网络 SDK 开发手册

VERSION 1.0.0.1 (Build 20180223)

2018-02-23

前　　言

非常感谢您使用我们公司的设备，我们将为您提供最好的服务。

本手册可能包含技术上不准确的地方或印刷错误，欢迎指正。我们将会定期更新手册的内容。

目　　录

1. 简 介

1.1 概述

欢迎使用我公司网络SDK编程手册,网络SDK是软件开发商在开发我司网络硬盘录像机监控联网应用时的开发套件。本文档详细描述了开发包中各个函数实现的功能、接口及其函数之间的调用关系和示例实现。

开发包所包括的文件有：

网络库	**NetSDK**	头文件
	NetSDK.lib	**Lib** 文件
	NetSDK.dll	接口库
辅助库	**DllDeinterlace.dll**	解码辅助库
	H264Play.dll	解码辅助库
	hi_h264dec_w.dll	解码辅助库

1.2 适用性

- 支持网络硬盘录像机的监视、回放、报警、远程配置、日志查询等功能。
- 支持 TCP 传输模式，设备端同时支持 10 个 TCP 连接。
- 可通过 SDK 回调接口开发流媒体转发、回放、报警等服务器程序。
- 客户端可以采用多种分辨率进行图像预览，支持的分辨率包括： QCIF、CIF、2CIF、、HalfD1、D1，VGA（640×480）等
- SDK 在录像回放/下载时,同一登陆 ID 对于同一通道在同一时间回放和下载不可同时进行操作。
- SDK 性能与设备的运行情况和运行客户端的计算机 CPU 能力密切相关,理论上能同时支持 2000 个用户注册; 同时支持 2000 路网络预览和网络回放;同时支持 2000 路报警上传;在图象显示方面同时支持 300 路。

设计原则

1.3 编程说明

- **初始化和清除**

1、 使用网络客户端软件包首先调用 **H264_DVR_Init()**对系统进行初始化，应用程序退出时调用 **H264_DVR_Cleanup()**释放所有占用的资源。

2、 大多数函数调用均应该在 **H264_DVR_Init()**之后，**H264_DVR_Cleanup()**之前，而 H264_DVR_GetLastError 可以在任何时候调用等等。

- **用户登录和注销**

用户在访问前端设备之前必须通过调用**H264_DVR_Login()**登录到前端设备上。此句柄就像一个会话通道，之后该用户可通过此句柄访问前端设备。退出该会话时则通过 H264_DVR_Logout()函数在前端设备上注销此句柄以终止该会话通道的使用。建立连接与登录是同步的。

- **心跳功能**

在本开发包中提供自动心跳功能(20 秒一次心跳)当设备断开能及时回调给客户端。

- **同步与异步**

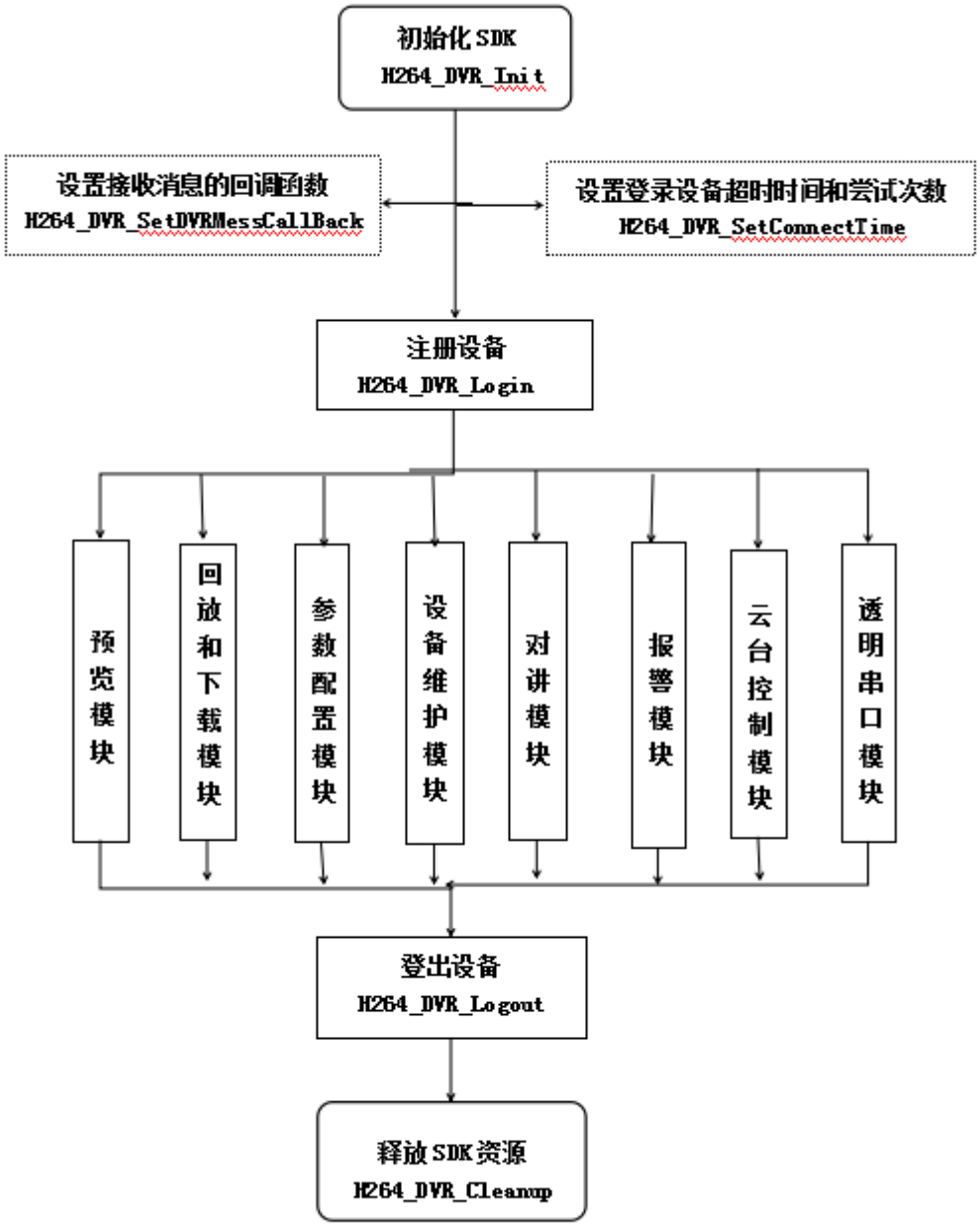
异步通过设置回调函数的方式实现，网络数据通过回调函数传达到应用程序,有些异步在设置后返回请求句柄，结束请求时将请求句柄提供给 SDK 以注销相关资源。

- **回调函数**

一般都有 *dwUser* 参数，由用户自定义需要的数据，一般用来传入类对象指针，方便回调处理在类中实现,回调应用都可以采取这种方式。

1.4 典型调用顺序

1.4.1 SDK 调用的基本流程图



其中虚线框的流程是可选部分，不会影响其他流程和模块的功能使用。按实现功能的不同可以分为十个模块。如果想实现每个模块的功能，就必须有初始化 SDK、注册设备、

登出设备和释放SDK资源这四个流程。

- 初始化 SDK（ H264_DVR_Init ()）：对整个网络 SDK 系统的初始化，内存预分配等操作。
- 设置连接超时时间（H264_DVR_SetConnectTime）：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。
- 设置接收异常消息的回调函数（H264_DVR_SetDVRMessCallBack）：此接口用于接收报警模块发生的异常信息。用户可以在初始化 SDK 后就设置该回调函数。
- 用户注册设备（H264_DVR_Login）：实现用户的注册功能，注册成功后，返回的用户 ID 作为其他功能操作的唯一标识。
- 预览模块：从前端设备取实时码流，解码显示以及播放控制等功能，同时支持软解码和解码卡解码。具体流程详见实时预览模块流程。
- 回放和下载模块：可以通过按时间和按文件名的方式远程回放或者下载的录像文件，后续可以进行解码或者存储。具体流程详见回放和下载模块流程。
- 参数配置模块：设置和获取前端设备的参数，主要包括设备参数、网络参数、通道参数、串口参数、报警参数、异常参数、和用户配置等参数信息。具体流程详见参数配置模块流程。
- 远程设备维护模块：实现关闭设备、重启设备和远程升级等维护工作。具体流程详见设备维护模块流程。
- 语音对讲转发模块：实现和前端设备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：处理设备上传的各种报警信号。报警分为“布防”和“监听”两种方式，在采用监听方式并且不需要获取用户 ID 的情况下，报警模块可以无需进行“用户注册”操作步骤。具体流程详见报警模块流程。
- 透明通道模块：透明通道是将 IP 数据报文解析后直接发送到串行口的一种技术。SDK

提供 485 和 232 串口类型。具体流程详见透明串口通道流程

1.4.2 各个模块涉及的接口函数

A. 初始化

SDK 初始化	H264_DVR_Init ()

B. SDK 功能信息获取

设置消息回调	H264_DVR_SetDVRMessCallBack ()

C. 注册登录设备

登入设备	H264_DVR_Login ()
报警消息推送通道	H264_DVR_SetupAlarmChan ()

D. 实时预览

打开监视通道	H264_DVR_RealPlay ()
	H264_DVR_StopRealPlay ()
监视数据回调保存	H264_DVR_SetRealDataCallBack ()

E. 设备参数配置与控制

参数配置	H264_DVR_GetDevConfig()
	H264_DVR_SetDevConfig()
查询日志	H264_DVR_FindDVRLog()
云台控制	H264_DVR_PTZControl ()
透明串口控制	H264_DVR_OpenTransComChannel ()
	H264_DVR_CloseTransComChannel ()

