>type</i>	the MQTT packet type
<i>dup</i>	the MQTT dup flag
<i>packetid</i>	the MQTT packet identifier

Returns

serialized length, or error if 0

5.54.1.2 [MQTTSerialize_ack_size\(\)](#) int MQTTSerialize_ack_size ()

5.54.1.3 [MQTTSerialize_puback\(\)](#) int MQTTSerialize_puback (

```
    unsigned char * buf,
    int buflen,
    unsigned short packetid )
```

Serializes a puback packet into the supplied buffer.

Parameters

<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer
<i>packetid</i>	integer - the MQTT packet identifier

Returns

serialized length, or error if 0

5.54.1.4 MQTTSerialize_pubcomp() `int MQTTSerialize_pubcomp (`

```
    unsigned char * buf,
    int buflen,
    unsigned short packetid )
```

Serializes a pubrel packet into the supplied buffer.

Parameters

<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer
<i>packetid</i>	integer - the MQTT packet identifier

Returns

serialized length, or error if 0

5.54.1.5 MQTTSerialize_publish() `int MQTTSerialize_publish (`

```
    unsigned char * buf,
    int buflen,
    unsigned char dup,
    int qos,
    unsigned char retained,
    unsigned short packetid,
    MQTTString topicName,
    unsigned char * payload,
    int payloadlen )
```

Serializes the supplied publish data into the supplied buffer, ready for sending

Parameters

<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer
<i>dup</i>	integer - the MQTT dup flag
<i>qos</i>	integer - the MQTT QoS value

Parameters

<i>retained</i>	integer - the MQTT retained flag
<i>packetid</i>	integer - the MQTT packet identifier
<i>topicName</i>	MQTTString - the MQTT topic in the publish
<i>payload</i>	byte buffer - the MQTT publish payload
<i>payloadlen</i>	integer - the length of the MQTT payload

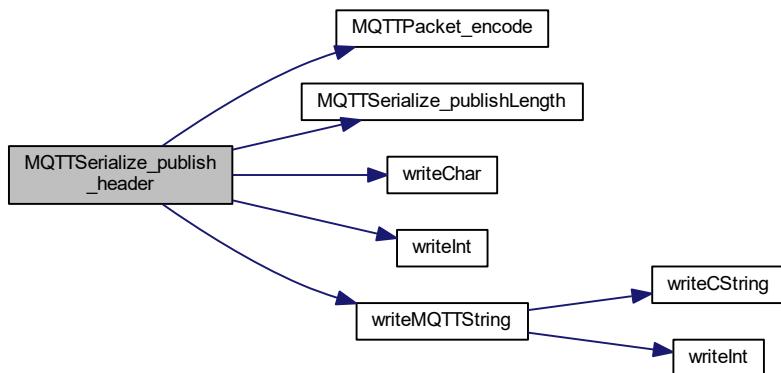
Returns

the length of the serialized data. <= 0 indicates error

5.54.1.6 MQTTSerialize_publish_header()

```
int MQTTSerialize_publish_header (
    unsigned char * buf,
    unsigned char dup,
    int qos,
    unsigned char retained,
    unsigned short packetid,
    MQTTString topicName,
    int payloadlen )
```

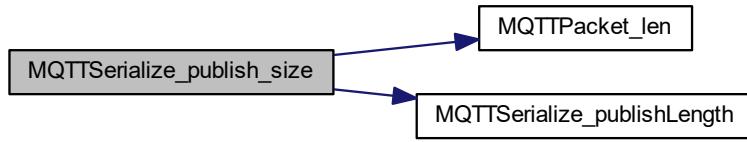
Here is the call graph for this function:



5.54.1.7 MQTTSerialize_publish_size()

```
int MQTTSerialize_publish_size (
    int qos,
    MQTTString topicName,
    int payloadlen )
```

Here is the call graph for this function:



5.54.1.8 **MQTTSerialize_publishLength()** int MQTTSerialize_publishLength (

```

int qos,
MQTTString topicName,
int payloadlen )

```

Determines the length of the MQTT publish packet that would be produced using the supplied parameters

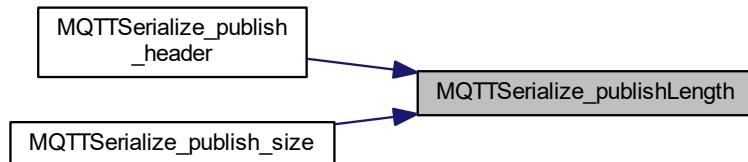
Parameters

<i>qos</i>	the MQTT QoS of the publish (packetid is omitted for QoS 0)
<i>topicName</i>	the topic name to be used in the publish
<i>payloadlen</i>	the length of the payload to be sent

Returns

the length of buffer needed to contain the serialized version of the packet

Here is the caller graph for this function:



```
5.54.1.9 MQTTSerialize_pubrel() int MQTTSerialize_pubrel (
    unsigned char * buf,
    int buflen,
    unsigned char dup,
    unsigned short packetid )
```

Serializes a pubrel packet into the supplied buffer.

Parameters

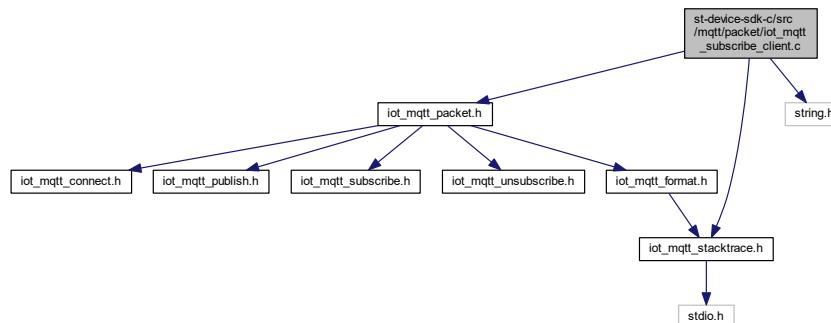
<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer
<i>dup</i>	integer - the MQTT dup flag
<i>packetid</i>	integer - the MQTT packet identifier

Returns

serialized length, or error if 0

5.55 st-device-sdk-c/src/mqtt/packet/iot_mqtt_subscribe_client.c File Reference

```
#include "iot_mqtt_packet.h"
#include "iot_mqtt_stacktrace.h"
#include <string.h>
Include dependency graph for iot_mqtt_subscribe_client.c:
```



Functions

- int **MQTTSerialize_subscribeLength** (int count, **MQTTString** topicFilters[])
- int **MQTTSerialize_subscribe** (unsigned char *buf, int buflen, unsigned char dup, unsigned short packetid, int count, **MQTTString** topicFilters[], int requestedQoSs[])
- int **MQTTSerialize_subscribe_size** (int count, **MQTTString** topicFilters[])
- int **MQTTDeserialize_suback** (unsigned short *packetid, int maxcount, int *count, int grantedQoSs[], unsigned char *buf, int buflen)

5.55.1 Function Documentation

```
5.55.1.1 MQTTDeserialize_suback() int MQTTDeserialize_suback (  
    unsigned short * packetid,  
    int maxcount,  
    int * count,  
    int grantedQoSs[],  
    unsigned char * buf,  
    int buflen )
```

Deserializes the supplied (wire) buffer into suback data

Parameters

<i>packetid</i>	returned integer - the MQTT packet identifier
<i>maxcount</i>	- the maximum number of members allowed in the grantedQoSs array
<i>count</i>	returned integer - number of members in the grantedQoSs array
<i>grantedQoSs</i>	returned array of integers - the granted qualities of service
<i>buf</i>	the raw buffer data, of the correct length determined by the remaining length field
<i>buflen</i>	the length in bytes of the data in the supplied buffer

Returns

error code. 1 is success, 0 is failure

```
5.55.1.2 MQTTSerialize_subscribe() int MQTTSerialize_subscribe (   
    unsigned char * buf,  
    int buflen,  
    unsigned char dup,  
    unsigned short packetid,  
    int count,  
    MQTTString topicFilters[],  
    int requestedQoSs[] )
```

Serializes the supplied subscribe data into the supplied buffer, ready for sending

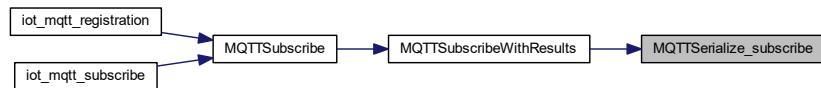
Parameters

<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer
<i>dup</i>	integer - the MQTT dup flag
<i>packetid</i>	integer - the MQTT packet identifier
<i>count</i>	- number of members in the topicFilters and reqQos arrays
<i>topicFilters</i>	- array of topic filter names
<i>requestedQoSs</i>	- array of requested QoS

Returns

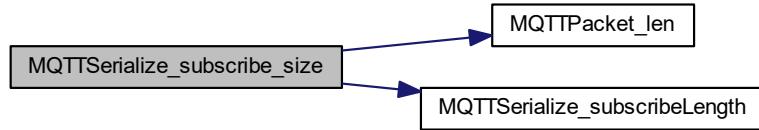
the length of the serialized data. <= 0 indicates error

Here is the caller graph for this function:

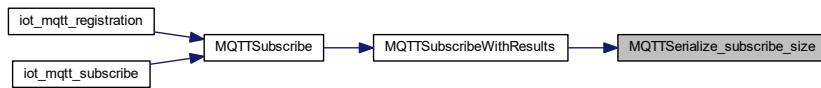
**5.55.1.3 MQTTSerialize_subscribe_size()**

```
int MQTTSerialize_subscribe_size (
    int count,
    MQTTString topicFilters[] )
```

Here is the call graph for this function:



Here is the caller graph for this function:

**5.55.1.4 MQTTSerialize_subscribeLength()**

```
int MQTTSerialize_subscribeLength (
    int count,
    MQTTString topicFilters[] )
```

Determines the length of the MQTT subscribe packet that would be produced using the supplied parameters

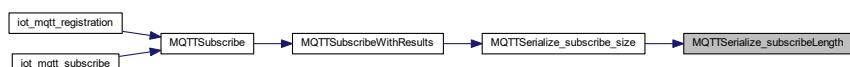
Parameters

<code>count</code>	the number of topic filter strings in <code>topicFilters</code>
<code>topicFilters</code>	the array of topic filter strings to be used in the publish

Returns

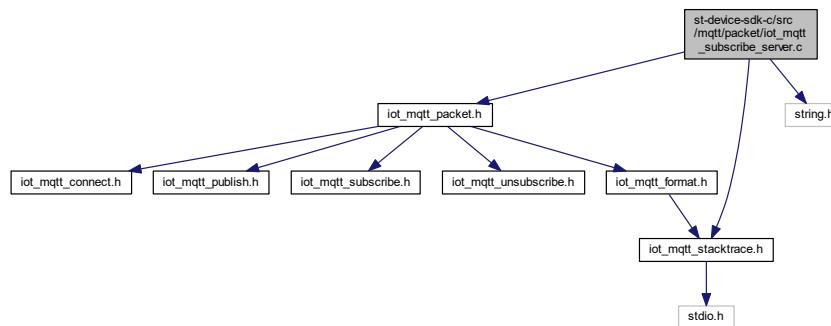
the length of buffer needed to contain the serialized version of the packet

Here is the caller graph for this function:



5.56 st-device-sdk-c/src/mqtt/packet/iot_mqtt_subscribe_server.c File Reference

```
#include "iot_mqtt_packet.h"
#include "iot_mqtt_stacktrace.h"
#include <string.h>
Include dependency graph for iot_mqtt_subscribe_server.c:
```



Functions

- int [MQTTDeserialize_subscribe](#) (unsigned char *dup, unsigned short *packetid, int maxcount, int *count, [MQTTString](#) topicFilters[], int requestedQoSs[], unsigned char *buf, int buflen)
- int [MQTTSerialize_suback](#) (unsigned char *buf, int buflen, unsigned short packetid, int count, int *grantedQoSs)

5.56.1 Function Documentation

```
5.56.1.1 MQTTDeserialize_subscribe() int MQTTDeserialize_subscribe (
    unsigned char * dup,
    unsigned short * packetid,
    int maxcount,
    int * count,
    MQTTString topicFilters[],
    int requestedQoSs[],
    unsigned char * buf,
    int buflen )
```

Deserializes the supplied (wire) buffer into subscribe data

Parameters

<i>dup</i>	integer returned - the MQTT dup flag
<i>packetid</i>	integer returned - the MQTT packet identifier
<i>maxcount</i>	- the maximum number of members allowed in the topicFilters and requestedQoSs arrays
<i>count</i>	- number of members in the topicFilters and requestedQoSs arrays
<i>topicFilters</i>	- array of topic filter names
<i>requestedQoSs</i>	- array of requested QoS
<i>buf</i>	the raw buffer data, of the correct length determined by the remaining length field
<i>buflen</i>	the length in bytes of the data in the supplied buffer

Returns

the length of the serialized data. <= 0 indicates error

```
5.56.1.2 MQTTSerialize_suback() int MQTTSerialize_suback (
```

```
    unsigned char * buf,
    int buflen,
    unsigned short packetid,
    int count,
    int * grantedQoSs )
```

Serializes the supplied suback data into the supplied buffer, ready for sending

Parameters

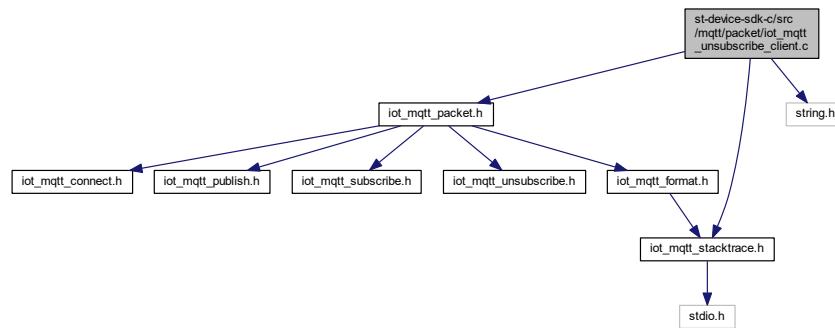
<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer
<i>packetid</i>	integer - the MQTT packet identifier
<i>count</i>	- number of members in the grantedQoSs array
<i>grantedQoSs</i>	- array of granted QoS

Returns

the length of the serialized data. <= 0 indicates error

5.57 st-device-sdk-c/src/mqtt/packet/iot_mqtt_unsubscribe_client.c File Reference

```
#include "iot_mqtt_packet.h"
#include "iot_mqtt_stacktrace.h"
#include <string.h>
Include dependency graph for iot_mqtt_unsubscribe_client.c:
```



Functions

- int [MQTTSerialize_unsubscribeLength](#) (int count, [MQTTString](#) topicFilters[])
- int [MQTTSerialize_unsubscribe](#) (unsigned char *buf, int buflen, unsigned char dup, unsigned short packetid, int count, [MQTTString](#) topicFilters[])
- int [MQTTSerialize_unsubscribe_size](#) (int count, [MQTTString](#) topicFilters[])
- int [MQTTDeserialize_unsuback](#) (unsigned short *packetid, unsigned char *buf, int buflen)

5.57.1 Function Documentation

5.57.1.1 [MQTTDeserialize_unsuback\(\)](#)

```
int MQTTDeserialize_unsuback (
    unsigned short * packetid,
    unsigned char * buf,
    int buflen )
```

Deserializes the supplied (wire) buffer into unsuback data

Parameters

<code>packetid</code>	returned integer - the MQTT packet identifier
<code>buf</code>	the raw buffer data, of the correct length determined by the remaining length field
<code>buflen</code>	the length in bytes of the data in the supplied buffer

Returns

error code. 1 is success, 0 is failure

```
5.57.1.2 MQTTSerialize_unsubscribe() int MQTTSerialize_unsubscribe (
    unsigned char * buf,
    int buflen,
    unsigned char dup,
    unsigned short packetid,
    int count,
    MQTTString topicFilters[] )
```

Serializes the supplied unsubscribe data into the supplied buffer, ready for sending

Parameters

<i>buf</i>	the raw buffer data, of the correct length determined by the remaining length field
<i>buflen</i>	the length in bytes of the data in the supplied buffer
<i>dup</i>	integer - the MQTT dup flag
<i>packetid</i>	integer - the MQTT packet identifier
<i>count</i>	- number of members in the topicFilters array
<i>topicFilters</i>	- array of topic filter names

Returns

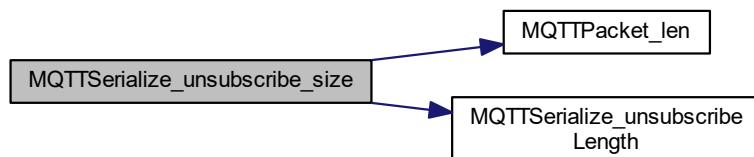
the length of the serialized data. <= 0 indicates error

Here is the caller graph for this function:

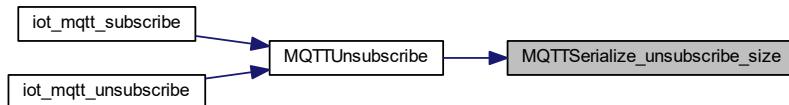


```
5.57.1.3 MQTTSerialize_unsubscribe_size() int MQTTSerialize_unsubscribe_size (
    int count,
    MQTTString topicFilters[] )
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.57.1.4 MQTTSerialize_unsubscribeLength() `int MQTTSerialize_unsubscribeLength (int count, MQTTString topicFilters[])`

Determines the length of the MQTT unsubscribe packet that would be produced using the supplied parameters

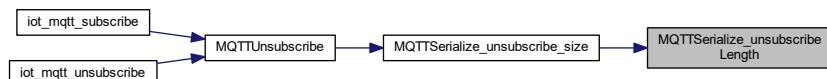
Parameters

<code>count</code>	the number of topic filter strings in <code>topicFilters</code>
<code>topicFilters</code>	the array of topic filter strings to be used in the publish

Returns

the length of buffer needed to contain the serialized version of the packet

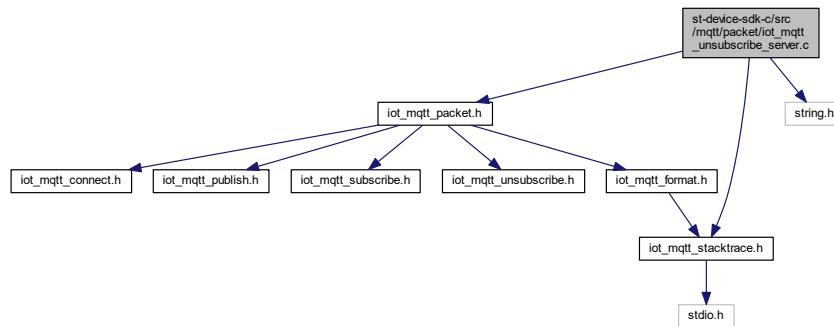
Here is the caller graph for this function:



5.58 st-device-sdk-c/src/mqtt/packet/iot_mqtt_unsubscribe_server.c File Reference

