ECx00U&EGx00U 系列

QuecOpen MQTT API 参考手册

LTE Standard 模块系列

版本：1.0

日期：2021-09-27

状态：受控文件

上海移远通信技术股份有限公司（以下简称“移远通信”）始终以为客户提供最及时、最全面的服务为宗旨。如需任何帮助，请随时联系我司上海总部，联系方式如下：

上海移远通信技术股份有限公司
上海市闵行区田林路 1016 号科技绿洲 3 期（B 区）5 号楼　　邮编：200233
电话：+86 21 5108 6236　　邮箱：info@quectel.com

或联系我司当地办事处，详情请登录：http://www.quectel.com/cn/support/sales.htm。

如需技术支持或反馈我司技术文档中的问题，请随时登陆网址：
http://www.quectel.com/cn/support/technical.htm 或发送邮件至：support@quectel.com。

前言

移远通信提供该文档内容以支持客户的产品设计。客户须按照文档中提供的规范、参数来设计产品。同时，您理解并同意，移远通信提供的参考设计仅作为示例。您同意在设计您目标产品时使用您独立的分析、评估和判断。在使用本文档所指导的任何硬软件或服务之前，请仔细阅读本声明。您在此承认并同意，尽管移远通信采取了商业范围内的合理努力来提供尽可能好的体验，但本文档和其所涉及服务是在“可用”基础上提供给您的。移远通信可在未事先通知的情况下，自行决定随时增加、修改或重述本文档。

使用和披露限制

许可协议

除非移远通信特别授权，否则我司所提供硬软件、材料和文档的接收方须对接收的内容保密，不得将其用于除本项目的实施与开展以外的任何其他目的。

版权声明

移远通信产品和本协议项下的第三方产品可能包含受移远通信或第三方材料、硬软件和文档版权保护的相关资料。除非事先得到书面同意，否则您不得获取、使用、向第三方披露我司所提供的文档和信息，或对此类受版权保护的资料进行复制、转载、抄袭、出版、展示、翻译、分发、合并、修改，或创造其衍生作品。移远通信或第三方对受版权保护的资料拥有专有权，不授予或转让任何专利、版权、商标或服务商标权的许可。为避免歧义，除了正常的非独家、免版税的产品使用许可，任何形式的购买都不可被视为授予许可。对于任何违反保密义务、未经授权使用或以其他非法形式恶意使用所述文档和信息的违法侵权行为，移远通信有权追究法律责任。

商标

除另行规定，本文档中的任何内容均不授予在广告、宣传或其他方面使用移远通信或第三方的任何商标、商号及名称，或其缩略语，或其仿冒品的权利。

第三方权利

您理解本文档可能涉及一个或多个属于第三方的硬软件和文档（“第三方材料”）。您对此类第三方材料的使用应受本文档的所有限制和义务约束。

移远通信针对第三方材料不做任何明示或暗示的保证或陈述，包括但不限于任何暗示或法定的适销性或特定用途的适用性、平静受益权、系统集成、信息准确性以及与许可技术或被许可人使用许可技术相关的不侵犯任何第三方知识产权的保证。本协议中的任何内容都不构成移远通信对任何移远通信产品或任何其他硬软件、设备、工具、信息或产品的开发、增强、修改、分销、营销、销售、提供销售或以其他方式维持生产的陈述或保证。此外，移远通信免除因交易过程、使用或贸易而产生的任何和所有保证。

免责声明

1) 移远通信不承担任何因未能遵守有关操作或设计规范而造成损害的责任。

2) 移远通信不承担因本文档中的任何因不准确、遗漏、或使用本文档中的信息而产生的任何责任。

3) 移远通信尽力确保开发中功能的完整性、准确性、及时性，但不排除上述功能错误或遗漏的可能。除非另有协议规定，否则移远通信对开发中功能的使用不做任何暗示或法定的保证。在适用法律允许的最大范围内，移远通信不对任何因使用开发中功能而遭受的损害承担责任，无论此类损害是否可以预见。

