

网络库使用说明书

Version:1.0.3.3

版本更新说明.....	4
一、功能说明.....	8
网络库主要功能.....	8
网络库文件说明.....	8
二、编程导引.....	9
网络库接口调用主要流程.....	9
实时数据流模块流程.....	10
云台控制模块流程.....	11
参数配置模块流程.....	12
语音对讲模块流程.....	13
报警模块流程.....	14
解码器模块流程.....	15
三、数据类型定义说明.....	17
四、错误定义说明.....	18
五、函数说明.....	18
5.1 初始化 SDK.....	18
HI_NET_DEV_Init.....	18
HI_NET_DEV_DeInit.....	18
5.2 用户注册.....	18
HI_NET_DEV_Login.....	18
HI_NET_DEV_LoginExt.....	19
HI_NET_DEV_Logout.....	20
HI_NET_DEV_SetConnectTimeout.....	20
HI_NET_DEV_SetReconnect.....	20
5.3 实时预览.....	21
HI_NET_DEV_StartStream.....	21
HI_NET_DEV_StartStreamExt.....	22
HI_NET_DEV_StopStream.....	23
HI_NET_DEV_MakeKeyFrame.....	23
5.4 实时预览数据回调.....	24
HI_NET_DEV_SetEventCallBack.....	24
HI_NET_DEV_SetStreamCallBack.....	25
HI_NET_DEV_SetDataCallBack.....	27
5.5 摄像机属性设置.....	28
HI_NET_DEV_SetConfig.....	28
HI_NET_DEV_GetConfig.....	48

5.6	云台控制.....	57
	HI_NET_DEV_PTZ_Ctrl_Standard.....	57
	HI_NET_DEV_PTZ_Ctrl_StandardEx.....	58
	HI_NET_DEV_PTZ_Ctrl_Preset.....	58
	HI_NET_DEV_PTZ_Ctrl_Extend.....	59
	HI_NET_DEV_PTZ_Fully_Trans.....	60
5.7	对讲.....	60
	HI_NET_DEV_StartVoice.....	60
	HI_NET_DEV_StopVoice.....	61
	HI_NET_DEV_SendVoiceData.....	61
5.8	录像抓拍.....	62
	HI_NET_DEV_StartRecord.....	63
	HI_NET_DEV_StopRecord.....	64
	HI_NET_DEV_GetRecordState.....	64
	HI_NET_DEV_SnapJpeg.....	65
5.9	设置操作通道.....	66
	HI_NET_DEV_SetChannel.....	66
	HI_NET_DEV_GetChannel.....	66
5.10	解码器.....	67
	解码器调用顺序.....	67
	HI_NET_DEV_GetDisplayCfg.....	67
	HI_NET_DEV_SetDisplayCfg.....	68
	HI_NET_DEV_StartDec.....	69
	HI_NET_DEV_StopDec.....	69
	HI_NET_DEV_GetLoopDecChnInfo.....	70
	HI_NET_DEV_SetLoopDecChnInfo.....	71
	HI_NET_DEV_GetLoopDecChnEnable.....	71
	HI_NET_DEV_SetLoopDecChnEnable.....	71
	HI_NET_DEV_GetLoopDecEnable.....	72
	HI_NET_DEV_GetChnInfo.....	72
	HI_NET_DEV_GetDecChnEnable.....	73
	HI_NET_DEV_SetDecChnEnable.....	74
	HI_NET_DEV_StartPassiveDecode.....	74
	HI_NET_DEV_StopPassiveDecode.....	75
	HI_NET_DEV_DecodeSendData.....	75
	解码器其他相关接口.....	76
5.11	AVI 文件解析.....	77
	AVI 解析调用顺序.....	77
	AVI 解析接口错误定义.....	77
	AVI_CreateReader.....	77
	AVI_DestroyReader.....	78
	AVI_ReadFrame.....	78
	AVI_SeekFrame.....	79
	AVI_ReadFileInfo.....	79

六、音频编解码说明.....	81
6.1 音频采集格式设置.....	81
6.2 音频采集流程.....	81
6.3 音频播放流程.....	82
6.4 音频编码.....	82
6.5 音频解码.....	84
七、附录.....	86
附录 I、文件夹列表.....	86
附录 II、Linux Demo 使用说明.....	86
附录 III、厂家代码和设备类型定义.....	86

版本更新说明

V1.0.3.3 2015-06-24

1. 修改 HI_NET_DEV_SetConfig，增加手动触发外置报警的功能，对应的命令为：HI_NET_DEV_CMD_EXT_ALARM，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。

v1.0.3.2 2015-03-07

1. 云台透传中透传的最大数据长度修改为 128，原本为 64
2. 修改 HI_NET_DEV_SetConfig，增加继电器开关功能，对应的命令为：HI_NET_DEV_CMD_RELAYCTRL，对应的结构体为：HI_S_RelayCtrl，

V1.0.2.9 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、更正 C#无法调用回调函数问题，请修正相应的回调函数，重新编译在原来的回调中加入 NETSDK_APICALL

```
typedef HI_S32 ( *HI_ON_STREAM_CALLBACK)(
    HI_U32 u32Handle, /* 句柄 */
    HI_U32 u32DataType, /* 数据类型，系统数据或音视频数据 */
    HI_U8* pu8Buffer, /* 数据包含帧头 */
    HI_U32 u32Length, /* 数据长度 */
    HI_VOID* pUserData /* 用户数据*/
);
```

变更为：

```
typedef HI_S32 (NETSDK_APICALL *HI_ON_STREAM_CALLBACK)(
    HI_U32 u32Handle, /* 句柄 */
    HI_U32 u32DataType, /* 数据类型，系统数据或音视频数据 */
    HI_U8* pu8Buffer, /* 数据包含帧头 */
    HI_U32 u32Length, /* 数据长度 */
    HI_VOID* pUserData /* 用户数据*/
);
```

V1.0.2.6 2013-04-13

- 1、增加 [AVI 文件解析](#) 接口。

V1.0.2.5 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》。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。

V1.0.2.3 2013-02-05

- 1、 修改 [HI_NET_DEV_SetConfig](#) 设置 OSD 参数 HI_NET_DEV_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。

V1.0.2.2 2012-12-10

- 2、 添加三码流接口 [HI_NET_DEV_StartStreamExt](#)。三码流需要设备支持；
- 3、 网络抓拍接口：[HI_NET_DEV_SnapJpeg](#)
- 4、 添加三码流控制接口：参阅 [HI_NET_DEV_SetConfig](#) 和 [HI_NET_DEV_GetConfig](#)

```
#define HI_NET_DEV_CMD_AUDIO_VOLUME_IN      0x1070  //音频输入音量
#define HI_NET_DEV_CMD_AUDIO_VOLUME_OUT     0x1071  //音频输出音量
```
- 5、 添加三码流控制接口：参阅 [HI_NET_DEV_SetConfig](#) 和 [HI_NET_DEV_GetConfig](#)

```
#define HI_NET_DEV_CMD_VIDEO_PARAM_EXT      0x1047  //视频参数
#define HI_NET_DEV_CMD_AUDIO_PARAM_EXT      0x1048  //音频参数
#define HI_NET_DEV_CMD_RESOLUTION_EXT      0x1049  //分辨率
```

V1.0.2.1 2012-10-22

- 1、 添加解码器 SDK
- 2、 添加 WIFI 控制接口：参阅 [HI_NET_DEV_SetConfig](#) 和 [HI_NET_DEV_GetConfig](#)

```
#define HI_NET_DEV_CMD_WIFI_PARAM          0x1030  //WIFI 参数
#define HI_NET_DEV_CMD_WIFI_SEARCH         0x1031  //WIFI 搜索
#define HI_NET_DEV_CMD_WIFI_CHECK          0x1035  //WIFI check
```

V1.0.1.9 2012-05-29

- 1、 增加动态 I 帧接口 [HI_NET_DEV_MakeKeyFrame](#)；

V1.0.1.8 2012-03-29

- 1、 修改字段为 S7、S9 的设备的默认值；
- 2、 增加 [HI_NET_DEV_LoginExt](#) 登陆接口，接口中带有超时时间；
- 3、 增加 [HI_NET_DEV_SetChannel](#) 和 [HI_NET_DEV_GetChannel](#)，用于设置 NVR 通道；
- 4、 增加 NVR 参数设置

```
#define HI_NET_NVR_CMD_NET_EXT              0x1050  // NVR 网络参数
#define HI_NET_NVR_CMD_RTSP_INFO            0x1051  // NVR rtsp 参数
#define HI_NET_NVR_CMD_USER                  0x1052  // NVR 用户参数
#define HI_NET_NVR_CMD_CHANNEL_INFO          0x1053  // NVR 通道参数
#define HI_NET_NVR_CMD_SEARCH                0x1055  // NVR 搜索设备
#define HI_NET_NVR_CMD_RECORD_INFO           0x1056  // NVR 通道录像参数
#define HI_NET_NVR_CMD_RECORD_SYS            0x1057  // NVR 系统参数
#define HI_NET_NVR_CMD_TIME                  0x1058  // NVR 时间参数
#define HI_NET_NVR_CMD_RESET                 0x1059  // NVR 恢复出厂设置
#define HI_NET_NVR_CMD_REBOOT                0x1060  // NVR 重启
#define HI_NET_NVR_CMD_RECORD_STATE           0x1061  // 获取录像状态
#define HI_NET_NVR_CMD_DISK_INFO             0x1062  // 获取硬盘信息
#define HI_NET_NVR_CMD_DISK_FORMAT           0x1063  // 格式化硬盘
```

```
#define HI_NET_NVR_CMD_RECORD_STATE_EX    0x1064    // 获取录像状态
```

具体参阅 [HI_NET_DEV_SetConfig](#) 和 [HI_NET_DEV_GetConfig](#)

V1.0.1.6 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》。

V1.0.1.5 2011-12-05

- 1、 修正 S7 字段无法获取上下左右翻转。

V1.0.1.4 2011-11-22

- 1、 添加心跳包处理，心跳包从 HI_NET_DEV_SetDataCallBack 回调出来，详细回到请查阅 [HI_NET_DEV_SetDataCallBack](#)。

V1.0.1.3 2011-11-01

- 1、 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》。

V1.0.1.2 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》。

V1.0.1.1 2011-07-22

- 1、 添加 AVI 录像接口 [HI_NET_DEV_StartRecord](#)、[HI_NET_DEV_StopRecord](#) 和 [HI_NET_DEV_GetRecordState](#)，并在事件回调中增加录像操作相关的操作：HI_NET_DEV_RECORD_START 和 HI_NET_DEV_RECORD_STOP。

V1.0.1.0 2011-07-02

- 1、 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》。
- 2、 添加控制输入报警开关接口：参阅 [HI_NET_DEV_SetConfig](#) 和 [HI_NET_DEV_GetConfig](#) 的 HI_NET_DEV_CMD_ATTR_EXT 选项。

V1.0.0.9 2011-06-08

- 1、 [HI_NET_DEV_PTZ_Ctrl_Standard](#) 和 [HI_NET_DEV_PTZ_Ctrl_StandardEx](#) 添加焦点调整和光圈变化命令


```
#define HI_NET_DEV_CTRL_PTZ_FOCUSIN          0x3007    //焦点前调
#define HI_NET_DEV_CTRL_PTZ_FOCUSOUT        0x3008    //焦点后调
#define HI_NET_DEV_CTRL_PTZ_APERTUREIN      0x3009    //光圈放大
#define HI_NET_DEV_CTRL_PTZ_APERTUREOUT    0x3010    //光圈变小
```
- 2、 添加网络参数接口，参阅 [HI_NET_DEV_SetConfig](#) 和 [HI_NET_DEV_GetConfig](#) 的 HI_NET_DEV_CMD_NET_EXT 选项。改指令将 HI_NET_DEV_CMD_NET_INFO 和 HI_NET_DEV_CMD_HTTP_PORT 合并。

V1.0.0.7 2011-03-12

- 1、 添加设备重启接口，参阅 [HI_NET_DEV_SetConfig](#) 的 HI_NET_DEV_CMD_REBOOT 选项。
- 2、 添加设备恢复出厂设置接口，参阅 [HI_NET_DEV_SetConfig](#) 的

HI_NET_DEV_CMD_RESET 选项。

3、 添加校时接口，参阅 [HI_NET_DEV_SetConfig](#) 和 [HI_NET_DEV_GetConfig](#) 的 HI_NET_DEV_CMD_SERVER_TIME 选项。

V1.0.0.5 2010-12-4

1、 更改网络库编译选项，Windows 下为默认，去掉 HI_OS_WIN32 编译选项，Linux 下编译要添加-DHI_OS_LINUX。

2、 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。

V1.0.0.4 2010-11-23

1、 更新 [HI_NET_DEV_GetConfig](#) 获取设备信息、产品 ID、用户连接数功能。

2、 修改开始对讲接口，可以兼容 G711 音频

一、功能说明

网络库主要功能

与摄像机通讯、获取码流、参数设置、语音对讲、语态控制、录像等功能。

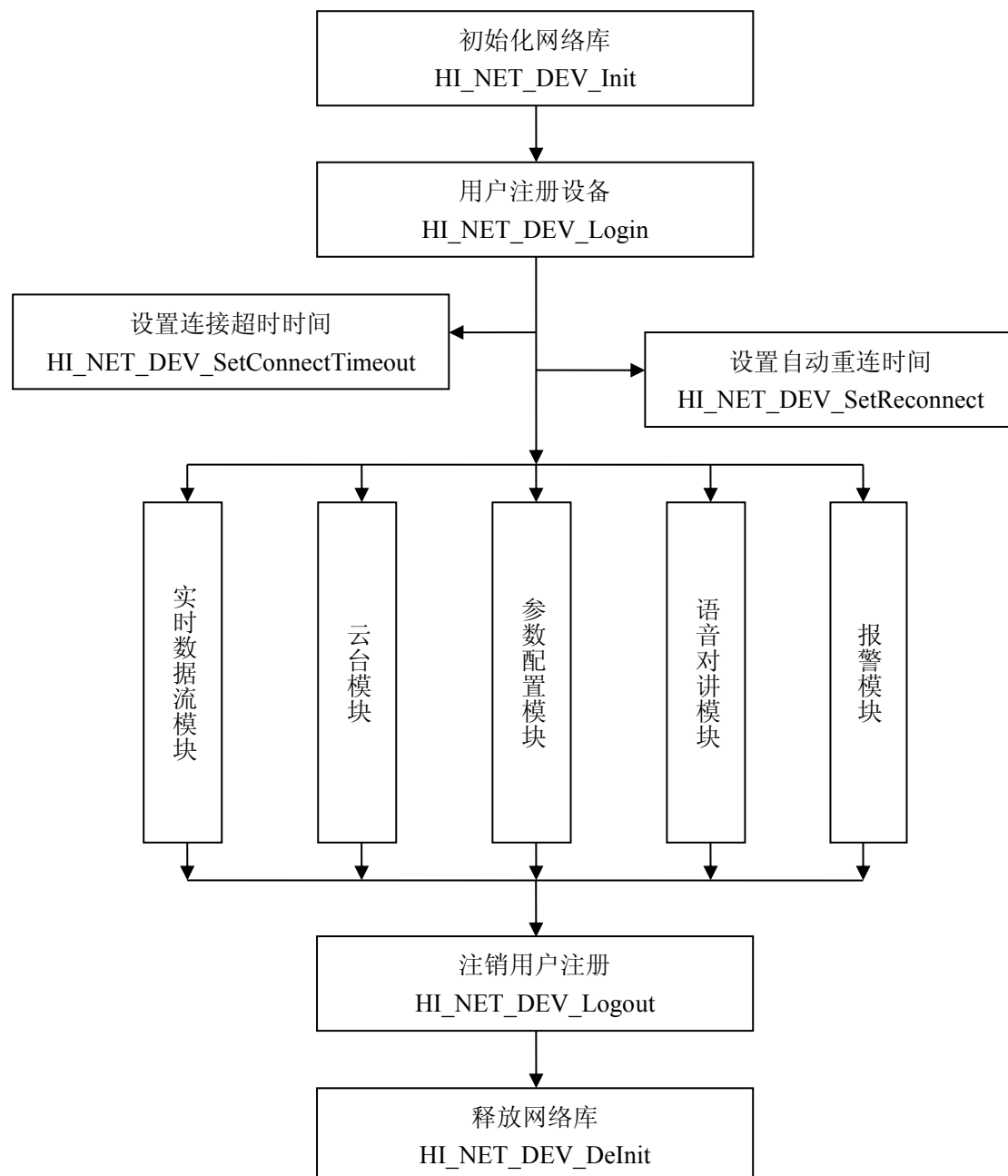
网络库用于只连接网络部分，回调出来的数据交给播放库处理，与播放库分开。具体用在如平台软件、集中管理客户端等工程中。