```
#include "iot_mqtt_packet.h"
#include "iot_mqtt_stacktrace.h"
#include <string.h>
```

Include dependency graph for iot_mqtt_unsubscribe_server.c:



Functions

- int [MQTTDeserialize_unsubscribe](#) (unsigned char *dup, unsigned short *packetid, int maxcount, int *count, [MQTTString](#) topicFilters[], unsigned char *buf, int len)
- int [MQTTSerialize_unsuback](#) (unsigned char *buf, int buflen, unsigned short packetid)

5.58.1 Function Documentation

5.58.1.1 [MQTTDeserialize_unsubscribe\(\)](#)

```
int MQTTDeserialize_unsubscribe (
    unsigned char * dup,
    unsigned short * packetid,
    int maxcount,
    int * count,
    MQTTString topicFilters[],
    unsigned char * buf,
    int len )
```

Deserializes the supplied (wire) buffer into unsubscribe data

Parameters

<i>dup</i>	integer returned - the MQTT dup flag
<i>packetid</i>	integer returned - the MQTT packet identifier
<i>maxcount</i>	- the maximum number of members allowed in the topicFilters and requestedQoSs arrays
<i>count</i>	- number of members in the topicFilters and requestedQoSs arrays
<i>topicFilters</i>	- array of topic filter names
<i>buf</i>	the raw buffer data, of the correct length determined by the remaining length field
<i>buflen</i>	the length in bytes of the data in the supplied buffer

Returns

the length of the serialized data. <= 0 indicates error

```
5.58.1.2 MQTTSerialize_unsuback() int MQTTSerialize_unsuback (
    unsigned char * buf,
    int buflen,
    unsigned short packetid )
```

Serializes the supplied unsuback data into the supplied buffer, ready for sending

Parameters

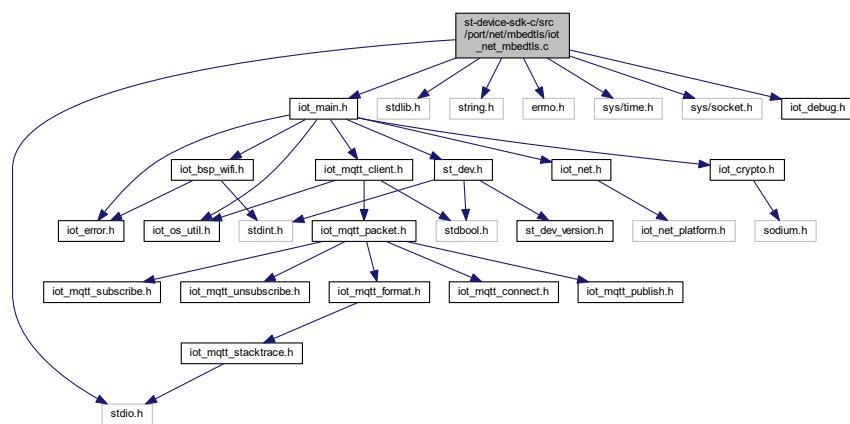
<i>buf</i>	the buffer into which the packet will be serialized
<i>buflen</i>	the length in bytes of the supplied buffer
<i>packetid</i>	integer - the MQTT packet identifier

Returns

the length of the serialized data. <= 0 indicates error

5.59 st-device-sdk-c/src/port/net/mbedtls/iot_net_mbedtls.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <sys/time.h>
#include <sys/socket.h>
#include "iot_main.h"
#include "iot_debug.h"
Include dependency graph for iot_net_mbedtls.c:
```



Functions

- `iot_error_t iot_net_init (iot_net_interface_t *net)`
Initialize the network structure for SSL connection.

5.59.1 Function Documentation

5.59.1.1 iot_net_init() `iot_error_t iot_net_init (iot_net_interface_t * net)`

Initialize the network structure for SSL connection.

Parameters

<code>n</code>	- <code>iot_net_interface</code> structure
----------------	--

Returns

`iot_error_t`

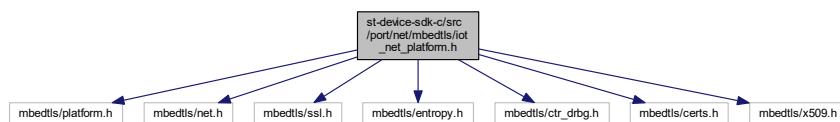
Return values

<code>IOT_ERROR_NONE</code>	success
<code>IOT_ERROR_NET_INVALID_INTERFACE</code>	error

5.60 st-device-sdk-c/src/port/net/mbedtls/iot_net_platform.h File Reference

```
#include "mbedtls/platform.h"
#include "mbedtls/net.h"
#include "mbedtls/ssl.h"
#include "mbedtls/entropy.h"
#include "mbedtls/ctr_drbg.h"
#include "mbedtls/certs.h"
#include "mbedtls/x509.h"

Include dependency graph for iot_net_platform.h:
```



Data Structures

- struct `iot_net_platform_context`
Contains connection context.

Typedefs

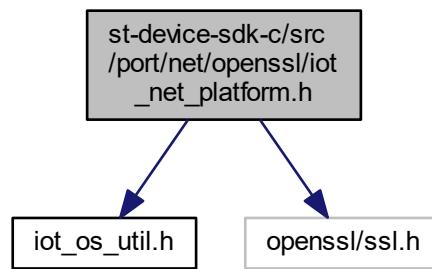
- typedef struct `iot_net_platform_context` `iot_net_platform_context_t`

5.60.1 Typedef Documentation

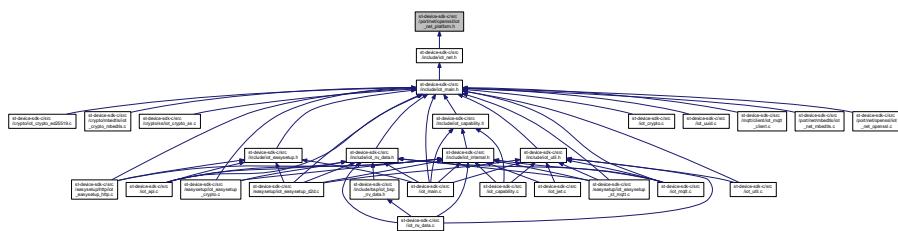
5.60.1.1 `iot_net_platform_context_t` typedef struct iot_net_platform_context iot_net_platform_context_t

5.61 st-device-sdk-c/src/port/net/openssl/iot_net_platform.h File Reference

```
#include "iot_os_util.h"
#include "openssl/ssl.h"
Include dependency graph for iot_net_platform.h:
```



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



Data Structures

- struct `iot_net_platform_context`

Contains connection context.

Typedefs

- `typedef struct iot_net_platform_context iot_net_platform_context_t`

Contains connection context.

5.61.1 Typedef Documentation

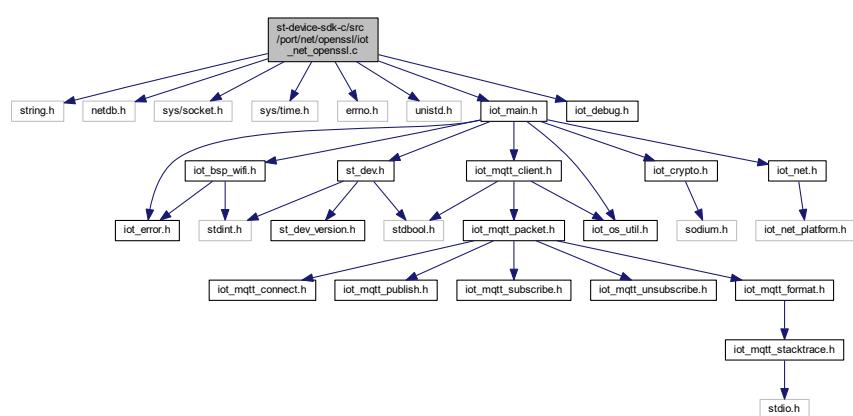
5.61.1.1 iot_net_platform_context_t [typedef struct iot_net_platform_context iot_net_platform_context_t](#)

Contains connection context.

5.62 st-device-sdk-c/src/port/net/openssl/iot_net_openssl.c File Reference

```
#include <string.h>
#include <netdb.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <errno.h>
#include <unistd.h>
#include "iot_main.h"
#include "iot_debug.h"
```

Include dependency graph for iot_net_openssl.c:



Functions

- void [__SSL_library_init](#) (void)
- void [SSL_library_init](#) (void)
- [iot_error_t iot_net_init](#) ([iot_net_interface_t](#) *n)

Initialize the network structure for SSL connection.

5.62.1 Function Documentation

5.62.1.1 __SSL_library_init() [void __SSL_library_init \(void \)](#)

5.62.1.2 iot_net_init() `iot_error_t` `iot_net_init` (
 `iot_net_interface_t` * `net`)

Initialize the network structure for SSL connection.

Parameters

<i>n</i>	- iot_net_interface structure
----------	---

Returns`iot_error_t`**Return values**

<i>IOT_ERROR_NONE</i>	success
<i>IOT_ERROR_NET_INVALID_INTERFACE</i>	error

5.62.1.3 SSL_library_init() `void SSL_library_init (void)`

Index

_IOT_NOTI_TYPE_DEV_DELETED
 iot_main.h, 194

_IOT_NOTI_TYPE_JWT_EXPIRED
 iot_main.h, 194

_IOT_NOTI_TYPE_QUOTA_REACHED
 iot_main.h, 194

_IOT_NOTI_TYPE_RATE_LIMIT
 iot_main.h, 194

_IOT_NOTI_TYPE_UNKNOWN
 iot_main.h, 194

__SSL_library_init
 iot_net_openssl.c, 401

_iot_easysetup_gen_get_payload
 iot_easysetup_http.c, 107

_iot_easysetup_gen_post_payload
 iot_easysetup_http.c, 108

_iot_mqtt_cmd_sub_callback
 iot_mqtt.c, 322

_iot_mqtt_noti_sub_callback
 iot_mqtt.c, 322

_iot_noti_type
 iot_main.h, 194

_iot_nv_read_data
 iot_nv_data.c, 329

_iot_nv_write_data
 iot_nv_data.c, 330

addr
 iot_mac, 51

all
 MQTTConnackFlags, 79
 MQTTConnectFlags, 80

args_str
 iot_cap_cmd_data_t, 13

ARRAY_SIZE
 iot_easysetup_http.c, 107

authmode
 iot_wifi_scan_result_t, 71

bits
 MQTTConnackFlags, 79
 MQTTConnectFlags, 81
 MQTTHeader, 82

broker_port
 iot_cloud_prov_data, 26

broker_url
 iot_cloud_prov_data, 27

bssid
 iot_wifi_conf, 68
 iot_wifi_prov_data, 70
 iot_wifi_scan_result_t, 71

buf
 iot_mqtt_ctx, 53
 MQTTClient, 76

buf_size
 MQTTClient, 76

bufchar
 iot_mqtt_packet.c, 377

byte
 MQTTHeader, 82

ca_cert
 iot_net_connection, 55

ca_cert_len
 iot_net_connection, 55

cacert
 iot_net_platform_context, 60

cap_handle_list
 iot_context, 31

capability
 iot_cap_handle, 21

cert
 iot_net_connection, 55

cert_len
 iot_net_connection, 55

cleansession
 MQTTClient, 76
 MQTTConnectFlags, 81
 MQTTPacket_connectData, 87

cli
 iot_mqtt_ctx, 53

client_ctxt
 iot_context, 31

clientID
 MQTTPacket_connectData, 87

cloud
 iot_device_prov_data, 48

cmd_cb
 iot_cap_cmd_set, 16

cmd_count
 iot_context, 31

cmd_data
 iot_cap_cmd_data_t, 14

cmd_err
 iot_context, 31

cmd_filter
 iot_mqtt_ctx, 53

cmd_list
 iot_cap_handle, 21

cmd_queue
 iot_context, 31

cmd_status
 iot_context, 31

cmd_type
 iot_cap_cmd_set, 16
 iot_command, 28

COLOR_CYAN
 iot_debug.h, 158

COLOR_END
 iot_debug.h, 158

command

iot_cap_cmd_set_list, 18
 command_timeout_ms
 MQTTClient, 76
 component
 iot_cap_handle, 21
 conf
 iot_net_platform_context, 60
 CONFIG_STDK_MQTT_DYNAMIC_BUFFER
 iot_mqtt_client.h, 222
 CONFIG_STDK_MQTT_PING_RETRY
 iot_mqtt_client.h, 222
 CONFIG_STDK_MQTT_PUBLISH_RETRY
 iot_mqtt_client.h, 222
 CONFIG_STDK_MQTT_RECV_BUFFER
 iot_mqtt_client.h, 222
 CONFIG_STDK_MQTT_RECV_CYCLE
 iot_mqtt_client.h, 222
 CONFIG_STDK_MQTT_SEND_BUFFER
 iot_mqtt_client.h, 222
 CONFIG_STDK_MQTT_SEND_CYCLE
 iot_mqtt_client.h, 222
 CONNACK
 iot_mqtt_packet.h, 245
 connack_return_codes
 iot_mqtt_connect.h, 235
 CONNECT
 iot_mqtt_packet.h, 245
 connect
 iot_net_interface, 58
 connection
 iot_net_interface, 58
 context
 iot_net_interface, 58
 count
 noti_data_raw_t::rate_limit, 11
 cstring
 MQTTString, 90
 ctr_drbg
 iot_net_platform_context, 60
 ctx
 iot_cap_handle, 21
 iot_net_platform_context, 60
 curr_otm_feature
 iot_context, 31
 curr_state
 iot_context, 31
 curve
 iot_crypto_ed25519_keypair, 39
 cycle
 iot_mqtt_client.c, 352

 data
 MQTTLenString, 83
 DefaultClient
 iot_mqtt_client.h, 223
 defaultMessageHandler
 MQTTClient, 76
 defaultUserData
 MQTTClient, 76

 deliverMessage
 iot_mqtt_client.c, 352
 devconf
 iot_context, 32
 device_info
 iot_context, 32
 device_onboarding_id
 iot_devconf_prov_data, 45
 device_type
 iot_devconf_prov_data, 45
 deviceld
 iot_registered_data, 65
 DISCONNECT
 iot_mqtt_packet.h, 245
 disconnect
 iot_net_interface, 58
 DLLExport
 iot_mqtt_client.h, 223
 iot_mqtt_connect.h, 235
 iot_mqtt_packet.h, 244
 iot_mqtt_publish.h, 254
 iot_mqtt_subscribe.h, 260
 iot_mqtt_unsubscribe.h, 265
 DLLImport
 iot_mqtt_client.h, 223
 iot_mqtt_connect.h, 235
 iot_mqtt_packet.h, 245
 iot_mqtt_publish.h, 254
 iot_mqtt_subscribe.h, 260
 iot_mqtt_unsubscribe.h, 265
 domain
 url_parse_t, 94
 dup
 MQTTHeader, 82
 MQTTMessage, 84

 easysetup_req_queue
 iot_context, 32
 easysetup_resp_queue
 iot_context, 32
 EASYSETUP_TIMEOUT_MS
 iot_main.c, 319
 ENTER
 iot_debug.h, 158
 entropy
 iot_net_platform_context, 60
 err
 iot_easysetup_payload, 50
 errors
 iot_mqtt_packet.h, 245
 ES_CONFIRM_MAX_DELAY
 iot_easysetup_d2d.c, 114
 es_crypto_cipher_info
 iot_context, 32
 es_httpd_handle
 iot_context, 32
 evt_type
 iot_cap_evt_data_t, 19
 evt_unit

