|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 【密级程度】 公开 √ 机密 绝密 \_\_   |  | | --- | | **SDK协议流程** |   文件编号：  版 本 号： V1.0  拟制日期： 2017-09-18  发布日期：   |  |  |  | | --- | --- | --- | | 批准 | 审核 | 编制 | |  |  | 罗清文 | | 日期： | 日期： | 日期：2017-09-17 |  |  |  | | --- | --- | | 发文部门 | 研发部 | | 收文部门 | 研发文控 √ | | 市场部 | | 采购部 | | 品质部 | |

# 目录

[目录 2](#_Toc494452699)

[1 概述 3](#_Toc494452700)

[2 协议说明 4](#_Toc494452701)

[2.1 自定义数据类型 4](#_Toc494452702)

[2.2 数据包命令值定义 4](#_Toc494452703)

[2.3 错误码定义 6](#_Toc494452704)

[2.4 UDP包数据格式 7](#_Toc494452705)

[2.5 UDP包数据扩展格式 7](#_Toc494452706)

[2.6 TCP包数据格式 8](#_Toc494452707)

[2.7 TCP数据包扩展格式 9](#_Toc494452708)

[3 协议流程 10](#_Toc494452709)

[3.1 UDP数据包协议流程 10](#_Toc494452710)

[3.2 TCP数据包协议流程 11](#_Toc494452711)

[4 TCP数据包定义 13](#_Toc494452712)

[4.1 传输协议版本协商 13](#_Toc494452713)

[4.2 SDK协议交互 13](#_Toc494452714)

[4.3 SDK协议版本协商 14](#_Toc494452715)

[4.4 错误码反馈 15](#_Toc494452716)

[4.5 TCP心跳包 16](#_Toc494452717)

**SDK协议流程**

# 概述

1. SDK协议使用UDP/TCP传输协议进行数据传输。
2. 下位机UDP监听端口为10001；下位机TCP监听端口为10001。
3. UDP为无连接协议，因此二次开发用户通过UDP Socket对控制卡发送对应端口号的数据包即可进行通信。
4. TCP为有连接协议，因此用户在使用TCP进行通信时，需要先建立TCP连接。建立TCP连接有两种方式：对下位机监听的TCP端口发起建立连接的请求；用户使用自定义的Tcp服务器并将服务器的地址和端口号设置到下位机中，下位机即可自动对该设置的地址发起建立连接请求。
5. SDK协议使用小端字节序；本文档中定义的结构体使用1字节对齐的方式对齐。
6. UDP数据包最大长度为9\*1024字节；TCP数据包最大长度为9\*1024字节。
7. 使用UDP传输协议进行数据传输时，不支持长度超过了9\*1024 - 6的命令包。
8. 使用TCP传输协议进行数据传输时，命令包的长度超过9\*1024 - 4时，需要做分包处理。
9. 下位机使用一个小于等于15字节（最大长度15字节）的设备ID来唯一标识当前控制卡。例如：A603-0000-A0000 和 A30-00-A0000均为合法的设备ID。当设备ID长度小于15字节时，使用’\0’来填充剩余的长度。
10. SDK接口数据均采用UTF-8编码格式进行传输。

# 协议说明

## 自定义数据类型

typedef unsigned long long huint64;

typedef unsigned int huint32;

typedef unsigned short huint16;

typedef unsigned char huint8;

typedef long long hint64;

typedef int hint32;

typedef short hint16;

typedef char hint8;

typedef float hfloat;

typedef double hdouble;

