PC Dongle RCSP 库文档

珠海市杰理科技股份有限公司
Zhuhai Jieli Technology Co.,LTD
版权所有，未经许可，禁止外传

修改记录

版本	更新日期	描述
V0.0.1	2022-11-10	初始版本

杰理科技 JIELI TECHNOLOGY

目录

一、库文件位置

下面为库文件，分别是 Release, Debug 版本。需要使用 VisualStudio 2015 编译使用

build-bin-out/libdongle.lib

build-bin-out/libdongled.lib

二、示例程序

示例程序为 demo/ 里面的程序。demo/mainwindow.cpp 里面为实现示例程序的代码

杰理科技 JIELI TECHNOLOGY

三、接口说明

大致的流程：

①打开 USB 设备

②构建运行环境

③运行环境中提交任务并等待完成

（一）打开 dongle （USB）设备

```
auto usbDev = std::make_shared<libdongle::UsbDevice>();
if(usbDev->openAsync(/* device path : std::string */)) {
    // OK
}
```

（二）构建运行环境

```
auto ex = std::make_shared<libdongle::Executor>(usbDev);
```

（三）提交任务，并等待任务完成

```
ex->submitTask(/* libdongle:: 命名空间下的任务 */);
ex->runAllTask();
```

所谓的任务，即一些和设备交互的流程。

包括，获取在线设备，进行 OTA 升级等。

1. 任务：获取在线设备

```
auto cb = getDongleCallback();
// cb 是一个 dongle_callback_t 类型，里面是一些回调接口
// 在线设备会通过回调函数返回
ex->submitTask(libdongle::read_online_device(*ex, cb));
ex->runAllTask();
```

2. 任务：OTA 升级远端设备

```
auto cb = getOtaCallback(/* channel */);
// cb 是一个 ota_upgrade_callback_t 类型，里面是一些回调接口
m_ex->submitTask(libdongle::ota_update_client(*m_ex,
                                               channel, /* 远端设备的 channel 号 */,
                                               buf      /* 升级的文件内容 */,
                                               auth1    /* 是否设备需要认证 */,
                                               auth2    /* 是否设备需要认证 */,
                                               cb));
m_ex->runAllTask();
```

3. 任务：OTA 升级 DONGLE 本身

```
bool needPost = false;
m_ex->submitTask(
    libdongle::ota_update_dongle(*m_ex, 2 /* 总是 2 */,
                                 buf      /* 升级文件的内容 */,
                                 auth1    /* 是否设备需要认证 */,
                                 auth2    /* 是否设备需要认证 */,
                                 cb,      /* 是一个 ota_upgrade_callback_t 类型，里面是一些回调 */
                                 needPost /* 是否有后处理步骤 */));
m_ex->runAllTask();

//
// 当执行返回后，needPost 是 true 的时候，需要重新枚举打开 USB 设备（即，DONGLE 设备）
// 然后执行下面的任务
auto usbDev = std::make_shared<libdongle::UsbDevice>();
usbDev->openAsync(/* PATH */);

m_ex = std::make_shared<libdongle::Executor>(usbDev);

m_ex->submitTask(libdongle::ota_update_dongle_single_post(*m_ex,
                    2,      /* 总是 2 */
                    buf,    /* 升级文件内容 */
```

```
                auth2, /* 是否设备需要认证 */
                cb));
m_ex->runAllTask();
```

杰理科技 JIELI TECHNOLOGY

四、回调接口类型说明

```
struct dongle_client_info_t
{
    uint8_t channel_seq;            // 远端设备的 channel 号
    uint8_t support_ota;            // 远端设备是否支持 ota 升级
    uint8_t mac_addr[6];            // 远端设备的 MAC 地址
    char device_name[256 - 8];   // 远端设备的设备名
};

struct dongle_callback_t
{
    // 一个透传的指针，用户自行指定
    void *ctx;

    // 当读取到 DONGLE 设备的信息的时候，调用这个回调函数
    void (*on_dongle_info)(void *ctx,
                           uint32_t sdk_version,
                           uint16_t vid,
                           uint16_t pid,
                           uint16_t pidVersion);

    // 当刷新远端设备列表的时候，通过这个回调函数返回
    void (*on_refresh_client_info)(void *ctx,                    // 透传的指针
                                   uint32_t client_count, // 远端设备个数
                                   struct dongle_client_info_t *client_infos); // 远端设备信息

    // 当出错时候，调用这个回调函数
    void (*on_error)(void *ctx, int error, const char *msg);
};

struct ota_upgrade_callback_t
{
    // 一个透传的指针，用户自行指定
    void *ctx;
```

```
    // OTA 流程开始时，调用这个回调
    void (*on_start_ota)(void *ctx);

    // OTA 流程结束时，调用这个回调
    void (*on_finish_ota)(void *ctx);

    // OTA 过程出现回连的时候，调用这个
    void (*on_need_reconnect)(void *ctx);

    // OTA 流程进度 step 是 0 ~ total 的数值
    void (*on_progress)(void *ctx, int step, int total);

    // OTA 流程停止时候，调用这个
    void (*on_stop_ota)(void *ctx);

    // OTA 流程被取消时候，调用这个
    void (*on_cancel_ota)(void *ctx);

    // OTA 流程出错时候，调用这个
    void (*on_error)(void *ctx, int error, const char *msg);
};
```

杰理科技 JIELI TECHNOLOGY

杰理科技 JIELI TECHNOLOGY