iot_cap_evt_data_t, 19
evt_value
 iot_cap_evt_data_t, 19

fd
 iot_bsp_fs_handle_t, 12

filename
 iot_bsp_fs_handle_t, 12

firmware_version
 iot_device_info, 47

fn
 iot_crypto_pk_context, 41

fp
 MQTTClient::MessageHandlers, 74

freq
 iot_wifi_scan_result_t, 72

fs_close_custom
 iot_easysetup_http.c, 108

fs_open_custom
 iot_easysetup_http.c, 109

FS_READONLY
 iot_bsp_fs.h, 121

FS_READWRITE
 iot_bsp_fs.h, 121

fs_state_free
 iot_easysetup_http.c, 109

fs_state_init
 iot_easysetup_http.c, 109

FUNC_ENTRY
 iot_mqtt_stacktrace.h, 258

FUNC_ENTRY_MAX
 iot_mqtt_stacktrace.h, 259

FUNC_ENTRY_MED
 iot_mqtt_stacktrace.h, 259

FUNC_ENTRY_NOLOG
 iot_mqtt_stacktrace.h, 259

FUNC_EXIT
 iot_mqtt_stacktrace.h, 259

FUNC_EXIT_MAX
 iot_mqtt_stacktrace.h, 259

FUNC_EXIT_MAX_RC
 iot_mqtt_stacktrace.h, 259

FUNC_EXIT_MED
 iot_mqtt_stacktrace.h, 259

FUNC_EXIT_MED_RC
 iot_mqtt_stacktrace.h, 259

FUNC_EXIT_NOLOG
 iot_mqtt_stacktrace.h, 259

FUNC_EXIT_RC
 iot_mqtt_stacktrace.h, 259

get_cgi_cmds
 iot_easysetup_http.c, 111

getfn
 MQTTTransport, 92

getLenStringLen
 iot_mqtt_packet.c, 377

grantedQoS
 MQTTSubackData, 91

handle
 iot_cap_handle_list, 22

HASH_SIZE
 iot_easysetup_d2d.c, 114

hash_token
 iot_crypto_ecdh_params, 37

hash_token_len
 iot_crypto_ecdh_params, 37

hashed_sn
 iot_devconf_prov_data, 45

HIT
 iot_debug.h, 158

httpd_cgi_handler
 iot_easysetup_http.c, 109

httpd_post_begin
 iot_easysetup_http.c, 109

httpd_post_finished
 iot_easysetup_http.c, 109

httpd_post_receive_data
 iot_easysetup_http.c, 110

id
 iot_uuid, 67

MQTTMessage, 84

info
 iot_crypto_pk_context, 41

init_cb
 iot_cap_handle, 21

init_usr_data
 iot_cap_handle, 21

int
 MQTTConnectFlags, 81

integer
 iot_cap_val_t, 25

iot_api.c
 iot_api_device_info_load, 297

 iot_api_device_info_mem_free, 298

 iot_api_onboarding_config_load, 298

 iot_api_onboarding_config_mem_free, 299

 iot_api_prov_data_mem_free, 299

 iot_api_read_device_identity, 300

 iot_command_send, 301

 iot_device_cleanup, 302

 iot_easysetup_request, 303

 iot_get_time_in_ms, 304

 iot_get_time_in_sec, 304

 iot_state_update, 304

 iot_api_device_info_load
 iot_api.c, 297

 iot_internal.h, 174

 iot_api_device_info_mem_free
 iot_api.c, 298

 iot_internal.h, 175

 iot_api_onboarding_config_load
 iot_api.c, 298

 iot_internal.h, 175

 iot_api_onboarding_config_mem_free
 iot_api.c, 299

 iot_internal.h, 176

iot_api_prov_data_mem_free
 iot_api.c, 299
 iot_internal.h, 176

iot_api_read_device_identity
 iot_api.c, 300
 iot_internal.h, 177

iot_bsp_debug
 iot_bsp_debug.h, 119
 iot_debug.h, 161

iot_bsp_debug.h
 iot_bsp_debug, 119
 iot_bsp_debug_check_heap, 119

iot_bsp_debug_check_heap
 iot_bsp_debug.h, 119
 iot_debug.h, 161

iot_bsp_fs.h
 FS_READONLY, 121
 FS_READWRITE, 121
 iot_bsp_fs_close, 121
 iot_bsp_fs_deinit, 122
 iot_bsp_fs_init, 122
 iot_bsp_fs_open, 123
 iot_bsp_fs_open_from_stnv, 124
 iot_bsp_fs_open_mode_t, 121
 iot_bsp_fs_read, 124
 iot_bsp_fs_remove, 125
 iot_bsp_fs_write, 126

iot_bsp_fs_close
 iot_bsp_fs.h, 121

iot_bsp_fs_deinit
 iot_bsp_fs.h, 122

iot_bsp_fs_handle_t, 12
 fd, 12
 filename, 12

iot_bsp_fs_init
 iot_bsp_fs.h, 122

iot_bsp_fs_open
 iot_bsp_fs.h, 123

iot_bsp_fs_open_from_stnv
 iot_bsp_fs.h, 124

iot_bsp_fs_open_mode_t
 iot_bsp_fs.h, 121

iot_bsp_fs_read
 iot_bsp_fs.h, 124

iot_bsp_fs_remove
 iot_bsp_fs.h, 125

iot_bsp_fs_write
 iot_bsp_fs.h, 126

iot_bsp_nv_data.h
 iot_bsp_nv_get_data_path, 127

iot_bsp_nv_get_data_path
 iot_bsp_nv_data.h, 127

iot_bsp_random
 iot_bsp_random.h, 129

iot_bsp_random.h
 iot_bsp_random, 129

iot_bsp_system.h
 iot_bsp_system_get_time_in_sec, 131

iot_bsp_system_get_uniqueid, 131
 iot_bsp_system_poweroff, 131
 iot_bsp_system_reboot, 132
 iot_bsp_system_set_time_in_sec, 132
 IOT_POWEROFF, 130
 IOT_REBOOT, 130

iot_bsp_system_get_time_in_sec
 iot_bsp_system.h, 131

iot_bsp_system_get_uniqueid
 iot_bsp_system.h, 131

iot_bsp_system_poweroff
 iot_bsp_system.h, 131

iot_bsp_system_reboot
 iot_bsp_system.h, 132

iot_bsp_system_set_time_in_sec
 iot_bsp_system.h, 132

iot_bsp_wifi.h
 iot_bsp_wifi_get_freq, 135
 iot_bsp_wifi_get_mac, 135
 iot_bsp_wifi_get_scan_result, 136
 iot_bsp_wifi_init, 136
 iot_bsp_wifi_set_mode, 137
 IOT_SOFT_AP_CHANNEL, 134
 IOT_WIFI_AUTH_MAX, 134
 iot_wifi_auth_mode_t, 134
 IOT_WIFI_AUTH_OPEN, 134
 IOT_WIFI_AUTH_WEP, 134
 IOT_WIFI_AUTH_WPA_ENTERPRISE, 134
 IOT_WIFI_AUTH_WPA2_PSK, 134
 IOT_WIFI_AUTH_WPA_PSK, 134
 IOT_WIFI_AUTH_WPA_WPA2_PSK, 134
 IOT_WIFI_CMD_TIMEOUT, 134
 IOT_WIFI_FREQ_2_4G_5G_BOTH, 135
 IOT_WIFI_FREQ_2_4G_ONLY, 135
 IOT_WIFI_FREQ_5G_ONLY, 135
 iot_wifi_freq_t, 134
 IOT_WIFI_MAX_BSSID_LEN, 134
 IOT_WIFI_MAX_PASS_LEN, 134
 IOT_WIFI_MAX_SCAN_RESULT, 134
 IOT_WIFI_MAX_SSID_LEN, 134
 IOT_WIFI_MODE_OFF, 135
 IOT_WIFI_MODE_P2P, 135
 IOT_WIFI_MODE_SCAN, 135
 IOT_WIFI_MODE_SOFTAP, 135
 IOT_WIFI_MODE_STATION, 135
 iot_wifi_mode_t, 135
 IOT_WIFI_MODE_UNDEFINED, 135

iot_bsp_wifi_get_freq
 iot_bsp_wifi.h, 135

iot_bsp_wifi_get_mac
 iot_bsp_wifi.h, 135

iot_bsp_wifi_get_scan_result
 iot_bsp_wifi.h, 136

iot_bsp_wifi_init
 iot_bsp_wifi.h, 136

iot_bsp_wifi_set_mode
 iot_bsp_wifi.h, 137

IOT_BUF_RX_SIZE

iot_main.h, 193
IOT_BUF_TX_SIZE
 iot_main.h, 193
iot_cap_call_init_cb
 iot_capability.c, 306
 iot_internal.h, 178
iot_cap_cmd_data_t, 13
 args_str, 13
 cmd_data, 14
 num_args, 14
iot_cap_cmd_set, 14
 cmd_cb, 16
 cmd_type, 16
 usr_data, 16
iot_cap_cmd_set_list, 16
 command, 18
 next, 18
iot_cap_cmd_set_list_t
 iot_capability.h, 139
iot_cap_cmd_set_t
 iot_capability.h, 139
iot_cap_evt_data_t, 18
 evt_type, 19
 evt_unit, 19
 evt_value, 19
 iot_capability.h, 139
IOT_CAP_HANDLE
 st_dev.h, 284
iot_cap_handle, 20
 capability, 21
 cmd_list, 21
 component, 21
 ctx, 21
 init_cb, 21
 init_usr_data, 21
iot_cap_handle_list, 22
 handle, 22
 next, 23
iot_cap_handle_list_t
 iot_main.h, 194
iot_cap_sub_cb
 iot_capability.c, 307
 iot_internal.h, 178
iot_cap_unit_t, 23
 string, 23
 type, 24
iot_cap_unit_type
 iot_capability.h, 139
IOT_CAP_UNIT_TYPE_STRING
 iot_capability.h, 139
IOT_CAP_UNIT_TYPE_UNUSED
 iot_capability.h, 139
iot_cap_val_t, 24
 integer, 25
 number, 25
 str_num, 25
 string, 25
 strings, 25
 type, 25
iot_cap_val_type
 st_dev.h, 286
IOT_CAP_VAL_TYPE_INT_OR_NUM
 st_dev.h, 286
IOT_CAP_VAL_TYPE_INTEGER
 st_dev.h, 286
IOT_CAP_VAL_TYPE_NUMBER
 st_dev.h, 286
IOT_CAP_VAL_TYPE_STR_ARRAY
 st_dev.h, 286
IOT_CAP_VAL_TYPE_STRING
 st_dev.h, 286
iot_cap_val_type_t
 st_dev.h, 284
IOT_CAP_VAL_TYPE_UNKNOWN
 st_dev.h, 286
iot_capability.c
 iot_cap_call_init_cb, 306
 iot_cap_sub_cb, 307
 iot_noti_sub_cb, 307
 MAX_SQNUM, 306
 st_cap_attr_create_int, 308
 st_cap_attr_create_number, 308
 st_cap_attr_create_string, 309
 st_cap_attr_create_string_array, 309
 st_cap_attr_free, 310
 st_cap_attr_send, 310
 st_cap_cmd_set_cb, 311
 st_cap_handle_init, 311
 st_conn_set_noti_cb, 313
iot_capability.h
 iot_cap_cmd_set_list_t, 139
 iot_cap_cmd_set_t, 139
 iot_cap_evt_data_t, 139
 iot_cap_unit_type, 139
 IOT_CAP_UNIT_TYPE_STRING, 139
 IOT_CAP_UNIT_TYPE_UNUSED, 139
iot_cloud_prov_data, 26
 broker_port, 26
 broker_url, 27
 label, 27
 location_id, 27
 room_id, 27
IOT_CMD_STATE_HANDLE
 iot_main.h, 195
iot_command, 27
 cmd_type, 28
 param, 28
IOT_COMMAND_CHECK_CLOUD_STATE
 iot_main.h, 195
IOT_COMMAND_CHECK_PROV_STATUS
 iot_main.h, 195
IOT_COMMAND_CLOUD_CONNECTING
 iot_main.h, 195
IOT_COMMAND_CLOUD_REGISTERED
 iot_main.h, 195
IOT_COMMAND_CLOUD_REGISTERING

iot_main.h, 195
 IOT_COMMAND_NETWORK_MODE
 iot_main.h, 195
 IOT_COMMAND_NOTIFICATION_RECEIVED
 iot_main.h, 195
 IOT_COMMAND_READY_TO_CTL
 iot_main.h, 195
 IOT_COMMAND_SELF_CLEANUP
 iot_main.h, 195
 iot_command_send
 iot_api.c, 301
 iot_internal.h, 179
 iot_command_type
 iot_main.h, 194
 IOT_COMMAND_TYPE_MAX
 iot_main.h, 195
 iot_connect_type
 iot_main.h, 195
 IOT_CONNECT_TYPE_COMMUNICATION
 iot_main.h, 195
 IOT_CONNECT_TYPE_REGISTRATION
 iot_main.h, 195
 iot_context, 28
 cap_handle_list, 31
 client_ctx, 31
 cmd_count, 31
 cmd_err, 31
 cmd_queue, 31
 cmd_status, 31
 curr_otm_feature, 31
 curr_state, 31
 devconf, 32
 device_info, 32
 easysetup_req_queue, 32
 easysetup_resp_queue, 32
 es_crypto_cipher_info, 32
 es_httpd_handle, 32
 iot_events, 32
 iot_reg_data, 32
 lookup_id, 33
 noti_cb, 33
 noti_usr_data, 33
 pin, 33
 prov_data, 33
 pub_queue, 33
 reged_cli, 33
 reported_stat, 33
 req_state, 34
 scan_num, 34
 scan_result, 34
 state_timer, 34
 status_cb, 34
 status_maps, 34
 status_usr_data, 34
 usr_events, 34
 iot_crypto.c
 iot_crypto_pk_free, 314
 iot_crypto_pk_init, 314
 iot_crypto_pk_sign, 315
 iot_crypto_pk_verify, 315
 iot_crypto.h
 IOT_CRYPTO_ALIGN_B64_LEN, 143
 iot_crypto_base64_decode, 147
 iot_crypto_base64_decode_urlsafe, 148
 iot_crypto_base64_encode, 148
 iot_crypto_base64_encode_urlsafe, 149
 IOT_CRYPTO_CAL_B64_LEN, 143
 iot_crypto_cipher_aes, 149
 IOT_CRYPTO_CIPHER_AES256, 147
 IOT_CRYPTO_CIPHER_DECRYPT, 147
 IOT_CRYPTO_CIPHER_ENCRYPT, 147
 iot_crypto_cipher_get_align_size, 150
 iot_crypto_cipher_info_t, 146
 iot_crypto_cipher_mode_t, 146
 iot_crypto_cipher_type_t, 147
 IOT_CRYPTO_CIPHER_UNKNOWN, 147
 iot_crypto_ecdh_gen_master_secret, 151
 iot_crypto_ecdh_params_t, 146
 iot_crypto_ed25519_convert_keypair, 151
 iot_crypto_ed25519_convert_pubkey, 152
 iot_crypto_ed25519_convert_seckey, 152
 iot_crypto_ed25519_free_keypair, 152
 iot_crypto_ed25519_init_keypair, 153
 iot_crypto_ed25519_keypair_t, 146
 IOT_CRYPTO_ED25519_LEN, 143
 IOT_CRYPTO_IV_LEN, 143
 iot_crypto_pk_context_t, 146
 IOT_CRYPTO_PK_ED25519, 147
 iot_crypto_pk_free, 153
 iot_crypto_pk_funcs_t, 146
 iot_crypto_pk_info_t, 146
 iot_crypto_pk_init, 153
 IOT_CRYPTO_PK_RSA, 147
 iot_crypto_pk_sign, 154
 IOT_CRYPTO_PK_TYPE_ED25519, 143
 IOT_CRYPTO_PK_TYPE_RSA, 143
 iot_crypto_pk_type_t, 147
 IOT_CRYPTO_PK_UNKNOWN, 147
 iot_crypto_pk_verify, 154
 IOT_CRYPTO_SECRET_LEN, 143
 iot_crypto_sha256, 155
 IOT_CRYPTO_SHA256_LEN, 143
 IOT_CRYPTO_SIGNATURE_LEN, 144
 iot_crypto_ss_decrypt, 156
 iot_crypto_ss_encrypt, 156
 IOT_ERROR_CRYPTO_BASE64, 144
 IOT_ERROR_CRYPTO_BASE64_URLSAFE, 144
 IOT_ERROR_CRYPTO_CIPHER, 144
 IOT_ERROR_CRYPTO_CIPHER_ALIGN, 144
 IOT_ERROR_CRYPTO_CIPHER_IVLEN, 144
 IOT_ERROR_CRYPTO_CIPHER_KEYLEN, 144
 IOT_ERROR_CRYPTO_CIPHER_OUTSIZE, 144
 IOT_ERROR_CRYPTO_CIPHER_UNKNOWN_MODE, 144
 IOT_ERROR_CRYPTO_CIPHER_UNKNOWN_TYPE, 144

IOT_ERROR_CRYPTO_ED_KEY_CONVERT, 145
IOT_ERROR_CRYPTO_PK_ECDH, 145
IOT_ERROR_CRYPTO_PK_INVALID_ARG, 145
IOT_ERROR_CRYPTO_PK_INVALID_CTX, 145
IOT_ERROR_CRYPTO_PK_INVALID_KEYLEN, 145
IOT_ERROR_CRYPTO_PK_NULL_FUNC, 145
IOT_ERROR_CRYPTO_PK_PARSEKEY, 145
IOT_ERROR_CRYPTO_PK_SIGN, 145
IOT_ERROR_CRYPTO_PK_UNKNOWN_KEYTYPE, iot_crypto_ed25519.c, 145
IOT_ERROR_CRYPTO_PK_VERIFY, 145
IOT_ERROR_CRYPTO_SHA256, 146
IOT_ERROR_CRYPTO_SS_KDF, 146
IOT_CRYPTO_ALIGN_B64_LEN
 iot_crypto.h, 143
iot_crypto_base64_decode
 iot_crypto.h, 147
 iot_crypto_mbedtls.c, 99
iot_crypto_base64_decode_urlsafe
 iot_crypto.h, 148
 iot_crypto_mbedtls.c, 99
iot_crypto_base64_encode
 iot_crypto.h, 148
 iot_crypto_mbedtls.c, 100
iot_crypto_base64_encode_urlsafe
 iot_crypto.h, 149
 iot_crypto_mbedtls.c, 100
IOT_CRYPTO_CAL_B64_LEN
 iot_crypto.h, 143
iot_crypto_cipher_aes
 iot_crypto.h, 149
 iot_crypto_mbedtls.c, 101
IOT_CRYPTO_CIPHER_AES256
 iot_crypto.h, 147
IOT_CRYPTO_CIPHER_DECRYPT
 iot_crypto.h, 147
IOT_CRYPTO_CIPHER_ENCRYPT
 iot_crypto.h, 147
iot_crypto_cipher_get_align_size
 iot_crypto.h, 150
 iot_crypto_mbedtls.c, 102
iot_crypto_cipher_info, 35
 iv, 36
 iv_len, 36
 key, 36
 key_len, 36
 mode, 36
 type, 36
iot_crypto_cipher_info_t
 iot_crypto.h, 146
iot_crypto_cipher_mode_t
 iot_crypto.h, 146
iot_crypto_cipher_type_t
 iot_crypto.h, 147
IOT_CRYPTO_CIPHER_UNKNOWN
 iot_crypto.h, 147
iot_crypto_ecdh_gen_master_secret
 iot_crypto.h, 147
 iot_crypto.h, 151
 iot_crypto_mbedtls.c, 103
iot_crypto_ecdh_params, 37
 hash_token, 37
 hash_token_len, 37
 s_pubkey, 38
 t_seckey, 38
iot_crypto_ecdh_params_t
 iot_crypto.h, 146
 iot_crypto_ed25519_convert_keypair, 96
 iot_crypto_ed25519_convert_pubkey, 96
 iot_crypto_ed25519_convert_seckey, 97
 iot_crypto_ed25519_free_keypair, 97
 iot_crypto_ed25519_init_keypair, 97
iot_crypto_ed25519_convert_keypair
 iot_crypto.h, 151
 iot_crypto_ed25519.c, 96
iot_crypto_ed25519_convert_pubkey
 iot_crypto.h, 152
 iot_crypto_ed25519.c, 96
iot_crypto_ed25519_convert_seckey
 iot_crypto.h, 152
 iot_crypto_ed25519.c, 97
iot_crypto_ed25519_free_keypair
 iot_crypto.h, 152
 iot_crypto_ed25519.c, 97
iot_crypto_ed25519_init_keypair
 iot_crypto.h, 153
 iot_crypto_ed25519.c, 97
iot_crypto_ed25519_keypair, 38
 curve, 39
 sign, 39
iot_crypto_ed25519_keypair_t
 iot_crypto.h, 146
IOT_CRYPTO_ED25519_LEN
 iot_crypto.h, 143
IOT_CRYPTO_IV_LEN
 iot_crypto.h, 143
iot_crypto_keypair, 39
 pubkey, 40
 seckey, 40
iot_crypto_mbedtls.c
 iot_crypto_base64_decode, 99
 iot_crypto_base64_decode_urlsafe, 99
 iot_crypto_base64_encode, 100
 iot_crypto_base64_encode_urlsafe, 100
 iot_crypto_cipher_aes, 101
 iot_crypto_cipher_get_align_size, 102
 iot_crypto_ecdh_gen_master_secret, 103
 iot_crypto_sha256, 103
iot_crypto_pk_context, 40
 fn, 41
 info, 41
iot_crypto_pk_context_t
 iot_crypto.h, 146
IOT_CRYPTO_PK_ED25519
 iot_crypto.h, 147

