A76xx Series Open SDK_ MQTT_应用指导

LTE 模组

芯讯通无线科技(上海)有限公司
上海市长宁区临虹路289号3号楼芯讯通总部大楼
电话：86-21-31575100
技术支持邮箱：support@simcom.com
官网：www.simcom.com

名称：	A76xx Series Open SDK_MQTT_应用指导
版本：	V1.00
类别：	应用文档
状态：	已发布

版权声明

芯讯通无线科技(上海)有限公司
上海市长宁区临虹路289号3号楼芯讯通总部大楼
电话：86-21-31575100
邮箱：simcom@simcom.com
官网：www.simcom.com

了解更多资料，请点击以下链接：
http://cn.simcom.com/download/list-230-cn.html

技术支持，请点击以下链接：
http://cn.simcom.com/ask/index-cn.html 或发送邮件至 support@simcom.com

Version History

Version	Date	Owner	What is new
V1.00	2022-10-26		第一版

About this Document

本文档适用于 A1803S open 系列、A1603 open 系列、A1606 open 系列。

目录

1MQTT 基础介绍

MQTT 是一个客户端服务端架构的发布/订阅模式的消息传输协议。它的设计思想是轻巧、开放、简单、规范，易于实现。这些特点使得它对很多场景来说都是很好的选择，特别是对于受限的环境如机器与机器的通信(M2M)以及物联网环境(IoT)。MQTT 二次开发默认版本为 3.3.1。

1.1 MQTT 简介

MQTT(Message Queuing Telemetry Transport，消息队列遥测传输协议)，是一种基于发布/订阅(publish/subscribe)模式的“轻量级”通讯协议，该协议构建于 TCP/IP 协议上，由 IBM 在 1999 年发布。MQTT 最大优点在于，可以以极少的代码和有限的带宽，为连接远程设备提供实时可靠的消息服务。做为一种低开销、低带宽占用的即时通讯协议，使其在物联网、小型设备、移动应用等方面有较广泛的应用。

MQTT 协议设计规范：

1．精简，不添加可有可无的功能。

2．发布/订阅(Pub/Sub)模式，方便消息在传感器之间传递，解耦 Client/Server 模式，不必预先知道对方的存在(ip/port)，不必同时运行。

3. 允许用户动态创建主题(不需要预先创建主题)，零运维成本。

4. 把传输量降到最低以提高传输效率。

5. 把低带宽、高延迟、不稳定的网络等因素考虑在内。

6．支持连续的会话保持和控制(心跳)。

7. 理解客户端计算能力可能很低。

8. 提供服务质量(quality of level : QoS)管理。

9.不强求传输数据的类型和格式，保持灵活性(指的是应用层业务数据)。

1.2 发布/订阅模式

MQTT 是基于发布(Publish)/订阅(Subscribe)模式来进行通信及数据交换的，与 HTTP 的请求(Request)/应答(Response)的模式有本质的不同，发布者和订阅者之间并不需要直接建立联系。发布者与订阅者不必了解彼此，只要认识同一个消息代理即可。

1.3 主题

可以理解为消息的类型，订阅者订阅（Subscribe）后，就会收到该主题的消息内容 。MQTT 通过主题对消息进行分类，本质上是一个 UTF-8 的字符串。可以通过斜杠表示多个层级关系。主题不需要创建便可直接使用。

1.4 服务质量

消息服务质量（QoS）支持，可靠传输保证；有三种消息发布服务质量：

QoS0："至多一次"，消息发布完全依赖底层 TCP/IP 网络。会发生消息丢失或重复。这一级别可用于如下情况，环境传感器数据，丢失一次读记录无所谓，因为不久后还会有第二次发送。这一种方式主要用于普通 APP 的推送，倘若你的智能设备在消息推送时未联网，推送过去没收到，再次联网也就收不到了。