4) 移远通信对第三方网站及第三方资源的信息、内容、广告、商业报价、产品、服务和材料的可访问性、安全性、准确性、可用性、合法性和完整性不承担任何法律责任。

文档历史

修订记录

版本	日期	作者	变更表述
-	2021-08-20	Herry GENG/ Fei XUE	文档创建
1.0	2021-09-27	Herry GENG/ Fei XUE	受控版本

目录

表格索引

1 引言

移远通信 LTE Standard ECx00U 系列和 EGx00U 模块支持 QuecOpen®方案；QuecOpen®是基于 RTOS 的嵌入式开发平台，可简化 IoT 应用的软件设计和开发过程。有关 QuecOpen®的详细信息，请参考*文档 [1]*。

本文档主要介绍在 QuecOpen®方案下，移远通信 ECx00U 系列和 EGx00U 模块的 MQTT 相关 API、调用流程以及示例。

1.1. 适用模块

表 1：适用模块

模块系列	模块
ECx00U	EC200U 系列
	EC600U 系列
EGx00U	EG500U-CN
	EG700U-CN

2 MQTT 相关接口调用流程

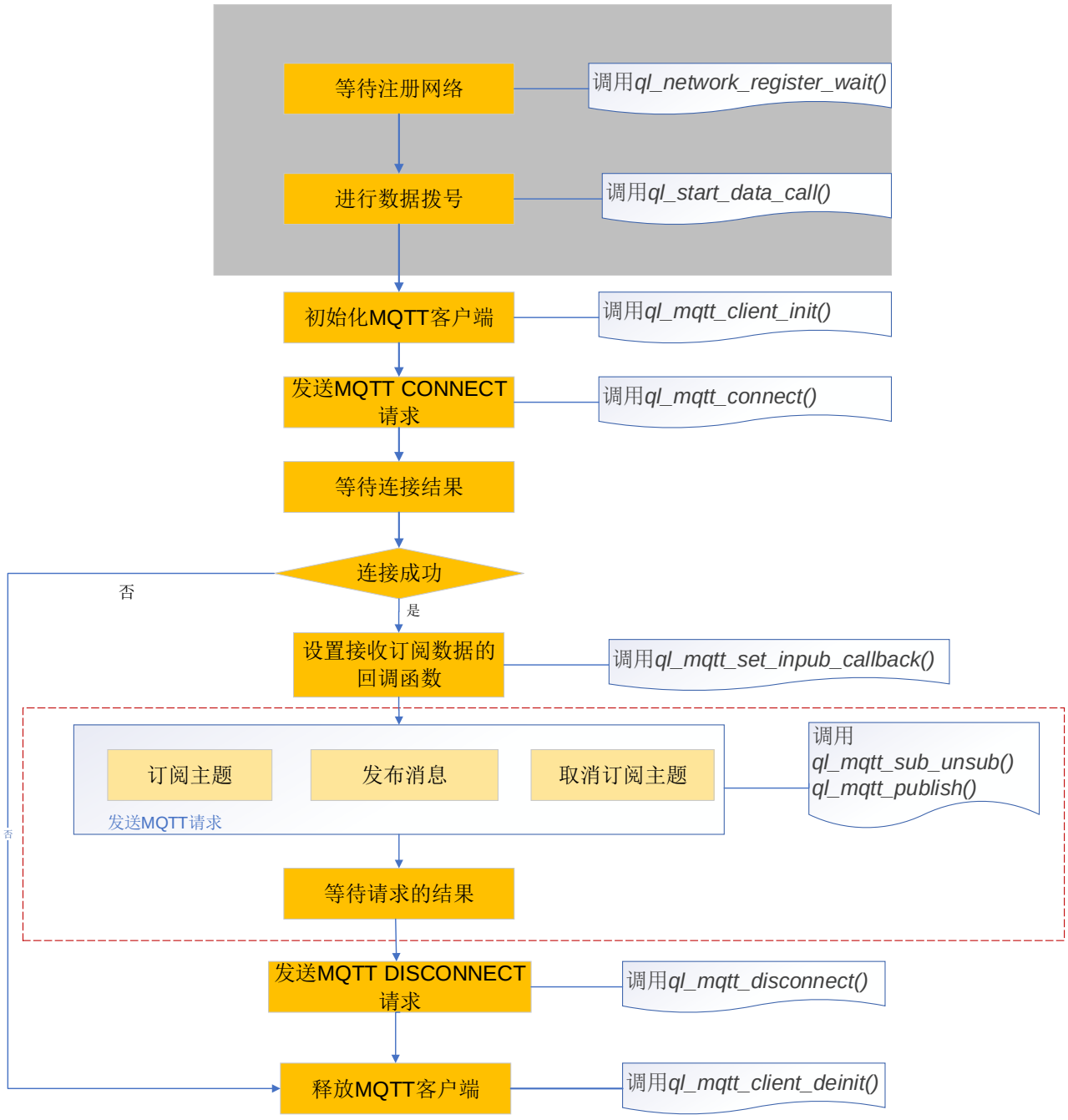

图 1：MQTT 接口调用流程

注网及数据拨号成功后，主要按照以下步骤调用 MQTT 相关的 API：

1. 通过 *ql_mqtt_client_init()*创建了一个 MQTT 客户端句柄；
2. 通过 *ql_mqtt_connect()*向服务器发送 CONNECT 请求，在 MQTT 客户端与服务器之间建立 MQTT 会话连接；
3. 通过 *ql_mqtt_unsub()*订阅和取消订阅主题；
4. 通过 *ql_mqtt_publish()*发布指定主题的消息；
5. 可通过 *ql_mqtt_disconnect()*向服务器发送 DISCONNECT 请求，断开 MQTT 客户端与服务器之间的 MQTT 会话连接。

备注

MQTT 服务基于指定的数据通道进行数据收发，因此在发送 MQTT 请求之前，首先需注网并拨号以建立数据通道（如***图 1*** 中的灰底部分）。若在其他任务中已经建立了数据通道，可跳过注网拨号步骤，直接调用 *ql_mqtt_client_init()*创建 MQTT 客户端句柄。数据拨号相关函数的详情可参考***文档 [2]***。

3 MQTT API

3.1. 头文件

MQTT API 的头文件为 *ql_mqttclient.h*，位于*\components\ql-kernel\inc* 目录下。

3.2. 函数概览

表 2：函数概览

函数	说明
ql_mqtt_client_init()	初始化 MQTT 客户端资源并创建一个新的 MQTT 客户端句柄
ql_mqtt_connect()	向服务器发送 CONNECT 请求，在客户端与服务器之间建立 MQTT 会话连接
