ECx00U&EGx00U 系列

QuecOpen SSL API 参考手册

LTE Standard 模块系列

版本：1.0

日期：2021-12-17

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。

隐私声明

为实现移远通信产品功能，特定设备数据将会上传至移远通信或第三方服务器（包括运营商、芯片供应商或您指定的服务器）。移远通信严格遵守相关法律法规，仅为实现产品功能之目的或在适用法律允许的情况下保留、使用、披露或以其他方式处理相关数据。当您与第三方进行数据交互前，请自行了解其隐私保护和数据安全政策。

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
-	2021-03-02	Kruskal ZHU	文档创建
1.0	2021-12-17	Kruskal ZHU	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，ECx00U 系列和 EGx00U 模块在 App 侧的 SSL 相关 API、调用流程及示例。

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 SSL 相关接口调用流程

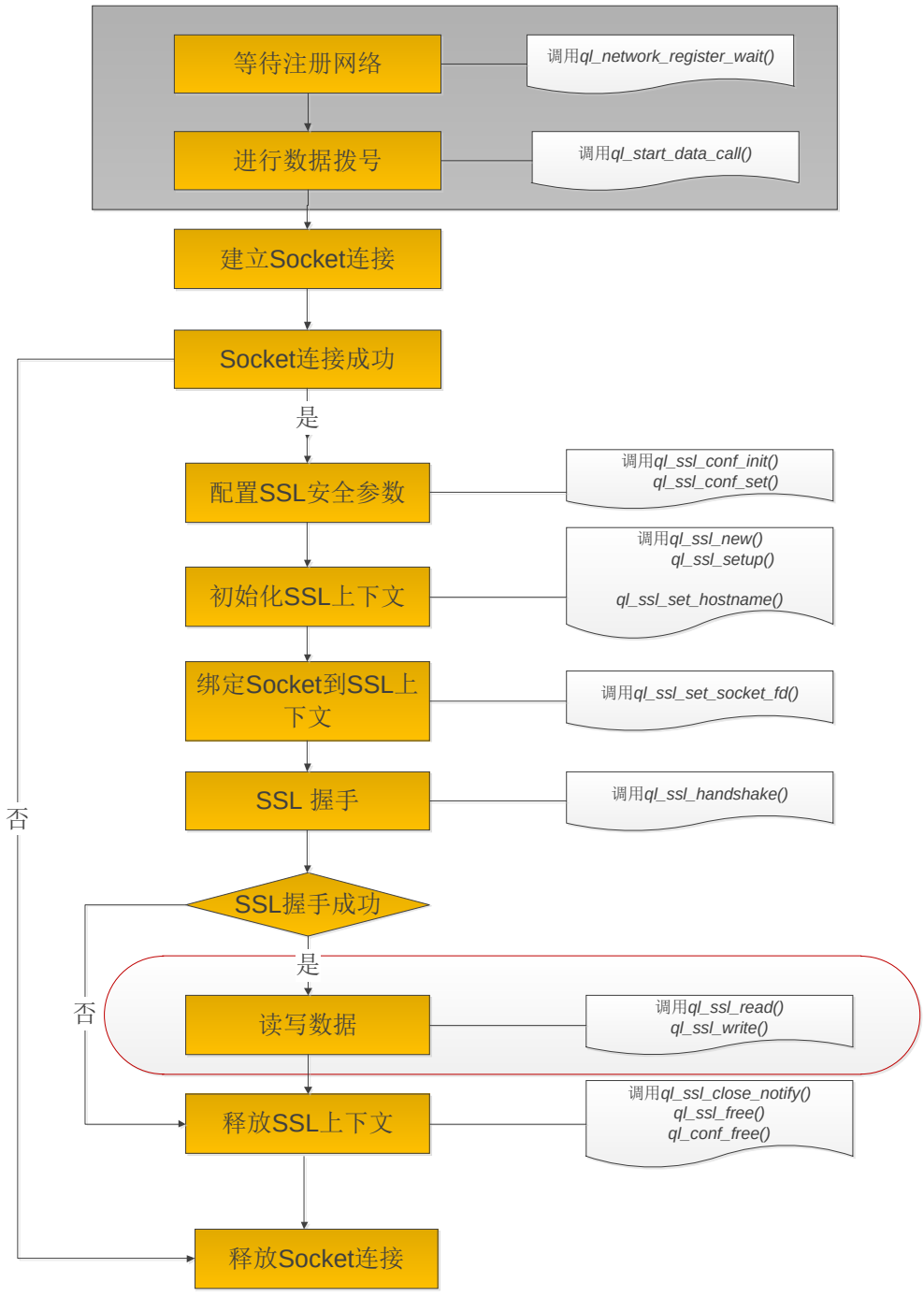

图 1：SSL 接口调用流程

在建立SSL会话连接之前，首先需注网并拨号以建立数据通道（如上图中灰底部分），详情参考***文档 [3]***。若已经在其他任务中建立了数据通道，可跳过注网拨号的步骤。

SSL 会话是基于 Socket 连接建立的，因此在 SSL 握手之前，需要先创建 Socket，绑定已经激活数据通道 IP，建立 Socket 连接。Socket 连接建立成功后，配置 SSL 安全参数并初始化 SSL 上下文，将对应的 Socket 描述符绑定到指定的 SSL 上下文。SSL 握手成功后，即可读写数据，数据读写完成后即可释放 SSL 上下文和 Socket 连接。

