杰理科技异常信息工具使用说明文档

V1.0 - 2022 年 06 月 30 日

目录

声明

- 本项目所参考、使用技术必须全部来源于公知技术信息，或自主创新设计。
- 本项目不得使用任何未经授权的第三方知识产权的技术信息。
- 如个人使用未经授权的第三方知识产权的技术信息，造成的经济损失和法律后果由个人承担。

第 1 章 概述

本文档旨在说明对应版本的固件发生崩溃后，用户需要如何使用杰理手机蓝牙库获取固件的异常信息数据并生成本地文件，并通过杰理异常信息工具打开文件分析异常信息的。

第 2 章 固件说明

在 board_701xxx_demo_global_build_cfg.h 的文件中，将相关宏定义设置为 1。

```
#define CONFIG_DEBUG_RECORD_ENABLE                    1          //是否支持将异常信息记录到 Flash 功能
```

开启后，手表崩溃时会自动保存最近一次异常信息到 flash。通过使用杰理手机蓝牙库获取到固件的异常信息成功后，固件本地会自动清理异常信息（仅限手机获取后会自动清理）。

2.1 获取异常信息接口

导入头文件#include "asm/debug_record.h"，

```
/* ---------------------------------------------------------------------------- */
/**
 * @brief 用户获取异常记录信息
 *
 * @param info: struct debug_record_info 类型，函数执行结果如下:
 *                      1) 系统存在异常信息, record_len 为异常信息长度，单位为 byte;
 *                      2) 系统不存在异常信息, record_len 为 0;
 */
/* ---------------------------------------------------------------------------- */
struct debug_record_info   {
    u32 record_len; //异常记录数据长度
    u8 *record_buf;          //返回异常数据记录地址，该 buf 只读;
};
void user_debug_record_info_get(struct debug_record_info *info);
```

以上 user_debug_record_info_get 函数获得的结构体->record_buf 即是崩溃信息的原始数据，用户可使用该数据自行转换为 bin 文件或通过第 3 章的方法获得 bin 文件，供第 4 章的工具进行解析。

2.2 删除异常信息接口

导入头文件#include "asm/debug_record.h"，

```
/* ---------------------------------------------------------------------------- */
/**
 * @brief 清除 log 信息
 */
/* ---------------------------------------------------------------------------- */
void user_debug_record_info_clear(void);
```

第 3 章 手机 APP 使用说明

3.1 iOS 端 SDK 获取异常信息说明

3.1.1 流程说明

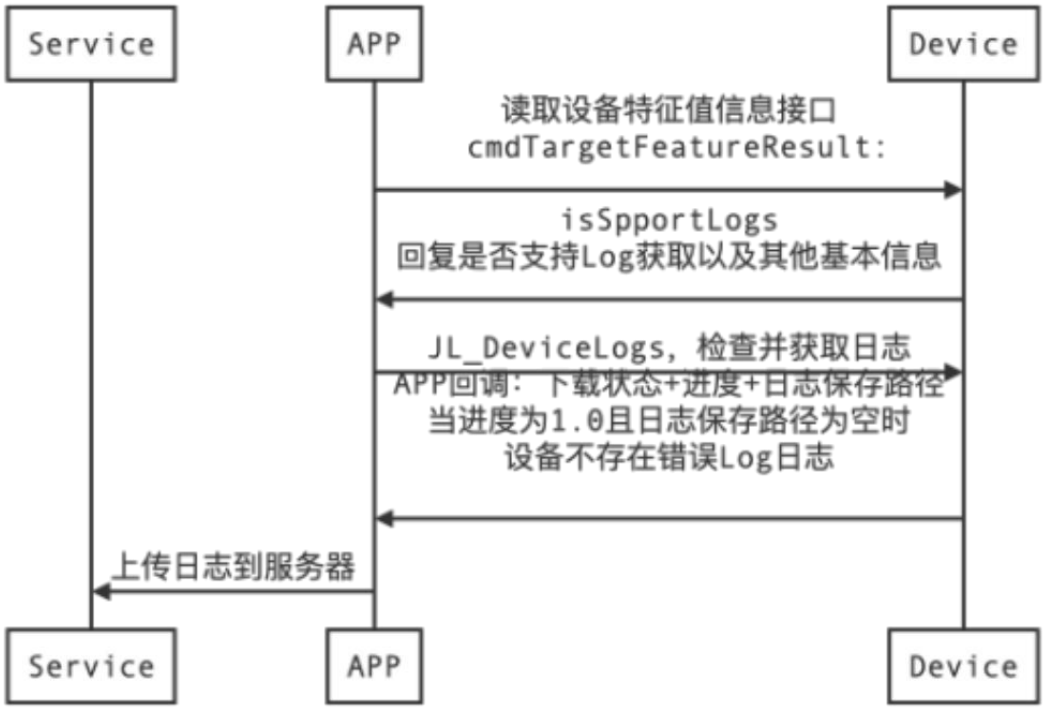
- APP 连接上设备后，会通过读取设备特征值信息接口，拓展功能标记位检查该设备是否支持读取设备日志信息相关内容：