ql_mqtt_publish()	发布指定主题的消息
ql_mqtt_sub_unsub()	订阅/取消订阅指定主题
ql_mqtt_disconnect()	向服务器发送 DISCONNECT 请求，断开 MQTT 客户端与服务器之间的 MQTT 会话连接
ql_mqtt_set_inpub_callback()	设置接收服务器发布消息的处理回调函数
ql_mqtt_client_is_connected()	查询客户端与服务器之间的会话连接是否已经建立
ql_mqtt_client_deinit()	释放 MQTT 客户端资源
ql_mqtt_onenet_generate_auth_token()	生成 OneNET 平台需要的密码

3.3. API 详解

3.3.1. ql_mqtt_client_init

该函数用于初始化 MQTT 客户端资源并创建一个新的 MQTT 客户端句柄。

- **函数原型**

```
int ql_mqtt_client_init(mqtt_client_t *client, int cid)
```

- **参数**

client：
[Out] MQTT 客户端句柄。

cid：
[In] 数据通道号。

- **返回值**

详见***第 3.3.1.1 章***。

3.3.1.1. mqtt_error_code_e

MQTT 错误码类型枚举定义如下：

```
typedef enum{
    /*成功*/
    MQTTCLIENT_SUCCESS                  = 0,
    /*无效参数*/
    MQTTCLIENT_INVALID_PARAM            = -1,
    /*操作未完成，结果等待异步通知*/
    MQTTCLIENT_WOUNDBLOCK                = -2,
    /*内存不足*/
    MQTTCLIENT_OUT_OF_MEM               = -3,
    /*内存分配出错*/
    MQTTCLIENT_ALLOC_FAIL               = -4,
    /*TCP 连接建立失败*/
    MQTTCLIENT_TCP_CONNECT_FAIL          = -5,
    /*MQTT 会话没有建立*/
    MQTTCLIENT_NOT_CONNECT               = -6,
    /*发送请求失败*/
```

```
    MQTTCLIENT_SEND_PKT_FAIL                = -7,
    /*错误请求*/
    MQTTCLIENT_BAD_REQUEST                  = -8,
    /*超时错误*/
    MQTTCLIENT_TIMEOUT                      = -9,
}mqtt_error_code_e
```

