芯祥联科技 MQTT Client SDK 商用使用手册

适用版本：MQTT_CLIENT_SDK V3.0.0（最终定稿版，推荐正式上线）

适用协议：MQTT 3.1.1 / MQTT 5.0 全特性

核心特性：纯主任务轮询、无线程依赖、商用级健壮性、零内存泄漏、全平台适配，极简+高级可配，平衡好用+够用

文档面向：嵌入式/物联网/服务器端 开发工程师

技术支持：hezuo@xxltech.cn

一、SDK 概述

1.1 产品定位

芯祥联 MQTT Client SDK 是一款**轻量级、高性能、跨平台**的 MQTT 客户端开发套件，专为商用物联网产品设计，完美支持 MQTT 5.0 全特性与 MQTT 3.1.1 兼容模式，无第三方依赖，可无缝移植至 MCU、Linux、Windows、RTOS 等平台。核心设计遵循“极简可配、高级全量”原则，普通用户零门槛上手，高级用户可按需开启所有功能，覆盖100%物联网业务场景。

1.2 核心优势

- **无线程设计**：纯主任务轮询，无多线程竞争风险，适配裸机/RTOS 环境
- **全特性支持**：MQTT 5.0 全报文类型+全属性字段，满足高阶商用需求，共享订阅、消息过期等特性原生支持
- **高健壮性**：自动重连、会话恢复、消息重传、内存安全管理，断网自愈，异常容错
- **极简接口**：标准化 API，仅7个核心接口，全生命周期覆盖，调用流程固定，用户易记易用
- **商用级保障**：零内存泄漏、所有配置有合理默认值，错误码/枚举统一管理，无冗余无兼容问题
- **灵活可配**：普通用户无需修改任何配置，一行创建客户端；高级用户按需修改配置，功能拉满

1.3 功能清单

分类	支持功能

基础功能	连接/断开、发布/订阅、取消订阅、心跳保活、遗嘱消息、连接状态查询
MQTT 5.0 特性	会话过期、消息过期、共享订阅（自动拼接前缀）、主题别名、用户属性、原因码、最大报文限制
增强特性	Payload 加密、用户名密码认证、断线自动重连（可配置间隔和次数）、QoS0/1/2 全支持、协议轮询处理
平台适配	Windows/Linux/STM32/ESP32/FreeRTOS/UCOS 等，支持Windows DLL导出

二、环境搭建

2.1 文件结构

代码块

```
├── include/                    // 头文件目录
│   ├── mqtt_client_sdk.h       // SDK 对外标准接口（唯一需要包含的头文件）
│   ├── platform_adapter.h      // 平台适配接口
│   ├── agent_log.h             // 日志接口
│   └── type.h                  // 基础数据类型定义
├── src/                        // 源码目录
│   ├── mqtt_client_sdk.c       // SDK 核心实现
│   ├── mqtt_client.c           // MQTT 协议栈实现
│   └── platform_adapter.c      // 平台适配实现（延时、内存、网络）
├── demo/                       // 演示用例目录（可选）
│   └── cli_demo.c              // CLI 演示用例（启用 CLIENT50_CMD_DEMO=1 生效）
└── doc/                        // 文档目录（可选）
    └── mqtt_client_sdk_manual.md  // SDK 使用手册
```

2.2 编译配置

1. **添加文件**：工程中添加 src 目录下的 3 个核心 .c 文件（mqtt_client_sdk.c、mqtt_client.c、platform_adapter.c）

2. **头文件路径**：添加 SDK 中 include 目录路径，确保工程能正常引用头文件
3. **宏定义**（可选）：

   - `CLIENT50_CMD_DEMO=1`：启用 CLI 演示用例
   - `SELF_TEST==1`：自测模式，禁用Windows DLL导出宏
   - `BUILD_MQTT_DLL`：Windows 环境下，编译生成DLL动态库
   - 平台相关宏：`_WIN32`/`__linux__`/`STM32` 等，用于适配不同平台

2.3 平台适配要求

SDK 依赖 3 个基础平台接口，需根据硬件平台实现：

1. **延时函数**：`ADAPTER_DelayMs(uint32_t ms)`
2. **内存管理**：`malloc/free`（或自定义内存分配）
3. **网络接口**：TCP 客户端连接/发送/接收（SDK 已封装，仅需适配底层）