- 通过 JLModel_Device 类 isSupportLog 属性判断是否支持。（可通过 JL_ManagerM 类的 outputDeviceModel 方法获得 JLModel_Device）

- 从设备处获取日志信息： iOS 端通过 JL_ManagerM 类的 mDeviceLogs 属性的 deviceLogDownload：方法进行获取，当设备具备日志信息时，可通过回调结果进行处理

- 日志信息处理：上层应用需要根据实际业务需求，对日志信息进行保存/上传服务器等操作。

工作时序图如下：



3.1.2 示例代码

```
1.  /// 获取设备特征信息
2.  [self.manager cmdTargetFeatureResult:^(JL_CMDStatus status, uint8_t sn, NSData * _Nullable data) {
3.      if (status == JL_CMDStatusSuccess) {
4.          JLModel_Device *model = [self.manager outputDeviceModel];
5.          // 判断是否支持日志获取
6.          if (model.isSupportLog) {
7.
8.              // 获取设备日志
9.              [self.manager.mDeviceLogs deviceLogDownload:^(DeviceLogType type, float progress, NSString * _Nullable tempSavePath) {
10.                 switch (type) {
11.                     case LogTypeDownloading:{
12.                         NSLog(@"progress:%.2f",progress);
13.                     }break;
14.                     case LogTypeSucceed:{
15.                         NSLog(@"device log get success! filePath:%@",tempSavePath);
16.                     }break;
17.                     case LogTypeFailed:{
18.                         NSLog(@"device log get falied! device log may not exit!");
19.                     }break;
20.                     default:
21.                         break;
22.                 }
23.             }];
24.         }
25.     }
26. }];
```

3.1.3 错误码

回调码	说明
LogTypoDownloading	正在下载设备日志
LogTypeFailed	下载失败/设备不存在日志
LogTypeSucceed	日志下载成功

