版本 1.2

2023-05-08



IPC 应用程序接口

API 参考

# IPC 应用程序接口

## 前言

### 读者对象

本参考适用于程序员阅读，描述了基于IPC应用库开发的各种参考信息。使用本参考的程序员应该：

* 熟练使用 C/C++语言
* 掌握基本的库函数调用
* 熟悉Windows或Linux等平台的开发环境

### 内容简介

本参考首先概述了IPC应用库 API 函数种类及其关联，然后分别详细介绍了各种参考信息。本参考内容组织如下。

|  |  |
| --- | --- |
| 章节 | 内容 |
| 概述 | 介绍IPC应用库开发包组件和软硬件开发环境。  阅读本主题后，您将对客户端IPC开发库有一个整体了解 |
| API函数 | 本主题供您查阅开发库的API参考信息，详细介绍每一个API接口函数 |
| 数据类型 | 介绍API用到的通用数据类型定义及结构定义 |
| API应用实例 | 通过实例介绍开发库API的使用方法 |

### 格式约定

|  |  |
| --- | --- |
| 格式 | 说明 |
| 黑体 | 正文和标题 |
| 楷体 | 警告、提示等内容一律用楷体，并且在内容前后增加线条与正文隔离 |
| 等宽 | 代码采用等宽字体 |

### 修改记录

|  |  |  |
| --- | --- | --- |
| 版本 | 日期 | 修改说明 |
| 1. .2 | 2023-05-08 | 增加云台控制函数说明 |
| 1.1 | 2022-12-08 | 增加API函数说明 |
| 1.0 | 2022-05-08 | 第一次版本 |

目录

[IPC 应用程序接口 1](#_Toc121494686)

[前言 1](#_Toc121494687)

[读者对象 1](#_Toc121494688)

[内容简介 1](#_Toc121494689)

[格式约定 1](#_Toc121494690)

[修改记录 1](#_Toc121494691)

[概述 3](#_Toc121494692)

[描述范围 3](#_Toc121494693)

[函数列表 4](#_Toc121494694)

[函数描述方式 4](#_Toc121494695)

[结构体描述方式 5](#_Toc121494696)

[函数说明 6](#_Toc121494697)

[NETSDK\_Init 6](#_Toc121494698)

[NETSDK\_Cleanup 6](#_Toc121494699)

[NETSDK\_SearchDevice 7](#_Toc121494700)

[NETSDK\_Login 8](#_Toc121494701)

[NETSDK\_Logout 9](#_Toc121494702)

[NETSDK\_CreateRealPlay 10](#_Toc121494703)

[NETSDK\_ DestoryRealPlay 10](#_Toc121494704)

[NETSDK\_ StartRealPlay 11](#_Toc121494705)

[NETSDK\_ SetRealStreamStatus 12](#_Toc121494706)

[NETSDK\_ StopRealPlay 14](#_Toc121494707)

[数据类型与数据结构 14](#_Toc121494708)

[通用数据类型描述 14](#_Toc121494709)

[数据结构描述 15](#_Toc121494710)

[API 应用实例 18](#_Toc121494711)

[获取码流的流程图 18](#_Toc121494712)

[程序实例 19](#_Toc121494713)

## 概述

### 描述范围

本公司提供的IPC应用库软件是一套高性能、高可靠性、兼容性良好的应用开发包。其内部完成了IPC通信和获取流数据的主要流程，并对外提供了灵活简单的 API，用户可快速地开发应用程序。

本软件为用户提供各种平台下的静态库调用形式，可更方便地开发应用程序。主要组件及相关说明如表 1所示。

表 1

|  |  |  |
| --- | --- | --- |
| 组件 | 名称 | 说明 |
| API接口 | ipsdk\_client.h  ipsdk\_data\_def.h  ipsdk\_debug.h  ipsdk\_def\_type.h  ipsdk\_erp.h  ipsdk\_server.h  ipsdk\_version.h | 用户工程中，只需包含 ipsdk\_client.h即可，此文件会包含其他必须文件 |
| 静态库 | yssdk\_client.lib | 静态库分多平台（Windows/Linux），多架构（x86和x64），多目标（Debug/Release），编译的时候要链接正确的版本 |
| 示范代码 |  |  |