- **参数**

参数	描述
MQTTCLIENT_SUCCESS	函数执行成功
MQTTCLIENT_INVALID_PARAM	无效参数
MQTTCLIENT_WOUNDBLOCK	操作未完成，结果等待异步通知
MQTTCLIENT_OUT_OF_MEM	内存不足
MQTTCLIENT_ALLOC_FAIL	内存分配出错
MQTTCLIENT_TCP_CONNECT_FAIL	TCP 连接建立失败
MQTTCLIENT_NOT_CONNECT	MQTT 会话没有建立
MQTTCLIENT_SEND_PKT_FAIL	发送请求失败
MQTTCLIENT_BAD_REQUEST	错误请求
MQTTCLIENT_TIMEOUT	超时错误

3.3.2. ql_mqtt_connect

该函数用于向服务器发送 CONNECT 请求，在客户端与服务器之间建立 MQTT 会话连接。

- **函数原型**

```
int ql_mqtt_connect(mqtt_client_t *client, const char *host, mqtt_connection_cb_t cb, void *arg, const struct mqtt_connect_client_info_t *client_info, mqtt_state_exception_cb_t exp_cb)
```

- **参数**

client：
[In] MQTT 客户端句柄。由 *ql_mqtt_client_init()*获取，详情参考***第 3.3.1 章***。

host：
[In] MQTT 服务器地址；以 mqtt://或者 mqtts://开头。例如，mqtt://220.180.239.212:8306 或 mqtts://220.180.239.212:8307。

cb：
[In] CONNECT 请求结果回调函数。详情参考***第 3.3.2.1 章***。

arg：
[In] CONNECT 请求结果回调函数的回调参数。

client_info：
[In] MQTT 客户端的信息。详情参考***第 3.3.2.2 章***。

exp_cb：
[In] MQTT 会话连接异常断开的回调函数。详情参考***第 3.3.2.3 章***。

- **返回值**

详见***第 3.3.1.1 章***。

3.3.2.1. mqtt_connection_cb_t

该回调函数由内核调用以通知应用层 MQTT CONNECT 请求结果。

- **函数原型**

```
typedef void (*mqtt_connection_cb_t)(mqtt_client_t *client, void *arg, mqtt_connection_status_e status)
```

- **参数**

client：
[In] MQTT 客户端句柄。由 *ql_mqtt_client_init()*获取，详情参考***第 3.3.1 章***。

arg：
[In] 回调参数。由 *ql_mqtt_connect()*传入，详情参考***第 3.3.2 章***。

status：
[In] CONNECT 请求的结果。详情参考***第 3.3.2.4 章***。

- **返回值**

无

3.3.2.2. mqtt_connect_client_info_t

MQTT 客户端的信息定义如下：

```
struct mqtt_connect_client_info_t {    const char *client_id;
    const char *client_id;
    const char* client_user;
    const char* client_pass;
    unsigned short keep_alive;
    const  char*  will_topic;
    const char* will_msg;
    unsigned char will_qos;
    unsigned char will_retain;
    unsigned char clean_session;
    struct mqtt_ssl_config_t    *ssl_cfg;
}
```