F. 回放/下载通道

查询录像	H264_DVR_FindFile ()
	H264_DVR_FindFileByTime()
回放及控制	H264_DVR_PlayBackByName()
	H264_DVR_PlayBackByTime()
	H264_DVR_PlayBackControl()
	H264_DVR_StopPlayBack()

下载	H264_DVR_GetFileByName () H264_DVR_GetFileByTime() H264_DVR_GetDownloadPos() H264_DVR_StopGetFile ()

G. 远程控制

远程升级	H264_DVR_Upgrade() H264_DVR_GetUpgradeState() H264_DVR_CloseUpgradeHandle()
重启/清除日志	H264_DVR_ControlDVR ()

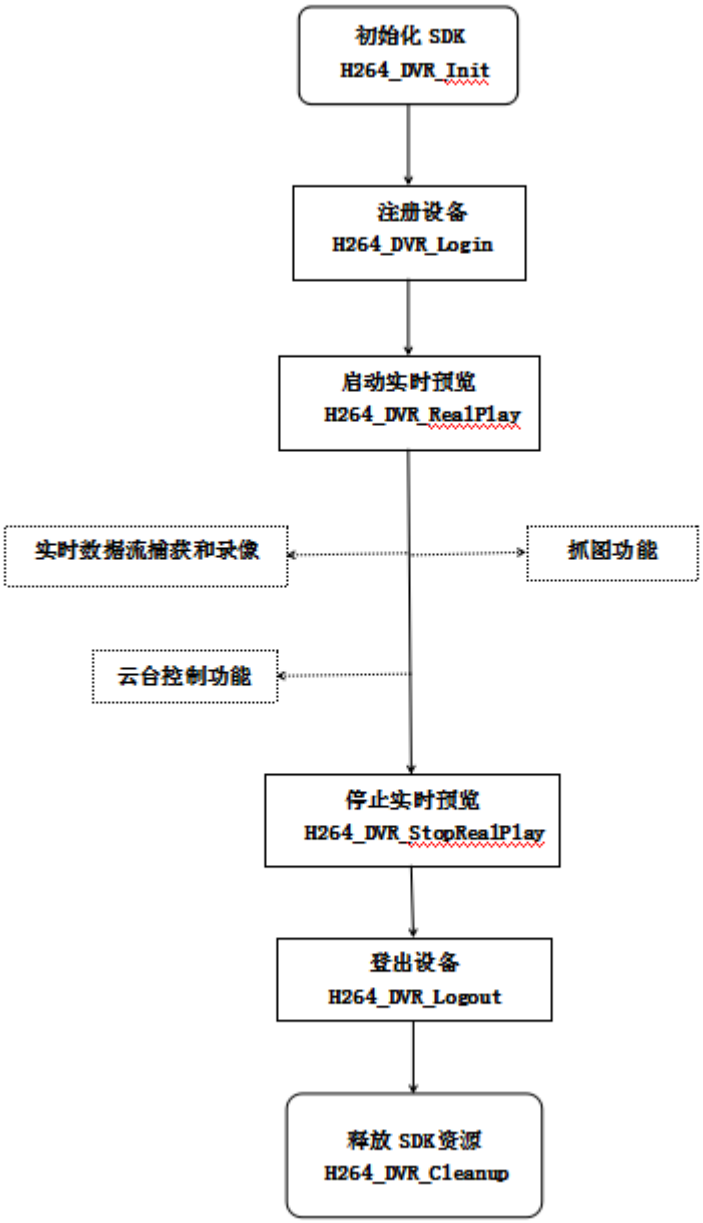
H. 注销断开设备

停止报警消息订阅	H264_DVR_CloseAlarmChan ()
登出设备	H264_DVR_Logout ()

I. 释放 SDK 资源

SDK 退出	H264_DVR_Cleanup ()

1.4.3 实时预览模块流程



图中虚线框部分的模块是与预览模块相关，必须在启动预览后才能调用，这些模块之间是并列的关系，各自完成相应的功能。

相关功能：

- 实时数据的捕获主要实现实时回调和本地录像功能。相关的接口有：

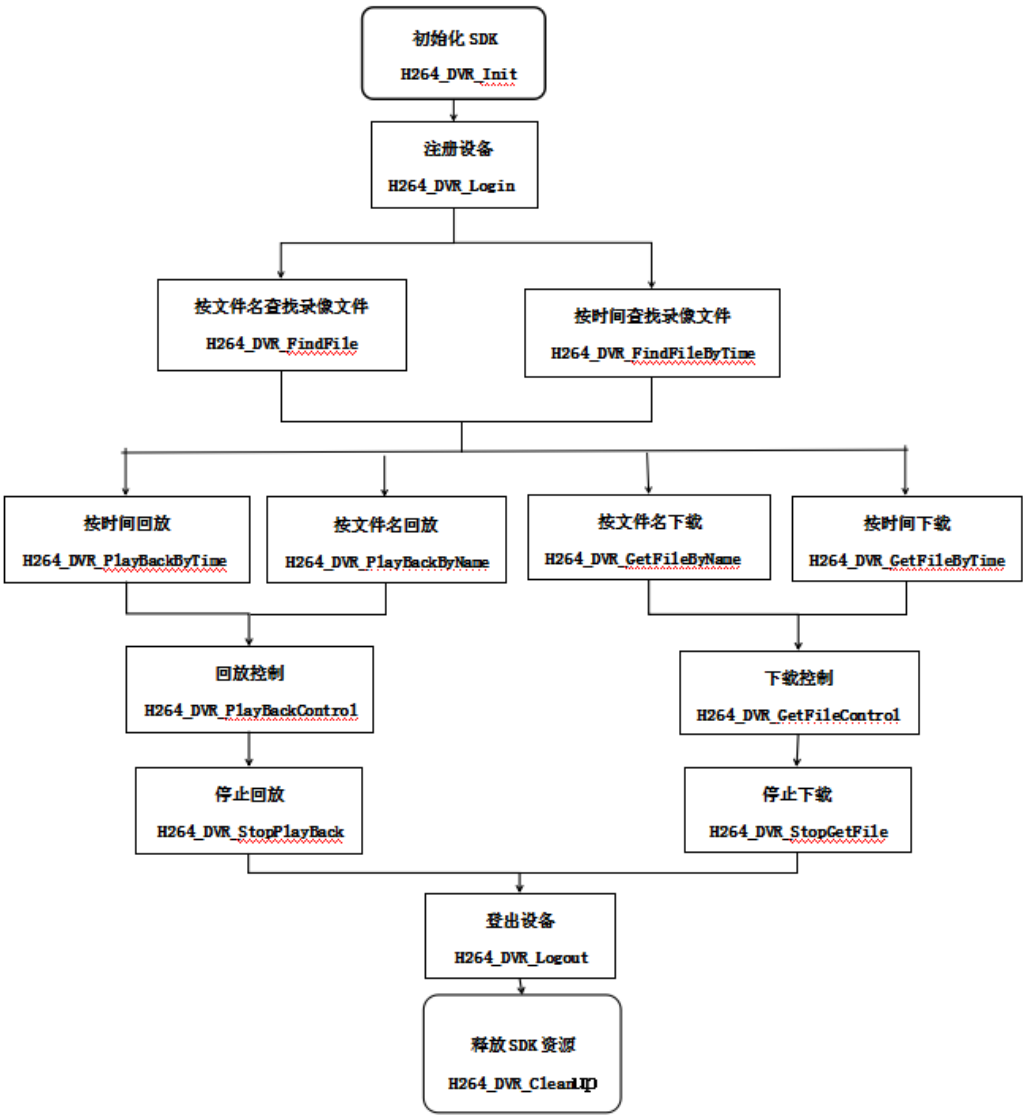
H264_DVR_SetRealDataCallBack、H264_DVR_DelRealDataCallBack、H264_DVR_StartDVRRecord、H264_DVR_StopDVRRecord 等。

- 抓图功能主要实现对当前解码图像的捕获。相关的接口有：

  H264_DVR_CatchPic（注：这个抓图接口需要设备的录像配置中支持抓图才可以实现）或者可以调用 PlaySDK 的 H264_Play_CatchPic 抓图接口（注：只适用于预览的时候）。

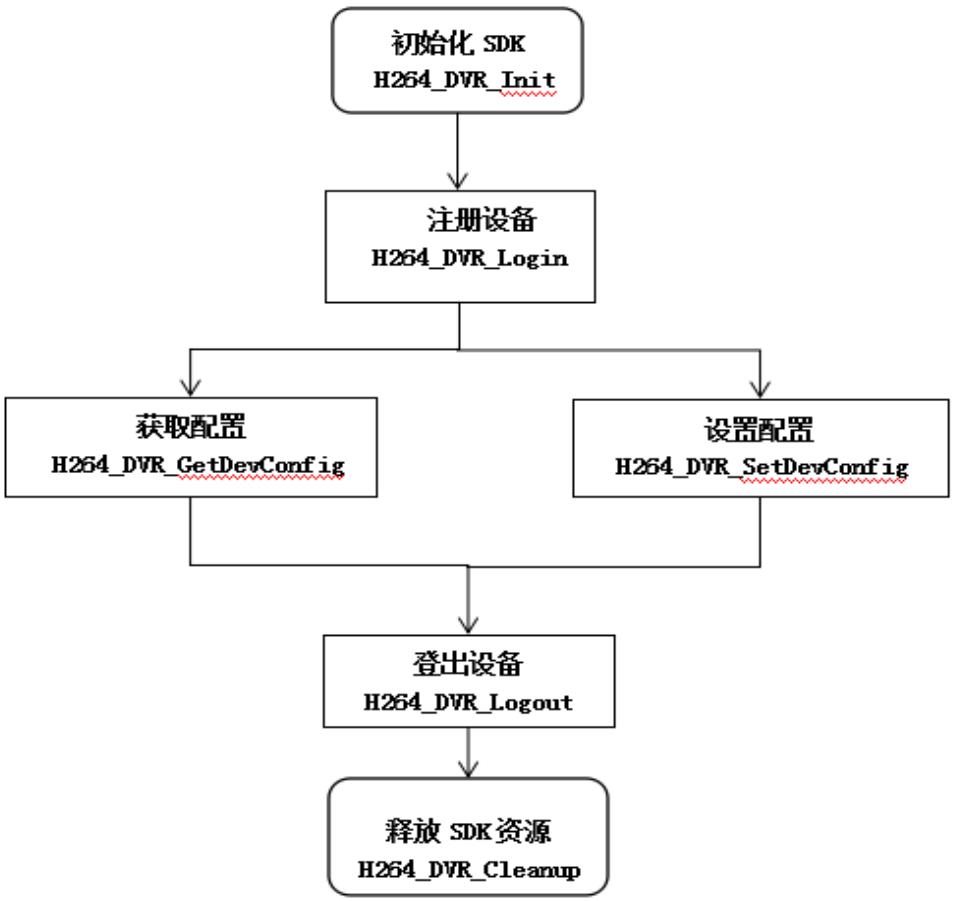
- 云台控制模块主要是在开启预览的前提下实现对云台控制的操作功能，相关的接口有：H264_DVR_PTZControl 等。