网络库文件说明

网络库	hi_net_dev_sdk.h	头文件
	NetLib.lib	LIB 库文件
	NetLib.dll	DLL 库文件
	libNetLib.so	Linux 动态库
公用文件	hi_dataType.h	头文件

二、编程导引

网络库接口调用主要流程

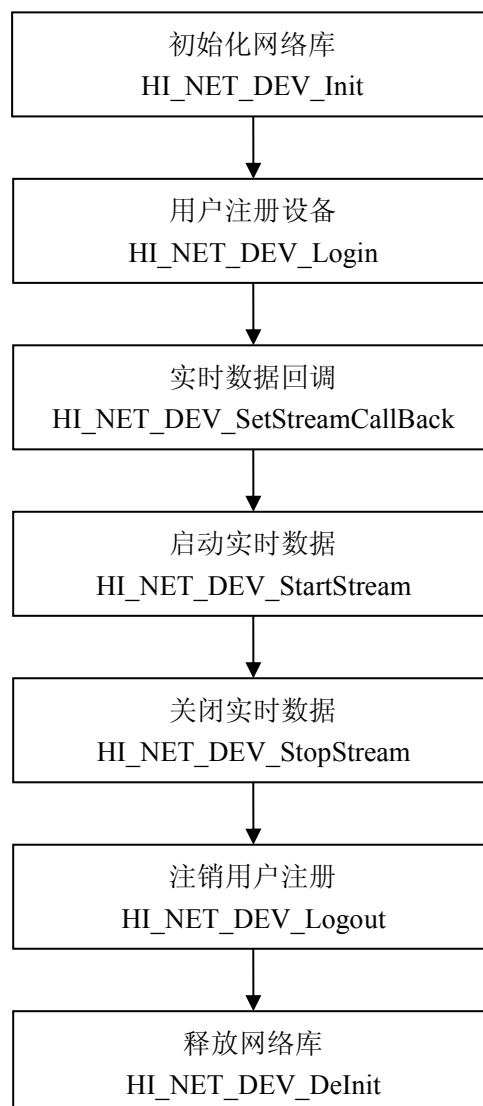


按实现功能的不同可以分成五个模块，实现每个模块的功能时初始化网络库、用户注册设备、注销设备和释放网络库资源这 4 个流程是必不可少的。

- 初始化网络库（HI_NET_DEV_Init 接口）：对整个网络库系统的初始化；
- 设置连接超时时间（HI_NET_DEV_SetConnectTimeout 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_NET_DEV_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；

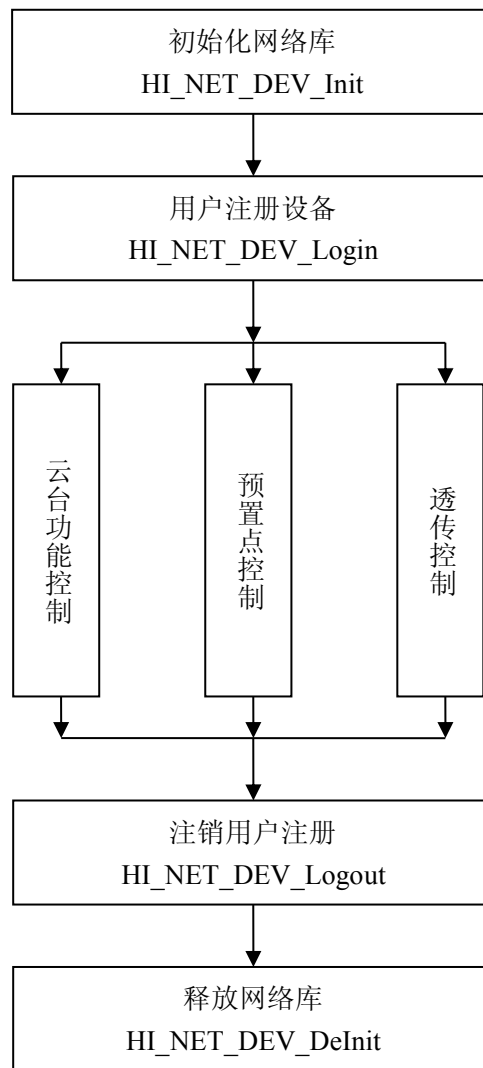
- 实时数据流模块：实时数据流启动后通过回调函数获取摄像机的当前实时数据。具体流程详见实时数据流模块流程；
- 云台控制模块：云台控制分为基本功能控制、云台的预置点控制和云台透传。具体流程详见云台控制模块流程；
- 参数配置模块：设置和获取前端摄像机的参数，主要包括设备参数、网络参数、报警参数、异常参数、用户配置等参数信息。具体流程详见参数配置模块流程；
- 语音对讲转发模块：实现和前端服务器的语音数据对讲和语音数据获取，音频编码格式可以指定。具体流程详见语音对讲模块流程；
- 报警模块：处理前端服务器上传的各种报警信号。报警分为“移动报警”和“输入报警”两种数据。具体流程详见报警模块流程。

实时数据流模块流程



实时数据流启动后通过回调函数获取摄像机的当前实时数据。每一帧数据都是完整的数据帧，数据帧包含帧头，用于区分数据的类型。相关接口有：HI_NET_DEV_StartStream 、HI_NET_DEV_StopStream 、HI_NET_DEV_SetStreamCallBack。

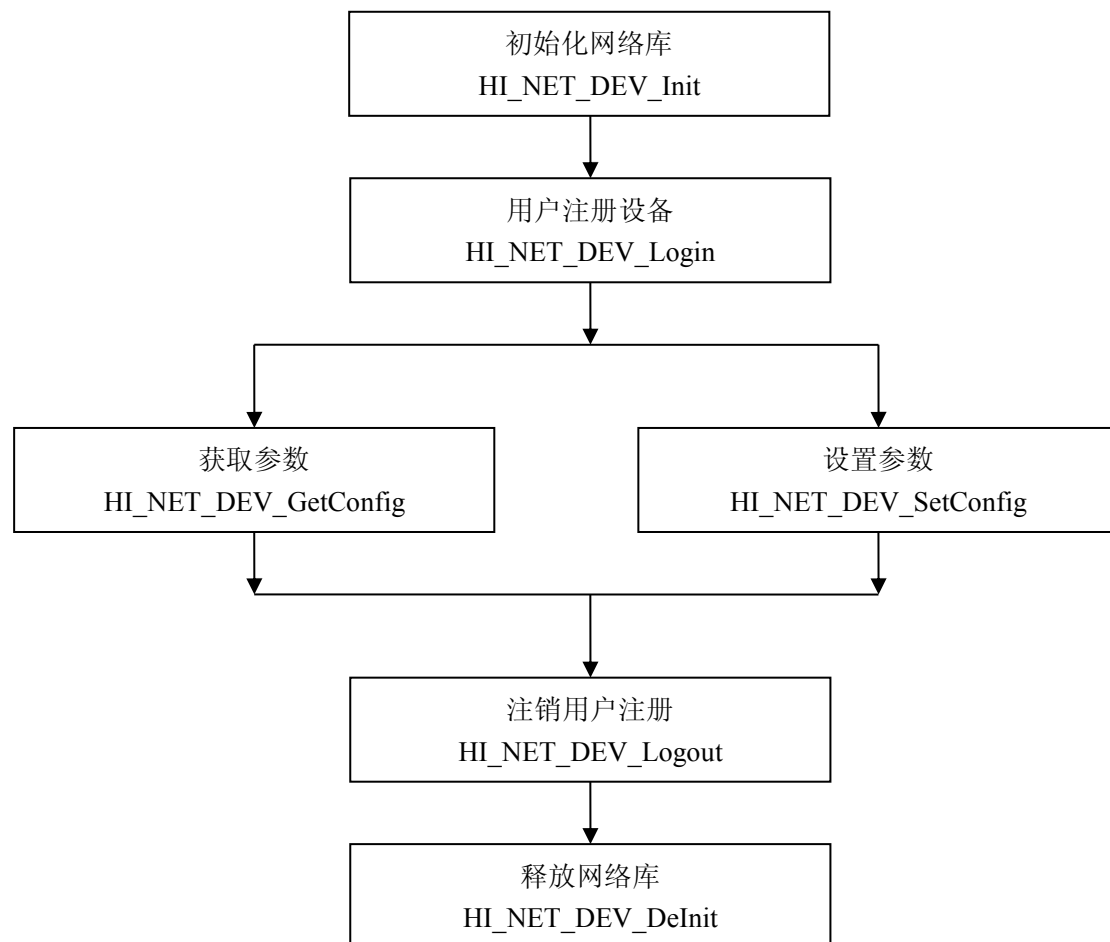
云台控制模块流程



云台控制在初始化网络库和注册设备后就可以使用，包含云台基本功能控制、预置点控制和透传控制。

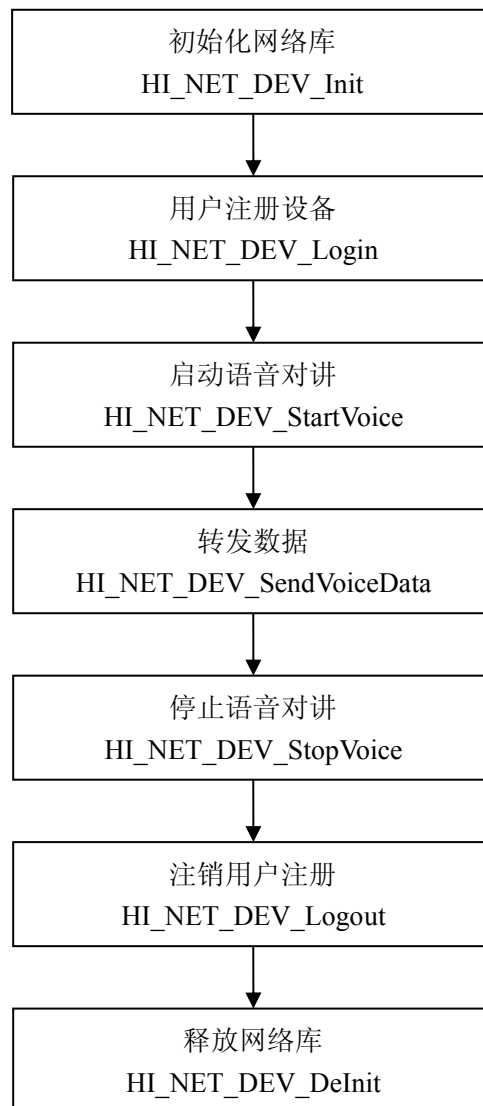
- 云台空能控制：包括上下左右方向、停止、聚焦以及灯光雨刷等扩展功能，相关接口有：HI_NET_DEV_PTZ_Ctrl_Standard、HI_NET_DEV_PTZ_Ctrl_StandardEx、HI_NET_DEV_PTZ_Ctrl_Extend;
- 预置点控制：最大可以设置 256 个预置点（具体设备设置的个数不同），相关接口有：HI_NET_DEV_PTZ_Ctrl_Preset;
- 透传控制：通过串口控制云台，相关接口有：HI_NET_DEV_PTZ_Fully_Trans。

参数配置模块流程



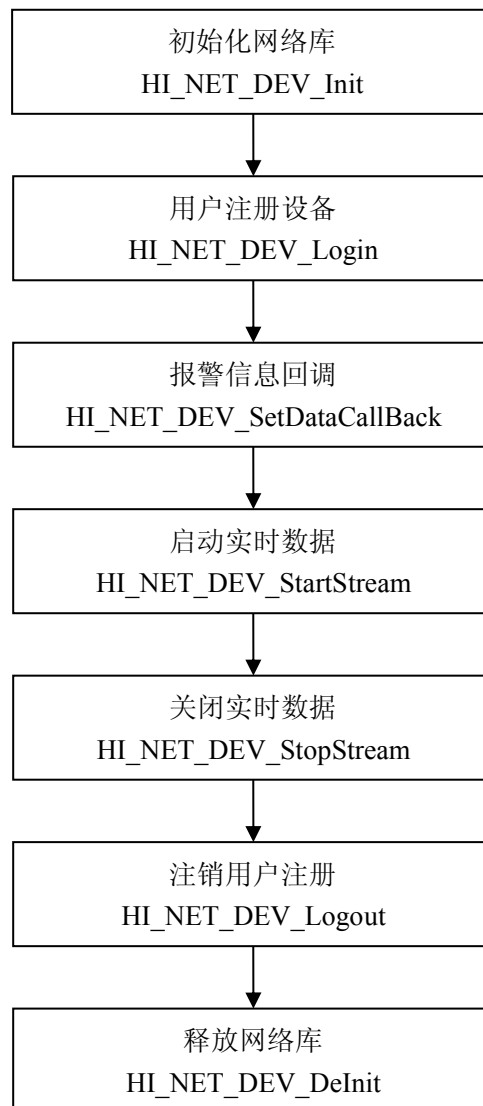
实现参数配置首先必须做好初始化网络库和用户注册这两个步骤,将用户注册接口返回的句柄作为配置接口的首个参数。建议在每次设置某类参数之前,先调用获取参数的接口 (HI_NET_DEV_GetConfig) 得到完整的参数结构,修改需要更改的参数,作为设置参数接口中的输入参数,最后调用设置参数接口 (HI_NET_DEV_SetConfig),返回成功即设置成功。