3 SSL API

3.1. 头文件

ql_ssl.h 文件即为 SSL API 的头文件，位于 *components\ql-kernel\inc* 目录下；若无特别说明，本文档所提及的头文件均位于该目录下。

3.2. 函数概览

表 2：函数概览

函数	说明
ql_ssl_conf_init()	初始化 SSL 默认配置参数
ql_ssl_conf_set()	设置指定的 SSL 配置参数
ql_ssl_conf_get()	获取指定的 SSL 配置参数
ql_ssl_conf_set_by_id()	设置 SSL 上下文 ID 对应的 SSL 上下文配置参数
ql_ssl_conf_get_by_id()	获取 SSL 上下文 ID 对应的 SSL 上下文配置参数
ql_ssl_conf_free()	释放 SSL 配置参数占用的资源
ql_ssl_new()	生成一个新的 SSL 上下文
ql_ssl_setup()	配置 SSL 上下文
ql_ssl_set_socket_fd()	绑定 Socket 描述符和 SSL 上下文
ql_ssl_set_hostname()	设置服务器的域名
ql_ssl_handshake()	执行 SSL 握手
ql_ssl_close_notify()	通知服务器 SSL 连接正在关闭

ql_ssl_get_bytes_avail()	获取 SSL 上下文中数据缓冲区中可读取的数据长度
ql_ssl_read()	读取 SSL 上下文中数据缓冲区可读取的数据
ql_ssl_write()	通过 SSL 上下文向服务器发送数据
ql_ssl_free()	释放 SSL 上下文占用的资源
ql_ssl_handshake_finished()	查询 SSL 握手是否完成
ql_ssl_ciphersuit_is_valid()	查询指定的加密套件 ID 是否有效