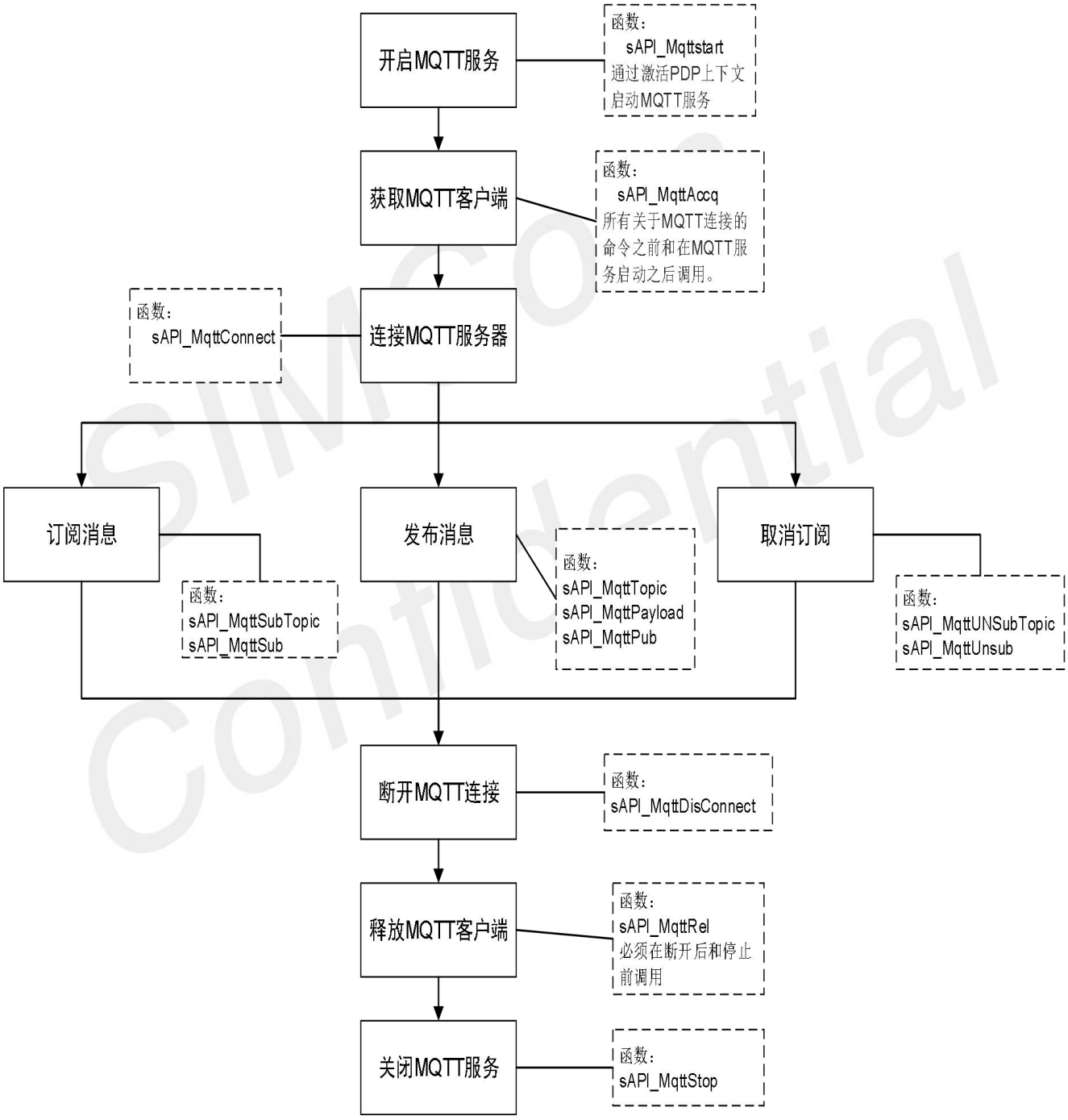
QoS1："至少一次"，确保消息到达，但消息重复可能会发生。

QoS2："只有一次"，确保消息到达一次。在一些要求比较严格的计费系统中，可以使用此级别。在计费系统中，消息重复或丢失会导致不正确的结果。这种最高质量的消息发布服务还可以用于即时通讯类的 APP 的推送，确保用户收到且只会收到一次。

