IPCAMERA SDK 使用说明书

Version : 1.0.5.3

版本更新说明

V1. 0. 5. 2　　2015-11-05
增加自绘功能接口
增加设置自绘回调功能接口函数 HI_SDK_SetPostDrawCallBackEx
增加错误码定义 #define HI_ERR_CALLBACK_PLAYER　　0x32003

V1. 0. 5. 2　　2015-10-09
1. 合并版本差异，1080 bmp 抓图 bug。

V1. 0. 5. 1　　2015-9-29
2. 增加抓 YUV 数据函数 HI_SDK_SnapYUVData，对应结构体 HI_YUV_INFO_S，成员 pData 指向用户预先分配的内存，成员 nDataLen 预分配内存的大小。

V1. 0. 5. 0　2015-07-10
3. HI_SDK_SetConfig 中增加设置报警时间段的命令命令 HI_CMD_QUANTUM_TIME，其中 HI_SDK_SetConfig 第一个参数为设备句柄，第二个参数为命令类型，这里可设置为 HI_CMD_QUANTUM_TIME，第三个参数为命令对应的结构体 Buff，其类型位：HI_S_QUANTUM_TIME，第四个参数为缓存 Buf 的大小。即结构体 HI_S_QUANTUM_TIME 的大小，

V1. 0. 4. 9　　2015-06-24
4. HI_SDK_SetConfig 中增加手动触发继电器的命令 HI_CMD_EXT_ALARM，其对应的结构体为：HI_S_ExtAlarm 具体用法可参考 SDK Demo 中报警功能下的外置报警抓拍

v1.0.4.8　2015-03-07
1. 云台透传中透传的最大数据长度由 64 修改为 128
2. 修改 HI_SDK_SetConfig，设置继电器开启关闭，其命令为：HI_CMD_RELAYCTRL，对应的结构体为：HI_S_RelayCtrl

v1.0.2.7　2013-09-27
1、　添加新设备类型，字段 Se、Sf，详情请查阅《厂家代码和设备类型定义》。

v1.0.2.7　2013-07-15
1、　修正不请求码流情况下修改网络参数失败 bug

v1.0.2.6　2013-04-01
1、　添加新设备类型，字段 Sc，详情请查阅《厂家代码和设备类型定义》。
2、　Sc 设备有两套分辨，第一套 960P\VGA\QVGA，第二套 720P\Q720\QQ720，用户可以根据实际应用选择需要的分辨率。

V1. 0. 3. 4　2013-02-05

1、 修改 HI_SDK_SetConfig 设置 OSD 参数 HI_CMD_OSD_PARAM 中文，linux 下，设备类型如果为 C5，中文字符必须转换成 UTF-8。

V1.0.3.3 2012-12-10

1、 添加三码流接口 HI_SDK_RealPlayExt。三码流需要设备支持；
2、 添加 D3D 显示支持，默认显示为 DDraw
3、 添加网络抓拍接口 HI_SDK_SnapJpeg
4、 添加设置显示获取模式接口 HI_SDK_GetDisplayMode 和 HI_SDK_SetDisplayMode
5、 添加 D3D 模式下图像叠加开关 HI_SDK_SetDisplayCallback
6、 添加音频输入输出音量接口：参阅 HI_SDK_SetConfig 和 HI_SDK_GetConfig

```
#define HI_CMD_AUDIO_VOLUME_IN          0x1070 //音频输入音量
#define HI_CMD_AUDIO_VOLUME_OUT         0x1071 //音频输出音量
```

7、 添加 WIFI 控制接口：参阅 HI_SDK_SetConfig 和 HI_SDK_GetConfig

```
#define HI_CMD_VIDEO_PARAM_EXT          0x1047  //视频参数
#define HI_CMD_AUDIO_PARAM_EXT          0x1048  //音频参数
#define HI_CMD_RESOLUTION_EXT           0x1049  //分辨率
```

V1.0.3.2 2012-10-22

1、 添加 WIFI 控制接口：参阅 HI_SDK_SetConfig 和 HI_SDK_GetConfig

```
#define HI_CMD_WIFI_PARAM       0x1030  //WIFI 参数
#define HI_CMD_WIFI_SEARCH      0x1031  //WIFI 搜索
#define HI_CMD_WIFI_CHECK       0x1035  //WIFI check
```

v1.0.3.1 2012-05-29

1、 增加动态 I 帧接口 HI_SDK_MakeKeyFrame；

v1.0.3.0 2012-03-29

1、 修改字段为 S7、S9 的设备的默认值；
2、 增加 HI_SDK_LoginExt 登陆接口，接口带有超时时间；
3、 增加 HI_SDK_SetChannel 和 HI_SDK_GetChannel，用于设置 NVR 通道；
4、 增加 NVR 参数设置

```
#define HI_NVR_CMD_NET_EXT          0x1050  // NVR 网络参数
#define HI_NVR_CMD_RTSP_INFO        0x1051  // NVR rtsp 参数
#define HI_NVR_CMD_USER             0x1052  // NVR 用户参数
#define HI_NVR_CMD_CHANNEL_INFO     0x1053  // NVR 通道参数
#define HI_NVR_CMD_SEARCH           0x1055  // NVR 搜索设备
#define HI_NVR_CMD_RECORD_INFO      0x1056  // NVR 通道录像参数
#define HI_NVR_CMD_RECORD_SYS       0x1057  // NVR 系统参数
#define HI_NVR_CMD_TIME             0x1058  // NVR 时间参数
#define HI_NVR_CMD_RESET            0x1059  // NVR 恢复出厂设置
#define HI_NVR_CMD_REBOOT           0x1060  // NVR 重启
#define HI_NVR_CMD_RECORD_STATE     0x1061  // 获取录像状态
#define HI_NVR_CMD_DISK_INFO        0x1062  // 获取硬盘信息
#define HI_NVR_CMD_DISK_FORMAT      0x1063  // 格式化硬盘
```

#define HI_NVR_CMD_RECORD_STATE_EX 0x1064 // 获取录像状态

具体参阅 HI_SDK_SetConfig 和 HI_SDK_GetConfig

v1.0.2.9 2011-12-16

1、 添加新设备类型，字段 S9，详情请查阅《厂家代码和设备类型定义》。

v1.0.2.8 2011-11-22

1、 添加心跳包处理，心跳包从 HI_SDK_SetMessageCallBack 回调出来，详细回到请查阅 HI_SDK_SetMessageCallBack。

v1.0.2.7 2011-11-01

3、 添加新设备类型，字段 S8，详情请查阅《厂家代码和设备类型定义》。

v1.0.2.6 2011-09-20

1、 添加新设备类型，字段 S7，详情请查阅《厂家代码和设备类型定义》。

v1.0.2.5 2011-09-06

2、 修改回调 HI_SDK_SetRealDataCallBack，当调用该回调函数时同时能播放音视频。

v1.0.2.4 2011-07-22

1、 网络库新添加接口 HI_NET_DEV_StartRecord 、 HI_NET_DEV_StopRecord 和 HI_NET_DEV_GetRecordState 录像操作接口，取消播放库里的录像接口 HI_PLAYER_StartRecord、HI_PLAYER_StopRecord。更新说明请参阅网络库和播放库使用说明书。

v1.0.2.3 2011-07-02

2、 添加新设备类型，字段 S5、S6，详情请查阅《厂家代码和设备类型定义》。

3、 添加录像类型 AVI，接口 HI_SDK_StartRecord 中第三参数指定为：FILE_FORMAT_AVI，HI_SDK_Playback 支持 AVI 文件播放。

4、 添加控制输入报警开关接口：参阅 HI_SDK_SetConfig 和 HI_SDK_GetConfig 的 HI_CMD_ATTR_EXT 选项。

v1.0.2.2 2011-06-08

1、 HI_SDK_PTZControl 和 HI_SDK_PTZControlEx 添加焦点调整和光圈变化命令

```
#define HI_CTRL_PTZ_FOCUSIN          0x3007  //焦点前调
#define HI_CTRL_PTZ_FOCUSOUT         0x3008  //焦点后调
#define HI_CTRL_PTZ_APERTUREIN       0x3009  //光圈放大
#define HI_CTRL_PTZ_APERTUREOUT      0x3010  //光圈变小
```

2、 添加显示区域电子放大接口：HI_SDK_DisplayAll

3、 添加网络参数接口，参阅 HI_SDK_SetConfig 和 HI_SDK_GetConfig 的 HI_CMD_NET_EXT 选项。改指令将 HI_CMD_NET_INFO 和 HI_CMD_HTTP_PORT 合并。

v1.0.2.0 2010-03-10

1、 添加设备重启接口，参阅 HI_SDK_SetConfig 的 HI_CMD_REBOOT 选项。
2、 添加设备恢复出厂设置接口，参阅 HI_SDK_SetConfig 的 HI_CMD_RESET 选项。
3、 添加校时接口，参阅 HI_SDK_SetConfig 和 HI_SDK_SetConfig 的 HI_CMD_SERVER_TIME 选项。

v1.0.1.9 2010-02-21
1、 修改 YUV 回调函数 HI_SDK_SetDecCallBack 接口，调用回调函数视频只回调出 YUV 数据，而不显示在窗口中；
2、 添加 HI_SDK_SetDrawWnd 接口，接口用于设置显示窗口，当 pWnd 等于 NULL 时，窗口的 DDRAW 将销毁；设置为窗口句柄时，DDRAW 将重现初始化并显示图像。
3、 增加 HI_SDK_GetMediaAttr 接口，获取设置播放音视频属性参数。

v1.0.1.8 2010-12-31
1、 修改录像接口，可以添加录像时间长度；
2、 合并录像库接口，ASF 与复合流录像公用一个接口，参数配置不同。

v1.0.1.5 2010-12-4
1、 添加云台原点和上下左右巡航接口，目前仅支持设备信息中有 Z0 字段的设备。

v1.0.1.4 2010-12-1
1、添加捕获实时数据接口 HI_SDK_SaveRealData 和 HI_SDK_StopSaveRealData，保存为自定义格式录像
2、 添加解码控制接口 HI_SDK_PauseDecode 和 HI_SDK_ResumeDecode。

内容简介

SDK 主要功能

预览、设置参数、报警、云台控制、录像、回放、语音对讲、抓拍、声音控制等功能。

SDK 库文件说明

SDK 库	hi_sdk_api.h	头文件
	HISDK.lib	LIB 库文件
	HISDK.dll	DLL 库文件
网络库	hi_net_dev_sdk.h	头文件
	NetLib.lib	LIB 库文件
	NetLib.dll	DLL 库文件
播放库	HsPlayer.h	头文件
	HIPlayer.lib	LIB 库文件
	HIPlayer.dll	DLL 库文件
搜索库	hi_vscp_devs_cli.h	头文件
	SearchLib.lib	LIB 库文件
	SearchLib.dll	DLL 库文件
公用文件	hi_dataType.h	头文件

客户端 SDK 开发包中包含以上各个组件，用户可以根据需要选择其中的一部分组件，以下将对各个组件在 SDK 中的作用和使用条件分别说明。

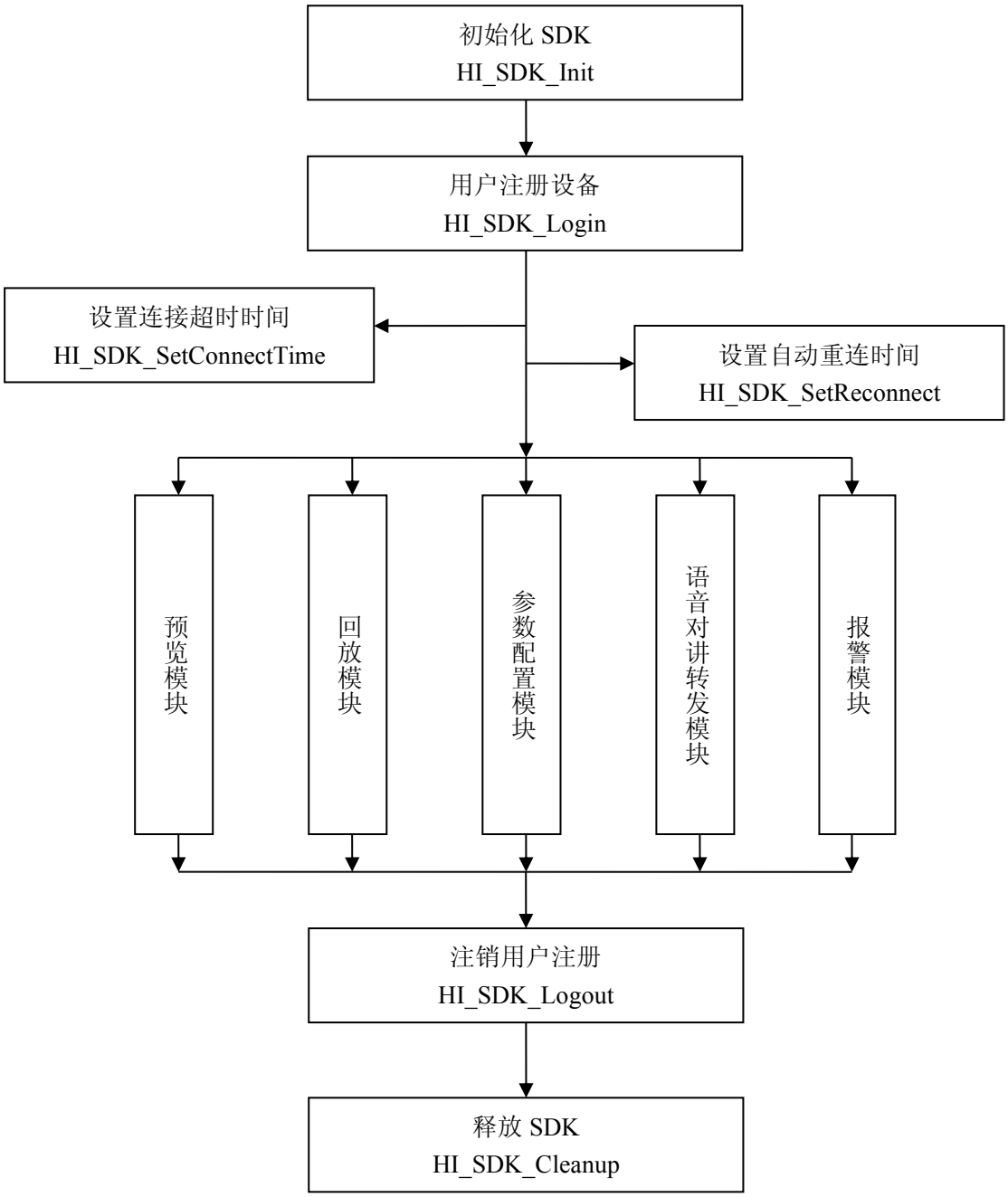
- SDK 库：将 HIPlayer.dll 和 NetLib.dll 封装，用于非平台开发，具体接口查阅本文档第一部分《SDK 使用说明》。
- 播放库：用于播放数据流与文件，用于平台开发，使用请查阅《网络库使用说明书说明书》PDF 文档。
- 网络库：用于平台开发。使用请查阅《网络库使用说明书说明书》PDF 文档。
- 搜索库：接口说明请查阅本文档第三部分《搜索 SDK 说明》。

平台开发：用网络库开发转发平台，播放库处理数据以及显示，播放库还用于播放录像文件；

客户端开发：用 SDK 库，集合了网络库和播放库的功能。

编程导引

SDK 接口调用主要流程



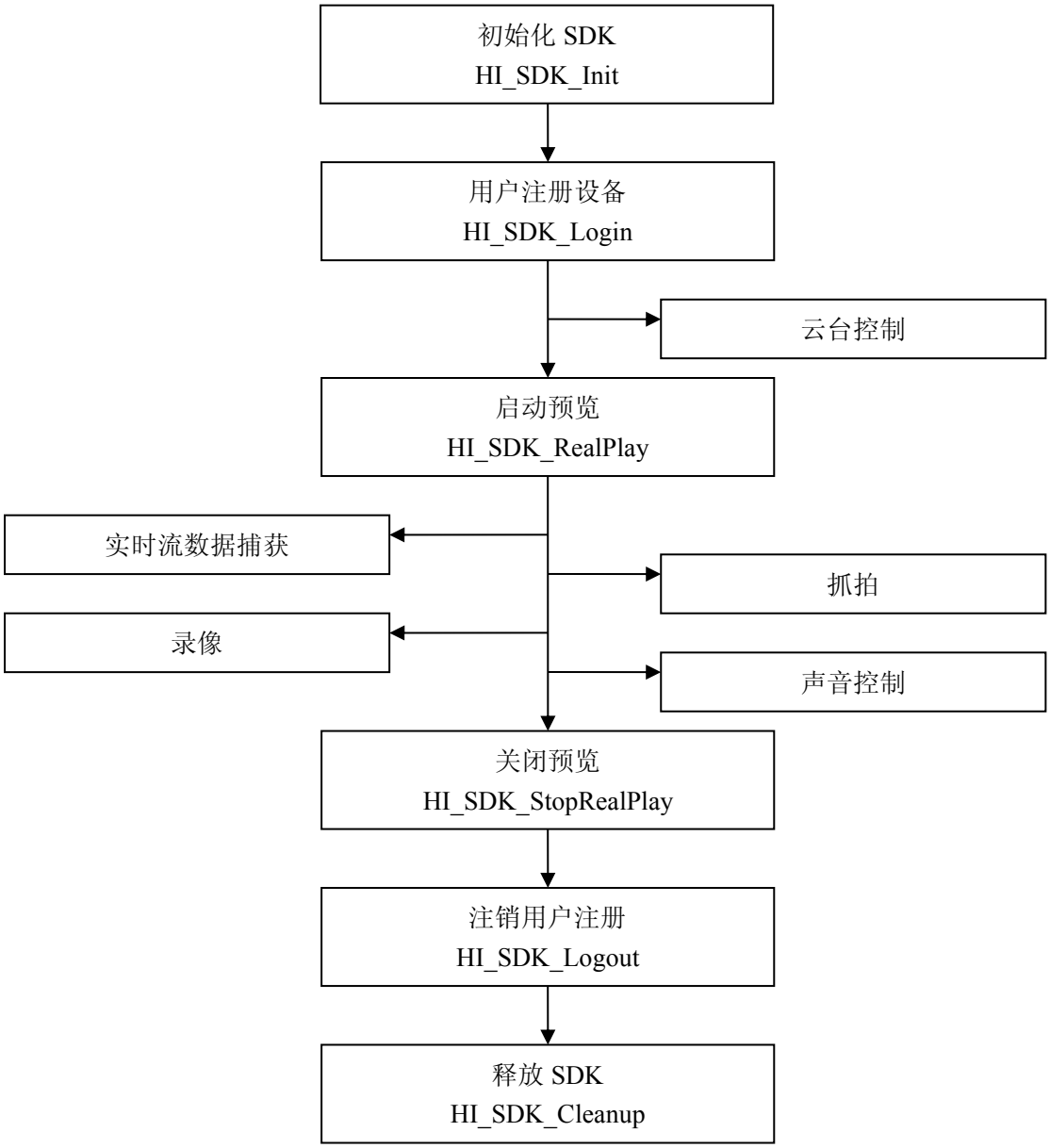
按实现功能的不同可以分成五个模块，实现每个模块的功能时初始化 SDK、用户注册设备、注销设备和释放 SDK 资源这 4 个流程是必不可少的。

- 初始化 SDK（HI_SDK_Init 接口）：对整个网络 SDK 系统的初始化。
- 设置连接超时时间（HI_SDK_SetConnectTime 接口）：这部分为可选，用于设置 SDK 中的网络连接超时时间，用户可以根据自己的需要设置该值。在不调用此接口设置超时时间的情况下，将采用 SDK 中的默认值。
- 用户注册设备（HI_SDK_Login 接口）：实现用户的注册功能，注册成功后，返回的用户 ID 作为其他功能操作的唯一标识。
- 预览模块：启动预览后可以捕获实时数据、抓拍、录像、控制音频等。云台控制无

需启动预览。具体流程详见预览模块流程。

- 回放模块：回放功能目前仅支持本地文件回放。具体流程详见回放模块流程。
- 参数配置模块：设置和获取前端摄像机的参数，主要包括设备参数、网络参数、报警参数、异常参数、用户配置等参数信息。具体流程详见参数配置模块流程。
- 语音对讲转发模块：实现和前端服务器的语音数据对讲和语音数据获取，音频编码格式可以指定。具体流程详见语音对讲转发模块流程。
- 报警模块：处理前端服务器上传的各种报警信号。报警分为“移动报警”和“输入报警”两种数据。具体流程详见报警模块流程。

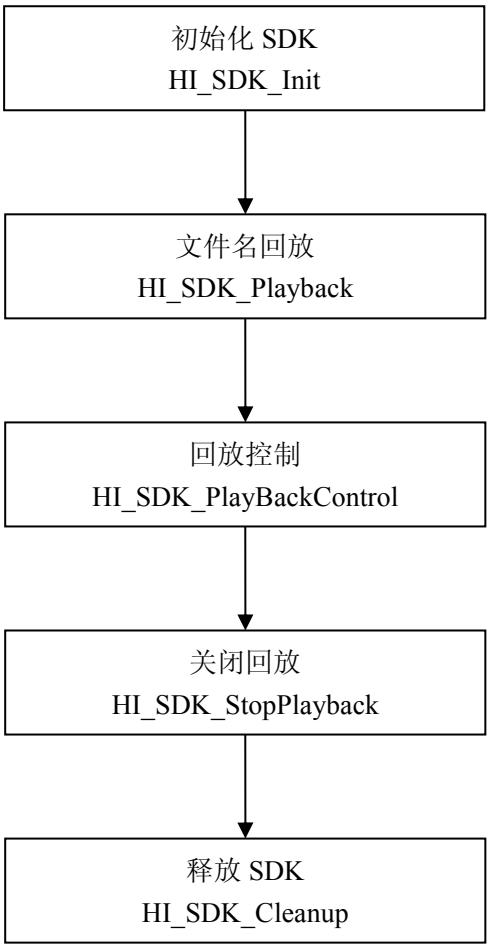
预览模块流程



启动预览后可以捕获实时数据、抓拍、录像、控制音频等。云台控制无需启动预览。

- 捕获实时数据：捕获实时数据使用回调函数 HI_SDK_SetRealDataCallBack，数据包含数据头；
- 录像：录像分为两种录像方式，一种是路成 ASF 文件格式，另一种则可以录成自定义格式。关于录像的格式以及函数说明请参阅录像接口（HI_SDK_StartRecord）说明。
- 抓拍：抓拍有两种格式，JPG 和 BMP，BMP 格式使用接口 HI_SDK_CapturePicture，JPG 格式使用接口 HI_SDK_CaptureJPEGPicture。
- 声音控制：声音控制功能主要实现声音的打开和关闭、音量的控制。相关接口有：HI_SDK_SetVolume、HI_SDK_GetVolume、HI_SDK_SetMute、HI_SDK_GetMute 等。
- 云台控制：云台控制无需预览，只要注册成功就能控制云台，相关接口有：HI_SDK_PTZControl、HI_SDK_PTZControlEx、HI_SDK_PTZPreset 等。