3.3. 函数详解

3.3.1. ql_ssl_conf_init

该函数用于初始化 SSL 默认配置参数。

- **函数原型**

```
int   ql_ssl_conf_init(ql_ssl_config *conf);
```

- **参数**

conf：
[In] SSL 配置的句柄，请参考***第 3.3.3.1 章***。

- **返回值**

详情请参考***第 3.3.1.1 章***。

3.3.1.1. ql_ssl_error_code_e

SSL 结果码表示函数是否执行成功，枚举定义如下：

```
typedef enum{
    QL_SSL_SUCCESS                  = 0,
    QL_SSL_ERROR_UNKOWN             = -1,
    QL_SSL_ERROR_WOUNDBLOCK         = -2,
    QL_SSL_ERROR_INVALID_PARAM      = -3,
    QL_SSL_ERROR_OUT_OF_MEM         = -4,
    QL_SSL_ERROR_NOT_SUPPORT        = -5,
    QL_SSL_ERROR_HS_FAILURE         = -6,
```

```
    QL_SSL_ERROR_DECRYPT_FAILURE   = -7,
    QL_SSL_ERROR_ENCRYPT_FAILURE   = -8,
    QL_SSL_ERROR_HS_INPROGRESS     = -9,
    QL_SSL_ERROR_BAD_REQUEST       = -10,
    QL_SSL_ERROR_WANT_READ         = -11,
    QL_SSL_ERROR_WANT_WRITE        = -12,
    QL_SSL_ERROR_SOCKET_RESET      = -13,
}ql_ssl_error_code_e;
```

- **成员**

成员	描述
QL_SSL_SUCCESS	API 执行成功
QL_SSL_ERROR_UNKOWN	未知错误
QL_SSL_ERROR_WOUNDBLOCK	操作未完成，结果等待异步通知
QL_SSL_ERROR_INVALID_PARAM	无效参数
QL_SSL_ERROR_OUT_OF_MEM	内存不够
QL_SSL_ERROR_NOT_SUPPORT	操作不支持
QL_SSL_ERROR_HS_FAILURE	SSL 握手失败
QL_SSL_ERROR_DECRYPT_FAILURE	解密失败
QL_SSL_ERROR_ENCRYPT_FAILURE	加密失败
QL_SSL_ERROR_HS_INPROGRESS	正在进行 SSL 握手
QL_SSL_ERROR_BAD_REQUEST	请求错误
QL_SSL_ERROR_WANT_READ	无可读数据
QL_SSL_ERROR_WANT_WRITE	无可写入数据
QL_SSL_ERROR_SOCKET_RESET	Socket 异常断开

3.3.2. ql_ssl_conf_set

该函数用于设置指定的 SSL 配置参数。

- **函数原型**

```
int    ql_ssl_conf_set(ql_ssl_config *conf, int type, ...);
```

- **参数**

conf：
[In] SSL 配置的句柄，请参考***第 3.3.3.1 章***。

type：
[In] SSL 参数配置类型，请参考***第 3.3.2.1 章***。

- **返回值**

详情请参考***第 3.3.1.1 章***。

3.3.2.1. ql_ssl_config_type_e

SSL 参数配置类型，枚举定义如下：

```
typedef enum{
    QL_SSL_CONF_VERSION           = 1,
    QL_SSL_CONF_TRANSPORT         = 2,
    QL_SSL_CONF_CIPHERSUITE       = 3,
    QL_SSL_CONF_AUTHMODE          = 4,
    QL_SSL_CONF_CACERT            = 5,
    QL_SSL_CONF_OWNCERT           = 6,
    QL_SSL_CONF_SNI               = 7,
    QL_SSL_CONF_HS_TIMEOUT        = 8,
    QL_SSL_CONF_IGNORE_LOCALTM = 9,
    QL_SSL_CONF_HS_TIMEOUT_FUNC=  10,
    QL_SSL_CONF_IGNORE_INVALID_CERT_SIGN  =  11,
    QL_SSL_CONF_IGNORE_CERT_ITEM          =  12,
    QL_SSL_CONF_IGNORE_MULTI_CERTCHAIN_VERIFY   =  13
    #ifdef  QL_SSL_TLS_SESSION_CACHE_FEATURE
    QL_SSL_CONF_SESSION_CACHE   =  14,
    #endif
}ql_ssl_config_type_e;
```

- **成员**

成员	描述
QL_SSL_CONF_VERSION	设置 SSL 版本信息，详情请参考***第 3.3.2.2 章***
QL_SSL_CONF_TRANSPORT	设置 SSL 通信方式，详情请参考***第 3.3.2.3 章***
QL_SSL_CONF_CIPHERSUITE	设置 SSL 加密套件