1.4.4 回放和下载模块流程



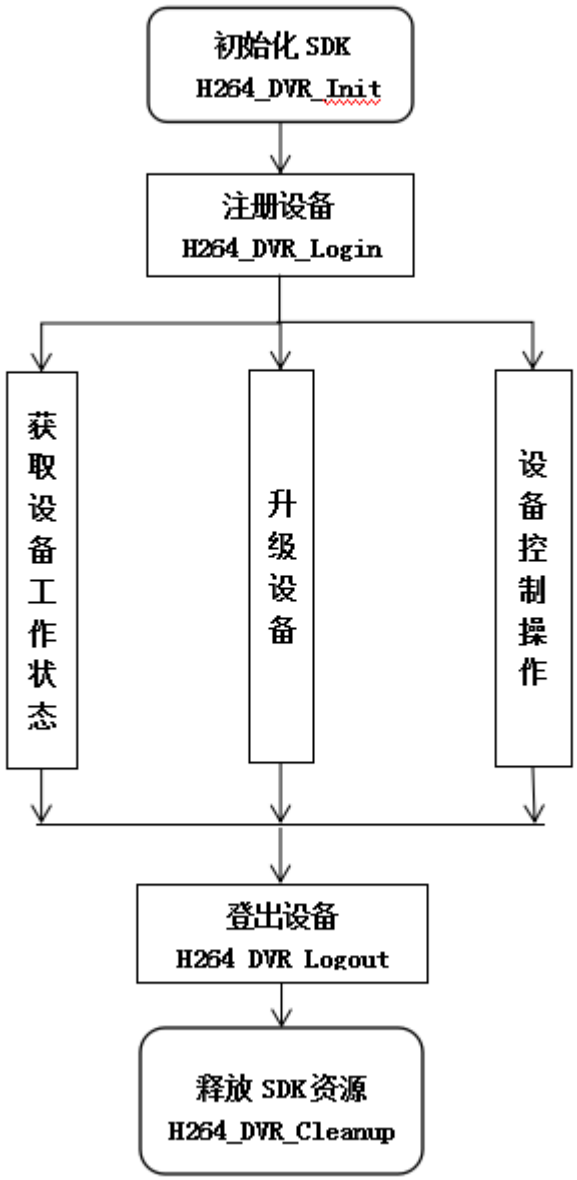
- 按文件回放或下载需要通过查找录像文件功能先获取文件信息（相关接口 H264_DVR_FindFile、H264_DVR_FindFileByTime），然后根据获取到的文件名开始回放或下载（相关接口 H264_DVR_PlayBackByName、H264_DVR_GetFileByName），在调用了回放或下载的接口后，需要调用控制接口（H264_DVR_PlayBackControl、H264_DVR_GetFileControl）。
- 按时间回放或下载文件时，用户可以无需调用查找录像文件的相关接口，只要在接口中指定开始和结束时间，调用回放或下载接口（相关接口 H264_DVR_PlayBackByTime、H264_DVR_GetFileByTime），需要进行其他操作时调用控制接口（H264_DVR_PlayBackControl、H264_DVR_GetFileControl）。

1.4.5 参数配置模块流程



- 实现参数配置首先必须做好初始化 SDK 和用户注册这两个步骤，将用户注册接口返回的 ID 号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（H264_DVR_GetDevConfig）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（H264_DVR_SetDevConfig），返回成功即设置成功。

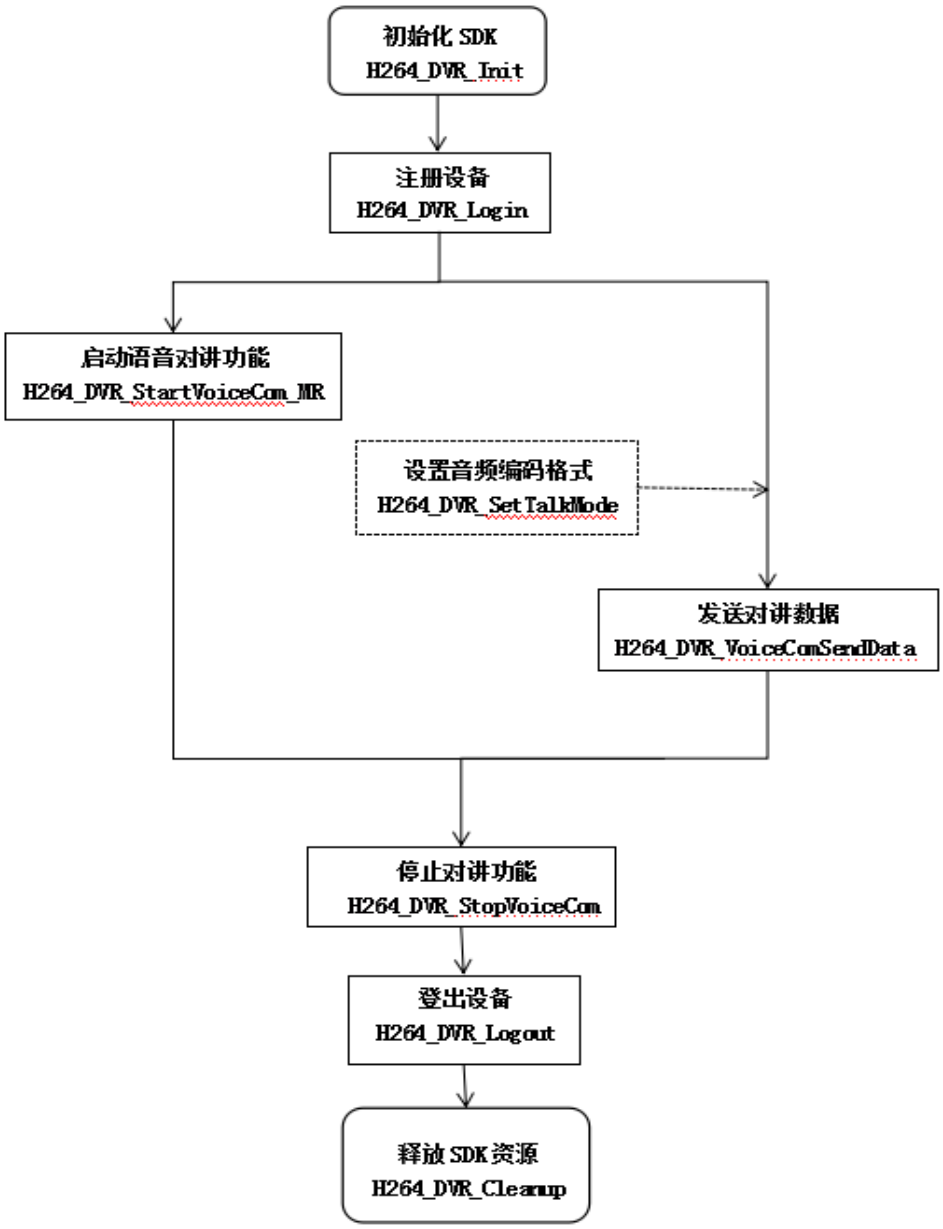
1.4.6 设备维护模块流程



远程设备维护模块包括获取设备工作状态、远程和本地升级、对设备进行重启控制等功能。

- 获取设备工作状态：可以获取到设备当前的通道状态、报警输入和输出口状态等信息。相关接口有：H264_DVR_GetDVRWorkState 等。
- 设备本地和远程升级：对设备进行升级，并且可以获取当前升级的进度和状态。相关接口有： H264_DVR_Upgrade、H264_DVR_Upgrade_Cloud 等。
- 设备控制操作，可以实现重启设备、清除日志、关机等功能。相关的接口有：H264_DVR_ControlDVR 等。