回放模块流程



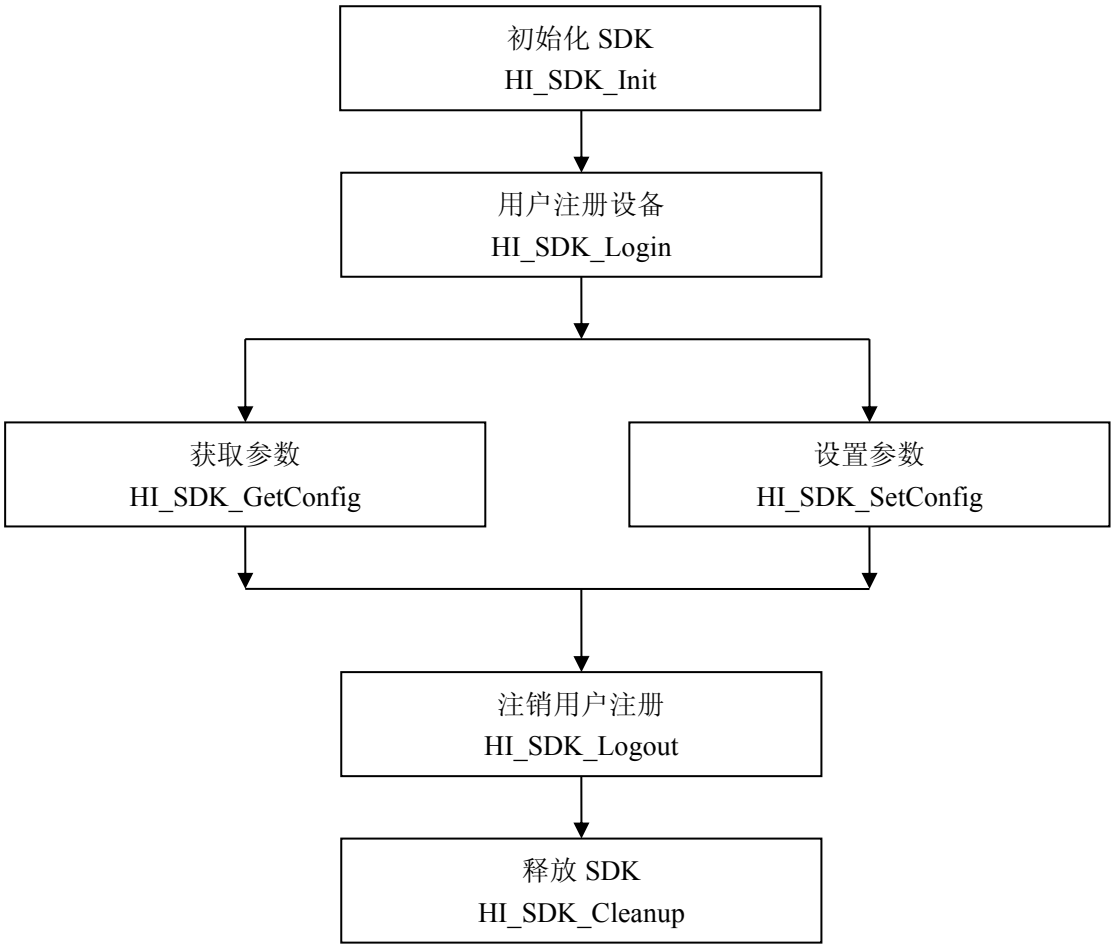
回放功能目前仅支持本地文件回放，相关接口有：HI_SDK_Playback、HI_SDK_PlayBackControl、HI_SDK_StopPlayback。

其中，回放控制有如下功能：

- 播放：开始播放文件；
- 停止：停止文件播放，指针返回文件头；
- 暂停：暂停播放；
- 调整播放速度：调整速度；
- 单帧：单帧播放；
- 获取/设置播放位置：跳转；
- 设置播放音量：音量设置；
- 获取文件总时间和播放时间：文件的总时间和当前播放的时间。

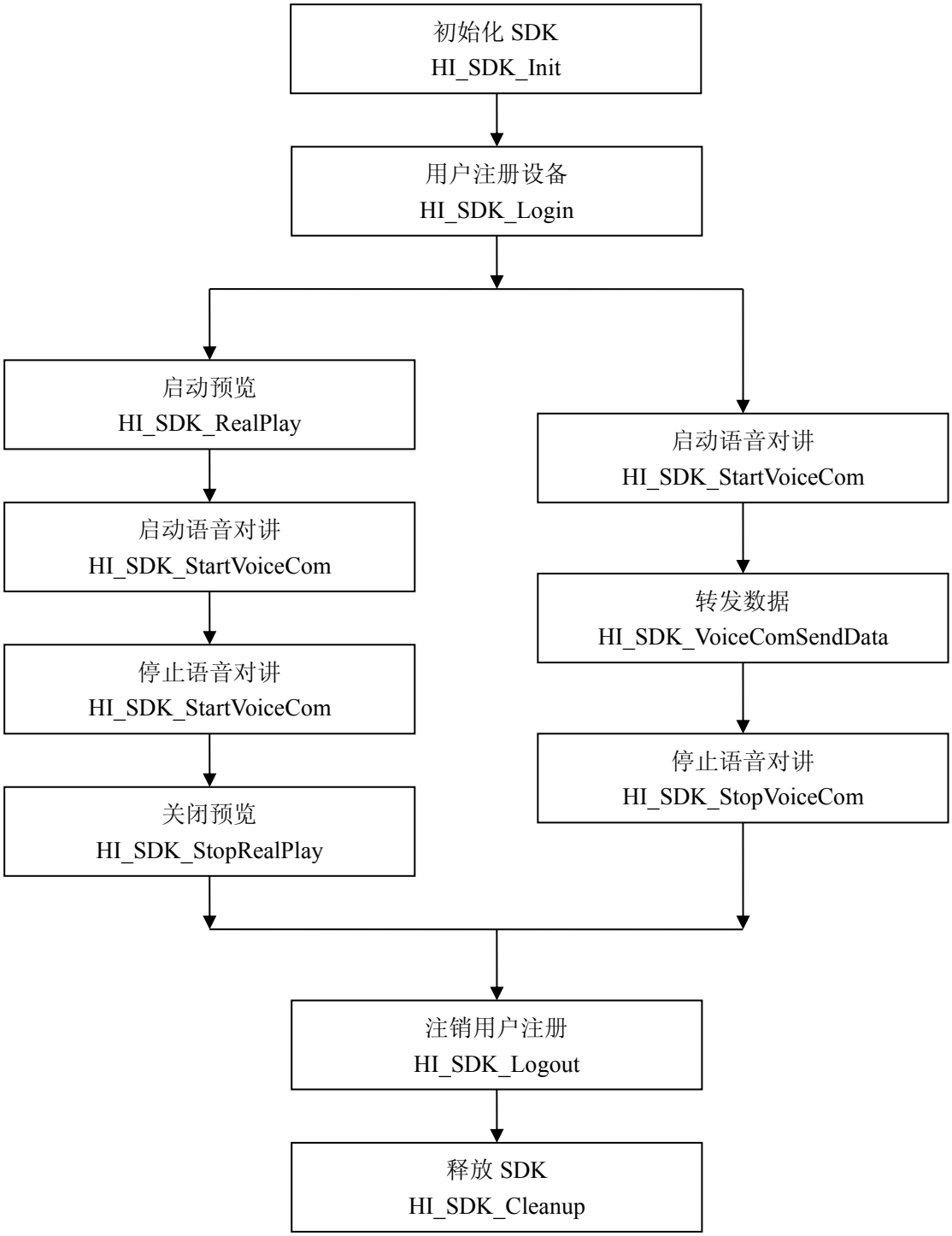
详细使用方法请查阅 HI_SDK_PlayBackControl 接口使用。

参数配置模块流程



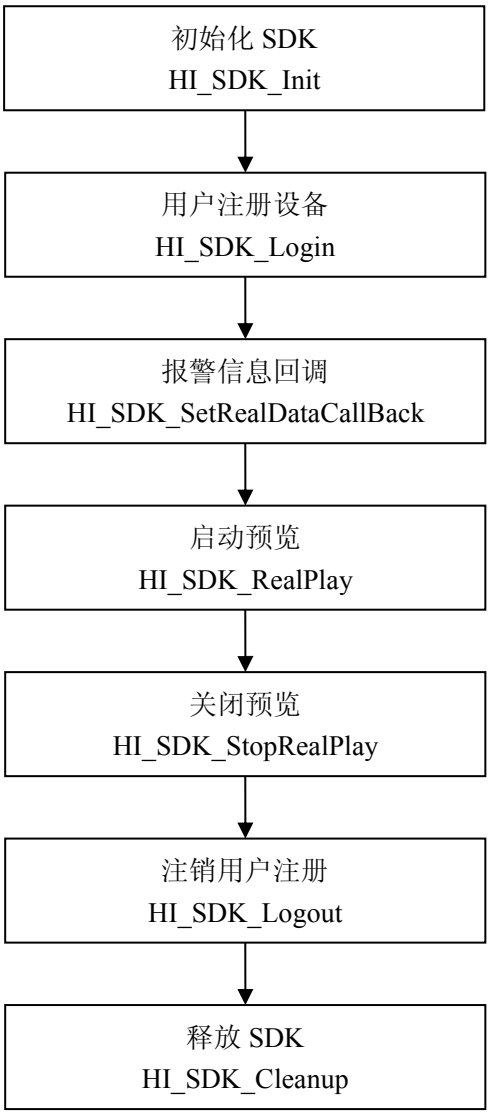
实现参数配置首先必须做好初始化 SDK 和用户注册这两个步骤，将用户注册接口返回的句柄作为配置接口的首个参数。建议在每次设置某类参数之前，先调用获取参数的接口（HI_SDK_GetConfig）得到完整的参数结构，修改需要更改的参数，作为设置参数接口中的输入参数，最后调用设置参数接口（HI_SDK_SetConfig），返回成功即设置成功。

语音对讲转发模块流程



语音对讲有两种方式，一种是直接从 PC 端获取音频数据（数据由播放库获取，已编码），通过 SDK 的内部自己将数据发送给摄像机端；另一种将自己准备好的音频数据发送到摄像机，音频数据必须编码成与摄像机端当前的音频编码的格式一致。

报警模块流程



报警回调可分为“移动报警”和“输入报警”两种数据。

- 移动报警：当检测到镜头相应区域有移动，回调函数将移动区域的相关数据输出；
- 输入报警：摄像机参数有数据变更时有输入报警信息。

具体使用请参阅报警回调函数 HI_SDK_SetMessageCallBack 的使用方法。

数据类型定义说明

```
typedef unsigned char           HI_U8;
typedef unsigned char           HI_UCHAR;
typedef unsigned short          HI_U16;
typedef unsigned int            HI_U32;

typedef signed char             HI_S8;
typedef short                   HI_S16;
typedef int                     HI_S32;

#ifndef _M_IX86
typedef unsigned long long      HI_U64;
typedef long long               HI_S64;
#else
typedef __int64                 HI_U64;
typedef __int64                 HI_S64;
#endif

typedef char                    HI_CHAR;
typedef char*                   HI_PCHAR;

typedef float                   HI_FLOAT;
typedef double                  HI_DOUBLE;
typedef void                    HI_VOID;

typedef unsigned long           HI_SIZE_T;
typedef unsigned long           HI_LENGTH_T;


typedef enum {
    HI_FALSE        = 0,
    HI_TRUE         = 1,
} HI_BOOL;

#ifndef NULL
#define NULL                    0L
#endif
#define HI_NULL                 0L
#define HI_NULL_PTR             0L

#define HI_SUCCESS              0
#define HI_FAILURE              (-1)
```

错误定义说明

```
#define  HI_ERR_SDK_HANDLE              0x30001    //操作句柄错误
#define  HI_ERR_PLAYER_NULLPTR          0x30002    //播放句柄错误
#define  HI_ERR_DRAW_NULLPTR            0x30003    //画图句柄错误
#define  HI_ERR_CMD_NULLPTR             0x30004    //参数为空
#define  HI_ERR_CMD_INVALID_ARG         0x30005    //参数格式错误
#define  HI_ERR_CMD_DISCONNECT          0x30006    //连接状态为非连接
#define  HI_ERR_PLAYER_WNDHWND          0x30008    //显示句柄错误
#define  HI_ERR_STATE_IS_PLAYING        0x30009    //播放状态
#define  HI_ERR_STATE_IS_STOP           0x30010    //停止状态
#define  HI_ERR_PLAYER_STOP             0x30011    //停止播放失败
#define  HI_ERR_PLAYER_DEC              0x30012    //解码失败
#define  HI_ERR_PLAYER_SNAP             0x30013    //抓图失败
#define  HI_ERR_PLAYER_PLAY             0x30014    //播放失败
#define  HI_ERR_PLAYER_STOP_TALK        0x30015    //停止对讲失败
#define  HI_ERR_PLAYER_START_TALK       0x30016    //开始对讲失败
#define  HI_ERR_PLAYER_PAUSE            0x30017    //暂停失败
#define  HI_ERR_PLAYER_SETRATE          0x30018    //设置播放速度失败
#define  HI_ERR_PLAYER_ONEBYONE         0x30019    //单帧播放失败
#define  HI_ERR_PLAYER_SETPOS           0x30020    //设置播放位置失败
#define  HI_ERR_PLAYER_GETPOS           0x30021    //获取播放位置失败
#define  HI_ERR_PLAYER_SETMUTE          0x30022    //设置静音失败
#define  HI_ERR_PLAYER_GETMUTE          0x30023    //获取静音失败
#define  HI_ERR_PLAYER_SETVOLUME        0x30024    //设置音量失败
#define  HI_ERR_PLAYER_GETVOLUME        0x30025    //获取音量失败
#define  HI_ERR_PLAYER_MEDIA_ATTR       0x30026    //设置播放属性失败
#define  HI_ERR_CALLBACK_DRAW           0x32001    //画图回调注册失败
#define  HI_ERR_CALLBACK_STATE          0x32002    //状态回调组成失败
#define  HI_ERR_REC_RECORDING           0x30050    //录像状态
#define  HI_ERR_REC_START_FAIL          0x30051    //开始录像失败
#define  HI_ERR_REC_STOP_FAIL           0x30052    //关闭录像失败
#define  HI_ERR_TALK_STARTING           0x30081    //对讲状态
#define  HI_ERR_TALK_NOSTARTING         0x30082    //对讲关闭状态
#define  HI_ERR_TALK_START_FAIL         0x30083    //打开对讲失败
#define  HI_ERR_TALK_SEND_FAIL          0x30084    //对讲传输失败
#define  HI_ERR_TALK_STOP_FAIL          0x30085    //停止对讲失败
#define  HI_ERR_PLAYER_OPENFILE         0x30100    //打开文件失败
#define  HI_ERR_PLAYER_CLOSEFILE        0x30100    //关闭文件失败
#define  HI_ERR_NET_PLAY                0x31001    //开始网络传输失败
#define  HI_ERR_NET_STOP                0x31002    //关闭网路传输失败
#define  HI_ERR_ATTR_NOSUPPORT          0x31003    //不支持属性
```

第一部分 SDK 使用说明

1.1 初始化 SDK

HI_SDK_Init

初始化，使用下面 SDK 接口前使用，仅在初始化 SDK 使用

```
HI_S32    HI_SDK_Init (
);
```

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

HI_SDK_Cleanup

释放 SDK，该函数放在最后释放

```
HI_S32    HI_SDK_Cleanup (
);
```

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

Remarks

HI_SDK_Init、HI_SDK_Cleanup 在一个程序中仅初始化一次，初始化 Socket

1.2 用户注册

HI_SDK_Login

用户设备注册，返回句柄供用户对设备的操作

```
HI_HANDLE    HI_SDK_Login (
        const HI_CHAR*    psHost,
        const HI_CHAR*    psUsername,
        const HI_CHAR*    psPassword,
        HI_U16            u16Port,
        HI_S32 *          ps32Err
);
```

Parameters

psHost

[IN] 主机，可以是 IP 地址也可以是域名

psUsername

[IN] 用户名

psPassword

[IN] 密码

u16Port

[IN] 端口号

ps32Err

[OUT] 输出错误信息

Return Values

成功将返回 HI_HANDLE 句柄，失败返回 0

HI_SDK_LoginExt

用户设备注册扩展接口，带超时输入，返回句柄供用户对设备的操作

```
HI_HANDLE    HI_SDK_LoginExt (
        const HI_CHAR*   psHost,
        const HI_CHAR*   psUsername,
        const HI_CHAR*   psPassword,
        HI_U16           u16Port,
        HI_U32           u32TimeOut,
        HI_S32 *         ps32Err
);
```

Parameters

psHost

[IN] 主机，可以是 IP 地址也可以是域名

psUsername

[IN] 用户名

psPassword

[IN] 密码

u16Port

[IN] 端口号

u32TimeOut

[IN] 超时时间，单位毫秒，默认 10000 毫秒

ps32Err

[OUT] 输出错误信息

Return Values

成功将返回 HI_HANDLE 句柄，失败返回 0

HI_SDK_Logout

用户取消登录

```
HI_S32    HI_SDK_Logout t(
        HI_HANDLE       lHandle
);
```

Parameters

lHandle

[IN] 操作句柄

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_SetConnectTime

设置连接超时时间，默认超时是 5 秒，单位是毫秒

```
HI_S32    HI_SDK_SetConnectTimeout (
    HI_HANDLE       lHandle,t
    HI_U32          u32Timeout
);
```

Parameters

lHandle

[IN] 操作句柄

u32Timeout

[IN] 超时时间，单位是毫秒

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_SetReconnect

设置自动重连间隔时间，默认为 10 秒，0 为不重连，单位是毫秒

```
HI_S32    HI_SDK_SetReconnect (
    HI_HANDLE       lHandle,
    HI_U32          u32Interval
);
```

Parameters

lHandle

[IN] 操作句柄

u32Interval

[IN] 自动重连间隔时间，单位是毫秒，默认时间是 10 秒

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

1.3 实时预览

HI_SDK_RealPlay

实时数据

```
HI_S32    HI_SDK_RealPlay (
```

```
    HI_HANDLE           lHandle,
    HI_VOID*            pWnd,
    HI_S_STREAM_INFO*   pstruStreamInfo
);
```

Parameters

lHandle

[IN] 操作句柄

PWnd

[IN] 显示窗口句柄

pstruStreamInfo

[IN] 操作参数

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remarks

```
// 开始流传输
typedef struct
{
    HI_U32      u32Channel;         //通道号，设置获取属性相对应
    HI_BOOL     blFlag;             //为真连接主码流，假连接次码流
    HI_U32      u32Mode;            //网络连接模式
    HI_U8       u8Type;             //流数据类型，视频，音频，其他数据
} HI_S_STREAM_INFO;

// 设备 通道号，目前仅支持一个通道
#define HI_CHANNEL_1      1
//#define HI_CHANNEL_2      2
//#define HI_CHANNEL_3      3
//#define HI_CHANNEL_4      4

// 连接网络连接模式，目前仅支持 TCP
#define HI_STREAM_MODE_TCP   0

// 流数据类型，目前不支持心跳数据
// 次码流不支持报警数据和心跳数据
#define HI_STREAM_VIDEO_ONLY      0x01
#define HI_STREAM_AUDIO_ONLY      0x02
#define HI_STREAM_VIDEO_AUDIO     0x03
#define HI_STREAM_VIDEO_DATA      0x05
#define HI_STREAM_AUDIO_DATA      0x06
#define HI_STREAM_ALL             0x07
```

若需要捕获实时码流数据，可以调用接口 HI_SDK_SetRealDataCallBack 或

HI_SDK_SetDecCallBack 注册捕获码流数据的回调函数，并在回调函数中自行处理回调上来的码流数据。

HI_SDK_RealPlayExt

实时数据

```
HI_S32   HI_SDK_RealPlayExt (
    HI_HANDLE                   lHandle,
    HI_VOID*                    pWnd,
    HI_S_STREAM_INFO_EXT*       pstruStreamInfo
);
```

Parameters

u32Handle

[IN] 操作句柄

pstruStreamInfo

[IN] 操作参数

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

Remarks

```
// 开始流传输
typedef struct
{
    HI_U32      u32Channel;         //通道号，设置获取属性相对应
    HI_U32      u32Stream;          //1:连接主码流，2:连接次码流  3:第三码流
    HI_U32      u32Mode;            //网络连接模式
    HI_U8       u8Type;             //流数据类型，视频，音频，其他数据
} HI_S_STREAM_INFO_EXT;

u32Stream 参数：
#define HI_STREAM_1     0
#define HI_STREAM_2     1
#define HI_STREAM_3     2

// 设备通道号，摄像机仅支持一个通道，转发支持多通道
#define HI_CHANNEL_1     1

// 连接网络连接模式，目前仅支持 TCP
#define HI_STREAM_MODE_TCP   0

// 流数据类型，目前不支持心跳数据
// 次码流不支持报警数据和心跳数据
#define HI_STREAM_VIDEO_ONLY      0x01
```

```
#define HI_STREAM_AUDIO_ONLY        0x02
#define HI_STREAM_VIDEO_AUDIO       0x03
#define HI_STREAM_VIDEO_DATA        0x05
#define HI_STREAM_AUDIO_DATA        0x06
#define HI_STREAM_ALL               0x07
```