QL_SSL_CONF_AUTHMODE	设置 SSL 校验方式，详情请参考***第 3.3.2.4 章***
QL_SSL_CONF_CACERT	设置 SSL CA 证书列表
QL_SSL_CONF_OWNCERT	设置 SSL 本地证书
QL_SSL_CONF_SNI	设置 SSL 是否开启服务器名称指示功能
QL_SSL_CONF_HS_TIMEOUT	设置 SSL 握手超时时长
QL_SSL_CONF_IGNORE_LOCALTM	设置是否忽略本地时间
QL_SSL_CONF_HS_TIMEOUT_FUNC	设置 SSL 握手的超时处理函数
QL_SSL_CONF_IGNORE_INVALID_CERT_SIGN	设置是否忽略服务器发送的 SSL 证书的检查
QL_SSL_CONF_IGNORE_CERT_ITEM	设置是否忽略服务器发送的 SSL 证书项的检查
QL_SSL_CONF_IGNORE_MULTI_CERTCHAIN_VERIFY	设置是否忽略 SSL 多级证书链验证
QL_SSL_CONF_SESSION_CACHE	设置是否开启 SSL 会话复用功能

3.3.2.2. ql_ssl_version_type_e

SSL 版本信息，枚举定义如下：

```
typedef enum
{
    QL_SSL_VERSION_0 = 0,
    QL_SSL_VERSION_1,
    QL_SSL_VERSION_2,
    QL_SSL_VERSION_3,
    QL_SSL_VERSION_ALL
} ql_ssl_version_type_e;
```

- **成员**

成员	描述
QL_SSL_VERSION_0	SSL 3.0 版本
QL_SSL_VERSION_1	TLS 1.0 版本（SSL 3.1）
QL_SSL_VERSION_2	TLS 1.1 版本（SSL 3.2）
QL_SSL_VERSION_3	TLS 1.2 版本（SSL 3.3）

QL_SSL_VERSION_ALL	支持所有版本的 SSL

3.3.2.3. ql_ssl_transport_type_e

SSL 通信方式，枚举定义如下：

```
typedef enum{
    QL_SSL_TLS_PROTOCOL   = 0,
    QL_SSL_DTLS_PROTOCOL = 1,
}ql_ssl_transport_type_e;
```

- **成员**

成员	描述
QL_SSL_TLS_PROTOCOL	基于 TCP Socket 通信
QL_SSL_DTLS_PROTOCOL	基于 UDP Socket 通信

3.3.2.4. ql_ssl_authmode_e

SSL 校验方式，枚举定义如下：

```
typedef enum
{
    QL_SSL_VERIFY_NULL             =  0x0000,
    QL_SSL_VERIFY_SERVER           = 0x0001,
    QL_SSL_VERIFY_CLIENT_SERVER = 0x0002,
} ql_ssl_authmode_e;
```

- **成员**

成员	描述
QL_SSL_VERIFY_NULL	服务器不要求校验客户端时，此选项表示客户端不校验服务器； 服务器要求校验客户端时，设置为此选项会导致 SSL 握手失败
QL_SSL_VERIFY_SERVER	服务器不要求校验客户端时，此选项表示客户端校验服务器； 服务器要求校验客户端时，设置为此选项会导致 SSL 握手失败

QL_SSL_VERIFY_CLIENT_SERVER	服务器不要求校验客户端时，此选项等同于 *QL_SSL_VERIFY_SERVER*； 服务器要求校验客户端时，必须设置为此选项