三、核心数据类型

3.1 基础类型定义

代码块

```
// 布尔类型
typedef unsigned char  bool_xxl;
#define TRUE_XXL   1
#define FALSE_XXL  0

// MQTT协议版本枚举（统一归一，SDK.h唯一管理）
typedef enum {
    MQTT_SDK_PROTOCOL_V311 = 0,    // MQTT 3.1.1 协议版本（默认）
    MQTT_SDK_PROTOCOL_V50  = 1     // MQTT 5.0 协议版本（支持共享订阅、消息过期等高级特性）
} MqttSdkProtocolVersion;

// QoS 等级（MQTT标准协议字段，必保留）
typedef enum {
    MQTT_SDK_QOS0 = 0,  // 最多一次，最快，默认值，适合：数据上报、日志推送
    MQTT_SDK_QOS1 = 1,  // 至少一次，可靠，适合：控制指令、配置下发、告警推送
    MQTT_SDK_QOS2 = 2   // 恰好一次，最高可靠，适合：金融支付、重要指令(极少用)
} MqttSdkQos;

```

```
// SDK统一错误码（语义清晰，全覆盖场景，0成功/负数失败，用户易判断）
typedef enum {
    MQTT_SDK_OK          = 0,    // 操作成功 ✅
    MQTT_SDK_ERR_PARAM   = -1,   // 参数错误(空指针/非法配置/端口超限) ❌
    MQTT_SDK_ERR_MEM     = -2,   // 内存分配失败(创建客户端/缓存不足) ❌
    MQTT_SDK_ERR_NO_CONN = -3,   // 未连接服务器(未建联就发布/订阅) ❌
    MQTT_SDK_ERR_CONN    = -4,   // 连接失败(网络错误/地址错误/认证失败/版本不兼容) ❌
    MQTT_SDK_ERR_PUB     = -5,   // 发布失败(消息过长/主题非法/QoS错误) ❌
    MQTT_SDK_ERR_SUB     = -6,   // 订阅失败(主题非法/权限不足/订阅超限) ❌
    MQTT_SDK_ERR_PKT_ID  = -7,   // Packet ID非法(超出1-128范围) ❌
    MQTT_SDK_ERR_BUSY    = -8,   // 客户端忙，无法执行当前操作 ❌
    MQTT_SDK_ERR_CREATE  = -9,   // 客户端创建失败 ❌
    MQTT_SDK_ERR_TIMEOUT = -10,  // 操作超时 ❌
    MQTT_SDK_ERR_OTHER   = -99   // 其他错误(兜底，兼容所有未枚举异常) ❌
} MqttSdkErrCode;

// 客户端连接状态枚举（对外公开，业务层直接调用，与内核完全一致）
typedef enum
{
    MQTT_SDK_STATE_DISCONNECTED = 0,  // 未连接 / 已断开连接
    MQTT_SDK_STATE_CONNECTING    = 1,  // 正在连接（已发送CONNECT报文，等待Broker的CONNACK应答）
    MQTT_SDK_STATE_CONNECTED     = 2   // 连接成功（已收到CONNACK，可正常发布/订阅/收发消息）
} MqttSdkConnState;

// Broker协议交互反馈类型枚举（精准命名，剥离Conn歧义）
typedef enum {
    MQTT_SDK_BROKER_RESP_CONNACK = 0,     // Broker反馈：连接应答（CONNACK）
    MQTT_SDK_BROKER_RESP_DISCONNECT = 1, // Broker反馈：断开连接（主动/异常断开）
    MQTT_SDK_BROKER_RESP_PUBACK = 2,      // Broker反馈：QoS1发布应答（PUBACK）
    MQTT_SDK_BROKER_RESP_SUBACK = 3,      // Broker反馈：订阅应答（SUBACK）
    MQTT_SDK_BROKER_RESP_UNSUBACK = 4,    // Broker反馈：取消订阅应答（UNSUBACK）
    MQTT_SDK_BROKER_RESP_QOS2_ACK = 5     // Broker反馈：QoS2消息确认（PUBREC/PUBCOMP）
} MqttSdkBrokerRespType;

// 客户端句柄（不透明指针，屏蔽内部实现，用户只用传参，不用关心内部结构）
typedef void* MqttSdkClient;
```