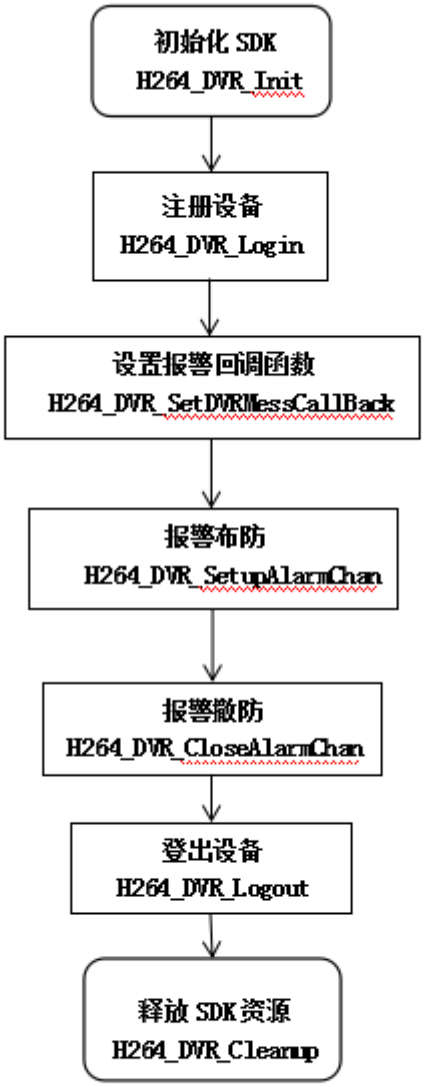
1.4.7 语音对讲模块



- 语音对讲功能实现 PC 机与设备间音频的发送和接收。在成功注册设备后调用 H264_DVR_StartVoiceCom_MR 接口完成,同时在该接口中用户可以通过设置回调函数获取当前设备发送的数据（按需要选择回调编码后数据）。
- 发送对讲数据到设备，编码格式可以进行设置，默认为 G711A 编码。相关的接口有 H264_DVR_SetTalkMode、H264_DVR_VoiceComSendData。

1.4.8 报警模块流程

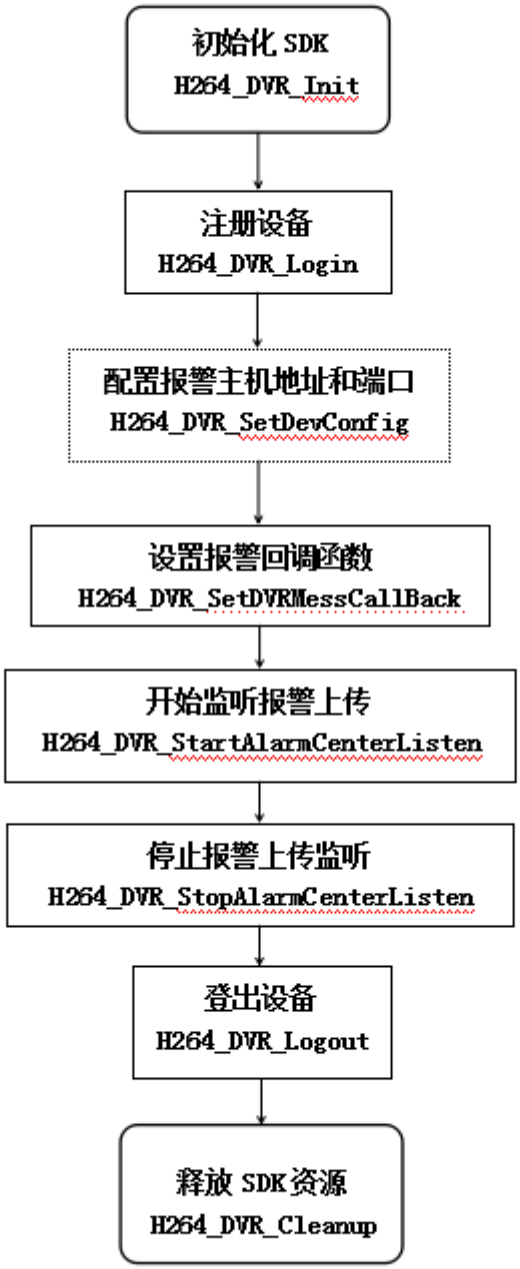
1.4.8.1 布防式报警



- 报警方式：SDK 主动连接设备，并发起报警上传命令，设备发生报警立即发送给 SDK。
- 由“报警(布防)的流程图”中看出,“布防”方式需要先进行用户注册(H264_DVR_Login)。接下来就是设置报警回调函数（ H264_DVR_SetDVRMessCallBack ），调用成功后还需要设置布防（ H264_DVR_SetupAlarmChan)。整个报警上传过程结束后还需要调

用撤防接口等操作。

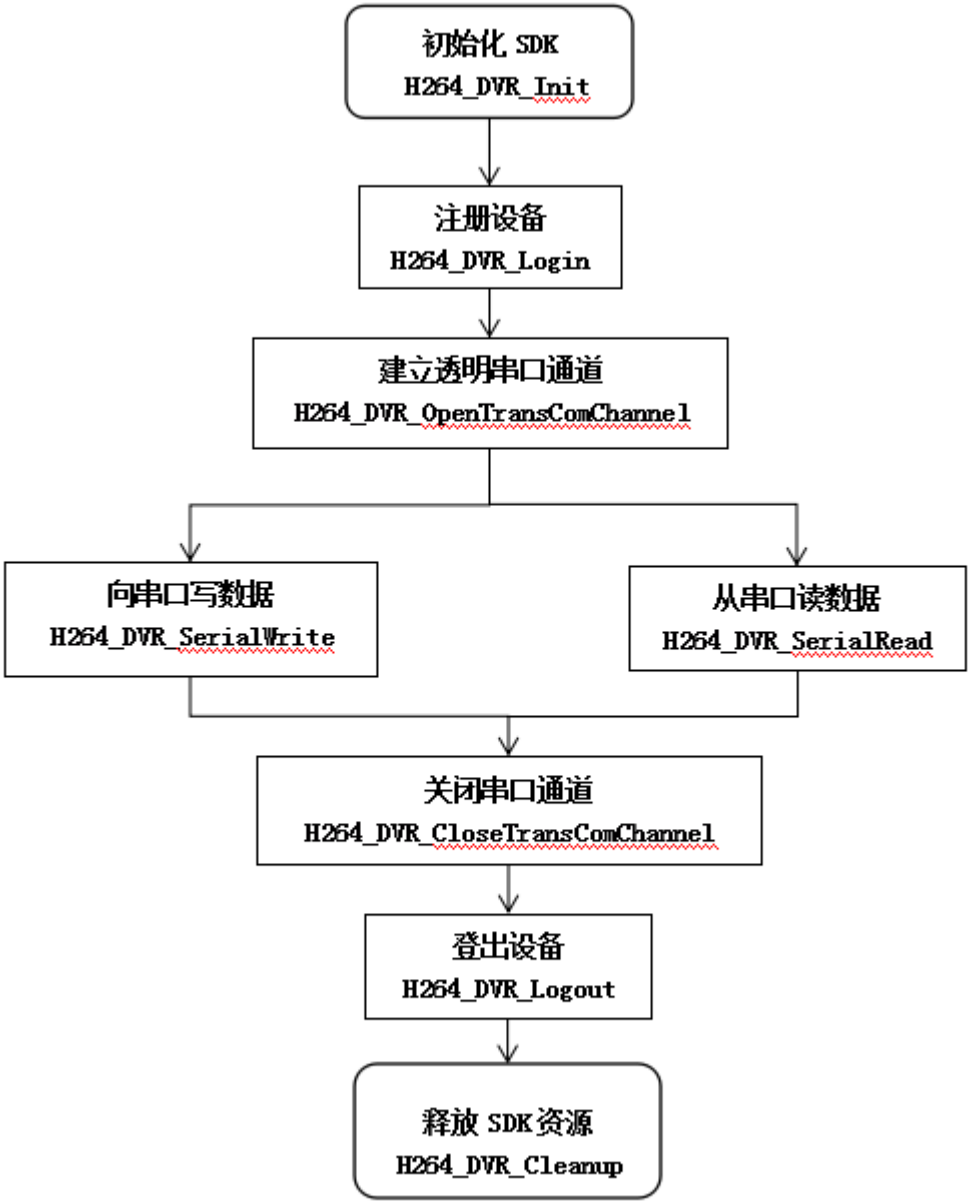
1.4.8.2 监听式报警



- 报警方式：SDK 不主动发起连接设备，只是在设定的端口上监听接收设备主动上传的报警信息。
- 这个过程需要远程配置设备的报警主机地址（即 PC 机地址）和报警主机端口（即 PC 的监听端口），报警主机就在该端口上监听接收设备主动上传的报警信息。如果报警

主机地址和报警主机端口已配置完成，那么“报警（监听）的流程图”中虚线框“用户注册”和“配置报警主机地址和端口”部分就可以省略，但事先没有配置，就必须调用参数配置接口（H264_DVR_SetDevConfig）对设备的网络参数进行配置。对以上需要配置的参数都设置完后，调用 H264_DVR_StartAlarmCenterListen 函数，开启 SDK 的监听端口，准备接收设备上传的报警信息。

1.4.9 透明串口通道模块



- SDK 提供将 485 和 232 串口类型。调用 H264_DVR_OpenTransComChannel 建立透明通道，H264_DVR_SerialWrite 向串口写数据和 H264_DVR_SerialRead 从串口读数据。整个过程结束还需要断开透明通道（ H264_DVR_CloseTransComChannel）等操作。

2 数据结构定义

2.1 系统时间结构 SDK_SYSTEM_TIME

```
typedef struct SDK_SYSTEM_TIME
{
    int  year;///< 年。
    int  month;///< 月，January = 1, February = 2, and so on.
    int  day;///< 日。
    int  wday;///< 星期，Sunday = 0, Monday = 1, and so on
    int  hour;///< 时。
    int  minute;///< 分。
    int  second;///< 秒。
    int  isdst;///< 夏令时标识。
  }SDK_SYSTEM_TIME;
```

2.2 录像文件相关结构体

2.2.1 查询条件结构体 H264_DVR_FINDINFO

```
typedef struct
{
    int nChannelNO;          //通道号
    int nFileType;           //文件类型, 见SDK_File_Type
    H264_DVR_TIME startTime; //开始时间
    H264_DVR_TIME endTime;   //结束时间
    char szCard[32];         //卡号
    void *hWnd;              //输出窗口句柄（为NULL时候，网络数据与解码播放分开
处理）
}H264_DVR_FINDINFO;
```