用户可在多种编译环境上进行基于应用库的应用程序开发，兼容微软公司的Windows 7 或更高版本的主流视窗操作系统，兼容 Intel 公司和 AMD 公司自 2002年来推出的绝大部分面向 PC 机的 CPU 芯片组。其主要开发以及运行环境说明如表 1-2所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 分类 | 兼容配置 | 推荐配置 | 说明 |
| 编译器 | Visual Studio 2015 | Visual Studio 2017 |  |
| 操作系统 | Windows 7  Windows 8  Windows 10  Windows 11 | Windows 10+ |  |
| 硬件 | Intel P3 系列  Intel P4 系列  Intel Core 系列  AMD Athlon64 系列  AMD Sempron 系列  AMD Athlon 系列 | CPU 主频在3.0GHz 以  上、内存大小在 512MB 以上的 PC |  |

### 函数列表

|  |  |
| --- | --- |
| 函数 | 功能 |
| NETSDK\_Init | 创建、初始化SDK库 |
| NETSDK\_Cleanup | 清理、销毁分配的内存 |
| NETSDK\_SearchDevice | 搜索网络中的IPC设备 |
| NETSDK\_Login | 登录目标IPC |
| NETSDK\_Logout | 注销登录 |
| NETSDK\_CreateRealPlay | 建立流获取对象句柄 |
| NETSDK\_DestoryRealPlay | 关闭并销毁流获取对象句柄 |
| NETSDK\_StartRealPlay | 获取实时流数据 |
| NETSDK\_SetRealStreamStatus | 相应实时流状态的通知 |
| NETSDK\_StopRealPlay | 停止获取流数据 |
| NETSDK\_ProductTest | 云台操作 |

### 函数描述方式

本章用6个域对API参考信息进行描述。

|  |  |
| --- | --- |
| 参数域 | 作用 |
| 目的 | 简要描述 API 的主要功能。 |
| 语法 | 列出 API 的语法样式。 |
| 描述 | 简要描述 API 的工作过程。 |
| 参数 | 列出 API 的参数、参数说明及参数属性。 |
| 返回值 | 列出 API 的返回值及返回值说明。 |
| 注意 | 使用 API 时应注意的事项。 |

### 结构体描述方式

|  |  |
| --- | --- |
| 参数域 | 作用 |
| 说明 | 简要描述结构体所实现的功能。 |
| 定义 | 列出结构体的定义。 |
| 注意事项 | 列出结构体的注意事项。 |

## 函数说明

### NETSDK\_Init

#### 目的

创建、初始化

#### 语法

*IPSDK\_Int* NETSDK\_Init();

#### 描述

初始化SDK库，分配上下文内存和环境

#### 参数

无

#### 返回值

|  |  |
| --- | --- |
| 返回值 | 描述 |
| 0 | 成功 |
| 非0 | 初始化失败，内存分配失败或网络不可用 |

#### 注意

此函数必须在所有函数调用之前调用，否则所有函数都不会正常工作

### NETSDK\_Cleanup

#### 目的

清理、销毁SDK库

#### 语法

*IPSDK\_Int* NETSDK\_Cleanup();

#### 描述

使用结束后，销毁SDK工作时分配的内存空间，以防止内存泄漏

#### 参数

无

#### 返回值

|  |  |
| --- | --- |
| 返回值 | 描述 |
| 0 | 成功 |
| 非0 | 失败 |

#### 注意

此函数必须在最后调用，不允许重复销毁。

### NETSDK\_SearchDevice

#### 目的

获得网络中所有设备的基本信息

#### 语法

*IPSDK\_Int* NETSDK\_SearchDevice(*fSearchDevCallBack* *fSearchCallback*,

*IPSDK\_PVoid* *lParam*, *IPSDK\_Int* *nTimeout*, *IPSDK\_ULong* *ulBindAddr*);