LogTypeNoFile	设备没有错误日志

3.2 Android 端 SDK 获取异常信息说明

3.2.1 流程图

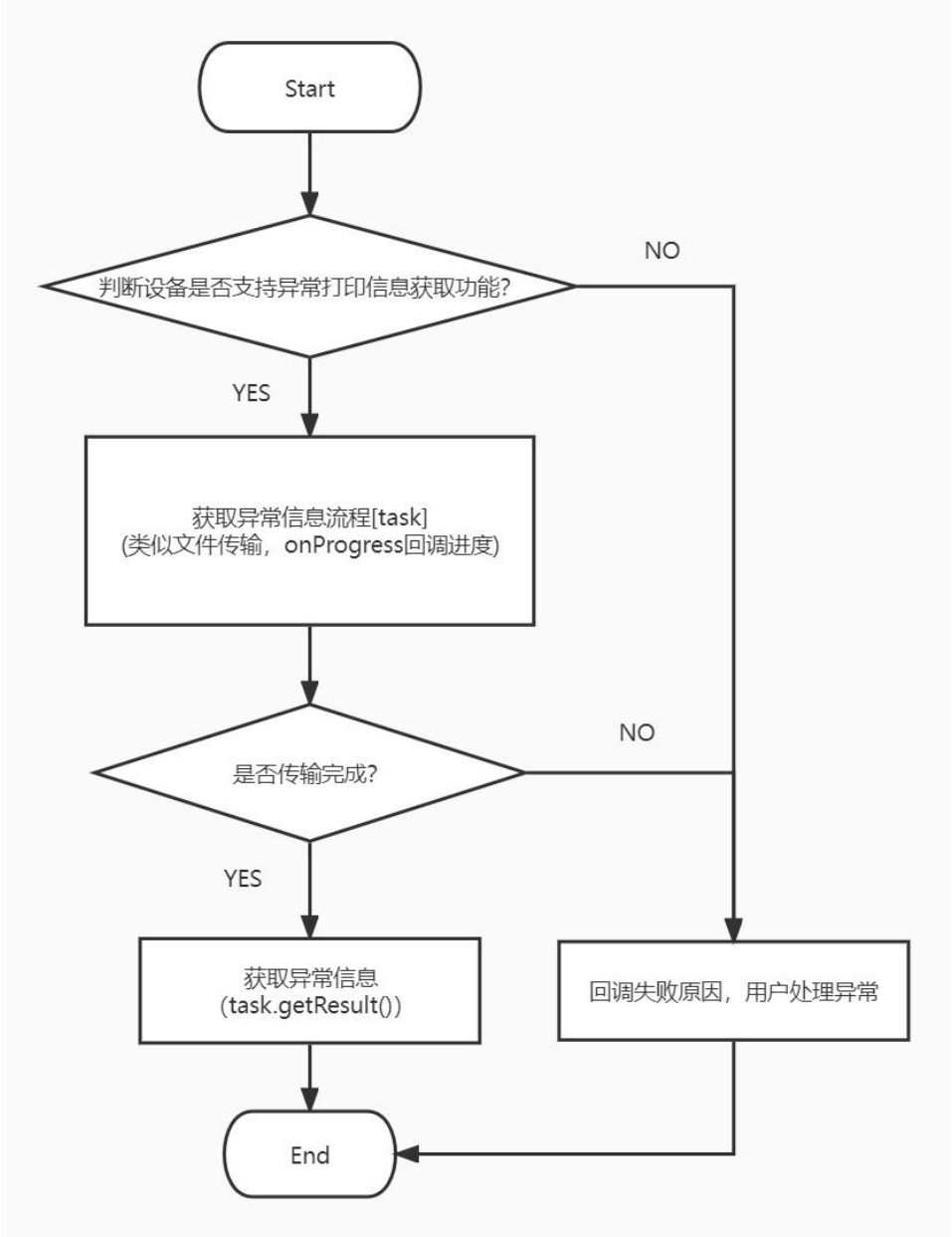


图 3.2.1 -1

说明:

1. 用户可以保存获取的异常信息，也可以上传到服务器。

2. 具体实现参考**[HealthAide]** com.jieli.healthaide.tool.watch.synctask.DeviceLogcatSyncTask