3.3.3. ql_ssl_conf_get

该函数用于获取指定的 SSL 配置参数。

- **函数原型**

```
int   ql_ssl_conf_get(ql_ssl_config *conf, int type, ...);
```

- **参数**

conf：
[In] SSL 属性配置，请参考***第 3.3.3.1 章***。

type：
[In] SSL 参数配置类型，请参考***第 3.3.2.1 章***。

- **返回值**

详情请参考***第 3.3.1.1 章***。

3.3.3.1. ql_ssl_config

SSL 属性配置，结构体定义如下：

```
typedef struct{
    int                             ssl_version;
    int                             transport; //0: TLS   1: DTLS
    int                             *ciphersuites;
    int                             auth_mode;
    int                             sni_enable;
    char                            *ca_cert_path[QL_MAX_CA_CERT_CNT];
    char                            *own_cert_path;
    char                            *own_key_path;
    char                            *own_key_pwd;
    int                             ssl_negotiate_timeout;
    ql_ssl_handshake_timeout_cb     negotiate_timeout_cb;
    void                            *negotiate_timeout_cb_arg;
    int                             ignore_invalid_certsign;
    uint32_t                        ignore_certitem;
```

```
    int                             ignore_multi_certchain_verify;
    #ifdef QL_SSL_TLS_SESSION_CACHE_FEATURE
    ql_ssl_session                  ssl_session_cache;
    #endif
}ql_ssl_config;
```

- **参数**

类型	参数	描述
int	*ssl_version*	SSL 版本信息
int	*transport*	SSL 协议类型 *0*：TLS *1*：DTLS
int	*ciphersuites*	加密套件
int	*auth_mode*	校验方式
int	*sni_enable*	是否配置 SNI
char	*ca_cert_path*	CA 证书路径列表，最多可为 *QL_MAX_CA_CERT_CNT*
char	*own_cert_path*	本地证书路径
char	*own_key_path*	本地私钥文件路径
char	*own_key_pwd*	本地私钥文件的加密口令；若无，则为 NULL
int	*ssl_negotiate_timeout*	SSL 协商的最大超时时间
ql_ssl_handshake_timeout_cb	*negotiate_timeout_cb*	SSL 握手超时回调函数，详情请参考***第 3.3.3.2 章***
void	*negotiate_timeout_cb_arg*	传入 SSL 握手超时回调函数的参数
int	*ignore_invalid_certsign*	设置是否忽略服务器发送的 SSL 证书的检查
uint32_t	*ignore_certitem*	设置是否忽略服务器发送的 SSL 证书项的检查
int	*ignore_multi_certchain_verify*	设置是否忽略多级证书链的验证
ql_ssl_session	*ssl_session_cache*	SSL 会话复用配置，详情请参考***第 3.3.3.3 章***

3.3.3.2. ql_ssl_handshake_timeout_cb

该回调函数用于定义 SSL 握手超时回调函数的函数指针。

- **函数原型**

```
typedef void(*ql_ssl_handshake_timeout_cb)(ql_ssl_context *ssl_ctx, void *arg);
```

- **参数**

ssl_ctx：
[In] SSL 上下文的句柄指针，请参考***第 3.3.7.1 章***。

arg：
[In] 用户自定义回调函数参数的指针。

3.3.3.3. ql_ssl_session

SSL 会话复用配置，结构体定义如下：

```
typedef struct {
uint8_t                 session_cache_enable;
ip_addr_t               remote_ip;
uint16_t                remote_port;
uint8_t                 hostname_temp[256];
uint8_t                 session_hostname[256];
mbedtls_ssl_session     ssl_session;
} ql_ssl_session;
```

- **参数**

类型	参数	描述
uint8_t	*session_cache_enable*	是否开启会话复用功能 0：关闭 1：开启
ip_addr_t	*remote_ip*	用于保存对端服务器 IP 地址
uint16_t	*remote_port*	用于保存对端服务器端口号
uint8_t	*hostname_temp*	临时存储对端服务器主机名
uint8_t	*session_hostname*	用于会话复用的真实主机名