#### 描述

用户可通过此函数查看当前网络中有多少设备以及各个设备的基本信息。

#### 参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 取值范围 | 输入/输出 | 描述 |
| fSearchCallback | - | 输入 | 回调函数地址 |
| lParam |  | 输入 | 提供给回调函数的参数 |
| nTimeout | 0 – ∞ | 输入 | 网络通信等待超时上限 |
| ulBindAddr | - | 输入 | 本机的IP地址 |

#### 返回值

|  |  |
| --- | --- |
| 返回值 | 描述 |
| 0 | 成功调用 |
| 非0 | 调用函数出错 |

#### 注意

此函数使用了回调，因此在使用之前必须先声明和定义fSearchCalback回调函数，回调函数的声明为：

typedef *IPSDK\_Int*(IPSDK\_CALLBACK \**fSearchDevCallBack*)(*IPSDK\_Int* nStateCode, *IPSDK\_SearchDevice*\* pSearchDevice, *IPSDK\_PVoid* pUserParam);

##### 回调函数说明

|  |  |  |
| --- | --- | --- |
| 参数 | 成员 | 描述 |
| nStateCode |  | 通知代码 |
| pSearchDevice | strDeviceId | 设备ID |
|  | stLanInfo | 设备网络信息（包含网络地址、通信端口等） |
|  | stDeviceInfo | 设备信息（包含版本、硬件、云ID等） |
| pUserParam |  | 调用NETSDK\_SearchDevice的时候传递给lParam的值 |

函数在执行当中每找到一台设备就会通过回调函数通知用户，用户可以保存设备相关信息以便在后续的执行中使用。

### NETSDK\_Login

#### 目的

登录目标IPC，创建有效会话句柄

#### 语法

*IPSDK\_HLOGIN* NETSDK\_Login(const *IPSDK\_Char*\* *pstrHostAddr*, *IPSDK\_Int* *nSessionPort*, *IPSDK\_Int* *nStreamPort*, const *IPSDK\_Char*\* *pstrUserName*, const *IPSDK\_Char*\* *pstrPassword*, *IPSDK\_PVoid* *lParam*);

#### 描述

初始化会话，建立和IPC通信的渠道，同时IPC会校验用户的有效性

#### 参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 取值范围 | 输入/输出 | 描述 |
| pstrHostAddr | - | 输入 | 目标IPD的IP地址 |
| nSessionPort | 1-65535 | 输入 | 通话端口 |
| nStreamPort | 1-65535 | 输入 | 流采集端口 |
| pstrUserName |  | 输入 | 用户名称 |
| pstrPassword |  | 输入 | 用户密码 |
| lParam |  |  | 保留参数 |

#### 返回值

|  |  |
| --- | --- |
| 返回值 | 描述 |
| 0 | 登录失败 |
| 非0 | 登录成功，返回值为会话句柄 |

#### 注意

无

### NETSDK\_Logout

#### 目的

销毁会话句柄

#### 语法

*IPSDK\_Int* NETSDK\_Logout(*IPSDK\_HLOGIN* *hUserId*);

#### 描述

会话结束后，销毁分配的内存空间，以防止内存泄漏

#### 参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 取值范围 | 输入/输出 | 描述 |
| hUserId | - | 输入 | 待销毁的会话句柄 |

#### 返回值

|  |  |
| --- | --- |
| 返回值 | 描述 |
| 0 | 成功 |
| 非0 | 失败 |

#### 注意

会话句柄的合法性由用户保证，不允许重复销毁。

销毁后的句柄需要手动置空。

### NETSDK\_CreateRealPlay

#### 目的

创建有效流通道句柄

#### 语法

*IPSDK\_HCHANNEL* NETSDK\_CreateRealPlay(*IPSDK\_HLOGIN* *hHandle*, const *IPSDK\_ChannelInfo*\* *pChannelInfo*);

#### 描述

在建立会话之后，建立和IPC的流通道，为播放做准备。

#### 参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 成员 | 输入/输出 | 描述 |
| hHandle | - | 输入 | 会话句柄 |
| pChannelInfo | nChannel | 输入 | 设备通道 |
|  | nStream | 输入 | 码流通道 |

