**\_M\_INTERFACE 接口说明文档**

**V2.0**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本信息 | | | |
| 版本号 | 描述 | 日期 | 作者 |
| V1.1 | Create | 2017/10/02 | Leo |
| V2.0 | 修改为中文版本，并增加新增接口 | 2019/07/18 | Leo |
|  |  |  |  |

# 简介

HVP软件框架实现了IPC的全部功能， 内部包含20+模块， 模块间相互调用使用中间接口层机制， 该层提供数个函数指针， 框架层实现函数体， 模块开发如果需要用到其他模块的数据/方法， 可以通过该层接口完成。



# 接口说明

## void (\*IF\_WriteLog) (int nModuleType,int nLogLevel,unsigned long nEventType,const char \* szFormat,...);

**Function explain**: 该接口用来记录日志， EventType如果为0， 则记录在/mnt/mtd/log/debug.log。 如果为1， 则记录在/mnt/mtd/log/oper.log

**Parameter**:

|  |  |
| --- | --- |
| nModuleType | 模块名称，参见 \_NSD\_MODULE\_TYPE\_E (privydef.h) |
| nLogLevel | 日志级别，参见 \_NSD\_LOG\_LEVEL\_E (privydef.h) |
| nEventType | 日志类型，参见 \_NSD\_SUB\_MODULE\_TYPE\_E (privydef.h) |
| szFormat | Log info |

**Example:**

|  |
| --- |
| CVideoLib::GetInstance().IF\_WriteLog(NSDMOD\_VIDEO ,NSDLOG\_INFO ,0,"this is a demo"); |

## void (\*IF\_ModPluse)(int nModuleType);

**Function explain**:该接口为心跳接口，用来向软件看门狗发心跳，如果间隔大于30秒，软件看门狗则停止喂硬件狗， 从而导致重启设备

**Parameter**:

|  |  |
| --- | --- |
| nModuleType | Defined in \_NSD\_MODULE\_TYPE\_E (privydef.h) |

**Example**: NA

## int (\*IF\_GetParam)(NSD\_CONFIG\_TYPE\_E nParamType,void \* lpParam,unsigned long nCh,unsigned long nRsvd);

**Function explain**: 获取当前设置参数

**Parameter**:

|  |  |
| --- | --- |
| nParamType | 参数类型， 参见\_NSD\_CONFIG\_TYPE\_E (nsddefined.h) |
| lpParam | 参数结构指针， 结构类型需要和nParamType配套， 否则可能会造成指针越界 |
| nCh | 通道数，在部分参数类型中表示ID，例如IO，预置位等 |
| nRsvd | 预留参数 |

**Return:**

0: Success

非0: NSD\_ERRCODE\_E

**Example**:

|  |
| --- |
| N[SD\_SYS\_INFO\_S](#_NSD_SYS_INFO_S) stSysInfo={0};  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_SYS, &stSysInfo, 0, 0);  [NSD\_DEVICE\_S](#_NSD_DEVICE_S) stDevice={0};  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_DEVICE, &stDevice, 0, 0);  [NSD\_TIME\_S](#_NSD_TIME_S) stTimeCfg = {0};  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_TIME, &stTimeCfg, 0,0);  [NSD\_VIDEO\_CFG\_S](#_NSD_VIDEO_CFG_S) stVideoCfg = {0};  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_VIDEO, &stVideoCfg, nCh, 0);  [NSD\_AUDIO\_CFG\_S](#_NSD_AUDIO_CFG_S) stAudioCfg = {0};  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_AUDIO, &stAudioCfg, nCh, 0);  [NSD\_MOTION\_CFG\_S](#_NSD_MOTION_CFG_S) stMotionCfg = {0};  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_VMOTION, &stMotionCfg, nCh, 0);  [NSD\_DI\_CFG\_S](#_NSD_DI_CFG_S) stDICfg = {0};  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_DI, &stDICfg, nCh, 0);  [NSD\_NET\_CFG\_S](#_NSD_NET_CFG_S) netCfg;  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_NET,&netCfg,0,0);  [NSD\_USER\_CFG\_S](#_NSD_USER_CFG_S) stUserCfg = {0};  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_USER, &stUserCfg, i, 0);  [NSD\_USER\_GROUP\_CFG\_S](#_NSD_USER_GROUP_CFG_S) stGroupCfg = {0};  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_USERGROUP, &stGroupCfg, nId, 0);  [NSD\_UART\_CFG\_S](#_NSD_UART_CFG_S) serial;  COnvifLib::GetInstance().IF\_GetParam(NSDCFG\_UART,&serial,nChannel,0);  [NSD\_OSD\_CFG\_S](#_NSD_OSD_CFG_S) stOsdCfg;  int COnvifLib::GetInstance().IF\_GetParam(NSDCFG\_OSD,&stOsdCfg,nCh,nStream);  [NSD\_PTZ\_CFG\_S](#_NSD_PTZ_CFG_S) m\_ptz\_cfg;  CPtzLib::GetInstance().IF\_GetParam(NSDCFG\_PTZ, &m\_ptz\_cfg, 0, 0);    [NSD\_PPPOE\_CFG\_S](#_NSD_PPPOE_CFG_S) stPppoeCfg = {0};  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_PPPOE, &stPppoeCfg, 0, 0);  [NSD\_DDNS\_CFG\_S](#_NSD_DDNS_CFG_S) stDDNSCfg = {0};  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_DDNS, &stDDNSCfg, 0, 0);  [NSD\_ETH\_CFG\_S](#_NSD_ETH_CFG_S) stEthCfg = {0};  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_ETH, &stEthCfg, 0, 0);  [NSD\_NTP\_CFG\_S](#_NSD_NTP_CFG_S) stNtpCfg = {0};  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_NTP, &stNtpCfg, 0, 0);  [NSD\_UPNP\_CFG\_S](#_NSD_UPNP_CFG_S) stUpnp = {0};  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_UPNP, &stUpnp, 0, 0);  [NSD\_FTP\_CFG\_S](#_NSD_FTP_CFG_S) stFtpCfg = {0};  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_FTP, &stFtpCfg, 0, 0);  [NSD\_SMTP\_CFG\_S](#_NSD_SMTP_CFG_S) stSMTPCfg = {0};  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_SMTP, &stSMTPCfg, 0, 0);  [NSD\_MASK\_CFG\_S](#_NSD_MASK_CFG_S) stMaskCfg = {0};  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_VMASK, &stMaskCfg, nCh, 0);  [NSD\_IMG\_CFG\_S](#_NSD_IMG_CFG_S) stImg={0};  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_IMG, &stImg, 0, 0);  [NSD\_DO\_CFG\_S](#_NSD_DO_CFG_S) stDOCfg = {0};  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_DO, &stDOCfg, nCh, 0);  [NSD\_VOUT\_CFG\_S](#_NSD_VOUT_CFG_S) stVoutCfg = {0};  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_VOUT, &stVoutCfg, nCh, 0);  [NSD\_AWB\_CFG\_S](#_NSD_AWB_CFG_S) stAWB={0};  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_AWB, &stAWB, nCh, nProfile);  [NSD\_AE\_CFG\_S](#_NSD_AE_CFG_S) stAE={0};  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_AE, &stAE, nCh, nProfile);  NSD\_AF\_CFG\_S stAF={0};  CWebLib::GetInstance().IF\_GetParam(NSDCFG\_AF, &stAF, nCh, nProfile);  [NSD\_ISP\_CFG\_S](#_NSD_ISP_CFG_S) ispcfg;  COnvifLib::GetInstance().IF\_GetParam(NSDCFG\_ISP,&ispcfg,nChannel,nProfileId);  [NSD\_PLATFORM\_RTMP\_S](#_NSD_PLATFORM_RTMP_S) m\_rtmp\_param  IF\_GetParam(NSDCFG\_PLATFORM,&m\_rtmp\_param,0,PTTYPE\_RTMP);  [NSD\_PLATFORM\_ONVIF\_S](#_NSD_PLATFORM_ONVIF_S) m\_onvif;  IF\_GetParam(NSDCFG\_PLATFORM,&m\_onvif,0,PTTYPE\_ONVIF);  [NSD\_RTSP\_CFG\_S](#_NSD_RTSP_CFG_S) stRtsp = {0};  IF\_SetParam(NSDCFG\_RTSP, &stRtsp, 0, 0);  [NSD\_ROI\_CFG\_S](#_NSD_ROI_CFG_S) stROICfg={0};  IF\_GetParam(NSDCFG\_ROI, &stROICfg, nCh, nStream);  [NSD\_ACTION\_S](#_NSD_ACTION_S) stAction = {0};  IF\_GetParam(NSDCFG\_ACTION, &stAction, i, 0);  [NSD\_USER\_GROUP\_CFG\_S](#_NSD_USER_GROUP_CFG_S) stGroupCfg = {0};  IF\_GetParam(NSDCFG\_USERGROUP, &stGroupCfg, nId, 0);  [NSD\_SCHEME\_CFG\_S](#_NSD_SCHEME_CFG_S) stSchemeCfg = {0};  IF\_GetParam(NSDCFG\_SCHEME, &stSchemeCfg, nId, 0);  [NSD\_FILTER\_CFG\_S](#_NSD_FILTER_CFG_S) stFilterCfg = {0};  IF\_GetParam(NSDCFG\_FILTER, &stFilterCfg, 0, 0);  [NSD\_FISHEYE\_CFG\_S](#_NSD_FISHEYE_CFG_S) fisheye\_cfg={0};  IF\_GetParam(NSDCFG\_FISHEYE,&fisheye\_cfg,m\_nChn,0)  [NSD\_AI\_FACE\_S](#_NSD_AI_FACE_S) obj={0};  IF\_GetParam(NSDCFG\_AI\_FACE,&obj,m\_nChn,0)  [NSD\_AI\_S](#_NSD_AI_S) obj={0};  IF\_GetParam(NSDCFG\_AI,&obj,m\_nChn,0)  [NSD\_SCREEN\_CFG\_S](#_NSD_SCREEN_CFG_S) screen;  IF\_GetParam(NSDCFG\_SCREEN,&screen,0,0);  [NSD\_WIEGAND\_CFG\_S](#_NSD_WIEGAND_CFG_S) wiegandCfg;  IF\_GetParam(NSDCFG\_WIEGAND, &wiegandCfg, 0, 0);  [NSD\_RELAY\_CTL\_DOOR\_S](#_NSD_RELAY_CTL_DOOR_S) relayCtrlDoorCfg;  IF\_GetParam(NSDCFG\_RELAYDOOR, &relayCtrlDoorCfg, 0, 0);  [NSD\_HTTP\_CFG\_S](#_NSD_HTTP_CFG_S) httpCfg;  IF\_GetParam(NSDCFG\_HTTP, &httpCfg, 0, 0); |

## int (\*IF\_SetParam)(NSD\_CONFIG\_TYPE\_E nParamType,void \* lpParam,unsigned long nCh,unsigned long nRsvd);

**Function explain**: 设置参数

**Parameter**:

|  |  |
| --- | --- |
| nParamType | 参数类型 \_NSD\_CONFIG\_TYPE\_E (nsddefined.h) |
| lpParam | 参数结构指针， 结构类型需要和nParamType配套， 否则可能会造成指针越界 |
| nCh | 通道数，在部分参数类型中表示ID，例如IO，预置位等 |
| nRsvd | 预留参数 |

**Return:**

0: Success

非0: NSD\_ERRCODE\_E

**Example**:

使用和IF\_GetParam类似， 参见上述示例

## Int (\*IF\_GetDefaultParam)(NSD\_CONFIG\_TYPE\_E nParamType,void \* lpParam,unsigned long nCh,unsigned long nRsvd);

**Function explain**: 获取参数的默认值

**Parameter**:

|  |  |
| --- | --- |
| nParamType | 参数类型，参见\_NSD\_CONFIG\_TYPE\_E (nsddefined.h) |
| lpParam | 参数结构指针， 结构类型需要和nParamType配套， 否则可能会造成指针越界 |
| nCh | 通道数，在部分参数类型中表示ID，例如IO，预置位等 |
| nRsvd | 预留参数 |

**Return:**

0: Success

非0: NSD\_ERRCODE\_E

**Example**:

参考IF\_GetParam

## Int (\*IF\_VerifyParam)(NSD\_CONFIG\_TYPE\_E nParamType,void \* lpParam,unsigned short nCh,unsigned long nRsvd);

**Function explain**: 验证参数合法性， 如果参数不合法， 可以使用该接口把参数修正参数值(目前已废弃)

**Parameter**:

|  |  |
| --- | --- |
| nParamType | 参见\_NSD\_CONFIG\_TYPE\_E (nsddefined.h) |
| lpParam | 参数结构指针， 结构类型需要和nParamType配套， 否则可能会造成指针越界 |
| nCh | 通道数，在部分参数类型中表示ID，例如IO，预置位等 |
| nRsvd | 预留参数 |

**Return:**

0: 成功

非0: NSD\_ERRCODE\_E

**Example**:

**Notes：**

**已废弃**

## Int (\*IF\_OnParamChanged)(NSD\_CONFIG\_TYPE\_E nParamType,void \* lpParam,unsigned long nCh,unsigned long nRsvd);

**Function explain**: 参数改变后通知其他模块响应参数变化的接口

**Parameter**:

|  |  |
| --- | --- |
| nParamType | 参见 \_NSD\_CONFIG\_TYPE\_E (nsddefined.h) |
| lpParam | 参数结构指针， 结构类型需要和nParamType配套， 否则可能会造成指针越界 |
| nCh | 通道数，在部分参数类型中表示ID，例如IO，预置位等 |
| nRsvd | 预留参数 |

**Return:**

0: 成功

非0: NSD\_ERRCODE\_E

**Example**:

CParamLib::GetInstance().IF\_OnParamChanged(NSDCFG\_SMTP,lpSmtpCfg,nChn,0);

**Notes：**

**该接口为Param模块内部使用， 其他模块请勿调用**

## Int (\*IF\_UpdateNetworkInfo)(int nWorkType,unsigned long nIp, unsigned long nSubMask, unsigned long nGateWay,unsigned long nDns1, unsigned long nDns2);

**Function explain**: 网络参数更新后，通过该接口通知参数模块更新IP地址，子网掩码等网络参数

**Parameter**:

|  |  |
| --- | --- |
| nWorkType | 始终为 0 |
| nIp | IP地址 |
| nSubMask | 子网掩码 |
| nGateWay | 网关 |
| nDns1 | 主DNS |
| nDns2 | 备用DNS |

**Return:**

0: 成功

非0: NSD\_ERRCODE\_E

**Example**:

**Notes：**

**该接口为Eth模块内部使用， 其他模块请勿调用**

## Int (\*IF\_UpdatePPPOEInfo)(unsigned long \* lpPppoeAddr,unsigned char \* lpStatus,unsigned long \* lpPppoeTimes);

**Function explain**: 当PPPoE连接成功后， 通知参数模块更新参数

**Parameter**:

|  |  |
| --- | --- |
| lpPppoeAddr | IP地址 |
| lpStatus | PPPoE状态 |
| lpPppoeTimes | 拨号次数 |

**Return:**

0: 成功

非0: NSD\_ERRCODE\_E

**Example**:

**Notes：**

**该接口为Misc模块内部使用， 其他模块请勿调用**

## Int (\*IF\_UpdateUpnpInfo)(unsigned short \* lpHttpPort,unsigned short \* lpDataPort);

**Function explain**: 当Upnp映射成功后， 通知参数模块更新Http端口和数据命令端口

**Parameter**:

|  |  |
| --- | --- |
| lpHttpPort | HTTP端口 |
| lpDataPort | 数据端口 |

**Return:**

0: 成功

非0: NSD\_ERRCODE\_E

**Example**:

**Notes：**

**该接口为Misc模块内部使用， 其他模块请勿调用**

## Int (\*IF\_ProcCmd)(NSD\_CMD\_TYPE\_E nCmdType,void \* lpData,unsigned long nDataLen,unsigned long nCh,unsigned long nRsvd);