3.2.2 示例代码

```
//创建任务
final ReadLogcatTask task = new ReadLogcatTask(mWatchManager);
//设置监听器
task.setListener(new TaskListener() {
    @Override
    public void onBegin() {
        //回调任务开始
    }

    @Override
    public void onProgress(int progress) {
        //回调进度
    }

    @Override
    public void onFinish() {
        //回调任务完成
        JL_Log.i(tag, "ReadLogcatTask: onFinish : read logcat size = " + task.getResult().length);
        uploadLogFile(task.getResult());
    }

    @Override
    public void onError(int code, String msg) {
        //回调任务异常信息
        JL_Log.w(tag, "ReadLogcatTask: onError : " + code + ", " + msg);
        if (null != finishListener) finishListener.onFinish();
    }

    @Override
    public void onCancel(int reason) {
        //回调任务被取消，该任务暂不支持取消操作
    }
});
//执行任务
task.start();
```

3.2.3 错误码

码值	对应的常量	描述
0x6464	ReadLogcatTask#ERR_DEVICE_NOT_CONNECT	设备未连接
0x6465	ReadLogcatTask#ERR_FUNC_NOT_SUPPORT	功能不支持
0x6466	ReadLogcatTask#ERR_COMMAND_RESPONSE	设备回复失败状态
0x6467	ReadLogcatTask#ERR_SEND_COMMAND	发送命令异常
0x6468	ReadLogcatTask#ERR_WAIT_DATA_TIMEOUT	等待设备回复超时
0x6469	ReadLogcatTask#ERR_CRC_CHECK	CRC 校验失败

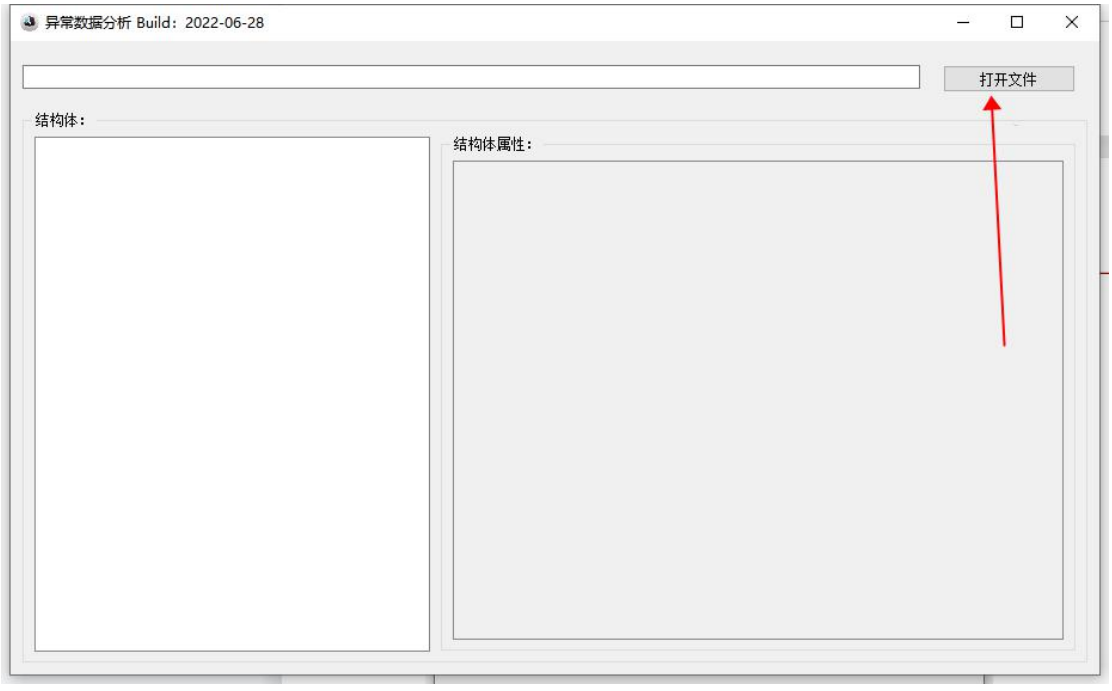
第 4 章 异常数据分析工具使用说明

打开工具， 

点击异常数据分析

，

点击打开文件，



选中第 2 章或第 3 章中获得的 bin 文件，即可查看固件异常日志信息，