iot_crypto_pk_free
 iot_crypto.c, 314
 iot_crypto.h, 153

iot_crypto_pk_funcs, 41
 name, 42
 sign, 42
 verify, 42

iot_crypto_pk_funcs_t
 iot_crypto.h, 146

iot_crypto_pk_info, 43
 pubkey, 43
 pubkey_len, 43
 seckey, 44
 seckey_len, 44
 type, 44

iot_crypto_pk_info_t
 iot_crypto.h, 146

iot_crypto_pk_init
 iot_crypto.c, 314
 iot_crypto.h, 153

IOT_CRYPTO_PK_RSA
 iot_crypto.h, 147

iot_crypto_pk_sign
 iot_crypto.c, 315
 iot_crypto.h, 154

IOT_CRYPTO_PK_TYPE_ED25519
 iot_crypto.h, 143

IOT_CRYPTO_PK_TYPE_RSA
 iot_crypto.h, 143

iot_crypto_pk_type_t
 iot_crypto.h, 147

IOT_CRYPTO_PK_UNKNOWN
 iot_crypto.h, 147

iot_crypto_pk_verify
 iot_crypto.c, 315
 iot_crypto.h, 154

IOT_CRYPTO_SECRET_LEN
 iot_crypto.h, 143

iot_crypto_sha256
 iot_crypto.h, 155
 iot_crypto_mbedtls.c, 103

IOT_CRYPTO_SHA256_LEN
 iot_crypto.h, 143

IOT_CRYPTO_SIGNATURE_LEN
 iot_crypto.h, 144

iot_crypto_ss.c
 iot_crypto_ss_decrypt, 104
 iot_crypto_ss_encrypt, 105

iot_crypto_ss_decrypt
 iot_crypto.h, 156
 iot_crypto_ss.c, 104

iot_crypto_ss_encrypt
 iot_crypto.h, 156
 iot_crypto_ss.c, 105

IOT_CTX
 st_dev.h, 285

iot_ctx
 iot_mqtt_ctx, 53

IOT_DEBUG
 iot_debug.h, 158

iot_debug.h
 COLOR_CYAN, 158
 COLOR_END, 158
 ENTER, 158
 HIT, 158
 iot_bsp_debug, 161
 iot_bsp_debug_check_heap, 161
 IOT_DEBUG, 158
 IOT_DEBUG_CHECK, 159
 IOT_DEBUG_LEVEL_DEBUG, 160
 IOT_DEBUG_LEVEL_ERROR, 160
 IOT_DEBUG_LEVEL_INFO, 160
 IOT_DEBUG_LEVEL_MAX, 160
 IOT_DEBUG_LEVEL_NONE, 160
 iot_debug_level_t, 160
 IOT_DEBUG_LEVEL_WARN, 160
 IOT_DEBUG_PREFIX, 159
 IOT_ERROR, 159
 IOT_ERROR_CHECK, 159
 IOT_INFO, 159
 IOT_MEM_CHECK, 160
 IOT_WARN, 160
 IOT_WARN_CHECK, 160
 LEAVE, 160

IOT_DEBUG_CHECK
 iot_debug.h, 159

IOT_DEBUG_LEVEL_DEBUG
 iot_debug.h, 160

IOT_DEBUG_LEVEL_ERROR
 iot_debug.h, 160

IOT_DEBUG_LEVEL_INFO
 iot_debug.h, 160

IOT_DEBUG_LEVEL_MAX
 iot_debug.h, 160

IOT_DEBUG_LEVEL_NONE
 iot_debug.h, 160

iot_debug_level_t
 iot_debug.h, 160

IOT_DEBUG_LEVEL_WARN
 iot_debug.h, 160

IOT_DEBUG_PREFIX
 iot_debug.h, 159

IOT_DEFAULT_TIMEOUT
 iot_internal.h, 173

IOT_DELAY
 iot_os_util.h, 269

iot_devconf_prov_data, 44
 device_onboarding_id, 45
 device_type, 45
 hashed_sn, 45
 mnid, 45
 ownership_validation_type, 46
 pk_type, 46
 setupid, 46
 vid, 46

iot_device_cleanup

iot_api.c, 302
iot_internal.h, 180
iot_device_info, 46
firmware_version, 47
iot_device_prov_data, 47
cloud, 48
wifi, 49
iot_easystartup.h
iot_easystartup_create_ssid, 167
iot_easystartup_deinit, 168
iot_easystartup_init, 168
iot_easystartup_request_handler, 168
IOT_ERROR_EASYSETUP_400_BASE, 163
IOT_ERROR_EASYSETUP_500_BASE, 163
IOT_ERROR_EASYSETUP_CERTIFICATE_NOT_FOUND, 163
IOT_ERROR_EASYSETUP_CONFIRM_DENIED, 163
IOT_ERROR_EASYSETUP_CONFIRM_TIMEOUT, 163
IOT_ERROR_EASYSETUP_INTERNAL_CRITICAL_ERROR, 163
IOT_ERROR_EASYSETUP_INTERNAL_SERVER_ERROR, 163
IOT_ERROR_EASYSETUP_INTERNAL_WARNING, 163
IOT_ERROR_EASYSETUP_INVALID_BROKER_URL, 164
IOT_ERROR_EASYSETUP_INVALID_PAYLOAD, 164
IOT_ERROR_EASYSETUP_INVALID_PIN, 164
IOT_ERROR_EASYSETUP_INVALID_QR, 164
IOT_ERROR_EASYSETUP_INVALID_REQUEST, 164
IOT_ERROR_EASYSETUP_INVALID_SEQUENCE, 164
IOT_ERROR_EASYSETUP_MAX, 164
IOT_ERROR_EASYSETUP_NOT_SUPPORTED, 164
IOT_ERROR_EASYSETUP_PIN_NOT_FOUND, 164
IOT_ERROR_EASYSETUP RAND_DECODE_ERROR, 164
IOT_ERROR_EASYSETUP_RPK_NOT_FOUND, 165
IOT_ERROR_EASYSETUP_SERIAL_NOT_FOUND, 165
IOT_ERROR_EASYSETUP_SHARED_KEY_CREATION_FAILED, 165
IOT_ERROR_EASYSETUP_SPUB_DECODE_ERROR, 165
IOT_ERROR_EASYSETUP_WIFI_SCANLIST_NOT_FOUND, 165
IOT_ES_URI_GET_DEVICEINFO, 165
IOT_ES_URI_GET_LOGS_DUMP, 165
IOT_ES_URI_GET_LOGS_SYSTEMINFO, 165
IOT_ES_URI_GET_POST_RESPONSE, 165
IOT_ES_URI_GET_WIFISCANINFO, 165
IOT_ES_URI_POST_CONFIRM, 166
IOT_ES_URI_POST_CONFIRMINFO, 166
IOT_ES_URI_POST_KEYINFO, 166
IOT_ES_URI_POST_LOGS, 166
IOT_ES_URI_POST_SETUPCOMPLETE, 166
IOT_ES_URI_POST_WIFIPROVISIONINGINFO, 166
IOT_OVF_TYPE_BUTTON, 166
IOT_OVF_TYPE JUSTWORKS, 166
IOT_OVF_TYPE_PIN, 166
IOT_OVF_TYPE_QR, 166
OVF_BIT_BUTTON, 167
OVF_BIT JUSTWORKS, 167
OVF_BIT_MAX_FEATURE, 167
OVF_BIT_PIN, 167
OVF_BIT_QR, 167
ownership_validation_feature, 166
iot_easystartup_create_ssid
iot_easystartup.h, 167
iot_easystartup_d2d.c, 114
iot_easystartup_crypto.c
iot_es_crypto_free_pk, 112
iot_es_crypto_init_pk, 112
iot_es_crypto_load_pk, 112
ES_CONFIRM_MAX_DELAY, 114
HASH_SIZE, 114
iot_easystartup_create_ssid, 114
iot_easystartup_request_handler, 115
JSON_H, 114
MAC_ADDR_BUFFER_SIZE, 114
PIN_SIZE, 114
st_conn_ownership_confirm, 116
URL_BUFFER_SIZE, 114
WIFIINFO_BUFFER_SIZE, 114
iot_easystartup_deinit
iot_easystartup.h, 168
iot_easystartup_http.c, 110
iot_easystartup_http.c
_iot_easystartup_gen_get_payload, 107
_iot_easystartup_gen_post_payload, 108
ARRAY_SIZE, 107
fs_close_custom, 108
fs_open_custom, 109
fs_state_free, 109
fs_state_init, 109
get_cgi_cmds, 111
httpd_cgi_handler, 109
httpd_post_begin, 109
httpd_post_finished, 109
httpd_post_receive_data, 110
MD_easystartup_deinit, 110
iot_easystartup_init, 110
LWIP_HTTPD_POST_MAX_PAYLOAD_LEN, 107
MAX_PAYLOAD_LENGTH, 107
post_cgi_cmds, 111
iot_easystartup_init
iot_easystartup.h, 168

iot_easysetup_http.c, 110
 iot_easysetup_payload, 49
 err, 50
 payload, 50
 step, 50
 iot_easysetup_request
 iot_api.c, 303
 iot_internal.h, 181
 iot_easysetup_request_handler
 iot_easysetup.h, 168
 iot_easysetup_d2d.c, 115
 iot_easysetup_st_mqtt.c
 iot_es_connect, 117
 iot_es_disconnect, 117
 iot_easysetup_step
 iot_main.h, 195
 IOT_EASYSETUP_STEP_CONFIRM
 iot_main.h, 195
 IOT_EASYSETUP_STEP_CONFIRMINFO
 iot_main.h, 195
 IOT_EASYSETUP_STEP_DEVICEINFO
 iot_main.h, 195
 IOT_EASYSETUP_STEP_KEYINFO
 iot_main.h, 195
 IOT_EASYSETUP_STEP_LOG_CREATE_DUMP
 iot_main.h, 195
 IOT_EASYSETUP_STEP_LOG_GET_DUMP
 iot_main.h, 195
 IOT_EASYSETUP_STEP_LOG_SYSTEMINFO
 iot_main.h, 195
 IOT_EASYSETUP_STEP_SETUPCOMPLETE
 iot_main.h, 195
 IOT_EASYSETUP_STEP_WIFIPROVISIONINGINFO
 iot_main.h, 195
 IOT_EASYSETUP_STEP_WIFISCANINFO
 iot_main.h, 195
 IOT_ERROR
 iot_debug.h, 159
 iot_error.h
 IOT_ERROR_BAD_REQ, 170
 IOT_ERROR_CONNECT_FAIL, 170
 IOT_ERROR_CRYPTO_BASE, 170
 IOT_ERROR_DEINIT_FAIL, 170
 IOT_ERROR_EASYSETUP_BASE, 170
 IOT_ERROR_FS_CLOSE_FAIL, 170
 IOT_ERROR_FS_DECRYPT_FAIL, 170
 IOT_ERROR_FS_ENCRYPT_FAIL, 170
 IOT_ERROR_FS_ENCRYPT_INIT, 170
 IOT_ERROR_FS_NO_FILE, 170
 IOT_ERROR_FS_OPEN_FAIL, 170
 IOT_ERROR_FS_READ_FAIL, 170
 IOT_ERROR_FS_REMOVE_FAIL, 170
 IOT_ERROR_FS_WRITE_FAIL, 170
 IOT_ERROR_INIT_FAIL, 170
 IOT_ERROR_INVALID_ARGS, 170
 IOT_ERROR_JWT_BASE, 170
 IOT_ERROR_MEM_ALLOC, 170
 IOT_ERROR_MQTT_CONNECT_FAIL, 170
 IOT_ERROR_MQTT_NETCONN_FAIL, 170
 IOT_ERROR_MQTT_PUBLISH_FAIL, 170
 IOT_ERROR_MQTT_REJECT_CONNECT, 170
 IOT_ERROR_MQTT_SERVER_UNAVAIL, 170
 IOT_ERROR_NET_CONNECT, 170
 IOT_ERROR_NET_INVALID_INTERFACE, 170
 IOT_ERROR_NONE, 170
 IOT_ERROR_NOT_IMPLEMENTED, 170
 IOT_ERROR_NV_DATA_ERROR, 170
 IOT_ERROR_NV_DATA_NOT_EXIST, 170
 IOT_ERROR_PROV_FAIL, 170
 IOT_ERROR_READ_FAIL, 170
 IOT_ERROR_REG_UPDATED, 170
 iot_error_t, 170
 IOT_ERROR_TIMEOUT, 170
 IOT_ERROR_UNINITIALIZED, 170
 IOT_ERROR_UUID_FAIL, 170
 IOT_ERROR_WRITE_FAIL, 170
 IOT_ERROR_BAD_REQ
 iot_error.h, 170
 IOT_ERROR_CHECK
 iot_debug.h, 159
 IOT_ERROR_CONNECT_FAIL
 iot_error.h, 170
 IOT_ERROR_CRYPTO_BASE
 iot_error.h, 170
 IOT_ERROR_CRYPTO_BASE64
 iot_crypto.h, 144
 IOT_ERROR_CRYPTO_BASE64_URLSAFE
 iot_crypto.h, 144
 IOT_ERROR_CRYPTO_CIPHER
 iot_crypto.h, 144
 IOT_ERROR_CRYPTO_CIPHER_ALIGN
 iot_crypto.h, 144
 IOT_ERROR_CRYPTO_CIPHER_IVLEN
 iot_crypto.h, 144
 IOT_ERROR_CRYPTO_CIPHER_KEYLEN
 iot_crypto.h, 144
 IOT_ERROR_CRYPTO_CIPHER_OUTSIZE
 iot_crypto.h, 144
 IOT_ERROR_CRYPTO_CIPHER_UNKNOWN_MODE
 iot_crypto.h, 144
 IOT_ERROR_CRYPTO_CIPHER_UNKNOWN_TYPE
 iot_crypto.h, 144
 IOT_ERROR_CRYPTO_ED_KEY_CONVERT
 iot_crypto.h, 145
 IOT_ERROR_CRYPTO_PK_ECDH
 iot_crypto.h, 145
 IOT_ERROR_CRYPTO_PK_INVALID_ARG
 iot_crypto.h, 145
 IOT_ERROR_CRYPTO_PK_INVALID_CTX
 iot_crypto.h, 145
 IOT_ERROR_CRYPTO_PK_INVALID_KEYLEN
 iot_crypto.h, 145
 IOT_ERROR_CRYPTO_PK_NULL_FUNC
 iot_crypto.h, 145
 IOT_ERROR_CRYPTO_PK_PARSEKEY
 iot_crypto.h, 145