**Function explain**: 该接口可以执行PTZ动作，重启，复位，强制I帧，SD卡格式化，Mipi屏幕亮度等动作。 目前支持的动作列表为

NSDCMD\_REBOOT: 重启设备

NSDCMD\_DEFAULT\_PARAM: 参数复位

NSDCMD\_FORCE\_KEYFRAME: 强制I帧

NSDCMD\_PTZ\_CTRL: 控制PTZ

NSDCMD\_IO\_CTRL: 控制IO输出

NSDCMD\_PTZ\_GET\_POSITION: 获取PTZ的绝对位置

NSDCMD\_PTZ\_SET\_POSITION: 设置PTZ的绝对位置

NSDCMD\_DO\_STATUS: 控制报警输出

NSDCMD\_BALL\_3D: 球机3D

NSDCMD\_SDCARD\_FORMAT: SD卡格式化

NSDCMD\_RESET\_AI: 复位ai模块

NSDCMD\_SCREEN\_LUMA: 调节MIPI屏的屏幕亮度

**Parameter**:

|  |  |
| --- | --- |
| nCmdType | 参见 NSD\_CMD\_TYPE\_E |
| lpData | 数据 |
| nDataLen | 数据长度 |
| nCh | 通道ID |
| nRsvd | 预留 |

**Return:**

0: 成功

非0: NSD\_ERRCODE\_E

**Example**:

|  |
| --- |
| 重启设备:  CWebLib::GetInstance().IF\_ProcCmd(NSDCMD\_REBOOT, NULL,0, 0, 0);  复位所有参数:  COnvifLib::GetInstance().IF\_ProcCmd(NSDCMD\_DEFAULT\_PARAM, NULL, 0, 0, 1);  //Rsvd 1: 复位后重启设备 0： 复位后不重启设备  复位IMG模块参数:  CWebLib::GetInstance().IF\_ProcCmd(NSDCMD\_DEFAULT\_PARAM, 0, 0,NSDPAR\_IMG, 1);  控制 PTZ:  NSD\_PTZ\_CTRL\_S ptz\_ctrl;  ptz\_ctrl.u8UartNo=0;  ptz\_ctrl.u8ChannelNo=0;  ptz\_ctrl.u8ProtoType = 0xff;  ptz\_ctrl.u8Command = NSDPTZ\_BRUSHCLOSE;  COnvifLib::GetInstance().IF\_ProcCmd(NSDCMD\_PTZ\_CTRL,&ptz\_ctrl,sizeof(ptz\_ctrl),0,0);  设置屏幕亮度为85%  unsigned char u8NormalBrightness = 85;  CAiLib::GetInstance().IF\_ProcCmd(NSDCMD\_SCREEN\_LUMA,(void \*) u8NormalBrightness,0,0,0); |

## Int (\*IF\_WriteVideoFrameToCache)(unsigned short nCh,int nStreamType,[LPNSD\_FRAME\_HEAD\_S](#_NSD_FRAME_HEAD_S) lpFrameHead,const unsigned char \* lpFrameBuf,unsigned long nBufSize);

**Function explain**: 写视频数据到缓存

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道ID，从0开始 |
| nStreamType | 码流ID，0为抓拍流，视频流从1开始 |
| lpFrameHead | 视频数据头 |
| lpFrameBuf | 视频数据 |
| nBufSize | 视频数据长度 |

**Notes：**

**该接口为video模块内部使用， 其他模块请勿调用**

## Long (\*IF\_ReadVideoFrameFromCache)(unsigned short nCh,int nStreamType,unsigned char \* lpFrameBuf,unsigned long & nBufSize, int nFirstFlag,unsigned long nNeedFrameNo,unsigned long & nRealFrameNo);

**Function explain**: 从缓存中读编码后数据

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道ID，从0开始 |
| nStreamType | 码流ID，0为抓拍流，视频流从1开始 |
| lpFrameBuf | 视频数据. 数据头+编码后数据， 视频数据偏移量为sizeof(NSD\_FRAME\_HEAD\_S) |
| nBufSize | 数据长度. 视频数据长度+ sizeof(NSD\_FRAME\_HEAD\_S) |
| nFirstFlag | 1: nNeedFrameNo 被忽略. 接口会返回最近的一个I帧  0:获取帧号为nNeedFrameNo的那帧 |
| nNeedFrameNo | 待获取的帧的帧号 |
| nRealFrameNo | 返回的帧的实际帧号 |

**Return:**

NSDERR\_NOENOUGH\_BUF: 传入缓存不够大

小于0: 缓存内无视频

大于0: 帧大小

**Notes：**

1. 第一次获取视频数据的时候， 需要把nFirstFlag设置为1， 然后接口会返回最近的一个I帧以及帧号。 那么获取下一帧的时候， nFirstFlag需要设置为0， 然后nNeedFrameNo应该给上一帧序号+1.
2. 需要先调用IF\_PreviewStreamAddRef来启动对应码流的编码， 然后再调用该接口获取视频数据， 用完以后要调用IF\_PreviewStreamRelease来释放编码器资源

**Example**:

|  |
| --- |
| int bNeedKey=1;  int nNeedVideoFrameNo,nRealFrameNo;  int nBufSize=0;  int\_nRealLen=CRtspLib::GetInstance().IF\_ReadVideoFrameFromCache(0,0,m\_av\_read\_buffer.GetBuffer(),nBufSize,bNeedKey,nNeedVideoFrameNo,nRealFrameNo);  if(nRealLen==NSDERR\_NOENOUGH\_BUF)  {  //first time call the interface with 0 buffer size can get the real frame size, then use the real size to get the frame  m\_av\_read\_buffer.Alloc(nBufSize);  nBufSize = m\_av\_read\_buffer.GetBufferSize();  nRealLen=CRtspLib::GetInstance().IF\_ReadVideoFrameFromCache(real\_no,sub,m\_av\_read\_buffer.GetBuffer(),nBufSize,bNeedKey,nNeedVideoFrameNo,nRealFrameNo);  }  if(nRealLen>0)  {  nNeedVideoFrameNo= nRealFrameNo + 1;  bNeedKey=0;  NSD\_FRAME\_HEAD\_S \* lpFrameHead=(NSD\_FRAME\_HEAD\_S \*)m\_av\_read\_buffer.GetBuffer();  if(lpFrameHead->u16FrameFlag==FE\_MAGIC\_FLAG)  {  if(lpFrameHead->videoInfo.u8EncodeType==NSD\_CODEC\_MJPEG)  rtp\_send\_mjpeg\_video((char \*)m\_av\_read\_buffer.GetBuffer() + sizeof(NSD\_FRAME\_HEAD\_S), 0, nRealLen-sizeof(NSD\_FRAME\_HEAD\_S),RTP\_PT\_JPEG,lpFrameHead->u32TimeStamp\*90,false,0,0,NULL);  else if(lpFrameHead->videoInfo.u8EncodeType==NSD\_CODEC\_H264)  rtp\_send\_h264\_video((char \*)m\_av\_read\_buffer.GetBuffer() + sizeof(NSD\_FRAME\_HEAD\_S), 0, nRealLen-sizeof(NSD\_FRAME\_HEAD\_S),RTP\_PT\_H264,lpFrameHead->u32TimeStamp\*90,false,0,0,NULL);  else if(lpFrameHead->videoInfo.u8EncodeType==NSD\_CODEC\_H265)  rtp\_send\_h265\_video((char \*)m\_av\_read\_buffer.GetBuffer() + sizeof(NSD\_FRAME\_HEAD\_S), 0, nRealLen-sizeof(NSD\_FRAME\_HEAD\_S),RTP\_PT\_H265,lpFrameHead->u32TimeStamp\*90,false,0,0,NULL);  }  m\_av\_read\_buffer.RecheckSize(nRealLen+pre\_offsize);  } |

## Int (\*IF\_WriteAudioFrameToCache)(unsigned short nCh,[LPNSD\_FRAME\_HEAD\_S](#_NSD_FRAME_HEAD_S) lpFrameHead,const unsigned char \* lpFrameBuf,unsigned long nBufSize);

**Function explain**: 写音频数据到缓存

**Parameter**:

|  |  |
| --- | --- |
| nCh | 音频通道，从0开始 |
| lpFrameHead | 音频数据头 |
| lpFrameBuf | 音频数据 |
| nBufSize | 音频数据长度 |

**Notes：**

**该接口为audio模块内部使用， 其他模块请勿调用**

## Long (\*IF\_ReadAudioFrameFromCache)(unsigned short nCh,unsigned char \* lpFrameBuf,unsigned long & nBufSize, int nFirstFlag,unsigned long nNeedFrameNo,unsigned long &nRealFrameNo);

**Function explain**: 在缓存中读取音频数据

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道ID，从0开始 |
| lpFrameBuf | 音频数据(包含音频头), 数据偏移为 sizeof(NSD\_FRAME\_HEAD\_S) |
| nBufSize | 音频数据长度(包含音频头长度) |
| nFirstFlag | 1: nNeedFrameNo 会被忽略，返回最近的音频帧  0:获取帧号为nNeedFrameNo的音频帧 |
| nNeedFrameNo | 待获取的音频帧号 |
| nRealFrameNo | 返回的实际帧号 |

**Return:**

NSDERR\_NOENOUGH\_BUF: 传入的缓存大小不够

小于0: 缓存内无数据

大于0: 帧大小

**Notes：**

第一次获取音频数据的时候，需要设置bNewFrame为true，然后接口会返回最近的帧及帧号。 当获取下一帧时，bNewFrame需要设置为false， 然后nNeedFrameNo设置为上一帧的帧号+1 。

**Example**:

|  |
| --- |
| int bNewAudio =1;  int nNeedAudioFrameNo,nRealFrameNo;  int nBufSize=0;  int nRealLen = CRtspLib::GetInstance().IF\_ReadAudioFrameFromCache(0, m\_av\_read\_buffer.GetBuffer() , nBufSize, bNewAudio,nNeedAudioFrameNo, nRealFrameNo);  if(nRealLen==NSDERR\_NOENOUGH\_BUF)  {  m\_av\_read\_buffer.Alloc(nBufSize);  nBufSize = m\_av\_read\_buffer.GetBufferSize();  nRealLen = CRtspLib::GetInstance().IF\_ReadAudioFrameFromCache(0 , m\_av\_read\_buffer.GetBuffer() , nBufSize, bNewAudio, nNeedAudioFrameNo, nRealFrameNo);  }  if(nRealLen>0)  {  bNewAudio=0;  nNeedAudioFrameNo=nRealFrameNo+1;  NSD\_FRAME\_HEAD\_S \* lpFrameHead=(NSD\_FRAME\_HEAD\_S \*)m\_av\_read\_buffer.GetBuffer();  if(lpFrameHead->u16FrameFlag==FE\_MAGIC\_FLAG)  {  int payload=0;  switch(lpFrameHead->audioInfo.u8EncodeType)  {  case NSD\_CODEC\_G711U:  payload=RTP\_PT\_ULAW;  break;  case NSD\_CODEC\_G711A:  payload=RTP\_PT\_ALAW;  break;  case NSD\_CODEC\_AAC:  payload=RTP\_PT\_AAC;  break;  default:  payload=RTP\_PT\_S16BE\_MONO;//RTP\_PT\_ADPCM;  break;  }  ULONGLONG u64Stamp=lpFrameHead->u32TimeStamp;  u64Stamp\*=lpFrameHead->audioInfo.u32AudioSamples/1000;  if(lpFrameHead->audioInfo.u8EncodeType==NSD\_CODEC\_AAC)  {  int k=nRealLen-sizeof(NSD\_FRAME\_HEAD\_S)-7;  char \* pp=(char \*)(m\_av\_read\_buffer.GetBuffer()+sizeof(NSD\_FRAME\_HEAD\_S)+7); rtp\_send\_aac\_audio(pp,sizeof(NSD\_FRAME\_HEAD\_S)+7,k,payload,u64Stamp,false,0,0,NULL);  }  else  rtp\_send\_data((char \*)m\_av\_read\_buffer.GetBuffer() + sizeof(NSD\_FRAME\_HEAD\_S), sizeof(NSD\_FRAME\_HEAD\_S), nRealLen-sizeof(NSD\_FRAME\_HEAD\_S),payload,u64Stamp,false,0,0,NULL,false,1);  }  } |

## Int (\*IF\_WriteTalkFrameToCache)([LPNSD\_FRAME\_HEAD\_S](#_NSD_FRAME_HEAD_S) lpFrameHead,const unsigned char \* lpFrameBuf,unsigned long nBufSize);

**Function explain**: 写对讲数据到缓存

**Parameter**:

|  |  |
| --- | --- |
| lpFrameHead | 音频头 |
| lpFrameBuf | 音频数据 |
| nBufSize | 音频数据长度 |

**Notes：**

目前只支持G.711和PCM格式的音频

|  |
| --- |
| NSD\_FRAME\_HEAD\_S head;  memset(&head,0,sizeof(NSD\_FRAME\_HEAD\_S));  head.u16FrameFlag=FE\_MAGIC\_FLAG;  head.u8FrameType=NSD\_FRAME\_A;  head.u32FrameNo=m\_frame\_no++;  head.u32TimeStamp=0;  head.u32FrameSize=m\_nAudioRecvOff;  head.audioInfo.u8EncodeType=NSD\_CODEC\_G711A;  head.audioInfo.u8AudioChannels=1;  head.audioInfo.u8AudioBits=16;  head.audioInfo.u32AudioSamples=8000;  CRtspLib::GetInstance().IF\_WriteTalkFrameToCache(&head,m\_audio\_recv\_buffer.GetBuffer(),m\_nAudioRecvOff); |

## long (\*IF\_ReadTalkFrameFromCache)(unsigned char \* lpFrameBuf,unsigned long & nBufSize,bool bNewFrame,unsigned long nNeedFrameNo,unsigned long & nRealFrameNo);

**Function explain**: 读取对讲数据到缓存

**Parameter**:

|  |  |
| --- | --- |
| lpFrameBuf | 音频数据(包含音频头), 数据偏移为 sizeof(NSD\_FRAME\_HEAD\_S) |
| nBufSize | 音频数据长度(包含音频头长度) |
| bNewFrame | 1: nNeedFrameNo 会被忽略，返回最近的音频帧  0:获取帧号为nNeedFrameNo的音频帧 |
| nNeedFrameNo | 待获取的音频帧号 |
| nRealFrameNo | 返回的实际帧号 |

**Notes：**

**该接口为audio模块内部使用， 其他模块请勿调用**

## Int (\*IF\_WriteMDFrameToCache)(unsigned short nCh,[LPNSD\_FRAME\_HEAD\_S](#_NSD_FRAME_HEAD_S) lpFrameHead,const unsigned char \* lpFrameBuf,unsigned long nBufSize);

**Function explain**: write motion detection data to cache buffer

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道号，从0开始 |
| lpFrameHead | 移动侦测数据头 |
| lpFrameBuf | 移动侦测数据。18\*22 矩阵 |
| nBufSize | 移动侦测数据长度 |