1.5 消息类型

1. CONNECT：客户端连接到服务器
2. CONNACK：确认连接请求
3. PUBLISH：发布消息
4. PUBACK：发布确认，是 QoS 1 给 PUBLISH 消息的回复
5. PUBREC：QoS 2 消息流的第一部分，表示发布的消息已接收
6. PUBREL：QoS 2 消息流的第二部分，表示发布的消息已释放
7. PUBCOMP：QoS 2 消息流的第三部分，表示消息发布完成
8. SUBSCRIBE：订阅主题
9. SUBACK：订阅确认
10. UNSUBSCRIBE：取消订阅
11. UNSUBACK：取消订阅确认
12. PINGREQ：心跳请求
13. PINGRESP：确认心跳
14. DISCONNECT：断开连接

2 使用流程



3API 介绍

头文件：#include "simcom_mqtts_client.h"

MQTT API 的一部分包含一些用于不同状态的参数，如果对于特定的条件是不必要的，这些参数应该设置为 NULL 或 0。有一些不重要的参数可以设置为默认值，这在 API 参数描述中定义。

所有 API 函数都是阻塞函数，只有在函数完全执行后才返回，一些 API，如 sAPI_MqttConnect，可能需要几秒钟。

3.1 sAPI_MqttStart 启动 MQTT 服务

sAPI_MqttStart 用于通过激活 PDP 上下文启动 MQTT 服务。您必须在任何其他 MQTT 相关操作之前使用此 API。

接口:	SCmqttReturnCode **sAPI_MqttStart**(int channel);
参数:	[in] **channel**: urc 传输通道(到串口或其他通道); 0: 串口 1: usb 口 -1: 忽略
返回值:	SCmqttReturnCode，成功或错误代码

3.2 sAPI_MqttStop 停止 MQTT 服务

sAPI_MqttStop 用于停止 MQTT 服务。

接口:	SCmqttReturnCode **sAPI_MqttStop**(void);
参数:	无

返回值:	SCmqttReturnCode，成功或错误代码

NOTE

这个 API 用于停止 MQTT 服务。您可以在断开和释放后使用它。

3.3 sAPI_MqttAccq 获取 MQTT 客户端

sAPI_MqttAccq 用于获取 MQTT 客户端。它必须在所有关于 MQTT 连接的命令之前和在 MQTT 服务启动之后调用。

接口:	SCmqttReturnCode **sAPI_MqttAccq**(SCmqttOperationType commad_type, char *string_return, int client_index, char *clientID, int server_type, sMsgQRef msgQ_urc);
参数:	[in] **commad_type**: SC_MQTT_OP_SET: 通过该命令设置参数 SC_MQTT_OP_GET: 获取通过此命令设置的参数 [out] **string_return**: 在 commad_type==SC_MQTT_OP_GET 时，返回的字符串; [in] **client_index**: 标识客户端的数字参数。允许的值范围是 0 到 1。 [in] **clientID**: UTF 编码字符串. 它为客户端指定一个唯一标识符。字符串长度为 1 ~ 128 字节。 [in] **server_type**: 标识服务器类型的数字参数。缺省值为 0。 0: TCP 的 MQTT 服务器 1: SSL/TLS 的 MQTT 服务器 [in] **msgQ_urc**: MQTT 服务的 urc 消息的消息队列。
返回值:	SCmqttReturnCode，成功或错误代码

NOTE

这用于获取 MQTT 客户端。它必须在所有关于 MQTT 连接的命令之前和启动 MQTT 服务之后调用
对于 MQTT 消息队列(msgQ_urc)，只接收来自 MQTT 服务器的订阅消息数据。

3.4 sAPI_MqttRel 释放 MQTT 客户端

sAPI_MqttRel 用于释放 MQTT 客户端。它必须在断开后和停止前调用。

接口:	SCmqttReturnCode **sAPI_MqttRel**(int client_index);
参数:	[in] **client_index:** 标识客户端的数字参数。允许的值范围是 0 到 1。
返回值:	SCmqttReturnCode，成功或错误代码

NOTE

用于释放 MQTT 客户端。它必须在断开后和停止前调用。

3.5 sAPI_MqttSslCfg 设置 SSL 上下文

sAPI_MqttSslCfg 用于设置当 SSL 连接到 SSL/TLS MQTT 服务器时要在 SSL 连接中使用的 SSL 上下文。它必须在连接之前和启动之后调用。连接失败或断开连接后，设置将被清除。