语音对讲模块流程



语音对讲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。音频的编码解码请查阅音频编解码部分和 Demo。

报警模块流程

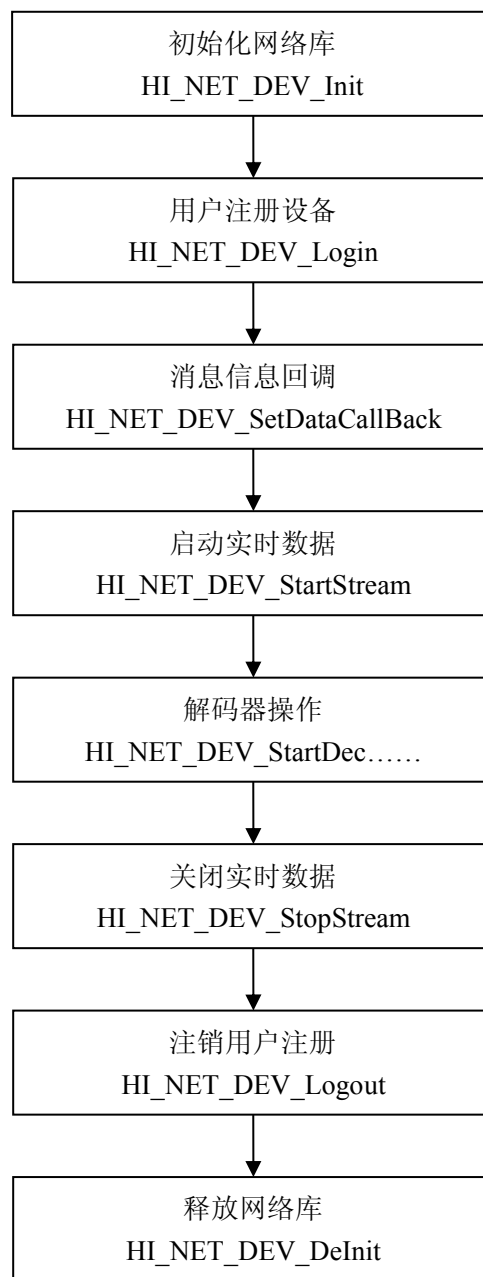


报警回调可分为“移动报警”和“输入报警”两种数据。

- 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_NET_DEV_SetDataCallBack 的使用方法。

解码器模块流程



HI_NET_DEV_SetDataCallBack 回调出来的数据是一个字符串，其中回调 HI_U32 u32DataType 的值为 3。字符串的长度用与通道数一致，如解码器的总通道数为 9，回调出来的字符串长度为 9，如果字符串为 012000000 表示第一通道当前没有解码，第二通道当前为动态解码，第三通道当前正在轮巡，即字符的每一个字节表示为：0 没有解码，1 动态解码，2 轮巡

```

HI_U32 u32Handle = 0;
//登陆解码器
s32Ret = HI_NET_DEV_Login(&u32Handle, "admin", "admin", "192.168.1.24", 80);
if(s32Ret != HI_SUCCESS)
    return HI_FAILURE;

```

```
//启用时间回调
HI_NET_DEV_SetEventCallBack(u32Handle, OnNetEventCallBack, (HI_VOID*)this);
//启用状态回调功能
HI_NET_DEV_SetDataCallBack(u32Handle, OnNetDataCallBack, this);

//启用解码器
HI_S_STREAM_INFO sStreamInfo;
.....
s32Ret = HI_NET_DEV_StartStream(u32Handle, &sStreamInfo);
if(s32Ret != HI_SUCCESS)
{
    HI_NET_DEV_Logout(u32Handle);
    u32Handle = 0;
    return HI_FAILURE;
}

//开始解码器相关操作
HI_S_CHN_INFO sChnInfo;
memset(&sChnInfo, 0, sizeof(HI_S_CHN_INFO));
strcpy(sChnInfo.sHost, "192.168.1.88");
.....
s32Ret = HI_NET_DEV_StartDec(u32Handle, u32Channel, &sChnInfo);
.....
.....
//销毁解码器
if(u32Handle != 0)
{
    HI_NET_DEV_StopStream(u32Handle);
    HI_NET_DEV_Logout(u32Handle);
    u32Handle = 0;
}
```


三、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)
```

四、错误定义说明

```
#define HI_NET_DEV_PARAM_ERROR          0x41001
#define HI_NET_DEV_MEMORY_ERROR        0x41002
#define HI_NET_DEV_NOT_SUPPORT         0x41003
#define HI_NET_DEV_PARAM_CHECK_ERROR   0x51001 //Paramter input error
#define HI_NET_DEV_PARAM_CMD_ERROR     0x51002 //No command
#define HI_NET_DEV_PARAM_PARSE_ERROR   0x51003 //parsse command
#define HI_NET_DEV_NET_CONNECT_FAIL    0x52001 //connect host failure
#define HI_NET_DEV_NET_TRANSFER_FAIL   0x52002 //transfer host failure
#define HI_NET_DEV_NET_RETURN_ERROR    0x52003 //host return error
#define HI_NET_DEV_NET_NOT_SUPPORT     0x53000 //device not support the paramter
```

五、函数说明

5.1 初始化 SDK

HI_NET_DEV_Init

初始化，调用其他函数前使用，仅在初始化 SDK 使用

```
HI_S32 HI_NET_DEV_Init(
);
```

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

HI_NET_DEV_DeInit

释放 SDK，仅在释放 SDK 使用

```
HI_S32 HI_NET_DEV_DeInit(
);
```

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

Remarks

HI_NET_DEV_Init、HI_NET_DEV_DeInit 在一个程序中仅初始化一次，初始化 Socket

5.2 用户注册

HI_NET_DEV_Login

用户设备注册

```
HI_S32 HI_NET_DEV_Login(
    HI_U32*          pu32Handle,
    const HI_CHAR*   psUsername,
```

```

        const HI_CHAR*      psPassword,
        const HI_CHAR*      psHost,
        HI_U16               u16Port
    );

```

Parameters

pu32Handle
[OUT] 操作句柄

psUsername
[IN] 用户名

psPassword
[IN] 密码

psHost
[IN] 主机，可以是 IP 地址也可以是域名

u16Port
[IN] 端口号

Return Values

HI_SUCCESS 表示成功，

HI_NET_DEV_NET_CONNECT_FAIL 表示连接失败；

HI_NET_DEV_NET_TRANSFER_FAIL 表示域名解析失败；

HI_NET_DEV_NET_RETURN_ERROR 表示主机错误。

HI_NET_DEV_LoginExt

用户设备注册扩展，带超时

```

HI_S32 HI_NET_DEV_LoginExt (
    HI_U32*          pu32Handle,
    const HI_CHAR*   psUsername,
    const HI_CHAR*   psPassword,
    const HI_CHAR*   psHost,
    HI_U16            u16Port,
    HI_U32            u32TimeOut
);

```

Parameters

pu32Handle
[OUT] 操作句柄

psUsername
[IN] 用户名

psPassword
[IN] 密码

psHost
[IN] 主机，可以是 IP 地址也可以是域名

u16Port

[IN] 端口号

u32TimeOut

[IN] 超时时间，单位毫秒，默认 10000 毫秒

Return Values

HI_SUCCESS 表示成功，

HI_NET_DEV_NET_CONNECT_FAIL 表示连接失败；

HI_NET_DEV_NET_TRANSFER_FAIL 表示域名解析失败；

HI_NET_DEV_NET_RETURN_ERROR 表示主机错误。

HI_NET_DEV_Logout

用户取消登录

```
HI_S32 HI_NET_DEV_Logout(
    HI_U32      u32Handle
);
```

Parameters

u32Handle

[IN] 操作句柄

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

HI_NET_DEV_SetConnectTimeout

设置连接超时时间，默认超时是 5 秒，单位是毫秒

```
HI_S32 HI_NET_DEV_SetConnectTimeout (
    HI_U32      u32Handle
    HI_U32      u32Timeout
);
```

Parameters

u32Handle

[IN] 操作句柄

u32Timeout

[IN] 超时时间，单位是毫秒

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

HI_NET_DEV_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_NET_DEV_SetReconnect (
```

```
        HI_U32      u32Handle
        HI_U32      u32Interval
    );
```

Parameters

u32Handle
[IN] 操作句柄

u32Interval
[IN] 自动重连间隔时间，单位是毫秒

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

5.3 实时预览

HI_NET_DEV_StartStream

实时数据

```
HI_S32 HI_NET_DEV_StartStream (
    HI_U32      u32Handle
    HI_S_STREAM_INFO* 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_BOOL     bIFlag;               //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_NET_DEV_CHANNEL_1    1
// #define HI_NET_DEV_CHANNEL_2    2
```

```

#define HI_NET_DEV_CHANNEL_3    3
#define HI_NET_DEV_CHANNEL_4    4

// 连接网络连接模式，目前仅支持 TCP
#define HI_NET_DEV_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_NET_DEV_STREAM_VIDEO_ONLY    0x01
#define HI_NET_DEV_STREAM_AUDIO_ONLY    0x02
#define HI_NET_DEV_STREAM_VIDEO_AUDIO    0x03
#define HI_NET_DEV_STREAM_VIDEO_DATA    0x05
#define HI_NET_DEV_STREAM_AUDIO_DATA    0x06
#define HI_NET_DEV_STREAM_ALL           0x07

```

HI_NET_DEV_StartStreamExt

实时数据

```

HI_S32 HI_NET_DEV_StartStreamExt (
    HI_U32                u32Handle
    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_NET_DEV_STREAM_1    0
#define HI_NET_DEV_STREAM_2    1

```

```

#define HI_NET_DEV_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_NET_DEV_CHANNEL_1     1

// 连接网络连接模式，目前仅支持 TCP
#define HI_NET_DEV_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_NET_DEV_STREAM_VIDEO_ONLY      0x01
#define HI_NET_DEV_STREAM_AUDIO_ONLY     0x02
#define HI_NET_DEV_STREAM_VIDEO_AUDIO    0x03
#define HI_NET_DEV_STREAM_VIDEO_DATA     0x05
#define HI_NET_DEV_STREAM_AUDIO_DATA     0x06
#define HI_NET_DEV_STREAM_ALL            0x07

```

HI_NET_DEV_StopStream

停止数据流

```

HI_S32 HI_NET_DEV_StopStream(
    HI_U32      u32Handle
);

```

Parameters

u32Handle
[IN] 操作句柄

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

HI_NET_DEV_MakeKeyFrame

动态创建一个 I 帧

```

HI_S32 HI_NET_DEV_MakeKeyFrame (
    HI_U32      u32Handle,
    HI_U32      u32Channel
);

```

Parameters

u32Handle
[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。

5.4 实时预览数据回调

HI_NET_DEV_SetEventCallBack

事件数据回调

```
HI_S32 HI_NET_DEV_SetEventCallBack(  
    HI_U32          u32Handle  
    HI_ON_EVENT_CALLBACK cbEventCallBack,  
    HI_VOID*        pUserData  
);
```

Parameters

- u32Handle
[IN] 操作句柄
- cbEventCallBack
[IN] 事件数据回调函数
- pUserData
[IN] 用户数据

Callback Function

```
typedef HI_S32 (*HI_ON_EVENT_CALLBACK) (  
    HI_U32      u32Handle,  
    HI_U32      u32Event,  
    HI_VOID*    pUserData  
);
```

Callback Function Parameters

- u32Handle
操作句柄
- u32Event
事件

宏定义	宏定义值	含义
HI_NET_DEV_CONNECTING	0	正在连接
HI_NET_DEV_CONNECTED	1	已经连接
HI_NET_DEV_CONNECT_FAILED	2	连接失败
HI_NET_DEV_ABORTIBE_DISCONNECTED	3	关闭连接
HI_NET_DEV_NORMAL_DISCONNECTED	4	关闭连接
HI_NET_DEV_RECONNECTING	5	重新连接
HI_NET_DEV_RECORD_START	6	开始录像
HI_NET_DEV_RECORD_STOP	7	停止录像

- pUserData
用户数据

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

HI_NET_DEV_SetStreamCallBack

码流数据回调

```
HI_S32 HI_NET_DEV_SetStreamCallBack (  
    HI_U32          u32Handle  
    HI_ON_STREAM_CALLBACK cbStreamCallBack,  
    HI_VOID*        pUserData  
);
```

Parameters

- u32Handle
[IN] 操作句柄
- cbStreamCallBack
[IN] 码流数据回调函数
- pUserData
[IN] 用户数据

Callback Function

```
typedef HI_S32 (*HI_ON_STREAM_CALLBACK)(  
    HI_U32      u32Handle,  
    HI_U32      u32DataType,  
    HI_U8*      pu8Buffer,  
    HI_U32      u32Length,  
    HI_VOID*    pUserData  
);
```

Callback Function Parameters

- u32Handle
操作句柄
- u32DataType
数据类型，音视频数据或头文件数据

宏定义	宏定义值	含义
HI_NET_DEV_AV_DATA	0	音视频数据
HI_NET_DEV_SYS_DATA	1	文件数据

- pu8Buffer
数据包含帧头
- u32Length
数据长度
- pUserData
用户数据

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

Remarks

- 1、连接上的第一个数据包为 HI_NET_DEV_SYS_DATA 类型。
- 2、如果 pu8Buffer 数据为 HI_NET_DEV_SYS_DATA，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_NET_DEV_AUDIO_TYPE_G711	0	G711
HI_NET_DEV_AUDIO_TYPE_G726	1	G726
HI_NET_DEV_AUDIO_TYPE_AMR	2	AMR

```
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_NET_DEV_SYS_FLAG 0x53565848。

- 3、如果 pu8Buffer 数据为 HI_NET_DEV_AV_DATA，pu8Buffer 的帧头是由 HI_S_SysHeader 结构组成：

```
typedef struct
{
    HI_U32  u32AVFrameFlag;           // 帧标志
    HI_U32  u32AVFrameLen;            // 帧的长度
    HI_U32  u32AVFramePTS;            // 时间戳
    HI_U32  u32VFrameType;            // 视频的类型，I 帧或 P 帧
} HI_S_AVFrame;
```

u32AVFrameFlag 格式如下表：

宏定义	宏定义值	含义
HI_NET_DEV_VIDEO_FRAME_FLAG	0x46565848	视频数据
HI_NET_DEV_AUDIO_FRAME_FLAG	0x46415848	音频数据

u32VFrameType 格式如下表：

宏定义	宏定义值	含义
-----	------	----

HI_NET_DEV_VIDEO_FRAME_I	1	I 帧
HI_NET_DEV_VIDEO_FRAME_P	2	P 帧

HI_NET_DEV_SetDataCallBack

信息数据回调