#### 返回值

|  |  |
| --- | --- |
| 返回值 | 描述 |
| 0 | 建立通道失败 |
| 非0 | 成功建立通道，返回值为通道句柄 |

#### 注意

用户要保证会话句柄的有效性。

目前设备通道一般为0。

码流通道目前有两个，0代表主码流，1代表从码流。

### NETSDK\_ DestoryRealPlay

#### 目的

销毁流通道句柄

#### 语法

*IPSDK\_Int* NETSDK\_DestoryRealPlay(*IPSDK\_HCHANNEL* *hChannel*);

#### 描述

取码流完毕后，销毁分配的内存空间，以防止内存泄漏

#### 参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 取值范围 | 输入/输出 | 描述 |
| hChannel | - | 输入 | 待销毁的通道句柄 |

#### 返回值

|  |  |
| --- | --- |
| 返回值 | 描述 |
| 0 | 成功 |
| 非0 | 失败 |

#### 注意

通道句柄的合法性由用户保证，不允许重复销毁。

销毁后的句柄需要手动置空。

### NETSDK\_ StartRealPlay

#### 目的

获得IPC中的数据流

#### 语法

*IPSDK\_Int* NETSDK\_StartRealPlay(*IPSDK\_HCHANNEL* *hChannel*, *fRealStreamCallBack* *cbRealDataCallBack*, *IPSDK\_PVoid* *pUserData*, *IPSDK\_Int* *bBlocked*);

#### 描述

用户可通过此函数获取IPC的实时码流。

#### 参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 取值范围 | 输入/输出 | 描述 |
| hChannel | - | 输入 | 通道句柄 |
| cbRealDataCallBack | - | 输入 | 接收流数据的回调函数 |
| pUserData | - | 输入 | 保留给用户的参数 |
| bBlocked | TRUE/FALSE | 输入 | 同步/异步方式 |

#### 返回值

|  |  |
| --- | --- |
| 返回值 | 描述 |
| 0 | 成功调用 |
| 非0 | 调用函数出错 |

#### 注意

此函数使用了回调，因此在使用之前必须先声明和定义fRealStreamCallBack回调函数，回调函数的声明为：

typedef *IPSDK\_Int*(IPSDK\_CALLBACK \**fRealStreamCallBack*)(*IPSDK\_HANDLE* lRealHandle, *IPSDK\_Int* nChannel, *IPSDK\_Int* nStream, *IPSDK\_Int* nDataType, *IPSDK\_PChar* pDataBuf, *IPSDK\_Int* nDataLen, *IPSDK\_PVoid* pUserBuf);

##### 回调函数说明

|  |  |
| --- | --- |
| 参数 | 描述 |
| lRealHandle | 通道句柄 |
| nChannel | 通道号 |
| nStream | 码流号 |
| nDataType | 数据类型 |
| pDataBuf | 码流数据起始地址 |
| nDataLen | 码流数据长度 |
| pUseBuf | 调用NETSDK\_StartRealPlay的时候传递给pUserData的值 |

函数在执行当中每获得一个数据包就会通过回调函数提供给用户，用户在回调函数中获得数据之后可以进行保存、解码播放、转码等各种操作。

### NETSDK\_ SetRealStreamStatus

#### 目的

获取数据流的过程中，实时了解当前通信状态

#### 语法

*IPSDK\_Int* NETSDK\_SetRealStreamStatus(*IPSDK\_HCHANNEL* *hChannel*, *fStatusEventCallBack* *fStatusEvent*, *IPSDK\_PVoid* *lParam*);

#### 描述

用户可通过此函数获取IPC的实时通信状态。

用户可以通过在fStatusEvent回调函数中返回非0数据来退出。

#### 参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 取值范围 | 输入/输出 | 描述 |
| hChannel | - | 输入 | 通道句柄 |
| fStatusEvent | - | 输入 | 接收状态通知的回调函数 |
| lParam | - | 输入 | 保留给用户的参数 |