接口:	SCmqttReturnCode **sAPI_MqttSslCfg**(SCmqttOperationType commad_type, char *string_return, int client_index, int ssl_ctx_index);
参数:	[in] **commad_type**: SC_MQTT_OP_SET: 通过该命令设置参数 SC_MQTT_OP_GET: 获取通过此命令设置的参数 [out] **string_return**: 在 commad_type == SC_MQTT_OP_GET 时，返回的字符串; [in] **client_index**: 标识客户端的数字参数。允许的值范围是 0 到 1。

	[in] **ssl_ctx_index**: SSL 连接中将使用的 SSL 上下文 ID。参考 SSL 相关 API。
返回值:	SCmqttReturnCode，成功或错误代码

3.6 sAPI_MqttWillTopic 输入遗嘱消息的主题

sAPI_MqttWillTopic 用于输入遗嘱消息的主题。

接口:	SCmqttReturnCode **sAPI_MqttWillTopic**(int client_index, char *topic_data, int topic_length);
参数:	[in] **client_index**: 标识客户端的数字参数。允许的值范围是 0 到 1。 [in] **topic_data**: 发布消息的主题字符串，它应该是 UTF 编码的字符串。取值范围是 1 ~ 1024 字节。 [in] **topic_length**: 输入主题数据的长度。
返回值:	SCmqttReturnCode，成功或错误代码

3.7 sAPI_MqttWillMsg 输入遗嘱消息的消息体

sAPI_MqttWillMsg 用于输入遗嘱消息的消息体。

接口:	SCmqttReturnCode **sAPI_MqttWillMsg**(int client_index, char *msg_data, int msg_length);
参数:	[in] **client_index**: 标识客户端的数字参数。允许的值范围是 0 到 1。 [in] **msg_data**: 遗嘱消息字符串，它应该是 UTF 编码的字符串。取值范围是 1 ~ 1024 字节。 [in] **msg_length**: 遗嘱消息数据的长度。

返回值:	SCmqttReturnCode，成功或错误代码

3.8 sAPI_MqttConnect 连接 MQTT 服务器

sAPI_MqttConnect 用于连接到 MQTT 服务器。

接口:	SCmqttReturnCode **sAPI_MqttConnect**(SCmqttOperationType commad_type, char *string_return, int client_index, char *server_addr, int keepalive_time, int clean_session, char *user_name, char *pass_word);
参数:	[in] **commad_type**: SC_MQTT_OP_SET: 通过该命令设置参数 SC_MQTT_OP_GET: 获取通过此命令设置的参数 [out] **string_return**: 在 commad_type== SC_MQTT_OP_GET 时，返回的字符串; [in] **client_index**: 标识客户端的数字参数。允许的值范围是 0 到 1。 [in] **server_addr**: 描述服务器地址和端口的字符串。字符串长度范围是 9 ~ 256 字节。字符串应该是这样的"tcp://116.247.119.165:5141"，必须以"tcp://"开头。如果<server_addr>不包含端口，则默认端口为 1883。 [in] **keepalive_time**: 从客户端接收到的两个消息之间的时间间隔。当很长一段时间没有消息发送给服务器时，客户端会发送一个 keep-alive 报文。取值范围为 1s ~ 64800s(18 小时); [in] **clean_session**: 清除会话标志。取值范围为 0 ~ 1，默认值为 0。 [in] **user_name**: 用户名标识连接到服务器时可用于身份验证的用户名。字符串长度为 1~256 字节。 [in] **pass_word**: 与用户对应的密码，连接到服务器时可以使用该密码进行认

	证。字符串长度为 1 ~ 256 字节。
返回值:	SCmqttReturnCode，成功或错误代码

NOTE

如果您没有在连接 SSL/TLS MQTT 服务器之前通过 sAPI_MqttSslCfg 设置 SSL 上下文，那么它将在连接到服务器时使用<client_index> SSL 上下文。

