行业开发平台设备接入 MQTT SDK 使用说明

版本号：1.0.1

日期	版本	作者	更新记录	备注
2020/03/10	v1.0.0	蒋鹏飞、吴优、陈龙宇	初版发布。	
2020/05/11	v1.0.1	吴优	增加关于设备登录心跳参数说明	

目　　录

一．前言

本文用于说明行业开发平台设备接入MQTT SDK的使用方法，适用于“MCU+标准通信模组”或单板SOC的方案。用户根据使用的硬件平台集成行业SDK并调用相关API接口即可实现行业平台的快速接入。

二．SDK 文件说明

表2-1　SDK文件说明

目录			说明
include			相关头文件
services(业务层)	hy	hy_api	业务接口
		hy_core	业务主逻辑
		hy_reg_svc	设备注册服务
		hy_data_svc	设备数据服务
		hy_data_pack	数据封包解包服务
		hy_params	设备参数存取服务
	user	hy_user_api	行业API接口（用户调用）
		hy_user_func	数据解封包服务
		hy_user_config	产品功能配置文件，由行业开发平台根据创建的功能点生成
protocol(协议层)	mqtt		Paho MQTT库
security(安全层)	tls		wolfssl加密
platforms(平台层)	common		链表创建、内存申请等通用适配接口
	net		网络相关接口
	os		系统相关接口
	storage		数据存取接口
	timer		定时器相关配置接口

三．SDK API 接口说明

3.1 回调接口

3.1.1 用户事件回调

功能	由设备登录接口传入，当需要通知用户事件发生时，会调用所传入函数进行通知
函数指针定义	typedef hy_ret_t (*user_event_cb)(struct user_evt_t *evt)

参数	struct user_evt_t
	{
	enum hy_evt_type_t evt_type;
	uint8_t evt_id;
	};
	• evt_type：事件类型
	• evt_id：事件id或执行id

说明	事件类型仅包括HY_EVT_EVENT与HY_EVT_EXEC

3.1.2 子设备操作回调

功能	由设备登录接口传入，当子设备进行下发或查询命令时，会调用所传入函数进行通知
函数指针定义	typedef hy_ret_t (*user_sub_event_cb)(struct user_sub_evt_t *evt)

参数	struct user_sub_evt_t
	{
	enum hy_evt_type_t evt_type;
	struct sub_dev_info_t dev_info;
	struct sub_func_info_t func_info;
	};
	• evt_type：事件类型
	• dev_info：设备信息（产品id、设备sn）
	• func_info：功能点信息（功能点id、数据、数据长度）

说明	事件类型包括HY_EVT_SUB_WRITE与HY_EVT_SUB_READ

3.2 直连/网关设备接口

3.2.1 设备登录

功能	发起设备注册、登录请求
函数定义	hy_ret_t hy_user_api_login(const uint8_t *product_key, const uint8_t *product_id, const uint8_t *device_sn, uint16_t timeout_s, uint32_t life_time, bool is_security, user_event_cb user_evt_cb, user_sub_event_cb user_sub_evt_cb)

参数	product_key：产品注册码 product_id：产品id device_sn：设备sn timeout_s：登录超时时间（秒） life_time：心跳周期（秒），心跳时间设置必须小于5分钟 is_security：是否加密 user_evt_cb：用户事件回调 user_sub_evt_cb：子设备操作回调
返回值	错误码

3.2.2 设备注销

功能	发起设备注销请求
函数定义	hy_ret_t hy_user_api_logout(void)
参数	无
返回值	错误码

3.2.3 数据解析

功能	数据上下行解析（登录成功后需频繁调用）
函数定义	hy_ret_t hy_user_api_loop(void)
参数	无
返回值	错误码

3.2.4 数据上报

功能	设备主动上报数据
函数定义	hy_ret_t hy_user_api_notify(uint16_t func_id)
参数	func_id：功能点id
返回值	错误码

3.3 网关-子设备接口

3.3.1 子设备注册

功能	发起子设备注册请求

函数定义	hy_ret_t hy_user_api_sub_register(const uint8_t *product_key, const uint8_t *product_id, const uint8_t *device_sn)
参数	product_key：产品注册码 product_id：产品id device_sn：设备sn
返回值	错误码