2.2.2 返回录像信息结构体 H264_DVR_FILE_DATA

```
typedef struct
{
    int ch;                          //通道号
    int size;                        //文件大小
    char sFileName[108];             ///< 文件名
```

```
    SDK_SYSTEM_TIME stBeginTime;    ///< 文件开始时间
    SDK_SYSTEM_TIME stEndTime;      ///< 文件结束时间
    void *hWnd;                     //输出窗口句柄（为NULL时候，网络数据与解码播放分开处理）
}H264_DVR_FILE_DATA;
```

2.2.3 按时间段查询结构体 SDK_SearchByTime

```
typedef struct SDK_SearchByTime
{
    int nHighChannel;           ///< 33~64录像通道号掩码
    int nLowChannel;            ///< 1~32录像通道号掩码
    int nFileType;              ///< 文件类型, 见SDK_File_Type
    SDK_SYSTEM_TIME stBeginTime;///< 查询开始时间
    SDK_SYSTEM_TIME stEndTime;  ///< 查询结束时间
    int    iSync;               ///< 是否需要同步
    unsigned int  nHighStreamType;///< 33~64录像的码流类型,二进制位为代表主码流，代表辅码流
    unsigned int nLowStreamType;///< 1~32录像的码流类型,二进制位为代表主码流，代表辅码流
    }SDK_SearchByTime;
```

2.2.4 查询结果相关相关结构体 SDK_SearchByTimeResult

```
//每个通道的录像信息
typedef struct SDK_SearchByTimeInfo
{
    int iChannel;           ///< 录像通道号
    ///< 录像记录用个字节的位来表示一天中的分钟
    ///< 0000:无录像0001:F_COMMON 0002:F_ALERT 0003:F_DYNAMIC 0004:F_CARD 0005:F_HAND
    unsigned char cRecordBitMap[720];
}SDK_SearchByTimeInfo;

typedef struct SDK_SearchByTimeResult
{
    int nInfoNum;                                   ///< 通道的录像记录信息个数
    SDK_SearchByTimeInfo ByTimeInfo[NET_MAX_CHANNUM];    ///< 通道的录像记录信息
}SDK_SearchByTimeResult;
```

2.3 配置信息结构_SDK_CONFIG_TYPE

H264_DVR_GetDevConfig、H264_DVR_SetDevConfig 的命令定义

dwCommand定义	功能	对应的结构体
E_SDK_CONFIG_USER	用户信息，包含了权限列表，用户列表和组列表	USER_MANAGE_INFO
E_SDK_CONFIG_ADD_USER	增加用户	USER_INFO
E_SDK_CONFIG_MODIFY_USER	修改用户	CONF_MODIFYUSER
E_SDK_CONFIG_DELETE_USER	删除用户	USER_INFO
E_SDK_CONFIG_ADD_GROUP	增加组	USER_GROUP_INFO
E_SDK_CONFIG_MODIFY_GROUP	修改组	CONF_MODIFYGROUP
E_SDK_COFIG_DELETE_GROUP	删除组	USER_GROUP_INFO
E_SDK_CONFIG_MODIFY_PSW	修改密码	CONF_MODIFY_PSW
E_SDK_CONFIG_ABILITY_SYSFUNC	支持的网络功能	SDK_SystemFunction
E_SDK_CONFIG_ABILTY_ENCODE	首先获得编码能力	CONFIG_EncodeAbility
E_SDK_CONFIG_ABILITY_PTZPRO	云台协议	SDK_PTZPROTOCOLFUNC
E_SDK_COMFIG_ABILITY_COMMPRO	串口协议	SDK_COMMFUNC
E_SDK_CONFIG_ABILITY_MOTION_FUNC	动态检测快	SDK_MotionDetectFunction
E_SDK_CONFIG_ABILITY_BLIND_FUNC	视频遮挡块	SDK_BlindDetectFunction
E_SDK_CONFIG_ABILITY_DDNS_SERVER	DDNS 服务支持类型	SDK_DDNSServiceFunction
E_SDK_CONFIG_ABILITY_TALK	对讲编码类型	SDK_DDNSServiceFunction
E_SDK_CONFIG_SYSINFO	系统信息	H264_DVR_DEVICEINFO
E_SDK_CONFIG_SYSNORMAL	普通配置	SDK_CONFIG_NORMAL
E_SDK_CONFIG_SYSENCODE	编码配置	SDK_EncodeConfigAll

E_SDK_CONFIG_SYSNET	网络配置	SDK_CONFIG_NET_COMMON
E_SDK_CONFIG_PTZ	云台页面	SDK_STR_PTZCONFIG_ALL
E_SDK_CONFIG_COMM	串口页面	SDK_CommConfigAll
E_SDK_CONFIG_RECORD	录像设置界面	SDK_RECORDCONFIG
E_SDK_CONFIG_MOTION	动态检测页面	SDK_MOTIONCONFIG
E_SDK_CONFIG_SHELTER	视频遮挡	SDK_BLINDDETECTCONFIG
E_SDK_CONFIG_VIDEO_LOSS	视频丢失	SDK_VIDEOLOSSCONFIG
E_SDK_CONFIG_ALARM_IN	报警输入	SDK_ALARM_INPUTCONFIG
E_SDK_CONFIG_ALARM_OUT	报警输出	SDK_AlarmOutConfigAll
E_SDK_CONFIG_DISK_MANAGER	硬盘管理界面	SDK_StorageDeviceControl
E_SDK_CONFIG_OUT_MODE	输出模式界面	SDK_VideoWidgetConfigAll
E_SDK_CONFIG_CHANNEL_NAME	通道名称	SDK_ChannelNameConfigAll
E_SDK_CONFIG_AUTO	自动维护界面配置	SDK_AutoMaintainConfig
E_SDK_CONFIG_DEFAULT	恢复默认界面配置	SDK_SetDefaultConfigTypes
E_SDK_CONFIG_DISK_INFO	硬盘信息	SDK_StorageDeviceInformationAll
E_SDK_CONFIG_LOG_INFO	查询日志	SDK_LogList
E_SDK_CONFIG_NET_IPFILTER	黑白名单配置	SDK_NetIPFilterConfig
E_SDK_CONFIG_NET_DHCP	DHCP 配置	SDK_NetDHCPConfigAll
E_SDK_CONFIG_NET_DDNS	DDNS 信息	SDK_NetDDNSConfigALL
E_SDK_CONFIG_NET_EMAIL	EMAIL	SDK_NetEmailConfig
E_SDK_CONFIG_NET_MULTICAST	组播	SDK_NetMultiCastConfig
E_SDK_CONFIG_NET_NTP	NTP	SDK_NetNTPConfig
E_SDK_CONFIG_NET_PPPOE	PPOE	SDK_NetPPPoEConfig
E_SDK_CONFIG_NET_DNS	DNS	SDK_NetDNSConfig
E_SDK_CONFIG_NET_FTPSERVER	FTP	SDK_FtpServerConfig
E_SDK_CONFIG_SYS_TIME	系统时间	SDK_SYSTEM_TIME
E_SDK_CONFIG_ABILITY_LANG	支持语言	SDK_MultiLangFunction

E_SDK_CONFIG_COMBINEENCODE	组合编码	SDK_CombineEncodeConfigAll
E_SDK_CONFIG_COMBINEENCODEMODE	组合编码模式	SDK_CombEncodeModeAll
E_SDK_WORK_STATE	运行状态	SDK_DVR_WORKSTATE
E_SDK_ABILITY_LANGLIST	实际支持的语言集	SDK_MultiLangFunction
E_SDK_CONFIG_NET_ARSP	ARSP	SDK_NetARSPConfigAll
E_SDK_CONFIG_SNAP_STORAGE	抓图设置	SDK_SnapshotConfig
E_SDK_CONFIG_NET_3G	3G 拨号	SDK_Net3GConfig
E_SDK_CONFIG_NET_MOBILE	手机监控	SDK_NetMoblieConfig
E_SDK_CONFIG_UPGRADEINFO	获取升级信息	SDK_UpgradeInfo
E_SDK_ABILITY_VSTD	实际支持的视频制式	SDK_MultiVstd
E_SDK_CONFIG_NET_UPNP	UPUN 设置	SDK_NetUPNPConfig
E_SDK_CONFIG_NET_WIFI	WIFI	SDK_NetWifiConfig
E_SDK_CONFIG_NET_WIFI_AP_LIST	搜索到的 WIFI 列表	SDK_NetWifiDeviceAll
E_SDK_CONFIG_SYSENCODE_SIMPLIIFY	简化的编码配置	SDK_EncodeConfigAll_SIMPLIIFY
E_SDK_CONFIG_ALARM_CENTER	告警中心	SDK_NetAlarmServerConfigAll
E_SDK_CONFIG_NET_ALARM	网络告警	SDK_NETALARMCONFIG_ALL
E_SDK_CONFIG_NET_PHONEMSG	短信	SDK_NetShortMsgCfg
E_SDK_CONFIG_NET_PHONEMEDIAMSG	彩信	SDK_NetMultimediaMsgCfg
E_SDK_CONFIG_NET_RTSP	RTSP	SDK_NetRTSPConfig
E_SDK_CONFIG_COMM485	串口 485 协议配置	SDK_STR_RS485CONFIG_ALL
E_SDK_COMFIG_ABILITY_COMMPRO485	串口 485 协议	SDK_COMMFUNC
E_SDK_CONFIG_SYS_TIME_NORTC	设置系统时间 noRTC	SDK_SYSTEM_TIME
E_SDK_CONFIG_CHANNELTILE_DOT	修改 IPC 通道名需要的点阵信息	SDK_TitleDot
E_SDK_CONFIG_CAMERA	摄像机参数	SDK_CameraParam
E_SDK_CONFIG_ABILITY_C	摄像机能力级	SDK_CameraAbility