mbedtls_ssl_session	*ssl_session*	用于存储当前会话数据

3.3.4. ql_ssl_conf_set_by_id

该函数用于设置 SSL 上下文 ID 对应的 SSL 上下文配置参数。

- **函数原型**

```
int ql_ssl_conf_set_by_id(int ctx_id, int type, ...);
```

- **参数**

ctx_id：
[In] 整型。SSL 上下文 ID。范围：0~5。

type：
[In] SSL 的参数配置类型，请参考***第 3.3.2.1 章***。

- **返回值**

详情请参考***第 3.3.1.1 章***。

3.3.5. ql_ssl_conf_get_by_id

该函数用于获取 SSL 上下文 ID 对应的 SSL 上下文配置参数。

- **函数原型**

```
int ql_ssl_conf_get_by_id(int ctx_id, int type, ...);
```

- **参数**

ctx_id：
[In] 整型。SSL 上下文 ID。范围：0~5。

type：
[In] SSL 的参数配置类型，请参考***第 3.3.2.1 章***。

- **返回值**

详情请参考***第 3.3.1.1 章***。

3.3.6. ql_ssl_conf_free

该函数用于释放 SSL 配置参数占用的资源。

- **函数原型**

```
int    ql_ssl_conf_free(ql_ssl_config *conf);
```

- **参数**

conf：
[In] SSL 配置的句柄，请参考***第 3.3.3.1 章***。

- **返回值**

详情请参考***第 3.3.1.1 章***。

3.3.7. ql_ssl_new

该函数用于生成一个新的 SSL 上下文。

- **函数原型**

```
int    ql_ssl_new(ql_ssl_context *ssl);
```

- **参数**

ssl：
[Out] SSL 上下文的句柄指针，请参考***第 3.3.7.1 章***。

- **返回值**

详情请参考***第 3.3.1.1 章***。

3.3.7.1. ql_ssl_context

SSL 上下文定义如下：

```
typedef int    ql_ssl_context;
```

- **参数**

类型	参数	描述
int	*ql_ssl_context*	SSL 上下文的句柄

3.3.8. ql_ssl_setup

该函数用于配置 SSL 上下文。

- **函数原型**

```
int    ql_ssl_setup(ql_ssl_context *ssl, ql_ssl_config *conf);
```

- **参数**

ssl：
[In] SSL 上下文的句柄指针，请参考***第 3.3.7.1 章***。

conf：
[In] SSL 属性配置，请参考***第 3.3.3.1 章***。

- **返回值**

详情请参考***第 3.3.1.1 章***。

3.3.9. ql_ssl_set_socket_fd

该函数用于绑定 Socket 描述符和 SSL 上下文。

- **函数原型**

```
int    ql_ssl_set_socket_fd(ql_ssl_context *ssl, int sock_fd);
```

- **参数**

ssl：
[In] SSL 上下文的句柄指针，请参考***第 3.3.7.1 章***。

sock_fd：
[In] Socket 描述符。

- **返回值**

详情请参考***第 3.3.1.1 章***。

3.3.10. ql_ssl_set_hostname

该函数用于设置服务器的域名，仅在配置 SNI 的情况下有效。

- **函数原型**

```
int    ql_ssl_set_hostname(ql_ssl_context *ssl, const char *hostname);
```