```
HI_S32 HI_NET_DEV_SetDataCallBack (  
    HI_U32          u32Handle  
    HI_ON_DATA_CALLBACK cbDataCallBack,  
    HI_VOID*        pUserData  
);
```

Parameters

- u32Handle
[IN] 操作句柄
- cbDataCallBack
[IN] 信息数据回调函数
- pUserData
[IN] 用户数据

Callback Function

```
typedef HI_S32 (*HI_ON_DATA_CALLBACK)(  
    HI_U32      u32Handle,  
    HI_U32      u32DataType,  
    HI_U8*      pu8Buffer,  
    HI_U32      u32Length,  
    HI_VOID*    pUserData  
);
```

Callback Function Parameters

- u32Handle
操作句柄
- u32DataType
数据类型

宏定义	宏定义值	含义
HI_NET_DEV_MOTION_DETECTION	0	移动侦测报警
HI_NET_DEV_INPUT_ALARM	1	输入报警
HI_NET_DEV_KEEP_ALIVE	2	心跳包

pu8Buffer
数据。如果为 HI_NET_DEV_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_NET_DEV_MOTION_AREA_1	1	区域 1
HI_NET_DEV_MOTION_AREA_2	2	区域 2
HI_NET_DEV_MOTION_AREA_3	3	区域 3
HI_NET_DEV_MOTION_AREA_4	4	区域 4

u32Length

数据长度，HI_NET_DEV_MOTION_DETECTION，两个区域同时就有：

u32Length = 2*sizeof(HI_S_ALARM_MD)

u32DataType

用户数据

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

5.5 摄像机属性设置

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

HI_NET_DEV_SetConfig

设置摄像机参数

```
HI_S32 HI_NET_DEV_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_NET_DEV_CMD_DISPLAY	0x1001	图像参数
HI_NET_DEV_CMD_DISPLAY_EXT	0x1002	翻转
HI_NET_DEV_CMD_INFRARED	0x1003	红外

HI_NET_DEV_CMD_VIDEO_PARAM	0x1004	视频参数
HI_NET_DEV_CMD_OSD_PARAM	0x1005	OSD 参数
HI_NET_DEV_CMD_AUDIO_PARAM	0x1006	音频参数
HI_NET_DEV_CMD_AUDIO_INPUT	0x1007	音频输入
HI_NET_DEV_CMD_RESOLUTION	0x1008	图像分辨率
HI_NET_DEV_CMD_FREQUENCY	0x1009	频率
HI_NET_DEV_CMD_PTZ_PARAM	0x1010	云台信息
HI_NET_DEV_CMD_MD_PARAM	0x1011	移动报警信息
HI_NET_DEV_CMD_NET_INFO	0x1012	网络信息
HI_NET_DEV_CMD_HTTP_PORT	0x1013	网页端口号
HI_NET_DEV_CMD_SERVER_TIME	0x1017	设置摄像机时间
HI_NET_DEV_CMD_REBOOT	0x1018	重启
HI_NET_DEV_CMD_RESET	0x1019	恢复出厂设置
HI_NET_DEV_CMD_NET_EXT	0x1022	设置网络参数
HI_NET_DEV_CMD_ATTR_EXT	0x1026	控制输入报警
HI_NET_NVR_CMD_NET_EXT	0x1050	NVR 网络参数
HI_NET_NVR_CMD_RTSP_INFO	0x1051	NVR rtsp 参数
HI_NET_NVR_CMD_USER	0x1052	NVR 用户参数
HI_NET_NVR_CMD_CHANNEL_INFO	0x1053	NVR 通道参数
HI_NET_NVR_CMD_RECORD_INFO	0x1056	NVR 通道录像参数
HI_NET_NVR_CMD_RECORD_SYS	0x1057	NVR 录像系统参数
HI_NET_NVR_CMD_TIME	0x1058	NVR 时间设置
HI_NET_NVR_CMD_RESET	0x1059	NVR 恢复出厂设置
HI_NET_NVR_CMD_REBOOT	0x1060	NVR 重启
HI_NET_NVR_CMD_DISK_FORMAT	0x1063	NVR 格式化硬盘
HI_NET_DEV_CMD_WIFI_PARAM	0x1030	WIFI 参数设置
HI_NET_DEV_CMD_WIFI_CHECK	0x1035	WIFI check
HI_NET_DEV_CMD_VIDEO_PARAM_EXT	0x1047	视频参数（扩展）
HI_NET_DEV_CMD_AUDIO_PARAM_EXT	0x1048	音频参数（扩展）
HI_NET_DEV_CMD_RESOLUTION_EXT	0x1049	分辨率参数（扩展）
HI_NET_DEV_CMD_AUDIO_VOLUME_IN	0x1070	音频输入音量
HI_NET_DEV_CMD_AUDIO_VOLUME_OUT	0x1071	音频输出音量
HI_NET_DEV_CMD_RELAYCTRL	0x1084	继电器控制
HI_NET_DEV_CMD_EXT_ALARM	0x1085	外置报警抓拍

pBuf

[IN] 设置数据

u32BufLen

[IN] 数据长度

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

Remarks

1、HI_NET_DEV_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_NET_DEV_SetConfig( lHandle, // HI_NET_DEV_GetConfig
                      HI_NET_DEV_CMD_DISPLAY,
                      &sDisplay,
                      sizeof(HI_S_Display));
```

2、HI_NET_DEV_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_NET_DEV_SCENE_AUTO	0	自动
HI_NET_DEV_SCENE_INDOOR	1	室内
HI_NET_DEV_SCENE_OUTDOOR	2	室外

Example:

```
HI_S_Display_Ext sDisplayEx;
sDisplayEx.blFlip = HI_FALSE;
sDisplayEx.blMirror = HI_FALSE;
sDisplayEx.s32Scene = HI_NET_DEV_SCENE_AUTO;
HI_NET_DEV_SetConfig( lHandle, // HI_NET_DEV_GetConfig
                      HI_NET_DEV_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_NET_DEV_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_NET_DEV_INFRARED_AUTO	0	自动
HI_NET_DEV_INFRARED_ON	1	开
HI_NET_DEV_INFRARED_OFF	2	关

Example:

```
HI_S_Infrared sInfrared;
sInfrared.s32Infrared = HI_NET_DEV_INFRARED_AUTO;
HI_NET_DEV_SetConfig ( IHandle,          // HI_NET_DEV_GetConfig
                      HI_NET_DEV_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_NET_DEV_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_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 sVideo;
// 注：u32Channel 与 HI_S_STREAM_INFO 一致
sVideo.u32Channel = HI_NET_DEV_CHANNEL_1;
sVideo.blFlag = HI_TRUE;
sVideo.u32Bitrate = 1024;
sVideo.u32Frame = 25;
```

```
sVideo.u32Iframe = 50;
sVideo.blCbr = HI_FALSE;
sVideo.u32ImgQuality = 1;
HI_NET_DEV_SetConfig ( lHandle,           // HI_NET_DEV_GetConfig
                        HI_NET_DEV_CMD_VIDEO_PARAM,
                        &sVideo,
                        sizeof(HI_S_Video));
```

5、HI_NET_DEV_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_NET_DEV_SetConfig ( lHandle,           // HI_NET_DEV_GetConfig
                        HI_NET_DEV_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_NET_DEV_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_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_NET_DEV_AUDIO_TYPE_G711	0	G711
HI_NET_DEV_AUDIO_TYPE_G726	1	G726
HI_NET_DEV_AUDIO_TYPE_AMR	2	AMR

Example:

```
HI_S_Audio sAudio;
// 注：u32Channel 与 HI_S_STREAM_INFO 一致
sAudio.u32Channel = HI_NET_DEV_CHANNEL_1;
sAudio.blFlag = HI_TRUE;
sAudio.blEnable = HI_TRUE;
sAudio.u32Type = HI_NET_DEV_AUDIO_TYPE_G711;
HI_NET_DEV_SetConfig ( lHandle,           // HI_NET_DEV_GetConfig
                      HI_NET_DEV_CMD_AUDIO_PARAM,
                      &sAudio,
                      sizeof(HI_S_Audio));
```

7、HI_NET_DEV_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_NET_DEV_SetConfig ( lHandle,           // HI_NET_DEV_GetConfig
                      HI_NET_DEV_CMD_AUDIO_INPUT,
                      &audioInput,
                      sizeof(HI_S32));
```

8、HI_NET_DEV_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_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_NET_DEV_RESOLUTION_VGA	0	VGA： 640x480
HI_NET_DEV_RESOLUTION_QVGA	1	QVGA： 320x240
HI_NET_DEV_RESOLUTION_QQVGA	2	QQVGA： 160x120, 160x112
HI_NET_DEV_RESOLUTION_D1	3	D1： 704x576, 704x480
HI_NET_DEV_RESOLUTION_CIF	4	CIF： 352x288, 352x240
HI_NET_DEV_RESOLUTION_QCIF	5	QCIF： 176x144, 176x120,

		176x112
HI_NET_DEV_RESOLUTION_720P	6	720P: 1280x720
HI_NET_DEV_RESOLUTION_Q720	7	Q720: 640x352
HI_NET_DEV_RESOLUTION_QQ72	8	QQ720: 320x176
HI_NET_DEV_RESOLUTION_UXGA	9	UXGA: 1600x1200
HI_NET_DEV_RESOLUTION_960H	10	960H: 960x576
HI_NET_DEV_RESOLUTION_Q960H	11	Q960H: 480x288
HI_NET_DEV_RESOLUTION_QQ960H	12	QQ960H: 240x144
HI_NET_DEV_RESOLUTION_1080P	13	1080P: 1920x1080
HI_NET_DEV_RESOLUTION_960P	14	960P: 1280x960

Example:

```

HI_S_Resolution sResolution;
// 注: u32Channel 与 HI_S_STREAM_INFO 一致
sResolution.u32Channel = HI_NET_DEV_CHANNEL_1;
sResolution.bIFlag = HI_TRUE;
sResolution.u32Resolution = HI_NET_DEV_RESOLUTION_CIF;
HI_NET_DEV_SetConfig ( lHandle,                // HI_NET_DEV_GetConfig
                      HI_NET_DEV_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_NET_DEV_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_NET_DEV_SetConfig ( lHandle,                // HI_NET_DEV_GetConfig
                      HI_NET_DEV_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_NET_DEV_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_NET_DEV_PTZ_PRO_PELCOD	0	PELCO-D
HI_NET_DEV_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_NET_DEV_PTZ_B110	110	110
HI_NET_DEV_PTZ_B300	300	300
HI_NET_DEV_PTZ_B1200	1200	1200
HI_NET_DEV_PTZ_B2400	2400	2400
HI_NET_DEV_PTZ_B4800	4800	4800
HI_NET_DEV_PTZ_B9600	9600	9600
HI_NET_DEV_PTZ_B19200	19200	19200
HI_NET_DEV_PTZ_B38400	38400	38400
HI_NET_DEV_PTZ_B57600	57600	57600

u32DataBit 数据位数据如下表:

宏定义	宏定义值	含义
HI_NET_DEV_PTZ_DATA_5	5	
HI_NET_DEV_PTZ_DATA_6	6	
HI_NET_DEV_PTZ_DATA_7	7	
HI_NET_DEV_PTZ_DATA_8	8	

u32StopBit 停止位数据如下表:

宏定义	宏定义值	含义
HI_NET_DEV_PTZ_STOP_1	1	
HI_NET_DEV_PTZ_STOP_2	2	

u32Parity 校验数据如下表:

宏定义	宏定义值	含义
HI_NET_DEV_PTZ_PARITY_NONE	0	无
HI_NET_DEV_PTZ_PARITY_ODD	1	奇校验
HI_NET_DEV_PTZ_PARITY_EVEN	2	偶校验

Example:

```

HI_S_PTZ sPtz;
sPtz.u32Protocol = HI_NET_DEV_PTZ_PRO_PELCOD;
sPtz.u32Address = 1;
sPtz.u32Baud = HI_NET_DEV_PTZ_B9600;
sPtz.u32DataBit = HI_NET_DEV_PTZ_DATA_8;
sPtz.u32StopBit = HI_NET_DEV_PTZ_STOP_1;
sPtz.u32Parity = HI_NET_DEV_PTZ_PARITY_NONE;

```

```

HI_NET_DEV_SetConfig ( lHandle,                // HI_NET_DEV_GetConfig
                      HI_NET_DEV_CMD_PTZ_PARAM,
                      &sPtz,
                      sizeof(HI_S_PTZ));

```

11、HI_NET_DEV_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_NET_DEV_CHANNEL_1;
sMdParam.u32Area = 1;
sMdParam.blEnable = HI_TRUE;
sMdParam.u32Sensitivity = 50;
sMdParam.u32X = 100;
sMdParam.u32Y = 100;
sMdParam.u32Width = 200;
sMdParam.u32Height = 200;
HI_NET_DEV_SetConfig ( lHandle,                // HI_NET_DEV_GetConfig
                      HI_NET_DEV_CMD_MD_PARAM,
                      &sMdParam,
                      sizeof(HI_S_MD_PARAM));

```

注：次码流不支持移动侦测。

12、HI_NET_DEV_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_NET_DEV_SetConfig ( lHandle,           // HI_NET_DEV_GetConfig
                      HI_NET_DEV_CMD_NET_INFO,
                      &sNetInfo,
                      sizeof(HI_S_NETINFO));

```

13、HI_NET_DEV_CMD_HTTP_PORT

```

typedef struct HI_HTTPPORT
{
    HI_U32      u32HttpPort;
} HI_S_HTTPPORT;

```

Example:

```

HI_S_HTTPPORT sHttpPort;
sHttpPort.u32HttpPort = 80;
HI_NET_DEV_SetConfig ( lHandle,           // HI_NET_DEV_GetConfig
                      HI_NET_DEV_CMD_HTTP_PORT,
                      &sHttpPort,
                      sizeof(HI_S_HTTPPORT));

```

14、HI_NET_DEV_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_NET_DEV_SetConfig ( lHandle,
                      HI_NET_DEV_CMD_SERVER_TIME,
                      &sServerTime,
                      sizeof(HI_S_SERVERTIME));

```

15、HI_NET_DEV_CMD_REBOOT

重启摄像机

Example:

```
HI_NET_DEV_SetConfig (lHandle, HI_NET_DEV_CMD_REBOOT,NULL,0);
```

16、HI_NET_DEV_CMD_RESET

恢复出厂设置

Example:

```
HI_NET_DEV_SetConfig (lHandle, HI_NET_DEV_CMD_RESET,NULL,0);
```

17、HI_NET_DEV_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_NET_DEV_SetConfig ( lHandle,                // HI_NET_DEV_GetConfig
                      HI_NET_DEV_CMD_NET_EXT,
                      & sNetExt,
                      sizeof(HI_S_NET_EXT));
```

18、HI_NET_DEV_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_NET_DEV_SetConfig( lHandle,
                      HI_NET_DEV_CMD_ATTR_EXT,
                      & sAttrExt,
                      sizeof(HI_S_ATTR_EXT));

```

19、HI_NET_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_NET_DEV_SetConfig( lHandle,          // HI_NET_DEV_GetConfig
                      HI_NET_NVR_CMD_NET_EXT,

```

```
& sAttrExt,
sizeof(HI_S_NET_EXT));
```

20、HI_NET_NVR_CMD_RTSP_INFO

设置 NVR rtsp 参数信息

```
typedef struct HI_RTSPINFO
{
    HI_U32 u32RtspPort; //RTSP 端口
    HI_U32 u32AuthFlag; //是否启用 RTSP, 1 代表启用, 其他代表不启用
} HI_S_RTSPINFO;
```

Example:

```
HI_S_RTSPINFO sRtspInfo;
sRtspInfo.u32AuthFlag = 1;
sRtspInfo.u32RtspPort = 554;
HI_NET_DEV_SetConfig ( lHandle, // HI_NET_DEV_GetConfig
HI_NET_NVR_CMD_RTSP_INFO,
& sRtspInfo,
sizeof(HI_S_RTSPINFO));
```

21、HI_NET_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_NET_DEV_SetConfig ( lHandle,
HI_NET_NVR_CMD_USER,
& sUserInfo,
sizeof(HI_S_USER));
```

22、HI_NET_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(sNvrChn.sChnInfo.sHost, "192.168.1.20");
sNvrChn.sChnInfo.u32Port = 80;
sNvrChn.sChnInfo.u32Enable = 1;
... ..
HI_NET_DEV_SetConfig( lHandle,          // HI_NET_DEV_GetConfig
                      HI_NET_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_NET_DEV_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。

23、HI_NET_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], "PPPPPPPPPPPPPPPPPPPPPP
                                //PPPPPPPPNNNNNNNNNNNNPPPPPPPPPPPPPP
                                //PPPPPPPP");P 代表计划录像，N 代表不录像。

}HI_S_RECORD_INFO;

```

Example:

```

HI_S_RECORD_INFO sRecInfo;

```

注：调用一次只能获取或设置一个通道，可以配合 [HI_NET_DEV_SetChannel](#) 设置 NVR 的通道再来操作。

```

HI_S_SERVERTIME sServerTime;
memcpy(sServerTime.sTime, "20110311091208", sizeof(sServerTime.sTimezone));
HI_NET_DEV_SetConfig ( lHandle,           // HI_NET_DEV_GetConfig
                      HI_NET_NVR_CMD_TIME,
                      &sServerTime,
                      sizeof(HI_S_SERVERTIME));

```

26、HI_NET_NVR_CMD_REBOOT

重启摄像机

Example:

```
HI_NET_DEV_SetConfig (lHandle, HI_NET_NVR_CMD_REBOOT, NULL, 0);
```

27、HI_NET_NVR_CMD_RESET

恢复出厂设置

Example:

```
HI_NET_DEV_SetConfig (lHandle, HI_NET_NVR_CMD_RESET, NULL, 0);
```

28、HI_NET_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;
```

```
sDisFormat.s32DiskNum = 1;
```

```
HI_NET_DEV_SetConfig (lHandle,
                      HI_NET_NVR_CMD_DISK_FORMAT,
                      &sDisFormat,
                      Sizeof(HI_DISK_FORMAT));
```

29、HI_NET_DEV_CMD_WIFI_PARAM

Wifi 参数设置

```
#define WIFI_NET_INFRA 0
```

```
#define WIFI_NET_ADHOC 1
```

```
#define WIFI_AUTH_NONE 0
```

```
#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_NET_DEV_SetConfig (IHandle,
                      HI_NET_DEV_CMD_WIFI_PARAM,
                      & sWifi,
                      Sizeof(HI_S_WIFI_PARAM));

```

30、HI_NET_DEV_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_NET_DEV_SetConfig( m_uiHandle,
                              HI_NET_DEV_CMD_WIFI_CHECK,
                              &sWifiParam,
                              sizeof(HI_S_WIFI_PARAM));

if(HI_SUCCESS != s32Ret)
{
    return;
}

HI_S32 s32Enable = 0;
s32Ret = HI_NET_DEV_GetConfig( m_uiHandle,
                              HI_NET_DEV_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！

31、 HI_NET_DEV_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_NET_DEV_CHANNEL_1;
sVideo.u32Stream = HI_NET_DEV_STREAM_1;
sVideo.u32Bitrate = 1024;
sVideo.u32Frame = 25;
sVideo.u32Iframe = 50;
sVideo.blCbr = HI_FALSE;
sVideo.u32ImgQuality = 1;
HI_NET_DEV_SetConfig ( IHandle,                // HI_NET_DEV_GetConfig
                      HI_NET_DEV_CMD_VIDEO_PARAM_EXT,
                      &sVideo,
                      sizeof(HI_S_Video_Ext));

```

32、 HI_NET_DEV_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_NET_DEV_AUDIO_TYPE_G711	0	G711
HI_NET_DEV_AUDIO_TYPE_G726	1	G726

Example:

```
HI_S_Audio_Ext sAudio;
// 注：u32Channel 与 HI_S_STREAM_INFO 一致
sAudio.u32Channel = HI_NET_DEV_CHANNEL_1;
sAudio.u32Stream = HI_NET_DEV_STREAM_1;
sAudio.blEnable = HI_TRUE;
sAudio.u32Type = HI_NET_DEV_AUDIO_TYPE_G711;
HI_NET_DEV_SetConfig ( lHandle,           // HI_NET_DEV_GetConfig
                      HI_NET_DEV_CMD_AUDIO_PARAM_EXT,
                      &sAudio,
                      sizeof(HI_S_Audio_Ext));
```

33、HI_NET_DEV_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_NET_DEV_RESOLUTION_VGA	0	VGA: 640x480
HI_NET_DEV_RESOLUTION_QVGA	1	QVGA: 320x240
HI_NET_DEV_RESOLUTION_QQVGA	2	QQVGA: 160x120, 160x112
HI_NET_DEV_RESOLUTION_D1	3	D1: 704x576, 704x480
HI_NET_DEV_RESOLUTION_CIF	4	CIF: 352x288, 352x240
HI_NET_DEV_RESOLUTION_QCIF	5	QCIF : 176x144 , 176x120 , 176x112
HI_NET_DEV_RESOLUTION_720P	6	720P: 1280x720

Example:

```
HI_S_Resolution_Ext sResolution;
// 注：u32Channel 与 HI_S_STREAM_INFO 一致
sResolution.u32Channel = HI_NET_DEV_CHANNEL_1;
sResolution.u32Stream = HI_NET_DEV_STREAM_1;
```

```

sResolution.u32Resolution = HI_NET_DEV_RESOLUTION_CIF;
HI_NET_DEV_SetConfig ( lHandle,          // HI_NET_DEV_GetConfig
                      HI_NET_DEV_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。

34、 HI_NET_DEV_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_NET_DEV_SetConfig ( lHandle,          // HI_NET_DEV_GetConfig
                      HI_NET_DEV_CMD_AUDIO_VOLUME_IN,
                      &sAuVolume,
                      sizeof(HI_S_AudioVolume));
```

35、 HI_NET_DEV_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_NET_DEV_SetConfig ( lHandle,          // HI_NET_DEV_GetConfig
                      HI_NET_DEV_CMD_AUDIO_VOLUME_OUT,
                      &sAuVolume,
                      sizeof(HI_S_AudioVolume));
```

36、 HI_NET_DEV_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;
```

37、 HI_NET_DEV_CMD_EXT_ALARM

Example:

```
HI_S_ExtAlarm m_ExtAlarm;  
m_ExtAlarm.bEnable = HI_TRUE;  
HI_NET_DEV_SetConfig (IHandle, //设备句柄  
    HI_NET_DEV_CMD_EXT_ALARM, //命令类型  
    &m_ExtAlarm,  
    Sizeof(HI_S_ExtAlarm));
```

HI_NET_DEV_GetConfig

获取摄像机参数

```
HI_S32 HI_NET_DEV_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_NET_DEV_GET_PRODUCT_VENDOR	0x1000	厂商信息
HI_NET_DEV_CMD_DISPLAY	0x1001	图像参数
HI_NET_DEV_CMD_DISPLAY_EXT	0x1002	翻转
HI_NET_DEV_CMD_INFRARED	0x1003	红外
HI_NET_DEV_CMD_VIDEO_PARAM	0x1004	视频参数
HI_NET_DEV_CMD_OSD_PARAM	0x1005	OSD 参数
HI_NET_DEV_CMD_AUDIO_PARAM	0x1006	音频参数
HI_NET_DEV_CMD_AUDIO_INPUT	0x1007	音频输入
HI_NET_DEV_CMD_RESOLUTION	0x1008	图像分辨率
HI_NET_DEV_CMD_FREQUENCY	0x1009	频率
HI_NET_DEV_CMD_PTZ_PARAM	0x1010	云台信息
HI_NET_DEV_CMD_MD_PARAM	0x1011	移动报警信息
HI_NET_DEV_CMD_NET_INFO	0x1012	网络配置信息
HI_NET_DEV_CMD_HTTP_PORT	0x1013	网页端口号
HI_NET_DEV_CMD_DEVICE_INFO	0x1014	设备信息
HI_NET_DEV_CMD_PRODUCTID	0x1015	产品 ID
HI_NET_DEV_CMD_USERNUM	0x1016	用户连接数

HI_NET_DEV_CMD_SERVER_TIME	0x1017	获取摄像机时间
HI_NET_DEV_CMD_NET_EXT	0x1022	获取网络参数
HI_NET_DEV_CMD_ATTR_EXT	0x1026	获取输入报警参数
HI_NET_NVR_CMD_NET_EXT	0x1050	NVR 网络参数
HI_NET_NVR_CMD_RTSP_INFO	0x1051	NVR rtsp 参数
HI_NET_NVR_CMD_USER	0x1052	NVR 用户参数
HI_NET_NVR_CMD_CHANNEL_INFO	0x1053	NVR 通道信息
HI_NET_NVR_CMD_SEARCH	0x1055	NVR 搜索 NVR 网络中的摄像机
HI_NET_NVR_CMD_RECORD_INFO	0x1056	NVR 通道录像参数
HI_NET_NVR_CMD_RECORD_SYS	0x1057	NVR 全局参数
HI_NET_NVR_CMD_TIME	0x1058	NVR 时间参数
HI_NET_NVR_CMD_RECORD_STATE	0x1061	NVR 录像状态
HI_NET_NVR_CMD_DISK_INFO	0x1062	NVR 硬盘状态
HI_NET_NVR_CMD_RECORD_STATE_EX	0x1064	NVR 录像状态
HI_NET_DEV_CMD_WIFI_PARAM	0x1030	WIFI 参数设置
HI_NET_DEV_CMD_WIFI_SEARCH	0x1031	WIFI 搜索
HI_NET_DEV_CMD_WIFI_CHECK	0x1035	WIFI check
HI_NET_DEV_CMD_VIDEO_PARAM_EXT	0x1047	视频参数（扩展）
HI_NET_DEV_CMD_AUDIO_PARAM_EXT	0x1048	音频参数（扩展）
HI_NET_DEV_CMD_RESOLUTION_EXT	0x1049	分辨率参数（扩展）
HI_NET_DEV_CMD_AUDIO_VOLUME_IN	0x1070	音频输入音量
HI_NET_DEV_CMD_AUDIO_VOLUME_OUT	0x1071	音频输出音量

pBuf

[OUT] 获取数据

u32BufLen

[IN] 数据长度

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

Remarks

获取相关信息用到的结构体与设置的结构体相同，请查阅 [HI_NET_DEV_SetConfig](#) 中相关结构体的定义，设置中没有用到的结构体：

1、HI_NET_DEV_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_NET_DEV_GetConfig ( IHandle,
                      HI_NET_DEV_GET_PRODUCT_VENDOR,
                      &sProduct,
                      sizeof(HI_S_ProductVendor));

```

2、HI_NET_DEV_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_NET_DEV_GetConfig ( IHandle,
                      HI_NET_DEV_CMD_DEVICE_INFO,
                      &sDeviceInfo,
                      sizeof(HI_DEVICE_INFO));

```

3、HI_NET_DEV_CMD_PRODUCTID

产品 ID 用字符串表示。

Example:

```

HI_CHAR sID[64] = {0};
HI_NET_DEV_GetConfig (IHandle,  HI_NET_DEV_CMD_PRODUCTID,  sID,
sizeof(sID));

```

4、HI_NET_DEV_CMD_USERNUM

获取用户数据用到 int 来获取即可。

Example:

```

int nNum = 0;
HI_NET_DEV_GetConfig (IHandle,  HI_NET_DEV_CMD_USERNUM,  &nNum,
sizeof(int));

```

5、HI_NET_DEV_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_NET_DEV_GetConfig (lHandle, HI_NET_DEV_CMD_SERVER_TIME, &
sServerTime, sizeof(HI_S_SERVERTIME));

```

6、HI_NET_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_NET_DEV_GetConfig ( lHandle,
                        HI_NET_NVR_CMD_NET_EXT,
                        & sAttrExt,
                        sizeof(HI_S_NET_EXT));

```

7、HI_NET_NVR_CMD_RTSP_INFO

获取 NVR rtsp 参数信息

```
typedef struct HI_RTSPINFO
```

```

{
    HI_U32 u32RtspPort; //RTSP 端口
    HI_U32 u32AuthFlag; //是否启用 RTSP，1 代表启用，其他代表不启用
} HI_S_RTSPINFO;

```

Example:

```
HI_S_RTSPINFO sRtspInfo;
```

```

HI_NET_DEV_GetConfig ( lHandle,
                        HI_NET_NVR_CMD_RTSP_INFO,
                        & sRtspInfo,
                        sizeof(HI_S_RTSPINFO));

```

8、HI_NET_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_NET_DEV_GetConfig ( lHandle,
                        HI_NET_NVR_CMD_USER,
                        & sUserInfo,
                        sizeof(HI_S_USERINFO));

```

9、HI_NET_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_NET_DEV_GetConfig ( lHandle,
                        HI_NET_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_NET_DEV_SetChannel](#)