3.2 核心配置结构体（MqttSdkClientCfg）

商用核心配置结构体，所有字段均有合理默认值，支持普通/进阶/高级三种使用方式，无嵌套结构体，配置一目了然。

代码块

```
typedef struct {
    // -------------------------- 【必填项 3个】无默认值，必须赋值 ----------------------------
    const char*     client_id;      // 客户端唯一标识 (必填，如："device_001_123456")
    const char*     broker_ip;      // 服务器IP/域名 (必填，如："192.168.1.100"、"mqtt.xxx.com")
    uint16_xxl      broker_port;    // 服务器端口 (必填，默认常用值：1883(MQTT)、8883(MQTTS))

    // -------------------------- 【基础可选配置】MQTT标准字段，常用 ------------------------------
    bool_xxl        clean_start;    // 是否清理会话 (可选，默认：true) MQTT标准字段
    uint32_xxl      keep_alive;     // 心跳间隔(秒) (可选，默认：30) MQTT标准字段，0=关闭心跳
    const char*     username;       // 用户名 (可选，默认：NULL) 业务刚需-身份认证
    const char*     password;       // 密码 (可选，默认：NULL) 业务刚需-身份认证

    // -------------------------- 【高级可选配置】业务刚需+MQTT高级特性，按需开启 ------------------------
    bool_xxl        crypto_enable;  // 是否开启消息加密 (可选，默认：false) 业务刚需-数据安全
    const char*     crypto_key;     // 加密密钥(16字节) (可选，默认：NULL) crypto_enable=true时生效
    const char*     crypto_iv;      // 加密向量(16字节) (可选，默认：NULL) crypto_enable=true时生效
    bool_xxl        will_enable;    // 是否开启遗嘱消息 (可选，默认：false) MQTT标准-设备离线告警
    MqttSdkMsg      will_msg;       // 遗嘱消息内容 (可选) will_enable=true时生效
    uint32_xxl      reconnect_intv; // 断线重连间隔(秒) (可选，默认：3) 业务刚需-断网自愈
    uint32_xxl      max_reconnect;  // 最大重连次数 (可选，默认：0) 0=无限重连，>0=指定次数

    // -------------------------- 【协议版本配置】核心开关，有默认值 ----------------------------
    int             mqtt_version;   // MQTT协议版本 (可选，默认：MQTT_SDK_PROTOCOL_V311)
                                    // 赋值为 MQTT_SDK_PROTOCOL_V50 即启用5.0，否则默认3.1.1

    // ====================== MQTT5.0专属全局配置【全部有默认值，3.1.1用户无需配置】======================
    // 所有字段默认值均为：不启用/无限制/最优默认行为，3.1.1版本下SDK自动忽略这些字段
    uint32_xxl      v50_msg_expiry;     // 消息默认过期时间(秒)，默认：0 → 消息永不过期
```

```
    uint32_xxl      v50_session_expiry;// 会话过期时间(秒)，默认：0 → 断开连接后会话立即失效
    uint32_xxl      v50_max_pkt_size;  // 最大报文大小(字节)，默认：0 → 不限制报文大小
    const char*     v50_share_group;   // 默认共享订阅组名（仅MQTT 5.0生效）
                                       // 客户端侧：默认NULL → 不启用共享订阅；非NULL时，订阅主题自动拼接$share/组名/前缀，仅用于标识订阅所属共享组
                                       // Broker侧：该字段用于共享组管理、客户端负载均衡、消息分发等核心业务逻辑，是共享订阅的核心配置项
} MqttSdkClientCfg;
```