#define MAX\_DEVICE\_ID\_LENGHT 15

#define MD5\_LENGHT 32

#define LOCAL\_TCP\_VERSION 0x1000005

#define LOCAL\_UDP\_VERSION 0x1000005

## 数据包命令值定义

enum HcmdType

{

kTcpHeartbeatAsk = 0x005f, ///< TCP心跳包请求

kTcpHeartbeatAnswer = 0x0060, ///< TCP心跳包反馈

kSearchDeviceAsk = 0x1001, ///< 搜索设备请求

kSearchDeviceAnswer = 0x1002, ///< 搜索设备应答

kErrorAnswer = 0x2000, ///< 出错反馈

kSDKServiceAsk = 0x2001, ///< 版本协商请求

kSDKServiceAnswer = 0x2002, ///< 版本协商应答

kSDKCmdAsk = 0x2003, ///< sdk命令请求

kSDKCmdAnswer = 0x2004, ///< sdk命令反馈

kFileStartAsk = 0x8001, ///< 文件开始传输请求

kFileStartAnswer = 0x8002, ///< 文件开始传输应答

kFileContentAsk = 0x8003, ///< 携带文件内容的请求

kFileContentAnswer = 0x8004, ///< 写文件内容的应答

kFileEndAsk = 0x8005, ///< 文件结束传输请求

kFileEndAnswer = 0x8006, ///< 文件结束传输应答

kReadFileAsk = 0x8007, ///< 回读文件请求

kReadFileAnswer = 0x8008, ///< 回读文件应答

};

## 错误码定义

enum HerrorCode

{

kSuccess = 0, ///< 正常状态

kWriteFinish, ///< 写文件完成

kProcessError, ///< 流程错误

kVersionTooLow, ///< 版本过低

kDeviceOccupa, ///< 设备被占用

kFileOccupa, ///< 文件被占用

kReadFileExcessive, ///< 回读文件用户过多

kInvalidPacketLen, ///< 数据包长度错误

kInvalidParam, ///< 无效的参数

kNotSpaceToSave, ///< 存储空间不够

kCreateFileFailed, ///< 创建文件失败

kWriteFileFailed, ///< 写文件失败

kReadFileFailed, ///< 读文件失败

kInvalidFileData, ///< 无效的文件数据

kFileContentError, ///< 文件内容出错

kOpenFileFailed, ///< 打开文件失败

kSeekFileFailed, ///< 定位文件失败

kRenameFailed, ///< 重命名失败

kFileNotFound, ///< 文件未找到

kFileNotFinish, ///< 文件未接收完成

kXmlCmdTooLong, ///< xml命令过长

kInvalidXmlIndex, ///< 无效的xml命令索引值

kParseXmlFailed, ///< 解析xml出错

kInvalidMethod, ///< 无效的方法名

kMemoryFailed, ///< 内存错误

kSystemError, ///< 系统错误

kUnsupportVideo, ///< 不支持的视频

kNotMediaFile, ///< 不是多媒体文件

kParseVideoFailed, ///< 解析视频文件失败

kUnsupportFrameRate, ///< 不支持的波特率

kUnsupportResolution, ///< 不支持的分辨率(视频)

kUnsupportFormat, ///< 不支持的格式(视频)

kUnsupportDuration, ///< 不支持的时间长度(视频)

kDownloadFileFailed, ///< 下载文件失败

kScreenNodeIsNull, ///< 显示屏节点为空

kNodeExist, ///< 节点已存在

kNodeNotExist, ///< 节点不存在

kPluginNotExist, ///< 插件不存在

kRunningError, ///< 运行错误

kUnsupportMethod, ///< 不支持的方法

kInvalidGUID, ///< 非法的guid

};

## UDP包数据格式

#pragma pack(1)

//结构定义

typedef struct HUdpHeader

{

huint32 version; ///< 版本号

huint16 cmd; ///< 命令值

} HUdpHeader;

#pragma pack()

|  |  |  |
| --- | --- | --- |
| 字段 | 版本号(Version) | 命令(Cmd) |
| 字节数 | 4字节 | 2字节 |

说明：

1. 该数据包为所有使用UDP传输协议数据包的固定头部。
2. 该数据包一般用来对所有下位机发起通信。（并不区分设备）
3. Version表示当前协议的版本号。
4. Cmd表示当前数据包的命令值。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 版本号(Version) | 命令(Cmd) | 设备ID（ID） | 扩展数据（Data） |
| 字节数 | 4字节 | 2字节 | 15字节 |  |