HI_SDK_StopRealPlay

停止数据流

```
HI_S32   HI_SDK_StopRealPlay (
    HI_HANDLE       lHandle
);
```

Parameters

lHandle

[IN] 操作句柄

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_MakeKeyFrame

动态创建一个 I 帧

```
HI_S32   HI_SDK_MakeKeyFrame (
    HI_HANDLE       lHandle,
    HI_U32          u32Channel
);
```

Parameters

lHandle

[IN] 操作句柄

u32Channel

[IN] 通道，11 表示主码流，12 表示次码流

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

1.4 摄像机属性设置

摄像机是否支持属性，可以通过获取 HI_GET_PRODUCT_VENDOR 中 sProduct 判断，具体请参阅附录厂家代码和设备类型定义章节。

HI_SDK_SetConfig

设置摄像机参数

```
HI_S32   HI_SDK_SetConfig (
    HI_U32          u32Handle
```

```
    HI_U32          u32Command,
    HI_VOID*        pBuf,
    HI_U32          u32BufLen
);
```

Parameters

u32Handle

[IN] 操作句柄

u32Command

[IN] 操作参数命令

宏定义	宏定义值	含义
HI_CMD_DISPLAY	0x1001	图像参数
HI_CMD_DISPLAY_EXT	0x1002	翻转
HI_CMD_INFRARED	0x1003	红外
HI_CMD_VIDEO_PARAM	0x1004	视频参数
HI_CMD_OSD_PARAM	0x1005	OSD 参数
HI_CMD_AUDIO_PARAM	0x1006	音频参数
HI_CMD_AUDIO_INPUT	0x1007	音频输入
HI_CMD_RESOLUTION	0x1008	图像分辨率
HI_CMD_FREQUENCY	0x1009	频率
HI_CMD_PTZ_PARAM	0x1010	云台信息
HI_CMD_MD_PARAM	0x1011	移动报警信息
HI_CMD_NET_INFO	0x1012	网络信息
HI_CMD_HTTP_PORT	0x1013	网页端口号
HI_CMD_SERVER_TIME	0x1017	设置摄像机时间
HI_CMD_REBOOT	0x1018	重启
HI_CMD_RESET	0x1019	恢复出厂设置
HI_CMD_NET_EXT	0x1022	设置网络参数
HI_CMD_ATTR_EXT	0x1026	控制输入报警
HI_NVR_CMD_NET_EXT	0x1050	NVR 网络参数
HI_NVR_CMD_RTSP_INFO	0x1051	NVR rtsp 参数
HI_NVR_CMD_USER	0x1052	NVR 用户参数
HI_NVR_CMD_CHANNEL_INFO	0x1053	NVR 通道参数
HI_NVR_CMD_RECORD_INFO	0x1056	NVR 通道录像参数
HI_NVR_CMD_RECORD_SYS	0x1057	NVR 录像系统参数
HI_NVR_CMD_TIME	0x1058	NVR 时间设置
HI_NVR_CMD_RESET	0x1059	NVR 恢复出厂设置
HI_NVR_CMD_REBOOT	0x1060	NVR 重启
HI_CMD_WIFI_PARAM	0x1030	WIFI 参数设置
HI_CMD_WIFI_SEARCH	0x1031	WIFI 搜索
HI_CMD_WIFI_CHECK	0x1035	WIFI check

HI_CMD_VIDEO_PARAM_EXT	0x1047	视频参数（扩展）
HI_CMD_AUDIO_PARAM_EXT	0x1048	音频参数（扩展）
HI_CMD_RESOLUTION_EXT	0x1049	分辨率参数（扩展）
HI_CMD_AUDIO_VOLUME_IN	0x1070	音频输入音量
HI_CMD_AUDIO_VOLUME_OUT	0x1071	音频输出音量
HI_CMD_QUANTUM_TIME	0x1072	报警时间段
HI_CMD_RELAYCTRL	0x1084	继电器控制
HI_CMD_EXT_ALARM	0x1085	手动触发外置报警

pBuf

[IN] 设置数据

u32BufLen

[IN] 数据长度

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remarks

1、 **HI_CMD_DISPLAY**

```
typedef struct HI_Display
{
    HI_U32      u32Brightness;      //亮度，范围[0~255]
    HI_U32      u32Saturation;      //饱和度，范围[0~255]
    HI_U32      u32Contrast;        //对比度，范围[0~255]，高清[1~7]
    HI_U32      u32Hue;             //色度，范围[0~255]，高清无
} HI_S_Display;
```

注：u32Brightness 值等于-1，将设置为默认值。色彩支持请参阅附录厂家代码和设备类型定义的 S 字段。

Example：

```
HI_S_Display sDisplay;
// sDisplay.u32Brightness = -1;   //设置默认值
sDisplay.u32Brightness = 100;
sDisplay.u32Saturation = 100;
sDisplay.u32Contrast = 100;
sDisplay.u32Hue = 100;
HI_SDK_SetConfig (   lHandle,                  // HI_SDK_GetConfig
                     HI_CMD_DISPLAY,
                     &sDisplay,
                     sizeof(HI_S_Display));
```

2、 **HI_CMD_DISPLAY_EXT**

```
typedef struct HI_Display_Ext
{
    HI_BOOL     blFlip;             //上下翻转
```

```
    HI_BOOL     blMirror;       //左右翻转
    HI_S32      s32Scene;       //场景，自动、室内、室外
} HI_S_Display_Ext;
```

宏定义	宏定义值	含义
HI_SCENE_AUTO	0	自动
HI_SCENE_INDOOR	1	室内
HI_SCENE_OUTDOOR	2	室外

Example：

```
HI_S_Display_Ext sDisplayEx;
sDisplayEx.blFlip = HI_FALSE;
sDisplayEx.blMirror = HI_FALSE;
sDisplayEx.s32Scene = HI_SCENE_AUTO;
HI_SDK_SetConfig (  lHandle,                // HI_SDK_GetConfig
                    HI_CMD_DISPLAY_EXT,
                    &sDisplayEx,
                    sizeof(HI_S_Display_Ext));
```

注：设备支持请参阅附录厂家代码和设备类型定义的 S 字段。

3、 HI_CMD_INFRARED

```
typedef struct HI_Infrared
{
    HI_S32      s32Infrared;    //红外状态开关
} HI_S_Infrared;
```

宏定义	宏定义值	含义
HI_INFRARED_AUTO	0	自动
HI_INFRARED_ON	1	开
HI_INFRARED_OFF	2	关

Example：

```
HI_S_Infrared sInfrared;
sInfrared.s32Infrared = HI_INFRARED_AUTO;
HI_SDK_SetConfig (  lHandle,                // HI_SDK_GetConfig
                    HI_CMD_INFRARED,
                    &sInfrared,
                    sizeof(HI_S_Infrared));
```

注：设备支持请参阅附录厂家代码和设备类型定义的 S 字段。

4、 HI_CMD_VIDEO_PARAM

```
typedef struct HI_Video
{
    HI_U32      u32Channel;     //通道
    HI_BOOL     blFlag;         //主次码流标志，0-次码流，1 主码流
    HI_U32      u32Bitrate;     //码率 Kb
```

```
    HI_U32      u32Frame;       //帧率
    HI_U32      u32Iframe;      //主帧间隔（1-300）
    HI_BOOL     blCbr;          //视频编码控制 0-可变码率，1-固定码率
    HI_U32      u32ImgQuality;  //视频编码质量（1-6）
} HI_S_Video;
```

注：u32Channel 与 HI_SDK_RealPlay 的参数 HI_S_STREAM_INFO 中 u32Channel 一致。获取和设置都应当相同。

Example：

```
HI_S_Video sVideo;
// 注：u32Channel 与 HI_S_STREAM_INFO 一致
sVideo.u32Channel = HI_CHANNEL_1;
sVideo.blFlag = HI_TRUE;
sVideo.u32Bitrate = 1024;
sVideo.u32Frame = 25;
sVideo.u32Iframe = 50;
sVideo.blCbr = HI_FALSE;
sVideo.u32ImgQuality = 1;
HI_SDK_SetConfig (  lHandle,                // HI_SDK_GetConfig
                    HI_CMD_VIDEO_PARAM,
                    &sVideo,
                    sizeof(HI_S_Video));
```

5、 HI_CMD_OSD_PARAM

```
typedef struct HI_OSD
{
    HI_BOOL     blEnTime;       //叠加时间
    HI_BOOL     blEnName;       //叠加名称
    HI_CHAR     sName[64];      //OSD 名称  //最大 18 字节
} HI_S_OSD;
```

Example：

```
HI_S_OSD sOSD;
sOSD.blEnTime = HI_TRUE;
sOSD.blEnName = HI_TRUE;
strcpy(sOSD. sName, “IPCAM”);
HI_SDK_SetConfig (  lHandle,                // HI_SDK_GetConfig
                    HI_CMD_OSD_PARAM,
                    &sOSD,
                    sizeof(HI_S_OSD));
```

注：C5 设备类型的摄像机 linux 下 OSD 如果为中文 OSD，必须以 UTF-8 传入，获取到的字符也是 UTF-8

6、 HI_CMD_AUDIO_PARAM

```
typedef struct HI_Audio
{
    HI_U32      u32Channel;     //通道
    HI_BOOL     blFlag;         //主次码流标志，0-次码流，1 主码流
    HI_BOOL     blEnable;       //是否采集音频
    HI_U32      u32Type;        //音频格式
} HI_S_Audio;
```

注：u32Channel 与 HI_SDK_RealPlay 的参数 HI_S_STREAM_INFO 中 u32Channel 一致。获取和设置都应当相同。

u32Type 格式如下表：

宏定义	宏定义值	含义
HI_AUDIO_TYPE_G711	0	G711
HI_AUDIO_TYPE_G726	1	G726
HI_AUDIO_TYPE_AMR	2	AMR

Example：

```
HI_S_Audio sAudio;
// 注：u32Channel 与 HI_S_STREAM_INFO 一致
sAudio.u32Channel = HI_CHANNEL_1;
sAudio.blFlag = HI_TRUE;
sAudio.blEnable = HI_TRUE;
sAudio.u32Type = HI_AUDIO_TYPE_G711;
HI_SDK_SetConfig (  lHandle,                // HI_SDK_GetConfig
                    HI_CMD_AUDIO_PARAM,
                    &sAudio,
                    sizeof(HI_S_Audio));
```

7、 HI_CMD_AUDIO_INPUT

```
typedef enum HI_AudioInput
{
    AUDIO_INPUT_MIC = 100,      //麦克输入
    AUDIO_INPUT_LINE = 10       //线性输入
} HI_E_AudioInput;
```

Example：

```
HI_S32 audioInput = AUDIO_INPUT_MIC;
HI_SDK_SetConfig (  lHandle,                // HI_SDK_GetConfig
                    HI_CMD_AUDIO_INPUT,
                    &audioInput,
                    sizeof(HI_S32));
```

8、 HI_CMD_RESOLUTION

```
typedef struct HI_Resolution
{
```

```
    HI_U32      u32Channel;     //通道
    HI_BOOL     blFlag;         //主次码流标志，0-次码流，1 主码流
    HI_U32      u32Resolution;  //清晰度
} HI_S_Resolution;
```

注：u32Channel 与 HI_SDK_RealPlay 的参数 HI_S_STREAM_INFO 中 u32Channel 一致。获取和设置都应当相同。

u32Resolution 值如下表：

宏定义	值	含义
HI_RESOLUTION_VGA	0	VGA：640x480
HI_RESOLUTION_QVGA	1	QVGA：320x240
HI_RESOLUTION_QQVGA	2	QQVGA：160x120，160x112
HI_RESOLUTION_D1	3	D1：704x576，704x480
HI_RESOLUTION_CIF	4	CIF：352x288，352x240
HI_RESOLUTION_QCIF	5	QCIF：176x144，176x120，176x112
HI_RESOLUTION_720P	6	720P：1280x720

Example：

```
HI_S_Resolution sResolution;
// 注：u32Channel 与 HI_S_STREAM_INFO 一致
sResolution.u32Channel = HI_CHANNEL_1;
sResolution.blFlag = HI_TRUE;
sResolution.u32Resolution = HI_RESOLUTION_CIF;
HI_SDK_SetConfig (  lHandle,                // HI_SDK_GetConfig
                    HI_CMD_RESOLUTION,
                    &sResolution,
                    sizeof(HI_S_Resolution));
```

注：分辨率设备支持请参阅附录厂家代码和设备类型定义的 S 字段。

9、 HI_CMD_FREQUENCY

```
typedef enum HI_Frequency
{
    FREQ_50HZ_PAL = 50,     //50HZ
    FREQ_60HZ_NTSC = 60     //60HZ
} HI_E_Frequency;
```

Example：

```
HI_U32 sFrequency = FREQ_50HZ_PAL;
HI_SDK_SetConfig (  lHandle,                // HI_SDK_GetConfig
                    HI_CMD_FREQUENCY,
                    &sFrequency,
                    sizeof(HI_U32));
```

注：附录厂家代码和设备类型定义，目前不支持的设置频率的设备有 S1，S2 字段。

10、 HI_CMD_PTZ_PARAM

```
typedef struct HI_PTZ
{
    HI_U32      u32Protocol;      //协议
    HI_U32      u32Address;       //地址码，范围[0~255]
    HI_U32      u32Baud;          //波特率
    HI_U32      u32DataBit;       //数据位
    HI_U32      u32StopBit;       //停止位
    HI_U32      u32Parity;        //校验
} HI_S_PTZ;
```

u32Protocol 协议值如下表：

宏定义	宏定义值	含义
HI_PTZ_PRO_PELCOD	0	PELCO-D
HI_PTZ_PRO_PELCOP	1	PELCO-P

u32Baud 波特率数据如下表：

宏定义	宏定义值	含义
HI_PTZ_B110	110	110
HI_PTZ_B300	300	300
HI_PTZ_B1200	1200	1200
HI_PTZ_B2400	2400	2400
HI_PTZ_B4800	4800	4800
HI_PTZ_B9600	9600	9600
HI_PTZ_B19200	19200	19200
HI_PTZ_B38400	38400	38400
HI_PTZ_B57600	57600	57600

u32DataBit 数据位数据如下表：

宏定义	宏定义值	含义
HI_PTZ_DATA_5	5	
HI_PTZ_DATA_6	6	
HI_PTZ_DATA_7	7	
HI_PTZ_DATA_8	8	

u32StopBit 停止位数据如下表：

宏定义	宏定义值	含义
HI_PTZ_STOP_1	1	
HI_PTZ_STOP_2	2	

u32Parity 校验数据如下表：

宏定义	宏定义值	含义
HI_PTZ_PARITY_NONE	0	无
HI_PTZ_PARITY_ODD	1	奇校验
HI_PTZ_PARITY_EVEN	2	偶校验

Example：

```
HI_S_PTZ sPtz;
sPtz. u32Protocol = HI_PTZ_PRO_PELCOD;
sPtz. u32Address = 1;
```

```
sPtz. u32Baud = HI_PTZ_B9600;
sPtz. u32DataBit = HI_PTZ_DATA_8;
sPtz. u32StopBit = HI_PTZ_STOP_1;
sPtz. u32Parity = HI_PTZ_PARITY_NONE;
HI_SDK_SetConfig (    lHandle,                  // HI_SDK_GetConfig
                      HI_CMD_PTZ_PARAM,
                      &sPtz,
                      sizeof(HI_S_PTZ));
```

11、 HI_CMD_MD_PARAM

```
typedef struct HI_MD_PARAM
{
    HI_U32      u32Channel;      //通道
    HI_U32      u32Area;         //矩形区域（1~4）
    HI_BOOL     blEnable;        //是否启用
    HI_U32      u32Sensitivity;  //灵敏度（0~100）
    HI_U32      u32X;            //x 坐标
    HI_U32      u32Y;            //y 坐标
    HI_U32      u32Width;        //矩形宽度
    HI_U32      u32Height;       //矩形高度
} HI_S_MD_PARAM;
```

Example：

```
HI_S_MD_PARAM sMdParam;
// 注：u32Channel 与 HI_S_STREAM_INFO 一致
sMdParam.u32Channel = HI_CHANNEL_1;
sMdParam.u32Area = 1;
sMdParam.bEnable = HI_TRUE;
sMdParam.u32Sensitivity = 50;
sMdParam.u32X = 100;
sMdParam.u32Y = 100;
sMdParam.u32Width = 200;
sMdParam.u32Height = 200;
HI_SDK_SetConfig (    lHandle,                  // HI_SDK_GetConfig
                      HI_CMD_MD_PARAM,
                      &sMdParam,
                      sizeof(HI_S_MD_PARAM));
```

注：次码流不支持移动侦测。

12、 HI_CMD_NET_INFO

```
typedef struct tagHI_NETINFO
{
    HI_CHAR     aszServerIP[40];      //IP 地址
    HI_CHAR     aszNetMask[40];       //子网掩码
    HI_CHAR     aszGateWay[40];       //网关
```

```
    HI_CHAR     aszMacAddr[40];         //MAC 地址
    HI_CHAR     aszFDNSIP[40];          //first DNSIP
    HI_CHAR     aszSDNSIP[40];          //DNSIP
    HI_S32      s32DhcpFlag;            //DHCP
    HI_S32      s32DnsDynFlag;          //DNS 动态分配标识*/
}HI_S_NETINFO, *PHI_S_NETINFO;
```

Example：

```
HI_S_NETINFO sNetInfo;
strcpy(sNetInfo. aszServerIP, “192.168.1.88”);
… …
HI_SDK_SetConfig (  lHandle,                // HI_SDK_GetConfig
                    HI_CMD_NET_INFO,
                    &sNetInfo,
                    sizeof(HI_S_NETINFO));
```

13、 HI_CMD_HTTP_PORT

```
typedef struct HI_HTTPPORT
{
    HI_U32      u32HttpPort;
} HI_S_HTTPPORT;
```

Example：

```
HI_S_HTTPPORT sHttpPort;
sHttpPort.u32HttpPort = 80;
HI_SDK_SetConfig (  lHandle,                // HI_SDK_GetConfig
                    HI_CMD_HTTP_PORT,
                    &sHttpPort,
                    sizeof(HI_S_HTTPPORT));
```

14、 HI_CMD_SERVER_TIME

设置摄像机端时间

```
typedef struct hiSERVERTIME_INFO_S
{
    HI_CHAR sTime[32];              //摄像机时间，格式 2011.03.11.09.12.08
} HI_S_SERVERTIME;
```

sTime 为摄像机的时间，格式为 2011.03.11.09.12.08，即 2011-3-11 09:12:08

Example：

```
HI_S_SERVERTIME sServerTime;
memcpy(sServerTime.sTime, “2011.03.11.09.12.08” , sizeof(sServerTime.sTimezone));
HI_SDK_SetConfig (  lHandle,
                        HI_CMD_SERVER_TIME,
                        &sServerTime,
                        sizeof(HI_S_SERVERTIME));
```

15、 HI_CMD_REBOOT

重启摄像机

Example：

HI_SDK_SetConfig (lHandle, HI_CMD_REBOOT,NULL,0);

16、 HI_CMD_RESET

恢复出厂设置

Example：

HI_SDK_SetConfig (lHandle, HI_CMD_RESET,NULL,0);

17、 HI_CMD_NET_EXT

```
typedef struct HI_NET_EXT
{
    HI_S_NETINFO    sNetInfo;
    HI_S_HTTPPORT sHttpPort;
}HI_S_NET_EXT;

typedef struct HI_HTTPPORT
{
    HI_U32        u32HttpPort;
} HI_S_HTTPPORT;

typedef struct tagHI_NETINFO
{
    HI_CHAR     aszServerIP[40];        //IP 地址
    HI_CHAR     aszNetMask[40];         //子网掩码
    HI_CHAR     aszGateWay[40];         //网关
    HI_CHAR     aszMacAddr[40];         //MAC  地址
    HI_CHAR     aszFDNSIP[40];          //first DNSIP
    HI_CHAR     aszSDNSIP[40];          //DNSIP
    HI_S32      s32DhcpFlag;            //DHCP
    HI_S32      s32DnsDynFlag;          //DNS  动态分配标识*/
}HI_S_NETINFO, *PHI_S_NETINFO;
```

Example：

```
HI_S_NET_EXT sNetExt;
strcpy(sNetExt.sNetInfo. aszServerIP, “192.168.1.88”);
… …
HI_SDK_SetConfig (    lHandle,                  // HI_SDK_GetConfig
                      HI_CMD_NET_EXT,
                      & sNetExt,
                      sizeof(HI_S_NET_EXT));
```

18、 HI_CMD_ATTR_EXT

设置输入报警开关

```
typedef struct HI_ATTR_EXT
{
```

```
    HI_U32    u32Enable;    //1-启用，0-禁用
    HI_U32    u32Flag;      //0-关闭，1-打开
}HI_S_ATTR_EXT;
```

Example：

```
HI_S_ATTR_EXT sAttrExt;
sAttrExt.u32Enable = 1;
sAttrExt.u32Flag = 0;
HI_SDK_SetConfig (    lHandle,
                      HI_CMD_ATTR_EXT,
                      & sAttrExt,
                      sizeof(HI_S_ATTR_EXT));
```

19、 HI_NVR_CMD_NET_EXT

设置 NVR 网络参数

```
typedef struct HI_NET_EXT
{
    HI_S_NETINFO    sNetInfo;
    HI_S_HTTPPORT sHttpPort;
}HI_S_NET_EXT;

typedef struct HI_HTTPPORT
{
    HI_U32        u32HttpPort;
} HI_S_HTTPPORT;

typedef struct tagHI_NETINFO
{
    HI_CHAR     aszServerIP[40];        //IP 地址
    HI_CHAR     aszNetMask[40];         //子网掩码
    HI_CHAR     aszGateWay[40];         //网关
    HI_CHAR     aszMacAddr[40];         //MAC  地址
    HI_CHAR     aszFDNSIP[40];          //first DNSIP
    HI_CHAR     aszSDNSIP[40];          //DNSIP
    HI_S32      s32DhcpFlag;            //DHCP
    HI_S32      s32DnsDynFlag;          //DNS  动态分配标识*/
}HI_S_NETINFO, *PHI_S_NETINFO;
```