3.3 消息结构体（MqttSdkMsg）

收发通用，极简够用，无冗余字段，支持消息过期配置，适配共享订阅场景。

代码块

```
typedef struct {
    const char*     topic;          // 消息主题（必填，如："device/data"）
    const uint8_xxl*  payload;        // 消息内容（必填，二进制/字符串均可，无格式限制）
    uint32_xxl      payload_len;    // 内容长度（必填，字节数，必须准确）
    MqttSdkQos      qos;            // 服务质量（可选，默认填 MQTT_SDK_QOS0 即可）
    bool_xxl        retain;         // 保留消息（可选，默认false，MQTT标准字段）
    uint32_xxl      msg_expiry;     // 单条消息过期时间(秒)，0=使用客户端全局配置，>0=自定义过期时间
                                    // 共享订阅场景下：过期时间生效，避免离线客户端重连后接收过期消息
} MqttSdkMsg;
```

3.4 Broker反馈结构体（MqttSdkBrokerRespStatus）

极致极简版，语义精准，用户一眼看懂，用于接收Broker的各类协议交互反馈。

代码块

```
typedef struct {
    bool_xxl                client_connected;  // 客户端连接状态（仅CONNACK/DISCONNECT场景有效）
    MqttSdkBrokerRespType   resp_type;         // Broker反馈类型（连接/订阅/发布等场景）
    MqttSdkErrCode          err_code;          // 交互错误码（0=成功，负数=失败，客户仅需判断这个）
    uint16_xxl              pkt_id;            // 报文ID（SUBACK/UNSUBACK/PUBACK/QoS2场景关联用）
```

```
6   } MqttSdkBrokerRespStatus;
```

3.5 回调函数类型

参数极简，含义清晰，用户易实现，覆盖消息接收和Broker反馈两大核心场景。

代码块

```
1   // 订阅消息接收回调（用户核心实现，必填）
2   // @param msg 收到的MQTT消息，直接使用即可
3   typedef void (*MqttSdkMsgCb)(const MqttSdkMsg* msg);
4
5   // Broker协议交互反馈回调（用户可选实现，推荐）
6   // @param resp_status Broker协议交互反馈详情（包含反馈类型、错误码、报文ID等）
7   typedef void (*MqttSdkBrokerRespCb)(const MqttSdkBrokerRespStatus*
    resp_status);
```

四、标准 API 接口（商用必用，7个核心接口，全生命周期覆盖）

调用顺序：全局初始化 → 创建客户端 → 连接服务器 → 发布/订阅/取消订阅 → 断开连接 → 销毁客户端 → 全局反初始化

4.1 全局初始化/反初始化

1. xxl_mqtt_sdk_global_init

功能：SDK 全局初始化，整个程序仅调用1次。

代码块

```
1   MQTT_API MqttSdkErrCode xxl_mqtt_sdk_global_init(void);
```

返回值：`MQTT_SDK_OK` 成功，其他错误码为失败。

2. xxl_mqtt_sdk_global_deinit

功能：SDK 全局反初始化，程序退出前调用1次，释放全局资源。

代码块

```
MQTT_API void xxl_mqtt_sdk_global_deinit(void);
```

说明：无返回值，调用后SDK全局资源全部释放，不可再调用其他SDK接口。

4.2 客户端生命周期管理

1. xxl_mqtt_sdk_client_create

功能：创建 MQTT 客户端实例，分配内存并初始化配置（第一步）。

代码块

```
MQTT_API MqttSdkClient xxl_mqtt_sdk_client_create(
    const MqttSdkClientCfg* cfg,
    MqttSdkMsgCb msg_cb,
    MqttSdkBrokerRespCb broker_resp_cb
);
```

参数：

- `cfg`：客户端配置结构体指针，普通用户仅需赋值3个必填项，其余默认即可。
- `msg_cb`：消息接收回调，必填，用于接收Broker下发的订阅消息。
- `broker_resp_cb`：Broker协议交互反馈回调，可选，传NULL则不接收反馈通知。

返回值：非空 = 客户端句柄（创建成功），NULL = 创建失败（可通过错误码排查原因）。

2. xxl_mqtt_sdk_client_destroy

功能：销毁 MQTT 客户端实例，释放所有分配的资源（最后一步）。

代码块

```
MQTT_API void xxl_mqtt_sdk_client_destroy(MqttSdkClient client);
```

参数：`client` - 客户端句柄（创建客户端时返回的非空值）。

说明：无返回值；Broker侧会彻底清理该客户端在共享组中的所有注册信息，释放共享组相关资源。

3. xxl_mqtt_sdk_client_connect

功能：发起与 MQTT Broker 的连接请求（第二步）。

代码块

```
1   MQTT_API MqttSdkErrCode xxl_mqtt_sdk_client_connect(MqttSdkClient client);
```