## UDP包数据扩展格式

#pragma pack(1)

//结构定义

typedef struct HUdpExt

{

huint32 version; ///< 版本号

huint16 cmd; ///< 命令值

hint8 devID[MAX\_DEVICE\_ID\_LENGHT]; ///< 设备ID

hint8 ext[0]; ///< 扩展数据起始地址

} HUdpExtAsk, HUdpExtAnswer;

#pragma pack()

说明：

1. 该数据包主要为定向与设备通信而定义。下位机接收到该数据包后，会根据数据包中的“设备ID”字段与设备的ID号进行比较，如果匹配成功则进行解析并处理数据包，否则则丢弃该数据包。
2. Version表示当前协议的版本号。
3. Cmd表示当前数据包的命令值。
4. ID表示设备ID。
5. Data表示扩展的数据内容；Data的最大长度不能超过9\*1024 – 21。

## TCP包数据格式

#pragma pack(1)

//结构定义

typedef struct HTcpHeader

{

huint16 len; ///< 命令包长度

huint16 cmd; ///< 命令值

} HTcpHeader;

#pragma pack()

|  |  |  |
| --- | --- | --- |
| 字段 | 命令包长度(Len) | 命令(Cmd) |
| 字节数 | 2字节 | 2字节 |

说明：

1. 该数据包为所有使用TCP传输协议数据包的固定头部。
2. 由于TCP传输协议面向字节流，可能会出现粘包的情况，所以需要做数据包分离处理。
3. Len表示整个命令包的长度，包括了Len和Cmd字段的长度。
4. Cmd表示当前数据包的命令值。

## TCP数据包扩展格式

#pragma pack(1)

//结构定义

typedef struct HTcpExt

{

huint16 len; ///< 命令包长度

huint16 cmd; ///< 命令值

hint8 ext[0]; ///< 扩展数据起始地址

} HTcpExtAsk, HTcpExtAnswer;

#pragma pack()

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 命令包长度(Len) | 命令(Cmd) | 扩展数据(Data) |
| 字节数 | 2字节 | 2字节 | 9\*1024 – 4字节 |

说明：

1. Len表示整个命令包的字节长度：Len = 4 + DataLen。例如：扩展数据占4字节，那么Len的值为8。
2. Cmd表示当前数据包的命令值。
3. Data表示扩展的数据内容。

# 协议流程

## UDP数据包协议流程



说明：

1. UDP数据包采用一问一答的方式进行通信。
2. 上位机发送请求数据包进行询问，下位机接收到请求数据包后，发送应答数据包进行反馈。
3. 使用UDP数据包通信的主要功能有：
   1. 搜索局域网中的控制卡；
   2. 获取控制卡基本设备信息；
   3. 设置控制卡的有线网络地址信息；
   4. 设置用户自定义TCP服务器地址信息；