设置 NVR 的通道再来操作。

10、HI_NET_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], "PPPPPPPPPPPPPPPPPPPPPP
                                //PPPPPPPPNNNNNNNNNNNNPPPPPPPPPPPPPP
                                //PPPPPPPPPP");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_NET_DEV_GetConfig ( IHandle,
                        HI_NET_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_NET_DEV_SetChannel](#) 设置 NVR 的通道再来操作。

11、HI_NET_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_NET_DEV_GetConfig ( IHandle,
```

```

HI_NET_NVR_CMD_RECORD_SYS,
& sNvrRecSys,
sizeof(HI_S_RECORD_SYS));

```

12、HI_NET_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_NET_DEV_GetConfig ( lHandle,
                       HI_NET_NVR_CMD_TIME,
                       &sServerTime,
                       sizeof(HI_S_SERVERTIME));

```

13、HI_NET_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_NET_DEV_GetConfig ( lHandle,
                       HI_NET_NVR_CMD_SEARCH,
                       &sSearchInfo,
                       sizeof(HI_S_SEARCH_INFO));

```

14、HI_NET_NVR_CMD_RECORD_STATE

HI_NET_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_NET_DEV_GetConfig ( lHandle,
                        HI_NET_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_NET_DEV_GetConfig ( lHandle,
                        HI_NET_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_NET_DEV_SetChannel](#) 设置 NVR 的通道再来操作。

15、HI_NET_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_NET_DEV_GetConfig ( lHandle,
                        HI_NET_NVR_CMD_DISK_INFO,
                        &sDiskInfo,

```

```
sizeof(HI_S_DISK_INFO));
```

16、 HI_NET_DEV_CMD_WIFI_SEARCH

查找 WIFI

```
#define WIFI_NET_INFRA 0
```

```
#define WIFI_NET_ADHOC 1
```

```
#define WIFI_AUTH_NONE 0
```

```
#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 sWifiPt[MAX_WFPT];
} HI_S_WIFI_INFO;
```

Example:

```
HI_S_WIFI_INFO sWifiInfo;
```

```
memset(&sWifiInfo, 0, sizeof(HI_S_WIFI_INFO));
```

```
s32Ret = HI_NET_DEV_GetConfig( m_uiHandle,
                                HI_NET_DEV_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);
}
```

5.6 云台控制

摄像机是否支持云台属性，可以通过获取 HI_NET_DEV_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_NET_DEV_PTZ_Ctrl_Standard

云台控制操作，设备含有 Z0 字段的设备不支持。

```
HI_S32 HI_NET_DEV_PTZ_Ctrl_Standard (
    HI_U32      u32Handle
    HI_U32      u32Command,
    HI_U32      u32Speed
);
```

Parameters

u32Handle

[IN] 操作句柄

u32Command

[IN] 云台控制命令

宏定义	宏定义值	含义
HI_NET_DEV_CTRL_PTZ_STOP	0x3000	停止云台
HI_NET_DEV_CTRL_PTZ_UP	0x3001	云台上仰
HI_NET_DEV_CTRL_PTZ_DOWN	0x3002	云台下俯
HI_NET_DEV_CTRL_PTZ_LEFT	0x3003	云台左转
HI_NET_DEV_CTRL_PTZ_RIGHT	0x3004	云台右转
HI_NET_DEV_CTRL_PTZ_ZOOMIN	0x3005	焦距变大(倍率变大)
HI_NET_DEV_CTRL_PTZ_ZOOMOUT	0x3006	焦距变小(倍率变小)
HI_NET_DEV_CTRL_PTZ_FOCUSIN	0x3007	焦点前调
HI_NET_DEV_CTRL_PTZ_FOCUSOUT	0x3008	焦点后调
HI_NET_DEV_CTRL_PTZ_APERTUREIN	0x3009	光圈变小
HI_NET_DEV_CTRL_PTZ_APERTUREOUT	0x3010	光圈变大

u32Speed

[IN] 速度

```
#define HI_NET_DEV_CTRL_PTZ_SPEED_MAX 0x3F //最大速度
#define HI_NET_DEV_CTRL_PTZ_SPEED_MIN 0x00 //最小速度
```

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

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_NET_DEV_PTZ_Ctrl_StandardEx

云台控制操作扩展，单步执行。

```
HI_S32 HI_NET_DEV_PTZ_Ctrl_StandardEx (  
    HI_U32      u32Handle  
    HI_U32      u32Command,  
);
```

Parameters

- u32Handle
[IN] 操作句柄
- u32Command
[IN] 云台控制命令

宏定义	宏 定 义 值	含 义
HI_NET_DEV_CTRL_PTZ_STOP	0x3000	停止云台
HI_NET_DEV_CTRL_PTZ_UP	0x3001	云台上仰
HI_NET_DEV_CTRL_PTZ_DOWN	0x3002	云台下俯
HI_NET_DEV_CTRL_PTZ_LEFT	0x3003	云台左转
HI_NET_DEV_CTRL_PTZ_RIGHT	0x3004	云台右转
HI_NET_DEV_CTRL_PTZ_ZOOMIN	0x3005	焦距变大(倍率变大)
HI_NET_DEV_CTRL_PTZ_ZOOMOUT	0x3006	焦距变小(倍率变小)
HI_NET_DEV_CTRL_PTZ_FOCUSIN	0x3007	焦点前调
HI_NET_DEV_CTRL_PTZ_FOCUSOUT	0x3008	焦点后调
HI_NET_DEV_CTRL_PTZ_APERTUREIN	0x3009	光圈变小
HI_NET_DEV_CTRL_PTZ_APERTUREOUT	0x3010	光圈变大

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

Remarks

云台控制扩展用于单步执行单步移动。

HI_NET_DEV_PTZ_Ctrl_Preset

云台预置点操作

```
HI_S32 HI_NET_DEV_PTZ_Ctrl_Preset (  
    HI_U32      u32Handle
```

```
HI_U32    u32Command,  
HI_U32    u32Preset  
);
```

Parameters

u32Handle

[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_NET_DEV_CTRL_PTZ_GOTO_PRESET	0x3015	转到预置点
HI_NET_DEV_CTRL_PTZ_SET_PRESET	0x3016	设置预置点
HI_NET_DEV_CTRL_PTZ_CLE_PRESET	0x3017	清除预置点

u32Preset

[IN] 预置点

```
#define HI_NET_DEV_CTRL_PTZ_PRESET_MAX 255
```

```
#define HI_NET_DEV_CTRL_PTZ_PRESET_MIN 0
```

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。

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

HI_NET_DEV_PTZ_Ctrl_Extend

云台控制扩展

```
HI_S32 HI_NET_DEV_PTZ_Ctrl_Extend (  
    HI_U32    u32Handle  
    HI_U32    u32Command,  
);
```

Parameters

u32Handle

[IN] 操作句柄

u32Command

[IN] 云台控制命令

宏定义	宏定义值	含义
HI_NET_DEV_CTRL_PTZ_LIGHT_ON	0x3021	灯光开
HI_NET_DEV_CTRL_PTZ_LIGHT_OFF	0x3022	灯光关
HI_NET_DEV_CTRL_PTZ_WIPER_ON	0x3023	雨刷开
HI_NET_DEV_CTRL_PTZ_WIPER_OFF	0x3024	雨刷关
HI_NET_DEV_CTRL_PTZ_AUTO_ON	0x3025	自动开
HI_NET_DEV_CTRL_PTZ_AUTO_OFF	0x3026	自动关

HI_NET_DEV_CTRL_PTZ_HOME	0x3027	回到原点
HI_NET_DEV_CTRL_PTZ_CRUISE_V	0x3028	上下巡航
HI_NET_DEV_CTRL_PTZ_CRUISE_H	0x3029	左右巡航

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

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_NET_DEV_PTZ_Fully_Trans

透明云台操作

```
HI_S32 HI_NET_DEV_PTZ_Fully_Trans (
    HI_U32      u32Handle
    HI_CHAR*    psBuf,
    HI_U32      u32BufLen
);
```

Parameters

u32Handle
[IN] 操作句柄

psBuf
[IN] 存放云台控制码缓冲区的指针

u32BufLen
[IN] 云台控制码的长度，
#define HI_NET_DEV_CTRL_PTZ_FT_BUF_LEN 128

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

5.7 对讲

HI_NET_DEV_StartVoice

透明云台操作

```
HI_S32 HI_NET_DEV_StartpVoice (
    HI_U32      u32Handle,
    HI_U32      u32AudioType
);
```

Parameters

u32Handle
[IN] 操作句柄

u32AudioType

```

struAAttr.lBitRate      = 64000;
struAAttr.lSamplesPerSec = 8000;
struAAttr.lBitsPerSample = 8;
struAAttr.lBlockAlign   = 1;
struAAttr.lChannels      = 1;
struAAttr.length         = 0;
struAAttr.lFrameFlag     = 0;
struAAttr.pReserved     = NULL;

```

```

struAAttr.lBitRate      = 16000;
struAAttr.lSamplesPerSec = 8000;
struAAttr.lBitsPerSample = 2;
struAAttr.lBlockAlign   = 1;
struAAttr.lChannels      = 1;
struAAttr.length         = 0;
struAAttr.lFrameFlag     = 0;
struAAttr.pReserved      = NULL;

```

```

        HI_U32      u32Handle
    );

```

```

        HI_U32    u32Handle
        HI_CHAR   *psBuf,
        HI_U32    u32BufLen,
        HI_U64    u64Pts
    );

```

Parameters

u32Handle
[IN] 操作句柄

psBuf
[IN] 音频数据

u32BufLen
[IN] 数据长度的长度，

u64Pts
[IN] 时间戳

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

Remarks

发送给摄像机的音频格式要求与摄像机端音频的类型相一致，如：如果摄像机端音频类型为 G711，发送过去的音频为 G711 压缩 8K、64 位、单声道数据；如果摄像机端音频类型为 G726，发送过去的音频为 G726 压缩 8K、16 位、单声道数据。

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

5.8 录像抓拍

回调函数回调出来的复合流可以直接录制到文件当中，文件的播放可以用播放器直接播放。复合录像为自定义格式，通过网络库获取的数据保存为文件即可，具体格式如下：

Example:

```

FILE *pFile = NULL;
HI_S32 OnStreamCallback(           //网络库实时数据回调
    HI_U32 u32Handle,              //句柄
    HI_U32 u32DataType,            //数据类型，视频或音频数据或音视频复合数据
    HI_U8* pu8Buffer,              //数据包含帧头
    HI_U32 u32Length,              //数据长度
    HI_VOID* pUserData)            //用户数据
{
    if (u32DataType == HI_NET_DEV_SYS_DATA)
    {
        if (pFile != NULL)

```

```

    {
        fclose(pFile);
        pFile = NULL;
    }

    pFile = fopen("d://video1.hx", "ab+");
    if (NULL == pFile)
        return;

    fwrite(pu8Buffer, 1, u32Length, pFile);
}
else if (u32DataType == HI_NET_DEV_AV_DATA)
{
    if (NULL != pFile)
        fwrite(pu8Buffer, 1, u32Length, pFile);
}
}

```

其中 u32DataType 类型为 HI_NET_DEV_SYS_DATA 时为流信息 HI_S_SysHeader，为 HI_NET_DEV_AV_DATA 时数据中包含了帧头信息 HI_S_AVFrame 和数据块，数据块的长度和 HI_S_AVFrame 的 u32AVFrameLen 一致，保存后文件结构如下：

```

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

```

注：上述例子是在有数据就开始录像，如果想要在特定时间段录像可以将 HI_S_SysHeader 结构体保存下来，录像开始时将 HI_S_SysHeader 保存到文件的最前面，然后再存储实时流数据即可。

回调数据请查阅 HI_NET_DEV_SetStreamCallBack 函数的使用。

自定义格式录像在播放库中打开可以直接播放。更详细的录像调用请参阅播放库中的 DEMO。

HI_NET_DEV_StartRecord

AVI 录像

```

HI_S32 HI_NET_DEV_StartRecord (
    HI_U32      u32Handle,
    HI_CHAR*    psPath,

```

```
HI_U32    u32Type,  
HI_U32    u32Flag  
);
```

Parameters

u32Handle
[IN] 操作句柄

psPath
[IN] 路径+文件名

u32Type
[IN] 录像类型

#define HI_NET_DEV_VIDEO_AVI 0 //AVI
#define HI_NET_DEV_VIDEO_ASF 1 //ASF
#define HI_NET_DEV_VIDEO_264 2 //自定义格式

u32Flag
[IN] 保留字段，可直接填写为 0

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

HI_NET_DEV_StopRecord

停止 AVI 录像

```
HI_S32 HI_NET_DEV_StopRecord (  
    HI_U32    u32Handle  
);
```

Parameters

u32Handle
[IN] 操作句柄

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

HI_NET_DEV_GetRecordState

获取录像状态

```
HI_S32 HI_NET_DEV_GetRecordState (  
    HI_U32    u32Handle  
);
```

Parameters

u32Handle
[IN] 操作句柄

Return Values

HI_SUCCESS 表示正在录像，HI_FAILURE 表示没有录像。

HI_NET_DEV_SnapJpeg

网络抓拍

```
HI_S32 HI_NET_DEV_SnapJpeg (
    HI_U32      u32Handle,
    HI_U8*      pu8Data,
    HI_S32      s32BufLen,
    HI_S32      *pSize
);
```

Parameters

u32Handle

[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_NET_DEV_Login](#))成功后即可使用，申请的内存存在外部进行，申请内存大小不能小于：

```
#define HI_NET_DEV_SNAP_BUF_LEN_MIN 1024
```

具体用法如下：

```
char *sData = (char*)malloc(1024*1024);
int nSize = 0;
s32Ret = HI_NET_DEV_SnapJpeg(m_uiHandle, (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;
```

5.9 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_NET_DEV_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_NET_DEV_SetChannel

设置当前操作通道

```
HI_S32 HI_NET_DEV_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_NET_DEV_GetChannel

获取当前操作通道

```
HI_S32 HI_NET_DEV_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

5.10 解码器

解码器调用顺序

```

HI_NET_DEV_Login           //登陆解码器
HI_NET_DEV_SetDataCallBack //设置状态回调
HI_NET_DEV_StartStream     //开始解码器，保持连接和获取当前状态
.....
HI_NET_DEV_StartDec        //解码器操作
.....
HI_NET_DEV_StopStream      //停止
HI_NET_DEV_Logout          //取消登陆

```

解码器设置状态回调的目的是返回当前解码器各个通道中的状态，如回调中 u32DataType 的值为 3，字符串为 012000000 表示第一通道当前没有解码，第二通道当前为动态解码，第三通道当前正在轮巡，即字符串的每一个字节表示为：0 没有解码，1 动态解码，2 轮巡，回调的频率是每 0.5 秒一次。