参数：`client` - 客户端句柄。

返回值：`MQTT_SDK_OK` 连接请求发送成功，其他错误码为失败（具体失败原因可通过Broker反馈回调查看）。

4. xxl_mqtt_sdk_client_disconnect

功能：主动断开与 MQTT Broker 的连接（优雅断开，发送 DISCONNECT 报文）。

代码块

```
1   MQTT_API MqttSdkErrCode xxl_mqtt_sdk_client_disconnect(MqttSdkClient client);
```

参数：`client` - 客户端句柄。

返回值：`MQTT_SDK_OK` 断开成功，其他错误码为失败。

说明：Broker侧断开连接时，会自动将该客户端从所属共享组中移除，避免消息分发异常。

4.3 核心业务接口（发布/订阅/取消订阅）

1. xxl_mqtt_sdk_client_publish

功能：发布 MQTT 消息（核心功能）。

代码块

```
1   MQTT_API MqttSdkErrCode xxl_mqtt_sdk_client_publish(MqttSdkClient client,
    const MqttSdkMsg* msg);
```

参数：

- `client` - 客户端句柄。
- `msg` - 要发布的消息结构体，需正确赋值主题、消息内容、内容长度，其他字段可选。

返回值：`MQTT_SDK_OK` 发布成功，其他错误码为失败（如消息过长、主题非法、未连接等）。

2. xxl_mqtt_sdk_client_subscribe

功能：订阅 MQTT 主题（核心功能），支持普通订阅和MQTT 5.0共享订阅。

代码块

```
1   MQTT_API MqttSdkErrCode xxl_mqtt_sdk_client_subscribe(
```

```
    MqttSdkClient client,
    const char* topic,
    MqttSdkQos qos,
    const char* session_share_group /* = NULL */
);
```

参数：

- `client` - 客户端句柄。
- `topic` - 要订阅的主题：普通订阅直接传入原始主题（如"test"）；MQTT 5.0共享订阅传入原始主题即可，SDK自动拼接$share/共享组名/前缀。
- `qos` - 订阅的消息质量等级，需与发布消息的QoS匹配。
- `session_share_group` - 【新增】会话级共享组（可选，默认NULL）：
  - NULL：使用实例级默认共享组（cfg->v50_share_group）；若实例级也为NULL，则非共享订阅。
  - 空字符串 ""：强制禁用共享订阅（无视实例级配置，优先级最高）。
  - 非空字符串：使用该会话级共享组（优先级高于实例级）。

返回值：`MQTT_SDK_OK` 订阅成功，其他错误码为失败（如主题非法、权限不足、未连接等）。

说明：Broker侧订阅时会根据主题解析出共享组名，完成共享组客户端注册。

3. xxl_mqtt_sdk_client_unsubscribe

功能：取消订阅 MQTT 主题。

代码块

```
MQTT_API MqttSdkErrCode xxl_mqtt_sdk_client_unsubscribe(
    MqttSdkClient client,
    const char* topic,
    const char* session_share_group /* = NULL */
);
```

参数：

- `client` - 客户端句柄。
- `topic` - 要取消的主题：普通订阅直接传入原始主题（如"test"）；MQTT 5.0共享订阅必须传入完整的共享订阅主题（格式：$share/共享组名/原始主题，如"$share/testg/test"）。
- `session_share_group` - 【新增】会话级共享组（可选，默认NULL）：

- NULL：匹配实例级默认共享组的订阅记录；若实例级为NULL，则匹配非共享订阅记录。
- 空字符串 ""：强制匹配非共享订阅记录（无视实例级配置）。
- 非空字符串：匹配该会话级共享组的订阅记录。

返回值：`MQTT_SDK_OK` 取消订阅成功，其他错误码为失败。

说明：MQTT 5.0共享订阅取消规则：此处参数用于简化匹配逻辑，无需手动拼接$share前缀；Broker侧取消时会根据该主题解析出共享组名，完成共享组客户端注销。