**Notes：**

**该接口为video模块内部使用， 其他模块请勿调用**

## Long (\*IF\_ReadMDFrameFromCache)(unsigned short nCh,unsigned char \* lpFrameBuf,unsigned long & nBufSize,bool bNewFrame,unsigned long nNeedTickCount,unsigned long & nRealTickCount);

**Function explain**: read motion detection data from cache buffer

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道号，从0开始 |
| lpFrameBuf | 移动侦测数据(包含数据头)，数据偏移为sizeof(NSD\_FRAME\_HEAD\_S) |
| nBufSize | 移动侦测数据长度(包含数据长度) |
| bNewFrame | True： nNeedFrameNo 被忽略. 接口返回最近的一帧数据 |
| nNeedTickCount | 待获取的帧号 |
| nRealTickCount | 实际返回的帧号 |

**Return:**

NSDERR\_NOENOUGH\_BUF: 传入的缓存大小不够

<0: 缓存内无帧

>0: 帧大小

**Example**:

|  |
| --- |
| unsigned long nBufSize=0;  unsigned long nRealTick=0;  unsigned char u8Block[MD\_ROW\_NUM\*MD\_COL\_NUM]={0};  memset(u8Block,0,MD\_ROW\_NUM\*MD\_COL\_NUM);  int nRealLen=CRtspLib::GetInstance().IF\_ReadMDFrameFromCache(0, m\_md\_read\_buffer.GetBuffer(),nBufSize, bNewMD, nNeedMDTick,nRealTick);  if(nRealLen==NSDERR\_NOENOUGH\_BUF)  {  m\_md\_read\_buffer.Alloc(nBufSize);  nBufSize = m\_md\_read\_buffer.GetBufferSize();  nRealLen = CRtspLib::GetInstance().IF\_ReadMDFrameFromCache(0, m\_md\_read\_buffer.GetBuffer(), nBufSize, bNewMD, nNeedMDTick,nRealTick);  }  if(nRealLen>0)  {  bNewMD=false;  nNeedMDTick=nRealTick+1;  LPNSD\_FRAME\_HEAD\_S pFrameHead=(LPNSD\_FRAME\_HEAD\_S)m\_md\_read\_buffer.GetBuffer();  if (pFrameHead->u8FrameType==NSD\_FRAME\_MD)  {  memcpy(u8Block,m\_md\_read\_buffer.GetBuffer()+sizeof(NSD\_FRAME\_HEAD\_S),pFrameHead->u32FrameSize);  }  } |

## void (\*IF\_ClearCache)(int nCacheType,unsigned short nCh,int nStreamType);

**Function explain**: 清空指定缓存

**Parameter**:

|  |  |
| --- | --- |
| nCacheType | 缓存类型，参照 \_NSD\_CACHE\_TYPE\_E（privydef.h） |
| nCh | 通道号 |
| nStreamType | 码流ID |

**Notes：**

不建议调用该接口， 避免对其他模块造成影响

## Long (\*IF\_AddUartCB)(unsigned short nCh,OnSerialData lpCallback);

**Function explain**: 向UART模块注册回调，通道号为nCh的串口有数据时，会通过回调通知注册者

**Parameter**:

|  |  |
| --- | --- |
| nCh | 串口通道，0为球机通道 1为RS485 |
| lpCallback | 回调函数 |

**Return：**

返回注册句柄， 移除时需要用到该句柄

## Int (\*IF\_RemoveUartCB)(unsigned short nCh,long hHandle);

**Function explain**: 移除注册的回调函数

**Parameter**:

|  |  |
| --- | --- |
| nCh | 串口通道，0为球机通道 1为RS485 |
| hHandle | 注册句柄 |

**Note**:

## Long (\*IF\_WriteUartData)(unsigned short nCh,const char \* lpWriteBuf,unsigned long nWriteBufSize);

**Function explain**: 向串口发送数据

**Parameter**:

|  |  |
| --- | --- |
| nCh | 串口通道，0为球机通道 1为RS485 |
| lpWriteBuf | 串口数据 |
| nWriteBufSize | 串口数据长度 |

**Note**:

检测sys.def 文件， 目录为 /mnt/mtd/code/fixed/sys.def， rs232对应的nCh为0， rs485对应的字段为1， 如果想使用该接口， 相应的字段需要为1。

如果发送后对端收不到数据，需要检查/mnt/mtd/nsd/config/setting/uart.conf里面的配置是否符合预期(设备名，波特率，停止位，校验位等)

## Long (\*IF\_ReadUartData)(unsigned short nCh,char \* lpReadBuf,unsigned long nReadBufSize,int nTimeout);

**Function explain**: 从串口读取数据

**Parameter**:

|  |  |
| --- | --- |
| nCh | 串口通道，0为球机通道 1为RS485 |
| lpReadBuf | 读到的串口数据 |
| nReadBufSize | 数据长度 |
| nTimeout | 超时时间， 单位:ms |

**Note**:

检测sys.def 文件， 目录为 /mnt/mtd/code/fixed/sys.def， rs232对应的nCh为0， rs485对应的字段为1， 如果想使用该接口， 相应的字段需要为1。

如果无法收到数据，需要检查/mnt/mtd/nsd/config/setting/uart.conf里面的配置是否符合预期(设备名，波特率，停止位，校验位等)

## void (\*IF\_UartWaitForSafeWriteTime)(unsigned short nCh,unsigned long nMilliSecond);

**Function explain**: 用来保证串口发送的最小间隔时间， 调用该接口后， 如果距离上次串口操作时间小于nMilliSecond时间，则等待nMilliSecond时间后返回。 如果大于nMilliSecond， 则立刻返回。

**Parameter**:

|  |  |
| --- | --- |
| nCh | 串口通道 |
| nMilliSecond | 睡眠时间 单位: ms |

**Note**:

## Long (\*IF\_FindPTZCmd)(unsigned short nCh,int nCmd,char \* lpReadBuf,unsigned long nReadBufSize,int nTimeout);

**Function explain**: 根据命令ID获取PTZ指令

**Note**:

未实现

## Long (\*IF\_AddGetPTZPosCallback)(unsigned short nCh, get\_ptz\_position\_callback pGetPTZCallback);

**Function explain**: 未实现

## int (\*IF\_RemoveGetPTZPosCallback)(unsigned short nCh,long hHandle);

**Function explain**: 未实现

## Int (\*IF\_StartTalk)();

**Function explain**: 启动对讲， 开启后audio模块会从缓存读取音频数据并播放

**Note**:

## Int (\*IF\_StopTalk)();

**Function explain**: 停止对讲

**Note**:

## Int (\*IF\_PreviewStreamAddRef)(unsigned short nCh,int nStreamType);

**Function explain**: 向通道为nCh,码流为nStreamType增加一个引用计数， 如果计数从0变为1则开启对应的编码器.

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道号 从0开始 |
| nStreamType | 码流ID， 0为抓图码流， 视频流从1开始 |

**Note**:

## Int (\*IF\_PreviewStreamRelease)(unsigned short nCh,int nStreamType);

**Function explain**: 把通道为nCh，码流ID为nStreamType的引用计数减少1， 如果计数变为0则关闭对应的编码器

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道号 从0开始 |
| nStreamType | 码流ID， 0为抓图码流， 视频流从1开始 |

**Note**:

## Long (\*IF\_GetPreviewCount)(unsigned short nCh,int nStreamType);

**Function explain**: 获取引用计数的数量

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道号 从0开始 |
| nStreamType | 码流ID， 0为抓图码流， 视频流从1开始 |

**Note**:

## Int (\*IF\_GetVideoMediaInfo)(unsigned short nCh,int nStreamType,unsigned char \* lpSps,unsigned long & nSpsLen,unsigned char \* lpPps,unsigned long & nPpsLen);

**Function explain**: 获取nCh通道，nStreamType码流的视频基本信息

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道号 从0开始 |
| nStreamType | 码流ID， 0为抓图码流， 视频流从1开始 |

**Note**:

该接口只适用于 H.264 or H.265 通道

## Int (\*IF\_RtspReset)(unsigned short nCh,int nStreamType,int nAVFlag);

**Function explain**: 复位RTSP服务器

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道号 从0开始 |
| nStreamType | 码流ID， 0为抓图码流， 视频流从1开始 |
| nAVFlag | 0: 视频 1:音频 |

**Note**:

未实现

## int (\*IF\_VASync)(unsigned short nCh,int nSyncType,unsigned long nSyncParam);

**Function explain**: 如果音频模块在视频模块前初始化，用该接口来完成芯片SDK的系统初始化。 因为通常芯片SDK系统初始化是在视频模块内完成。

**Parameter**:

|  |  |
| --- | --- |
| nCh | Channel ID (from 0 - max) |
| nSyncType | 0: init 1:uninit |
| nSyncParam | Reserved |

**Note**:

该接口为video和audio模块内部使用， 其他模块请勿调用

## int (\*IF\_Scale)([AI\_VIDEO\_FRAME\_S](#_AI_VIDEO_FRAME_S) \*p\_Frame, [AI\_VIDEO\_FRAME\_S](#_AI_VIDEO_FRAME_S) \*p\_OutFrame);

**Function explain**: YUV缩放接口， 针对AI模块开发用户， 如果获取到的YUV不适合算法使用， 可以使用该接口改变YUV的大小。

**Parameter**:

|  |  |
| --- | --- |
| p\_Frame | 源数据 |
| p\_OutFrame | 目标数据 |

**Notes：**

**Example**:

|  |
| --- |
| int scale(AI\_VIDEO\_FRAME\_S \*p\_Frame)  {  AI\_VIDEO\_FRAME\_S out;  out.nWidth = 640;  out.nHeight= 360;  CAilib::Getinstance()->IF\_Scale(pFrame, &out);  …  return 0;  } |

## int (\*IF\_GetZoomPosition)(unsigned short nCh);

**Function explain**: 未实现

## Int (\*IF\_StartRecord)(int nCh,int nStreamType,int nRecTime,int nRecPreTime);

**Function explain**: 启动SD卡录像

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道号，从0开始 |
| nStreamType | 码流ID，0为抓拍流，视频流从1开始 |
| nRecTime | 录像时间 单位:ms |
| nRecPreTime | 预录时间 单位:second |

**Note**:

## Int (\*IF\_StopRecord)(int nCh);

**Function explain**: 停止SD卡录像

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道号，从0开始 |

**Note**:

## Int (\*IF\_StartFtpRecord)(int nCh,int nStreamType,int nRecTime,int nRecPreTime);

**Function explain**: 开启FTP录像

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道号，从0开始 |
| nStreamType | 码流ID，0为抓拍流，视频流从1开始 |
| nRecTime | 录像时间 单位:ms |
| nRecPreTime | 预录时间 单位:second |

**Note**:

## Int (\*IF\_StopFtpRecord)(int nCh);

**Function explain**: 停止FTP录像

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道ID |

**Note**:

## void \* (\*IF\_OpenRecordService)();

**Function explain**: 开启录像回放服务

**Parameter**: NA

**Return:**

录像服务句柄

**Note**:

如果要回放SD卡录像的话，需要首先通过该接口来开启服务

## void (\*IF\_CloseRecordService)(void \* hRecService);

**Function explain**: 停止录像回放服务

**Parameter**:

|  |  |
| --- | --- |
| hRecService | 录像服务句柄 |

**Note**:

## Int (\*IF\_QueryRecord)(void \* hRecService,[LPNSD\_RECORD\_FILE\_S](#_NSD_RECORD_FILE_S) lpSearch);

**Function explain**: 根据条件查询录像

**Parameter**:

|  |  |
| --- | --- |
| hRecService | 录像服务句柄 |
| lpSearch | 查询条件 |

**Note**:

## Int (\*IF\_GetRecordInfo)(void \* hRecService,int nIndex,[LPNSD\_RECORD\_FILE\_S](#_NSD_RECORD_FILE_S) lpSearchRet,int & nCount);

**Function explain**: 获取录像查询信息

**Parameter**:

|  |  |
| --- | --- |
| hRecService | 录像服务句柄 |
| nIndex | 录像索引 |
| lpSearchRet | 查询结果 |
| nCount | 输入最大的查询文件数量， 该参数在接口调用后会返回实际查询到的文件数量 |

**Note**:

调用该接口前需要调用IF\_QueryRecord 进行查询

**Example:**

|  |
| --- |
| void \* hRec=CNetLib::GetInstance().IF\_OpenRecordService();  if(hRec==NULL)  return NSDERR\_OUTOFMEM;  int nCount=CNetLib::GetInstance().IF\_QueryRecord(hRec,lpRecordFile);  if(nCount<0)  {  CNetLib::GetInstance().IF\_CloseRecordService(hRec);  return nCount;  }  else  {  lpHead->u32ErrCode\_a = NSDERR\_SUCCESS;  NSD\_RECORD\_FILE\_S rf[36];  memset(rf,0,sizeof(NSD\_RECORD\_FILE\_S)\*36);  int nIndex=0;  while(1)  {  int n=36;  if(CNetLib::GetInstance().IF\_GetRecordInfo(hRec,nIndex,rf,n)!=0)  {  CNetLib::GetInstance().IF\_CloseRecordService(hRec);  return NSDERR\_UNKNOW;  }  if(n==0)  break;  nIndex+=n;  lpHead->u32PacketSize\_ar = sizeof(NSD\_RECORD\_FILE\_S)\*n;  if(m\_lpChannel)  m\_lpChannel->send\_packet\_ex(&addr,lpHead,(const char \*)rf,lpHead->u32PacketSize\_ar);  }  CNetLib::GetInstance().IF\_CloseRecordService(hRec);  } |

## Int (\*IF\_OpenReplay)(void \* hRecService,[LPNSD\_RECORD\_FILE\_S](#_NSD_RECORD_FILE_S) lpPlay);

**Function explain**: 开启录像回放

**Parameter**:

|  |  |
| --- | --- |
| hRecService | 录像服务句柄 |
| lpPlay | 文件信息 |

**Note**:

## Int (\*IF\_ReplayCtrl)(void \* hRecService,NSD\_REPLAY\_CTRL\_E nCtrlType,long nParam1,unsigned long nParam2);

**Function explain**: 控制回放，比如开启回放，快放，慢放等

**Parameter**:

|  |  |
| --- | --- |
| hRecService | 录像服务句柄 |
| nCtrlType | 控制类型 |
| nParam1 | 参数1 |
| nParam2 | 参数2 |

**Note**:

**Example:**

|  |
| --- |
| m\_hRec=CNetLib::GetInstance().IF\_OpenRecordService();  if(m\_hRec==0)  return NSDERR\_OUTOFMEM;  CNetLib::GetInstance().IF\_SetRecordCallback(m\_hRec,OnRecordStreamCallback,OnRecordPlayEnd,this);  long nRet=CNetLib::GetInstance().IF\_OpenReplay(m\_hRec,lpRecordFile);  if(nRet!=0)  {  CNetLib::GetInstance().IF\_SetRecordCallback(m\_hRec,NULL,NULL,NULL);  if(m\_lpChannel)  m\_lpChannel->answer(addr,lpHead,nRet);  CNetLib::GetInstance().IF\_CloseRecordService(m\_hRec);  m\_hRec=NULL;  return NSDERR\_SUCCESS;  }  CNetLib::GetInstance().IF\_ReplayCtrl(m\_hRec,NSD\_REPLAY\_PLAY,0,0); |

## Int (\*IF\_CloseReplay)(void \* hRecService);

**Function explain**: 停止录像回放

**Parameter**:

|  |  |
| --- | --- |
| hRecService | 录像服务句柄 |

**Note**:

## Int (\*IF\_RecordSavePicture)(unsigned short nCh,unsigned char \* lpPicData,unsigned long nDataLen);

**Function explain**: 未实现

## Int (\*IF\_SetRecordCallback)(void \* hRecService,record\_stream\_callback pStreamCallback,record\_playend\_callback pEndCallback,void \* pContext);