Example：

```
HI_S_NET_EXT sNetExt;
strcpy(sNetExt.sNetInfo. aszServerIP, “192.168.1.88”);
… …
HI_SDK_SetConfig (    lHandle,          // HI_SDK_GetConfig
                      HI_NVR_CMD_NET_EXT,
                      & sAttrExt,
                      sizeof(HI_S_NET_EXT));
```

20、 HI_NVR_CMD_USER

设置 NVR 用户信息

```
typedef struct HI_USER
{
    HI_CHAR sUsername[32];   //用户名，用户名只有 admin、user 和 guest
    HI_CHAR sPassword[32];   //密码
} HI_S_USER;
```

Example：

```
HI_S_USER sUserInfo;
strcpy(sUserInfo.sUsername, "admin");
strcpy(sUserInfo. sPassword, "admin");
HI_SDK_SetConfig (    lHandle,
                      HI_NVR_CMD_USER,
                      & sUserInfo,
                      sizeof(HI_S_USER));
```

21、 HI_NVR_CMD_CHANNEL_INFO

设置 NVR 通道信息

```
typedef struct HI_CHN_INFO
{
    HI_U32 u32Enable;          //设置通道状态 0-禁用，1-启用
    HI_CHAR sHost[24];         //设备 IP 地址
    HI_BOOL bStream;           //码流，在 NVR 中暂时不起作用
    HI_U32 u32Port;            //端口
    HI_U32 u32Chn;             //通道，在 NVR 中不支持
    HI_CHAR sUsername[32];     //用户名
    HI_CHAR sPassword[32];     //密码
}HI_S_CHN_INFO;

typedef struct hiNVR_CHN
{
    HI_CHAR sName[32];  //通道名称，字符要求是 UTF-8，
                        //例如中文字符要转成 UTF-8
    HI_S_CHN_INFO sChnInfo;
}HI_S_NVR_CHN;
```

Example：

```
HI_S_NVR_CHN sNvrChn;
strcpy(sNvrChnInfo.sChnInfo.sHost, “192.168.1.20”);
sNvrChn.sChnInfo.u32Port = 80;
sNvrChn.sChnInfo.u32Enable = 1;
… …
HI_SDK_SetConfig (    lHandle,          // HI_SDK_GetConfig
                      HI_NVR_CMD_CHANNEL_INFO,
                      & sNvrChn,
                      sizeof(HI_S_NVR_CHN));
```

注：调用一次只能获取或设置一个通道，可以配合 HI_SDK_SetChannel 设置 NVR 的通道再来操作。

*** 通道名称如果是宽字符，要求转换成 UTF-8 格式，如果不是，设置将失败。获取通道信息返回的名称也是宽字符也是 UTF-8 格式的，需要转换。**

22、 HI_NVR_CMD_RECORD_INFO

设置 NVR 通道录像信息

```
typedef struct HI_RECORD_INFO
{
    HI_BOOL bStream;      //通道录像码流，HI_TRUE-主码流，HI_FALSE-次码流
    HI_U32 u32SetupAlarm;     //联动录像开关，0-禁用，1-启用
    HI_U32 u32InputAlarm;     //输入报警联动开关，0-禁用，1-启用
    HI_U32 u32MdAlarm;        //移动侦测联动开关，0-禁用，1-启用
    HI_CHAR sRecInfo[7][48+1];   //计划录像录像时间段，7 天，没半小时为一个
                                 单元间隔，如星期一时间内的计划录像时间段
                                 为                              ：
                                 strcpy(sRecInfo[1], ”PPPPPPPPPPPPPPPPPPPP
                                 PPPPPPPPPPNNNNNNNNNNPPPPPPPPPPPPPP
                                 PPPPPPPP”);启用 P 代表计划录像，N 代表不录
                                 像。

}HI_S_RECORD_INFO;
```

Example：

```
HI_S_RECORD_INFO sRecInfo;
sRecInfo.bStream = HI_TRUE;
… …
strcpy(sRecInfo.sRecInfo[0], ”PPPPPPPPPPPPPPPPPPPPPPPPPPPPNNNNNNNNNN
NPPPPPPPPPPPPPPPPPPPPPP”); //星期天计划录像表
… …
HI_SDK_SetConfig (   lHandle,              // HI_SDK_GetConfig
                     HI_NVR_CMD_RECORD_INFO,
                     & sRecInfo,
                     sizeof(HI_S_RECORD_INFO));
```

注：调用一次只能获取或设置一个通道，可以配合 HI_SDK_SetChannel 设置 NVR 的通道再来操作。

23、 HI_NVR_CMD_RECORD_SYS

设置 NVR 全局信息

```
typedef struct HI_RECORD_SYS
{
    HI_U32 u32RecLen;         //录像文件时长[1-30 分钟]
    HI_U32 u32AlarmLen;       //报警延续时长[5-60 秒]
    HI_U32 u32Cover;          //磁盘满是否覆盖[0-否, 1-是]
```

```
    HI_U32 u32PlanRecFlag;     //计划录像开关[0-关, 1-开]
    HI_U32 u32PreRec;          //报警预录时长[1-5 秒]
    HI_U32 u32RecType;         //录像文件格式类型[1-264, 0-AVI]
    HI_U32 u32DiskRemain;      //磁盘剩余空间[1-10 G]
}HI_S_RECORD_SYS;
```

Example：

```
HI_S_RECORD_SYS sNvrRecSys;
sNvrRecSys.u32RecLen = 10;
sNvrRecSys.u32PreRec = 1;
sNvrRecSys.u32AlarmLen = 10;
… …
HI_SDK_SetConfig (   lHandle,                  // HI_SDK_GetConfig
                     HI_NVR_CMD_RECORD_SYS,
                     & sNvrRecSys,
                     sizeof(HI_S_RECORD_SYS));
```

24、 HI_NVR_CMD_TIME

设置 NVR 前端时间

```
typedef struct hiSERVERTIME_INFO_S
{
    HI_CHAR sTime[32];              //NVR 时间，格式 20110311091208
} HI_S_SERVERTIME;
```

sTime 为摄像机的时间，格式为 20110311091208，即 2011-3-11 09:12:08

Example：

```
HI_S_SERVERTIME sServerTime;
memcpy(sServerTime.sTime, “20110311091208” , sizeof(sServerTime.sTimezone));
HI_SDK_SetConfig (   lHandle,              // HI_SDK_GetConfig
                     HI_NVR_CMD_TIME,
                     &sServerTime,
                     sizeof(HI_S_SERVERTIME));
```

25、 HI_NVR_CMD_REBOOT

重启摄像机

Example：

```
HI_SDK_SetConfig (lHandle, HI_NVR_CMD_REBOOT, NULL, 0);
```

26、 HI_NVR_CMD_RESET

恢复出厂设置

Example：

```
HI_SDK_SetConfig (lHandle, HI_NVR_CMD_RESET, NULL, 0);
```

27、 HI_NVR_CMD_DISK_FORMAT

格式化硬盘

```
typedef struct HI_DISK_FORMAT
{
```

```
    HI_S32 s32DiskNum;   //硬盘分区，从 1 开始，第一块硬盘既是 1
}HI_DISK_FORMAT;
```

Example：

```
HI_DISK_FORMAT sDisFormat;
sDiskFormat.s32DiskNum = 1;
HI_SDK_SetConfig (lHandle,
                  HI_NVR_CMD_DISK_FORMAT,
                  &sDisFormat,
                  Sizeof(HI_DISK_FORMAT));
```

28、 HI_CMD_WIFI_PARAM

Wifi 参数设置

```
#define WIFI_NET_INFRA   0
#define WIFI_NET_ADHOC 1

#define WIFI_AUTH_NONE0
#define WIFI_AUTH_WEP   1
#define WIFI_AUTH_WPA   2
#define WIFI_AUTH_WPA2 3

#define WIFI_ENC_TKIP     0
#define WIFI_ENC_AES      1
typedef struct HI_WIFI_PARAM
{
    HI_CHAR sSsID[32];    //wifi SSID
    HI_CHAR sKey[32];     //wifi  密钥
    HI_U32 u32Enable;     //wifi  开关，  1-开启  0-关闭
    HI_U32 u32Auth;       //加密方式
    HI_U32 u32Enc;        //密码类型
    HI_U32 u32Mode;       //连接模式，1-点对点，0-路由
}HI_S_WIFI_PARAM;
```

Example：

```
HI_S_WIFI_PARAM sWifi;
strcpy(sWifi.sSsID, “linksys”);
……
HI_SDK_SetConfig (lHandle,
                  HI_CMD_WIFI_PARAM,
                  & sWifi,
                  Sizeof(HI_S_WIFI_PARAM));
```

29、 HI_CMD_WIFI_CHECK

Wifi check

Example：

```
HI_S_WIFI_PARAM sWifiParam;
//memset(&sWifiParam, 0, sizeof(HI_S_WIFI_PARAM));
```

```
strcpy(sWifiParam.sKey, "1234567890");
strcpy(sWifiParam.sSsID, "linksys");
sWifiParam.u32Mode = WIFI_NET_INFRA;
sWifiParam.u32Auth = WIFI_AUTH_WPA2;
sWifiParam.u32Enc = WIFI_ENC_AES;
s32Ret = HI_SDK_SetConfig( m_uiHandle,
                           HI_CMD_WIFI_CHECK,
                           &sWifiParam,
                           sizeof(HI_S_WIFI_PARAM));

if(HI_SUCCESS != s32Ret)
{
    return;
}

HI_S32 s32Enable = 0;
s32Ret = HI_SDK_GetConfig(m_uiHandle,
                           HI_CMD_WIFI_CHECK,
                           &s32Enable,
                           sizeof(HI_S32));

if(HI_SUCCESS != s32Ret)
{
    MessageBox ("check fail");
    return;
}
```

s32Enable 等于 1 表示 check 成功，否则失败！

30、 HI_CMD_VIDEO_PARAM_EXT

```
typedef struct HI_Video_Ext
{
    HI_U32      u32Channel;     //通道
    HI_U32      u32Stream;      // 0-次码流，1 主码流，2-第三码流
    HI_U32      u32Bitrate;     //码率 Kb
    HI_U32      u32Frame;       //帧率
    HI_U32      u32Iframe;      //主帧间隔（1-300）
    HI_BOOL     blCbr;          //视频编码控制 0-可变码率，1-固定码率
    HI_U32      u32ImgQuality;  //视频编码质量（1-6）
} HI_S_Video_Ext;
```

注：u32Channel 与 HI_NET_DEV_StartStream 的参数 HI_S_STREAM_INFO 中 u32Channel 一致。获取和设置都应当相同。

Example：

```
HI_S_Video_Ext sVideo;
```

```
// 注：u32Channel 与 HI_S_STREAM_INFO 一致
sVideo.u32Channel = HI_CHANNEL_1;
sVideo.u32Stream = HI_STREAM_1;
sVideo.u32Bitrate = 1024;
sVideo.u32Frame = 25;
sVideo.u32Iframe = 50;
sVideo.blCbr = HI_FALSE;
sVideo.u32ImgQuality = 1;
HI_SDK_SetConfig (    lHandle,                  // HI_SDK_GetConfig
                      HI_CMD_VIDEO_PARAM_EXT,
                      &sVideo,
                      sizeof(HI_S_Video_Ext));
```

31、 HI_CMD_AUDIO_PARAM_EXT

```
typedef struct HI_Audio_Ext
{
    HI_U32      u32Channel;     //通道
    HI_U32      u32Stream;      // 0-次码流，1 主码流，2-第三码流
    HI_BOOL     blEnable;       //是否采集音频
    HI_U32      u32Type;        //音频格式
} HI_S_Audio_Ext;
```

注：u32Channel 与 HI_NET_DEV_StartStream 的参数 HI_S_STREAM_INFO 中 u32Channel 一致。获取和设置都应当相同。

u32Type 格式如下表：

宏定义	宏定义值	含义
HI_AUDIO_TYPE_G711	0	G711
HI_AUDIO_TYPE_G726	1	G726

Example：

```
HI_S_Audio_Ext sAudio;
// 注：u32Channel 与 HI_S_STREAM_INFO 一致
sAudio.u32Channel = HI_CHANNEL_1;
sAudio.u32Stream = HI_STREAM_1;
sAudio.blEnable = HI_TRUE;
sAudio.u32Type = HI_AUDIO_TYPE_G711;
HI_SDK_SetConfig (    lHandle,                  // HI_SDK_GetConfig
                      HI_CMD_AUDIO_PARAM_EXT,
                      &sAudio,
                      sizeof(HI_S_Audio_Ext));
```

32、 HI_CMD_RESOLUTION_EXT

```
typedef struct HI_Resolution_Ext
{
    HI_U32      u32Channel;     //通道
    HI_U32      u32Stream;      // 0-次码流，1 主码流，2-第三码流
```

```
    HI_U32        u32Resolution;     //清晰度
} HI_S_Resolution_Ext;
```

注：u32Channel 与 HI_NET_DEV_StartStream 的参数 HI_S_STREAM_INFO 中 u32Channel 一致。获取和设置都应当相同。

u32Resolution 值如下表：

宏定义	值	含义
HI_RESOLUTION_VGA	0	VGA：640x480
HI_RESOLUTION_QVGA	1	QVGA：320x240
HI_RESOLUTION_QQVGA	2	QQVGA：160x120，160x112
HI_RESOLUTION_D1	3	D1：704x576，704x480
HI_RESOLUTION_CIF	4	CIF：352x288，352x240
HI_RESOLUTION_QCIF	5	QCIF：176x144，176x120，176x112
HI_RESOLUTION_720P	6	720P：1280x720

Example：

```
HI_S_Resolution_Ext sResolution;
// 注：u32Channel 与 HI_S_STREAM_INFO 一致
sResolution.u32Channel = HI_CHANNEL_1;
sResolution. u32Stream = HI_STREAM_1;
sResolution.u32Resolution = HI_RESOLUTION_CIF;
HI_SDK_SetConfig (    lHandle,                // HI_SDK_GetConfig
                      HI_CMD_RESOLUTION_EXT,
                      &sResolution,
                      sizeof(HI_S_Resolution_Ext));
```

注：分辨率设备支持请参阅附录厂家代码和设备类型定义的 S 字段。

33、 HI_CMD_AUDIO_VOLUME_IN

```
typedef struct HI_AudioVolume
{
    HI_U32 u32AudioVolume;  //音频音量，范围：1--100
} HI_S_AudioVolume;
```

Example：

```
HI_S_AudioVolume sAuVolume;
sAuVolume.u32AudioVolume = 80
HI_SDK_SetConfig (    lHandle,                // HI_SDK_GetConfig
                      HI_CMD_AUDIO_VOLUME_IN,
                      &sAuVolume,
                      sizeof(HI_S_AudioVolume));
```

34、 HI_CMD_AUDIO_VOLUME_OUT

```
typedef struct HI_AudioVolume
{
    HI_U32 u32AudioVolume;  //音频音量，范围：1--100
```

```
} HI_S_AudioVolume;
```

Example：

```
HI_S_AudioVolume sAuVolume;
sAuVolume.u32AudioVolume = 80
HI_SDK_SetConfig (    lHandle,                  // HI_SDK_GetConfig
                      HI_CMD_AUDIO_VOLUME_OUT,
                      &sAuVolume,
                      sizeof(HI_S_AudioVolume));
```

35、 HI_CMD_RELAYCTRL

```
typedef struct HI_RELAYCTRL {
    HI_BOOL bEnable;      //是否开启继电器 开启：HI_TRUE， 关闭：HI_FALSE
} HI_S_RelayCtrl;
```

36、 HI_CMD_EXT_ALARM

手动触发一个外置报警

Example：

```
HI_S_ExtAlarm m_ExtAlarm;
m_ExtAlarm.bEnable=HI_TRUE;
HI_SDK_SetConfig (lHandle,
                      HI_NET_DEV_CMD_EXT_ALARM_SNAP,
                      &m_ExtAlarm,
                      Sizeof(HI_S_ExtAlarm);
```

37 HI_CMD_QUANTUM_TIME

设置报警时间段，计划录像时间段，计划抓拍时间段等

```
#define HI_QT_TYPE_ALARM 0
#define HI_QT_TYPE_PLAN   1
#define HI_QT_TYPE_SNAP   2
typedef struct HI_QUANTUM_TIME
{
        HI_U32 u32QtType;              //HI_QT_TYPE_ALARM,
        HI_QT_TYPE_PLAN
        HI_CHAR sDayData[7][48+1];     //P, N
}HI_S_QUANTUM_TIME;
```

Example:

```
HI_QUANTUM_TIME QuanTum;
HI_SDK_SetConfig (lHandle,
                  HI_CMD_QUANTUM_TIME,
                  &QuanTum,
                  Sizeof(HI_QUANTUM_TIME);
```

HI_SDK_GetConfig

获取摄像机参数

```
HI_S32    HI_SDK_GetConfig (
        HI_U32              u32Handle
        HI_U32              u32Command,
        HI_VOID*            pBuf,
        HI_U32              u32BufLen
);
```

Parameters

u32Handle

[IN] 操作句柄

u32Command

[IN] 操作参数命令

宏定义	宏定义值	含义
HI_GET_PRODUCT_VENDOR	0x1000	厂商信息
HI_CMD_DISPLAY	0x1001	图像参数
HI_CMD_DISPLAY_EXT	0x1002	上下翻白平衡
HI_CMD_INFRARED	0x1003	红外
HI_CMD_VIDEO_PARAM	0x1004	视频参数
HI_CMD_OSD_PARAM	0x1005	OSD 参数
HI_CMD_AUDIO_PARAM	0x1006	音频参数
HI_CMD_AUDIO_INPUT	0x1007	音频输入
HI_CMD_RESOLUTION	0x1008	图像分辨率
HI_CMD_FREQUENCY	0x1009	频率
HI_CMD_PTZ_PARAM	0x1010	云台信息
HI_CMD_MD_PARAM	0x1011	移动报警信息
HI_CMD_NET_INFO	0x1012	网络配置信息
HI_CMD_HTTP_PORT	0x1013	网页端口号
HI_CMD_DEVICE_INFO	0x1014	设备信息
HI_CMD_PRODUCTID	0x1015	产品 ID
HI_CMD_USERNUM	0x1016	用户连接数
HI_CMD_SERVER_TIME	0x1017	获取摄像机时间
HI_CMD_NET_EXT	0x1022	获取网络参数
HI_CMD_ATTR_EXT	0x1026	获取输入报警参数
HI_NVR_CMD_NET_EXT	0x1050	NVR 网络参数
HI_NVR_CMD_RTSP_INFO	0x1051	NVR rtsp 参数
HI_NVR_CMD_USER	0x1052	NVR 用户参数
HI_NVR_CMD_CHANNEL_INFO	0x1053	NVR 通道信息
HI_NVR_CMD_SEARCH	0x1055	NVR 搜索 NVR 网络中的摄像机
HI_NVR_CMD_RECORD_INFO	0x1056	NVR 通道录像参数

HI_NVR_CMD_RECORD_SYS	0x1057	NVR 全局参数
HI_NVR_CMD_TIME	0x1058	NVR 时间参数
HI_NVR_CMD_RECORD_STATE	0x1061	NVR 录像状态
HI_NVR_CMD_DISK_INFO	0x1062	NVR 硬盘状态
HI_NVR_CMD_RECORD_STATE_EX	0x1064	NVR 录像状态
HI_CMD_WIFI_PARAM	0x1030	WIFI 参数设置
HI_CMD_WIFI_SEARCH	0x1031	WIFI 搜索
HI_CMD_WIFI_CHECK	0x1035	WIFI check
HI_CMD_VIDEO_PARAM_EXT	0x1047	视频参数（扩展）
HI_CMD_AUDIO_PARAM_EXT	0x1048	音频参数（扩展）
HI_CMD_RESOLUTION_EXT	0x1049	分辨率参数（扩展）
HI_CMD_AUDIO_VOLUME_IN	0x1070	音频输入音量
HI_CMD_AUDIO_VOLUME_OUT	0x1071	音频输出音量
HI_CMD_QUANTUM_TIME	0x1072	报警时间段

pBuf

[OUT] 获取数据

u32BufLen

[IN] 数据长度

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remarks

各个命令对应的结构体同 HI_SDK_SetConfig 中一致，其中 HI_SDK_SetConfig 中没有的属性如下：

1、HI_GET_PRODUCT_VENDOR

```
typedef struct HI_ProductVendor
{
    HI_CHAR     sProduct[32];       //产品 ID
    HI_CHAR     sVendor[32];        //供应商 ID
}HI_S_ProductVendor;
```

Example：

```
HI_S_ProductVendor sProduct;
HI_SDK_GetConfig (    lHandle,
                      HI_GET_PRODUCT_VENDOR,
                      &sProduct,
                      sizeof(HI_S_ProductVendor));
```

2、HI_CMD_DEVICE_INFO

```
typedef struct tagHI_DEVICE_INFO
{
    HI_CHAR aszServerSerialNumber[40 + 1];      //设备序列号
    HI_CHAR aszServerSoftVersion[64 + 1];       //软件版本
```

```
    HI_CHAR aszServerName[40 + 1];          //服务器名称
    HI_CHAR aszServerModel[40 + 1];         //型号
    HI_CHAR aszStartDate[40 + 1];           //系统启动日期时间
    HI_S32 s32ConnectState;                 //网络连接状态
}HI_DEVICE_INFO, *PHI_DEVICE_INFO;
```

Example：

```
HI_DEVICE_INFO sDeviceInfo;
HI_SDK_GetConfig (  lHandle,
                    HI_CMD_DEVICE_INFO,
                    &sDeviceInfo,
                    sizeof(HI_DEVICE_INFO));
```

3、HI_CMD_PRODUCTID

产品 ID 用字符串表示。

Example：

```
HI_CHAR sID[64] = {0};
HI_SDK_GetConfig(lHandle, HI_CMD_PRODUCTID, sID, sizeof(sID));
```