- **参数**

ssl：
[In] SSL 上下文的句柄指针，请参考***第 3.3.7.1 章***。

hostname：
[In] 服务器的域名。

- **返回值**

详情请参考***第 3.3.1.1 章***。

3.3.11. ql_ssl_handshake

该函数用于执行 SSL 握手。

- **函数原型**

```
int    ql_ssl_handshake(ql_ssl_context *ssl);
```

- **参数**

ssl：
[In] SSL 上下文的句柄指针，请参考***第 3.3.7.1 章***。

- **返回值**

详情请参考***第 3.3.1.1 章***。

3.3.12. ql_ssl_close_notify

该函数用于通知服务器 SSL 连接正在关闭。

- **函数原型**

```
int    ql_ssl_close_notify(ql_ssl_context *ssl);
```

- **参数**

ssl：
[In] SSL 上下文的句柄指针，请参考***第 3.3.7.1 章***。

- **返回值**

详情请参考***第 3.3.1.1 章***。

3.3.13. ql_ssl_get_bytes_avail

该函数用于在 SSL 握手完成后，获取 SSL 上下文中数据缓冲区中可读取的数据长度。

- **函数原型**

```
int    ql_ssl_get_bytes_avail(ql_ssl_context *ssl);
```

- **参数**

ssl：
[In] SSL 上下文的句柄指针，请参考***第 3.3.7.1 章***。

- **返回值**

小于 0　　返回 SSL 结果码，请参考***第 3.3.1.1 章***
大于等于 0　返回缓冲区中可读取的数据长度

3.3.14. ql_ssl_read

该函数用于在 SSL 握手完成后，读取 SSL 上下文中数据缓冲区可读取的数据。

- **函数原型**

```
int    ql_ssl_read(ql_ssl_context *ssl, unsigned char *buf, size_t len);
```

- **参数**

ssl：
[In] SSL 上下文的句柄指针，请参考***第 3.3.7.1 章***。

buf：
[In] 存储读取数据的缓冲区。

len：
[In] 待读取数据长度。

- **返回值**

小于 0　　返回 SSL 结果码，请参考***第 3.3.1.1 章***
大于等于 0　返回当前已从缓冲区成功读取的数据字节数

3.3.15. ql_ssl_write

该函数用于在 SSL 握手完成后，通过 SSL 上下文向服务器发送数据。

- **函数原型**

```
int    ql_ssl_write(ql_ssl_context *ssl, const unsigned char *buf, size_t len);
```

- **参数**

ssl：
[In] SSL 上下文的句柄指针，请参考***第 3.3.7.1 章***。

buf：
[In] 存储数据的缓冲区。

len：
[In] 待发送的数据长度。

- **返回值**

小于 0　　返回 SSL 结果码，请参考***第 3.3.1.1 章***
大于等于 0　返回当前已从缓冲区成功写入的数据字节数

3.3.16. ql_ssl_free

该函数用于释放 SSL 上下文占用的资源。

- **函数原型**

```
int    ql_ssl_free(ql_ssl_context *ssl);
```

- **参数**

ssl：
[In] SSL 上下文的句柄指针，请参考***第 3.3.7.1 章***。

- **返回值**

详情请参考***第 3.3.1.1 章***。

3.3.17. ql_ssl_handshake_finished

该函数用于查询 SSL 握手是否完成。

- **函数原型**

```
int   ql_ssl_handshake_finished(ql_ssl_context *ssl);
```

- **参数**

ssl：
[In] SSL 上下文的句柄指针，请参考***第 3.3.7.1 章***。

- **返回值**

1　　SSL 握手已完成
0　　SSL 握手未完成
小于 0　返回 SSL 结果码，请参考***第 3.3.1.1 章***

3.3.18. ql_ssl_ciphersuit_is_valid

该函数用于查询指定的加密套件 ID 是否有效。

- **函数原型**

```
int   ql_ssl_ciphersuit_is_valid(int cs_id);
```

- **参数**

cs_id：
[In] 加密套件的 ID，详情请参考 SDK 中的 *ql_ssl.h*。

- **返回值**

1　有效
0　无效

4 示例

SSL API 接口的示例文件为 *ssl_demo.c*，位于 QuecOpen SDK 包的*\components\ql-application\ssl* 目录下。*ssl_demo.c* 文件中主要包括建立 TCP 连接，配置 SSL 会话属性，建立 SSL 连接，通过 SSL 连接进行数据读写等操作。用户可自行查看接口函数的完整示例。

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_Log_抓取指导
[3] 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
AP	Application Processor	应用处理器
App	Application	应用
DTLS	Datagram Transport Layer Security	数据包传输层安全协议
EVB	Evaluation Board	评估板
IoT	Internet of Things	物联网
PC	Personal Computer	个人计算机
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
SNI	Server Name Indication	服务器名称指示
TLS	Transport Layer Security	传输层安全（协议）

USB	Universal Serial Bus	通用串行总线
UDP	User Datagram Protocol	用户数据报协议