- **参数**

类型	参数	描述
const char *	*client_id*	客户端 ID
const char *	*client_user*	CONNECT 请求时携带的用户名；若无，则设置为 NULL
const char *	*client_pass*	CONNECT 请求时携带的密码；若无，则设置为 NULL
unsigned short	*keep_alive*	客户端与服务器之间的保活间隔；单位：秒
const char *	*will_topic*	MQTT 遗嘱的主题；若无，则设置为 NULL
const char*	*will_msg*	MQTT 遗嘱的消息；若无，则设置为 NULL
unsigned char	*will_qos*	MQTT 遗嘱消息的 Qos 等级
unsigned char	*will_retain*	遗嘱消息被服务器发布出去后，*will_retain* 为 True 时，服务器会一直保留这条遗嘱消息；*will_retain* 为 False 时，遗嘱消息会被立即删除
unsigned char	*clean_session*	*0* 表示开启 MQTT 会话复用机制； *1* 表示关闭 MQTT 会话复用机制
struct mqtt_ssl_config_t *	*ssl_cfg*	MQTT SSL 的配置，详情参考***第 3.3.2.5 章***；若无 SSL，则设置为 NULL

3.3.2.3. mqtt_state_exception_cb_t

该回调函数由内核调用以通知应用层 MQTT 会话连接异常断开。例如，掉网或保活超时导致 MQTT 会话连接断开。

- **函数原型**

```
typedef void(*mqtt_state_exception_cb_t)(mqtt_client_t *client)
```

- **参数**

client：
[In] MQTT 客户端句柄。由 *ql_mqtt_client_init()*获取，详情参考***第 3.3.1 章***。

- **返回值**

无

3.3.2.4. mqtt_connection_status_e

MQTT CONNECT 请求结果的定义如下：

```
typedef enum
{
   MQTT_CONNECT_ACCEPTED                    = 0,*/
   MQTT_CONNECT_REFUSED_PROTOCOL_VERSION    = 1, */
   MQTT_CONNECT_REFUSED_IDENTIFIER          = 2, */
   MQTT_CONNECT_REFUSED_SERVER              = 3,*/
   MQTT_CONNECT_REFUSED_USERNAME_PASS       = 4,*/
   MQTT_CONNECT_REFUSED_NOT_AUTHORIZED      = 5,*/
   MQTT_CONNECT_TCP_CONNECTED_FAILURE       = 254,*/
   MQTT_CONNECT_OUT_OF_MEMORY               = 255,*/
   MQTT_CONNECT_DISCONNECTED                = 256,*/
   MQTT_CONNECT_TIMEOUT                     = 257
} mqtt_connection_status_e
```

- **参数**

参数	描述
MQTT_CONNECT_ACCEPTED	MQTT 会话 CONNECT 成功
MQTT_CONNECT_REFUSED_PROTOCOL_VERSION	版本信息错误导致 CONNECT 失败

MQTT_CONNECT_REFUSED_IDENTIFIER	客户端 ID 检查不通过导致 CONNECT 失败
MQTT_CONNECT_REFUSED_SERVER	Server 拒绝导致 CONNECT 失败
MQTT_CONNECT_REFUSED_USERNAME_PASS	用户名/密码错误导致 CONNECT 失败
MQTT_CONNECT_REFUSED_NOT_AUTHORIZED	鉴权失败导致 CONNECT 失败
MQTT_CONNECT_TCP_CONNECTED_FAILURE	TCP 连接建立失败导致 CONNECT 失败
MQTT_CONNECT_OUT_OF_MEMORY	内存不足导致 CONNECT 失败
MQTT_CONNECT_DISCONNECTED	连接断开导致 CONNECT 失败
MQTT_CONNECT_TIMEOUT	超时错误导致 CONNECT 失败

3.3.2.5. mqtt_ssl_config_t

MQTT 中 SSL 配置定义如下：

```
struct mqtt_ssl_config_t{/
    int     ssl_ctx_id;/
    int     verify_level;
    char    *cacert_path;
    char    *client_cert_path;
    char    *client_key_path;
    char    *client_key_pwd;
}
```

- **参数**

类型	参数	描述
int	*ssl_ctx_id*	SSL 上下文 ID
int	*verify_level*	SSL 校验等级
char *	*cacert_path*	CA 证书的路径
char *	*client_cert_path*	客户端证书的路径
char *	*client_key_path*	客户端私钥文件的路径
char *	*client_key_pwd*	客户端私钥文件加密的口令