## TCP数据包协议流程



说明：

1. 上位机可以主动发起建立TCP连接的请求给下位机以建立TCP连接。
2. 上位机可以将自定义的TCP服务器地址告知下位机，下位机在上电开机后会根据设置的地址发起建立TCP连接的请求与自定义TCP服务器建立TCP连接。
3. 建立好TCP连接后, 双发要互发心跳包来维持连接。
4. 建立好TCP连接后，上位机与下位机进行传输协议版本协商，选取双方都支持的传输协议版本。
5. 假设上位机支持的最低传输协议版本号为PC\_TRAN\_MIN\_VERSION，最高传输协议版本号为PC\_TRAN\_MAX\_VERSION。
6. 假设下位机支持的最低传输协议版本号为ARM\_TRAN\_MIN\_VERSION，最高传输协议版本号为ARM\_TRAN\_MAX\_VERSION。
7. 上位机发送版本协商的请求（版本号为PC\_TRAN\_MAX\_VERSION）给下位机。
8. 下位机收到版本协商的请求后，提取命令包中的版本号（PC\_VERSION），与当前支持的传输协议版本集进行比较：
   1. 如果PC\_VERSION小于ARM\_TRAN\_MIN\_VERSION时，反馈传输协议版本过低的错误信息给上位机。
   2. 如果PC\_VERSION大于ARM\_TRAN\_MAX\_VERSION时，下位机反馈版本信息（版本号为ARM\_TRAN\_MAX\_VERSION）给上位机，并选取传输协议版本为ARM\_TRAN\_MAX\_VERSION。
   3. 如果PC\_VERSION大于等于ARM\_TRAN\_MIN\_VERSION并且PC\_VERSION小于等于ARM\_TRAN\_MAX\_VERSION时，下位机将当前会话支持的传输协议切换至与上位机相同的版本，并反馈与上位机传输协议版本号相同的版本号给上位机。
9. 如果上位机接收到传输协议版本过低的错误码信息后，则应该做断开连接处理。
10. 如果上位机接收到了版本协商的反馈信息后，提取命令包中的版本号（ARM\_VERSION）, 与当前支持的传输协议版本集进行比较：
    1. 如果ARM\_VERSION小于PC\_TRAN\_MIN\_VERSION时，上位机应该断开与下位机的连接，并以非网络传输的方式对下位机进行固件升级或者选取较低版本的上位机与下位机进行网络通信。
    2. 如果ARM\_VERSION大于等于PC\_TRAN\_MIN\_VERSION并且ARM\_VERSION小于等于PC\_TRAN\_MAX\_VERSION时，上位机选取传输协议版本为ARM\_VERSION。
11. 上位机与下位机协商好传输协议后，上位机和下位机就可以使用协商好的传输协议版本进行通信了。
12. 使用TCP数据包通信的主要功能有：
    1. 文件的下发和回读。
    2. XML格式的sdk请求和反馈。（SDK的大部分功能均采用XML格式信息进行交互）

# TCP数据包定义

## 传输协议版本协商

传输协议版本协商由两方面组成：上位机发送“协议版本请求”给下位机；下位机发送“协议版本应答”给上位机。

1. 协议版本请求

#pragma pack(1)

//结构定义

typedef struct HTcpVersion

{

huint16 len; ///< 命令包长度

huint16 cmd; ///< 命令值

huint32 version; ///< 传输协议版本号

} HTcpVersionAsk, HTcpVersionAnswer;

#pragma pack()

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 命令包长度(Len) | 命令(Cmd) | 版本号(Version) |
| 字节数 | 2字节 | 2字节 | 4字节 |
| 取值 | 8 | kSDKServiceAsk | PC\_VERSION |

1. 协议版本应答

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 命令包长度(Len) | 命令(Cmd) | 版本号(Version) |
| 字节数 | 2字节 | 2字节 | 4字节 |
| 取值 | 8 | kSDKServiceAnswer | ARM\_VERSION |

## SDK协议交互

SDK协议交互由两方面组成：上位机发送“SDK接口请求”给下位机；下位机发送“SDK接口应答”给上位机。

1. SDK接口请求

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段 | 命令包长度(Len) | 命令(Cmd) | 总长度 | 索引 | SDK接口数据 |
| 字节数 | 2字节 | 2字节 | 4字节 | 4字节 | N字节 |
| 取值 | Len | kSDKCmdAsk | Total | Index | XmlData |

说明：

#pragma pack(1)

//结构定义

typedef struct HTcpSDKCmd

{

huint16 len; ///< 命令包长度

huint16 cmd; ///< 命令值

huint32 total; ///< sdk接口数据总长度

huint32 index; ///< sdk接口数据当前索引

hint8 xml[1]; ///< sdk接口数据

} HTcpSDKCmdAsk, HTcpSDKCmdAnswer;

#pragma pack()