4.4 辅助接口

1. xxl_mqtt_sdk_client_get_conn_state

功能：查询客户端当前连接状态（对外唯一核心连接状态查询接口）。

代码块

```
MQTT_API MqttSdkConnState xxl_mqtt_sdk_client_get_conn_state(MqttSdkClient client);
```

参数：`client` - 客户端句柄。

返回值：`MqttSdkConnState` 枚举值，对应未连接、正在连接、已连接三种状态。

说明：只读接口、非阻塞、无副作用、线程安全，业务层可直接调用判断连接状态。

2. xxl_mqtt_sdk_client_loop

功能：SDK 核心轮询处理函数（SDK的核心心脏），处理下行报文、回调、重传、心跳等逻辑。

代码块

```
MQTT_API MqttSdkErrCode xxl_mqtt_sdk_client_loop(MqttSdkClient client);
```

参数：`client` - 客户端句柄。

返回值：`MQTT_SDK_OK` 处理成功，其他错误码为处理失败。

关键说明：

- 该函数为非阻塞函数，单次调用单次处理，执行完立即返回。
- 必须在「用户的主循环/专属线程」中**周期性调用**，建议调用频率：10~50ms/次。
- 该函数内部完成：下行报文接收解析、订阅消息回调、QoS消息重传、心跳保活、断线状态检测。
- 调用频率越高，MQTT响应越及时，最低不低于200ms/次。

五、商用开发注意事项

5.1 接口调用规范

1. 严格遵循接口调用顺序，不可跳过步骤（如未创建客户端直接调用连接接口）。
2. 客户端句柄为非空值时才可调用相关接口，销毁后不可再使用该句柄。
3. `xxl_mqtt_sdk_client_loop` 必须周期性调用，否则会导致消息接收延迟、心跳异常、重连失败。
4. 发布/订阅/取消订阅接口，需在客户端连接成功（状态为MQTT_SDK_STATE_CONNECTED）后调用。

5.2 配置相关注意事项

1. 客户端配置结构体中，3个必填项（client_id、broker_ip、broker_port）必须赋值，否则创建客户端失败。
2. 加密功能启用时（crypto_enable=true），必须正确赋值crypto_key（16字节）和crypto_iv（16字节），否则加密失败。
3. MQTT 5.0共享订阅配置时，v50_share_group非NULL即可启用，SDK自动拼接$share前缀，无需用户手动处理。
4. 重连配置中，max_reconnect设为0表示无限重连，适合需要长期在线的商用场景；设为具体数值时，重连失败后不再尝试重连。

5.3 错误处理建议

1. 所有接口调用后，建议判断返回值，根据错误码排查问题（错误码含义参考3.1节）。
2. 启用Broker反馈回调（broker_resp_cb），可实时获取连接、发布、订阅等操作的反馈，便于问题定位。
3. 连接失败时，优先排查Broker地址、端口、用户名密码是否正确，网络是否通畅。

5.4 共享订阅专项说明

1. 仅MQTT 5.0版本支持共享订阅，3.1.1版本下SDK会自动忽略共享订阅相关配置。
2. 订阅时，SDK自动拼接$share/共享组名/主题前缀，用户只需传入原始主题即可。
3. 取消订阅时，必须传入完整的共享订阅主题（含$share前缀），否则无法匹配订阅记录。
4. Broker侧通过共享组名实现消息负载均衡，同一共享组内的客户端会分摊接收消息。

六、版本说明

版本号：V3.0.0（最终定稿版，推荐正式上线）

更新说明：

- 完善错误码体系，新增PKT_ID非法、客户端忙、创建失败等错误码，编号连续无冗余。
- 新增连接状态查询接口、会话级共享组配置，优化共享订阅逻辑。
- 优化结构体命名和字段含义，所有配置项增加合理默认值，简化用户配置成本。
- 支持Windows DLL导出，适配多平台编译，增加自测模式宏定义。
- 完善回调函数设计，参数极简，语义清晰，降低用户实现成本。

技术支持：若有接口使用疑问或问题排查，可联系 hezuo@xxltech.cn

（注：文档部分内容可能由 AI 生成）