#### 返回值

|  |  |
| --- | --- |
| 返回值 | 描述 |
| 0 | 成功调用 |
| 非0 | 调用函数出错 |

#### 注意

此函数使用了回调，因此在使用之前必须先声明和定义fStatusEventCallBack回调函数，回调函数的声明为：

typedef *IPSDK\_Int*(IPSDK\_CALLBACK \**fStatusEventCallBack*)(*IPSDK\_Int* nStateCode, *IPSDK\_PChar* pResponse, *IPSDK\_PVoid* pUserParam);

##### 回调函数说明

|  |  |
| --- | --- |
| 参数 | 描述 |
| nStateCode | 状态代码 |
| pResponse | 消息内容 |
| pUserParam | 调用NETSDK\_SetRealStreamStatus的时候传递给lParam的值 |

函数在执行当中通过此回调函数提供给用户当前运行状态，用户通过状态代码了解实时的通信情况，还可以通过返回非0的代码立刻退出当前获取码流的运行过程。

### NETSDK\_ StopRealPlay

#### 目的

停止取码流过程

#### 语法

*IPSDK\_Int* NETSDK\_StopRealPlay(*IPSDK\_HCHANNEL* *hChannel*);

#### 描述

停止由StartRealPlay创建的取码流过程

#### 参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 取值范围 | 输入/输出 | 描述 |
| hChannel | - | 输入 | 通道句柄 |

#### 返回值

|  |  |
| --- | --- |
| 返回值 | 描述 |
| 0 | 成功 |
| 非0 | 失败 |

#### 注意

通道句柄的合法性由用户保证。

### NETSDK\_ ProductTest

#### 目的

对目标IPC发送指令ID以及包含JSON内容的参数，达到操作云台的各种功能的效果

#### 语法

*IPSDK\_Int* NETSDK\_ProductTest(*IPSDK\_HLOGIN* *hUserId*, *IPSDK\_Int nTestId*, *fProductTestCallBack* *cbProductTest*, *IPSDK\_CPChar* *pParamJson*, *IPSDK\_PVoid* *pUser*);

#### 描述

用户可通过此函数操作云台的移动（上下左右）、停止、复位以及镜头（变倍、光圈、变焦）的调整。

用户通过在cbProductTest回调函数中获取IPC更多的反馈信息。

#### 参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数 | 取值范围 | | 输入/输出 | 描述 |
| hUserId | - | | 输入 | 用户当前的登录句柄，通过NETSDK\_Login 获取 |
| nTestId | 80-85 | | 输入 | 云台操作代码 |
| 80 | pParamJson = {"direction":"up","speed":1} | 云台移动   * direction: 云台转动方向，up/down/left/right对应上/下/左右四个方位 * speed:云台转动速度（1-4） |
| 81 | pParamJson = nullptr | 云台停止 |
| 82 | pParamJson = nullptr | 云台复位 |
| 83 | pParamJson = nullptr | 查询云台预置位信息 |
| 84 | pParamJson = {"operate":"set","preset":1} | 云台预置操作   * operate:预置位操作。 * set:设置预置位 * call:调用预置位 * clear:清除预置位 * preset:当前操作的预置位标号 |
| 85 | pParamJson = {"operate":"zoom\_tele"} | 云台镜头操作   * operate:镜头操作。 * zoom\_tele:变倍（放大） * zoom\_wide:变倍（缩小） * focus\_near:变焦（近焦） * focus\_far:变焦（远焦） * iris\_open:光圈（打开阑孔） * iris\_close:光圈（关闭阑孔） |
| cbProductTest | - | | 输入 | 接收云台操作反馈的回调函数 |
| pParamJson | -参考nTestId参数 | | 输入 | 与nTestId相关的附加参数 |
| pUser | - | | 输入 | 保留给用户的参数 |

#### 返回值

|  |  |
| --- | --- |
| 返回值 | 描述 |
| 0 | 成功调用 |
| 非0 | 调用函数出错 |

#### 注意

此函数使用了回调，因此在使用之前必须先声明和定义fProductTestCallback回调函数，回调函数的声明为：

typedef *IPSDK\_Int*(IPSDK\_CALLBACK \* *fProductTestCallback*)(*IPSDK\_Int* nTestId, *IPSDK\_Int* nCode, *IPSDK\_CPChar* pResultJson, *IPSDK\_PVoid* pUser);