另外可以调用 HI_NET_DEV_SetEventCallBack 实时联系与解码器的连接状态。

HI_NET_DEV_GetDisplayCfg

获取解码器系统信息

```

HI_S32 HI_NET_DEV_GetDisplayCfg (
    HI_U32          u32Handle,
    HI_S_DISPLAY_CFG *pDisplayCfg
);

```

Parameters

u32Handle
[IN] 操作句柄

pDisplayCfg
[OUT] 解码器信息

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

Struct

```

typedef struct hiDISPLAY_CFG
{
    VIDEO_MODE_E eVideoMode;           // CVBS[0 或 1],VGA[9 或 10]
    PICTURE_NUM_E ePictureNum;         //通道分割
    DISPLAY_TYPE_E eDpyType;           //显示设备 VGA 或 CVBS
    DISPLAY_MODE_E eDpyMode;           // P、N 制
    DISPLAY_FLAG_E eDpyFlag;           //显示模式：全屏或按比例显示
} HI_S_DISPLAY_CFG;

```

```

typedef enum{
    PICTURE_NUM_1 = 1,    //单画面分割
    PICTURE_NUM_4 = 4,    //4 画面分割
    PICTURE_NUM_9 = 9,    //9 画面分割
    PICTURE_NUM_BUTT
}PICTURE_NUM_E;

typedef enum{
    DISPLAY_TYPE_VGA = 0,    //VGA
    DISPLAY_TYPE_CVBS = 2,    //CVBS
    DISPLAY_TYPE_BUTT
}DISPLAY_TYPE_E;

typedef enum{
    DISPLAY_MODE_PAL = 25,    //P 制
    DISPLAY_MODE_NTSC = 30,    //N 制
    DISPLAY_MODE_BUTT
}DISPLAY_MODE_E;

typedef enum{
    VIDEO_MODE_PAL          = 0,    // P 制，只有在 CVBS 才能设置
    VIDEO_MODE_NTSC         = 1,    // N 制，只有在 CVBS 才能设置

    VIDEO_MODE_1280x1024_60 = 9,    //只有在 VGA 才能设置
    VIDEO_MODE_1366x768_60  = 10,    //只有在 VGA 才能设置
    VIDEO_MODE_BUTT
}VIDEO_MODE_E; /*CVBS[0-1],VGA[9-10]*/

typedef enum{
    DISPLAY_FLAG_AUTO = 0,    //自动按比例显示
    DISPLAY_FLAG_FIXED = 1,    //画面满屏显示
    DISPLAY_FLAG_BUTT
}DISPLAY_FLAG_E;

```

HI_NET_DEV_SetDisplayCfg

设置解码器系统信息

```

HI_S32 HI_NET_DEV_SetDisplayCfg (
    HI_U32          u32Handle,
    HI_S_DISPLAY_CFG *pDisplayCfg
);

```

Parameters

u32Handle
[IN] 操作句柄

pDisplayCfg
[IN] 解码器信息

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

HI_NET_DEV_StartDec

启用动态解码

```
HI_S32 HI_NET_DEV_StartDec (  
    HI_U32          u32Handle,  
    HI_U32          u32Channel,  
    HI_S_CHN_INFO   *pDevInfo  
);
```

Parameters

u32Handle
[IN] 操作句柄
u32Channel
[IN] 解码器通道
pDevInfo
[IN] 解码通道相关参数

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

Struct

```
typedef struct HI_CHN_INFO  
{  
    HI_U32 u32Enable;           //启动与否标志 0-禁用，1-启用  
    HI_CHAR sHost[24];         //IP 地址  
    HI_BOOL bStream;           //码流  
    HI_U32 u32Port;            //端口  
    HI_U32 u32Chn;             //设备通道  
    HI_CHAR sUsername[32];     //用户名  
    HI_CHAR sPassword[32];     //密码  
}HI_S_CHN_INFO;
```

Remarks

开启通道动态解码，如果断线，解码器将自动重连，调用停止动态解码接口停止动态解码，调用轮巡接口将停止动态解码开始轮巡。

HI_NET_DEV_StopDec

停止动态解码

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

Parameters

u32Handle
[IN] 操作句柄

u32Channel
[IN] 解码器通道

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

HI_NET_DEV_GetLoopDecChnInfo

获取轮巡解码通道参数

```
HI_S32 HI_NET_DEV_GetLoopDecChnInfo (
    HI_U32          u32Handle,
    HI_U32          u32Channel,
    HI_S_LOOP_INFO  *pLoopInfo
);
```

Parameters

u32Handle
[IN] 操作句柄

u32Channel
[IN] 解码器通道

pLoopInfo
[OUT] 轮巡的解码通道参数

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

Struct

```
#define MAX_LOOP_CHN 36
typedef struct HI_LOOP_INFO
{
    HI_U32 u32LoopTime;    //轮巡间隔时间, 单位毫秒
    HI_S_CHN_INFO  sChnInfo[MAX_LOOP_CHN]; //轮巡列表
}HI_S_LOOP_INFO;
```

Remarks

获取轮巡解码通道参数，每个通道最多有 36 路设备进行轮巡

HI_NET_DEV_SetLoopDecChnInfo

设置轮巡解码通道参数

```
HI_S32 HI_NET_DEV_SetLoopDecChnInfo (  
    HI_U32          u32Handle,  
    HI_U32          u32Channel,  
    HI_S_LOOP_INFO  *pLoopInfo  
);
```

Parameters

u32Handle
[IN] 操作句柄

u32Channel
[IN] 解码器通道

pLoopInfo
[IN] 轮巡的解码通道参数

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

HI_NET_DEV_GetLoopDecChnEnable

获取单个解码通道轮巡开关

```
HI_S32 HI_NET_DEV_GetLoopDecChnEnable (  
    HI_U32          u32Handle,  
    HI_U32          u32Channel,  
    HI_U32          *pu32Enable  
);
```

Parameters

u32Handle
[IN] 操作句柄

u32Channel
[IN] 解码器通道

pu32Enable
[OUT] 0 表示关闭，1 表示打开

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

HI_NET_DEV_SetLoopDecChnEnable

设置单个解码通道轮巡开关

```
HI_S32 HI_NET_DEV_SetLoopDecChnEnable (  
    HI_U32          u32Handle,  
    HI_U32          u32Channel,
```

```
HI_U32      u32Enable
);
```

Parameters

u32Handle
[IN] 操作句柄

u32Channel
[IN] 解码器通道

u32Enable
[IN] 0 表示关闭，1 表示打开

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

HI_NET_DEV_GetLoopDecEnable

获取所有解码通道轮巡开关

```
HI_S32 HI_NET_DEV_GetLoopDecEnable (
    HI_U32      u32Handle,
    HI_CHAR      *psEnable,
    HI_U32      s32BufLen
);
```

Parameters

u32Handle
[IN] 操作句柄

psEnable
[OUT] 轮巡开关，传入的 HI_CHAR 的长度必须大于 9，如 char sEnable[9]，其中 sEnable[0]表示第一通道的轮巡开关、sEnable[1]表示第二通道的轮巡开关..... 如返回为 010000000，第二通道正在轮巡，其他通道轮巡处于关闭状态。

s32BufLen
[IN] psEnable 缓存长度

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

HI_NET_DEV_GetChnInfo

获取当前解码通道信息

```
HI_S32 HI_NET_DEV_GetChnInfo (
    HI_U32      u32Handle,
    HI_U32      u32Channel,
    HI_S_DEC_CHN_INFO *pDecChnInfo
);
```


Parameters

u32Handle
[IN] 操作句柄

u32Channel
[IN] 解码器通道

pDecChnInfo
[OUT] 解码通道信息

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

Struct

```
typedef struct HI_DEC_CHN_INFO
{
    HI_S_CHN_INFO  sChnInfo;    //通道相关信息
    HI_U32 u32DecState;          //解码通道状态：0-不解码 1-动态解码 2-循环解
                                //码 3-按时间回放 4-按文件回放
    HI_S_TIME sStartTime;        //按时间回放开始时间
    HI_S_TIME sStopTime;         //按时间回放结束时间
    HI_CHAR sFileName[128];      //按文件回放文件名
}HI_S_DEC_CHN_INFO;

typedef struct HI_TIME
{
    HI_U32  u32Year;             //年
    HI_U32  u32Month;            //月
    HI_U32  u32Day;              //日
    HI_U32  u32Hour;             //时
    HI_U32  u32Minute;           //分
    HI_U32  u32Second;           //秒
}HI_S_TIME;
```

HI_NET_DEV_GetDecChnEnable

获取当前解码通道开关

```
HI_S32 HI_NET_DEV_GetDecChnEnable (
    HI_U32          u32Handle,
    HI_U32          u32Channel,
    HI_U32          *pu32Enable
);
```

Parameters

u32Handle
[IN] 操作句柄

u32Channel

[IN] 解码器通道

pu32Enable

[OUT] 0 表示关闭；1 表示打开

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

HI_NET_DEV_SetDecChnEnable

设置当前解码通道开关

```
HI_S32 HI_NET_DEV_SetDecChnEnable (  
    HI_U32          u32Handle,  
    HI_U32          u32Channel,  
    HI_U32          u32Enable  
);
```

Parameters

u32Handle

[IN] 操作句柄

u32Channel

[IN] 解码器通道

u32Enable

[IN] 0 表示关闭；1 表示打开

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

HI_NET_DEV_StartPassiveDecode

解码器推送方式开启

```
HI_S32 HI_NET_DEV_StartPassiveDecode (  
    HI_U32          u32Handle,  
    HI_U32          u32Channel,  
    HI_U32*         pu32PassiveHandle  
);
```

Parameters

u32Handle

[IN] 操作句柄

u32Channel

[IN] 解码器通道，从 1 开始

pu32PassiveHandle

[OUT] 返回码流传输的句柄

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

HI_NET_DEV_StopPassiveDecode

解码器推送方式关闭

```
HI_S32 HI_NET_DEV_StopPassiveDecode (
    HI_U32          u32PassiveHandle
);
```

Parameters

u32PassiveHandle

[IN] HI_NET_DEV_StartPassiveDecode 返回的句柄

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

Remarks

停止解码器推送必须保证 HI_NET_DEV_DequeueSendData 已经停止工作。

HI_NET_DEV_DequeueSendData

解码器推送方式数据接口

```
HI_S32 HI_NET_DEV_DequeueSendData (
    HI_U32          u32PassiveHandle,
    HI_U8           *pSendBuf,
    HI_U32          u32BufSize
);
```

Parameters

u32PassiveHandle

[IN] HI_NET_DEV_StartPassiveDecode 返回的句柄

pSendBuf

[IN] 传输的 buffer 数据

u32BufSize

[IN] 传输 buffer 数据大小

Return Values

HI_SUCCESS 表示成功，HI_FAILURE 表示失败。

Remarks

数据发送格式：

```
typedef struct hiFrmHeader
```

```
{
```

```
    HI_U32 u32AVFrameFlag; //视频标志 HI_NET_DEV_VIDEO_FRAME_FLAG
```

```

//音频标志 HI_NET_DEV_AUDIO_FRAME_FLAG
HI_U32 u32AVFrameLen; //数据长度
HI_U32 u32AVFramePTS; //数据时间戳，暂时无效
HI_U32 u32VFrameType; //视频帧标志：
// I 帧：HI_NET_DEV_VIDEO_FRAME_I
// P 帧：HI_NET_DEV_VIDEO_FRAME_P
HI_U32 u32Width; //视频宽
HI_U32 u32Height; //视频高
HI_U32 u32Format; //音频格式，解码器暂时不支持音频
} HI_S_FrmHeader;

```

用法，解码器的推送方式必须按照指定格式发送，可以将帧头信息与帧数据分开发送，也可以组合发送，如：

```

HI_S_FrmHeader sSendFrame;
sSendFrame.u32AVFrameFlag = HI_NET_DEV_VIDEO_FRAME_FLAG;
sSendFrame.u32AVFrameLen = buflen;
sSendFrame.u32AVFramePTS = 0;
sSendFrame.u32VFrameType = HI_NET_DEV_VIDEO_FRAME_I;
sSendFrame.u32Width = 704;
sSendFrame.u32Height = 576;

```

分开发送：

```

s32Ret = HI_NET_DEV_DecodeSendData(u32PassiveHandle,
                                   (HI_U8*)&sSendFrame, sizeof(HI_S_FrmHeader));
if(s32Ret == HI_SUCCESS)
{
    s32Ret = HI_NET_DEV_DecodeSendData(u32PassiveHandle, pu8Buffer, buflen);
}

```

组合发送：

```

memcpy(u8Buf, &sSendFrame, sizeof(HI_S_FrmHeader));
memcpy(u8Buf+sizeof(HI_S_FrmHeader), pu8Buffer, buflen);
s32Ret = HI_NET_DEV_DecodeSendData(u32PassiveHandle, u8Buf,
                                   buflen+ sizeof(HI_S_FrmHeader));

```

解码器其他相关接口

重启解码器

```
HI_NET_DEV_SetConfig (u32Handle, HI_NET_DEV_CMD_REBOOT, NULL, 0);
```

获取设置解码器网络信息

```

HI_S_NET_EXT sNetExt;
HI_NET_DEV_GetConfig(u32Handle, HI_HI_NET_DEV_CMD_NET_EXT,
                    &sNetExt, sizeof(HI_S_NET_EXT));
HI_NET_DEV_SetConfig(u32Handle, HI_HI_NET_DEV_CMD_NET_EXT,

```

```
&sNetExt, sizeof(HI_S_NET_EXT));
```

获取设置解码器用户信息

```
HI_S_USER sUser;
HI_NET_DEV_GetConfig(m_u32Handle, HI_HI_NET_CMD_USER,
    &sUser, sizeof(HI_S_USER));
HI_NET_DEV_SetConfig(m_u32Handle, HI_HI_NET_CMD_USER,
    &sUser, sizeof(HI_S_USER));
```

5.11 AVI 文件解析

AVI 解析调用顺序

AVI_CreateReader	//创建文件解析句柄
AVI_ReadFileInfo	//读取 AVI 文件信息
AVI_ReadFrame	//读取 AVI 文件帧信息
AVI_DestroyReader	//销毁文件解析句柄

AVI 解析接口错误定义

#define ERR_AVI_OK	0x00000000	//成功
#define ERR_AVI_INVALIDARG	0x80000001	//参数错误
#define ERR_AVI_MALLOC	0x80000003	//分配内存失败
#define ERR_AVI_OUTOFSEEKTIME	0x8000000a	//跳转超过总时间范围
#define ERR_AVI_OPEN_FILE	0x80000012	//文件打开失败
#define ERR_AVI_ENDFILE	0x80000013	//读取帧数据失败
#define ERR_AVI_READ_FRAME	0x80000014	
#define ERR_AVI_INVALID_STREAM	0x80000015	
#define ERR_AVI_WRITE_FRAME	0x80000016	
#define ERR_AVI_READ_HEADER	0x80000017	//解析 AVI 文件头失败
#define ERR_AVI_WRITE_HEADER	0x80000018	