AMERA		
E_SDK_CONFIG_STORAGENOTEXIST	硬盘不存在	SDK_VIDEOLOSSCONFIG
E_SDK_CONFIG_STORAGELOWSPACE	硬盘容量不足	SDK_StorageLowSpaceConfig
E_SDK_CONFIG_STORAGEFAILURE	硬盘出错	SDK_StorageFailConfig
E_SDK_CFG_NETIPCONFLICT	IP 冲突	SDK_VIDEOLOSSCONFIG
E_SDK_CFG_NETABORT	网络异常	SDK_VIDEOLOSSCONFIG
E_SDK_CONFIG_CHNSTATUS	通道状态	SDK_NetDecorderChnStatusAll
E_SDK_CONFIG_CHNMODE	通道模式	SDK_NetDecorderChnModeConfig
E_SDK_CONFIG_NET_DAS	主动注册	SDK_DASSerInfo
E_SDK_CONFIG_CAR_INPUT_EXCHANGE	外部信息输入与车辆状态的对应关系	SDK_CarStatusExchangeAll
E_SDK_CONFIG_DELAY_TIME	车载系统延时配置	SDK_CarDelayTimeConfig
E_SDK_CONFIG_NET_ORDER	网络优先级	SDK_NetOrderConfig
E_SDK_CONFIG_ABILITY_NETORDER	网络优先级设置能力	SDK_NetOrderFunction
E_SDK_CONFIG_GPS_TIMING	GPS 校时相关配置	SDK_GPSTimingConfig
E_SDK_CONFIG_VIDEO_ANALYZE	视频分析	SDK_ANALYSECONFIG
E_SDK_CONFIG_NAT_STATUS_INFO	Nat 状态信息	SDK_NatStatusInfo
E_SDK_CONFIG_MEDIA_WATERMARK	水印设置	SDK_WaterMarkConfigAll
E_SDK_CONFIG_ENCODE_STATICPARAM	编码器静态参数	SDK_EncodeStaticParamAll
E_SDK_CONFIG_DIGMANAGER_SHOW	通道管理显示配置	SDK_DigManagerShowStatus
E_SDK_CONFIG_ABILITY_ANALYZEABILITY	智能分析能力	SDK_ANALYZEABILITY
E_SDK_CONFIG_NAT	NAT 功能，MTU 值配置	SDK_NatConfig
E_SDK_CONFIG_CPCINFO	智能 CPC 计数数据信息	SDK_CPCDataAll
E_SDK_CONFIG_STORAGE_POSITION	录像存储设备类型	SDK_RecordStorageType

E_SDK_CONFIG_ABILITY_CARSTATUSNUM	车辆状态数	SDK_CarStatusNum
E_SDK_CFG_VPN	VPN	SDK_VPNConfig
E_SDK_CFG_VIDEOOUT	VGA 视频分辨率	SDK_VGAresolution
E_SDK_CFG_ABILITY_VGARESOLUTION	支持的 VGA 视频分辨率列表	SDK_VGAResolutionAbility
E_SDK_CFG_NET_LOCALSEARCH	搜索设备,设备端的局域网设备	SDK_NetDevList
E_SDK_CFG_ENCODE_STATICPARAM_V2	DVR 编码器静态参数	SDK_EncodeStaticParamV2
E_SDK_ABILITY_ENC_STATICPARAM	静态编码能力集	SDK_EncStaticParamAbility
E_SDK_CFG_MAIL_TEST	邮件测试	SDK_NetEmailConfig
E_SDK_CFG_SPVMN_PLATFORM	28181 协议配置	SDK_ASB_NET_VSP_CONFIG
E_SDK_CFG_PMS	手机服务	SDK_PMSConfig
E_SDK_CFG_OSD_INFO	屏幕提示信息	SDK_OSDInfoConfigAll
E_SDK_CFG_DIGITAL_REAL	真正支持的通道模式	SDK_VideoChannelManage
E_SDK_ABILITY_PTZCONTROL	PTZ 控制能力级	SDK_PTZControlAbility
E_SDK_CFG_PARAM_EX	摄像头扩展参数	SDK_CameraParamEx
E_SDK_GPS_STATUS	GPS 连接信息	SDK_GPSStatusInfo
E_SDK_WIFI_STATUS	WIFI 连接信息	SDK_WifiStatusInfo
E_SDK_3G_STATUS	3G 连接信息	SDK_WirelessStatusInfo
E_SDK_DAS_STATUS	主动注册状态	SDK_DASStatusInfo
E_SDK_ABILITY_DECODE_DELEY	解码策略能力	SDK_DecodeDeleyTimePrame
E_SDK_CFG_DECODE_PARAM	解码最大延时	SDK_DecodeParam
E_SDK_ABILITY_ONVIF_SUB_PROTOCOL	onvif 子协议	SDK_AbilityMask
E_SDK_CFG_CAR_BOOT_TYPE	车载开关机模式	SDK_CarBootTypeConfig
E_SDK_CFG_IPC_ALARM	IPC 网络报警	SDK_IPCAlarmConfigAll
E_SDK_CFG_TIME_ZONE	时区配置	SDK_TimeZone
E_SDK_ABILITY_MAX_PRE_RECORD	最大可设置预录时间~30	SDK_AbilityMask
E_SDK_CFG_DIG_TIME_SYN	数字通道时间同步配置	SDK_TimeSynParam
E_SDK_CFG_DIGITAL_ENCODE	数字通道精简版编码配置	SDK_EncodeConfigAll_SIMPLIFY

E_SDK_CFG_DIGITAL_ABILITY	数字通道的编码能力	SDK_DigitDevInfo
E_SDK_CFG_ENCODECH_DISPLAY	IE 端编码配置显示的前端通道号	SDK_EncodeChDisplay
E_SDK_CFG_RESUME_PTZ_STATE	开机云台状态	SDK_ResumePtzState
E_SDK_ABILITY_AHD_ENCODE_L	AHDL 能力级	SDK_AHDEncodeLMask
E_SDK_CFG_SPEEDALARM	速度报警	SDK_SpeedAlarmConfigAll
E_SDK_CFG_CORRESPONDENT_INFO	用户自定义配置	SDK_CorrespondentOwnInfo
E_SDK_SET_OSDINFO	OSD 信息设置(此项功能只支持模拟通道)	SDK_OSDInfo
E_SDK_SET_OSDINFO_V2	OSD 信息叠加，不保存配置（即断电重启后无叠加效果）注：IPC 设备以点阵的方式传字符	SDK_OSDInfoConfigAll
E_SDK_ABILITY_SUPPORT_EXTSTREAM	支持辅码流录像	SDK_AbilityMask
E_SDK_CFG_EXT_RECORD	辅码流配置	SDK_RECORDCONFIG_ALL/ SDK_RECORDCONFIG
E_SDK_CFG_UPGRADE_VERSION_LIST	云升级文件列表	SDK_CloudUpgradeList
E_SDK_OPERATION_SET_LOGO	视频上叠加厂家的 LOGO	SDK_SetLogo
E_SDK_OPEARTION_SPLIT_CONTROL	画面分割模式	SDK_SplitControl
E_SDK_OPERATION_UTC_TIME_SETTING	设置 UTC 时间	SDK_SYSTEM_TIME
E_SDK_CFG_ENCODE_SmartH264	SmartH264+配置	SDK_SmartH264ParamAll
E_SDK_CFG_WIFI_INFO	无线 WIFI 信息	SDK_WifiInfo
E_SDK_CFG_NET_RTMP	RTMP 协议	SDK_NetRTMPConfig
E_SDK_CFG_SNAP_SCHEDULE	定时抓图配置	SDK_SnapConfigAll
E_SDK_CFG_PTZPRESET	预置点配置	SDK_PtzPreset
E_SDK_CFG_PTZTOUR	巡航配置	SDK_PtzTour
E_SDK_CFG_PWD_SAFETY	安全问题相关配置(用于重置密码)	SDK_PasswordSafety
E_SDK_ABILITY_QUESTION_DELIVERY	获取密码找回问题	SDK_QuestionDelivery

2.4 磁盘存储控制相关结构体

2.4.1 存储设备控制类型 SDK_StorageDeviceControlTypes

```
enum SDK_StorageDeviceControlTypes
{
    SDK_STORAGE_DEVICE_CONTROL_SETTYPE,        ///< 设置类型
    SDK_STORAGE_DEVICE_CONTROL_RECOVER,        ///< 恢复错误
    SDK_STORAGE_DEVICE_CONTROL_PARTITIONS,     ///< 分区操作
    SDK_STORAGE_DEVICE_CONTROL_CLEAR,          ///< 清除操作

   SDK_STORAGE_DEVICE_CONTROL_ADDNAS,          ///<添加NAS
   SDK_STORAGE_DEVICE_CONTROL_CHANGENAS,       ///修改NAS
   SDK_STORAGE_DEVICE_CONTROL_DELNAS,          ///<删除NAS
   SDK_STORAGE_DEVICE_CONTROL_NR,
};
```

2.4.2 存储设备控制 SDK_StorageDeviceControl

```
typedef struct SDK_StorageDeviceControl
{
    int iAction;          ///< 见enum SDK_StorageDeviceControlTypes
    int iSerialNo;        ///< 磁盘序列号
    int iPartNo;          ///< 分区号
    int iType;            ///< enum SDK_StorageDeviceClearTypes或者
                          SDK_FileSystemDriverTypes
    int iPartSize[4/*MAX_DRIVER_PER_DISK*/]; ///< 各个分区的大小
}SDK_StorageDeviceControl;
```