IOT_ERROR_CRYPTO_PK_SIGN	IOT_ERROR_EASYSETUP_WIFI_SCANLIST_NOT_FOUND
<i>iot_crypto.h</i> , 145	<i>iot_easysetup.h</i> , 165
IOT_ERROR_CRYPTO_PK_UNKNOWN_KEYTYPE	IOT_ERROR_FS_CLOSE_FAIL
<i>iot_crypto.h</i> , 145	<i>iot_error.h</i> , 170
IOT_ERROR_CRYPTO_PK_VERIFY	IOT_ERROR_FS_DECRYPT_FAIL
<i>iot_crypto.h</i> , 145	<i>iot_error.h</i> , 170
IOT_ERROR_CRYPTO_SHA256	IOT_ERROR_FS_ENCRYPT_FAIL
<i>iot_crypto.h</i> , 146	<i>iot_error.h</i> , 170
IOT_ERROR_CRYPTO_SS_KDF	IOT_ERROR_FS_ENCRYPT_INIT
<i>iot_crypto.h</i> , 146	<i>iot_error.h</i> , 170
IOT_ERROR_DEINIT_FAIL	IOT_ERROR_FS_NO_FILE
<i>iot_error.h</i> , 170	<i>iot_error.h</i> , 170
IOT_ERROR_EASYSETUP_400_BASE	IOT_ERROR_FS_OPEN_FAIL
<i>iot_easysetup.h</i> , 163	<i>iot_error.h</i> , 170
IOT_ERROR_EASYSETUP_500_BASE	IOT_ERROR_FS_READ_FAIL
<i>iot_easysetup.h</i> , 163	<i>iot_error.h</i> , 170
IOT_ERROR_EASYSETUP_BASE	IOT_ERROR_FS_REMOVE_FAIL
<i>iot_error.h</i> , 170	<i>iot_error.h</i> , 170
IOT_ERROR_EASYSETUP_CERTIFICATE_NOT_FOUND	IOT_ERROR_FS_WRITE_FAIL
<i>iot_easysetup.h</i> , 163	<i>iot_error.h</i> , 170
IOT_ERROR_EASYSETUP_CONFIRM_DENIED	IOT_ERROR_INIT_FAIL
<i>iot_easysetup.h</i> , 163	<i>iot_error.h</i> , 170
IOT_ERROR_EASYSETUP_CONFIRM_TIMEOUT	IOT_ERROR_INVALID_ARGS
<i>iot_easysetup.h</i> , 163	<i>iot_error.h</i> , 170
IOT_ERROR_EASYSETUP_INTERNAL_CRITICAL_ERROR	IOT_ERROR_JWT_BASE
<i>iot_easysetup.h</i> , 163	<i>iot_error.h</i> , 170
IOT_ERROR_EASYSETUP_INTERNAL_SERVER_ERROR	IOT_ERROR_JWT_CJSON
<i>iot_easysetup.h</i> , 163	<i>iot_jwt.h</i> , 190
IOT_ERROR_EASYSETUP_INTERNAL_WARNING	IOT_ERROR_JWT_INVALID_ARG
<i>iot_easysetup.h</i> , 163	<i>iot_jwt.h</i> , 190
IOT_ERROR_EASYSETUP_INVALID_BROKER_URL	IOT_ERROR_JWT_MALLOC
<i>iot_easysetup.h</i> , 164	<i>iot_jwt.h</i> , 190
IOT_ERROR_EASYSETUP_INVALID_PAYLOAD	IOT_ERROR_MEM_ALLOC
<i>iot_easysetup.h</i> , 164	<i>iot_error.h</i> , 170
IOT_ERROR_EASYSETUP_INVALID_PIN	IOT_ERROR_MQTT_CONNECT_FAIL
<i>iot_easysetup.h</i> , 164	<i>iot_error.h</i> , 170
IOT_ERROR_EASYSETUP_INVALID_QR	IOT_ERROR_MQTT_NETCONN_FAIL
<i>iot_easysetup.h</i> , 164	<i>iot_error.h</i> , 170
IOT_ERROR_EASYSETUP_INVALID_REQUEST	IOT_ERROR_MQTT_PUBLISH_FAIL
<i>iot_easysetup.h</i> , 164	<i>iot_error.h</i> , 170
IOT_ERROR_EASYSETUP_INVALID_SEQUENCE	IOT_ERROR_MQTT_REJECT_CONNECT
<i>iot_easysetup.h</i> , 164	<i>iot_error.h</i> , 170
IOT_ERROR_EASYSETUP_MAX	IOT_ERROR_MQTT_SERVER_UNAVAIL
<i>iot_easysetup.h</i> , 164	<i>iot_error.h</i> , 170
IOT_ERROR_EASYSETUP_NOT_SUPPORTED	IOT_ERROR_NET_CONNECT
<i>iot_easysetup.h</i> , 164	<i>iot_error.h</i> , 170
IOT_ERROR_EASYSETUP_PIN_NOT_FOUND	IOT_ERROR_NET_INVALID_INTERFACE
<i>iot_easysetup.h</i> , 164	<i>iot_error.h</i> , 170
IOT_ERROR_EASYSETUP RAND DECODE_ERROR	IOT_ERROR_NONE
<i>iot_easysetup.h</i> , 164	<i>iot_error.h</i> , 170
IOT_ERROR_EASYSETUP_RPK_NOT_FOUND	IOT_ERROR_NOT_IMPLEMENTED
<i>iot_easysetup.h</i> , 165	<i>iot_error.h</i> , 170
IOT_ERROR_EASYSETUP_SERIAL_NOT_FOUND	IOT_ERROR_NV_DATA_ERROR
<i>iot_easysetup.h</i> , 165	<i>iot_error.h</i> , 170
IOT_ERROR_EASYSETUP_SHARED_KEY_CREATION	IOT_ERROR_NV_DATA_NOT_EXIST
<i>iot_error.h</i> , 165	<i>iot_error.h</i> , 170
IOT_ERROR_EASYSETUP_SPUB_DECODE_ERROR	IOT_ERROR_PROV_FAIL
<i>iot_error.h</i> , 165	<i>iot_error.h</i> , 170

IOT_ERROR_READ_FAIL
iot_error.h, 170

IOT_ERROR_REG_UPDATED
iot_error.h, 170

iot_error_t
iot_error.h, 170

IOT_ERROR_TIMEOUT
iot_error.h, 170

IOT_ERROR_UNINITIALIZED
iot_error.h, 170

IOT_ERROR_UUID_FAIL
iot_error.h, 170

IOT_ERROR_WRITE_FAIL
iot_error.h, 170

iot_es_connect
iot_easystep_st_mqtt.c, 117
iot_internal.h, 181

iot_es_crypto_free_pk
iot_easystep_crypto.c, 112
iot_internal.h, 182

iot_es_crypto_init_pk
iot_easystep_crypto.c, 112
iot_internal.h, 182

iot_es_crypto_load_pk
iot_easystep_crypto.c, 112
iot_internal.h, 182

iot_es_disconnect
iot_easystep_st_mqtt.c, 117
iot_internal.h, 183

IOT_ES_URI_GET_DEVICEINFO
iot_easystep.h, 165

IOT_ES_URI_GET_LOGS_DUMP
iot_easystep.h, 165

IOT_ES_URI_GET_LOGS_SYSTEMINFO
iot_easystep.h, 165

IOT_ES_URI_GET_POST_RESPONSE
iot_easystep.h, 165

IOT_ES_URI_GET_WIFISCANINFO
iot_easystep.h, 165

IOT_ES_URI_POST_CONFIRM
iot_easystep.h, 166

IOT_ES_URI_POST_CONFIRMINFO
iot_easystep.h, 166

IOT_ES_URI_POST_KEYINFO
iot_easystep.h, 166

IOT_ES_URI_POST_LOGS
iot_easystep.h, 166

IOT_ES_URI_POST_SETUPCOMPLETE
iot_easystep.h, 166

IOT_ES_URI_POST_WIFIPROVISIONINGINFO
iot_easystep.h, 166

IOT_EVENT
st_dev.h, 285

IOT_EVENT_BIT_ALL
iot_main.h, 193

IOT_EVENT_BIT_CAPABILITY
iot_main.h, 193

IOT_EVENT_BIT_COMMAND

iot_main.h, 193

IOT_EVENT_BIT_EASYSETUP_CONFIRM
iot_main.h, 193

IOT_EVENT_BIT_EASYSETUP_REQ
iot_main.h, 193

IOT_EVENT_BIT_EASYSETUP_RESP
iot_main.h, 193

iot_events
iot_context, 32

iot_get_time_in_ms
iot_api.c, 304
iot_internal.h, 183

iot_get_time_in_sec
iot_api.c, 304
iot_internal.h, 184

IOT_INFO
iot_debug.h, 159

iot_internal.h
iot_api_device_info_load, 174
iot_api_device_info_mem_free, 175
iot_api_onboarding_config_load, 175
iot_api_onboarding_config_mem_free, 176
iot_api_prov_data_mem_free, 176
iot_api_read_device_identity, 177
iot_cap_call_init_cb, 178
iot_cap_sub_cb, 178
iot_command_send, 179
IOT_DEFAULT_TIMEOUT, 173
iot_device_cleanup, 180
iot_easystep_request, 181
iot_es_connect, 181
iot_es_crypto_free_pk, 182
iot_es_crypto_init_pk, 182
iot_es_crypto_load_pk, 182
iot_es_disconnect, 183
iot_get_time_in_ms, 183
iot_get_time_in_sec, 184
iot_mqtt_connect, 184
iot_mqtt_disconnect, 185
IOT_MQTT_KEEPALIVE_INTERVAL, 173
iot_mqtt_publish, 185
iot_mqtt_registration, 186
iot_mqtt_subscribe, 187
iot_mqtt_unsubscribe, 187
iot_noti_sub_cb, 188
IOT_PAYLOAD_SIZE, 173
IOT_PUB_QUEUE_LENGTH, 173
IOT_PUB_TOPIC_EVENT, 173
IOT_PUB_TOPIC_REGISTRATION, 173
IOT_QUEUE_LENGTH, 173
iot_state_update, 188
IOT_SUB_TOPIC_COMMAND, 173
IOT_SUB_TOPIC_NOTIFICATION, 173
IOT_SUB_TOPIC_REGISTRATION, 174
IOT_TASK_NAME, 174
IOT_TASK_PRIORITY, 174
IOT_TASK_STACK_SIZE, 174
IOT_TOPIC_SIZE, 174

iot_jwt.c
 iot_jwt_create, 317

iot_jwt.h
 IOT_ERROR_JWT_CJSON, 190
 IOT_ERROR_JWT_INVALID_ARG, 190
 IOT_ERROR_JWT_MALLOC, 190
 iot_jwt_create, 190

iot_jwt_create
 iot_jwt.c, 317
 iot_jwt.h, 190

iot_mac, 50
 addr, 51

iot_main.c
 EASYSETUP_TIMEOUT_MS, 319
 NEXT_STATE_TIMEOUT_MS, 319
 st_conn_cleanup, 319
 st_conn_init, 319
 st_conn_start, 320

iot_main.h
 _IOT_NOTI_TYPE_DEV_DELETED, 194
 _IOT_NOTI_TYPE_JWT_EXPIRED, 194
 _IOT_NOTI_TYPE_QUOTA_REACHED, 194
 _IOT_NOTI_TYPE_RATE_LIMIT, 194
 _IOT_NOTI_TYPE_UNKNOWN, 194
 _iot_noti_type, 194
 IOT_BUF_RX_SIZE, 193
 IOT_BUF_TX_SIZE, 193
 iot_cap_handle_list_t, 194
 IOT_CMD_STATE_HANDLE, 195
 IOT_COMMAND_CHECK_CLOUD_STATE, 195
 IOT_COMMAND_CHECK_PROV_STATUS, 195
 IOT_COMMAND_CLOUD_CONNECTING, 195
 IOT_COMMAND_CLOUD_REGISTERED, 195
 IOT_COMMAND_CLOUD_REGISTERING, 195
 IOT_COMMAND_NETWORK_MODE, 195
 IOT_COMMAND_NOTIFICATION_RECEIVED, 195
 IOT_COMMAND_READY_TO_CTL, 195
 IOT_COMMAND_SELF_CLEANUP, 195
 iot_command_type, 194
 IOT_COMMAND_TYPE_MAX, 195
 iot_connect_type, 195
 IOT_CONNECT_TYPE_COMMUNICATION, 195
 IOT_CONNECT_TYPE_REGISTRATION, 195
 iot_easystep_step, 195
 IOT_EASYSETUP_STEP_CONFIRM, 195
 IOT_EASYSETUP_STEP_CONFIRMINFO, 195
 IOT_EASYSETUP_STEP_DEVICEINFO, 195
 IOT_EASYSETUP_STEP_KEYINFO, 195
 IOT_EASYSETUP_STEP_LOG_CREATE_DUMP, 195
 IOT_EASYSETUP_STEP_LOG_GET_DUMP, 195
 IOT_EASYSETUP_STEP_LOG_SYSTEMINFO, 195
 IOT_EASYSETUP_STEP_SETUPCOMPLETE, 195
 IOT_EASYSETUP_STEP_WIFIPROVIONINGINFO, 195

IOT_EASYSETUP_STEP_WIFISCANINFO, 195
IOT_EVENT_BIT_ALL, 193
IOT_EVENT_BIT_CAPABILITY, 193
IOT_EVENT_BIT_COMMAND, 193
IOT_EVENT_BIT_EASYSETUP_CONFIRM, 193
IOT_EVENT_BIT_EASYSETUP_REQ, 193
IOT_EVENT_BIT_EASYSETUP_RESP, 193
IOT_REG_UUID_STR_LEN, 194
IOT_STATE_CHANGE_FAILED, 196
IOT_STATE_CLOUD_CONNECTED, 196
IOT_STATE_CLOUD_CONNECTING, 196
IOT_STATE_CLOUD_DISCONNECTED, 196
IOT_STATE_CLOUD_REGISTERING, 196
IOT_STATE_INITIALIZED, 196
iot_state_opt, 195
IOT_STATE_OPT_NEED_INTERACT, 195
IOT_STATE_OPT_NONE, 195
IOT_STATE_PROV_CONFIRMING, 196
IOT_STATE_PROV_DONE, 196
IOT_STATE_PROV_ENTER, 196
iot_state_t, 194
iot_state_type, 196
IOT_STATE_UNKNOWN, 196
IOT_WIFI_PROV_PASSWORD_LEN, 194
IOT_WIFI_PROV_SSID_LEN, 194

IOT_MEM_CHECK
 iot_debug.h, 160

iot_mqtt.c
 _iot_mqtt_cmd_sub_callback, 322
 _iot_mqtt_noti_sub_callback, 322
 iot_mqtt_connect, 323
 iot_mqtt_disconnect, 324
 iot_mqtt_publish, 324
 iot_mqtt_registration, 325
 iot_mqtt_subscribe, 326
 iot_mqtt_unsubscribe, 326

iot_mqtt_client.c
 cycle, 352
 deliverMessage, 352
 keepalive, 353
 MQTTCleanSession, 354
 MQTTClientInit, 354
 MQTTCloseSession, 355
 MQTTConnect, 356
 MQTTConnectWithResults, 356
 MQTTDisconnect, 357
 MQTTPublish, 358
 MQTTSetMessageHandler, 359
 MQTTSubscribe, 360
 MQTTSubscribeWithResults, 360
 MQTTUnsubscribe, 361
 MQTTYield, 363
 waitfor, 364

iot_mqtt_client.h
 CONFIG_STDK_MQTT_DYNAMIC_BUFFER, 222
 CONFIG_STDK_MQTT_PING_RETRY, 222
 CONFIG_STDK_MQTT_PUBLISH_RETRY, 222
 CONFIG_STDK_MQTT_RECV_BUFFER, 222

CONFIG_STDK_MQTT_RECV_CYCLE, 222
 CONFIG_STDK_MQTT_SEND_BUFFER, 222
 CONFIG_STDK_MQTT_SEND_CYCLE, 222
 DefaultClient, 223
 DLLExport, 223
 DLLImport, 223
 IOT_MQTT_PRIORITY, 223
 IOT_MQTT_STACK_SIZE, 223
 MAX_MESSAGE_HANDLERS, 223
 MAX_PACKET_ID, 223
 MessageData, 223
 messageHandler, 223
 MQTT_BUFFER_OVERFLOW, 224
 MQTT_DISCONNECTED, 224
 MQTT_FAILURE, 224
 MQTT_SUCCESS, 224
 MQTTClient, 223
 MQTTClientInit, 224
 MQTTConnackData, 224
 MQTTConnect, 225
 MQTTConnectWithResults, 226
 MQTTDisconnect, 227
 MQTTMessage, 224
 MQTTPublish, 228
 MQTTSetMessageHandler, 229
 MQTTSubackData, 224
 MQTTSubscribe, 230
 MQTTSubscribeWithResults, 230
 MQTTUnsubscribe, 231
 MQTTYield, 233
 QoS, 224
 QOS0, 224
 QOS1, 224
 QOS2, 224
 returnCode, 224
 SUBFAIL, 224
 iot_mqtt_connect
 iot_internal.h, 184
 iot_mqtt.c, 323
 iot_mqtt_connect.h
 connack_return_codes, 235
 DLLExport, 235
 DLLImport, 235
 MQTT_BAD_USERNAME_OR_PASSWORD, 236
 MQTT_CLIENTID_REJECTED, 236
 MQTT_CONNECTION_ACCEPTED, 236
 MQTT_NOT_AUTHORIZED, 236
 MQTT_SERVER_UNAVAILABLE, 236
 MQTT_UNACCEPTABLE_PROTOCOL, 236
 MQTTDeserialize_connack, 236
 MQTTDeserialize_connect, 236
 MQTTPacket_connectData_initializer, 235
 MQTTPacket_willOptions_initializer, 235
 MQTTSerialize_connack, 237
 MQTTSerialize_connect, 237
 MQTTSerialize_connect_size, 238
 MQTTSerialize_disconnect, 238
 MQTTSerialize_disconnect_size, 239
 iot_mqtt_connect_client.c
 MQTTSerialize_pingreq, 239
 MQTTSerialize_pingreq_size, 240
 iot_mqtt_connect_server.c
 min, 370
 MQTTDeserialize_connect, 370
 MQTTPacket_checkVersion, 371
 MQTTSerialize_connack, 371
 iot_mqtt_ctx, 51
 buf, 53
 cli, 53
 cmd_filter, 53
 iot_ctx, 53
 mqtt_connected, 53
 net, 53
 noti_filter, 54
 readbuf, 54
 iot_mqtt_deserialize_publish.c
 min, 372
 MQTTDeserialize_ack, 372
 MQTTDeserialize_publish, 373
 iot_mqtt_disconnect
 iot_internal.h, 185
 iot_mqtt.c, 324
 iot_mqtt_format.c
 MQTTPacket_getName, 374
 MQTTPacket_names, 376
 MQTTStringFormat_ack, 374
 MQTTStringFormat_connack, 375
 MQTTStringFormat_connect, 375
 MQTTStringFormat_publish, 375
 MQTTStringFormat_suback, 375
 MQTTStringFormat_subscribe, 375
 MQTTStringFormat_unsubscribe, 375
 iot_mqtt_format.h
 MQTTFormat_toClientString, 241
 MQTTFormat_toServerString, 241
 MQTTPacket_getName, 241
 MQTTStringFormat_ack, 242
 MQTTStringFormat_connack, 242
 MQTTStringFormat_connect, 242
 MQTTStringFormat_publish, 242
 MQTTStringFormat_suback, 242
 MQTTStringFormat_subscribe, 242
 MQTTStringFormat_unsubscribe, 243
 IOT_MQTT_KEEPALIVE_INTERVAL
 iot_internal.h, 173
 iot_mqtt_packet.c
 bufchar, 377