4、HI_CMD_USERNUM

获取用户数据用到 int 来获取即可。

Example：

```
int nNum = 0;
HI_SDK_GetConfig(lHandle, HI_CMD_USERNUM, &nNum, sizeof(int));
```

5、HI_CMD_SERVER_TIME

获取摄像机端时间

```
typedef struct hiSERVERTIME_INFO_S
{
    HI_CHAR sTime[32];              //摄像机时间，格式 20110311091208
} HI_S_SERVERTIME;
```

sTime 为摄像机的时间，格式为 20110311091208，即 2011-3-11 09:12:08

Example：

```
HI_S_SERVERTIME sServerTime;
HI_SDK_GetConfig (  lHandle,
                    HI_CMD_SERVER_TIME,
                    &sServerTime,
                    sizeof(HI_S_SERVERTIME));
```

6、 HI_NVR_CMD_NET_EXT

获取 NVR 网络参数

```
typedef struct HI_NET_EXT
{
    HI_S_NETINFO    sNetInfo;
    HI_S_HTTPPORT sHttpPort;
}HI_S_NET_EXT;
```

```
typedef struct HI_HTTPPORT
{
    HI_U32      u32HttpPort;
} HI_S_HTTPPORT;

typedef struct tagHI_NETINFO
{
    HI_CHAR     aszServerIP[40];        //IP 地址
    HI_CHAR     aszNetMask[40];         //子网掩码
    HI_CHAR     aszGateWay[40];         //网关
    HI_CHAR     aszMacAddr[40];         //MAC 地址
    HI_CHAR     aszFDNSIP[40];          //first DNSIP
    HI_CHAR     aszSDNSIP[40];          //DNSIP
    HI_S32      s32DhcpFlag;            //DHCP
    HI_S32      s32DnsDynFlag;          //DNS 动态分配标识*/
}HI_S_NETINFO, *PHI_S_NETINFO;
```

Example：

```
HI_S_NET_EXT sNetExt;
HI_SDK_GetConfig (  lHandle,
                    HI_NVR_CMD_NET_EXT,
                    & sAttrExt,
                    sizeof(HI_S_NET_EXT));
```

7、 HI_NVR_CMD_USER

获取 NVR 用户信息

```
typedef struct HI_USER
{
    HI_CHAR sUsername[32];  //用户名，用户名只有 admin、user 和 guest
    HI_CHAR sPassword[32];  //密码
} HI_S_USER;

typedef struct HI_USERINFO
{
    HI_S_USER sUser[3];  //用户名只有 admin、user 和 guest
} HI_S_USERINFO;
```

Example：

```
HI_S_USERINFO sUserInfo;
HI_SDK_GetConfig (  lHandle,
                    HI_NVR_CMD_USER,
                    & sUserInfo,
                    sizeof(HI_S_USERINFO));
```

8、 HI_NVR_CMD_CHANNEL_INFO

获取 NVR 通道信息

```
typedef struct HI_CHN_INFO
{
    HI_U32 u32Enable;          //设置通道状态 0-禁用，1-启用
    HI_CHAR sHost[24];         //设备 IP 地址
    HI_BOOL bStream;           //码流，在 NVR 中暂时不起作用
    HI_U32 u32Port;            //端口
    HI_U32 u32Chn;             //通道，在 NVR 中不支持
    HI_CHAR sUsername[32];     //用户名
    HI_CHAR sPassword[32];     //密码
}HI_S_CHN_INFO;
```

Example：

```
HI_S_CHN_INFO sNvrChnInfo;
HI_SDK_GetConfig (   lHandle,
                     HI_NVR_CMD_CHANNEL_INFO,
                     & sNvrChnInfo,
                     sizeof(HI_S_CHN_INFO));
```

注：调用一次只能获取或设置一个通道，可以配合 HI_SDK_SetChannel 设置 NVR 的通道再来操作。

9、 HI_NVR_CMD_RECORD_INFO

获取 NVR 通道录像信息

```
typedef struct HI_RECORD_INFO
{
    HI_BOOL bStream;      //通道录像码流，HI_TRUE-主码流，HI_FALSE-次码流
    HI_U32 u32SetupAlarm;      //联动录像开关，0-禁用，1-启用
    HI_U32 u32InputAlarm;      //输入报警联动开关，0-禁用，1-启用
    HI_U32 u32MdAlarm;         //移动侦测联动开关，0-禁用，1-启用
    HI_CHAR sRecInfo[7][48+1];    //计划录像录像时间段，7 天，没半小时为一个
                                  单元间隔，如星期一时间内的计划录像时间段
                                  为                              ：
                                  strcpy(sRecInfo[1], ”PPPPPPPPPPPPPPPPPPPP
                                  PPPPPPPPPPNNNNNNNNNNPPPPPPPPPPPPPP
                                  PPPPPPPPP”);启用 P 代表计划录像，N 代表不录
                                  像。

}HI_S_RECORD_INFO;
```

Example：

```
HI_S_RECORD_INFO sRecInfo;
HI_SDK_GetConfig (   lHandle,
                     HI_NVR_CMD_RECORD_INFO,
                     & sRecInfo,
                     sizeof(HI_S_RECORD_INFO));
```

注：调用一次只能获取或设置一个通道，可以配合 HI_SDK_SetChannel 设置

NVR 的通道再来操作。

10、 HI_NVR_CMD_RECORD_SYS

获取 NVR 全局信息

```
typedef struct HI_RECORD_SYS
{
    HI_U32 u32RecLen;          //录像文件时长[1-30 分钟]
    HI_U32 u32AlarmLen;        //报警延续时长[5-60 秒]
    HI_U32 u32Cover;           //磁盘满是否覆盖[0-否, 1-是]
    HI_U32 u32PlanRecFlag;     //计划录像开关[0-关, 1-开]
    HI_U32 u32PreRec;          //报警预录时长[1-5 秒]
    HI_U32 u32RecType;         //录像文件格式类型[1-264, 0-AVI]
    HI_U32 u32DiskRemain;      //磁盘剩余空间[1-10 G]
}HI_S_RECORD_SYS;
```

Example：

```
HI_S_RECORD_SYS sNvrRecSys;
HI_SDK_GetConfig (   lHandle,
                     HI_NVR_CMD_RECORD_SYS,
                     & sNvrRecSys,
                     sizeof(HI_S_RECORD_SYS));
```

11、 HI_NVR_CMD_TIME

获取 NVR 前端时间

```
typedef struct hiSERVERTIME_INFO_S
{
    HI_CHAR sTime[32];          //NVR 时间，格式 20110311091208
} HI_S_SERVERTIME;
```

sTime 为摄像机的时间，格式为 20110311091208，即 2011-3-11 09:12:08

Example：

```
HI_S_SERVERTIME sServerTime;
HI_SDK_GetConfig (   lHandle,
                     HI_NVR_CMD_TIME,
                     &sServerTime,
                     sizeof(HI_S_SERVERTIME));
```

12、 HI_NVR_CMD_SEARCH

搜索与 NVR 在一个局域网内的摄像机

```
typedef struct HI_DEVINFO
{
    HI_CHAR sHost[32];          //IP 地址
    HI_U32 u32Port;             //端口
}HI_S_DEVINFO;

#define MAX_SEARCH_NUM 64       //最大搜索设备的书
typedef struct HI_SEARCH_INFO
```

```
{
    HI_U32 u32Num;                  //返回设备的数量
    HI_S_DEVINFO sDevInfo[MAX_SEARCH_NUM]; //设备信息
}HI_S_SEARCH_INFO;
```

Example：

```
HI_S_SEARCH_INFO sSearchInfo;
HI_SDK_GetConfig (  lHandle,
                    HI_NVR_CMD_SEARCH,
                    &sSearchInfo,
                    sizeof(HI_S_SEARCH_INFO));
```

13、 HI_NVR_CMD_RECORD_STATE
HI_NVR_CMD_RECORD_STATE_EX

获取录像状态，两者的区别是 EX 可以同时获取多个通道的状态

```
typedef struct HI_REC_STATE
{
    HI_U32 u32link;         //录像连接状态  0-表示没有连接，1-表示连接
    HI_U32 u32Record;       //录像状态  0-无录像，2-报警录像，3-计划录像
}HI_S_REC_STATE;
```

Example：

```
HI_S_REC_STATE sRecState;
HI_SDK_GetConfig (  lHandle,
                        HI_NVR_CMD_RECORD_STATE,
                        &sRecState,
                        sizeof(HI_S_REC_STATE));
```

如果想一次获取多通道状态，可以定义一个结构体如下：

```
typedef struct HI_STATES
{
    HI_S_REC_STATE sRecState[16];       //16 个通道同时获取
}HI_S_STATES;
```

Example：

```
HI_S_STATES sRecState;
HI_SDK_GetConfig (  lHandle,
                    HI_NVR_CMD_RECORD_STATE_EX,
                    &sRecState,
                    sizeof(HI_S_STATES));
```

注：调用一次只能获取或设置一个通道，可以配合 HI_SDK_SetChannel 设置 NVR 的通道再来操作。

14、 HI_NVR_CMD_DISK_INFO

获取硬盘信息

```
typedef struct HiDISK
```

```
{
    HI_U32 u32Total;        //硬盘总大小，  单位：KB
    HI_U32 u32Free;         //硬盘可用大小，  单位：KB
}HI_S_DISK;

#define MAX_DISK_NUM 20          //最大 20 块硬盘
typedef struct HI_DISK_INFO
{
    HI_S32 s32Num;                          //硬盘总数
    HI_S_DISK sDisk[MAX_DISK_NUM];     //硬盘相关信息
}HI_S_DISK_INFO;
```

Example：

```
HI_S_DISK_INFO sDiskInfo;
HI_SDK_GetConfig (    lHandle,
                      HI_NVR_CMD_DISK_INFO,
                      & sDiskInfo,
                          sizeof(HI_S_DISK_INFO));
```

15、 HI_CMD_WIFI_SEARCH

查找 WIFI

```
#define WIFI_NET_INFRA   0
#define WIFI_NET_ADHOC 1

#define WIFI_AUTH_NONE0
#define WIFI_AUTH_WEP   1
#define WIFI_AUTH_WPA   2
#define WIFI_AUTH_WPA2 3

#define WIFI_ENC_TKIP     0
#define WIFI_ENC_AES      1

typedef struct HI_WFPT
{
    HI_CHAR sEssID[32];
    HI_S32 s32Chn;
    HI_S32 s32Rssi;
    HI_U32 u32Enc;
    HI_U32 u32Auth;
    HI_U32 u32Net;
}HI_S_WFPT;

#define MAX_WFPT 64
typedef struct HI_WIFI_INFO
{
    HI_S32 s32Num;
    HI_S_WFPT sWfPt[MAX_WFPT];
```

```
}HI_S_WIFI_INFO;
```

Example：

```
HI_S_WIFI_INFO sWifiInfo;
memset(&sWifiInfo, 0, sizeof(HI_S_WIFI_INFO));
s32Ret = HI_SDK_GetConfig(m_uiHandle,
                          HI_CMD_WIFI_SEARCH,
                          &sWifiInfo,
                          sizeof(HI_S_WIFI_INFO));

if(HI_SUCCESS != s32Ret)
{
    MessageBox("Wifi seach fail!");
    return;
}

for(int i=0; i<sWifiInfo.s32Num; i++)
{
    printf("SSID:%s, AUTH:%d, ENC:%d, NET:%d\n",
        sWifiInfo.sWfPt[i].sEssID,
        sWifiInfo.sWfPt[i].u32Auth,
        sWifiInfo.sWfPt[i].u32Enc,
        sWifiInfo.sWfPt[i].u32Net);
}
```

1.5 预览解码效果控制

HI_SDK_SetPlayerBufNumber

设置网络延时和播放流畅度可以通过此接口来进行调节

```
HI_S32   HI_SDK_SetPlayerBufNumbe r(
    HI_HANDLE       lHandle,
    HI_S32          s32BufNum
);
```

Parameters

lHandle

[IN] 操作句柄

s32BufNum

[IN] 所要设置的单视频播放时缓冲区最大的帧数，取值范围（高清[0-20]，普通[0-50]），SDK 默认的帧缓冲区大小为 0

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remarks

设置网络延时和播放流畅度可以通过此接口来进行调节。s32RBNum 值越大，播放的流畅性越好，相对延时就大；s32RBNum 值越小，播放的延时就小，但是当网络不太顺畅的时候，会有丢帧现象，影响播放的流畅性。

1.6 云台控制

摄像机是否支持云台属性，可以通过获取 HI_GET_PRODUCT_VENDOR 中 sProduct 的 Z 字段判断，具体请参阅附录厂家代码和设备类型定义章节。

HI_SDK_PTZControl

云台控制操作，含有 Z0 字段的设备不支持。

```
HI_S32   HI_SDK_PTZControl (
        HI_HANDLE       lHandle,
        HI_U32          u32Command,
        HI_U32          u32Speed
);
```

Parameters

lHandle

[IN] 操作句柄

u32Command

[IN] 云台控制命令

宏定义	宏定义值	含义
HI_CTRL_PTZ_STOP	0x3000	停止云台
HI_CTRL_PTZ_UP	0x3001	云台上仰
HI_CTRL_PTZ_DOWN	0x3002	云台下俯
HI_CTRL_PTZ_LEFT	0x3003	云台左转
HI_CTRL_PTZ_RIGHT	0x3004	云台右转
HI_CTRL_PTZ_ZOOMIN	0x3005	焦距变大(倍率变大)
HI_CTRL_PTZ_ZOOMOUT	0x3006	焦距变小(倍率变小)
HI_CTRL_PTZ_FOCUSIN	0x3007	焦点前调
HI_CTRL_PTZ_FOCUSOUT	0x3008	焦点后调
HI_CTRL_PTZ_APERTUREIN	0x3009	光圈变小
HI_CTRL_PTZ_APERTUREOUT	0x3010	光圈变大
HI_CTRL_PTZ_LIGHT_ON	0x3021	灯光开
HI_CTRL_PTZ_LIGHT_OFF	0x3022	灯光关
HI_CTRL_PTZ_WIPER_ON	0x3023	雨刷开
HI_CTRL_PTZ_WIPER_OFF	0x3024	雨刷关
HI_CTRL_PTZ_AUTO_ON	0x3025	自动开
HI_CTRL_PTZ_AUTO_OFF	0x3026	自动关
HI_CTRL_PTZ_HOME	0x3027	回到原点
HI_CTRL_PTZ_CRUISE_V	0x3028	上下巡航
HI_CTRL_PTZ_CRUISE_H	0x3029	左右巡航

u32Speed

[IN] 速度

#define HI_CTRL_PTZ_SPEED_MAX 0x3F //最大速度

#define HI_CTRL_PTZ_SPEED_MIN 0x00 //最小速度

Return Values

云台控制发送命令无返回值。

Remarks

通过厂商 ID 的 Z 字段判断是否支持该属性。HI_S_ProductVendor 中 sProduct 值。

HI_SDK_PTZControlEx

云台控制操作扩展，单步执行。

```
HI_S32   HI_SDK_PTZControlEx (
    HI_HANDLE       lHandle,
    HI_U32          u32Command,
);
```

Parameters

lHandle

[IN] 操作句柄

u32Command

[IN] 云台控制命令

宏定义	宏定义值	含义
HI_CTRL_PTZ_STOP	0x3000	停止云台
HI_CTRL_PTZ_UP	0x3001	云台上仰
HI_CTRL_PTZ_DOWN	0x3002	云台下俯
HI_CTRL_PTZ_LEFT	0x3003	云台左转
HI_CTRL_PTZ_RIGHT	0x3004	云台右转
HI_CTRL_PTZ_ZOOMIN	0x3005	焦距变大(倍率变大)
HI_CTRL_PTZ_ZOOMOUT	0x3006	焦距变小(倍率变小)
HI_CTRL_PTZ_FOCUSIN	0x3007	焦点前调
HI_CTRL_PTZ_FOCUSOUT	0x3008	焦点后调
HI_CTRL_PTZ_APERTUREIN	0x3009	光圈变小
HI_CTRL_PTZ_APERTUREOUT	0x3010	光圈变大

Return Values

云台控制发送命令无返回值。

Remarks

云台控制扩展用于单步执行单步移动。

HI_SDK_PTZPreset

云台预置点操作

```
HI_S32   HI_SDK_PTZPreset (
        HI_HANDLE       lHandle,
        HI_U32          u32Command,
        HI_U32          u32Preset
);
```

Parameters

lHandle

[IN] 操作句柄

u32Command

[IN] 云台预置点控制命令

宏定义	宏定义值	含义
HI_CTRL_PTZ_GOTO_PRESET	0x3015	转到预置点
HI_CTRL_PTZ_SET_PRESET	0x3016	设置预置点
HI_CTRL_PTZ_CLE_PRESET	0x3017	清除预置点

u32Preset

[IN] 预置点

```
#define HI_CTRL_PTZ_PRESET_MAX     255
#define HI_CTRL_PTZ_PRESET_MIN     0
```

Return Values

云台控制发送命令无返回值。

Remarks

通过厂商 ID 的 Z 字段判断是否支持该属性。HI_S_ProductVendor 中 sProduct 值。

HI_SDK_TransPTZ

透明云台操作

```
HI_S32   HI_SDK_TransPTZ (
        HI_HANDLE       lHandle,
        HI_CHAR*        psBuf,
        HI_U32          u32BufLen
);
```

Parameters

lHandle

[IN] 操作句柄

psBuf

[IN] 控制云台命令数据，命令数据只能是 128 个字节组成的串，如 ff01100800041d。

u32BufLen

[IN] 云台控制码的长度，

```
#define HI_CTRL_PTZ_FT_BUF_LEN     128
```

Return Values

云台控制发送命令无返回值。

Remarks

透传函数通过 845 口控制云台，只发送数据不接收数据，不同云台控制设备透传码不相同，获取设备的透传码可查看设备相关说明书。

通过厂商 ID 的 Z 字段判断是否支持该属性。HI_S_ProductVendor 中 sProduct 值。

1.7 实时预览数据回调

HI_SDK_SetRealDataCallBack

注册码流数据回调，注册后 SDK 中将不解码显示

```
HI_S32   HI_SDK_SetRealDataCallBack (
        HI_HANDLE           lHandle,
        HI_U32              u32Chn,
        OnRealDataCallBack  streamCallBack,
        HI_VOID*            pUserData
);
```

Parameters

lHandle

[IN] 操作句柄

u32Chn

[IN] 整形参数

streamCallBack

[IN] 码流数据回调函数

pUserData

[IN] 用户数据

Callback Function

```
typedef   HI_S32   (*OnRealDataCallBack)(
                HI_U32          u32Chn,
                MEDIA_TYPE_E    eStreamType,
                HI_VOID*        pStreamData,
                HI_S32          s32DataNum,
                HI_U32          s32Pts,
                HI_S32          s32KeyFrame,
                HI_VOID*        pUserData
                );
```

Callback Function Parameters

u32Chn

整形参数

eStreamType

数据类型，音视频数据或头文件数据

宏定义	宏定义值	含义
MEDIA_VIDEO	1	视频数据
MEDIA_AUDIO	2	音频数据
MEDIA_SYSTERM	4	系统头

pStreamData

数据包含帧头

s32DataNum

数据长度

s32Pts

时间戳

s32KeyFrame

视频关键帧 1-I 帧，0-P 帧

pUserData

用户数据

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remarks

eStreamType 类型为 **MEDIA_SYSTERM**，pu8Buffer 的结构是由 HI_S_SysHeader 结构组成：

```
typedef struct
{
    HI_U32    u32Width;          //视频宽
    HI_U32    u32Height;         //视频高
} HI_S_VideoHeader;

typedef struct
{
     HI_U32   u32Format;         //音频格式
} HI_S_AudioHeader;
```

宏定义	宏定义值	含义
HI_AUDIO_TYPE_G711	0	G711
HI_AUDIO_TYPE_G726	1	G726
HI_AUDIO_TYPE_AMR	2	AMR

音频采集：8K，16 位，单声道

```
typedef struct
{
    HI_U32              u32SysFlag;
    HI_S_VideoHeader    struVHeader;
    HI_S_AudioHeader    struAHeader;
} HI_S_SysHeader;
```

其中 u32SysFlag 为宏定义#define HI_SYS_FLAG 0x53565848。

eStreamType 类型为 **MEDIA_VIDEO**，pu8Buffer 是 H264 的码流；

eStreamType 类型为 **MEDIA_AUDIO**，pu8Buffer 是音频数据，如果用非海思解码库，要将头部 4 字节的去掉。

HI_SDK_SetDecCallBack

注册解码数据回调

```
HI_S32  HI_SDK_SetDecCallBack (
    HI_HANDLE           lHandle,
    HI_U32              u32Chn
    OnDecCallBack       CallBack,
    HI_VOID*            pUserData
);
```

Parameters

lHandle

[IN] 操作句柄

u32Chn

[IN] 整形参数

CallBack

[IN] 解码数据回调函数

pUserData

[IN] 用户数据

Callback Function

```
typedef  LONG  (*OnDecCallBack)(
         HI_U32                  u32Chn,
         const FRAME_INFO_S *pFrameInfo,
         HI_VOID *pUserData
         );
```

Callback Function Parameters

u32Chn

整形参数

pFrameInfo

帧类型

```
typedef struct hiFRAME_INFO_S
{
    HI_U8* pY;        //解码后视频数据 Y  分量
    HI_U8* pU;        //解码后视频数据 U 分量
    HI_U8* pV;        //解码后视频数据 V 分量
    long nWidth;      //视频宽
    long nHeight;     //视频高
```

```
        long nType;          //data type:YUV420
        long nYPich;
        long nUVPich;
        HI_U64 u64Pts;
    }
    FRAME_INFO_S;
```

pData

解码数据回调

pUserData

用户数据

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_SetMessageCallBack

注册报警信息数据回调

```
HI_S32   HI_SDK_SetMessageCallBack (
    HI_HANDLE           lHandle,
    HI_U32              u32Chn
    OnMessageCallBack   CallBack,
    HI_VOID *           pUserData
);
```