##### 回调函数说明

|  |  |
| --- | --- |
| 参数 | 描述 |
| nTestId | 和NETSDK\_ ProductTest 中的nTestId一样，便于回调函数区分当前操作 |
| nCode | 状态代码，0代表成功 |
| pResultJson | 返回给用户的内容 |
| 当nTestId == 83时，消息内容为当前预设信息，例如：{“list”:[1,2,3]} |
| pUser | 调用NETSDK\_ ProductTest的时候传递给pUser的值 |

函数在执行当中通过此回调函数获取返回信息。

## 数据类型与数据结构

### 通用数据类型描述

API 用到的主要数据类型定义如下

typedef     *char*                *IPSDK\_Int8*;

typedef     *unsigned* *char*       *IPSDK\_UInt8*;

typedef     *short*               *IPSDK\_Int16*;

typedef     *unsigned* *short*      *IPSDK\_UInt16*;

typedef     *int*                 *IPSDK\_Int32*;

typedef     *unsigned* *int*        *IPSDK\_UInt32*;

typedef     *unsigned* *long* *long*  *IPSDK\_UInt64*;

typedef     *unsigned* *long*       *IPSDK\_ULong*;

typedef     *long*                *IPSDK\_Long*;

typedef     *char*                *IPSDK\_Char*;

typedef     *char*\*               *IPSDK\_PChar*;

typedef     const *char*\*         *IPSDK\_CPChar*;

typedef     *unsigned* *char*       *IPSDK\_Byte*;

typedef     *unsigned* *char*\*      *IPSDK\_PByte*;

typedef     *int*                 *IPSDK\_Int*;

typedef     *void*                *IPSDK\_Void*;

typedef     *void*\*               *IPSDK\_PVoid*;

typedef     *void*\*               *IPSDK\_HANDLE*;

typedef     *int*                 *IPSDK\_Boolean*;

typedef     *float*               *IPSDK\_Float*;

typedef     *int*                 *IPSDK\_ERR*;

typedef     *IPSDK\_Int32*         *IPSDK\_Size*;

### 数据结构描述

#### IPSDK\_SearchDevice

##### 说明

搜索设备的过程中返回给用户的设备信息描述

##### 定义

typedef *struct* *tag\_ipsdk\_device\_info\_*

{

*IPSDK\_Char* strDevModel[32];         /// 设备型号定义。

*IPSDK\_Char* strDevMagic[64];         /// 设备平台唯一号

*IPSDK\_Char* strDevName[32];          /// 设备名称

*IPSDK\_Char* strDevBuildDate[32];     /// 设备发布版本日期

*IPSDK\_Char* strDevHardVer[32];       /// 硬件版本号

*IPSDK\_Char* strDevSoftVer[32];       /// 软件版本号

*IPSDK\_Char* strDevSerialId[32];      /// 出厂序列号

*IPSDK\_Char* strDevCloudId[32];       /// 设备的云 ID 号。

*IPSDK\_Char* strDevOemName[32];       /// 设备OEM厂家名字

*IPSDK\_Char* strDevOemId[32];         /// 设备OEM的ID

*IPSDK\_Char* strDevOemSN[64];         /// 设备平台唯一号

*IPSDK\_Char* res[64];                 /// 扩展结构

} *IPSDK\_DeviceInfo*;

typedef *struct* *tag\_ipsdk\_lan\_info\_*

{

*IPSDK\_Int* nSessionPort;             /// 会话端口

*IPSDK\_Int* nStreamPort;              /// 直播端口

*IPSDK\_Char* strDns1[64];             /// dns域名

*IPSDK\_Char* strDns2[64];             /// dns域名

*struct*

    {

*IPSDK\_Char* strNetType[16];      /// 网络类型: 有线[Wired]/无线[Wireless]

*IPSDK\_Char* strIpAddr[16];

*IPSDK\_Char* strMask[16];

*IPSDK\_Char* strGateway[16];

*IPSDK\_Char* strMacAddr[24];