getLenStringLen, 377
MAX_NO_OF_REMAINING_LENGTH_BYTES,
 377
MQTTPacket_decode, 377
MQTTPacket_decodeBuf, 378
MQTTPacket_encode, 378
MQTTPacket_equals, 378
MQTTPacket_len, 379
MQTTPacket_msgTypesToString, 379
MQTTPacket_read, 379
MQTTPacket_readnb, 380
MQTTstrlen, 380
readChar, 381
readInt, 381
readMQTTLenString, 381
writeChar, 382
writeCString, 382
writeInt, 383
writeMQTTString, 383
iot_mqtt_packet.h
 CONNACK, 245
 CONNECT, 245
 DISCONNECT, 245
 DLLExport, 244
 DLLImport, 245
 errors, 245
 MQTTDeserialize_ack, 245
 MQTTPACKET_BUFFER_TOO_SHORT, 245
 MQTTPacket_decode, 246
 MQTTPacket_decodeBuf, 246
 MQTTPacket_encode, 246
 MQTTPacket_equals, 247
 MQTTPacket_len, 247
 MQTTPacket_msgTypesToString, 248
 MQTTPacket_read, 248
 MQTTPACKET_READ_COMPLETE, 245
 MQTTPACKET_READ_ERROR, 245
 MQTTPacket_readnb, 248
 MQTTSerialize_ack, 249
 MQTTSerialize_ack_size, 249
 MQTTString_initializer, 245
 MQTTstrlen, 249
 msgTypes, 245
 PINGREQ, 245
 PINGRESP, 245
 PUBACK, 245
 PUBCOMP, 245
 PUBLISH, 245
 PUBREC, 245
 PUBREL, 245
 readChar, 250
 readInt, 250
 readMQTTLenString, 250
 SUBACK, 245
 SUBSCRIBE, 245
 UNSUBACK, 245
 UNSUBSCRIBE, 245
 writeChar, 251
writeCString, 251
writeInt, 252
writeMQTTString, 252
IOT_MQTT_PRIORITY
 iot_mqtt_client.h, 223
iot_mqtt_publish
 iot_internal.h, 185
 iot_mqtt.c, 324
iot_mqtt_publish.h
 DLLExport, 254
 DLLImport, 254
 MQTTDeserialize_publish, 254
 MQTTSerialize_puback, 255
 MQTTSerialize_pubcomp, 255
 MQTTSerialize_publish, 255
 MQTTSerialize_publish_header, 256
 MQTTSerialize_publish_size, 256
 MQTTSerialize_pubrel, 257
iot_mqtt_registration
 iot_internal.h, 186
 iot_mqtt.c, 325
iot_mqtt_serialize_publish.c
 MQTTSerialize_ack, 385
 MQTTSerialize_ack_size, 385
 MQTTSerialize_puback, 385
 MQTTSerialize_pubcomp, 386
 MQTTSerialize_publish, 386
 MQTTSerialize_publish_header, 387
 MQTTSerialize_publish_size, 387
 MQTTSerialize_publishLength, 388
 MQTTSerialize_pubrel, 388
IOT_MQTT_STACK_SIZE
 iot_mqtt_client.h, 223
iot_mqtt_stacktrace.h
 FUNC_ENTRY, 258
 FUNC_ENTRY_MAX, 259
 FUNC_ENTRY_MED, 259
 FUNC_ENTRY_NOLOG, 259
 FUNC_EXIT, 259
 FUNC_EXIT_MAX, 259
 FUNC_EXIT_MAX_RC, 259
 FUNC_EXIT_MED, 259
 FUNC_EXIT_MED_RC, 259
 FUNC_EXIT_NOLOG, 259
 FUNC_EXIT_RC, 259
 NOSTACKTRACE, 259
iot_mqtt_subscribe
 iot_internal.h, 187
 iot_mqtt.c, 326
iot_mqtt_subscribe.h
 DLLExport, 260
 DLLImport, 260
 MQTTDeserialize_suback, 260
 MQTTDeserialize_subscribe, 261
 MQTTSerialize_suback, 261
 MQTTSerialize_subscribe, 263
 MQTTSerialize_subscribe_size, 263
iot_mqtt_subscribe_client.c

MQTTDeserialize_suback, 389
 MQTTSerialize_subscribe, 390
 MQTTSerialize_subscribe_size, 391
 MQTTSerialize_subscribeLength, 391
 iot_mqtt_subscribe_server.c
 MQTTDeserialize_subscribe, 392
 MQTTSerialize_suback, 393
 iot_mqtt_unsubscribe
 iot_internal.h, 187
 iot_mqtt.c, 326
 iot_mqtt_unsubscribe.h
 DLLExport, 265
 DLLImport, 265
 MQTTDeserialize_unsuback, 265
 MQTTDeserialize_unsubscribe, 265
 MQTTSerialize_unsuback, 266
 MQTTSerialize_unsubscribe, 266
 MQTTSerialize_unsubscribe_size, 267
 iot_mqtt_unsubscribe_client.c
 MQTTDeserialize_unsuback, 394
 MQTTSerialize_unsubscribe, 394
 MQTTSerialize_unsubscribe_size, 395
 MQTTSerialize_unsubscribeLength, 396
 iot_mqtt_unsubscribe_server.c
 MQTTDeserialize_unsubscribe, 397
 MQTTSerialize_unsuback, 398
 iot_net.h
 iot_net_connection_t, 197
 iot_net_init, 197
 iot_net_interface_t, 197
 iot_net_connection, 54
 ca_cert, 55
 ca_cert_len, 55
 cert, 55
 cert_len, 55
 key, 56
 key_len, 56
 port, 56
 url, 56
 iot_net_connection_t
 iot_net.h, 197
 iot_net_init
 iot_net.h, 197
 iot_net_mbedtls.c, 399
 iot_net_openssl.c, 401
 iot_net_interface, 56
 connect, 58
 connection, 58
 context, 58
 disconnect, 58
 read, 58
 select, 58
 show_status, 58
 write, 59
 iot_net_interface_t
 iot_net.h, 197
 iot_net_mbedtls.c
 iot_net_init, 399
 iot_net_openssl.c
 iot_net_init, 401
 iot_net_openssl.c
 __SSL_library_init, 401
 iot_net_init, 401
 SSL_library_init, 403
 iot_net_platform.h
 iot_net_platform_context_t, 400, 401
 iot_net_platform_context, 59
 cacert, 60
 conf, 60
 ctr_drbg, 60
 ctx, 60
 entropy, 60
 method, 60
 read_count, 60
 server_fd, 61
 socket, 61
 ssl, 61
 iot_net_platform_context_t
 iot_net_platform.h, 400, 401
 iot_noti_data_t, 61
 raw, 62
 type, 63
 iot_noti_sub_cb
 iot_capability.c, 307
 iot_internal.h, 188
 iot_noti_type
 st_dev.h, 286
 IOT_NOTI_TYPE_DEV_DELETED
 st_dev.h, 286
 IOT_NOTI_TYPE_QUOTA_REACHED
 st_dev.h, 286
 IOT_NOTI_TYPE_RATE_LIMIT
 st_dev.h, 286
 iot_noti_type_t
 st_dev.h, 285
 IOT_NOTI_TYPE_UNKNOWN
 st_dev.h, 286
 iot_nv_data.c
 _iot_nv_read_data, 329
 _iot_nv_write_data, 330
 iot_nv_deinit, 330
 iot_nv_erase, 331
 iot_nv_erase_prov_data, 332
 iot_nv_get_client_certificate, 333
 iot_nv_get_cloud_prov_data, 334
 iot_nv_get_device_id, 335
 iot_nv_get_private_key, 335
 iot_nv_get_prov_data, 336
 iot_nv_get_public_key, 337
 iot_nv_get_root_certificate, 338
 iot_nv_get_serial_number, 338
 iot_nv_get_wifi_prov_data, 339
 iot_nv_init, 340
 iot_nv_set_cloud_prov_data, 341
 iot_nv_set_device_id, 342
 iot_nv_set_prov_data, 343
 iot_nv_set_wifi_prov_data, 343
 IOT_NVD_MAX_BSSID_LEN, 328

IOT_NVD_MAX_DATA_LEN, 329
IOT_NVD_MAX_ID_LEN, 329
IOT_NVD_MAX_PW_LEN, 329
IOT_NVD_MAX_UID_LEN, 329
iot_nv_data.h
 iot_nv_deinit, 200
 iot_nv_erase, 200
 iot_nv_erase_prov_data, 201
 iot_nv_get_client_certificate, 202
 iot_nv_get_cloud_prov_data, 203
 iot_nv_get_device_id, 204
 iot_nv_get_private_key, 204
 iot_nv_get_prov_data, 205
 iot_nv_get_public_key, 206
 iot_nv_get_root_certificate, 207
 iot_nv_get_serial_number, 207
 iot_nv_get_wifi_prov_data, 208
 iot_nv_init, 209
 iot_nv_set_cloud_prov_data, 210
 iot_nv_set_device_id, 211
 iot_nv_set_prov_data, 212
 iot_nv_set_wifi_prov_data, 212
 IOT_NVD_AP_AUTH_TYPE, 199
 IOT_NVD_AP_BSSID, 199
 IOT_NVD_AP_PASS, 199
 IOT_NVD_AP_SSID, 199
 IOT_NVD_CA_CERT, 200
 IOT_NVD_CLOUD_PROV_STATUS, 199
 IOT_NVD_DEVICE_ID, 200
 IOT_NVD_LABEL, 200
 IOT_NVD_LOCATION_ID, 200
 IOT_NVD_MAX, 200
 IOT_NVD_PRIVATE_KEY, 200
 IOT_NVD_PUBLIC_KEY, 200
 IOT_NVD_ROOM_ID, 200
 IOT_NVD_SERIAL_NUM, 200
 IOT_NVD_SERVER_PORT, 200
 IOT_NVD_SERVER_URL, 199
 IOT_NVD_SUB_CERT, 200
 iot_nvd_t, 199
 IOT_NVD_WIFI_PROV_STATUS, 199
iot_nv_deinit
 iot_nv_data.c, 330
 iot_nv_data.h, 200
iot_nv_erase
 iot_nv_data.c, 331
 iot_nv_data.h, 200
iot_nv_erase_prov_data
 iot_nv_data.c, 332
 iot_nv_data.h, 201
iot_nv_get_client_certificate
 iot_nv_data.c, 333
 iot_nv_data.h, 202
iot_nv_get_cloud_prov_data
 iot_nv_data.c, 334
 iot_nv_data.h, 203
iot_nv_get_device_id
 iot_nv_data.c, 335
 iot_nv_data.h, 204
 iot_nv_get_private_key
 iot_nv_data.c, 335
 iot_nv_data.h, 204
 iot_nv_get_prov_data
 iot_nv_data.c, 336
 iot_nv_data.h, 205
 iot_nv_get_public_key
 iot_nv_data.c, 337
 iot_nv_data.h, 206
 iot_nv_get_root_certificate
 iot_nv_data.c, 338
 iot_nv_data.h, 207
 iot_nv_get_serial_number
 iot_nv_data.c, 338
 iot_nv_data.h, 207
 iot_nv_get_wifi_prov_data
 iot_nv_data.c, 339
 iot_nv_data.h, 208
iot_nv_init
 iot_nv_data.c, 340
 iot_nv_data.h, 209
iot_nv_set_cloud_prov_data
 iot_nv_data.c, 341
 iot_nv_data.h, 210
iot_nv_set_device_id
 iot_nv_data.c, 342
 iot_nv_data.h, 211
iot_nv_set_prov_data
 iot_nv_data.c, 343
 iot_nv_data.h, 212
iot_nv_set_wifi_prov_data
 iot_nv_data.c, 343
 iot_nv_data.h, 212
IOT_NVD_AP_AUTH_TYPE
 iot_nv_data.h, 199
IOT_NVD_AP_BSSID
 iot_nv_data.h, 199
IOT_NVD_AP_PASS
 iot_nv_data.h, 199
IOT_NVD_AP_SSID
 iot_nv_data.h, 199
IOT_NVD_CA_CERT
 iot_nv_data.h, 200
IOT_NVD_CLOUD_PROV_STATUS
 iot_nv_data.h, 199
IOT_NVD_DEVICE_ID
 iot_nv_data.h, 200
IOT_NVD_LABEL
 iot_nv_data.h, 200
IOT_NVD_LOCATION_ID
 iot_nv_data.h, 200
IOT_NVD_MAX
 iot_nv_data.h, 200
IOT_NVD_MAX_BSSID_LEN
 iot_nv_data.c, 328
IOT_NVD_MAX_DATA_LEN
 iot_nv_data.c, 329

IOT_NVD_MAX_ID_LEN
 iot_nv_data.c, 329

IOT_NVD_MAX_PW_LEN
 iot_nv_data.c, 329

IOT_NVD_MAX_UID_LEN
 iot_nv_data.c, 329

IOT_NVD_PRIVATE_KEY
 iot_nv_data.h, 200

IOT_NVD_PUBLIC_KEY
 iot_nv_data.h, 200

IOT_NVD_ROOM_ID
 iot_nv_data.h, 200

IOT_NVD_SERIAL_NUM
 iot_nv_data.h, 200

IOT_NVD_SERVER_PORT
 iot_nv_data.h, 200

IOT_NVD_SERVER_URL
 iot_nv_data.h, 199

IOT_NVD_SUB_CERT
 iot_nv_data.h, 200

iot_nvd_t
 iot_nv_data.h, 199

IOT_NVD_WIFI_PROV_STATUS
 iot_nv_data.h, 199

iot_os_delay
 iot_os_util.h, 270

iot_os_eventgroup
 iot_os_util.h, 270

iot_os_eventgroup_clear_bits
 iot_os_util.h, 271

iot_os_eventgroup_create
 iot_os_util.h, 271

iot_os_eventgroup_delete
 iot_os_util.h, 272

iot_os_eventgroup_set_bits
 iot_os_util.h, 272

iot_os_eventgroup_wait_bits
 iot_os_util.h, 273

IOT_OS_FALSE
 iot_os_util.h, 269

iot_os_false
 iot_os_util.h, 281

IOT_OS_MAX_DELAY
 iot_os_util.h, 270

iot_os_max_delay
 iot_os_util.h, 282

iot_os_mutex, 63
 iot_os_util.h, 270

sem, 63

iot_os_mutex_init
 iot_os_util.h, 273

iot_os_mutex_lock
 iot_os_util.h, 274

iot_os_mutex_unlock
 iot_os_util.h, 275

iot_os_queue
 iot_os_util.h, 270

iot_os_queue_create

iot_os_util.h, 275

iot_os_queue_delete
 iot_os_util.h, 275

iot_os_queue_receive
 iot_os_util.h, 276

iot_os_queue_reset
 iot_os_util.h, 276

iot_os_queue_send
 iot_os_util.h, 276

iot_os_sem
 iot_os_util.h, 270

iot_os_thread
 iot_os_util.h, 270

iot_os_thread_create
 iot_os_util.h, 277

iot_os_thread_delete
 iot_os_util.h, 278

iot_os_thread_yield
 iot_os_util.h, 278

iot_os_timer
 iot_os_util.h, 270

iot_os_timer_count_ms
 iot_os_util.h, 278

iot_os_timer_destroy
 iot_os_util.h, 279

iot_os_timer_init
 iot_os_util.h, 279

iot_os_timer_isexpired
 iot_os_util.h, 280

iot_os_timer_left_ms
 iot_os_util.h, 281

 IOT_OS_TRUE
 iot_os_util.h, 270

iot_os_true
 iot_os_util.h, 282

iot_os_util.h

 IOT_DELAY, 269

iot_os_delay, 270

iot_os_eventgroup, 270

iot_os_eventgroup_clear_bits, 271

iot_os_eventgroup_create, 271

iot_os_eventgroup_delete, 272

iot_os_eventgroup_set_bits, 272

iot_os_eventgroup_wait_bits, 273

 IOT_OS_FALSE, 269

iot_os_false, 281

 IOT_OS_MAX_DELAY, 270

iot_os_max_delay, 282

iot_os_mutex, 270

iot_os_mutex_init, 273

iot_os_mutex_lock, 274

iot_os_mutex_unlock, 275

iot_os_queue, 270

iot_os_queue_create, 275

iot_os_queue_delete, 275

iot_os_queue_receive, 276

iot_os_queue_reset, 276

iot_os_queue_send, 276

iot_os_sem, 270
iot_os_thread, 270
iot_os_thread_create, 277
iot_os_thread_delete, 278
iot_os_thread_yield, 278
iot_os_timer, 270
iot_os_timer_count_ms, 278
iot_os_timer_destroy, 279
iot_os_timer_init, 279
iot_os_timer_isexpired, 280
iot_os_timer_left_ms, 281
IOT_OS_TRUE, 270
iot_os_true, 282
IOT_OVF_TYPE_BUTTON
 iot_easystart.h, 166
IOT_OVF_TYPE_JUSTWORKS
 iot_easystart.h, 166
IOT_OVF_TYPE_PIN
 iot_easystart.h, 166
IOT_OVF_TYPE_QR
 iot_easystart.h, 166
IOT_PAYLOAD_SIZE
 iot_internal.h, 173
iot_pin_t, 64
 pin, 64
 st_dev.h, 285
IOT_POWEROFF
 iot_bsp_system.h, 130
IOT_PUB_QUEUE_LENGTH
 iot_internal.h, 173
IOT_PUB_TOPIC_EVENT
 iot_internal.h, 173
IOT_PUB_TOPIC_REGISTRATION
 iot_internal.h, 173
IOT_QUEUE_LENGTH
 iot_internal.h, 173
iot_random_uuid_from_mac
 iot_uuid.c, 350
 iot_uuid.h, 219
IOT_REBOOT
 iot_bsp_system.h, 130
iot_reg_data
 iot_context, 32
IOT_REG_UUID_STR_LEN
 iot_main.h, 194
iot_registered_data, 65
 deviceld, 65
 new_reged, 65
 updated, 65
IOT_SOFT_AP_CHANNEL
 iot_bsp_wifi.h, 134
iot_stat_lv
 st_dev.h, 286
IOT_STAT_LV_DONE
 st_dev.h, 287
IOT_STAT_LV_FAIL
 st_dev.h, 287
IOT_STAT_LV_START
 st_dev.h, 287
 st_dev.h, 287
IOT_STAT_LV_STAY
 st_dev.h, 286
iot_stat_lv_t
 st_dev.h, 285
iot_state
 iot_state_data, 66
IOT_STATE_CHANGE_FAILED
 iot_main.h, 196
IOT_STATE_CLOUD_CONNECTED
 iot_main.h, 196
IOT_STATE_CLOUD_CONNECTING
 iot_main.h, 196
IOT_STATE_CLOUD_DISCONNECTED
 iot_main.h, 196
IOT_STATE_CLOUD_REGISTERING
 iot_main.h, 196
iot_state_data, 66
 iot_state, 66
 opt, 66
IOT_STATE_INITIALIZED
 iot_main.h, 196
iot_state_opt
 iot_main.h, 195
IOT_STATE_OPT_NEED_INTERACT
 iot_main.h, 195
IOT_STATE_OPT_NONE
 iot_main.h, 195
IOT_STATE_PROV_CONFIRMING
 iot_main.h, 196
IOT_STATE_PROV_DONE
 iot_main.h, 196
IOT_STATE_PROV_ENTER
 iot_main.h, 196
iot_state_t
 iot_main.h, 194
iot_state_type
 iot_main.h, 196
IOT_STATE_UNKNOWN
 iot_main.h, 196
iot_state_update
 iot_api.c, 304
 iot_internal.h, 188
iot_status
 st_dev.h, 287
IOT_STATUS_ALL
 st_dev.h, 287
IOT_STATUS_CONNECTING
 st_dev.h, 287
IOT_STATUS_IDLE
 st_dev.h, 287
IOT_STATUS_NEED_INTERACT
 st_dev.h, 287
IOT_STATUS_PROVISIONING
 st_dev.h, 287
iot_status_t
 st_dev.h, 285
IOT_SUB_TOPIC_COMMAND