**Function explain**: 设置录像数据回调， 启动录像回放后， 读到的数据会通过回调给到使用者

**Parameter**:

|  |  |
| --- | --- |
| hRecService | 录像服务句柄 |
| pStreamCallback | 录像数据回调函数 |
| pEndCallback | 播放结束回调函数 |
| pContext | 上下文 |

**Note**:

## Int (\*IF\_WritePictureFrame)(unsigned short nCh, [LPNSD\_SNAP\_HEAD\_S](#_NSD_SNAP_HEAD_S) head, const unsigned char \* lpFrameBuf, unsigned long nBufSize);

**Function explain**: 抓图数据写入缓存

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道ID |
| head | 图片头信息 |
| lpFrameBuf | 图片数据 |
| nBufSize | 图片数据长度 |

**Note**:

塞入缓存后需要及时取用， 否则500ms后会释放该图片， 避免缓存长期占用

**Example:**

|  |
| --- |
| [NSD\_SNAP\_HEAD\_S](#_NSD_SNAP_HEAD_S) head;  memset(&head, 0, sizeof(NSD\_SNAP\_HEAD\_S));  head.u16Magic = 0x7470;  head.u8Type = NSD\_AI\_FACE;  head.u8Company = NSD\_COMPANY\_UNKNOWN;  head.u32SessionId=nSessionId;  head.u8TotleNum=nTotleNum;  head.u8No=nNo;  struct tm t;  dt.GetTM(t);  head.u32TimeStamp = mktime(&t);  head.u16MicroTimeStamp = dt.GetMilliSecond();    head.faceResult.u16FaceId = -1;  head.faceResult.u8IsOrigPicture=1;  head.faceResult.u8Quality = nQuality;    head.faceResult.resultPosition.u16X =0;  head.faceResult.resultPosition.u16Y =0;  head.faceResult.resultPosition.u16Width = nWidth;  head.faceResult.resultPosition.u16Height = nHeight;    return CAiLib::GetInstance().IF\_WritePictureFrame(nChn, &head, pJpg,nJpgLen); |

## Long (\*IF\_ReadPictureFrameByFrom)( unsigned short nCh,unsigned long nFromType,[SPictureData](#_PictureData) &lpFrameBuf,unsigned long nNeedTick);

**Function explain**: 根据触发源读取图片

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道ID |
| nFromType | 参见\_NSD\_ACTION\_FROM\_E |
| lpFrameBuf | 图片数据(智能指针) |
| nNeedTick | 待读取的图片索引 |

**Note**:

**Example**:

|  |
| --- |
| [SPictureData](#_PictureData) pFrameBuf;  int nRealLen=IF\_ReadPictureFrameByFrom(0, NSDFROM\_FACE, pFrameBuf, m\_nPicTick);  if(nRealLen>0)  {  m\_nPicTick=pFrameBuf->m\_tick+1;  } |

## Long (\*IF\_ReadPictureFrameBySave)( unsigned short nCh,unsigned long nSaveType,[SPictureData](#_PictureData) &lpFrameBuf,unsigned long nNeedTick);

**Function explain**: 根据存储目的设置查找图片

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道ID |
| nSaveType | 参见 \_NSD\_SAVE\_TYPE\_E |
| lpFrameBuf | 图片数据 |
| nNeedTick | 待获取的图片索引 |

**Note**:

nSaveType取决于设置的联动动作内抓图路径， 比如设置了图片上传FTP， 那么就可以用NSD\_SAVE\_FTP来获取到图片

**Example**:

|  |
| --- |
| void CFtpPic::OnSnap()  {  [SPictureData](#_PictureData) spic;  int nRealLen=CFtpLib::GetInstance().IF\_ReadPictureFrameBySave(m\_nChannel,NSD\_SAVE\_FTP,spic,m\_nPicTick);  if(nRealLen>0)  {  m\_nPicTick=spic->m\_tick+1;  PictureItem\_S \*pic = new PictureItem\_S;  pic->m\_data = spic;  m\_pic\_queue.Put(pic);  }  } |

## Long (\*IF\_ReadPictureFrameByFromAndSave)( unsigned short nCh,unsigned long nFromType,unsigned long nSaveType,[SPictureData](#_PictureData) &lpFrameBuf,unsigned long nNeedTick);

**Function explain**: 根据保存路径或者触发源来获取图片

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道ID |
| nFromType | 参见 \_NSD\_ACTION\_FROM\_E |
| nSaveType | 参见 \_NSD\_SAVE\_TYPE\_E |
| lpFrameBuf | 图片数据 |
| nNeedTick | 待获取的图片索引 |

**Note**:

## Int (\*IF\_Shoot)(unsigned short nCh,NSD\_ACTION\_FROM\_E nFrom,unsigned long nNum, unsigned long nInterval,unsigned long nSaveType,unsigned long nPreSecond,unsigned long nPresistMillisec);

**Function explain**: 触发抓图， 调用该接口后， 会把抓图流的引用计数加1， 同时存储抓图相关数据

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道ID |
| nFromType | 参见 NSD\_ACTION\_FROM\_E (privydef.h) |
| nNum | 期望抓图的数量 |
| nInterval | 抓图间隔 单位:ms |
| nSaveType | 参见 NSD\_SAVE\_TYPE\_E (nsddefines.h) |
| nPreSecond | 抓图准备时间 单位:sec |
| nPresistMillisec | 持续时长，单位: ms 0或者-1为持续抓图 |

**Note**:

## Int (\*IF\_ResetShoot)(unsigned short nCh,NSD\_ACTION\_FROM\_E nFrom);

**Function explain**: 清楚抓图缓存

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道ID |
| nFromType | 参见 \_NSD\_ACTION\_FROM\_E |

**Note**:

## void (\*IF\_Alarm)(NSD\_ALARM\_TYPE\_E nAlarmType,bool bAlarm,unsigned short nCh);

**Function explain**: 发送报警信息

**Parameter**:

|  |  |
| --- | --- |
| nAlarmType | 报警类型 |
| bAlarm | 报警标识, true: 报警触发 false: 报警接触 |
| nCh | 报警通道 |

**Note**:

该接口调用后， 通知各模块报警的同时， 还会触发报警联动动作， 如果联动抓图， 会把触发源置为响应的报警源

**Example**:

|  |
| --- |
| //Face is detected  CAiLib::GetInstance().IF\_Alarm(NSD\_ALARM\_FACE, 1, 0);  //Face is not detected  CAiLib::GetInstance().IF\_Alarm(NSD\_ALARM\_FACE, 0, 0); |

## Long (\*IF\_GetIOStatus)(int nIOType,int nParam);

**Function explain**: 获取IO状态

**Parameter**:

|  |  |
| --- | --- |
| nIOType | 参见 \_NSD\_IO\_TYPE\_E |
| nParam | Reserved |

**Note**:

**Example**:

|  |
| --- |
| int nStatus=CVideoLib::GetInstance().IF\_GetIOStatus(NSDIO\_IR,-1);  if(nStatus)  return NSD\_DN\_STATUS\_NIGHT;  else  return NSD\_DN\_STATUS\_DAY; |

## int (\*IF\_IOCtl)(int nIOType,int nValue,int nChn,int nParam);

**Function explain**: 设置IO状态

**Parameter**:

|  |  |
| --- | --- |
| nIOType | 参见 \_NSD\_IO\_TYPE\_E |
| nValue | IO状态 |
| nChn | 通道ID |
| nParam | 当nIOType为 NSDIO\_OUTPUT时, 该参数表示持续时间. 单位:ms |

**Note**:

**Example**:

|  |
| --- |
| CUtilLib::GetInstance().IF\_IOCtl(NSDIO\_OUTPUT,1,0,1000); //设置报警输出0的电平为高， 持续时间1秒  CVideoLib::GetInstance().IF\_GetIOStatus(NSDIO\_IR,-1); //读取光敏电阻的状态  CVideoLib::GetInstance().IF\_GetIOStatus(NSDIO\_INPUT,0); //读取报警输入0的状态 |

## Long (\*IF\_GetCapabilty)(NSD\_CAPABILITY\_TYPE\_E nCapabiltyType,void \* lpRecvBuf,unsigned long nBufLen,unsigned short nCh,unsigned int nAppend);

**Function explain**: 获取设备能力

**Parameter**:

|  |  |
| --- | --- |
| nCapabiltyType | 参见 \_NSD\_CAPABILITY\_TYPE\_E (nsddefines.h) |
| lpRecvBuf | 设备能力接收缓存 |
| nBufLen | 缓存长度 |
| nCh | 通道ID |
| nAppend | 扩展字段 |

**Example**:

|  |
| --- |
| char szCapabilty[1024]={0};  COnvifLib::GetInstance().IF\_GetCapabilty(NSD\_CAPABILITY\_LIST\_VI,szCapabilty,1024,nCh,0);  int nCapabiltyNum=0;  memcpy(&nCapabiltyNum,szCapabilty,sizeof(int));  NSD\_CAPABILITY\_VI\_S\* pstCapabilityVI=(NSD\_CAPABILITY\_VI\_S\*)(szCapabilty+sizeof(int));  NSD\_CAPABILITY\_VENC\_S stH264Venc;  char szTmpBuf[256]={0};  int nVencNum=0;  nRet=COnvifLib::GetInstance().IF\_GetCapabilty(NSD\_CAPABILITY\_LIST\_VENC,szTmpBuf,256,nCh,nStream);  memcpy(&nVencNum,szTmpBuf,sizeof(int));  for (int i=0; i<nVencNum; i++)  {  NSD\_CAPABILITY\_VENC\_S\* pstVenc=(NSD\_CAPABILITY\_VENC\_S\*)(szTmpBuf+sizeof(int)+i\*sizeof(NSD\_CAPABILITY\_VENC\_S));  if ( pstVenc->u16EncodeType==NSD\_CODEC\_H264)  memcpy(&stH264Venc,pstVenc,sizeof(NSD\_CAPABILITY\_VENC\_S));  }  char szResolution[512]={0};  int nRet=COnvifLib::GetInstance().IF\_GetCapabilty(NSD\_CAPABILITY\_LIST\_RESOLUTION,szResolution,512,nCh,nStream);  int nResolutionNum=0;  memcpy(&nResolutionNum,szResolution,sizeof(int));  for(int i = 0; i < nResolutionNum;i++)  {  NSD\_RESOLUTION\_S\* pstResolution=(NSD\_RESOLUTION\_S\*)(szResolution+sizeof(int)+i\*sizeof(NSD\_RESOLUTION\_S));  }  char szListSample[512]={0};  COnvifLib::GetInstance().IF\_GetCapabilty(NSD\_CAPABILITY\_LIST\_AIN\_SAMPLE,szListSample,512,nCh,nStream);  int nSampleNum=0;  memcpy(&nSampleNum,szListSample,sizeof(int));  for(int j=0; j<nSampleNum; j++)  {  NSD\_FIXED\_S stSample={0};  memcpy(&stSample,(szListSample+sizeof(int)+j\*sizeof(NSD\_FIXED\_S)),sizeof(NSD\_FIXED\_S));  AudioEncoderConfigurationOptions->Options[i].BitrateList->Items[j] = 16 \* stSample.val/1000;  AudioEncoderConfigurationOptions->Options[i].SampleRateList->Items[j] = stSample.val/1000;  } |

## Int (\*IF\_ValidUserPwd)(const char \* lpUser,const char \* lpPwd,int & nMisc);

**Function explain**: 验证用户名密码

**Parameter**:

|  |  |
| --- | --- |
| lpUser | 用户名 |
| lpPwd | 密码 |
| nMisc | 用户权限，传出参数 |

**Return:**

0: 验证通过

非0: NSD\_ERRCODE\_E

**Note:**

## Int (\*IF\_OnHttpData)(const char \* szBuf,char \* szOutBuf,int nOutBufSize);

**Function explain**: 传递HTTP端口传来的数据

**Note:**

该接口为transfer模块内部使用， 其他模块不需要调用

## Int (\*IF\_OnNetworkReady)(int nWorkType);

**Function explain**: 网络模块初始化好网络后会用该接口通知其他模块

**Parameter**:

|  |  |
| --- | --- |
| nWorkType | 固定为0 |

**Note:**

内部接口

## Int (\*IF\_GetSDCardInfo)(unsigned int & nTotalSize,unsigned int & nFreeSize);

**Function explain**: 获取SD卡信息

**Parameter**:

|  |  |
| --- | --- |
| nTotalSize | 总SD卡大小， 单位: MB |
| nFreeSize | SD卡空闲大小， 单位: MB |

**Note:**

机型文件/mnt/mtd/code/fixed/sys.def中disknum字段需要为1

## Int (\*IF\_TestSmtp)(const char \* szTitle,const char \* szContext);

**Function explain**: 测试SMTP参数是否可以正常收发邮件

**Parameter**:

|  |  |
| --- | --- |
| szTitle | 邮件标题 |
| szContext | 邮件文本 |

**Note:**

## int (\*IF\_GetBinData)(unsigned char \* lpBinData,unsigned long & nDataLen,int rsvd);

**Function explain**: 后拼全景设备用该接口来获取标定文件

**Parameter**:

|  |  |
| --- | --- |
| lpBinData | 标定文件内容 |
| nDataLen | 标定文件长度 |

**Note:**

## Int (\*IF\_GetYUVData)(unsigned short nCh,[AI\_VIDEO\_FRAME\_S](#_AI_VIDEO_FRAME_S) \*\* llpVideoFrame,unsigned long nTimeoutMilliSec);

**Function explain**: 获取YUV数据

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道ID |
| llpVideoFrame | YUV数据 |
| nTimeoutMilliSec | 超时时间 , 单位:ms |

**Note:**

nPixelFormat默认为3， 不建议使用1， 因为海思默认出来的格式是NV21， 如果设置成1， 我们会在内部转化格式， 会有额外的时间消耗

Hisi3559A平台下如果是8路的前拼设备， 则不能用该接口获取YUV， 只能通过IF\_AddYuvStreamCallback 注册回调来获取YUV

**Return:**

0: 获取成功

非0: NSD\_ERRCODE\_E

**Example**:

|  |
| --- |
| //Set parameter of video channel. u8PixelFormat should be 1 or 3, 3 is recommanded  NSD\_AI\_S ai\_cfg;  ai\_cfg.u16AiImageWidth = 1920;  ai\_cfg.u16AiImageHeight = 1080;  ai\_cfg.u8PixelFormat = 3;  ai\_cfg.u8CacheNumber = 8;  ai\_cfg.u8CacheNumberForVout = 2;  IF\_SetParam(NSDCFG\_AI,&ai\_cfg,0,0);  AI\_VIDEO\_FRAME\_S \*llpVideoFrame=NULL;  if(CAiLib::GetInstance().IF\_GetYUVData(0,&llpVideoFrame,200)!=0)  {  printf("get yuv data failed\n");  usleep(40000);  continue;  }  if(llpVideoFrame==NULL)  {  printf("yuv data error\n");  usleep(40000);  continue;  }  int w=llpVideoFrame->nWidth;  int h=llpVideoFrame->nHeight;  CAiLib::GetInstance().IF\_ReleaseYUVData(0,llpVideoFrame); |

## Int (\*IF\_SendYUVToVenc)(unsigned short nCh,[AI\_VIDEO\_FRAME\_S](#_AI_VIDEO_FRAME_S) \* lpVideoFrame);

**Function explain**: 发送YUV给编码器

**Parameter**:

|  |  |
| --- | --- |
| nCh | Channel Id, ip camera is always 0 |
| llpVideoFrame | Yuv data |