Parameters

lHandle

[IN] 操作句柄

u32Chn

[IN] 整形参数

CallBack

[IN] 报警信息数据回调函数

pUserData

[IN] 用户数据

Callback Function

```
typedef   LONG   (*OnMessageCallBack)(
        HI_U32          u32Chn,
        MD_TYPE_E       eDataType,
        HI_U8*          pu8Buffer,
        HI_U32          u32Length,
        HI_VOID*        pUserData
        );
```

Callback Function Parameters

u32Chn

整形参数

eDataType

数据类型

宏定义	宏定义值	含义
HI_MOTION_DETECTION	0	移动侦测报警
HI_INPUT_ALARM	1	输入报警
HI_KEEP_ALIVE	2	心跳包

pu8Buffer

数据。如果为 HI_MOTION_DETECTION，数据将以 HI_S_ALARM_MD 结构存储：

```
typedef struct
{
    HI_U32      u32Area;        //区域
    HI_U32      u32X;           //x 坐标
    HI_U32      u32Y;           //y 坐标
    HI_U32      u32Width;       //矩形宽
    HI_U32      u32Height;      //矩形高
} HI_S_ALARM_MD;
```

u32Area 最大为 4，数据如下：

宏定义	宏定义值	含义
HI_MOTION_AREA_1	1	区域 1
HI_MOTION_AREA_2	2	区域 2
HI_MOTION_AREA_3	3	区域 3
HI_MOTION_AREA_4	4	区域 4

u32Length

数据长度，HI_MOTION_DETECTION，两个区域同时就有：

u32Length = 2*sizeof(HI_S_ALARM_MD)

u32DataType

用户数据

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_SetEventCallBack

事件数据回调

```
HI_S32   HI_SDK_SetEventCallBack (
    HI_HANDLE           lHandle,
    HI_U32              u32Chn,
    OnEventCallBack     eventCallBack,
    HI_VOID*            pUserData
);
```

Parameters

lHandle

[IN] 操作句柄

u32Chn

[IN] 整形参数

eventCallBack

[IN] 事件数据回调函数

pUserData

[IN] 用户数据

Callback Function

```
typedef LONG (*OnEventCallBack) (
        HI_U32          u32Chn,
        EVENT_TYPE_E    eEventType,
        HI_VOID*        pEventData,
        HI_S32          s32DataNum,
        HI_VOID*        pUserData
        );
```

Callback Function Parameters

u32Chn

整形参数

eEventType

事件类型

宏定义	宏定义值	含义
EVENT_LIVE_STOP	0	停止实时预览
EVENT_LIVE_PAUSE	1	暂停实时预览
EVENT_LIVE_PLAY	2	实时预览
EVENT_TALK_STOP	3	停止对讲
EVENT_TALK_PLAY	4	开始对讲
EVENT_TALK_ABNORM	5	对讲异常
EVENT_REC_STOP	6	停止录像
EVENT_REC_PLAY	7	开始录像
EVENT_REC_ABNORM	8	录像异常
EVENT_PLAYBACK_READ	9	回放就绪
EVENT_PLAYBACK_PLAY	10	开始回放
EVENT_PLAYBACK_PAUSE	11	暂停回放
EVENT_PLAYBACK_STOP	12	停止回放
EVENT_NET_CONNECTING	13	正在连接
EVENT_NET_CONNECTED	14	连接成功
EVENT_NET_DISCONNECT	15	连接失败
EVENT_NET_ABNORMAL	16	异常断开
EVENT_NET_RECONNECT	17	重新连接
EVENT_NET_CONNECTFAIL	18	连接失败
EVENT_REALDATA_STOP	19	捕获实时数据
EVENT_REALDATA_PLAY	20	停止捕获数据

pEventData

事件数据

s32DataNum

事件数据长度

pUserData

用户数据

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remarks

事件回调与网络线程在一个线程当中，如果处理窗口消息可以用 WINDOWS 窗口消息机制，如 POSTMESSAGE 等；其他可以用线程处理。

1.8 预览声音控制

HI_SDK_SetVolume

设置音量大小

```
HI_S32    HI_SDK_SetVolume (
        HI_HANDLE           lHandle,
        AUDIO_DIRECT_E      eDir,
        HI_S32              s32Volume
);
```

Parameters

u32Handle

[IN] 操作句柄

eDir

[IN] AUDIO_OUT 输出音频，AUDIO_IN 为输入音频(MIC)

s32Volume

[IN] 音频大小，范围[0,100]

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_GetVolume

获取当前音量大小

```
HI_S32    HI_SDK_GetVolume (
        HI_HANDLE           lHandle,
        AUDIO_DIRECT_E      eDir,
        HI_S32*             pVolume
);
```

Parameters

u32Handle

[IN] 操作句柄

eDir

[IN] AUDIO_OUT 输出音频，AUDIO_IN 为输入音频(MIC)

pVolume

[OUT] 音频大小，范围[0,100]

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_SetMute

设置静音/监听模式

```
HI_S32    HI_SDK_SetMute (
        HI_HANDLE           lHandle,
        AUDIO_DIRECT_E      eDir,
        AUDIO_MUTE_E        eMute
);
```

Parameters

u32Handle

[IN] 操作句柄

eDir

[IN] AUDIO_OUT 输出音频，AUDIO_IN 为输入音频(MIC)

eMute

[IN] AUDIO_MUTE_ON 静音状态，AUDIO_MUTE_OFF 监听状态

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_GetMute

得到静音/监听状态

```
HI_S32    HI_SDK_GetMute (
        HI_HANDLE           lHandle,
        AUDIO_DIRECT_E      eDir,
        AUDIO_MUTE_E*       pMute
);
```

Parameters

u32Handle

[IN] 操作句柄

eDir

[IN] AUDIO_OUT 输出音频，AUDIO_IN 为输入音频(MIC)

pMute

[OUT] AUDIO_MUTE_ON 静音状态，AUDIO_MUTE_OFF 监听状态

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

1.9 录像

HI_SDK_StartRecord

开始录像，录像支持两种格式：ASF 和自定义复合流录像，通过接口参数 eFileFormat 控制录像类型。

```
HI_S32   HI_SDK_StartRecord (
    HI_HANDLE          lHandle,
    HI_CHAR *          pFilePath,
    FILE_FORMAT_E      eFileFormat,
    MEDIA_TYPE_E       eFlag,
    HI_S32             s32FileTime
);
```

Parameters

lHandle

[IN] 操作句柄

pFilePath

[IN] 录像文件路径

eFileFormat

[IN] 文件格式，目前支持 AVI(FILE_FORMAT_AVI) 录像格式、SF(FILE_FORMAT_ASF)录像格式和复合流 (FILE_FORMAT_NUDE_STREAM) 录像格式。

eFlag

[IN] 录像形式，音频、视频、音视频，参考枚举 MEDIA_TYPE_E

s32FileTime

[IN] 录像时间长度，单位是秒，默认是 0，0 表示无限制。

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remarks

复合流录像：抓取实时数据，按顺序保存到文件中，文件格式前面部分包含了一个 HI_S_SysHeader 结构体的文件头，紧接着是 HI_S_AVFrame 结构体，保存了数据块的大小、类型等信息，然后就是数据块，通过 HI_S_AVFrame 中长度值定义数据块大小。结构如下：

HI_S_SysHeader

HI_S_AVFrame

数据块

HI_S_AVFrame

数据块

.....

HI_S_AVFrame

数据块
HI_S_AVFrame
数据块

保存后的数据可以用 SDK 中的函数 HI_SDK_Playback 或者播放库提供的 HI_PLAYER_OpenFile 接口播放。

HI_SDK_StopRecord

停止录像

```
HI_S32    HI_SDK_StopRecord (
    HI_HANDLE            lHandle
);
```

Parameters

lHandle
[IN] 操作句柄

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

1.10 抓拍

HI_SDK_CapturePicture

抓拍 BMP 图，包括实时预览和文件回放

```
HI_S32    HI_SDK_CapturePicture (
    HI_U32        u32Handle,
    HI_CHAR*    pszFilePath
);
```

Parameters

u32Handle
[IN] 操作句柄
pszFilePath
[IN] 抓拍路径

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_CaptureJPEGPicture

抓拍 JPG 图，包括实时预览和文件回放

```
HI_S32    HI_SDK_CaptureJPEGPicture (
    HI_HANDLE        lHandle,
    HI_CHAR*         sFilePath
```

```
);
```

Parameters

lHandle

[IN] 操作句柄

sFilePath

[IN] 抓拍路径

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_SnapYUVData

抓解码后的YUV 数据

```
HI_S32 HI_SDK_SnapYUVData(
        HI_HANDLE lHandle,
        HI_YUV_INFO_S* pYUVInfo,
        HI_S32 nYUVType,
        HI_S32 *pNeedSize
        );
```

Parameters

lHandle

[IN] 操作句柄

pYUVInfo

[IN][OUT] 存放ＹＵＶ数据。

nYUVType

[ＩＮ] 指定抓取的 YUV 类型，扩展用，恒为 0,目前只支持 YUV420 格式。

pNeedSize

[OUT] 存放ＹＵＶ所需内存的大小，不能为空。

Return Values

成功返回 HI_SUCCESS，失败返回 HI_FAILURE。

如果 pYUVInfo 为空，函数返回 HI_FAILURE,pNeedSize 返回 YUV 数据所需内存的大小。

具体用法如下：

```
Char * filename = ".....".;
HI_S32 nNeedSize = 0;
HI_YUV_INFO_S yuv;
memset(&yuv,0,sizeof(HI_YUV_INFO_S));
yuv.nDataLen = 1920*1080 * 3/2;
yuv.pData = (HI_U8*)malloc(yuv.nDataLen);
HI_S32 s32Ret = HI_FAILURE;
if(yuv.pData)
```

```
{
    if(HI_SDK_SnapYUVData(m_sCamInfo[m_u32CurScr].lHandle,&yuv,0,&nNeedSize)
== HI_SUCCESS)
    {
        //yuv,do some thing here
        Save_File(filename,yuv.pData,nNeedSize);
    }

    free(yuv.pData);
}
```

HI_SDK_SnapJpeg

网络抓拍

```
HI_S32   HI_SDK_SnapJpeg (
    HI_HANDLE       lHandle,
    HI_U8*          pu8Data,
    HI_S32          s32BufLen,
    HI_S32          *pSize
);
```

Parameters

lHandle

[IN] 操作句柄

pu8Data

[IN] 内存数据，JPG 格式

s32BufLen

[IN] 申请内存数据的长度，不能小于 **1024** 字节

pSize

[IN] 返回数据大小

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

Remarks

网络抓拍每秒抓拍最多抓取 2 张图片。

网络抓拍实现抓取网络图像，保存 JPG 格式的数据到内存中，接口再登录(HI_SDK_Login)成功后即可使用，申请的内存在外部进行，申请内存大小不能小于：

#define HI_SDK_SNAP_BUF_LEN_MIN 1024

具体用法如下：

```
char *sData = (char*)malloc(1024*1024);
int nSize = 0;
s32Ret = HI_SDK_SnapJpeg(m_lHandle, (HI_U8*)sData, 1024*1024, &nSize);
if(s32Ret == HI_SUCCESS)
```

```
{
    FILE *fp = fopen("D:\\photo.jpg", "wb+");
    if( !fp )
        free(sData);

    fwrite((const char*)sData, 1, nSize, fp);
    fclose( fp );
}
free(sData);
sData = NULL;
```

1.11 图像叠加显示

HI_SDK_InputDrawData

添加要叠加的图像信息、类型

```
HI_S32   HI_SDK_InputDrawData (
    HI_HANDLE          lHandle,
    DRAW_INFO_S*       pstrDrawData,
    HI_S32             s32StrSize,
    HI_S32             s32DrawState
);
```

Parameters

lHandle

[IN] 操作句柄

pstrDrawData

[IN] 信息缓冲

s32StrSize

[IN] 信息缓冲大小

s32DrawState

[IN] 显示类型，DRAW_STATE 和 EVENT_STATE 两种类型

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_ClearDrawData

清除指定的叠加图像信息

```
HI_S32   HI_SDK_ClearDrawData (
    HI_HANDLE      lHandle,
    HI_CHAR*       pDrawData,
    HI_S32         s32DrawState
);
```

Parameters

lHandle
[IN] 操作句柄
pDrawData
[IN] 信息缓冲
s32DrawState
[IN] 显示类型，DRAW_STATE 和 EVENT_STATE 两种类型

Return Values
成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_SelectPic

设置鼠标所在位置为焦点，该函数调用 DRAW 回调处理叠加的图像

```
HI_S32   HI_SDK_SelectPic (
    HI_HANDLE       lHandle,
    CPoint          point
);
```

Parameters
lHandle
[IN] 操作句柄
point
[IN] 当前鼠标所在的坐标

Return Values
成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_MouseMove

鼠标移动时调用该函数，函数调用 DRAW 回调更新 MD 区域坐标

```
HI_S32   HI_SDK_MouseMove (
    HI_HANDLE       lHandle,
    UINT            nFlags,
    CPoint          point,
    CRect           rcRect
);
```

Parameters
lHandle
[IN] 操作句柄
nFlags
[IN] 按键标记
point
[IN] 当前鼠标所在的坐标
rcRect

[IN] 窗口坐标

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_SetDrawCallBack

注册绘图回调，当鼠标移动对 MD 坐标信息修改时，调用该回调函数更新 MD 属性

```
HI_S32    HI_SDK_SetDrawCallBack (
        HI_HANDLE           lHandle,
        HI_U32              u32Chn,
        OnDrawCallBack      callBack,
        HI_VOID*            pUserData
);
```

Parameters

lHandle

[IN] 操作句柄

u32Chn

[IN] 整形参数

OnDrawCallBack

[IN] 事件数据回调函数

pUserData

[IN] 用户数据

Callback Function

```
typedef    LONG    (*OnDrawCallBack) (
                HI_U32          u32Chn,
                RECT            rcDrawRect,
                HI_CHAR*        pszName,
                HI_VOID*        pUserData
                );
```

Callback Function Parameters

u32Chn

整形参数

rcDrawRect

图标新坐标

pszName

图标的名称

pUserData

用户数据

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_EnablePic

显示隐藏叠加图像

```
HI_S32    HI_SDK_EnablePic (
        HI_HANDLE       lHandle,
        HI_CHAR*        pszName,
        HI_S32          s32EnableValue,
        HI_S32          s32DrawState
);
```

Parameters

lHandle

[IN] 操作句柄

pszName

[IN] 名称

s32EnableValue

[IN] 显示隐藏，0 为隐藏，1 为显示

s32DrawState

[IN] 显示类型，DRAW_STATE 和 EVENT_STATE 两种类型

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_GetPicInfo

获取图像高宽

```
HI_S32    HI_SDK_GetPicInfo (
        HI_HANDLE       lHandle,
        HI_S32*         pHeight,
        HI_S32*         pWidth
);
```

Parameters

lHandle

[IN] 操作句柄

pHeight

[OUT] 高度

pWidth

[OUT] 宽度

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_SetPostDrawCallBackEx

注册自绘回调

```
HI_SDK_SetPostDrawCallBackEx(
    HI_HANDLE                   lHandle,
    HI_U32                      32Chn,
    HISDK_PostDrawCallBackEx    callback,
    HI_VOID *                   pUserData
    );
```

Parameters

lHandle

[IN] 操作句柄

u32Chn

[IN] 整形参数

callback

[IN] 自绘回调

pUserData

[IN] 用户数据，默认为 NULL

Callback Function

```
typedef HRESULT (HISDK_CALLBACK *HISDK_PostDrawCallBackEx)(
    HI_VOID *       lRes,
    HI_VOID *       hDc,
    HI_S32          s32ImageWidth,
    HI_S32          s32ImageHeight,
    HI_S32          s32WndWidth,
    HI_S32          s32WndHeight,
    HI_S32          s32Offx,
    HI_S32          s32Offy,
    HI_U64          u64TimeStamp,
    HI_VOID *       pPara
    );
```

Callback Function Parameters

lRes

SDK 内部使用,保留.

hDc

自绘用 HDC 指针

s32ImageWidth

图像宽度

s32ImageHeight

图像高度

s32WndWidth

窗体宽度

s32WndHeight
窗体高度
s32Offx
水平偏移
s32Offy
垂直偏移
u64TimeStamp
时间戳
pUserData
用户数据

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remarks

如果 Callback 不为 NULL，SDK 将不向摄像机端发送音频数据，发送音频数据可以通

1.12 语音对讲转发

HI_SDK_StartVoiceCom

打开对讲

```
HI_S32    HI_SDK_StartVoiceCom (
        HI_HANDLE              lHandle,
        HI_U32                 u32Chn,
        OnVoiceDataCallBack    callback,
        HI_VOID *              pUserData
);
```

Parameters

lHandle
[IN] 操作句柄
u32Chn
[IN] 整形参数
Callback
[IN] 语音回调，默认为 NULL
pUserData
[IN] 用户数据，默认为 NULL

Callback Function

```
typedef    LONG    (*OnVoiceDataCallBack) (
                HI_U32 u32Chn,
                HI_U8* pBuf,
                HI_S32 s32Size,
                HI_U32 u32TimeStamp,
                HI_VOID *pUserData
```

```
);
```

Callback Function Parameters

u32Chn

整形参数

pBuf

音频数据

s32Size

音频数据大小

u32TimeStamp

时间戳

pUserData

用户数据

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remarks

如果 Callback 不为 NULL，SDK 将不向摄像机端发送音频数据，发送音频数据可以通过 HI_SDK_VoiceComSendData 函数发送

HI_SDK_StopVoiceCom

关闭对讲

```
HI_S32    HI_SDK_StopVoiceCom (
    HI_HANDLE       lHandle,
);
```

Parameters

lHandle

[IN] 操作句柄

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_VoiceComSendData

将采集回来的数据发送给对方

```
HI_S32    HI_SDK_VoiceComSendData (
    HI_HANDLE       lHandle,
    HI_CHAR*        psBuf,
    HI_U32          u32BufLen,
    HI_U64          u64Pts
);
```

Parameters

lHandle

[IN] 操作句柄

psBuf

[IN] 发送数据

u32BufLen

[IN] 数据长度

U64Pts

[IN] 时间戳

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remarks

对讲采集回来的数据要求是 8K，16 位，单声道 G726 压缩数据，具体用法请参阅 Demo 中的用法。

非海思解码库编码的音频需要在每个音频包前加入 4 个字节

G726：0x00 0x01 0x14 0x00

G711：0x00 0x01 0x50 0x00

1.13 录像回放

HI_SDK_Playback

回放

```
HI_HANDLE    HI_SDK_Playback (
        HI_CHAR*          psFilePath,
        HI_VOID*          pWnd
);
```

Parameters

psFilePath

[IN] 文件路径

pWnd

[IN] 回放窗口句柄

Return Values

成功将返回回放操作句柄 HI_HANDLE，失败将返回 0。

HI_SDK_StopPlayback

关闭回放

```
HI_S32    HI_SDK_StopPlayback (
        HI_HANDLE        lPlayHandle
);
```

Parameters

lPlayHandle

[IN] HI_SDK_Playback 返回的操作句柄

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_PlayBackControl

控制回放

```
HI_S32    HI_SDK_PlayBackControl (
        HI_HANDLE           lPlayHandle,
        PBCTRL_TYPE_E       s32Command,
        HI_S32              s32Value,
        HI_S32              *s32OutValue
);
```

Parameters

lPlayHandle

[IN] HI_SDK_Playback 返回的操作句柄

s32Command

[IN] 命令操作

定义	定义值	含义
PB_CTRL_PLAY	0	播放
PB_CTRL_STOP	1	停止
PB_CTRL_PAUSE	2	暂停
PB_CTRL_RATE	3	调整速度
PB_CTRL_FRAME	4	单帧
PB_CTRL_SETPOS	5	设置播放位置
PB_CTRL_GETPOS	6	获取播放位置
PB_CTRL_MUTE	7	静音/监听
PB_CTRL_VOLUME	8	设置音量
PB_CTRL_GETTIME	9	获取播放时间

s32Value

[IN] 设置操作值

s32OutValue

[OUT] 获取操作得到的值

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

1.14 解码操作

HI_SDK_PauseDecode

暂停解码，视频不显示

```
HI_S32    HI_SDK_PauseDecode (
        HI_HANDLE        lHandle
);
```

Parameters

lHandle

[IN] 操作句柄

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_ResumeDecode

恢复解码，恢复从 I 帧开始解码

```
HI_S32    HI_SDK_ResumeDecode (
        HI_HANDLE        lHandle
);
```

Parameters

lHandle

[IN] 操作句柄

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

1.15 设置操作通道

用户登录后，在不用请求音视频流的情况下，可以设置设备端的参数，默认设置的是第一通道。如果前端设备是多通道的设备（如 NVR），可以调用 HI_SDK_SetChannel 设置为当前通道，即可对通道进行参数设置，云台控制等操作。

HI_SDK_SetChannel

设置当前操作通道

```
HI_S32    HI_SDK_SetChannel (
        HI_U32          u32Handle,
        HI_U32          u32Channel
);
```

Parameters

u32Handle

[IN] 操作句柄

u32Channel

[IN] 通道+码流，**通道从 1 开始**，格式：**通道*10 + 1 或 通道*10 + 2**，1 代表主码流，2 代表次码流，如 11 即第一通道主码流，92 第九通道次码流

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

HI_SDK_GetChannel

获取当前操作通道

```
HI_S32    HI_SDK_GetChannel (
    HI_U32          u32Handle
);
```

Parameters

u32Handle

[IN] 操作句柄

Return Values

返回值返回的是通道，**通道从 1 开始**，格式：**通道*10 + 1 或 通道*10 + 2**，1 代表主码流，2 代表次码流，如 11 即第一通道主码流，92 第九通道次码流

1.16 其他

HI_SDK_GetSDKVersion

获取 SDK 版本号

```
HI_S32    HI_SDK_GetSDKVersion (
    HI_CHAR*    pVersion
);
```

Parameters

pVersion

[OUT] SDK 版本号

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_GetPlayRate

获取预览播放平均码流帧率

```
HI_S32    HI_SDK_GetPlayRate (
    HI_HANDLE       lHandle,
    HI_S32          *pFrameRate,
    HI_S32          *pBitRate
);
```