3.9 sAPI_MqttDisConnect 从服务器断开连接

sAPI_MqttDisConnect 用于从服务器断开连接。

接口:	SCmqttReturnCode **sAPI_MqttDisConnect**(SCmqttOperationType commad_type, char *string_return, int client_index, int timeout);
参数:	[in] **commad_type**: SC_MQTT_OP_SET: 通过该命令设置参数 SC_MQTT_OP_GET: 获取通过此命令设置的参数 [out] **string_return**: 在 commad_type==SC_MQTT_OP_GET 时，返回的字符串; [in] **client_index**: 标识客户端的数字参数。允许的值范围是 0 到 1。 [in] **timeout**: 断开连接的超时值。单位为秒。范围是 60 到 180S。默认值为 0(未设置超时值);
返回值:	SCmqttReturnCode，成功或错误代码

3.10 sAPI_MqttTopic 输入发布消息的主题

sAPI_MqttTopic 用于输入发布消息的主题。

接口:	SCmqttReturnCode **sAPI_MqttTopic**(int client_index, char *topic_data, int

	topic_length);
参数:	[in] **client_index**: 标识客户端的数字参数。允许的值范围是 0 到 1。 [in] **topic_data**: 发布消息的主题字符串，它应该是 UTF 编码的字符串。 [in] **topic_length**: 输入主题数据的长度。取值范围是 1 ~ 1024 字节。
返回值:	SCmqttReturnCode，成功或错误代码

NOTE

主题发布到服务器后将被清除。

3.11 sAPI_MqttPayload 输入发布消息的消息体

sAPI_MqttPayload 用于输入发布消息的消息体。

接口:	SCmqttReturnCode **sAPI_MqttPayload**(int client_index, char *payload_data, int payload_length);
参数:	[in] **client_index**: 标识客户端的数字参数。允许的值范围是 0 到 1。 [in] **payload_data**: 有效载荷数据的缓冲区。发布消息应该是 UTF 编码的字符串。 [in] **payload_length**: 输入消息数据的长度。取值范围是 1 ~ 10240 字节。
返回值:	SCmqttReturnCode，成功或错误代码

NOTE

主题发布到服务器后将被清除。

3.12 sAPI_MqttPub 发布消息到 MQTT 服务器

sAPI_MqttPub 用于向 MQTT 服务器发布消息。

接口:	SCmqttReturnCode **sAPI_MqttPub**(int client_index, int qos, int pub_timeout, int ratained, int dup);
参数:	[in] **client_index**: 标识客户端的数字参数。允许的值范围是 0 到 1。 [in] **qos**: 发布消息的 qos。取值范围为 0 ~ 2。 [in] **pub_timeout**: 断开连接的超时值。单位为秒。范围是 60 到 180S。默认值为 0(未设置超时值); [in] **ratained**: 发布消息的 retain 标志。取值为 0 ~ 1。缺省值为 0。 [in] **dup**: 消息的 dup 标志。取值为 0 ~ 1。缺省值为 0。当客户端或服务器试图重新传递消息时设置该标志。
返回值:	SCmqttReturnCode，成功或错误代码

NOTE

主题和有效载荷在发布之后会被清除。

3.13 sAPI_MqttSubTopic 输入订阅消息的主题

sAPI_MqttSubTopic 用于输入订阅消息的主题。

接口:	SCmqttReturnCode **sAPI_MqttSubTopic**(int client_index, char *sub_topic_data, int sub_topic_length, int qos);
参数:	[in] **client_index**: 标识客户端的数字参数。允许的值范围是 0 到 1。 [in] **sub_topic_data**: 订阅消息的主题。订阅消息主题应该是 UTF 编码的字符串。 [in] **sub_topic_length**: 输入主题数据的长度。取值范围是 1 ~ 10240 字节。 [in] **qos**: 发布消息的 qos。取值范围为 0 ~ 2。

返回值:	SCmqttReturnCode，成功或错误代码

NOTE

订阅成功后，主题将被清除。

3.14sAPI_MqttSub 向 MQTT 服务器订阅消息

sAPI_MqttSub 用于向 MQTT 服务器订阅消息，也可以用于订阅 sAPI_MqttSubTopic 输入的多个消息主题，详见 demo。

接口:	SCmqttReturnCode **sAPI_MqttSub**(int client_index, char *topic_data, int topic_length, int qos, int dup);
参数:	[in] **client_index**: 标识客户端的数字参数。允许的值范围是 0 到 1。 [in] **topic_data**: 发布消息的主题字符串。发布消息主题应该是 UTF 编码的字符串。 [in] **topic_length**: 输入主题数据的长度。取值范围是 1 ~ 1024 字节。 [in] **dup**: 消息的 dup 标志。取值为 0 ~ 1。缺省值为 0。当客户端或服务器试图重新传递消息时设置该标志。 [in] **qos**: 发布消息的 qos。取值范围为 0 ~ 2。
返回值:	SCmqttReturnCode，成功或错误代码

