版本 1.1

2022-12-08

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
1.1	2022-12-08	增加 API 函数说明
1.0	2022-05-08	第一次版本

目录

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+	

分类	兼容配置	推荐配置	说明
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

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。

参数域	作用
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 IParam 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

参数	取值范围	输入/输出	描述
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

参数	取值范围	输入/输出	描述
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。

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

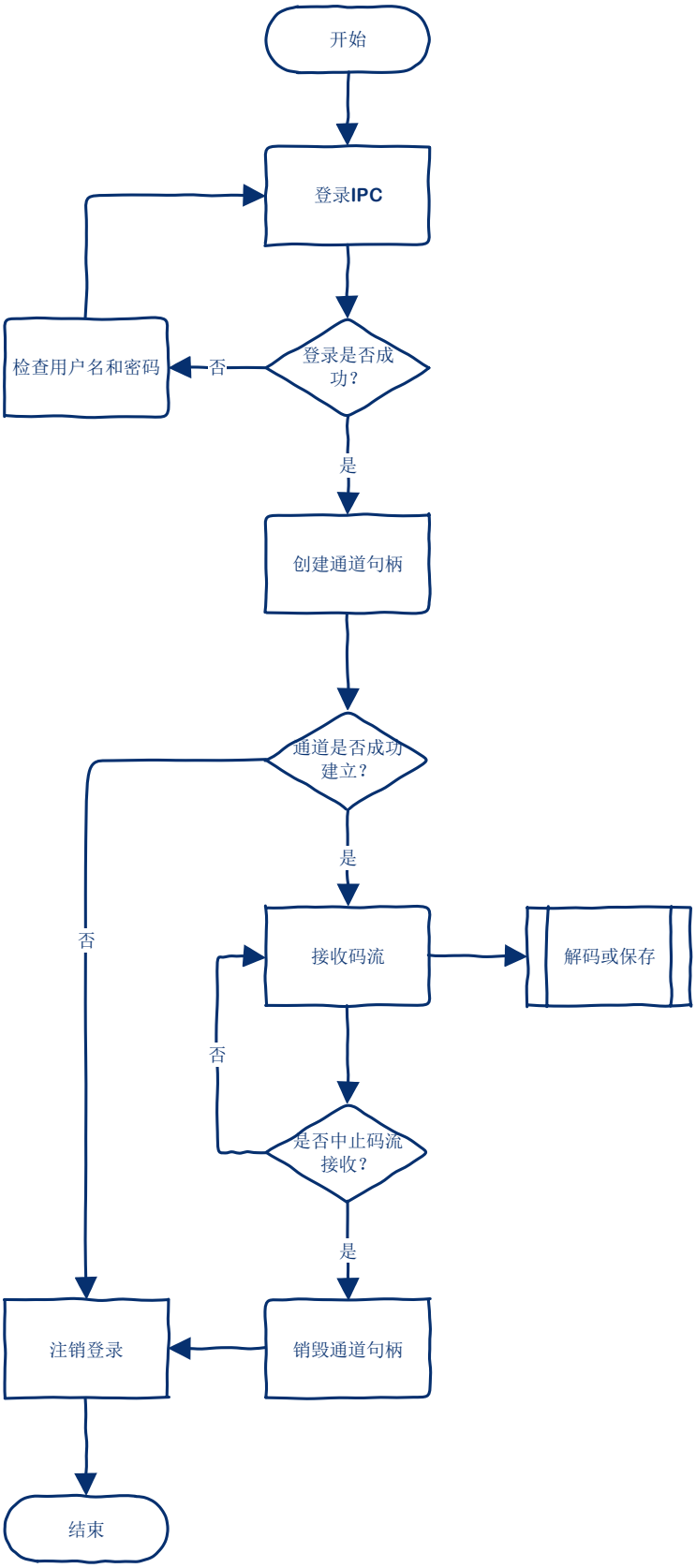
定义

```
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;
```

获取码流的流程图


开始
登录IPC
登录是否成功？
否
检查用户名和密码
是
创建通道句柄
通道是否成功建立？
否
是
接收码流
解码或保存
否
是否中止码流接收？
是
销毁通道句柄
注销登录
结束

程序实例