iot_internal.h, 173
IOT_SUB_TOPIC_NOTIFICATION
 iot_internal.h, 173
IOT_SUB_TOPIC_REGISTRATION
 iot_internal.h, 174
IOT_TASK_NAME
 iot_internal.h, 174
IOT_TASK_PRIORITY
 iot_internal.h, 174
IOT_TASK_STACK_SIZE
 iot_internal.h, 174
IOT_TOPIC_SIZE
 iot_internal.h, 174
iot_util.c
 iot_util_convert_channel_freq, 345
 iot_util_convert_mac_str, 346
 iot_util_convert_str_mac, 346
 iot_util_convert_str_uuid, 347
 iot_util_convert_uuid_str, 347
 iot_util_get_random_uuid, 348
 iot_util_url_parse, 348
iot_util.h
 iot_util_convert_channel_freq, 215
 iot_util_convert_mac_str, 215
 iot_util_convert_str_mac, 215
 iot_util_convert_str_uuid, 216
 iot_util_convert_uuid_str, 216
 iot_util_get_random_uuid, 217
 iot_util_url_parse, 218
iot_util_convert_channel_freq
 iot_util.c, 345
 iot_util.h, 215
iot_util_convert_mac_str
 iot_util.c, 346
 iot_util.h, 215
iot_util_convert_str_mac
 iot_util.c, 346
 iot_util.h, 215
iot_util_convert_str_uuid
 iot_util.c, 347
 iot_util.h, 216
iot_util_convert_uuid_str
 iot_util.c, 347
 iot_util.h, 216
iot_util_get_random_uuid
 iot_util.c, 348
 iot_util.h, 217
iot_util_url_parse
 iot_util.c, 348
 iot_util.h, 218
iot_uuid, 67
 id, 67
iot_uuid.c
 iot_random_uuid_from_mac, 350
 iot_uuid_from_mac, 350
iot_uuid.h
 iot_random_uuid_from_mac, 219
 iot_uuid_from_mac, 219
 iot_uuid_from_mac
 iot_uuid.h
 iot_random_uuid_from_mac
 iot_random_uuid_from_mac
iot_uuid_from_mac
 iot_random_uuid_from_mac, 219
 iot_uuid_from_mac, 219
iot_uuid_from_mac
 iot_uuid.c, 350
 iot_uuid.h, 219
IOT_WARN
 iot_debug.h, 160
IOT_WARN_CHECK
 iot_debug.h, 160
IOT_WIFI_AUTH_MAX
 iot_bsp_wifi.h, 134
iot_wifi_auth_mode_t
 iot_bsp_wifi.h, 134
IOT_WIFI_AUTH_OPEN
 iot_bsp_wifi.h, 134
IOT_WIFI_AUTH_WEP
 iot_bsp_wifi.h, 134
IOT_WIFI_AUTH_WPA2_ENTERPRISE
 iot_bsp_wifi.h, 134
IOT_WIFI_AUTH_WPA2_PSK
 iot_bsp_wifi.h, 134
IOT_WIFI_AUTH_WPA_PSK
 iot_bsp_wifi.h, 134
IOT_WIFI_AUTH_WPA_WPA2_PSK
 iot_bsp_wifi.h, 134
IOT_WIFI_CMD_TIMEOUT
 iot_bsp_wifi.h, 134
iot_wifi_conf, 68
 bssid, 68
 mode, 68
 pass, 69
 ssid, 69
IOT_WIFI_FREQ_2_4G_5G_BOTH
 iot_bsp_wifi.h, 135
IOT_WIFI_FREQ_2_4G_ONLY
 iot_bsp_wifi.h, 135
IOT_WIFI_FREQ_5G_ONLY
 iot_bsp_wifi.h, 135
iot_wifi_freq_t
 iot_bsp_wifi.h, 134
IOT_WIFI_MAX_BSSID_LEN
 iot_bsp_wifi.h, 134
IOT_WIFI_MAX_PASS_LEN
 iot_bsp_wifi.h, 134
IOT_WIFI_MAX_SCAN_RESULT
 iot_bsp_wifi.h, 134
IOT_WIFI_MAX_SSID_LEN
 iot_bsp_wifi.h, 134
IOT_WIFI_MODE_OFF
 iot_bsp_wifi.h, 135
IOT_WIFI_MODE_P2P
 iot_bsp_wifi.h, 135
IOT_WIFI_MODE_SCAN
 iot_bsp_wifi.h, 135
IOT_WIFI_MODE_SOFTAP
 iot_bsp_wifi.h, 135
IOT_WIFI_MODE_STATION
 iot_bsp_wifi.h, 135
iot_wifi_mode_t
 iot_bsp_wifi.h, 135

IOT_WIFI_MODE_UNDEFINED
 iot_bsp_wifi.h, 135

iot_wifi_prov_data, 69
 bssid, 70
 password, 70
 security_type, 70
 ssid, 70

IOT_WIFI_PROV_PASSWORD_LEN
 iot_main.h, 194

IOT_WIFI_PROV_SSID_LEN
 iot_main.h, 194

iot_wifi_scan_result_t, 71
 authmode, 71
 bssid, 71
 freq, 72
 rss, 72
 ssid, 72

isconnected
 MQTTClient, 76

iv
 iot_crypto_cipher_info, 36

iv_len
 iot_crypto_cipher_info, 36

JSON_H
 iot_easysetup_d2d.c, 114

keepalive
 iot_mqtt_client.c, 353

keepAliveInterval
 MQTTClient, 77
 MQTTPacket_connectData, 87

key
 iot_crypto_cipher_info, 36
 iot_net_connection, 56

key_len
 iot_crypto_cipher_info, 36
 iot_net_connection, 56

label
 iot_cloud_prov_data, 27

last_received
 MQTTClient, 77

last_sent
 MQTTClient, 77

LEAVE
 iot_debug.h, 160

len
 MQTTLenString, 84
 MQTTTransport, 92

lenstring
 MQTTString, 90

limit
 noti_data_raw_t::quota, 10

location_id
 iot_cloud_prov_data, 27

lookup_id
 iot_context, 33

LWIP_HTTPD_POST_MAX_PAYLOAD_LEN

iot_easysetup_http.c, 107

MAC_ADDR_BUFFER_SIZE
 iot_easysetup_d2d.c, 114

MAX_CAP_ARG
 st_dev.h, 284

MAX_MESSAGE_HANDLERS
 iot_mqtt_client.h, 223

MAX_NO_OF_REMAINING_LENGTH_BYTES
 iot_mqtt_packet.c, 377

MAX_PACKET_ID
 iot_mqtt_client.h, 223

MAX_PAYLOAD_LENGTH
 iot_easysetup_http.c, 107

MAX_SQNUM
 iot_capability.c, 306

message
 MessageData, 73
 MQTTPacket_willOptions, 89

MessageData, 72
 iot_mqtt_client.h, 223
 message, 73
 topicName, 73

messageHandler
 iot_mqtt_client.h, 223

messageHandlers
 MQTTClient, 77

method
 iot_net_platform_context, 60

min
 iot_mqtt_connect_server.c, 370
 iot_mqtt_deserialize_publish.c, 372

mnid
 iot_devconf_prov_data, 45

mode
 iot_crypto_cipher_info, 36
 iot_wifi_conf, 68

MQTT_BAD_USERNAME_OR_PASSWORD
 iot_mqtt_connect.h, 236

MQTT_BUFFER_OVERFLOW
 iot_mqtt_client.h, 224

MQTT_CLIENTID_REJECTED
 iot_mqtt_connect.h, 236

mqtt_connected
 iot_mqtt_ctx, 53

MQTT_CONNECTION_ACCEPTED
 iot_mqtt_connect.h, 236

MQTT_DISCONNECTED
 iot_mqtt_client.h, 224

MQTT_FAILURE
 iot_mqtt_client.h, 224

MQTT_NOTAUTHORIZED
 iot_mqtt_connect.h, 236

MQTT_SERVER_UNAVAILABLE
 iot_mqtt_connect.h, 236

MQTT_SUCCESS
 iot_mqtt_client.h, 224

MQTT_UNACCEPTABLE_PROTOCOL
 iot_mqtt_connect.h, 236

MQTTCleanSession
iot_mqtt_client.c, 354

MQTTClient, 75
 buf, 76
 buf_size, 76
 cleansession, 76
 command_timeout_ms, 76
 defaultMessageHandler, 76
 defaultUserData, 76
iot_mqtt_client.h, 223
 isconnected, 76
 keepAliveInterval, 77
 last_received, 77
 last_sent, 77
 messageHandlers, 77
 net, 77
 next_packetid, 77
 ping_outstanding, 77
 ping_retry_count, 77
 ping_wait, 77
 readbuf, 77
 readbuf_size, 77

MQTTClient::MessageHandlers, 74
 fp, 74
 topicFilter, 74
 userData, 74

MQTTClientInit
iot_mqtt_client.c, 354
iot_mqtt_client.h, 224

MQTTCloseSession
iot_mqtt_client.c, 355

MQTTConnackData, 78
iot_mqtt_client.h, 224
 rc, 78
 sessionPresent, 78

MQTTConnackFlags, 79
 all, 79
 bits, 79
 reserved, 79
 sessionpresent, 79

MQTTConnect
iot_mqtt_client.c, 356
iot_mqtt_client.h, 225

MQTTConnectFlags, 80
 all, 80
 bits, 81
 cleansession, 81
 int, 81
 password, 81
 username, 81
 will, 81
 willQoS, 81
 willRetain, 81

MQTTConnectWithResults
iot_mqtt_client.c, 356
iot_mqtt_client.h, 226

MQTTDeserialize_ack
iot_mqtt_deserialize_publish.c, 372

iot_mqtt_packet.h, 245

MQTTDeserialize_connack
iot_mqtt_connect.h, 236
iot_mqtt_connect_client.c, 365

MQTTDeserialize_connect
iot_mqtt_connect.h, 236
iot_mqtt_connect_server.c, 370

MQTTDeserialize_publish
iot_mqtt_deserialize_publish.c, 373
iot_mqtt_publish.h, 254

MQTTDeserialize_suback
iot_mqtt_subscribe.h, 260
iot_mqtt_subscribe_client.c, 389

MQTTDeserialize_subscribe
iot_mqtt_subscribe.h, 261
iot_mqtt_subscribe_server.c, 392

MQTTDeserialize_unsuback
iot_mqtt_unsubscribe.h, 265
iot_mqtt_unsubscribe_client.c, 394

MQTTDeserialize_unsubscribe
iot_mqtt_unsubscribe.h, 265
iot_mqtt_unsubscribe_server.c, 397

MQTTDisconnect
iot_mqtt_client.c, 357
iot_mqtt_client.h, 227

MQTTFormat_toClientString
iot_mqtt_format.h, 241

MQTTFormat_toServerString
iot_mqtt_format.h, 241

MQTTHeader, 82
 bits, 82
 byte, 82
 dup, 82
 qos, 83
 retain, 83
 type, 83

MQTTLenString, 83
 data, 83
 len, 84

MQTTMessage, 84
 dup, 84
 id, 84
iot_mqtt_client.h, 224
 payload, 85
 payloadlen, 85
 qos, 85
 retained, 85

MQTTPACKET_BUFFER_TOO_SHORT
iot_mqtt_packet.h, 245

MQTTPacket_checkVersion
iot_mqtt_connect_server.c, 371

MQTTPacket_connectData, 85
 cleansession, 87
 clientID, 87
 keepAliveInterval, 87
 MQTTVersion, 87
 password, 87
 struct_id, 87

struct_version, 87
username, 87
will, 87
willFlag, 87
MQTTPacket_connectData_initializer
 iot_mqtt_connect.h, 235
MQTTPacket_decode
 iot_mqtt_packet.c, 377
 iot_mqtt_packet.h, 246
MQTTPacket_decodeBuf
 iot_mqtt_packet.c, 378
 iot_mqtt_packet.h, 246
MQTTPacket_encode
 iot_mqtt_packet.c, 378
 iot_mqtt_packet.h, 246
MQTTPacket_equals
 iot_mqtt_packet.c, 378
 iot_mqtt_packet.h, 247
MQTTPacket_getName
 iot_mqtt_format.c, 374
 iot_mqtt_format.h, 241
MQTTPacket_len
 iot_mqtt_packet.c, 379
 iot_mqtt_packet.h, 247
MQTTPacket_msgTypesToString
 iot_mqtt_packet.c, 379
 iot_mqtt_packet.h, 248
MQTTPacket_names
 iot_mqtt_format.c, 376
MQTTPacket_read
 iot_mqtt_packet.c, 379
 iot_mqtt_packet.h, 248
MQTTPACKET_READ_COMPLETE
 iot_mqtt_packet.h, 245
MQTTPACKET_READ_ERROR
 iot_mqtt_packet.h, 245
MQTTPacket_readnb
 iot_mqtt_packet.c, 380
 iot_mqtt_packet.h, 248
MQTTPacket_willOptions, 88
 message, 89
 qos, 89
 retained, 89
 struct_id, 89
 struct_version, 89
 topicName, 89
MQTTPacket_willOptions_initializer
 iot_mqtt_connect.h, 235
MQTTPublish
 iot_mqtt_client.c, 358
 iot_mqtt_client.h, 228
MQTTSerialize_ack
 iot_mqtt_packet.h, 249
 iot_mqtt_serialize_publish.c, 385
MQTTSerialize_ack_size
 iot_mqtt_packet.h, 249
 iot_mqtt_serialize_publish.c, 385
MQTTSerialize_connack

 iot_mqtt_connect.h, 237
 iot_mqtt_connect_server.c, 371
MQTTSerialize_connect
 iot_mqtt_connect.h, 237
 iot_mqtt_connect_client.c, 366
MQTTSerialize_connect_size
 iot_mqtt_connect.h, 238
 iot_mqtt_connect_client.c, 366
MQTTSerialize_connectLength
 iot_mqtt_connect_client.c, 367
MQTTSerialize_disconnect
 iot_mqtt_connect.h, 238
 iot_mqtt_connect_client.c, 367
MQTTSerialize_disconnect_size
 iot_mqtt_connect.h, 239
 iot_mqtt_connect_client.c, 368
MQTTSerialize_pingreq
 iot_mqtt_connect.h, 239
 iot_mqtt_connect_client.c, 368
MQTTSerialize_pingreq_size
 iot_mqtt_connect.h, 240
 iot_mqtt_connect_client.c, 369
MQTTSerialize_puback
 iot_mqtt_publish.h, 255
 iot_mqtt_serialize_publish.c, 385
MQTTSerialize_pubcomp
 iot_mqtt_publish.h, 255
 iot_mqtt_serialize_publish.c, 386
MQTTSerialize_publish
 iot_mqtt_publish.h, 255
 iot_mqtt_serialize_publish.c, 386
MQTTSerialize_publish_header
 iot_mqtt_publish.h, 256
 iot_mqtt_serialize_publish.c, 387
MQTTSerialize_publish_size
 iot_mqtt_publish.h, 256
 iot_mqtt_serialize_publish.c, 387
MQTTSerialize_publishLength
 iot_mqtt_serialize_publish.c, 388
MQTTSerialize_pubrel
 iot_mqtt_publish.h, 257
 iot_mqtt_serialize_publish.c, 388
MQTTSerialize_suback
 iot_mqtt_subscribe.h, 261
 iot_mqtt_subscribe_server.c, 393
MQTTSerialize_subscribe
 iot_mqtt_subscribe.h, 263
 iot_mqtt_subscribe_client.c, 390
MQTTSerialize_subscribe_size
 iot_mqtt_subscribe.h, 263
 iot_mqtt_subscribe_client.c, 391
MQTTSerialize_subscribeLength
 iot_mqtt_subscribe_client.c, 391
MQTTSerialize_unsuback
 iot_mqtt_unsubscribe.h, 266
 iot_mqtt_unsubscribe_server.c, 398
MQTTSerialize_unsubscribe
 iot_mqtt_unsubscribe.h, 266

iot_mqtt_unsubscribe_client.c, 394
MQTTSerialize_unsubscribe_size
 iot_mqtt_unsubscribe.h, 267
 iot_mqtt_unsubscribe_client.c, 395
MQTTSerialize_unsubscribeLength
 iot_mqtt_unsubscribe_client.c, 396
MQTTSerialize_zero
 iot_mqtt_connect_client.c, 369
MQTTSetMessageHandler
 iot_mqtt_client.c, 359
 iot_mqtt_client.h, 229
MQTTString, 90
 cstring, 90
 lenstring, 90
MQTTString_initializer
 iot_mqtt_packet.h, 245
MQTTStringFormat_ack
 iot_mqtt_format.c, 374
 iot_mqtt_format.h, 242
MQTTStringFormat_connack
 iot_mqtt_format.c, 375
 iot_mqtt_format.h, 242
MQTTStringFormat_connect
 iot_mqtt_format.c, 375
 iot_mqtt_format.h, 242
MQTTStringFormat_publish
 iot_mqtt_format.c, 375
 iot_mqtt_format.h, 242
MQTTStringFormat_suback
 iot_mqtt_format.c, 375
 iot_mqtt_format.h, 242
MQTTStringFormat_subscribe
 iot_mqtt_format.c, 375
 iot_mqtt_format.h, 242
MQTTStringFormat_unsubscribe
 iot_mqtt_format.c, 375
 iot_mqtt_format.h, 243
MQTTstrlen
 iot_mqtt_packet.c, 380
 iot_mqtt_packet.h, 249
MQTTSubackData, 91
 grantedQoS, 91
 iot_mqtt_client.h, 224
MQTTSubscribe
 iot_mqtt_client.c, 360
 iot_mqtt_client.h, 230
MQTTSubscribeWithResults
 iot_mqtt_client.c, 360
 iot_mqtt_client.h, 230
MQTTTransport, 91
 getfn, 92
 len, 92
 multiplier, 92
 rem_len, 92
 sck, 92
 state, 92
MQTTUnsubscribe
 iot_mqtt_client.c, 361
 iot_mqtt_client.h, 231
MQTTVersion
 MQTTPacket_connectData, 87
MQTTYield
 iot_mqtt_client.c, 363
 iot_mqtt_client.h, 233
msgTypes
 iot_mqtt_packet.h, 245
multiplier
 MQTTTransport, 92
name
 iot_crypto_pk_funcs, 42
net
 iot_mqtt_ctx, 53
 MQTTClient, 77
new_reged
 iot_registered_data, 65
next
 iot_cap_cmd_set_list, 18
 iot_cap_handle_list, 23
next_packetid
 MQTTClient, 77
NEXT_STATE_TIMEOUT_MS
 iot_main.c, 319
NOSTACKTRACE
 iot_mqtt_stacktrace.h, 259
noti_cb
 iot_context, 33
noti_data_raw_t, 93
 quota, 93
 rate_limit, 94
noti_data_raw_t::quota, 9
 limit, 10
 used, 10
noti_data_raw_t::rate_limit, 10
 count, 11
 remainingTime, 11
 sequenceNumber, 11
 threshold, 11
noti_filter
 iot_mqtt_ctx, 54
noti_usr_data
 iot_context, 33
num_args
 iot_cap_cmd_data_t, 14
number
 iot_cap_val_t, 25
opt
 iot_state_data, 66
OVF_BIT_BUTTON
 iot_easysetup.h, 167
OVF_BIT JUSTWORKS
 iot_easysetup.h, 167
OVF_BIT_MAX_FEATURE
 iot_easysetup.h, 167
OVF_BIT_PIN
 iot_easysetup.h, 167