3.15sAPI_MqttUNSubTopic 输入退订消息的主题

sAPI_MqttUNSubTopic 用于输入退订消息的主题。

接口:	SCmqttReturnCode **sAPI_MqttUNSubTopic**(int client_index, char *unsub_topic_data, int unsub_topic_length);

参数:	[in] **client_index**: 标识客户端的数字参数。允许的值范围是 0 到 1。 [in] **unsub_topic_data**: 退订消息的主题。发布消息主题应该是 UTF 编码的字符串。 [in] **unsub_topic_length**: 输入主题数据的长度。取值范围是 1 ~ 1024 字节。
返回值:	SCmqttReturnCode，成功或错误代码

NOTE

退订成功后，主题将被清除。

3.16sAPI_MqttUnsub 向 MQTT 服务器取消订阅消息

sAPI_MqttUnsub 用于取消向 MQTT 服务器订阅消息，也可以用于取消订阅 sAPI_MqttUNSubTopic 输入的多个消息主题，详见 demo。

接口:	SCmqttReturnCode **sAPI_MqttUnsub**(int client_index, char *topic_data, int topic_length, int dup);
参数:	[in] **client_index**: 标识客户端的数字参数。允许的值范围是 0 到 1。 [in] **topic_data**: 取消订阅消息的主题字符串。发布消息主题应该是 UTF 编码的字符串。 [in] **topic_length**: 输入主题数据的长度。取值范围是 1 ~ 1024 字节。 [in] **dup**: 消息的 dup 标志。取值为 0 ~ 1。缺省值为 0。当客户端或服务器试图重新传递消息时设置该标志。
返回值:	SCmqttReturnCode，成功或错误代码

3.17 sAPI_MqttCfg 配置 MQTT 上下文

sAPI_MqttCfg 用于配置 MQTT 上下文。它必须在连接服务器之前和获取客户端之后调用。客户端释放后，设置将被清除。

接口	SCmqttReturnCode **sAPI_MqttCfg**(SCmqttOperationType commad_type, char *string_return, int client_index, int config_type, int related_value);
参数:	[in] **commad_type**: SC_MQTT_OP_SET: 通过该命令设置参数 SC_MQTT_OP_GET: 获取通过此命令设置的参数 [out] **string_return**: 在 commad_type==SC_MQTT_OP_GET 时，返回的字符串。 [in] **client_index**: 标识客户端的数字参数。允许的值范围是 0 到 1。 [in] **config_type**: 选择一个配置类型 0: "checkUTF8" 1: "optimeout" [in] **related_value**: (1); 如果 config_type = 0, related_value 设置检查字符串是否是 UTF8 编码的标志，默认值为 1。 0: 不检查 UTF8 编码。 1: 检查 UTF8 编码。 (2); 如果 config_type = 1, related_value 设置为超时值，用于设置发送或接收数据操作的最大超时时间间隔。取值范围是 20 ~ 120 秒，缺省值是 120 秒。
返回值:	SCmqttReturnCode，成功或错误代码

NOTE

在 sAPI_MqttRel 之后，该设置将被清除。

3.18 sAPI_MqttConnLostCb 处理 MQTT 客户端异常断开连接

sAPI_MqttConnLostCb 用于处理 MQTT 客户端的异常断开。建议在每次 sAPI_MqttConnect( )连接之前使用 sAPI_MqttConnLostCb，因为每次调用 sAPI_MqttRel( )成功会把注册的回调函数清除掉。不能在回调函数里直接重连，会占用底层的 task，导致后续接收异常，建议使用 sAPI_MsgQSend( )发送一个 MsgQ 来通知其他 task 处理重连。

接口:	void sAPI_MqttConnLostCb(mqtt_connlost_cb cb);
参数:	[in] **cb**: 当网络异常中断时调用的函数
返回值:	无