3.3.2 子设备注销

功能	发起子设备注销请求
函数定义	hy_ret_t hy_user_api_sub_deregister(const uint8_t *product_key, const uint8_t *product_id, const uint8_t *device_sn)
参数	product_key：产品注册码 product_id：产品id device_sn：设备sn
返回值	错误码

3.3.3 子设备上线

功能	发起子设备上线请求
函数定义	hy_ret_t hy_user_api_sub_online(const uint8_t *product_id, const uint8_t *device_sn)
参数	product_id：产品id device_sn：设备sn
返回值	错误码

3.3.4 子设备下线

功能	发起子设备下线请求
函数定义	hy_ret_t hy_user_api_sub_offline(const uint8_t *product_id, const uint8_t *device_sn)

参数	product_id：产品id device_sn：设备sn
返回值	错误码

3.3.5 子设备启用

功能	发起子设备启用请求
函数定义	hy_ret_t hy_user_api_sub_enable(const uint8_t *product_id, const uint8_t *device_sn)
参数	product_id：产品id device_sn：设备sn
返回值	错误码

3.3.6 子设备禁用

功能	发起子设备禁用请求
函数定义	hy_ret_t hy_user_api_sub_disable(const uint8_t *product_id, const uint8_t *device_sn)
参数	product_id：产品id device_sn：设备sn
返回值	错误码

3.3.7 子设备数据上报

功能	主动上报子设备数据
函数定义	hy_ret_t hy_user_api_sub_notify(const uint8_t *product_id, const uint8_t *device_sn, uint16_t func_id, uint8_t *val, uint32_t val_len, uint8_t *time)

参数	product_id：产品id device_sn：设备sn func_id：功能点id val：上传数据 val_len：上传数据长度 time：时间戳（不存在则为NULL）
返回值	错误码

四．底层系统接口

底层系统接口是SDK需要使用的底层功能，包括内存操作、定时器配置、随机数获取等接口。

4.1 内存接口

4.1.1 内存分配

功能	内存分配
函数定义	void *osl_malloc(uint32_t size)
参数	• size：申请内存长度
返回值	若分配成功，则返回指向被分配内存区域的指针；若分配失败，则返回NULL

4.1.2 内存释放

功能	内存释放
函数定义	void osl_free(void *ptr)
参数	• ptr：待释放内存区域地址
返回值	无

4.1.3 内存拷贝

功能	将源内存中的值拷贝到目标内存中
函数定义	void *osl_memcpy(void *dst, const void *src, uint16_t n)
参数	• dst：目标内存地址 • src：源内存地址 • n：拷贝字节数
返回值	无

4.1.4 内存初始化

功能	内存区域初始化，将指定内存区域设置为某个值
函数定义	void *osl_memset(void *dst, uint8_t val, uint16_t n)
参数	• dst：待初始化内存区域地址 • src：待初始化值 • n：待初始化内存区域长度
返回值	无

4.2 时间接口

4.2.1 线程睡眠

功能	毫秒级睡眠
函数定义	void misc_delay_ms(uint32_t ms)
参数	• ms：需要睡眠的毫秒数
返回值	无

4.2.2 倒计时器配置

功能	配置倒计时器
函数定义	tim_handle_t countdown_set(tim_handle_t handle, uint32_t ms)
参数	• handle：倒计时器操作句柄 • ms：倒计时器超时时间（毫秒）
返回值	成功：倒计时器操作句柄，失败：0

4.2.3 倒计时器启动

功能	启动倒计时
函数定义	tim_handle_t countdown_start(uint32_t ms)
参数	• ms：设置倒计时时间（毫秒）
返回值	成功：倒计时器操作句柄，失败：0

4.2.4 倒计时器剩余时间

功能	返回倒计时器剩余时间
函数定义	uint32_t countdown_left(tim_handle_t handle)
参数	• handle：倒计时器操作句柄
返回值	倒计时器剩余时间（毫秒）

4.2.5 倒计时器超时判断

功能	判断倒计时器是否超时
函数定义	uint32_t countdown_is_expired(tim_handle_t handle)
参数	• handle：倒计时器操作句柄
返回值	未超时：0，超时：1