2.5 帧信息相关结构体

```
enum MEDIA_PACK_TYPE
{
    FILE_HEAD =   0,            // 文件头
    VIDEO_I_FRAME = 1,          // 视频I帧
    VIDEO_B_FRAME =  2,         // 视频B帧
```

```
    VIDEO_P_FRAME = 3,          // 视频P帧
    VIDEO_BP_FRAME = 4,         // 视频BP帧
    VIDEO_BBP_FRAME  = 5,       // 视频B帧B帧P帧
    VIDEO_J_FRAME = 6,          // 图片帧
    AUDIO_PACKET = 10,          // 音频包
};

enum SDK_ENCODE_TYPE
{
    SDK_StreamTypeEmpty = 0,
    SDK_StreamTypeH264 = 2,
    SDK_StreamTypeJpeg = 3,
    SDK_StreamTypeGeneral = 4,
    SDK_StreamTypeH265 = 5,
    SDK_StreamTypePCM8 = 7,
    SDK_StreamTypeStd = 8
};

typedef struct
{
    int             nPacketType;      //包类型,见MEDIA_PACK_TYPE
    char*         pPacketBuffer;      //缓存区地址
    unsigned int   dwPacketSize;      //包的大小

   unsigned int   nEncodeType;         //数据格式类型见SDK_ENCODE_TYPE


    // 绝对时标
    int             nYear;            //时标:年
    int             nMonth;           //时标:月
    int             nDay;             //时标:日
    int             nHour;            //时标:时
    int             nMinute;          //时标:分
    int             nSecond;          //时标:秒
    unsigned int   dwTimeStamp;       //相对时标低位，单位为毫秒
    unsigned int   dwTimeStampHigh;    //相对时标高位，单位为毫秒
    unsigned int   dwFrameNum;         //帧序号
    unsigned int   dwFrameRate;        //帧率
    unsigned short uWidth;             //图像宽度
    unsigned short uHeight;            //图像高度
    unsigned int   Reserved[6];        //保留
} PACKET_INFO_EX;
```

2.6 本地播放控制 SDK_LoalPlayAction

```
//本地播放控制
enum SDK_LoalPlayAction
{
    SDK_Local_PLAY_PAUSE,        /*<! 暂停播放*/
    SDK_Local_PLAY_CONTINUE,     /*<! 继续正常播放*/
    SDK_Local_PLAY_FAST,          /*<! 加速播放*/
    SDK_Local_PLAY_SLOW,          /*<! 减速播放*/
};
```

2.7 子连接类型 SubConnType

```
typedef enum SubConnType
{
    conn_realTimePlay=1,
    conn_talk,
    conn_playback,
    conn_push
}SubConnType;
```

2.8 连接类型 SocketStyle

```
enum SocketStyle
{
    TCPSOCKET=0,
    UDPSOCKET,
    PLUGLANSOCKET=4,          //插座局域网登陆
    PLUGOUTERSOCKET,          //插座外网登陆
    P2P_TUTKSOCKET,           //TUTK P2P
    SOCKETNR
};
```

2.9 云升级相关结构体 SDK_CloudUpgradeVersion

```
typedef struct SDK_CloudUpgradeVersion
{
    char name[128];           // 版本名
```

```
    char date[12];          //版本日期，格式:"2014-08-26"
    unsigned int length;    // 升级文件长度
}SDK_CloudUpgradeVersion;
```

2.10 串口相关信息

2.10.1 串口类型 SERIAL_TYPE

```
typedef enum SERIAL_TYPE
{
    RS232 = 0,
    RS485 = 1,
}SERIAL_TYPE;
```

2.10.2 串口相关信息 TransComChannel

```
typedef struct __TransComChannel//透明串口
{
    SERIAL_TYPE TransComType;  //SERIAL_TYPE
    unsigned int baudrate;
    unsigned int databits;
    unsigned int stopbits;
    unsigned int parity;
} TransComChannel;
```

2.11 回放动作相关信息 SDK_PlayBackAction

```
//回放动作
enum SDK_PlayBackAction
{
    SDK_PLAY_BACK_PAUSE,          /*<! 暂停回放*/
    SDK_PLAY_BACK_CONTINUE,       /*<! 继续回放*/
    SDK_PLAY_BACK_SEEK,           /*<! 回放定位，时间s为单位*/
    SDK_PLAY_BACK_FAST,            /*<! 加速回放*/
    SDK_PLAY_BACK_SLOW,            /*<! 减速回放*/
    SDK_PLAY_BACK_SEEK_PERCENT,  /*<! 回放定位百分比*/
     SDK_PLAY_SET_TYPE,           /*<! 回放智能定位*/
};
```

2.12 主动服务回调数据相关结构体

2.12.1 设备信息 H264_DVR_DEVICEINFO

```
typedef struct _H264_DVR_DEVICEINFO
{
    char sSoftWareVersion[64]; ///< 软件版本信息
    char sHardWareVersion[64]; ///< 硬件版本信息
    char sEncryptVersion[64];  ///< 加密版本信息
    SDK_SYSTEM_TIME tmBuildTime;///< 软件创建时间
    char sSerialNumber[64];    ///< 设备序列号
    int byChanNum;             ///< 视频输入通道数
    int iVideoOutChannel;      ///< 视频输出通道数
    int byAlarmInPortNum;      ///< 报警输入通道数
    int byAlarmOutPortNum;     ///< 报警输出通道数
    int iTalkInChannel;        ///< 对讲输入通道数
    int iTalkOutChannel;    ///< 对讲输出通道数
    int iExtraChannel;         ///< 扩展通道数
    int iAudioInChannel;    ///< 音频输入通道数
    int iCombineSwitch;        ///< 组合编码通道分割模式是否支持切换
    int iDigChannel;        ///<数字通道数
    unsigned int uiDeviceRunTime;///<系统运行时间
    SDK_DeviceType deviceTye;  ///设备类型
    char sHardWare[64];        ///<设备型号
    char uUpdataTime[20];      ///<更新日期例如2013-09-03 14:15:13
    unsigned int uUpdataType;  ///<更新内容
   char sDeviceModel[16];       //设备型号(底层库从加密里获得，sHardWare针对
多个设备用同一个程序这种情况区分不了)
    int nLanguage;//国家的语言ID,0英语1中文2中文繁体3韩语4德语5葡萄牙语6俄语
    char sCloudErrCode[NET_MAX_PATH_LENGTH];//云登陆具体错误内容
    int status[32];
    //判断新过来的连接是不是通过代理转发的，如果是那么按照服务器
    //返回的限制条件来限制。
    //status[0] 路数限制:0代表不限制，n代表限制n路
    //status[1]码流限制。:不限制。限制不能观看主码流。
    //status[2]限制时间。:不限制。n:限制n分钟。
    //status[3]限制码率，目前分为四档。:不限制。:限制为CIF 6帧100K ，后续待定
    //status[4]保留位，后续扩充。
    //其中status[0]和status[1]在此处体现。
    //status[2]和status[3]在传输码流的过程中体现
}H264_DVR_DEVICEINFO,*LPH264_DVR_DEVICEINFO;
```

2.12.2 主动服务回调数据 H264_DVR_ACTIVEREG_INFO

```
typedef struct H264_DVR_ACTIVEREG_INFO
{
    char deviceSarialID[64];           //设备序列号，如果大于位则赋值
    H264_DVR_DEVICEINFO deviceInfo;    //设备信息
    char IP[IP_SIZE];                  //外网IP

}H264_DVR_ACTIVEREG_INFO;
```

2.12.3 主动注册配置相关结构体 SDK_DASSerInfo

```
typedef struct SDK_DASSerInfo
{
    bool enable;
    char serAddr[NET_NAME_PASSWORD_LEN];
    int  port;
    char userName[NET_NAME_PASSWORD_LEN];
    char passwd[NET_NAME_PASSWORD_LEN];
    char devID[NET_NAME_PASSWORD_LEN];

}SDK_DASSerInfo;
```

2.13 告警中心相关结构体

2.13.1 报警中心相关设置 SDK_NetAlarmServerConfigAll

```
//IP addr
typedef union
 {
   unsigned char     c[4];
   unsigned short    s[2];
   unsigned int      l;

}CONFIG_IPAddress;

///< 服务器结构定义
typedef struct SDK_RemoteServerConfig
{
    char ServerName[NET_NAME_PASSWORD_LEN];      ///< 服务名
    CONFIG_IPAddress ip;                   ///< IP地址
```

```
    int Port;                                   ///< 端口号
    char UserName[NET_NAME_PASSWORD_LEN];       ///< 用户名
    char Password[NET_NAME_PASSWORD_LEN];       ///< 密码
    bool Anonymity;                             ///< 是否匿名登录

}SDK_RemoteServerConfig;

//< 报警中心设置
typedef struct SDK_NetAlarmCenterConfig
{
    bool bEnable;                               ///< 是否开启
    char sAlarmServerKey[NET_NAME_PASSWORD_LEN]; ///< 报警中心协议类型名称,
    SDK_RemoteServerConfig Server;              ///< 报警中心服务器
    bool bAlarm;
    bool bLog;
}SDK_NetAlarmCenterConfig;

typedef struct SDK_NetAlarmServerConfigAll
{
    SDK_NetAlarmCenterConfig vAlarmServerConfigAll[NET_MAX_ALARMSERVER_TYPE];

}SDK_NetAlarmServerConfigAll;
```