NOTE

在 sAPI_MqttRel 之后，该设置将被清除。

4 变量定义

4.1 SCmqttResultType

```
typedef enum
{
    SC_MQTT_RESULT_SUCCESS = 0,                    //成功
    SC_MQTT_RESULT_FAIL,                           //失败
    SC_MQTT_RESULT_BAD_UTF8_STR,                   //错误的 UTF 编码
    SC_MQTT_RESULT_SOCK_CONN_FAIL,                 //socket 连接失败
    SC_MQTT_RESULT_SOCK_CREATE_FAIL,               //socket 创建失败
    SC_MQTT_RESULT_SOCK_CLOSE_FAIL,                //5 socket 关闭失败
    SC_MQTT_RESULT_RCV_FAIL,                       //接收失败
    SC_MQTT_RESULT_NETWORK_OPEN_FAIL,              //网络打开失败
    SC_MQTT_RESULT_NETWORK_CLOSE_FAIL,             //网络关闭失败
    SC_MQTT_RESULT_NETWORK_NO_OPEN,                //网络没有打开
    SC_MQTT_RESULT_CLINET_INDEX_ERR,               //10  客户端索引错误
    SC_MQTT_RESULT_NO_CONNECTION,                  //没有连接
    SC_MQTT_RESULT_INVALID_PARAMETER,              //无效参数
    SC_MQTT_RESULT_OPERATION_NOT_SUPPORT, //操作不支持
    SC_MQTT_RESULT_BUSY,                           //忙碌
    SC_MQTT_RESULT_REQ_CONNECTION_FAIL, //15 请求连接失败
    SC_MQTT_RESULT_SOCK_SENDING_FAIL,              //socket 发送失败
    SC_MQTT_RESULT_TIMOUT,                         //超时
    SC_MQTT_RESULT_TOPIC_EMPTY,                    //主题空
    SC_MQTT_RESULT_CLIENT_IN_USE,                  //客户端正在使用
    SC_MQTT_RESULT_CLIENT_NOT_ACCQ,                //20  客户端没有获取
    SC_MQTT_RESULT_CLIENT_NOT_REL,                 //客户端没有释放
    SC_MQTT_RESULT_EXCEED_MAX_VAL,                 //超过最大值
    SC_MQTT_RESULT_NETWORK_HAVE_OPENED, //网络已经打开
    SC_MQTT_RESULT_PACKET_FAIL,                    //打包失败
    SC_MQTT_RESULT_DNS_ERROR,                      //DNS 失败
    SC_MQTT_RESULT_SOCK_CLOSE,                     // 26  服务器关闭 socket
    SC_MQTT_RESULT_UNACCEPTED_PROTOCOL_VER, //27  连接拒绝:不接受的协议版本
    SC_MQTT_RESULT_ID_REJECTED,                   //28  连接拒绝:标识符被拒绝
    SC_MQTT_RESULT_SER_UNVAILBLE,                 //29  连接拒绝:服务器不可用
    SC_MQTT_RESULT_BAD_USRNAME_PWD,               //30  连接拒绝:错误的用户名或密码
    SC_MQTT_RESULT_NOT_AUTHORIZED,                // 31 连接拒绝：未授权
```

```
    SC_MQTT_RESULT_SSL_HANDSHAKE_ERR,      //32 SSL 握手失败
    SC_MQTT_RESULT_NOT_SET_CERTS,          //33  没有设置 ssl 证书
    SC_MQTT_RESULT_OPEN_SESSION_ERR,       //34 打开  ssl 会话失败
    SC_MQTT_RESULT_DISCONN_FAIL,           //断开连接失败
    SC_MQTT_RESULT_MAX
}SCmqttResultType;
```

4.2 ScmqttOperationType

```
typedef enum {
    SC_MQTT_OP_SET = 0,
    SC_MQTT_OP_GET,
}SCmqttOperationType;
```

5 结果代码

<err>	Meaning
0	Operation succeeded
1	Failed
2	Bad utf-8 string
3	Sock connect fail
4	Sock create fail
5	Sock close fail
6	Message receive fail
7	Network open fail
8	Network close fail
9	Network not opened
10	Client index error
11	No connection
12	Invalid parameter
13	Not supported operation
14	Client is busy
15	Require connection fail
16	Sock sending fail
17	Timeout
18	Topic is empty
19	Client is used
20	Client not acquired
21	Client not released
22	Length out of range
23	Network is opened
24	Packet fail
25	Dns error
26	Socket is closed by server
27	Connection refused: unaccepted protocol version
28	Connection refused: identifier rejected
29	Connection refused: server unavailable
30	Connection refused: bad user name or password
31	Connection refused: not authorized
32	Handshake fail
33	Not set certificate