4.2.6 倒计时器停止

功能	停止倒计时器，销毁资源
函数定义	void countdown_stop(tim_handle_t handle)
参数	• handle：倒计时器操作句柄
返回值	无

4.3 其他接口

4.3.1 创建互斥锁

功能	创建互斥锁
函数定义	void mutex_create(void **mutex)
参数	• mutex：用以接收创建完毕的锁的指针
返回值	无

4.3.2 销毁互斥锁

功能	销毁互斥锁

函数定义	void mutex_destroy(void *mutex)
参数	• mutex：需要销毁的锁指针
返回值	无

4.3.3 互斥锁加锁

功能	互斥锁加锁
函数定义	void mutex_lock(void *mutex)
参数	• mutex：需要上锁的锁指针
返回值	无

4.3.4 互斥锁解锁

功能	互斥锁解锁
函数定义	void mutex_unlock(void *mutex)
参数	• mutex：需要解锁的锁指针
返回值	无

五．底层网络接口

底层网络接口是指通信需要使用的网络功能，用于建立物理的网络连接，发送和接收网络数据。

5.1 创建网络连接

功能	建立网络连接
函数定义	int32_t network_connect(int8_t *host, uint16_t port, uint16_t timeout_ms);
参数	• host：目标地址，支持点分十进制IP和域名形式 • port：目标端口 • timeout_ms：执行连接的超时时间（毫秒）
返回值	创建成功：网络连接句柄，失败：-1

5.2 数据发送

功能	发送数据
函数定义	int32_t network_send(int32_t handle, uint8_t *data, uint16_t data_len, uint16_t timeout_ms)
参数	• handle：网络连接句柄 • data：需要发送的数据 • data_len：需要发送的数据长度 • timeout_ms：执行发送的超时时间（毫秒）
返回值	发送成功：发送的数据长度，发送失败：-1，超时：0

5.3 数据接收

功能	接收数据
函数定义	int32_t network_recv(int32_t handle, uint8_t *data, uint16_t data_len, uint16_t timeout_ms)
参数	• handle：网络连接句柄 • data：用于接收数据的缓冲区 • data_len：需要接收的数据长度 • timeout_ms：执行接收的超时时间（毫秒）
返回值	接收成功：接收的数据长度，接收失败：-1，超时：0

5.4 断开网络连接

功能	断开网络连接
函数定义	void network_disconnect(int32_t handle)
参数	• handle：需要断开的网络连接句柄
返回值	无

5.5 创建 SSL 连接

功能	创建带SSL加密的网络连接

函数定义	net_handle_t network_ssl_connect(int8_t *host, uint16_t port, const int8_t *ca_cert, uint16_t ca_cert_len, uint16_t timeout_ms)
参数	• host：目标地址，支持点分十进制IP和域名形式 • port：目标端口 • ca_cert：CA证书 • ca_cert_len：CA证书长度 • timeout_ms：执行连接的超时时间（毫秒）
返回值	创建成功：SSL连接句柄，失败：-1

5.6 数据发送（SSL）

功能	通过SSL连接发送数据
函数定义	int32_t network_ssl_send(net_handle_t handle, uint8_t *data, uint16_t data_len, uint16_t timeout_ms)
参数	• handle：SSL连接句柄 • data：需要发送的数据 • data_len：需要发送的数据长度 • timeout_ms：执行发送的超时时间（毫秒）
返回值	发送成功：发送的数据长度，发送失败：-1，超时：0

5.7 数据接收（SSL）

功能	从SSL连接接收数据
函数定义	int32_t network_ssl_recv(net_handle_t handle, uint8_t *data, uint16_t data_len, uint16_t timeout_ms)
参数	• handle：SSL连接句柄 • data：用于接收数据的缓冲区 • data_len：需要接收的数据长度 • timeout_ms：执行接收的超时时间（毫秒）

返回值	接收成功：接收的数据长度，接收失败：-1，超时：0

5.8 断开 SSL 连接

功能	断开SSL网络连接
函数定义	void network_ssl_disconnect(net_handle_t handle)
参数	• handle：需要断开的SSL连接句柄
返回值	无