*IPSDK\_Int* bEnableDhcp;

*EN\_IPSDK\_AllNetConnect* enAllNetConnect; /// 全网通

    } Lan[2];

*IPSDK\_Char* res[64];                 /// 扩展结构

} *IPSDK\_LanInfo*;

typedef *struct* *tag\_IPSDK\_UserInfo*

{

*IPSDK\_Char* strUserName[IPSDK\_USER\_MAX\_NAME\_SZ];     /// 用户名

*IPSDK\_Char* strPassword[IPSDK\_USER\_MAX\_PASS\_SZ];     /// 密码

*IPSDK\_Char* res[64];                 /// 扩展结构

} *IPSDK\_UserInfo*;

typedef *struct* *tag\_IPSDK\_SearchDevice*

{

*IPSDK\_Char* strDeviceId[64];     /// 自动获取填充设备对像ID,用来标识设备端设备

*EN\_IPSDK\_DeviceMode* nDeviceMode;    /// 设备（灯光）类型

*IPSDK\_DeviceInfo* stDeviceInfo;  /// 设备信息

*IPSDK\_LanInfo* stLanInfo;

*IPSDK\_UserInfo* stUserInfo;

*IPSDK\_UInt64* enLicenseMask;     /// 授权掩码，是EN\_IPSDK\_LicenseMask的组合

*EN\_IPSDK\_ProductCommand* enProductCmd;   /// 产测命令

} *IPSDK\_SearchDevice*;

#### IPSDK\_ChannelInfo

##### 说明

通道描述信息

##### 定义

typedef *struct* *tag\_ipsdk\_channel\_info\_*

{

*IPSDK\_Int* nChannel;     /// 设备通道

*IPSDK\_Int* nStream;      /// 主码流[0],从码流[1]

} *IPSDK\_ChannelInfo*;

#### IPSDK\_MediaFrame

##### 说明

媒体流结构体

##### 定义

typedef *struct* *tag\_ipsdk\_media\_frame\_*

{

*EN\_DATA\_TYPE* enMediaType;   /// 媒体数据类型

*IPSDK\_UInt64* u64SysTime;    /// 媒体帧系统时间(时分秒)ms

*IPSDK\_UInt64* u64Timstmp;    /// 媒体帧时间戳

*IPSDK\_UInt32* u32DataLen;    /// 媒体裸数据大小

*IPSDK\_UInt32* u32AlarmType;  /// 判断帧是什么报警类型: EN\_IPSDK\_STORE\_TYPE

*union*

    {

*struct*

        {

*IPSDK\_UInt32* u32FrameType;  /// 帧类型: EN\_IPSDK\_FRAME\_TYPE

*IPSDK\_UInt32* u32FrameSeq;   /// ref frame counter

*IPSDK\_UInt32* u32FrameRate;

*IPSDK\_UInt32* u32Width;

*IPSDK\_UInt32* u32Height;

        }video, h264, h265;

*struct*

        {

*IPSDK\_UInt32* u32SampleRate;     /// 采样率: 4000,8000,16000,32000

*IPSDK\_UInt32* u32SampleWidth;    /// 采样位: 8位，16位

*IPSDK\_UInt32* u32SampleChannel;  /// 采样声道: 1 单声道,2 双声道

*IPSDK\_UInt32* u32Ratio; /// if g711a.u == 2.0

*IPSDK\_UInt32* res;

        }audio, pcm, g711a, g711u;

    };

*struct*

    {

        ///H265有vps[32],sps[33],pps[34],idr[19] ==> [按顺序存储]

        ///H264有        sps[07],pps[08],idr[05] ==> [按顺序存储]

*IPSDK\_Char*  vvps\_buf[128]; /// 只针对视频信息

*IPSDK\_UInt32* vvps\_len;

*IPSDK\_Char*  vpps\_buf[128];

*IPSDK\_UInt32* vpps\_len;

*IPSDK\_Char*  vsps\_buf[128];

*IPSDK\_UInt32* vsps\_len;

    }media\_info;

} *IPSDK\_MediaFrame*;

## API 应用实例

### 获取码流的流程图



### 程序实例