Parameters

lHandle

[IN] 操作句柄

pFrameRate

[OUT] 帧率

pBitRate

[OUT] 码率

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_GetState

获取播放、对讲、录像状态

```
HI_S32    HI_SDK_GetState (
        HI_HANDLE       lHandle,
        STATE_ID_E      eStateID,
        HI_S32 *        pState
);
```

Parameters

lHandle

[IN] 操作句柄

eStateID

[IN] 类型

```
typedef enum hiSTATE_ID_E
{
    STATE_ID_PLAY = 0,                  //文件或流播放标志
    STATE_ID_REC,                       //录像标志
    STATE_ID_TALK,                      //对讲标志
    STATE_ID_SERVER_USERNUM,            //用户连接数
    STATE_ID_BUTT
} STATE_ID_E;
```

pState

[OUT] 状态

STATE_ID_E 对应的枚举如下：

1、STATE_ID_PLAY

```
typedef enum hiPLAY_STATE_E
{
    PLAY_STATE_PAUSE = 0,           //暂停
    PLAY_STATE_PLAY,                //播放
    PLAY_STATE_AUDIO,               //音频
    PLAY_STATE_VIDEO,               //视频
    PLAY_STATE_STOP,                //停止
    PLAY_STATE_BUTT
```

```
} PLAY_STATE_E;
2、STATE_ID_REC
typedef enum hiREC_STATE_E
{
    REC_STATE_RUN    = 0,            //正在录像
    REC_STATE_STOP,                  //停止录像
    REC_STATE_BUTT
} REC_STATE_E;
3、STATE_ID_TALK
typedef enum hiTALK_STATE_E
{
    TALK_STATE_RUN = 0,              //正在对讲
    TALK_STATE_STOP,                 //停止对讲
    TALK_STATE_BUTT
} TALK_STATE_E;
```

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_GetPlayerHandle

获取播放器句柄，包括实时预览和文件回放

```
HI_S32    HI_SDK_GetPlayerHandle (
    HI_HANDLE       lHandle,
    HI_VOID**       ppPlayerHandle
);
```

Parameters

lHandle

[IN] 操作句柄

ppPlayerHandle

[OUT] 播放库句柄

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_SetDrawWnd

改变播放时显示的窗口

```
HI_S32    HI_SDK_SetDrawWnd (
    HI_HANDLE       lHandle,
    HI_VOID*        pWnd
);
```

Parameters

lHandle

[IN] 操作句柄

pWnd

[IN] 窗口句柄

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remark

如果在播放时要将播放窗口换到另一个播放窗口，可以直接用改接口，只要将要显示的窗口句柄与相应的操作句柄关联起来即可。如果 pWnd 为空时，DDRAW 将销毁，即将不对视频进行显示；只有再次设置 pWnd 为非空时，才能再次显示。

HI_SDK_GetSupportAttr

获取摄像机支持属性

```
HI_S32    HI_SDK_GetSupportAttr (
    HI_HANDLE       lHandle,
    HI_S_SUPPORT*  pSupport
);
```

Parameters

lHandle

[IN] 操作句柄

pSupport

[OUT] HI_S_SUPPORT 结构体

```
typedef struct tagHI_SUPPORT
{
    HI_U32 u32Operation;        //操作属性，如夜视效果白平衡等等
    HI_U32 u32Reslution;        //主码流支持分辨率
    HI_U32 u32Reslution1;       //次码流支持分辨率
    HI_U32 u32FrameMax;         //最大帧数
    HI_U32 u32BitRateMin;       //主码流最小码率
    HI_U32 u32BitRateMax;       //主码流最大码率
    HI_U32 u32BitRateMin1;      //次码流最小码率
    HI_U32 u32BitRateMax1;      //次码流最大码率
}HI_S_SUPPORT;
```

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remarks

```
SUPPORTATTR_NIGHTVISION_SET_FLAG   (0x000000001<<1) //夜视
SUPPORTATTR_WHITEBALANCE_FLAG      (0x000000001<<3) //白平衡
SUPPORTATTR_FLIP_FLAG              (0x000000001<<4) //翻转
```

SUPPORTATTR_MIRROR_FLAG (0x000000001<<5) //镜像
SUPPORTATTR_BRIGHTNESS_FLAG (0x000000001<<6) //亮度
SUPPORTATTR_SATURATION_FLAG (0x000000001<<7) //饱和度
SUPPORTATTR_CONTRAST_FLAG (0x000000001<<8) //对比度
SUPPORTATTR_HUE_FLAG (0x000000001<<9) //色度
SUPPORTATTR_SUBSTREAM_FLAG (0x000000001<<10) //次码流
SUPPORTATTR_POWERFREQ_FLAG (0x000000001<<11) //频率

Example：

```
HI_S_SUPPORT sSupport;
HI_SDK_GetSupportAttr( lHandle, &sSupport );

if( sSupport.u32Operation |= SUPPORTATTR_SUBSTREAM_FLAG )
    //支持夜视效果
if( sSupport.u32Operation |= SUPPORTATTR_FLIP_FLAG )
    //支持翻转效果
if( sSupport.u32Operation |= SUPPORTATTR_POWERFREQ_FLAG )
    //支持频率设置
… …
if( sSupport.u32Reslution |= (0x00000001<<HI_RESOLUTION_VGA) )
    //主码流支持 VGA 分辨率
if( sSupport.u32Reslution |= (0x00000001<<HI_RESOLUTION_CIF) )
    //主码流支持 CIF 分辨率
```

HI_SDK_SetAutoAdjust

设置播放画面的显示比例

```
HI_S32    HI_SDK_SetAutoAdjust (
    HI_HANDLE        lHandle,
);
```

Parameters

lHandle

[IN] 操作句柄

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_GetAutoAdjust

获取播放画面的显示比例

```
HI_S32    HI_SDK_GetAutoAdjust (
    HI_HANDLE        lHandle,
);
```

Parameters

lHandle

[IN] 操作句柄

Return Values

HI_SUCCESS 表示当前显示为自动调节状态，HI_ FAILURE 表示非自动调节状态。

HI_SDK_GetMediaAttr

获取设置播放音视频属性参数

```
HI_S32    HI_SDK_GetMediaAttr (
        HI_HANDLE        lHandle,
        STREAM_ATTR_S *pStreamInfo
);
```

Parameters

lHandle

[IN] 操作句柄

pStreamInfo

[OUT] STREAM_ATTR_S 结构体

```
typedef struct tagPLAYERSDK_ATTR_VIDEO_STREAM_S
{
    PLAYERSDK_VIDEO_FORMAT_E eVEncode;    //视频格式
    long lHeight;              //video height
    long lWidth;               //video width
    long lBitRate;            //video bit rate
    long lFrameRate;          //video frame rate
}PLAYERSDK_ATTR_VIDEO_STREAM_S;
//audio attr
typedef struct tagPLAYERSDK_ATTR_AUDIO_S
{
    PLAYERSDK_AUDIO_FORMAT_E eAEncode; //audio encode format
    long lSamplesPerSec;              //audio's samples per second
    long lBitsPerSample;             //bits per sample
    long lBitRate;                    //audio's bit rate
    long lBlockAlign;                 //if block align
    long lChannels;                   //audio's channels
    long lFrameFlag;                  //audio's frame flag
    long length;                      //audio's size
    void *pReserved;
}PLAYERSDK_ATTR_AUDIO_S;
//frame image info

typedef struct hiSTREAM_ATTR_S
{
```

```
        PLAYERSDK_ATTR_VIDEO_STREAM_S struVAttr;
        PLAYERSDK_ATTR_AUDIO_S        struAAttr;
    } STREAM_ATTR_S;
```

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remarks

Example：

```
STREAM_ATTR_S struStreamInfo;
HI_SDK_GetMediaAttr(.lHandle, &struStreamInfo);
```

HI_SDK_DisplayAll

显示区域电子放大

```
HI_S32   HI_SDK_DisplayAll (
    HI_HANDLE       lHandle,
    HI_S32          s32Left,
    HI_S32          s32Top,
    HI_S32          s32Right,
    HI_S32          s32Bottom,
    HI_BOOL         bDisplayAll
);
```

Parameters

lHandle

[IN] 操作句柄

s32Left

[IN] 相对于显示屏幕的左上角坐标（x）

s32Top

[IN] 相对于显示屏幕的左上角坐标（y）

s32Right

[IN] 相对于显示屏幕的右下角坐标（x）

s32Bottom

[IN] 相对于显示屏幕的右下角坐标（y）

bDisplayAll

[IN] 是否显示整个图像，HI_TRUE—显示全部，HI_FALSE—使用区域放大功能
默认值 HI_TRUE，设置显示区域必须使用 HI_FALSE；

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remark

函数的功能在 SDK 内部显示动态电子放大，输入的坐标是相对窗口显示坐标。

HI_SDK_SetDisplayMode

设置显示模式

```
HI_S32    HI_SDK_SetDisplayMode (
        HI_HANDLE                   hHandle,
        PLAYER_DISPLAYMODE_E        eDisplayMode
);
```

Parameters

lHandle

[IN] 操作句柄

eDisplayMode

[IN] 显示模式

```
typedef enum hiPLAYER_DISPLAYMODE_E
{
    PLAYER_DPY_D3D = 0,     //D3D 模式
    PLAYER_DPY_DDRAW,       //Ddraw 模式
}PLAYER_DISPLAYMODE_E;
```

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_GetDisplayMode

获取当前显示模式

```
HI_S32    HI_SDK_GetDisplayMode (
        HI_HANDLE                   hHandle,
        PLAYER_DISPLAYMODE_E        *pDisplayMode
);
```

Parameters

lHandle

[IN] 操作句柄

pDisplayMode

[OUT] 显示模式

```
typedef enum hiPLAYER_DISPLAYMODE_E
{
    PLAYER_DPY_D3D = 0,     //D3D 模式
    PLAYER_DPY_DDRAW,       //Ddraw 模式
}PLAYER_DISPLAYMODE_E;
```

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_SDK_SetDisplayCallback

设置 D3D 显示模式下叠加标志

```
HI_S32    HI_SDK_SetDisplayCallback (
        HI_HANDLE        hHandle,
        HI_BOOL          bDrawCallback
);
```

Parameters

lHandle

[IN] 操作句柄

bDrawCallback

[IN] 启动 D3D 叠加标志

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remark

播放库为了适应在 WIN7 以及以上版本的需求，修改了默认显示方式，设置成 D3D 并保留了 Ddraw 模式。如果当前显示模式为 D3D 模式，需要从回调函数 HI_SDK_SetDrawCallBack 中调用画图句柄进行图像叠加(比如画报警框，叠加文字等)，需要将 D3D 模式打开，此函数功能为打开回调标志。

第二部分　OCX 控件接口

2.1、功能简介

客户端 OCX 提供图像实时预览、客户端抓拍、客户端录像、视频参数显示及设置、移动参数显示及设置、对讲、云台控制、本地回放等功能。
IE 主界面、图像参数配置和移动参数配置界面共用一个控件。其中移动参数配置界面将调用接口 SetUseMDPage。 可参考 IE 网页代码调用 OCX 接口。

使用方法：

使用控件必须注册，如果是从网页上下载的控件安装后就已经注册，如果不是要手工注册，如用命令 regsvr32+控件名称进行注册。

从网页上下载空间并安装控件，在 C:\WINDOWS\system32 目录下会有 WebClientPlus.ocx 空间以及有关的库文件。调用控件的方法是以组件形式插入到工程目录中（根据开发环境的不同，调用方法不同），这样就可以使用相关的接口。工程开发完成后，打包需要选择 OCX 控件自注册。

手动注册 OCX 控件：

1、在网页中调用 OCX 控件方法如下：

```
<SCRIPT type=text/JavaScript>
if (navigator.appName.indexOf("Microsoft Internet Explorer") != -1)
{
    document.open();
    document.write('<object
classid="clsid:42B182F9-3F08-484E-9913-07193A5D36A5"
codebase="WebClientPlus.ocx#version=3,0,1,1" id="DHiMPlayer" align="absbottom"
viewastext>');
    document.write('<p align="left" style="font-size:14px">');
    document.write('    <span id="t5">警告信息显示如下：
</span><br>');
    document.write(' <span id="t6">1. 您的电脑没有安装浏览视频控件。<br>2. 您已经安
```

```
装控件但版本是不最新，请重新安装控件。<br><br>　　请点击</span><a href="/web/ClientOCXPlus_Setup.exe" id="t7">下载控件</a>');
    document.write(' <span id="t8">然后点击</span> <b id="t9"> 运行 </b> <span id="t10">安装控件，重新刷新网页，浏览视频。</span></p>');
    document.write('<param name="_Version" value="65536"> <param name="_ExtentX" value="10954"> <param name="_ExtentY" value="6826">');
    document.write('<param name="_StockProps" value="0">');
    document.write('<embed src="65536" _version="65536" _extentx="10954" _extenty="6826" _stockprops="0" align="center" height="0" width="0"></object>');
    document.close();
}
</SCRIPT>
```

其中

clsid:42B182F9-3F08-484E-9913-07193A5D36A5 为 OCX 的 Clsid;

codebase="WebClientPlus.ocx 为 OCX 的名称;

version=3,0,1,1 OCX 版本号

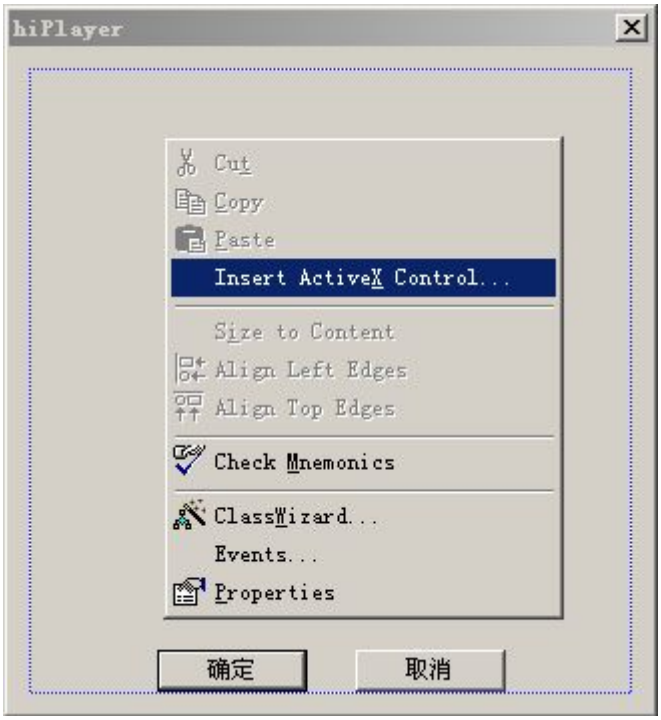
调用接口：

```
DHiMPlayer.SetUrl(url,80,streamnum,name0,password0);
DHiMPlayer.SetWndPos(0, 0, w, h);
DHiMPlayer.Play();
```

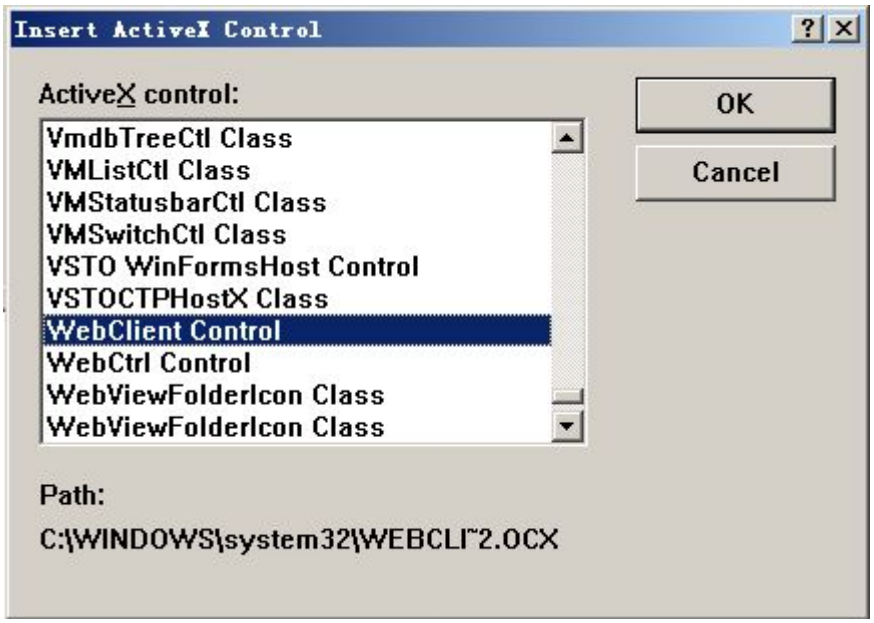
2、OCX 在开发环境中使用（以 VC++ 6.0 为例）

效果图：

1) 新建基于对话框的 mfc 工程，名称为 hiPlayer;



2) 右键对话框选择”Insert ActiveX Control…”，出现对话框 Insert ActiveX Control；



3) 选择已经注册好的 OCX 控件（必须注册 OCX），在对话框中将显示出 OCX 控件；

4) 为 OCX 控件添加成员变量 CwebClient m_hiPlayer；

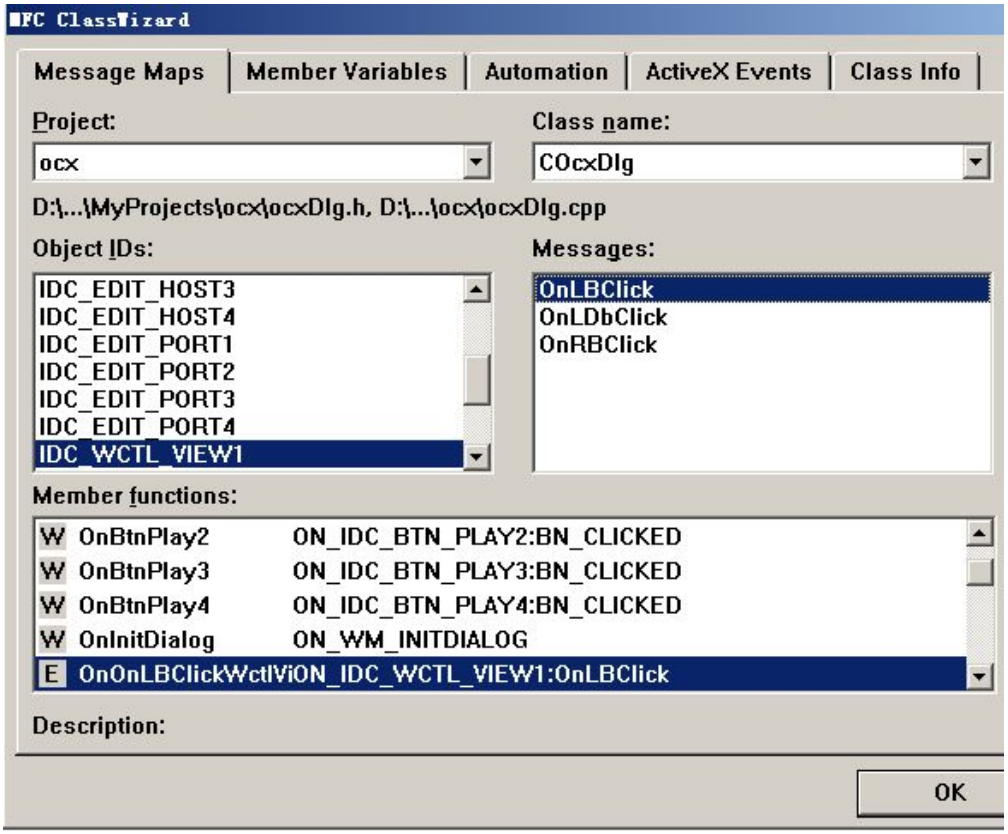
5) 在 OnOK 按钮消息中输入如下：

```
void CHiPlayerDlg::OnOK()
{
    m_hiPlayer.SetUrl("192.168.1.22", 80, 11, "admin", "admin");
    //第一通道主码流-11 第一通道次码流-12
    m_hiPlayer.Play();
    //CDialog::OnOK();
}
```

6） 编译即可运行。

注：不同的开发环境调用 OCX 控件的方法不同，具体就开发环境而定。

7） 点击控件消息，在 MFC ClassWizard 中选择控件，在 Messages:出现三个事件：OnLBClick（左击）、OnLDbClick（双击）和 OnRBClick（右击）三个事件，双击添加事件，在事件中添加代码。（版本在 3.0.2.2 以上才支持）



2.2、调用顺序

SetWndPos
SetUrl
Play

2.3、接口说明

2.3.1 设置播放窗口位置

设置播放窗口位置

```
long    SetWndPos (
        long        lLeft,
        long        lTop,
        long        lRight,
        long        lBottom
);
```

Parameters

lLeft

［IN］左侧坐标

lTop

［IN］顶部坐标

lRight

［IN］右侧坐标

lBottom

［IN］底部坐标

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

2.3.2 设置 URL

设置 URL

```
long    SetUrl (
        LPCTSTR     sHost,
        long        lPort,
        long        lChn,
        LPCTSTR     sUser,
        LPCTSTR     sPwd
);
```

Parameters

sHost

［IN］主机地址

lPort

［IN］端口号

lChn

［IN］通道码流（通道*10+码流，如：第一通道的主码流为 1*10+1=11；次码流 1*10+2=12），通道应用于转发服务器，如果非转发，通道为 1，即主码流为 11，次码流为 12

sUser

［IN］用户名

sPwd

［IN］密码

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

2.3.3 连接预览画面

连接预览画面

```
long    Play(
);
```

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

2.3.4 获取连接状态

获取连接状态

```
long    GetPlayState (
);
```

Return Values

返回 3 表示无音频视频，即没有连接，2 表示只有音频没有视频，1 表示只有视频没有音频。

2.3.5 停止预览

停止预览

```
long    Stop (
);
```

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

2.3.6 设置静音/监听

设置静音/监听

```
long    Mute (
);
```

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

2.3.7 获取音频状态

获取音频状态

```
BOOL    GetMuteState (
);
```

Return Values

HI_SUCCESS 表示静音，HI_ FAILURE 表示监听。

2.3.8 开始停止录像

开始停止录像

```
long    Record (
        long            lMode
```

```
);
```

开始录像，指定路径和文件名

```
long    RecordExt (
        LPCTSTR lpCstrFilename
);
```

停止录像

```
long    StopRecord (
);
```

Parameters

lMode

［IN］未使用

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

2.3.9 获取录像状态

获取音频状态

```
BOOL    GetRecState (
);
```

Return Values

HI_SUCCESS 表示正在录像，HI_ FAILURE 表示没有录像。

2.3.10 抓拍

抓拍，弹出保存对话框

```
long    Snapshot (
);
```

不弹出保存对话框

```
long    SnapshotEx (
);
```

自定路径和文件名

```
long    SnapshotExt (
        HI_U32 u32Type,
        LPCTSTR lpCstrFilename
);
```

u32Type：0 表示 JPG 格式，1 表示 BMP 格式

lpCstrFilename：路径+文件名

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

2.3.11 设置录像抓拍保存路径

设置录像抓拍保存路径，调用接口将出现选择目录对话框

```
long    SetRecordPath (
);//弹出选择路径窗口
```

设置录像抓拍保存路径，指定路径

```
long    SetRecordPathEx (
        LPCTSTR      lpStrPath
);//传递路径
```