六．SDK 移植说明

6.1 配置文件说明

用户在行业平台创建完产品与功能点之后会生成对应的配置文件，分别是hy_user_config.c文件与hy_user_config.h文件，配置文件中包含用户创建的功能点信息，用户需将文件放入services\user目录下。

6.1.1 数据缓冲区设置

用户需在hy_user_config.h文件中设置相应缓冲区的字节大小，具体可根据用户上传的数据量来决定，可以调整的宏包括HY_PACK_DATA_MAX_LEN与HY_PACK_DATA_STR_BUF_MAX_LEN。

6.1.2 数据下发处理

若用户在平台定义的功能点具备下发功能，则在 hy_user_config.c 文件中会生成对应的 func_xxx_set 函数（xxx 为功能点字段名称），用户需在其中根据下发的值实现相应的控制逻辑。

功能	用于实现数据下发操作
函数定义	int32_t func_xxx_set(uint8_t *val, uint32_t val_len)
参数	val：下发数据 val_len：下发数据长度
返回值	操作成功：0，操作失败：1

注：组合功能点的本质是多个基础功能点的集合，所以组合功能点数据下发时会分别进入其包含的基础功能点 set 函数中，用户只需在对应的基础功能点 set 函数中实现下发操作即可。当然，配置文件也会生成组合功能点 set 函数，下发

时也会进入该函数，表明是该组合功能点的下发，用户可以在其中添加打印或其他逻辑。

6.1.3 数据上报与查询处理

若用户在平台定义的功能点具备上报功能，则在 hy_user_config.c 文件中会生成对应的 func_xxx_get 函数（xxx 为功能点字段名称），用户需在其中获取用于上报或查询的功能点的值。用户在 get 函数中实现获取逻辑之后，在需要上报数据的时候调用 3.2.4 接口即可。

功能	用于获取上报或查询的功能点值
函数定义	int32_t func_xxx_get(uint8_t *buf, uint32_t buf_remain_len)
参数	buf：上报或查询数据存放地址 buf_remain_len：缓冲区剩余长度
返回值	上报或查询的数据长度

注：同理下发，组合功能点数据上报或查询时会分别进入其包含的基础功能点 get 函数中，用户只需在对应的基础功能点 get 函数中获取到需要上报的值即可。当然，配置文件也会生成组合功能点 get 函数，上报前也会进入该函数，表明是该组合功能点的上报，用户可以在其中添加打印或其他逻辑。

6.2 移植流程说明

1. 根据具体的硬件平台与对 platforms 文件夹下的底层接口进行适配，主要包括倒计时器接口与网络接口；

2. 行业 SDK 默认运行于 32 位环境，若使用 64 位环境，则将 include\platforms 里 countdown.h 和 network.h 文件中的 typedef 切换成 64bit platform；

3. 调用 3.2.1 接口发起行业平台注册与 OneNET 登录请求，并传入用户自己的回调函数，登录结果通过传入的回调函数进行通知；

4. 登录成功后，频繁调用 3.2.3 接口，保证数据上下行业务的正常；

5. 在配置文件中实现下发控制逻辑与功能点值的获取逻辑，并根据需要调用 3.2.4 接口上传数据；

6. 若存在子设备，则调用 3.3.1 与 3.3.3 进行子设备注册和上线，子设备的下发与查询命令会在用户注册的子设备操作回调函数中通知；

7. 数据上传、下发、查询等结果会在用户注册的事件回调函数中通知。

6.3 移植注意事项

1. 为保证平台下发功能的即时响应，3.2.3 接口必须以尽量小的时间间隔调用，因此勿在主循环中加入等待延时；

2. 勿在回调函数中直接调用设备相关操作函数，例如不要在配置文件的 set 函数中调用数据上报函数，推荐在回调中标记该操作，然后在主循环中执行；

3. SDK 代码使用了部分 C99 特性，编译时必须开启编译器的 C99 特性；

4. 在系统资源比较紧张的情况下，用户可适当减少功能点定义的数量，并尽量避免上传过长的数据（字符型和透传型），以减少内存的占用；

5. 若用户上传或下发的数据量超过 1024 字节，需修改 hy_data_svc.c 中定义的宏 DATA_CLIENT_SEND_BUF_LEN 与 DATA_CLIENT_RECV_BUF_LEN 的大小（默认为 1024）。