AVI_CreateReader

创建 AVI 解析句柄

```
HI_S32 AVI_CreateReader (
    HI_U32*    pAVIHandle,
    HI_U8*    pu8FileName
);
```

Parameters

pAVIHandle
[OUT] 返回操作句柄

pu8FileName

[IN] AVI 文件路径

Return Values

ERR_AVI_OK 表示成功，失败返回错误代码。

AVI_DestroyReader

销毁 AVI 解析句柄

```
HI_S32 AVI_DestroyReader (
    HI_U32          u32Handle
);
```

Parameters

u32Handle

[IN] 操作句柄

Return Values

ERR_AVI_OK 表示成功，失败返回错误代码。

AVI_ReadFrame

读取音视频帧信息

```
HI_S32 AVI_ReadFrame (
    HI_U32          u32Handle,
    AVI_FRAME_S     *pFrame
);
```

Parameters

u32Handle

[IN] 操作句柄

pFrame

[OUT] 帧信息

```
#define AVI_VIDEO_FRAME_FLAG 0x1
#define AVI_AUDIO_FRAME_FLAG 0x2
#define AVI_FRAME_KEY_P      0
#define AVI_FRAME_KEY_I      1
typedef struct hiAVI_FRAME_S
{
    HI_U64 u64Pts;           //时间戳
    HI_U64 u64Dts;           //
    HI_U8* pu8Data;          //帧数据
    HI_U32 u32Size;          //帧大小
    HI_U32 u32Type;          //帧类型
```

```

        HI_U32 u32KeyFlags; //视频帧 I 帧， P 帧
        HI_U32 u32Duration; //
    }AVI_FRAME_S;

```

Return Values

ERR_AVI_OK 表示成功，失败返回错误代码。

AVI_SeekFrame

时间跳转

```

HI_S32 AVI_SeekFrame (
    HI_U32          u32Handle,
    HI_S32          s32Pts
);

```

Parameters

u32Handle

[IN] 操作句柄

s32Pts

[IN] 跳转的时间，单位毫秒，值不能大于总时间

Return Values

ERR_AVI_OK 表示成功，失败返回错误代码。

AVI_ReadFileInfo

读取 AVI 相关信息

```

HI_S32 AVI_ReadFileInfo (
    HI_U32          u32Handle,
    AVI_INFO_S      *sAviInfo
);

```

Parameters

u32Handle

[IN] 操作句柄

sAviInfo

[OUT] AVI 文件相关信息

Return Values

ERR_AVI_OK 表示成功，失败返回错误代码。

Struct

//文件总时间

```

typedef struct hiAVI_DURATION_S
{

```

```
        HI_S32 s32Hours;        //时
        HI_S32 s32Mins;        //分
        HI_S32 s32Secs;        //秒
    } AVI_DURATION_S;

//视频信息
#define AVI_VIDEO_FORMAT_H264 0x00
typedef struct hiAVI_VSTREAM_S
{
    HI_U16 u16FormatTag;        //格式, AVI_VIDEO_FORMAT_H264
    HI_U16 u32FrameRate;        //帧率
    HI_U32 u32Width;            //宽
    HI_U32 u32Height;           //高
} AVI_VSTREAM_S;

//音频信息
#define AVI_AUDIO_FORMAT_G711A 0x00
#define AVI_AUDIO_FORMAT_G726 0x01
typedef struct hiAVI_ASTREAM_S
{
    HI_U16 u16FormatTag;        //音频格式, G711 G726
    HI_U16 u16Channels;         //声道
    HI_U32 u32SamplesPerSec;
    HI_U32 u32AvgBytesPerSec;
    HI_U16 u16BlockAlign;
    HI_U16 u16BitsPerSample;
} AVI_ASTREAM_S;

typedef struct hiAVI_INFO_S
{
    AVI_DURATION_S struDuration;
    AVI_VSTREAM_S struVStream;
    AVI_ASTREAM_S struAStream;
} AVI_INFO_S;
```


六、音频编解码说明

Demo 提供了音频的编解码、PC 获取音频、播放音频和对讲实例，请查阅网络库提供的 VC_demo。

编码：提供 G711 和 G726 两种编码格式，两种编码的格式有区别；

相关类：CHI_AENC_ENC，CHI_AI_MM

解码：提供 G711 和 G726 两种解码格式。

相关类：CHI_ADEC_DEC，CHI_AO_MM

6.1 音频采集格式设置

声卡输入和输出的音频属性可定义如下：

```
m_waveformat.wFormatTag      = WAVE_FORMAT_PCM;
m_waveformat.nChannels       = 1;
m_waveformat.nSamplesPerSec  = 8000;
m_waveformat.wBitsPerSample  = 16;
m_waveformat.cbSize          = 0;
m_waveformat.nBlockAlign     = 2;
m_waveformat.nAvgBytesPerSec = 16000;
```

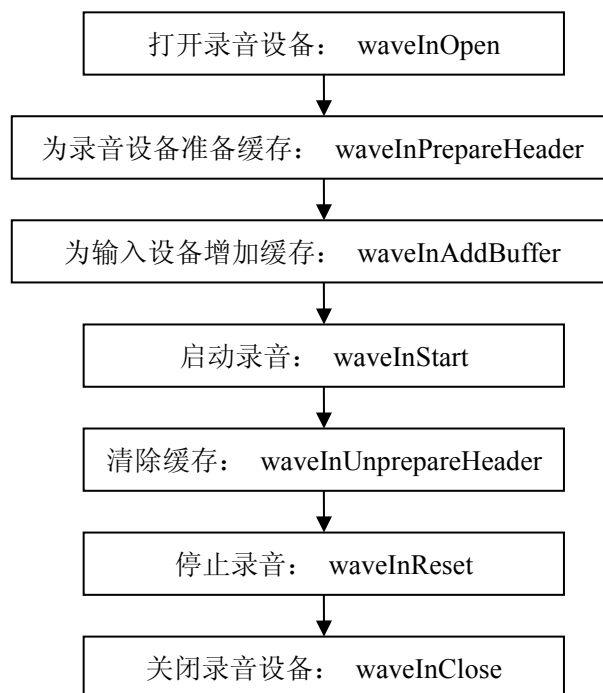
设置的音频格式类型是 PCM 格式，单通道，8000HZ 的采样率，每秒采集的数据大小为 16000bytes.其中，存在着下面的关系：

$nBlockAlign = nChannels * wBitsPerSample / 8 ;$

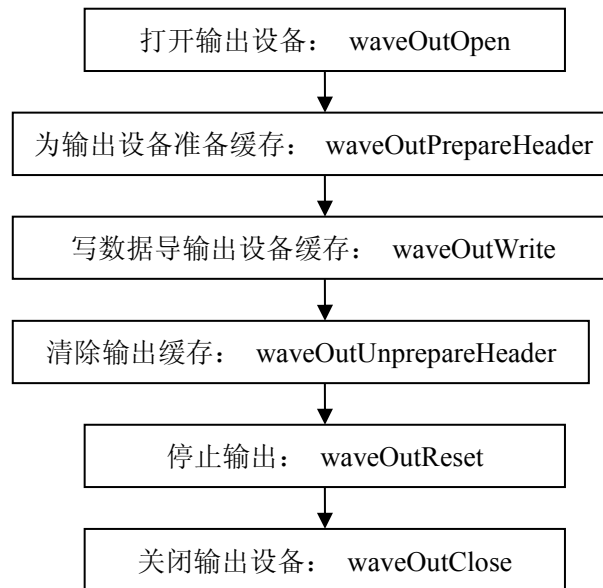
$nAvgBytesPerSec = nSamplesPerSec * nBlockAlign ;$

注：目前音频采集必须按照 320 字节单位采集，G711 和 G726 的编码库不支持 640 字节编码，所以采集一定要按照 320 个字节采集。

6.2 音频采集流程



6.3 音频播放流程



6.4 音频编码

设置音频编码库 hisi_voice_engine.lib，目前该库只支持格式有两种：G711A 和 G726。根据设置音频结构 PLAYER_ATTR_AUDIO_S 设置编码的格式并初始化编码库：

```

HRESULT CHI_AENC_ENC::HI_AENC_Init(PLAYER_ATTR_AUDIO_S *pAudioAttr)
{
    HI_S32 s32Rel;
    HI_U16 u16CodeType;

    if (NULL == pAudioAttr)
    {
        return HI_ERR_NULL_POINTER;
    }

    if (PLAYER_AUDIO_CODEC_FORMAT_G711A == pAudioAttr->eAEncode)
    {
        m_pAEncHandle = &g711_dec_state;
        u16CodeType = G711_A;
    }
    else if (PLAYER_AUDIO_CODEC_FORMAT_G711U == pAudioAttr->eAEncode)
    {
        m_pAEncHandle = &g711_dec_state;
        u16CodeType = G711_U;
    }
    else if (PLAYER_AUDIO_CODEC_FORMAT_G726 == pAudioAttr->eAEncode)
    {
        m_pAEncHandle = &g726_dec_state;
    }
}
  
```

```

        if (16000 == pAudioAttr->lBitRate)
        {
            u16CodeType = MEDIA_G726_16KBPS;
        }
        else if (24000 == pAudioAttr->lBitRate)
        {
            u16CodeType = MEDIA_G726_24KBPS;
        }
        else if (32000 == pAudioAttr->lBitRate)
        {
            u16CodeType = MEDIA_G726_32KBPS;
        }
        else if (40000 == pAudioAttr->lBitRate)
        {
            u16CodeType = MEDIA_G726_40KBPS;
        }
        else
        {
            return HI_ERR_INVALID_ARGUMENTS;
        }
    }
    else
    {
        return HI_ERR_INVALID_ARGUMENTS;
    }

    /*重新初始化解码*/
    s32Rel = HI_VOICE_EncReset(m_pAEncHandle, u16CodeType);
    return (s32Rel == HI_SUCCESS) ? HI_SUCCESS : HI_FAILURE;
}

```

其中:

```

#define G711_A          0x01  /* 64kbps G.711 A, see RFC3551.txt  4.5.14 PCMA */
#define G711_U          0x02  /* 64kbps G.711 U, see RFC3551.txt  4.5.14 PCMU */
#define MEDIA_G726_16KBPS      0x24  /* G726 16kbps for ASF ... */
#define MEDIA_G726_24KBPS      0x25  /* G726 24kbps for ASF ... */
#define MEDIA_G726_32KBPS      0x26  /* G726 32kbps for ASF ... */
#define MEDIA_G726_40KBPS      0x27  /* G726 40kbps for ASF ... */

```

音频编码函数:

```

HI_VOICE_API HI_RESULT HI_VOICE_EncodeFrame (
    HI_VOID    *pEncState, // HI_VOICE_EncReset 的句柄
    HI_S16     *pInputBuf,  //没有编码的数据
    HI_S16     *pOutputBuf, //编码后数据
    HI_S16     len          //没有编码数据的大小
)

```

```
);
```

编码后数据的大小:

```
*pOutLen = (pOutBuf[2])*2 + 4; //码流数据加上海思帧结构头
```

6.5 音频解码

设置音频解码库 hisi_voice_engine.lib，目前该库只支持格式有两种：G711A 和 G726。根据设置音频结构 PLAYER_ATTR_AUDIO_S 设置解码的格式并初始化解码：

```
HRESULT CHI_ADEC_DEC::HI_ADEC_Init(PLAYER_ATTR_AUDIO_S *pAudioAttr,
                                     PLAYER_ATTR_AUDIO_S *pOutPutAttr)
{
    if (NULL == pAudioAttr || NULL == pOutPutAttr)
    {
        return HI_ERR_NULL_POINTER;
    }

    HI_U16 u16CodeType;
    if (PLAYER_AUDIO_CODEC_FORMAT_G711A == pAudioAttr->eAEncode)
    {
        m_pDecCodeState = &g711_dec_state;
        u16CodeType = G711_A;
    }
    else if (PLAYER_AUDIO_CODEC_FORMAT_G711U == pAudioAttr->eAEncode)
    {
        m_pDecCodeState = &g711_dec_state;
        u16CodeType = G711_U;
    }
    else if (PLAYER_AUDIO_CODEC_FORMAT_G726 == pAudioAttr->eAEncode)
    {
        m_pDecCodeState = &g726_dec_state;
        if (16000 == pAudioAttr->lBitRate)
        {
            u16CodeType = MEDIA_G726_16KBPS;
        }
        else if (24000 == pAudioAttr->lBitRate)
        {
            u16CodeType = MEDIA_G726_24KBPS;
        }
        else if (32000 == pAudioAttr->lBitRate)
        {
            u16CodeType = MEDIA_G726_32KBPS;
        }
        else if (40000 == pAudioAttr->lBitRate)
```

```

    {
        u16CodeType = MEDIA_G726_40KBPS;
    }
    else
    {
        return HI_ERR_INVALID_ARGUMENTS;
    }
}
else
{
    return HI_ERR_INVALID_ARGUMENTS;
}

/*重新初始化解码*/
HI_S32 s32Rel = HI_VOICE_DecReset(m_pDecCodeState, u16CodeType);
if (HI_SUCCESS != s32Rel)
{
    return HI_FAILURE;
}

return HI_SUCCESS;
}

```

其中：

```

#define G711_A          0x01  /* 64kbps G.711 A, see RFC3551.txt  4.5.14 PCMA */
#define G711_U          0x02  /* 64kbps G.711 U, see RFC3551.txt  4.5.14 PCMU */
#define MEDIA_G726_16KBPS      0x24  /* G726 16kbps for ASF ... */
#define MEDIA_G726_24KBPS      0x25  /* G726 24kbps for ASF ... */
#define MEDIA_G726_32KBPS      0x26  /* G726 32kbps for ASF ... */
#define MEDIA_G726_40KBPS      0x27  /* G726 40kbps for ASF ... */

```

音频解码函数：

```

HI_VOICE_DecodeFrame(
    HI_VOID    *pDecState, // HI_VOICE_DecReset 的句柄
    HI_S16      *pInputBuf, //没有解码的数据
    HI_S16      *pOutputBuf, //解码后数据
    HI_S16      *pLen        //解码后数据大小
);

```

七、附录

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；
- Linux_demo 存放 Linux Demo 文件，有 GTK 界面和无命令界面；
- VC_demo 存放 mfc Demo；
- Player 录像文件播放器；
- Bin 执行文件存放路径。

附录 II 、Linux Demo 使用说明

将 libNetLib.so 拷贝到/lib 或/usr/lib 文件夹中，编译即可。如果使用 GTK 界面要求支持 GTK 环境。

编译 linux 程序要加入编译选项-DHI_OS_LINUX

附录III、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，第二个字节代表字段子类型。

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