3.3.3. ql_mqtt_publish

该函数用于发布指定主题的消息。

- **函数原型**

```
int ql_mqtt_publish(mqtt_client_t *client, const char *topic, const void *payload,  unsigned short payload_length, unsigned char qos, unsigned char retain, mqtt_request_cb_t cb, void *arg)
```

- **参数**

client：
[In] MQTT 客户端句柄。由 *ql_mqtt_client_init()*获取，详情参考***第 3.3.1 章***。

topic：
[In] 发布消息的主题。

payload：
[In] 发布的消息。

payload_length：
[In] 发布的消息长度。

qos：
[In] 发布消息的 QoS 等级。

retain：
[In] 是否保留发布的消息。
1 消息发布后存储于服务器上
0 消息发布后立即删除

cb：
[In] PUBLISH 请求结果的回调函数。详情参考***第 3.3.3.1 章***。

arg：
[In] PUBLISH 请求结果回调函数的回调参数。

- **返回值**

详见***第 3.3.1.1 章***。

3.3.3.1. mqtt_request_cb_t

该回调函数由内核调用以通知应用层 MQTT 订阅、取消订阅和 PUBLISH 请求结果。

- **函数原型**

```
typedef void (*mqtt_request_cb_t)(mqtt_client_t *client, void *arg,int err)
```

- **参数**

client：
[In] MQTT 客户端句柄。由 *ql_mqtt_client_init()*获取，详情参考***第 3.3.1 章***。

arg：
[In] 回调参数。由 *ql_mqtt_sub_unsub()*和 *ql_mqtt_publish()*传入，详情参考***第 3.3.4 章***和***第 3.3.3 章***。

err：
[In] 请求结果。详情参考***第 3.3.2.4 章***。

- **返回值**

无

3.3.4. ql_mqtt_sub_unsub

该函数用于订阅/取消订阅指定主题。

- **函数原型**

```
int ql_mqtt_sub_unsub(mqtt_client_t *client, const char *topic, unsigned char qos, mqtt_request_cb_t cb, void *arg,unsigned char sub)
```

- **参数**

client：
[In] MQTT 客户端句柄。由 *ql_mqtt_client_init()*获取，详情参考***第 3.3.1 章***。

topic：
[In] 订阅/取消订阅的主题。

qos：
[In] 订阅/取消订阅的主题的 QoS 等级。

cb：
[In] 订阅/取消订阅结果的回调函数。详情参考***第 3.3.3.1 章***。

arg：
[In] 订阅/取消订阅结果回调函数的回调参数。

sub：
[In] 操作类型。
1 订阅
0 取消订阅

- **返回值**

详见***第 3.3.1.1 章***。

3.3.5. ql_mqtt_disconnect

该函数用于向服务器发送 DISCONNECT 请求，断开 MQTT 客户端与服务器之间的 MQTT 会话连接。

- **函数原型**

```
int ql_mqtt_disconnect(mqtt_client_t *client, mqtt_disconnect_cb_t cb, void *arg)
```

- **参数**

client：
[In] MQTT 客户端句柄。由 *ql_mqtt_client_init()*获取，详情参考***第 3.3.1 章***。

cb：
[In] DISCONNECT 请求结果回调函数。详情参考***第 3.3.5.1 章***。

arg：
[In] DISCONNECT 请求结果回调函数的回调参数。

- **返回值**

详见***第 3.3.1.1 章***。

3.3.5.1. mqtt_disconnect_cb_t

该回调函数由内核调用以通知应用层 MQTT DISCONNECT 请求的结果。

- **函数原型**

```
typedef void (*mqtt_disconnect_cb_t)(mqtt_client_t *client, void *arg,int err)
```

- **参数**

client：
[In] MQTT 客户端句柄。由 *ql_mqtt_client_init()*获取，详情参考***第 3.3.1 章***。

arg：
[In] 回调参数。由 *ql_mqtt_disconnect()*传入，详情参考***第 3.3.2 章***。

err：
[In] 请求结果。详情参考***第 3.3.2.4 章***。

- **返回值**

无

3.3.6. ql_mqtt_set_inpub_callback

该函数用于设置接收服务器发布消息的处理回调函数。

- **函数原型**

```
int ql_mqtt_set_inpub_callback(mqtt_client_t *client, mqtt_incoming_publish_cb_t inpub_cb, void *arg)
```