OVF_BIT_QR
 iot_easystep.h, 167
ownership_validation_feature
 iot_easystep.h, 166
ownership_validation_type
 iot_devconf_prov_data, 46

param
 iot_command, 28
pass
 iot_wifi_conf, 69
password
 iot_wifi_prov_data, 70
 MQTTConnectFlags, 81
 MQTTPacket_connectData, 87
payload
 iot_easystep_payload, 50
 MQTTMessage, 85
payloadlen
 MQTTMessage, 85
pin
 iot_context, 33
 iot_pin_t, 64
PIN_SIZE
 iot_easystep_d2d.c, 114
ping_outstanding
 MQTTClient, 77
ping_retry_count
 MQTTClient, 77
ping_wait
 MQTTClient, 77
PINGREQ
 iot_mqtt_packet.h, 245
PINGRESP
 iot_mqtt_packet.h, 245
pk_type
 iot_devconf_prov_data, 46
port
 iot_net_connection, 56
 url_parse_t, 95
post_cgi_cmds
 iot_easystep_http.c, 111
protocol
 url_parse_t, 95
prov_data
 iot_context, 33
pub_queue
 iot_context, 33
PUBACK
 iot_mqtt_packet.h, 245
PUBCOMP
 iot_mqtt_packet.h, 245
pubkey
 iot_crypto_keypair, 40
 iot_crypto_pk_info, 43
pubkey_len
 iot_crypto_pk_info, 43
PUBLISH
 iot_mqtt_packet.h, 245

PUBREC
 iot_mqtt_packet.h, 245
PUBREL
 iot_mqtt_packet.h, 245

QoS
 iot_mqtt_client.h, 224
qos
 MQTTHeader, 83
 MQTTMessage, 85
 MQTTPacket_willOptions, 89

QOS0
 iot_mqtt_client.h, 224
QOS1
 iot_mqtt_client.h, 224
QOS2
 iot_mqtt_client.h, 224

quota
 noti_data_raw_t, 93

rate_limit
 noti_data_raw_t, 94

raw
 iot_noti_data_t, 62

rc
 MQTTConnackData, 78

read
 iot_net_interface, 58

read_count
 iot_net_platform_context, 60

readbuf
 iot_mqtt_ctx, 54
 MQTTClient, 77

readbuf_size
 MQTTClient, 77

readChar
 iot_mqtt_packet.c, 381
 iot_mqtt_packet.h, 250

readInt
 iot_mqtt_packet.c, 381
 iot_mqtt_packet.h, 250

readMQTTLenString
 iot_mqtt_packet.c, 381
 iot_mqtt_packet.h, 250

reged_cli
 iot_context, 33

rem_len
 MQTTTransport, 92

remainingTime
 noti_data_raw_t::rate_limit, 11

reported_stat
 iot_context, 33

req_state
 iot_context, 34

reserved
 MQTTConnackFlags, 79

retain
 MQTTHeader, 83

retained

MQTTMessage, 85
 MQTTPacket_willOptions, 89
 returnCode
 iot_mqtt_client.h, 224
 room_id
 iot_cloud_prov_data, 27
 rssi
 iot_wifi_scan_result_t, 72
 s_pubkey
 iot_crypto_ecdh_params, 38
 scan_num
 iot_context, 34
 scan_result
 iot_context, 34
 sck
 MQTTTransport, 92
 seckey
 iot_crypto_keypair, 40
 iot_crypto_pk_info, 44
 seckey_len
 iot_crypto_pk_info, 44
 security_type
 iot_wifi_prov_data, 70
 select
 iot_net_interface, 58
 sem
 iot_os_mutex, 63
 sequenceNumber
 noti_data_raw_t::rate_limit, 11
 server_fd
 iot_net_platform_context, 61
 sessionPresent
 MQTTConnackData, 78
 sessionpresent
 MQTTConnackFlags, 79
 setupid
 iot_devconf_prov_data, 46
 show_status
 iot_net_interface, 58
 sign
 iot_crypto_ed25519_keypair, 39
 iot_crypto_pk_funcs, 42
 socket
 iot_net_platform_context, 61
 ssid
 iot_wifi_conf, 69
 iot_wifi_prov_data, 70
 iot_wifi_scan_result_t, 72
 ssl
 iot_net_platform_context, 61
 SSL_library_init
 iot_net_openssl.c, 403
 st-device-sdk-c/doc/mainpage.dox, 95
 st-device-sdk-c/src/crypto/iot_crypto_ed25519.c, 95
 st-device-sdk-c/src/crypto/mbedtls/iot_crypto_mbedtls.c,
 98
 st-device-sdk-c/src/crypto/ss/iot_crypto_ss.c, 104
 st-device-sdk-c/src/easysetup/http/iot_easysetup_http.c,
 106
 st-device-sdk-c/src/easysetup/iot_easysetup_crypto.c,
 111
 st-device-sdk-c/src/easysetup/iot_easysetup_d2d.c,
 113
 st-device-sdk-c/src/easysetup/iot_easysetup_st_mqtt.c,
 116
 st-device-sdk-c/src/include/bsp/iot_bsp_debug.h, 118
 st-device-sdk-c/src/include/bsp/iot_bsp_fs.h, 120
 st-device-sdk-c/src/include/bsp/iot_bsp_nv_data.h, 126
 st-device-sdk-c/src/include/bsp/iot_bsp_random.h, 128
 st-device-sdk-c/src/include/bsp/iot_bsp_system.h, 129
 st-device-sdk-c/src/include/bsp/iot_bsp_wifi.h, 132
 st-device-sdk-c/src/include/iot_capability.h, 137
 st-device-sdk-c/src/include/iot_crypto.h, 140
 st-device-sdk-c/src/include/iot_crypto_internal.h, 157
 st-device-sdk-c/src/include/iot_debug.h, 157
 st-device-sdk-c/src/include/iot_easysetup.h, 161
 st-device-sdk-c/src/include/iot_error.h, 169
 st-device-sdk-c/src/include/iot_internal.h, 171
 st-device-sdk-c/src/include/iot_jwt.h, 189
 st-device-sdk-c/src/include/iot_main.h, 191
 st-device-sdk-c/src/include/iot_net.h, 196
 st-device-sdk-c/src/include/iot_nv_data.h, 198
 st-device-sdk-c/src/include/iot_util.h, 213
 st-device-sdk-c/src/include/iot_uuid.h, 218
 st-device-sdk-c/src/include/mqtt/iot_mqtt_client.h, 220
 st-device-sdk-c/src/include/mqtt/iot_mqtt_connect.h,
 234
 st-device-sdk-c/src/include/mqtt/iot_mqtt_format.h, 240
 st-device-sdk-c/src/include/mqtt/iot_mqtt_packet.h, 243
 st-device-sdk-c/src/include/mqtt/iot_mqtt_publish.h, 253
 st-device-sdk-c/src/include/mqtt/iot_mqtt_stacktrace.h,
 257
 st-device-sdk-c/src/include/mqtt/iot_mqtt_subscribe.h,
 260
 st-device-sdk-c/src/include/mqtt/iot_mqtt_unsubscribe.h,
 264
 st-device-sdk-c/src/include/os/iot_os_util.h, 268
 st-device-sdk-c/src/include/st_dev.h, 282
 st-device-sdk-c/src/include/st_dev_version.h, 295
 st-device-sdk-c/src/iot_api.c, 296
 st-device-sdk-c/src/iot_capability.c, 305
 st-device-sdk-c/src/iot_crypto.c, 313
 st-device-sdk-c/src/iot_jwt.c, 316
 st-device-sdk-c/src/iot_main.c, 318
 st-device-sdk-c/src/iot_mqtt.c, 321
 st-device-sdk-c/src/iot_nv_data.c, 327
 st-device-sdk-c/src/iot_util.c, 344
 st-device-sdk-c/src/iot_uuid.c, 349
 st-device-sdk-c/src/mqtt/client/iot_mqtt_client.c, 351
 st-device-sdk-c/src/mqtt/packet/iot_mqtt_connect_client.c,
 365
 st-device-sdk-c/src/mqtt/packet/iot_mqtt_connect_server.c,
 370
 st-device-sdk-c/src/mqtt/packet/iot_mqtt_deserialize_publish.c,
 372

st-device-sdk-c/src/mqtt/packet/iot_mqtt_format.c, 373
st-device-sdk-c/src/mqtt/packet/iot_mqtt_packet.c, 376
st-device-sdk-c/src/mqtt/packet/iot_mqtt_serialize_publish.c, 384
st-device-sdk-c/src/mqtt/packet/iot_mqtt_subscribe_client.c, 389
st-device-sdk-c/src/mqtt/packet/iot_mqtt_subscribe_server.c, 392
st-device-sdk-c/src/mqtt/packet/iot_mqtt_unsubscribe_client.c, 394
st-device-sdk-c/src/mqtt/packet/iot_mqtt_unsubscribe_server.c, 396
st-device-sdk-c/src/port/net/mbedtls/iot_net_mbedtls.c, 398
st-device-sdk-c/src/port/net/mbedtls/iot_net_platform.h, 399
st-device-sdk-c/src/port/net/openssl/iot_net_openssl.c, 401
st-device-sdk-c/src/port/net/openssl/iot_net_platform.h, 400
st_cap_attr_create_int
 iot_capability.c, 308
 st_dev.h, 287
st_cap_attr_create_number
 iot_capability.c, 308
 st_dev.h, 288
st_cap_attr_create_string
 iot_capability.c, 309
 st_dev.h, 288
st_cap_attr_create_string_array
 iot_capability.c, 309
 st_dev.h, 289
st_cap_attr_free
 iot_capability.c, 310
 st_dev.h, 289
st_cap_attr_send
 iot_capability.c, 310
 st_dev.h, 290
st_cap_cmd_cb
 st_dev.h, 285
st_cap_cmd_set_cb
 iot_capability.c, 311
 st_dev.h, 290
st_cap_handle_init
 iot_capability.c, 311
 st_dev.h, 291
st_cap_init_cb
 st_dev.h, 285
st_cap_noti_cb
 st_dev.h, 285
st_conn_cleanup
 iot_main.c, 319
 st_dev.h, 291
st_conn_init
 iot_main.c, 319
 st_dev.h, 292
st_conn_ownership_confirm
 iot_easystatus_d2d.c, 116
 st_dev.h, 293
 st_conn_set_noti_cb
 iot_capability.c, 313
 st_dev.h, 294
 st_conn_start
 iot_main.c, 320
 st_dev.h, 294
 st_dev.h
 IOT_CAP_HANDLE, 284
 iot_cap_val_type, 286
 IOT_CAP_VAL_TYPE_INT_OR_NUM, 286
 IOT_CAP_VAL_TYPE_INTEGER, 286
 IOT_CAP_VAL_TYPE_NUMBER, 286
 IOT_CAP_VAL_TYPE_STR_ARRAY, 286
 IOT_CAP_VAL_TYPE_STRING, 286
 iot_cap_val_type_t, 284
 IOT_CAP_VAL_TYPE_UNKNOWN, 286
 IOT_CTX, 285
 IOT_EVENT, 285
 iot_noti_type, 286
 IOT_NOTI_TYPE_DEV_DELETED, 286
 IOT_NOTI_TYPE_QUOTA_REACHED, 286
 IOT_NOTI_TYPE_RATE_LIMIT, 286
 iot_noti_type_t, 285
 IOT_NOTI_TYPE_UNKNOWN, 286
 iot_pin_t, 285
 iot_stat_lv, 286
 IOT_STAT_LV_DONE, 287
 IOT_STAT_LV_FAIL, 287
 IOT_STAT_LV_START, 287
 IOT_STAT_LV_STAY, 286
 iot_stat_lv_t, 285
 iot_status, 287
 IOT_STATUS_ALL, 287
 IOT_STATUS_CONNECTING, 287
 IOT_STATUS_IDLE, 287
 IOT_STATUS_NEED_INTERACT, 287
 IOT_STATUS_PROVISIONING, 287
 iot_status_t, 285
 MAX_CAP_ARG, 284
 st_cap_attr_create_int, 287
 st_cap_attr_create_number, 288
 st_cap_attr_create_string, 288
 st_cap_attr_create_string_array, 289
 st_cap_attr_free, 289
 st_cap_attr_send, 290
 st_cap_cmd_cb, 285
 st_cap_cmd_set_cb, 290
 st_cap_handle_init, 291
 st_cap_init_cb, 285
 st_cap_noti_cb, 285
 st_conn_cleanup, 291
 st_conn_init, 292
 st_conn_ownership_confirm, 293
 st_conn_set_noti_cb, 294
 st_conn_start, 294
 st_status_cb, 286
 st_dev_version.h

STDK_VERSION, 295
 STDK_VERSION_CODE, 295
 VER_MAJOR, 295
 VER_MINOR, 296
 VER_PATCH, 296
 st_status_cb
 st_dev.h, 286
 state
 MQTTTransport, 92
 state_timer
 iot_context, 34
 status_cb
 iot_context, 34
 status_maps
 iot_context, 34
 status_usr_data
 iot_context, 34
 STDK_VERSION
 st_dev_version.h, 295
 STDK_VERSION_CODE
 st_dev_version.h, 295
 step
 iot_easystatus_payload, 50
 str_num
 iot_cap_val_t, 25
 string
 iot_cap_unit_t, 23
 iot_cap_val_t, 25
 strings
 iot_cap_val_t, 25
 struct_id
 MQTTPacket_connectData, 87
 MQTTPacket_willOptions, 89
 struct_version
 MQTTPacket_connectData, 87
 MQTTPacket_willOptions, 89
 SUBACK
 iot_mqtt_packet.h, 245
 SUBFAIL
 iot_mqtt_client.h, 224
 SUBSCRIBE
 iot_mqtt_packet.h, 245
 t_seckey
 iot_crypto_ecdh_params, 38
 threshold
 noti_data_raw_t::rate_limit, 11
 topicFilter
 MQTTClient::MessageHandlers, 74
 topicName
 MessageData, 73
 MQTTPacket_willOptions, 89
 type
 iot_cap_unit_t, 24
 iot_cap_val_t, 25
 iot_crypto_cipher_info, 36
 iot_crypto_pk_info, 44
 iot_noti_data_t, 63
 MQTTHeader, 83
 UNSUBACK
 iot_mqtt_packet.h, 245
 UNSUBSCRIBE
 iot_mqtt_packet.h, 245
 updated
 iot_registered_data, 65
 url
 iot_net_connection, 56
 URL_BUFFER_SIZE
 iot_easysetup_d2d.c, 114
 url_parse_t, 94
 domain, 94
 port, 95
 protocol, 95
 used
 noti_data_raw_t::quota, 10
 userData
 MQTTClient::MessageHandlers, 74
 username
 MQTTConnectFlags, 81
 MQTTPacket_connectData, 87
 usr_data
 iot_cap_cmd_set, 16
 usr_events
 iot_context, 34
 VER_MAJOR
 st_dev_version.h, 295
 VER_MINOR
 st_dev_version.h, 296
 VER_PATCH
 st_dev_version.h, 296
 verify
 iot_crypto_pk_funcs, 42
 vid
 iot_devconf_prov_data, 46
 waitfor
 iot_mqtt_client.c, 364
 wifi
 iot_device_prov_data, 49
 WIFIINFO_BUFFER_SIZE
 iot_easysetup_d2d.c, 114
 will
 MQTTConnectFlags, 81
 MQTTPacket_connectData, 87
 willFlag
 MQTTPacket_connectData, 87
 willQoS
 MQTTConnectFlags, 81
 willRetain
 MQTTConnectFlags, 81
 write
 iot_net_interface, 59
 writeChar
 iot_mqtt_packet.c, 382
 iot_mqtt_packet.h, 251
 writeCString
 iot_mqtt_packet.c, 382

iot_mqtt_packet.h, 251
writeInt
 iot_mqtt_packet.c, 383
 iot_mqtt_packet.h, 252
writeMQTTString
 iot_mqtt_packet.c, 383
 iot_mqtt_packet.h, 252