**Note:**

未实现

## Int (\*IF\_ReleaseYUVData)(unsigned short nCh,[AI\_VIDEO\_FRAME\_S](#_AI_VIDEO_FRAME_S) \* lpVideoFrame);

**Function explain**: 释放YUV数据。 数据不用后要及时释放， 避免影响后续数据获取

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道ID |
| llpVideoFrame | YUV数据 |

**Note:**

## Long (\*IF\_AddAIAnalyseCallback)(unsigned short nCh,ai\_analyse\_callback lpCallback);

**Function explain**: HVP向AI模块注册结果回调， AI模块获取YUV并分析YUV，得到结果后， 通过该回调函数把结果分集发给HVP框架， HVP用该结果来在视频上画框。

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道ID |
| lpCallback | 回调函数 |

typedef int (\*ai\_analyse\_callback)(unsigned short nCh,int analyse\_type, YUVAnalyzeResult \* result);

|  |  |
| --- | --- |
| nCh | 通道ID |
| analyse\_type | 参见NSD\_AI\_ANALYSE\_TYPE\_E |
| result | 分析的结果集 |

YUVAnalyzeResult:

|  |  |
| --- | --- |
| analyzeType | 参见NSD\_AI\_ANALYSE\_TYPE\_E |
| resultNumber | 结果数量 |
| drawLineFlag | HVP是否根据结果集在视频上画框 |
| capturePicFlag | 未使用 |
| timestamp | 时间戳ms，如果是协处理器的方式，则该字段给帧ID |
| faceResult | 如果是人脸分析设备， 则使用该字段返回结果集 |
| commonResult | 如果是其他分析设备， 则使用该字段返回结果集 |

**Return:**

注册句柄

**Note:**

如果需要在视频流上画框, /mnt/mtd/code/fixed/sys.def里面的externtype字段需要为0x10000/0x20000/0x40000/0x80000

**Example**:

|  |
| --- |
| typedef struct \_CALLBACK\_FUNCTION\_WITH\_CH  {  int ch;  ai\_analyse\_callback callback\_func;  } CALLBACK\_FUNCTION\_WITH\_CH;  std::map<long, CALLBACK\_FUNCTION\_WITH\_CH> m\_callback\_list;  long AI\_AddAnalyseCallback (int p\_ch, ai\_analyse\_callback p\_callbackfunc)  {  long current\_handle = m\_handle++;  CALLBACK\_FUNCTION\_WITH\_CH obj;  obj.ch = p\_ch;  obj.callback\_func = p\_callbackfunc;  m\_callback\_list.insert(std::make\_pair(current\_handle, obj));  return current\_handle;  }  int threadbody()  {  YUVAnalyzeResult analyzeResult;  memset(&analyzeResult,0,sizeof(YUVAnalyzeResult));  analyzeResult.analyzeType = NSD\_AI\_FACE;  analyzeResult.drawLineFlag = 1;//if value is 1, nsd will draw a rectangle on the stream  analyzeResult.faceResult[0].resultPosition.u16X = 100；  analyzeResult.faceResult[0].resultPosition.u16Y = 100；  analyzeResult.faceResult[0].resultPosition.u16Width = 200；  analyzeResult.faceResult[0].resultPosition.u16Height = 200；  analyzeResult.faceResult[0].u32Color = 0x0000ff00;  analyzeResult.faceResult[0].u8NeedPrintOsd = 0;  analyzeResult.normalizationX = 1920;  analyzeResult.normalizationY = 1080;  std::map<long, CALLBACK\_FUNCTION\_WITH\_CH>::iterator it = CAiLib::GetInstance().m\_callback\_list.begin();  while (it != CAiLib::GetInstance().m\_callback\_list.end())  {  it->second.callback\_func(it->first, analyzeResult.analyzeType, &analyzeResult);  ++it;  }  return 0;  } |

## Int (\*IF\_RemoveAIAnalyseCallback)(unsigned short nCh,long hHandle);

**Function explain**: 移除注册的回调函数

**Parameter**:

|  |  |
| --- | --- |
| nCh | 通道ID |
| hHandle | 注册句柄 |

**Note:**

## Int (\*IF\_UpdateSys)(const char \* lpUpdateContent,unsigned long nContentLen,bool bAllowFreeBuffer,bool & bFreed,bool & bNeedReboot,update\_percent\_callback lpCallback,void \* lpContext1,void \* lpContext2);

**Function explain**: 设备升级

**Parameter**:

|  |  |
| --- | --- |
| lpUpdateContent | [in] 升级包内容 |
| nContentLen | [in] 升级包内容长度 |
| bAllowFreeBuffer | [in] 如果为true，接口调用完成后会释放lpUpdateContent |
| bFreed | [out] lpUpdateContent是否已经被释放 |
| bNeedReboot | [out]如果为true，则该接口调用完后需要重启设备 |
| lpCallback | [in] 升级进度回调函数 |
| lpContext1 | [in] 上下文1 |
| lpContext2 | [in] 上下文2 |

**Return:**

0: 升级成功

非0： 升级失败

**Note:**

## Int (\*IF\_UpdateChildSys)(unsigned short nTarget,unsigned short nTargetCh,const char\* lpUpdateBuf,unsigned long nBufSize);

**Function explain**: 设备升级

**Parameter**:

|  |  |
| --- | --- |
| nTarget | [in] 0本地 1远程 该接口需要设置为1 |
| nTargetCh | [in] 目标通道号 |
| lpUpdateBuf | [in] 升级包内容 |
| nBufSize | [in] 升级包长度 |

**Return:**

0: 升级成功

非0： 升级失败

**Note:**

## Long (\*IF\_AddYuvStreamCallback)(unsigned short nCh, video\_yuvdata\_callback lpCallback);

**Function explain**: AI模块向HVP注册回调，用来获取YUV数据。 如果为AI设备， 即/mnt/mtd/code/fixed/sys.def里面的externtype字段需要为0x10000/0x20000/0x40000/0x80000， 在从sensor获取数据后会通过该回调把数据给到AI模块进行预处理。

**Parameter**:

|  |  |
| --- | --- |
| nCh | [in] video channel |
| lpCallback | [in] call back function |

**Note:**

1. 这种方式AI模块在回调函数的实现中要尽快返回， 否则会阻塞主流程从而降帧。
2. 如果希望自行修改YUV，实现在画框、OSD、画点等可以使用此方法
3. 前拼8路设备必须使用此方法，其他机型可以使用IF\_GetYUVData来获取
4. IF\_AddYuvStreamCallback和IF\_GetYUVData区别是， IF\_GetYUVData是单独通道传递YUV数据， 所以分辨率可以和实时流分辨率不同。 IF\_AddYuvStreamCallback是把实时流回调给AI模块处理， 所以得到的YUV和实时流是同源的。

## Int (\*IF\_RemoveYuvStreamCallback)(unsigned short nCh,long hHandle);

**Function explain**: 移除回调函数

**Parameter**:

|  |  |
| --- | --- |
| nCh | [in] video channel |
| hHandle | [in] call back handle(Get the handle from IF\_AddYuvStreamCallback) |

**Note:**

## int (\*IF\_GetPlatformTypeList)();

**Function explain**: Remove call back functions from HVP

**Parameter**:

**Return：**

#define PTTYPE\_ONVIF 1

#define PTTYPE\_RTMP 2

#define PTTYPE\_ZENO 3

#define PTTYPE\_OWSP 4

#define PTTYPE\_GB28181 5

#define PTTYPE\_YNGT 6

#define PTTYPE\_TJCL408 7

#define PTTYPE\_YC 8

#define PTTYPE\_CUSTOMER 9

#define PTTYPE\_GB1400 10

例如，如果设备支持onvif和gb28181, 那么该接口会返回0x22(1<<PTTYPE\_ONVIF | 1<< PTTYPE\_GB28181)

**Note:**

## Long (\*IF\_AddAIYuvResultCallback)(unsigned short nCh,ai\_yuvresult\_callback lpCallback);

**Function explain**: 用作查找结果对应的原图。在协处理器的方式中，HVP会固定把BT1120数据的第一个字节设置为帧ID，范围是16-235. 协处理器返回的结果中需要给回这个帧ID，之后使用IF\_AddAIAnalyseCallback注册的回调把结果以及帧ID传入给HVP，接下来HVP会根据帧ID找到对应的原图，然后通过该接口把原图已经结果集给到AI模块进行处理

**Parameter**:

|  |  |
| --- | --- |
| nCh | [in] 通道号 |
| lpCallback | [in] 回调函数 |

**Note:**

/mnt/mtd/code/fixed/sys.def里面的externtype字段需要为0x40000

## int (\*IF\_RemoveAIAnalyseCallback)(unsigned short nCh,long hHandle);

**Function explain**: 注销回调

**Parameter**:

|  |  |
| --- | --- |
| nCh | [in] 通道号 |
| lpCallback | [in] 回调句柄 |

**Note:**

## Int (\*IF\_IsSDCardExist)();

**Function explain**: 用来判断SD卡是不是存在

**Parameter**:

**Return:**

0： 未插卡

1： 插卡

**Note:**

## int (\*IF\_SendLocalMsg)([LPNSD\_LOCAL\_MSG\_HEAD\_S](#_NSD_LOCAL_MSG_HEAD_S) lpMsgHead,const char \* lpMsg,unsigned int nMsgLen);

**Function explain**: 用作模块间消息传递， 通过该接口发送的数据， 会被所有通过IF\_AddLocalMsgCallback接口注册的回调所接收。

**Parameter**:

|  |  |
| --- | --- |
| lpMsgHead | [in] 消息头 |
| lpMsg | [in] 消息 |
| nMsgLen | [in] 消息长度 |

**Note:**

/mnt/mtd/code/fixed/sys.def里面的externtype字段需要为0x40000

**Example**:

|  |
| --- |
| std::string xmlstr="<?xml version=\"1.0\" encoding=\"GBK\"?>\n";  xmlstr+="<root>\n";  xmlstr+="<msg>\n";  xmlstr+="<chn>"+CHString::UIntToString(nCh)+"</chn>";  xmlstr+="<status>"+CHString::UIntToString(nDNStatus)+"</status>";  xmlstr+="</msg>\n";  xmlstr+="</root>\n";  NSD\_LOCAL\_MSG\_HEAD\_S local\_msg\_head;  memset(&local\_msg\_head,0,sizeof(NSD\_LOCAL\_MSG\_HEAD\_S));  local\_msg\_head.u32MsgType=NSD\_MSG\_DAY\_NIGHT\_SWITCH;  IF\_SendLocalMsg(&local\_msg\_head,xmlstr.c\_str(),xmlstr.length()); |

## Long (\*IF\_AddLocalMsgCallback)(local\_msg\_callback lpCallback,void \* pContext);

**Function explain**: 注册本地消息回调， 用于接收IF\_SendLocalMsg发送的数据

**Parameter**:

|  |  |
| --- | --- |
| lpCallback | [in] 回调函数 |
| pContext | [in] 上下文 |

**Note:**

## Int (\*IF\_RemoveLocalMsgCallback)(long hHandle);

**Function explain**: 移除消息回调

## Int (\*IF\_SendModuleMsg)([LPNSD\_MODULE\_MSG\_HEAD\_S](#_NSD_MODULE_MSG_HEAD_S) lpMsgHead,const char \* lpMsg,unsigned int nMsgLen,const void \*& pResultContent,unsigned int & nResultLen);

**Function explain**: 用作模块间消息传递， 通过该接口发送的数据， 会被所有通过IF\_SetModuleMsgCallback接口注册的回调所接收。

**Parameter**:

|  |  |
| --- | --- |
| lpMsgHead | [in] 消息头 |
| lpMsg | [in] 消息 |
| nMsgLen | [in] 消息长度 |
| pResultContent | [out] 回应信息内容 |
| nResultLen | [out] 回应信息长度 |

**Example**:

|  |
| --- |
| char szLocalMsg[4096] = {0};  sprintf(szLocalMsg, "<?xml version=\"1.0\" encoding=\"GBK\"?>\n"  "<root>\n"  "<msg order='1'>\n"  "<name>%s</name>\n"  "<sex>%d</sex>\n"  "<age>%d</age>\n"  "<begindate>%s</begindate>\n"  "<enddate>%s</enddate>\n"  "<cardid>%d</cardid>\n"  "<imgpath>%s</imgpath>\n"  "</msg>\n"  "</root>\n"  , name.c\_str()  , sex  , age  , objBeginDataTime.toString().c\_str()  , objEndDataTime.toString().c\_str()  , carid  , strFilePath.c\_str()  );  NSD\_MODULE\_MSG\_HEAD\_S head = {0};  head.u32MsgType = NSD\_MSG\_FACE\_TABLE\_ADD;  head.u32DestModule = NSDMOD\_AI;  unsigned int nResultLen = MAX\_MODULE\_MSG\_SIZE;  char szResult[MAX\_MODULE\_MSG\_SIZE];  const void \*result = szResult;  IF\_SendModuleMsg(&head,szLocalMsg,strlen(szLocalMsg),result,nResultLen); |

## Int (\*IF\_SetModuleMsgCallback)(unsigned int nModuleType,module\_msg\_callback lpCallback,void \* pContext);

**Function explain**: 用作模块间消息传递， 通过该接口发送的数据， 会被所有通过IF\_SetModuleMsgCallback接口注册的回调所接收。

**Parameter**:

|  |  |
| --- | --- |
| nModuleType | [in] 参见 NSD\_MODULE\_TYPE\_E |
| lpCallback | [in] 回调函数 |
| pContext | [in] 上下文 |

**Example**:

## Int (\*IF\_EncodeJpg)(int nIndex, [AI\_VIDEO\_FRAME\_S](#_AI_VIDEO_FRAME_S) \*lpVideoFrame, int nQuality, char \*&lpJpg, int &nJpgLength);

**Function explain**: 硬编码JPG

**Parameter**:

|  |  |
| --- | --- |
| nIndex | [in] 编码器序号范围0 - 1 |
| lpVideoFrame | [in] YUV |
| nQuality | [in] 质量 0-100 目前没用 |
| lpJpg | [out] jpg内容 |
| nJpgLength | [out] jpg长度 |

**Note：**

需要在2.8.5.20及之后版本使用

**Example**:

|  |
| --- |
| if(CAiLib::GetInstance().IF\_GetYUVData(0,&llpVideoFrame,200)!=0)  {  printf("get yuv data failed\n");  usleep(40000);  continue;  }  if(llpVideoFrame==NULL)  {  printf("yuv data error\n");  usleep(40000);  continue;  }  long long begintime = GetMicroSecond();  char \*pszJpg = NULL;  int nLength = 0;  int nRet = CAiLib::GetInstance().IF\_EncodeJpg(0, llpVideoFrame,10,pszJpg,nLength);  if (nRet != 0)  {  printf("encode failed\n");  }  else  {  printf("encode success length is %d, waste %lld\n", nLength, GetMicroSecond()-begintime);  }  delete[] pszJpg;  CAiLib::GetInstance().IF\_ReleaseYUVData(0, llpVideoFrame); |

## int (\*IF\_MmzAlloc)(unsigned long long \*pu32PhyAddr, void \*\*pVirtAddr, unsigned int nLen);

**Function explain**: 分配MMZ部分的内存

**Parameter**:

|  |  |
| --- | --- |
| pu32PhyAddr | [out] 物理地址 |
| pVirtAddr | [out] 虚拟地址 |
| nLen | [in] 期望分配的长度 |