1. Len值计算：Len = 2 + 2 + 4 + 4 + N;
2. Total表示：SDK接口数据的总长度；total最大值为2\*1024\*1024；
3. Index表示：当前数据包中的XmlData在SDK接口数据中的索引位置；
4. XmlData表示：SDK接口数据内容；XmlData在当前数据包中的最大长度值为9\*1024 – (2 + 2 + 4 + 4)个字节；
5. SDK接口应答

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段 | 命令包长度(Len) | 命令(Cmd) | 总长度 | 索引 | SDK接口数据 |
| 字节数 | 2字节 | 2字节 | 4字节 | 4字节 | N字节 |
| 取值 | Len | kSDKCmdAnswer | Total | Index | XmlData |

## SDK协议版本协商

SDK协议版本协商使用“SDK协议交互”的数据传输方式进行通信，本小结只叙述SDK接口数据的内容；它由两方面组成：上位机发送“SDK协议版本请求”给下位机；下位机发送“SDK协议版本应答”给上位机。

1. SDK协议版本请求的SDK接口数据

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<in method=**"GetIFVersion"**>

<version value=**"1000000"**/>

</in>

</sdk>

说明：

1. 在建立TCP连接并协商完传输协议版本后，应发送“SDK协议版本请求”给下位机进行SDK协议版本协商。
2. 该数据包中的字段只有version属性value的值是可变的，其他属性值均为固定值。
3. version属性value的值为表示sdk协议版本的16进制数值表示。
4. sdk协议当前版本为0x1000000。
5. SDK协议版本应答的SDK接口数据

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"f7b3b451c4f3cf3310e2d6714fbb50e8"**>

<out method=**"GetIFVersion"** result=**"kSuccess"**>

<version value=**"1000000"**/>

</out>

</sdk>

说明：

1. 上位机发送SDK协议版本请求给下位机后，下位机会返回一个guid值，该值用来为后续SDK协议交互使用。
2. out标签的属性result取值为非kSuccess表示出错，此时out无子节点。
3. result的取值参考2.3中的错误码定义。
4. version属性value的值为表示sdk协议版本的16进制数值表示。
5. sdk协议当前版本为0x1000000。

## 错误码反馈

#pragma pack(1)

//结构定义

typedef struct HCodeReply

{

huint16 len; ///< 命令包长度

huint16 cmd; ///< 命令值

huint16 code; ///< 错误码

} HCodeReply;

#pragma pack()

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 命令包长度(Len) | 命令(Cmd) | 错误码(Code) |
| 字节数 | 2字节 | 2字节 | 2字节 |
| 取值 | 6 | kErrorAnswer | code |

说明：

1. 下位机出错后，可通过发送该数据包通知上位机出错，并反馈相应的错误码。
2. 上位机出错后，可通过发送该数据包通知下位机出错，并反馈相应的错误码。
3. 下位机接收到错误码反馈后，下位机恢复到TCP连接刚建立好的状态。
4. code取值参考2.3错误码定义。

## TCP心跳包

1. TCP心跳包请求

|  |  |  |
| --- | --- | --- |
| 字段 | 命令包长度(Len) | 命令(Cmd) |
| 字节数 | 2字节 | 2字节 |
| 取值 | 4 | kTcpHeartbeatAsk |

说明：

上位机应该在超过30秒无任何数据发送给下位机时，发送此命令包给下位机用来维持连接的心跳；下位机以60秒为基准判断，如果超过60秒上位机没有发送任何数据给下位机，那么下位机会主动断开该TCP连接。

1. TCP心跳包反馈

|  |  |  |
| --- | --- | --- |
| 字段 | 命令包长度(Len) | 命令(Cmd) |
| 字节数 | 2字节 | 2字节 |
| 取值 | 4 | kTcpHeartbeatAnswer |

说明：

下位机在超过30秒无任何数据发送给上位机时，发送此命令包给上位机用来维持连接的心跳。