34	Open session failed
35	Disconnect from server failed

6 参考 example

```
#include "simcom_mqtts_client.h"
int ret = -1;
int error_code = 0;
mqtt_client_thread();     //客户端线程控制 MQTT 进程
{
ret = sAPI_MqttStart(-1);          //启动 MQTT 服务，没有 urc 传输通道
if(0 != ret);{
sAPI_Debug("start fail, error_code = %d", ret);
}
sAPI_MqttAccq(0, NULL, 0, "test0", 0, urc_mqtt_msgq_1); //获取一个客户端 client-0

sAPI_MqttCfg(0, NULL, 0, 0, 0);     //配置 client-0，不检查 UTF-8 编码

    sAPI_MqttConnLostCb(app_MqttLostConnCb);
ret = sAPI_MqttConnect(0, NULL, 0, "tcp://120.27.2.154:1883", 60, 1, NULL, NULL);
//连接到服务器
if(0 != ret);{
sAPI_Debug("connect fail, error_code = %d", ret);
}
sAPI_MqttSubTopic(0, "apitest", 7, 1);   //设置多主题消息(不超过 10 个);
sAPI_MqttSubTopic(0, "apitest_2", 9, 1);
sAPI_MqttSubTopic(0, "apitest_3", 9, 1);
. . .

ret = sAPI_MqttSub(0, NULL, 0, 0, 0);   //订阅多个输入主题
if(0 != ret);{
sAPI_Debug("sub fail, error_code = %d", ret);
}
ret = sAPI_MqttSub(0, "apitest1", 8, 0, 0);   //只订阅一个主题
if(0 != ret);{
sAPI_Debug("sub one topic fail, error_code = %d", ret);
}

sAPI_MqttTopic(0, "apitest", 7);   //输入发布主题消息

sAPI_MqttPayload(0, "come here, it is the test msg", strlen("come here, it is the test msg"));
//输入发布消息的有效负载(不超过 10240 字节);

ret = sAPI_MqttPub(0, 1, 60, 0, 0);   //将消息发布发到服务器
```

```
if(0 != ret);{
sAPI_Debug("pub message fail, error_code = %d", ret);
}

Sleep(30);
//在取消订阅之前休眠 30 秒，在取消订阅之前，接收任务可以收到关于订阅主题的消息

ret = sAPI_MqttUnsub(0, "apitest", 8, 0);   //取消订阅主题
if(0 != ret);{
sAPI_Debug("unsub a topic fail, error_code = %d", ret);
}

sAPI_MqttUNSubTopic(0, "apitest2", 8);   //设置多个退订主题(不超过 10 个);
. . .

ret = sAPI_MqttUnsub(0, NULL, 0, 0);   //取消对服务器的多主题订阅
if(0 != ret);{
sAPI_Debug("unsub multi topic fail, error_code = %d", ret);
}

Sleep(20);
//在断开连接前休眠 20 秒，接收任务无法从未订阅的主题接收消息
Ret = sAPI_MqttDisConnect(0, NULL, 0, 60);   //和服务器断开连接
if(0 != ret);{
sAPI_Debug("disconnect fail, error_code = %d", ret);
}
sAPI_MqttRel(0);   //释放 client-0

sAPI_MqttStop();   //停止 MQTT 服务
}

mqtt_receive_thread();        //线程准备接收关于订阅主题的消息
{
   mqtt_data *sub_data = NULL;
   FS_MSG_T msgQ_data_recv = {FS_MSG_INIT, 0, -1, NULL};
   simMsgRecv(urc_mqtt_msgq_1, &msgQ_data_recv, OS_SUSPEND);
   sub_data =(mqtt_data *);(msgQ_data_recv.arg3);   //这里获取订阅主题消息数据

}
```