**Note：**

需要在2.8.5.20及之后版本使用

## int (\*IF\_MmzFree)(unsigned long long pu32PhyAddr, void \*pVirtAddr);

**Function explain**: 释放MMZ部分的内存

**Parameter**:

|  |  |
| --- | --- |
| pu32PhyAddr | [in] 物理地址 |
| pVirtAddr | [in] 虚拟地址 |

**Note：**

需要在2.8.5.20及之后版本使用

# Data structure

## NSD\_TIME\_S

|  |  |
| --- | --- |
| u8Year | 1900 - now |
| u8Month | 1 – 12 |
| u8Day | 1 – 31 |
| u8Week | 1 – 7 |
| u8Hour | 0 – 23 |
| u8Minute | 0 – 59 |
| u8Second | 0 – 59 |
| u8Millisec | 0 – 999 |

## NSD\_SUMMER\_TIME\_S

|  |  |
| --- | --- |
| u8Month | 1 – 12 |
| u8WeekIndex | 1 – 5 . 当月的第几个星期 |
| u8Week | 1 – 7 |
| u8Hour | 0 – 23 |
| u8Minute | 0 – 59 |

## NSD\_SEGMENT\_S

|  |  |
| --- | --- |
| open | 0: 关闭1:开启 |
| beginday | 如果 type 为 0, 该字段单位为周，范围为 1 – 7  如果 type 为 1, 该字段单位为天，范围为 1 - 31 |
| beginhour | 0 – 23 |
| beginminute | 0 – 59 |
| beginsecond | 0 – 59 |
| beginmillisec | 0 - 999 |
| endday | 如果 type 为 0, 该字段单位为周，范围为 1 – 7  如果 type 为 1, 该字段单位为天，范围为 1 - 31 |
| endhour | 0 – 23 |
| endminute | 0 – 59 |
| endsecond | 0 – 59 |
| endmillisec | 0 - 999 |
| type | 0: beginday 和endday 单位为周  1: beginday 和 endday 单位为天 |

## NSD\_SIZE\_S

|  |  |
| --- | --- |
| cx | 宽度 |
| cy | 高度 |

## NSD\_RANGE\_S

|  |  |
| --- | --- |
| min | 最小值 |
| max | 最大值 |

## NSD\_POINT\_S

|  |  |
| --- | --- |
| x | 横坐标 |
| y | 纵坐标 |

## NSD\_AREA\_S

|  |  |
| --- | --- |
| u16X | 左上角横坐标 |
| u16Y | 左上角纵坐标 |
| u16Width | 区域宽度 |
| u16Height | 区域高度 |

## NSD\_CHN\_STREAM\_S

|  |  |
| --- | --- |
| u16ChannelNo | 通道号 |
| u16StreamNo | 码流号 |

## NSD\_RESOLUTION\_S

|  |  |
| --- | --- |
| width | 宽度 |
| height | 高度 |

## NSD\_ACTION\_S

|  |  |
| --- | --- |
| u8Valid | 0: 关闭 1:开启 |
| szActionName | 联动动作名称 |
| u8ActionType | typedef enum \_NSD\_ACTION\_TYPE\_E  {  NSD\_ACTION\_NONE,  NSD\_ACTION\_SNAP,  NSD\_ACTION\_RECORD,  NSD\_ACTION\_PTZ,  NSD\_ACTION\_DO,  NSD\_ACTION\_REBOOT,  }NSD\_ACTION\_TYPE\_E; |
| u8ChannelNo | 通道号 |
| u16PreSecond | 准备时长 |
| u32ActionMillisec | 动作时长 |
| u32SaveType | typedef enum \_NSD\_SAVE\_TYPE\_E  {  NSD\_SAVE\_NONE = 0,  NSD\_SAVE\_SD = 0x01,  NSD\_SAVE\_FTP = 0x02,  NSD\_SAVE\_EMAIL = 0x04,  NSD\_SAVE\_HOST = 0x08,  }NSD\_SAVE\_TYPE\_E; |
| u8FTPID | FTP 索引, -1:invalid |
| u8SMTPID | SMTP 索引, -1: invalid |
| u16SnapNum | 抓图张数 |
| u16SnapInterval | 抓图间隔 |
| u32RecStreamId | 录像的码流ID |
| u16PtzCmd | typedef enum \_NSD\_PTZCTL\_TYPE\_E  {  NSDPTZ\_UP=0, //(data2:speed)  NSDPTZ\_DOWN, //(data2:speed)  NSDPTZ\_LEFT, //(data1:speed)  NSDPTZ\_RIGHT, //(data1:speed)  NSDPTZ\_LEFT\_UP, //(data1,data2:speed)  NSDPTZ\_LEFT\_DOWN, //(data1,data2:speed)  NSDPTZ\_RIGHT\_UP, //(data1,data2:speed)  NSDPTZ\_RIGHT\_DOWN, //(data1,data2:speed)  NSDPTZ\_FOCUSADD, //focus near  NSDPTZ\_FOCUSSUB, //focus far  NSDPTZ\_IRISADD, //iris add  NSDPTZ\_IRISSUB, //iris reduce  NSDPTZ\_ZOOMADD, //zoom add  NSDPTZ\_ZOOMSUB, //zoom reduce  NSDPTZ\_STOP, //stop  NSDPTZ\_AUTOOPEN, //scan begin  NSDPTZ\_AUTOCLOSE, //scan end  NSDPTZ\_LAMPOPEN, //open lamp  NSDPTZ\_LAMPCLOSE, //close lamp  NSDPTZ\_BRUSHOPEN, //open brush  NSDPTZ\_BRUSHCLOSE, //close brush  NSDPTZ\_WATEROPEN, //open water  NSDPTZ\_WATERCLOSE, //close water  NSDPTZ\_SET\_PRESET, //set preset(data2: preset id)  NSDPTZ\_CALL\_PRESET, //call preset(data2: preset id)  NSDPTZ\_CLEAR\_PRESET, //remove preset(data2: preset id)  NSDPTZ\_FLIP, //mirror  NSDPTZ\_ZEROPAN, //go to zero pos  NSDPTZ\_RESET, //reset ptz  NSDPTZ\_TOUR\_RUN, //run tour(data1: tour index)  NSDPTZ\_TOUR\_STOP, //stop tour(data1: tour index)  NSDPTZ\_SET\_PT\_POSITION, // not impement  NSDPTZ\_SET\_ZOOM\_POSITION, // not impement  NSDPTZ\_SET\_ZOOM\_MULTI, // not impement  NSDPTZ\_UP\_LIMIT=100, // not impement  NSDPTZ\_DOWN\_LIMIT, // not impement  NSDPTZ\_LEFT\_LIMIT, // not impement  NSDPTZ\_RIGHT\_LIMIT, // not impement  NSDPTZ\_TOUR\_INSERT, //add preset to tour(no use)  NSDPTZ\_TOUR\_DELETE, //remove preset to tour(no use)  NSDPTZ\_TRACK\_REC, //begin track(data1:track index)  NSDPTZ\_TRACK\_SAVE, //stop trach(data1:track index)  NSDPTZ\_TRACK\_RUN, //run track(data1:track index)  NSDPTZ\_TRACK\_STOP, //stop track(data1:track index)  NSDPTZ\_SET\_PAN\_POS, //absolute pan position  NSDPTZ\_SET\_TILT\_POS, //absolute tilt position  NSDPTZ\_UP\_LIMIT\_CLEAR, // not impement  NSDPTZ\_DOWN\_LIMIT\_CLEAR, // not impement  NSDPTZ\_LEFT\_LIMIT\_CLEAR, // not impement  NSDPTZ\_RIGHT\_LIMIT\_CLEAR, // not impement  NSDPTZ\_MAX,  }NSD\_PTZCTL\_TYPE\_E; |
| u8PtzParam1 | Data1， 见上述注释 |
| u8PtzParam2 | Data2， 见上述注释 |
| u32DOStatus | 0: close 1: open |

## NSD\_SYS\_INFO\_S

|  |  |
| --- | --- |
| szDeviceID | 设备ID，MAC地址后六位 |
| szVersion | 软件版本号 |
| szMachineType | 机型号 |
| u8ChannelNum | 通道总数 |
| u8StreamNum | 码流总数 |
| u8AlarmInputNum | 报警输入数量 |
| u8AlarmOutputNum | 报警输出数量 |
| u8RS232Num | Rs232是否启用， 给球机用的串口 |
| u8RS485Num | RS485 是否启用 |
| u8AInEnable | 音频输入是否启用 |
| u8TalkEnable | 音频对讲是否启用 |
| u8DiskNum | SD卡是否启用 |
| u8Language | 默认语言  typedef enum \_NSD\_LANGUAGE\_TYPE\_E  {  NSD\_LANGUAGE\_NONE,  NSD\_LANGUAGE\_CHINESE,  NSD\_LANGUAGE\_ENGLISH,  NSD\_LANGUAGE\_RUSSIAN,  NSD\_LANGUAGE\_FRENCH,  NSD\_LANGUAGE\_JAPANESE,  NSD\_LANGUAGE\_GERMAN,  NSD\_LANGUAGE\_SPANISH,  NSD\_LANGUAGE\_PORTUGUESE,  NSD\_LANGUAGE\_POLISH,  NSD\_LANGUAGE\_CHINESE\_TRADITIONAL,  NSD\_LANGUAGE\_PERSIAN,  NSD\_LANGUAGE\_THAI,  NSD\_LANGUAGE\_CZECH,  NSD\_LANGUAGE\_TURKISH,  NSD\_LANGUAGE\_ITALIAN,  NSD\_LANGUAGE\_KOREAN,  NSD\_LANGUAGE\_VIETNAM,  NSD\_LANGUAGE\_BULGARIAN,  }NSD\_LANGUAGE\_TYPE\_E; |
| u8HardFlip | Sensor是否做硬件翻转  0: no flip  1: flip 180° |
| u8LensType | typedef enum \_NSD\_LENS\_TYPE\_E{  NSD\_LENS\_NONE,  NSD\_LENS\_FORECAM\_3\_10D5, //forecam 3-10.5  NSD\_LENS\_FORECAM\_7\_22, //forecam 7-22  NSD\_LENS\_YUTONG\_2D8\_12, //yutong 2.8-12  NSD\_LENS\_FORECAM\_2D8\_12,  NSD\_LENS\_FORECAM\_3D3\_10D5,  NSD\_LENS\_FORECAM\_9\_22,  NSD\_LENS\_2G,  NSD\_LENS\_FORECAM\_3D6\_10,  NSD\_LENS\_FORECAM\_CHANGLONG\_2D8\_12,  NSD\_LENS\_YUTONG\_2D8\_8,  NSD\_LENS\_YUTONG\_3D6\_11,  NSD\_LENS\_FISHEYE=0x80,  }NSD\_LENS\_TYPE\_E; |
| u16SensorType | Sensor类型 |
| u16FaceType | typedef enum \_NSD\_FACE\_TYPE\_E  {  NSD\_FACE\_NONE =0,  NSD\_FACE\_S =0x0001,  NSD\_FACE\_H =0x0002,  NSD\_FACE\_B =0x0004,  NSD\_FACE\_R =0x0010,  NSD\_FACE\_LUX =0x0020,  NSD\_FACE\_DPTZ =0x0040,  NSD\_FACE\_DC =0x1000,  }NSD\_FACE\_TYPE\_E; |
| u32ExternalType | 参见 NSD\_EXTERNAL\_TYPE\_E |
| u32ODM | ODM 号 |
| u8HWPlatform | 硬件平台 |
| u8HWPlatform\_child | 硬件子平台 |
| u32LanguageList | 支持的语言列表 |
| u8InnerVideoIn | 内部通道数 |
| u8WifiVersion | 0不支持wifi 1支持 |

## NSD\_DEVICE\_S

|  |  |
| --- | --- |
| szDeviceName | 设备名称 |
| u8MacAddr | MAC地址， 只读 |
| u32IPAddr | IP地址， 只读 |
| u32SubMask | 子网掩码， 只读 |
| u32GateWay | 网关， 只读 |
| u16ListenPort | 私有协议端口， 只读 |
| u16HttpPort | HTTP端口， 只读 |
| u16DoStatus | 报警输出状态，只读 |
| u16UpdatePort | 升级端口， 只读 |
| sysinfo | 系统信息， 只读 |
| u32DNSIP1 | DNS， 只读 |
| u32DNSIP2 | DNS， 只读 |
| u8DHCP1 | 是否开启DHCP， 只读 |
| u8DHCP2 | Reserved |
|  |  |

## NSD\_USER\_CFG\_S

|  |  |
| --- | --- |
| szName | 用户名 |
| szPassword | 密码 |
| u8GroupId | 用户所在组ID |
| u8Used | 是否有效 |

## NSD\_USER\_GROUP\_CFG\_S

|  |  |
| --- | --- |
| szName | 用户组名称 |
| u32Right | 权限  typedef enum \_NSD\_RIGHT\_TYPE\_E  {  NSD\_RIGHT\_PREVIEW=0x1,  NSD\_RIGHT\_PTZ=0x2,  NSD\_RIGHT\_MANAGE=0x4,  NSD\_RIGHT\_CONFIG=0x8,  }NSD\_RIGHT\_TYPE\_E; |

## NSD\_ETH\_CFG\_S

|  |  |
| --- | --- |
| u32IPAddr | IP地址，例如 0xc0a80102 表示 192.168.1.2 |
| u32SubMask | 子网掩码 |
| u32GateWay | 网关 |
| u8MacAddr | MAC地址 |
| u8DHCP1 | IP地址是否启用DHCP |
| u8DHCP2 | DNS是否启用DHCP |
| u8DHCPStatus | DHCP状态 |
| u32DNSIP1 | DNS 地址 |
| u32DNSIP2 | DNS 地址 |
| szIP6 | IPv6地址 |
| szSubMask6 | Ipv6 子网掩码 |
| szGateWay6 | Ipv6 网关 |

## NSD\_NET\_CFG\_S

|  |  |
| --- | --- |
| u16ListenPort | 私有协议监听端口 |
| u16HttpPort | HTTP端口 |
| u32MulticastAddr | 多播地址， 目前没用 |
| u16MulticastPort | 多播端口， 目前没用 |
| u16UpdatePort | 升级端口 |

## NSD\_PPPOE\_CFG\_S

|  |  |
| --- | --- |
| u8PPPOEOpen | 0:关闭 1:开启 |
| u8PPPOEStatus | 0:成功1:失败2:拨号中 |
| szPPPOEName | 用户 |
| szPPPOEPwd | 密码 |
| u32PPPOEAddr | PPPoE地址 |
| u32PPPOETimes | 在线时间 |

## NSD\_DDNS\_CFG\_S

|  |  |
| --- | --- |
| u8DDNSOpen | 0:关闭1:开启 |
| u8DDNSType | 0:dyndns 1:3322 |
| szDDNSAccount | DDNS账号 |
| szDDNSUsr | 用户名 |
| szDDNSPwd | 密码 |
| u16DDNSPort | 端口 |
| u32DDNSTimes | 更新时间 |

## NSD\_NTP\_CFG\_S

|  |  |
| --- | --- |
| u8NtpOpen | 0:关闭1:开启 |
| u8EnableSummer | 0: 不启用夏令时 1:启用夏令时 |
| u8NtpRefTime | NTP更新时间, 单位:hour |
| fTimeZone | 时区 -12 ~ 11 |
| szNtpURL | NTP 服务器域名 |
| summerBegin | 夏令时起始时间 |
| summerEnd | 夏令时结束时间 |
| u16SummerAdvanceTime | 夏令时时间偏移 |
| u16NtpPort | NTP服务器端口 |