- **参数**

client：
[In] MQTT 客户端句柄。由 *ql_mqtt_client_init()*获取，详情参考***第 3.3.1 章***。

inpub_cb：
[In] 接收服务器发布消息的回调函数。详情参考***第 3.3.6.1 章***。

arg：
[In] 接收服务器发布消息回调函数的参数。

- **返回值**

详见***第 3.3.1.1 章***。

3.3.6.1. mqtt_incoming_publish_cb_t

该回调函数由内核调用以通知应用层接收到发布消息。

- **函数原型**

```
typedef void (*mqtt_incoming_publish_cb_t)(mqtt_client_t *client, void *arg, int pkt_id, const char *topic, const unsigned char *payload, unsigned short payload_len)
```

- **参数**

client：
[In] MQTT 客户端句柄。由 *ql_mqtt_client_init()*获取，详情参考***第 3.3.1 章***。

arg：
[In] 回调参数。由 *ql_mqtt_set_inpub_callback()*传入，详情参考***第 3.3.6 章***。

pkt_id：
[In] 发布消息的 ID。

topic：
[In] 发布消息关联的主题。

payload：
[In] 发布消息的消息体。

payload_len：
[In] 发布消息的消息体长度。

- **返回值**

无

3.3.7. ql_mqtt_client_is_connected

该函数用于查询客户端与服务器之间的会话连接是否已经建立。

- **函数原型**

```
int ql_mqtt_client_is_connected(mqtt_client_t *client)
```

- **参数**

client：
[In] MQTT 客户端句柄。由 *ql_mqtt_client_init()*获取，详情参考***第 3.3.1 章***。

- **返回值**

1　　已建立连接
0　　未建立连接

3.3.8. ql_mqtt_client_deinit

该函数用于释放 MQTT 客户端资源。

- **函数原型**

```
int ql_mqtt_client_deinit(mqtt_client_t *client)
```

- **参数**

client：
[In] MQTT 客户端句柄。由 *ql_mqtt_client_init()*获取，详情参考***第 3.3.1 章***。

- **返回值**

详见***第 3.3.1.1 章***。

3.3.9. ql_mqtt_onenet_generate_auth_token

该函数用于生成 OneNET 平台需要的密码。

- **函数原型**

```
char *ql_mqtt_onenet_generate_auth_token(signed long long expire_time,char *product_id,char *device_name,char *version,char *access_key)
```

- **参数**

expire_time：
[In] OneNET 平台的 token 过期时间。单位：秒。

product_id：
[In] OneNET 平台的产品 ID。

device_name：
[In] OneNET 平台的设备名称。

version：
[In] OneNET 平台的版本。

access_key：
[In] OneNET 平台设备密码。

- **返回值**

NULL	函数执行失败
其他	函数执行成功，执行完后需使用 *free()*释放空间

4 示例

请参考移远通信 ECx00U 系列和 EGx00U 模块 QuecOpen SDK 中提供的 MQTT 客户端示例代码文件 *mqtt_demo.c*，该示例文件位于 *components\ql-application\mqtt* 路径下。

5 附录 参考文档及术语缩写

表 3：参考文档

文档名称
[1] Quectel_ECx00U&EGx00U 系列_QuecOpen_CSDK_快速开发指导
[2] Quectel_ECx00U&EGx00U 系列_QuecOpen_数据拨号_API_参考手册

表 4：术语缩写

缩写	英文全称	中文全称
API	Application Programming Interface	应用程序编程接口
CA	Certificate Authority	证书颁发机构
DNS	Domain Name Server	域名系统（服务）协议
LTE	Long-Term Evolution	长期演进
ID	Identifier	标识符
IoT	Internet of Things	物联网
MQTT	Message Queuing Telemetry Transport	消息队列遥测传输协议
QoS	Quality of Service	服务质量
RTOS	Real-Time Operating System	实时操作系统
SDK	Software Development Kit	软件开发工具包
SSL	Secure Sockets Layer	安全套接层
TCP	Transmission Control Protocol	传输控制协议