获取录像抓拍保存路径

```
BSTR    GetRecordPath (
);
```

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

注：SetRecordPath 将弹出选择路径对话框，SetRecordPathEx（控件版本在 3.0.2.2 以上才支持）传递路径的方法。

2.3.12 打开关闭对讲

打开关闭对讲

```
long    Talk (
);
```

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

2.3.13 获取对讲状态

获取对讲状态

```
BOOL    GetTalkState (
);
```

Return Values

HI_SUCCESS 表示正在对讲，HI_ FAILURE 表示停止对讲。

2.3.14 打开播放器

打开播放器

```
long    PlayBack (
```

```
);
```

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

2.3.15 云台控制

云台控制

```
long    PtzControl (
        long        lType,
        long        lSpeed
);
```

Parameters

lType

［IN］操作类型

值	含义
0	停止云台
1	云台上仰
2	云台下俯
3	云台左转
4	云台右转
5	焦距变大(倍率变大)
6	焦距变小(倍率变小)
7	灯光开
8	灯光关
9	雨刷开
10	雨刷关
11	自动开
12	自动关
13	焦点前调
14	焦点后调
15	光圈变大
16	光圈变小

lSpeed

［IN］参数

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

2.3.16 云台预置点调用

云台预置点调用

```
long    PTZPreset (
```

```
        long        lType,
        long        lPreset
    );
```

Parameters

lType

［IN］预置点类型（0-调整到预置点，1-设置预置点，2-清除预置点）

lPreset

［IN］参数，范围[0，255]

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

2.3.17 云台透传

云台透传

```
long    PtzControl (
        LPCTSTR      sCode,
        long         lSize
    );
```

Parameters

sCode

［IN］控制云台命令数据,命令数据只能是 64 个字节组成的串,如 ff01100800041d。

lSize

［IN］控制云台命令数据长度

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

2.3.18 鼠标操作云台

启用/禁用控件鼠标操作云台功能

```
long    SetUsePtzCtrl (
        long        lEnable
    );
```

Parameters

lEnable

［IN］启用/禁用控件鼠标操作云台功能，0 禁用，1 启用

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

2.3.19 打开关闭移动侦测区域设置

打开关闭移动侦测区域设置

```
long    OpenMDSetPage (
        long        lFlag
);
```

Parameters

lFlag

[IN] 0 表示为正常播放状态，1 表示移动侦测编辑状态

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

2.3.20 显示隐藏编辑区域

显示隐藏编辑区域

```
long    EnablePic (
        long        s32MDNum,
        long        s32EnableValue,
        long        s32Width,
        long        s32Height,
        long        s32X,
        long        s32Y
);
```

Parameters

s32MDNum

[IN] MD 区域（1~4）

s32EnableValue

[IN] 显示隐藏标志（1-显示，2-隐藏）

s32Width

[IN] MD 宽

s32Height

[IN] MD 高

s32X

[IN] MD x 坐标

s32Y

[IN] MD y 坐标

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

Remark

该函数只有在打开移动侦测区域设置后设置才能生效。

2.3.21 获取编辑区域属性

获取编辑区域属性

```
long    GetPic (
        long        s32MDNum,
        long        s32Flag,
);
```

Parameters

s32MDNum

［IN］MD 区域（1~4）

s32Flag

［IN］获取坐标标志（0-width，1-height，2-x，3-y）

Return Values

返回相应的坐标值

2.3.22 保存视频流属性

保存视频流属性到配置文件中

```
long    SetStreamNum (
        long        lStreamNum
);
```

Parameters

lStreamNum

［IN］视频流属性

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

2.3.23 获取视频流属性

从配置文件中获取视频流属性

```
long    GetStreamNum (
);
```

Return Values

11 表示主码流，12 表示次码流。

2.3.24 请求视频流

请求视频流，播放的时候摄像机不发送视频数据（设置后重连生效）

```
long    PauseVideo (
        long        lVideoTag
);
```

Parameters

lVideoTag

［IN］标志，0-请求视频，1-不请求视频

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

2.3.25 请求音频流

请求视频流，播放的时候摄像机不发送音频数据（设置后重连生效）

```
long    PauseAudio (
        long        lAudioTag
);
```

Parameters

lAudioTag

［IN］标志，0-请求音频，1-不请求音频

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

2.3.26 获取显示是否为正常比例

获取显示比例，0 表示拉伸模式，1 表示自动调节比例

```
long    GetAutoAdjust (
);
```

Return Values

0 表示拉伸模式，1 表示自动调节比例

2.3.27 设置自动调节模式

设置画面显示比例

```
long    SetAutoAdjust (
        long        lType
);
```

Parameters

lType

［IN］比例模式，0-拉伸，1-自动调节

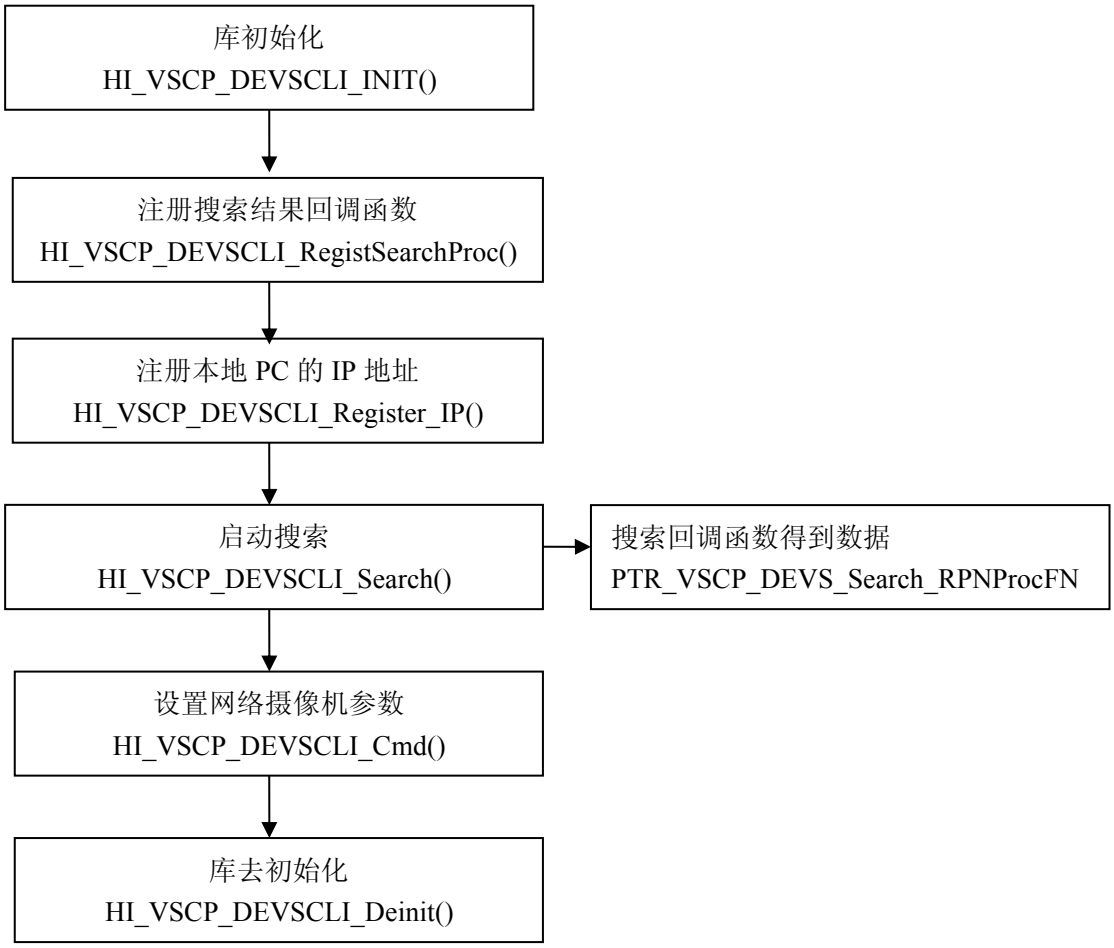
Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

第三部分　搜索 SDK 说明

Version：1.0.0.2

3.1、编程导引



3.2、数据结构

设备流信息：

```
typedef struct {
    HI_CHAR aszIP[HI_VSCP_IP_STRSIZE + 1];              /*IP 地址*/
    HI_CHAR aszMASK[HI_VSCP_IP_STRSIZE + 1];            /*子网掩码*/
    HI_CHAR aszMAC[HI_VSCP_MAC_STRSIZE + 1];            /*MAC 地址*/
    HI_CHAR aszGTW[HI_VSCP_IP_STRSIZE + 1];             /*网关地址*/
    HI_S32    s32Dhcp;      /* DHCP, 1 为开启，0 为关闭 */
    HI_S32    s32DnsFlag; /* DNS 设置标志，1 为自动，0 为手动*/
    HI_CHAR aszFdns[HI_VSCP_IP_STRSIZE + 1];     /*首选 DNS */
    HI_CHAR aszSdns[HI_VSCP_IP_STRSIZE + 1];    /* 备用 DNS */
} HI_S_VSCP_NETINFO;

typedef struct {
    HI_CHAR aszDevID[HI_VSCP_DEVID_STRSIZE + 1];  //设备 ID，随机生成的 32 个字符
```

```
    HI_CHAR aszDevMDL[HI_VSCP_DEVNAME_STRSIZE + 1];  //设备型号
    HI_CHAR aszSwVersion[HI_VSCP_SWVER_STRSIZE + 1]; //软件版本
    HI_CHAR aszDevName[HI_VSCP_DEVNAME_STRSIZE + 1]; //设备名
    HI_CHAR aszHttpPort[HI_VSCP_IP_STRSIZE + 1];     //HTTP 监听端口
    HI_S_VSCP_NETINFO struNetInfo;
} HI_S_VSCP_DEVINFO;
```

发送命令目标设备信息：

```
typedef  struct{
    HI_CHAR*  pszDevID;     //设备标识，设备唯一标识。该参数可通过搜索获得
    HI_CHAR*  pszUserName;  //用户名
    HI_CHAR*  pszPasswd;    //密码
}  HI_S_VSCP_DEVSCLI_DevInfo;
```

3.3、接口说明

3.3.1 初始化设备搜索

初始化

```
HI_S32   HI_VSCP_DEVSCLI_INIT (
    const HI_CHAR*      pszListenIP,
    HI_U16              u16Port,
    HI_U32              u32TimeOut,
    HI_VOID**           ppvHandle
);
```

Parameters

pszListenIP

[IN] 用于处理搜索应答的多播 IP。固定为"239.255.255.250"

u16Port

[IN] 用于处理搜索应答的多播端口号。固定为"8002"

u32TimeOut

[IN] 搜索超时值。单位：秒

ppvHandle

[IN] 输出搜索对象句柄

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

3.3.2 去初始化设备搜索

去初始化

```
HI_S32   HI_VSCP_DEVSCLI_Deinit (
    HI_VOID*    pvHandle
);
```

Parameters

pvHandle

[IN] 输出搜索对象句柄

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

3.3.3 注册搜索响应处理函数

注册搜索响应处理函数

```
HI_S32   HI_VSCP_DEVSCLI_Deinit (
    HI_VOID*                pvHandle,
    PTR_VSCP_DEVS_Search_RPNProcFN   pfunSearchRProc,
    HI_VOID*                pvUserData
);
```

Parameters

pvHandle

[IN] 搜索对象句柄

pfunSearchRProc

[IN] 搜索应答处理回调函数

pvUserData

[IN] 用户数据。该参数将通过搜索应答处理回调函数送出

Callback Function

```
typedef  HI_S32   (*PTR_VSCP_DEVS_Search_RPNProcFN) (
         const  HI_VOID*          pvHandle,
         HI_CHAR*                 pszRNPCode,
         HI_S_VSCP_DEVINFO*       pstruDevInfo,
         HI_VOID*                 pvUserData
         }
```

Callback Function Parameters

pvHandle

未用

pszRNPCode

返回值

pstruDevInfo

设备信息

pvUserData

用户数据

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

Remarks

由于参数 pstruDevInfo 指向的通道及码流属性由 SDK 内部分配空间，上层应用程序需在是使用完该参数后，调用 free 函数释放此空间。具体实现参考 SDK 提供的 DEMO 源码。

3.3.4 注册命令响应处理函数

注册命令响应处理函数

```
HI_S32  HI_VSCP_DEVSCLI_RegistCmdProc (
    HI_VOID*              pvHandle,
    PTR_VSCP_DEVS_Cmd_RPNProcFN   pfunCmdRProc,
    HI_VOID*              pvUserData
);
```

Parameters

pvHandle

[IN] 搜索对象句柄

pfunCmdRProc

[IN] 命令响应处理函数

pvUserData

[IN] 用户数据。该参数将通过命令响应处理回调函数送出

Callback Function

```
typedef  HI_S32  (*PTR_VSCP_DEVS_Cmd_RPNProcFN) (
        const  HI_VOID*          pvHandle,
        HI_CHAR*                 pszRNPCode,
        HI_S_VSCP_DEVSCLI_Cmd_ResponsInfo*  pstruResponseInfo,
        HI_VOID*                 pvUserData
        }
```

Callback Function Parameters

pvHandle

未用

pszRNPCode

返回值。当包含 200 时设置成功否则失败

pstruResponseInfo

未用

pvUserData

用户数据

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

3.3.5 注册接收搜索应答的本地 IP

注册接收搜索应答的本地 IP

```
HI_S32   HI_VSCP_DEVSCLI_Register_IP (
    HI_CHAR     aaszIP[][HI_VSCP_IP_STRSIZE+1],
    HI_U32      u32Num
);
```

Parameters

aaszIP

[OUT] 本地 IP 地址列表

u32Num

[OUT] 本地 IP 地址数量

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

3.3.6 发送搜索命令

发送搜索命令

```
HI_S32   HI_VSCP_DEVSCLI_Search (
    HI_VOID*    pvHandle
);
```

Parameters

pvHandle

[IN] 搜索对象句柄

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

3.3.7 发送设置命令

发送设置命令

```
HI_S32   HI_VSCP_DEVSCLI_Cmd (
    HI_VOID*                                pvHandle,
    const   HI_S_VSCP_DEVSCLI_DevInfo       *pstruDEV,
    HI_S32                                  s32Cmd,
    const HI_VOID*                          pData
);
```

Parameters

pvHandle

[IN] 搜索对象句柄

PstruDEV

[IN] 设备信息

s32Cmd

[IN] 设置类型

```
#define HI_VSCP_CMD_NET         0x01      //网络基本参数设置
#define HI_VSCP_CMD_PORT        0x02      //端口号
```

pData

[IN] 设置参数

1、HI_VSCP_CMD_NET：HI_S_VSCP_NETINFO 结构体

```
typedef struct {
    HI_CHAR aszIP[HI_VSCP_IP_STRSIZE + 1];       //IP 地址
    HI_CHAR aszMASK[HI_VSCP_IP_STRSIZE + 1];     //子网掩码
    HI_CHAR aszMAC[HI_VSCP_MAC_STRSIZE + 1];     //MAC 地址
    HI_CHAR aszGTW[HI_VSCP_IP_STRSIZE + 1];      //网关地址
    HI_S32    s32Dhcp;                //DHCP, 1 为开启，0 为关闭
    HI_S32    s32DnsFlag;         //DNS 设置标志，1 为自动，0 为手动
    HI_CHAR aszFdns[HI_VSCP_IP_STRSIZE + 1];     //首选 DNS
    HI_CHAR aszSdns[HI_VSCP_IP_STRSIZE + 1];     //备用 DNS
} HI_S_VSCP_NETINFO;
```

2、HI_VSCP_CMD_PORT：char str[16]

Return Values

HI_SUCCESS 表示成功，HI_ FAILURE 表示失败。

附录

Ⅰ、文件夹列表

Lib　　存放库文件，里面含有 libNetLib.so， NetLib.lib， NetLib.dll 三个文件；
Include　存放头文件；
VC_demo　　存放 mfc Demo；
Bin　　执行文件存放路径。

Ⅱ、厂家代码和设备类型定义

1. 厂商代码
   用于识别生产厂家。
   可以通过专用工具修改。用户只能读，不能修改。
   ACSII 码，32 个字节长度。

2. 设备类型
   用于识别设备类型，不同的设备功能不同。
   可以通过专用工具修改。用户只能读，不能修改。
   ACSII 码，32 个字节长度。

   每个字段 2 个字节。第一个字节代表字段总类型，第二个字段代表字段子类型。

字段 1	字段 2	字段 3	字段 4	字段 5	字段 6	字段 7	保留字段
芯片	制式	镜头	云台类型	网络类型	平台类型	语言类型	
‘C’	‘F’	‘S’	‘Z’	‘N’	‘P’	‘L’	

   1). 芯片字段 ’C’
   芯片的类型

‘0’	Hi3510
‘1’	Hi3512
‘5’	GM
‘6’	Hi3518

   注：C5、C6 芯片有 3 码流，每个码流不能修改分辨率

   2). 制式字段 ’F’
   视频输入制式,当前值如下：

‘0’	PAL 和 NTS 都支持
‘1’	PAL(704x576, 352x288, 176x144)最大 25 帧
‘2’	NTSC(704x480, 352x240, 176x120)最大 30 帧

   3). 镜头字段 ‘S’
   感光芯片的类型,如下：

‘0’	OV7725 红外控制	亮度，对比度，饱和度，色度，室内，室外，红外开关，上下反转，左右镜像。主码流：VGA, QVGA, QQVGA 次

		码流：QVGA, QQVGA
‘1’	CCDOSP	亮度，对比度，饱和度，色度。 主码流：D1,CIF,QCIF 次码流：CIF,QCIF
‘2’	CCD	亮度，对比度，饱和度，色度。 主码流：D1,CIF,QCIF 次码流：CIF,QCIF
‘3’	MT9D131	亮度，对比度(1-7)，饱和度，上下反转，左右镜像。 主码流：720P(最大 30 帧) 次码流：QVGA
‘4’	HDCCD	主码流：720P(最大 30 帧) 次码流：QVGA
‘5’	630D	亮度(0-6)，对比度(0-8)，饱和度(0-6)。 主码流：720P(最大 30 帧) 次码流：Q720P
‘6’	630C	亮度(0-4)，对比度(0-4)，饱和度(0-2)。 主码流：720P(最大 30 帧) 次码流：Q720P
‘7’	CMOS 720P	亮度，对比度(1-7)，饱和度，上下反转，左右镜像。 **C0 C1：** 主码流：720P(最大 30 帧)，Q720P 次码流：Q720P，QQ720P **C5：** 主码流：720P；第二码流：Q720P；第三码流：QQ720P
‘8’	633	亮度(0-6)，对比度(0-8)，饱和度(0-6)。 **C0 C1：** 主码流：720P(最大 30 帧)，Q720P 次码流：Q720P，QQ720P **C5：** 主码流：720P；第二码流：Q720P；第三码流：QQ720P
‘9’	CMOS 200M	亮度，对比度(1-7)，饱和度，上下反转，左右镜像。 **C0 C1：** 主码流：720P(最大 30 帧)，Q720P 次码流：Q720P，QQ720P UXGA(最大 15 帧)，VGA 次码流，QVGA **C5：** 主码流：720P；第二码流：Q720P；第三码流：QQ720P
‘a’	CCD 960	无图像设置 主码流：928x576, 464x288, 224x144 次码流：464x288, 224x144
‘c’		亮度(0-100)，对比度(0-100)，饱和度(0-100)，上下反转，左右镜像。 **720P 模式：** 主码流：720P；第二码流：Q720P；第三码流：QQ720P **960P 模式：** 主码流：960P；第二码流：VGA；第三码流：QVGA
‘e’		亮度(0-100)，对比度(0-100)，饱和度(0-255)，上下反转，左右镜像。 主码流：720P；第二码流：Q720P；第三码流：QQ720P
‘f’		亮度(0-100)，对比度(0-100)，饱和度(0-255)，上下反转，

		左右镜像。 **720P 模式：** 主码流：720P；第二码流：Q720P；第三码流：QQ720P **960P 模式：** 主码流：960P；第二码流：VGA；第三码流：QVGA

4). 云台字段‘Z’

云台的类型，如下：

‘0’	小球	上，下，左，右，上下巡航，左右巡航，回到中心位置，预置调用（最多 8 个）。没有串口设置。
‘1’	白色球	上，下，左，右，预置调用（最多 8 个）。固定串口设置。
‘2’	变焦球	上，下，左，右，变焦，预置调用（最多 8 个）。固定串口设置。
‘3’	标准	上，下，左，右，雨刷，灯光，预置调用设置。可以设置串口。
‘4’	变倍小球	上，下，左，右，上下巡航，左右巡航，回到中心位置，拉近，拉远。

5). 网络字段‘N’

网络类型，如下：

‘0’	支持有线
‘1’	支持 WIFI
‘2’	支持 EVDO
‘3’	支持 TD
‘4’	支持 WCDMA

6). 平台字段‘P’

平台类型，如下：

‘0’	无平台

7). 语言字段‘L’

语言类型，如下：

‘0’	中文
‘1’	英文

8). 保留字段用于以后扩展