## NSD\_UPNP\_CFG\_S

|  |  |
| --- | --- |
| u8UpnpOpen | 0: 关闭1:开启 |
| u8UpnpEthNo | 0: 本地网络 1:wifi |
| u8UpnpModel | 0: 固定映射 1: 自动映射 |
| u8UpnpRefTime | Upnp更新时间, 单位:hour |
| u32UpnpSvr | Upnp 服务器IP地址 |
| u16UpnpDataPort | 数据端口 |
| u16UpnpWedPort | 网页端口 |
| u16UpnpDataPortS | 映射后的外部数据端口 |
| u16UpnpWebPortS | 映射后的外部网页端口 |

## NSD\_FTP\_CFG\_S

|  |  |
| --- | --- |
| szFtpURL | FTP地址 |
| szFtpUsr | 用户名 |
| szFtpPsw | 密码 |
| szFtpPath | FTP根文件夹 |
| u16FtpPort | FTP端口 |

## NSD\_SMTP\_CFG\_S

|  |  |
| --- | --- |
| szMailHub | Smtp 服务器地址 |
| szMailAddr | 发送邮箱 |
| szMailUsr | 用户名 |
| szMailPsw | 密码 |
| szMailRctp | 收件人， 不同的地址用；隔开 |
| szTitle | 标题 |
| u8EnableSSL | 0：禁用ssl 1：启用ssl |
| u16MailPort | Smtp 端口 |

## NSD\_RTSP\_CFG\_S

|  |  |
| --- | --- |
| u8Authorized | 0: 不需要认证 1: 需要认证 |
| u16RtspPort | Rtsp 端口, 默认 554 |
| videoFlag | Sdp中的video字段, 不填则使用默认值 |
| audioFlag | Sdp中的audio字段, 不填则使用默认值 |
| audiobackFlag | Sdp中的audioback字段, 不填则使用默认值 |
| u16Mtu | mtu |

## NSD\_RTP\_CFG\_S

|  |  |
| --- | --- |
| u8RtpOpen | 0: 关闭RTP多播 1:启用RTP多播 |
| u8StreamId | 码流ID |
| u16RtpPort | Rtp 端口 |
| szRtpUrl | Rtp 地址 |
| u16Mtu | mtu |

## NSD\_ISP\_CFG\_S

|  |  |
| --- | --- |
| u8ColorMode | typedef enum \_NSD\_COLOR\_TYPE\_E{  NSD\_COLOR\_NONE,  NSD\_COLOR,  NSD\_BLACKWHITE,  NSD\_PSEUDO\_COLOR\_WHILTE\_HOT,  NSD\_PSEUDO\_COLOR\_IRON\_RED,  NSD\_PSEUDO\_COLOR\_GREED\_RED,  NSD\_PSEUDO\_COLOR\_LIGHT\_GREED\_RED,  NSD\_PSEUDO\_COLOR\_BLUE\_YELLOW,  NSD\_PSEUDO\_COLOR\_LIGHT\_BLUE\_YELLOW,  NSD\_PSEUDO\_COLOR\_BLUE\_GREED,  NSD\_PSEUDO\_COLOR\_BLACK\_HOT,  }NSD\_COLOR\_TYPE\_E; |
| u8Noise | 降噪 0 –100 |
| u8ImgStabilize | 图像稳定 0:close 1:open |
| u8WDRMode | WDR 0:close 1:low 2:high |
| u8Brightness | 0-100 |
| u8Saturation | 0-100 |
| u8Contrast | 0-100 |
| u8Sharpness | 0-100 |
| u8AutoIris | 0:manual 1:auto |
| u8IrisSpeed | 0-100 |
| u8Defog | 透雾强度 0-100 |
| u8GammaMode | Gamma mode |

Note:

You should get device capability to know the real range of each item.

## NSD\_AWB\_CFG\_S

|  |  |
| --- | --- |
| u8AWBMode | typedef enum \_NSD\_AWB\_MODE\_E{  NSD\_AWB\_NONE,  NSD\_AWB\_AUTO,  NSD\_AWB\_MANUAL,  NSD\_AWB\_SUNNY,  NSD\_AWB\_CLOUDY,  NSD\_AWB\_FLUORESCENT,  NSD\_AWB\_TUNGSTEN,  NSD\_AWB\_ATW,  NSD\_AWB\_INDOOR,  NSD\_AWB\_OUTDOOR,  }NSD\_AWB\_MODE\_E; |
| u8RGBSetting | 1:R 2:G 3:B |
| u16RGain | 0-100 手动模式生效 |
| u16GRGain | 0-100 手动模式生效 |
| u16GBGain | 0-100 手动模式生效 |
| u16BGain | 0-100 手动模式生效 |

## NSD\_AE\_CFG\_S

|  |  |
| --- | --- |
| u8AEMode | typedef enum \_NSD\_AE\_MODE\_E{  NSD\_AE\_NONE,  NSD\_AE\_AUTO,  NSD\_AE\_MANUAL,  NSD\_AE\_SHUTTER,  NSD\_AE\_IRIS,  NSD\_AE\_AGC,  }NSD\_AE\_MODE\_E; |
| u8Metering | typedef enum \_NSD\_AE\_METERING\_TYPE\_E{  NSD\_AE\_METERING\_AVERAGE,  NSD\_AE\_METERING\_CENTER,  NSD\_AE\_METERING\_SPOT,  NSD\_AE\_METERING\_CUSTOM,  }NSD\_AE\_METERING\_TYPE\_E; |
| u8AntiFlickerMode | 50HZ/60HZ |
| u32MaxShutterTime | 最大快门速度 自动模式生效 |
| u32FixShutterTime | 固定快门速度 手动模式生效 |
| u32MaxGain | 最大增益0-100 自动模式生效 |
| u32FixGain | 固定增益0-100 手动模式生效 |
| u8MeterWeight | 曝光权重 |
| u8NUCMode | NUC mode |
| u8BLC | 背光补偿 0:close 1:open |
| u8HL | 高光抑制 0:close 1:open |
| u16NUCInterval | NUC interval, unit:second |

## NSD\_IMG\_CFG\_S

|  |  |
| --- | --- |
| u8ImgProfile | 0:自动,1:白天配置,2:夜晚配置 |
| u8ICRMode | 0:主动模式,1:被动模式,2:计划模式 |
| u8DNSensitivity | 0-100 日夜切换灵敏度 |
| u8D2NThreshold | 0-100 日转夜切换阈值， 主动模式生效 |
| u8N2DThreshold | 0-100 夜转日切换阈值， 主动模式生效 |
| u32D2NSec | 白天切换夜晚的时间(3600\*hour+60\*min+sec)。 计划模式生效 |
| u32N2DSec | 夜晚切换白天的时间(3600\*hour+60\*min+sec)。 计划模式生效 |

## NSD\_VEC\_S

|  |  |
| --- | --- |
| u32MaxResolution | 最大分辨率, 宽\*高 |
| u8Qulity | 1-9 越高质量越好 |
| u8BitrateType | typedef enum \_NSD\_BITRATE\_TYPE\_E  {  NSD\_BITRATE\_NONE=0,  NSD\_BITRATE\_CBR,  NSD\_BITRATE\_VBR,  NSD\_BITRATE\_VBR\_PLUS,  }NSD\_BITRATE\_TYPE\_E; |
| u16EncodeType | typedef enum \_NSD\_CODEC\_TYPE\_E  {  NSD\_CODEC\_NONE,  NSD\_CODEC\_H264,  NSD\_CODEC\_MJPEG,  NSD\_CODEC\_G711A,  NSD\_CODEC\_G711U,  NSD\_CODEC\_PCM,  NSD\_CODEC\_H265,  NSD\_CODEC\_AAC,  }NSD\_CODEC\_TYPE\_E; |
| u16KeyInterval | I帧间隔 |
| u16ImageWidth | 宽度 |
| u16ImageHeight | 高度 |
| u32Bitrate | 码率 |
| u16FRNumerator | 帧率分母 |
| u16FRDenominator | 帧率分子 |
| u8BPInterval | 0: P 1: BP 2:BBP |
| u8Profiles | 1:baseline 2:main profile 3:high profile |

## NSD\_VIDEO\_CFG\_S

|  |  |
| --- | --- |
| u16ViImageWidth | 源分辨率宽度 |
| u16ViImageHeight | 源分辨率高度 |
| u8System | typedef enum \_NSD\_VIDEO\_SYSTEM\_TYPE\_E{  NSD\_VIDEO\_PAL=1,  NSD\_VIDEO\_NTSC,  }NSD\_VIDEO\_SYSTEM\_TYPE\_E; |
| u8RotateMode | typedef enum \_NSD\_ROTATE\_MODE\_E{  NSD\_ROTATE\_NONE,  NSD\_ROTATE\_90,  NSD\_ROTATE\_180,  NSD\_ROTATE\_270,  }NSD\_ROTATE\_MODE\_E; |
| u8Mirror | typedef enum \_NSD\_MIRROR\_MODE\_E{  MSD\_MIRROR\_NONE,  NSD\_MIRROR\_HORI,  NSD\_MIRROR\_VERT,  NSD\_MIRROR\_BOTH,  }NSD\_MIRROR\_MODE\_E; |
| u32SetMask | 0:全部设置,0x1:vec[0],0x2:vec[1],0x4:vec[2] eg |

## NSD\_VOUT\_CFG\_S

|  |  |
| --- | --- |
| u8Enable | 0:disable 1:enable |
| u8VOutDev | 输出方式， 参见NSD\_VOUT\_DEV\_TYPE\_E |
| u32VOutMode | 输出的视频格式， 参见NSD\_VOUT\_DEV\_MODE\_E |
| u8SettingMask | 0：设置所有VO参数 1：设置ALPHA和Colorkey |
| u8ChangeType | 0： 根据video\_cfg决定VO输出是否旋转镜像  1： 根据 u8RotateMode和u8Mirror决定VO是否旋转镜像 |
| u8RotateMode | VO是否旋转 |
| u8Mirror | VO是否镜像 |
| u8ScreenType | 屏类型  0)： 1200\*1920 8寸屏 TV080WUM-NH1  1)： 800\*1280 10.1寸屏 JD9365\_BOE10.1  2)： 800\*1280 8寸屏 JD9366\_BOE8.0  3)： 800\*1280 7寸屏 JD9366\_BOE7.0  4)： 720\*1280 5寸屏 NT35523  5): 720\*1280 5寸屏 ST7703 |
| u8Alpha | FB层的透明度 0 - 255 |
| u8ColorKeyEnable | 是否启用Colorkey |
| u32ColorKey | 掩码色， 例如红色为0x00ff0000, 绿色为0x0000ff00 |

## NSD\_FISHEYE\_CFG\_S

|  |  |
| --- | --- |
| u8EnableFlag | 0:不启用 1:启用 |
| u8MountMode | typedef enum \_NSD\_FISHEYE\_MOUNT\_MODE\_E  {  NSD\_FISHEYE\_DESKTOP\_MOUNT = 0, /\* desktop mount mode \*/  NSD\_FISHEYE\_CEILING\_MOUNT = 1, /\* ceiling mount mode \*/  NSD\_FISHEYE\_WALL\_MOUNT = 2, /\* wall mount mode \*/  }NSD\_FISHEYE\_MOUNT\_MODE\_E; |
| u8LayoutMode | typedef enum \_NSD\_FISHEYE\_LAYOUT\_MODE\_E  {  NSD\_FISHEYE\_LAYOUT\_ONE = 0,  NSD\_FISHEYE\_LAYOUT\_FOUR = 1,//2x2  NSD\_FISHEYE\_LAYOUT\_FOUR\_M1 = 2,  NSD\_FISHEYE\_LAYOUT\_FOUR\_M2 = 3  }NSD\_FISHEYE\_LAYOUT\_MODE\_E; |

## NSD\_MASK\_CFG\_S

|  |  |
| --- | --- |
| u8OpenMask | 0:不启用视频遮挡 1:启用视频遮挡 |
| u8SetAreaMask | 0:set all,0x1 0x2 0x4 eg. |
| area | Area list |

## NSD\_OSD\_AREA\_S

|  |  |
| --- | --- |
| u8Show | 0:不启用该条OSD 1:启用该条OSD |
| u32Color | typedef enum \_NSD\_COLOR\_INDEX\_E{  NSD\_COLOR\_BLACK=1,  NSD\_COLOR\_RED,  NSD\_COLOR\_GREEN,  NSD\_COLOR\_YELLOW,  NSD\_COLOR\_BLUE,  NSD\_COLOR\_MAGENTA,  NSD\_COLOR\_CYAN,  NSD\_COLOR\_WHITE,  }NSD\_COLOR\_INDEX\_E; |
| area | Area list |

## NSD\_OSD\_CFG\_S

|  |  |
| --- | --- |
| dateOsd | 数据OSD |
| titleOsd | 标题OSD |
| custom1Osd | 用户自定义OSD |
| custom2Osd | 用户自定义OSD |
| alarmOsd | 报警OSD |
| dateType | typedef enum \_NSD\_DATESHOW\_TYPE\_E  {  NSD\_DATE\_FORMAT1=1, //YYYY-MM-DD hh:mm:ss ww  NSD\_DATE\_FORMAT2, //hh:mm:ss YYYY-MM-DD ww  NSD\_DATE\_FORMAT3, //MM/DD/YYYY hh:mm:ss ww  NSD\_DATE\_FORMAT4, //hh:mm:ss MM/DD/YYYY ww  }NSD\_DATESHOW\_TYPE\_E; |
| titleInfo | 标题显示文本 |
| customInfo1 | 用户自定义显示文本 |
| customInfo2 | 用户自定义显示文本 |
| u32FontSize | typedef enum \_NSD\_FONT\_LEVEL\_E  {  NSD\_FONT\_AUTO,  NSD\_FONT\_LARGE,  NSD\_FONT\_SMALL,  }NSD\_FONT\_LEVEL\_E; |
| u32Transparency | 0 - 100 |
|  |  |

## NSD\_ROI\_CFG\_S

|  |  |
| --- | --- |
| u8Open | 0:关闭ROI 1:启用ROI |
| u8SetAreaMask | 0:设置所有 |
| roi | Roi area list |

## NSD\_MOTION\_CFG\_S

|  |  |
| --- | --- |
| u8OpenDetect | 0:禁用移动侦测 1:启用移动侦测 |
| u8Sensitivity | 单个宏块报警灵敏度， 0-100 越大越灵敏。 |
| u8Threshold | 总体宏块报警门限， 0-100 越大越容易报警 |
| u32Persistent | 报警持续时间 |
| u8Block | 检测区域设置 |
| segment | 检测时间段 |
| u32ActionId | 联动ID列表 |

## NSD\_SCHEME\_CFG\_S

|  |  |
| --- | --- |
| u8OpenDetect | 0: 不启用计划 1:启用计划 |
| u8Type | 0: 持续动作(条件满足就会执行) 1: 触发式动作(一个周期内只执行一次) |
| segment | 时间段 |
| u32ActionId | 联动ID列表 |
| szSchemeName | 计划名称 |

## NSD\_FILTER\_CFG\_S

|  |  |
| --- | --- |
| u8OpenIPFilter | 0: 不启用IP过滤 1:启用IP过滤 |
| ipaddr | IP列表 |
| u8OpenMacFilter | 0: 禁用MAC过滤 1:启用MAC过滤 |
| macaddr | MAC列表 |

## NSD\_AUDIO\_CFG\_S

|  |  |
| --- | --- |
| u8Open | 0: 禁用音频 1:启用音频 |
| u8InputType | typedef enum \_NSD\_AUDIO\_INPUT\_TYPE\_E  {  NSD\_AIN\_TYPE\_NONE,  NSD\_AIN\_TYPE\_MIC,  NSD\_AIN\_TYPE\_LINE,  }NSD\_AUDIO\_INPUT\_TYPE\_E; |
| u8EncodeType | typedef enum \_NSD\_CODEC\_TYPE\_E  {  NSD\_CODEC\_NONE,  NSD\_CODEC\_H264,  NSD\_CODEC\_MJPEG,  NSD\_CODEC\_G711A,  NSD\_CODEC\_G711U,  NSD\_CODEC\_PCM,  NSD\_CODEC\_H265,  NSD\_CODEC\_AAC,  }NSD\_CODEC\_TYPE\_E; |
| u8Bits | 16,32 |
| u32Samples | 8000, 16000,48000 |
| u8CaptureVolume | 0-255 |
| u8PlaybackVolume | 0-255 |
| u16PacketLen | 320 |
| u8ChannelNum | 1 |

## NSD\_PTZ\_POS\_S

|  |  |
| --- | --- |
| u32Pan | 0-36000 |
| u32Tilt | 0-18000 |
| u32Zoom | 0-max, for instance 1x = 100 2x=200 30x=3000 |

## NSD\_PTZ\_CFG\_S

|  |  |
| --- | --- |
| u16PTZProtocol | 0: pelco-d 1:pelco-p |
| u16Addr | 1 |
| u8UartNo | 0:rs232 1:rs485 |
| u8WatchEnable | 0: 禁用看守位 1: 启用看守位 |
| u8WatchAction | 看守位动作ID |
| u8WatchActionType | 看守位类型 0:none 1:preset 2:tour 3:track 4:home |
| u32WatchTime | 看守位时间，单位: ms |
| u8PowerActionType | 上电动作类型. 0:none 1:preset 2:tour 3:track 4:last action |
| u8PowerAction | 上电动作ID |
| u16SensorType | Not use |
| u32ZoomPos | Not use |
| u32PowerTime | 上电动作时间 , 单位 : ms |
| u8MemoryOperateType | 最后动作 0:none,1:tour,2:track,4:autoopen[inner used] |
| u8MemoryOperate | 最后动作ID |

## NSD\_PTZ\_CTRL\_S

|  |  |
| --- | --- |
| u8UartNo | 0: rs232 1:rs485 |
| u8ChannelNo | 1 |
| u8ProtoType | 0xff |
| u8Command | typedef enum \_NSD\_PTZCTL\_TYPE\_E  {  NSDPTZ\_UP=0, //(data2:speed)  NSDPTZ\_DOWN, //(data2:speed)  NSDPTZ\_LEFT, //(data1:speed)  NSDPTZ\_RIGHT, //(data1:speed)  NSDPTZ\_LEFT\_UP, //(data1,data2:speed)  NSDPTZ\_LEFT\_DOWN, //(data1,data2:speed)  NSDPTZ\_RIGHT\_UP, //(data1,data2:speed)  NSDPTZ\_RIGHT\_DOWN, //(data1,data2:speed)  NSDPTZ\_FOCUSADD, //focus near  NSDPTZ\_FOCUSSUB, //focus far  NSDPTZ\_IRISADD, //iris add  NSDPTZ\_IRISSUB, //iris reduce  NSDPTZ\_ZOOMADD, //zoom add  NSDPTZ\_ZOOMSUB, //zoom reduce  NSDPTZ\_STOP, //stop  NSDPTZ\_AUTOOPEN, //scan begin  NSDPTZ\_AUTOCLOSE, //scan end  NSDPTZ\_LAMPOPEN, //open lamp  NSDPTZ\_LAMPCLOSE, //close lamp  NSDPTZ\_BRUSHOPEN, //open brush  NSDPTZ\_BRUSHCLOSE, //close brush  NSDPTZ\_WATEROPEN, //open water  NSDPTZ\_WATERCLOSE, //close water  NSDPTZ\_SET\_PRESET, //set preset(data2: preset id)  NSDPTZ\_CALL\_PRESET, //call preset(data2: preset id)  NSDPTZ\_CLEAR\_PRESET, //remove preset(data2: preset id)  NSDPTZ\_FLIP, //mirror  NSDPTZ\_ZEROPAN, //go to zero pos  NSDPTZ\_RESET, //reset ptz  NSDPTZ\_TOUR\_RUN, //run tour(data1: tour index)  NSDPTZ\_TOUR\_STOP, //stop tour(data1: tour index)  NSDPTZ\_SET\_PT\_POSITION, // not impement  NSDPTZ\_SET\_ZOOM\_POSITION, // not impement  NSDPTZ\_SET\_ZOOM\_MULTI, // not impement  NSDPTZ\_UP\_LIMIT=100, // not impement  NSDPTZ\_DOWN\_LIMIT, // not impement  NSDPTZ\_LEFT\_LIMIT, // not impement  NSDPTZ\_RIGHT\_LIMIT, // not impement  NSDPTZ\_TOUR\_INSERT, //add preset to tour(no use)  NSDPTZ\_TOUR\_DELETE, //remove preset to tour(no use)  NSDPTZ\_TRACK\_REC, //begin track(data1:track index)  NSDPTZ\_TRACK\_SAVE, //stop trach(data1:track index)  NSDPTZ\_TRACK\_RUN, //run track(data1:track index)  NSDPTZ\_TRACK\_STOP, //stop track(data1:track index)  NSDPTZ\_SET\_PAN\_POS, //absolute pan position  NSDPTZ\_SET\_TILT\_POS, //absolute tilt position  NSDPTZ\_UP\_LIMIT\_CLEAR, // not impement  NSDPTZ\_DOWN\_LIMIT\_CLEAR, // not impement  NSDPTZ\_LEFT\_LIMIT\_CLEAR, // not impement  NSDPTZ\_RIGHT\_LIMIT\_CLEAR, // not impement  NSDPTZ\_MAX,  }NSD\_PTZCTL\_TYPE\_E; |
| u8Data1 | Data1, 参见上述注释 |
| u8Data2 | Data2, 参见上述注释 |

## NSD\_PTZ\_POSITION\_S

|  |  |
| --- | --- |
| u8ChannelNo | 1 |
| u8Action | set mask:1-p,2-t,4-z |
| u16Pan | 0~36000(\*100) |
| u16Tilt | 0~9000(\*100) |
| u16Zoom | 100~(\*100) |

## NSD\_DI\_CFG\_S

|  |  |
| --- | --- |
| u8Type | 0：NC 1：NO |
| u8OpenDetect | 0：Disable detect 1: Enable detect |
| segment | 检测时间 |
| u32ActionId | 联动动作ID列表 |

## NSD\_DO\_CFG\_S

|  |  |
| --- | --- |
| u16Status | 报警输出状态. |
| u16Mask | 报警ID |

## NSD\_UART\_CFG\_S

|  |  |
| --- | --- |
| u32BaudRate | 波特率300---115200 |
| u8Databit | 数据位0=5bit, 1=6bit, 2=7bit, 3=8bit |
| u8Stopbit | 停止位1=1bit, 2=2bit |
| u8Parity | 奇偶校验 |
| u8FlowCtrl | 流控 |
| devName | 设备名, 比如/dev/ttyAMA3 |
| u8GpioNo | 是否需要GPIO控制发送和接收(半双工) |
| u8UsePTZ | 如果期望该串口可以发送PTZ指令则设置为1， 否则为0 |

## NSD\_PLATFORM\_ONVIF\_S

|  |  |
| --- | --- |
| u8Enable | 0: 禁用 onvif 1:启用 onvif |
| u8EnableTalk | 0: 禁用对讲 1:启用对讲 |
| u8SyncTime | 0: 禁用同步时间 1:启用同步时间 |
| u8SyncParam | 0: 禁用同步参数 1:启用同步参数 |
| u8EnableMeta | 0: 禁用元数据 1:启用元数据 |
| u8Authorized | 0: 禁用认证 1:启用认证 |
| szCompanyName | 公司名称 |
| szHostName | Host name |
| szLocation | Location |

## NSD\_PLATFORM\_RTMP\_S

|  |  |
| --- | --- |
| u8Enable | 0: 禁用 rtmp 1:启用 rtmp |
| url | Rtmp server url |

## NSD\_AI\_FACE\_S

|  |  |
| --- | --- |
| u8Open | 0: 禁用人脸检测 1:启用人脸检测 |
| u8DrawLineFlag | 是否在实时流上面画框 |
| u8Sensitivity | 灵敏度 0-100 |
| u8CaptureFaceFlag | 0不抓取人脸 1抓人脸 2抓原图 3抓人脸和原图 |
| u16MinFacePixel | 最小人脸像素 |
| u8Blur | 模糊值 0-100 |
| u8Front | 正脸值 0-100 |
| u8SnapMode | NSD\_FACE\_SNAP\_TYPE\_E |
| u8SnapNumber | 抓拍张数，当抓拍模式为 NSDFACE\_MODE\_NUMBER时生效 |
| u8MaxFaceNumber | 最大检测人脸数 |
| u8SceneMode | Unused |
| u16MaxFacePixel | 最大人脸检测像素 |
| u8FaceAreaCoefficient | 人脸扩展系数，抓拍的时候按该系数扩大抓图区域 |
| u8PictureEncodeQuality | 图片质量0-100 |
| segment | 检测时间 |
| u32ActionId | 联动动作列表 |
| roiArea | 人脸检测区域 |
| filterArea | 不检测人脸的区域 |
| algVersion | 算法版本 |
| swVersion | 算法软件版本 |

## NSD\_AI\_S

|  |  |
| --- | --- |
| u16AiImageWidth | AI通道YUV的宽度 |
| u16AiImageHeight | AI通道YUV的高度 |
| u8PixelFormat | 1: YV12 2:Not support 3:NV21 (建议) |
| u8FrameAllocType | 目前没用 |
| u8CacheNumber | 实时流缓存帧数 ，范围2-8。 算法速度快的话可以设置小一些，算法速度慢建议设置大些， 避免画框不准 |
| u8CacheNumberForVout | 视频输出缓存帧数，范围2-8。算法速度快的话可以设置小一些，算法速度慢建议设置大些， 避免画框不准 |

Note:

海思芯片默认出的视频格式为NV21， 如果u8PixelFormat设置为1， HVP内部会做格式转换， 从而消耗一定时间。 所以u8PixelFormat字段建议设置为3.

## NSD\_LOCAL\_MSG\_HEAD\_S

|  |  |
| --- | --- |
| u32MsgType | 消息类型， 参见NSD\_LOCAL\_MSG\_TYPE\_E |

## NSD\_MODULE\_MSG\_HEAD\_S

|  |  |
| --- | --- |
| u32MsgType | 消息类型， 参见NSD\_LOCAL\_MSG\_TYPE\_E |
| u32DestModule | 目的模块， 参见NSD\_MODULE\_TYPE\_E |
| u8Misc | 子模块， 目前支持  LOCAL\_CHANNEL\_TYPE\_NULL和LOCAL\_CHANNEL\_TYPE\_UI |

## NSD\_SCREEN\_CFG\_S

|  |  |
| --- | --- |
| u8ShowFlag | VO是否显示 0：不显示 1：显示 |
| u8SaveEnergyFlag | 0 不节能 1节能 |
| u8Luma | 画面亮度 0-100，此处设置的是画面的亮度，并非屏幕亮度， 屏幕亮度需要使用IF\_ProcCmd接口设置NSDCMD\_SCREEN\_LUMA |
| u8Contrast | 画面对比度 |
| u8Hue | 画面色度 |
| u8Saturation | 画面饱和度 |
| u8SaveEnergyBrightness | 节能模式下屏幕亮度 0-100 |
| u8NormalBrightness | 正常状态下屏幕亮度 0-100 |

## NSD\_WIEGAND\_CFG\_S

门禁机型专用

|  |  |
| --- | --- |
| u8Enable | 0：禁用韦根输出 1：启用韦根输出 |
| u8WiegandType | 韦根类型 0:26bit 1:34bit |
| u32Duration | 周期， 目前没用 |

## NSD\_RELAY\_CTL\_DOOR\_S

门禁机型专用

|  |  |
| --- | --- |
| u8Enable | 0：禁用继电器输出 1：启用继电器输出 |
| u8IOIndex | IO索引 |
| u8ValidLevel | 1：NO 0：NC |
| u32Duration | 周期 |

## NSD\_HTTP\_CFG\_S

HTTP上传图片，门禁专用

|  |  |
| --- | --- |
| u8Enable | 0：禁用HTTP上传 1：启用HTTP上传 |
| u8UploadType | 0：只传图片 1：上传所有信息 |
| u16Port | HTTP端口 |
| u32IPAddr | HTTP地址 |
| szUrl | 上传URL |

## AI\_VIDEO\_FRAME\_S

|  |  |
| --- | --- |
| nWidth | YUV宽度 |
| nHeight | YUV高度 |
| nPixelFormat | 1:yuv420(YV12) 2:yuv422(暂不支持) 3:yuv420 (NV21) |
| pYUV | * 如果nPixelFormat为 1 or 2, pYUV[0] 表示Y数据,pYUV[1]表示U数据, pYUV[2]表示V数据 * 如果nPixelFormat为3, pYUV[0] 表示YUV数据, pYUV[1] 和 pYUV[2] 不需要使用 |
| nStride | YUV跨度 |
| nPts | 时间戳 |
| pInnerUse | 内部使用 |

## PictureData

typedef HDSOFT::CHAutoPtr<PictureData> SPictureData;

|  |  |
| --- | --- |
| m\_tick | 图片在缓存中的索引号 |
| m\_data | 图片数据 |
| m\_datalen | 图片长度 |
| m\_savetype | 保存类型， 参见NSD\_SAVE\_TYPE\_E (nsddefines.h) |
| m\_fromtype | 图片触发来源, 参见NSD\_ACTION\_FROM\_E (privydef.h) |
| m\_createtime | 图片创建时间 |
| m\_head | 图片头信息 |

## NSD\_SNAP\_HEAD\_S

|  |  |
| --- | --- |
| u16Magic | 魔法字， 固定为0x7470 |
| u8TotleNum | 此次会话中总图片数 |
| u8No | 此图片索引 |
| u8Type | 图片类型，参见NSD\_AI\_ANALYSE\_TYPE\_E |
| u8Company | 公司名称 |
| u16MicroTimeStamp | 时间戳毫秒部分 |
| u32TimeStamp | 时间戳 单位秒 |
| faceResult | 人脸检测区域 |
| commonResult | 检测区域 |
| u32SessionId | 会话ID |

## NSD\_RECORD\_FILE\_S

|  |  |
| --- | --- |
| u8ChannelNo | 通道号 |
| u8StreamId | 码流ID |
| u8MediaType | 媒体类型 1：视频 2：图片 |
| u8RecSize | 总录像大小 |
| u8Mark | 暂时没用 |
| tmStart | 录像起始时间 |
| tmEnd | 录像结束时间 |

## NSD\_FRAME\_HEAD\_S

|  |  |
| --- | --- |
| u16FrameFlag | 固定为0x7468 |
| u8FrameType | 帧类型， 参见NSD\_FRAME\_TYPE |
| u8HasChildFrame | 是否有子通道数据， 后拼全景使用 |
| u32FrameNo | 帧ID |
| u32TimeStamp | 时间戳 |
| u32WallClock | 挂钟时间 |
| videoInfo | 视频信息 |
| audioInfo | 音频信息 |
| u32FrameSize | 帧大小 |