2.13.2 报警中心消息内容 SDK_NetAlarmCenterMsg

```
//IP addr
typedef union
 {
   unsigned char     c[4];
   unsigned short    s[2];
   unsigned int      l;

}CONFIG_IPAddress;

// 告警中心消息内容
typedef struct SDK_NetAlarmCenterMsg
{
    CONFIG_IPAddress HostIP;            ///< 设备IP
    int  nChannel;                      ///< 通道
    int  nType;                         ///< 类型见AlarmCenterMsgType
    int  nStatus;                       ///< 状态见AlarmCenterStatus
    SDK_SYSTEM_TIME Time;               ///< 发生时间
    char sEvent[NET_MAX_INFO_LEN];      ///< 事件
    char sSerialID[NET_MAX_MAC_LEN];    ///< 设备序列号
```

```
    char sDescrip[NET_MAX_INFO_LEN];   ///< 描述
}SDK_NetAlarmCenterMsg;
```

2.14 DVR 工作状态相关结构体 SDK_DVR_WORKSTATE

```
// 告警状态
typedef struct SDK_DVR_ALARMSTATE
{
    char iVideoMotion[NET_MAX_MSK_SIZE];   ///< 移动侦测状态,用掩码表示通道号,byte0代表通道一,以此类推1: 有告警0: 无告警
    char iVideoBlind[NET_MAX_MSK_SIZE];        ///< 视频遮挡状态,用掩码表示通道号,byte0代表通道一,以此类推1: 有告警0: 无告警
    char iVideoLoss[NET_MAX_MSK_SIZE];     ///< 视频丢失状态,用掩码表示通道号,byte0代表通道一,以此类推1: 有告警0: 无告警
    char iAlarmIn[NET_MAX_MSK_SIZE];       ///< 告警输入状态,用掩码表示通道号,byte0代表通道一,以此类推1: 有告警0: 无告警
    char iAlarmOut[NET_MAX_MSK_SIZE];      ///< 告警输出状态,用掩码表示通道号,byte0代表通道一,以此类推1: 有告警0: 无告警
}SDK_DVR_ALARMSTATE;

// 通道状态
typedef struct SDK_DVR_CHANNELSTATE
{
    bool bRecord; ///< 是否正在录像
    int iBitrate; ///< 当前码率
}SDK_DVR_CHANNELSTATE;

// DVR工作状态
typedef struct SDK_DVR_WORKSTATE
{
    SDK_DVR_CHANNELSTATE vChnState[NET_MAX_CHANNUM];
    SDK_DVR_ALARMSTATE vAlarmState;
}SDK_DVR_WORKSTATE;
```

2.15 网络报警相关结构体 SDK_NetAlarmInfo

```
/// 网络报警
typedef struct SDK_NetAlarmInfo
{
    int iEvent;   //目前未使用
```

```
    int iState;    //每bit表示一个通道,bit0:第一通道,0-无报警1-有报警, 依次类推
}SDK_NetAlarmInfo;
```

3 接口定义

3.1 SDK 初始化

3.1.1 初始化 SDK H264_DVR_Init

函数： H264_DVR_API long H264_DVR_Init(

fDisConnect cbDisConnect,

unsigned long dwUser

);

参数： [in]cbDisConnect 断线回调函数，回调出当前网络已经断开的设备，对调用 SDK 的 H264_DVR_Logout 函数主动断开的设备不回调，设置为 NULL 时禁止回调

[in]dwUser 用户数据

```
typedef void (CALL_METHOD *fDisConnect)(
long lLoginID,
char *pchDVRIP,
long nDVRPort,
unsigned long dwUser
);
```

[out]lLoginID H264_DVR_Login 的返回值

[out]pchDVRIP 设备 IP

[out]nDVRPort 设备端口号

[out]dwUser 用户数据

返回值： 成功返回 TRUE，不成功返回 FALSE。

说明： 初始化 SDK, 在所有的 SDK 函数之前调用。

3.1.2 释放 SDK 资源 H264_DVR_Cleanup

函数： H264_DVR_API bool CALL_METHOD H264_DVR_Cleanup();

参数： 无

返回值：true表示成功，false表示失败

说明： 清空 SDK, 释放占用的资源，在所有的 SDK 函数之后调用。接口返回失败请调用 H264_DVR_GetLastError 获取错误码，通过错误码判断出错原因。

3.2 SDK 本地功能

3.2.1 设置连接超时时间和连接尝试次数 H264_DVR_SetConnectTime

```
函数： H264_DVR_API bool CALL_METHOD H264_DVR_SetConnectTime(
       long nWaitTime,
       long nTryTimes
       );
```

参数： [in]nWaitTime 超时时间（单位：ms，不设置时默认 5000ms）

[in]nTryTimes 尝试次数（单位：次数，不设置时默认 3 次）

返回值：true 表示成功。

说明： SDK 默认超时时间 5000ms，尝试次数 3 次。

3.2.2 绑定本地 IP H264_DVR_SetLocalBindAddress

```
函数： H264_DVR_API bool CALL_METHOD H264_DVR_SetLocalBindAddress(
       char* szIP
       );
```

参数： [in]szIP 需要绑定的IP地址

返回值：true表示成功，false表示失败。

说明：　设置绑定的ip地址（在多网卡的时候，可以指定绑定的ip地址）。

3.2.3 返回最后操作的错误码 H264_DVR_GetLastError

函数：　H264_DVR_API long CALL_METHOD H264_DVR_GetLastError();

参数：　无。

返回值：返回最后操作的错误码。具体详见错误码枚举。

说明：　返回值是设备返回的错误码。

3.3 用户注册

3.3.1 用户注册设备 H264_DVR_Login

```
函数：  H264_DVR_API long H264_DVR_Login (
        char *sDVRIP,
        unsigned short wDVRPort,
        char *sUserName,
        char *sPassword,
        LPH264_DVR_DEVICEINFO lpDeviceInfo,
        int *error,
        int socketType DEF_PARAM(0)
        );
```

参数：

[in]sDVRIP	设备 IP
[in]wDVRPort	设备端口
[in]sUserName	用户名
[in]sPassword	用户密码
[out]lpDeviceInfo	设备信息,属于输出参数
[out]error	返回登录错误码。

[in]socketTyle　　登入类型 参见：SocketStyle （默认为 TCPSOCKET，可以设为 NULL）

返回值：　失败返回 0，成功返回设备 ID，登录成功之后对设备的操作都可以通过此值(设备句柄)对应到相应的设备。可以调用接口 H264_DVR_GetLastError 获取具体的错误码。

说明：　注册用户到设备，当设备端把用户设置为复用（设备默认的用户如 admin,不能设置为复用），则使用该帐号可以多次向设备注册。

3.3.2 用户注销设备 H264_DVR_Logout

函数：　H264_DVR_API long CALL_METHOD H264_DVR_Logout(
long lLoginID
);

参数：　[in]lLoginID　　登录句柄

返回值：　1 表示成功，0 表示失败。

说明：

3.3.3 主动注册 H264_DVR_StartActiveRigister

函数：　H264_DVR_API bool CALL_METHOD H264_DVR_StartActiveRigister(
int nPort,
fMessCallBack cbFunc,
unsigned long dwDataUser
);

参数：　[in]nPort　　监听端口号,0<=nPort<=65535
[out]cbFunc[out]　　注册上线回调函数
[in]dwDataUser[in]　　回调函数参数
typedef bool (CALL_METHOD *fMessCallBack)(
long lLoginID,
char *pBuf,

```
unsigned long dwBufLen,
long dwUser,
int nType
);
```

[out]lLoginID	登录句柄
[out]pBuf	回调数据-- H264_DVR_ACTIVEREG_INFO
[out]dwBufLen	回调数据长度
[out]dwUser	用户参数
[out]nType	数据类型-- ALARM_TYPE

返回值： true表示成功，false表示失败。

说明： 需要调用参数配置接口（H264_DVR_SetDevConfig）对设备的网络参数进行配置。对应的配置命令E_SDK_CONFIG_NET_DAS，结构体为SDK_DASSerInfo。

3.4 实时监视

3.4.1 实时预览 H264_DVR_RealPlay

函数：

```
H264_DVR_API long H264_DVR_RealPlay(
long lLoginID,
LPH264_DVR_CLIENTINFO lpClientInfo
);
```

参数：

[in]lLoginID	H264_DVR_Login 的返回值
[in]lpClientInfo	播放句柄

返回值： 失败返回 小于等于 0，小于 0 可以用 H264_DVR_GetLastError 获得错误类型，成功返回实时监视 ID(实时监视句柄)，将作为相关函数的参数。

部分错误码分析:

(1)H264_DVR_SUB_CONNECT_ERROR = -11202:建立视频子连接失败，设备可能不在线或者可能在重启过程中。处理方法：收到断线回调后登出再登入。

(2)H264_DVR_SUB_CONNECT_SEND_ERROR = -11203:

a.局域网访问:子连接通讯失败,即子连接建立成功了,但是通讯失败了,设备在子连接建立成功后设备断开了,处理方法:再次调用 H264_DVR_RealPlay 会出现(1)的返回值。

b.主动注册访问: 主连接通讯失败,设备断线了,sdk 内部还没有收到断线回调.处理方法:等收到了断线回调再登出即可。

(3)H264_DVR_NOTVALID = -11206:非法错误, 主连接已断开,设备已断开过, 然后重启成功了,但是还没收到断线回调,这个时候还在使用之前的登录句柄。处理方法:收到断线回调后登出再登入。

说明: 根据登录时获取到的设备信息,调用本接口,就可以打开任何有效的一路实时监视,并通过 H264_DVR_SetRealDataCallBack 设备的回调得到原始数据(注:如果对 lpClientInfo 中的 hWnd 赋值就可以完成播放,而不需要对回调出来的数据送解码播放),成功返回实时监视 ID,用于以下对本监视通道的控制和操作。

3.4.2 停止预览 H264_DVR_StopRealPlay

函数:
```
H264_DVR_API bool CALL_METHOD H264_DVR_StopRealPlay(
long lRealHandle,
void*hWnd DEF_PARAM(0)
);
```

参数:		
	[in]lRealHandle	H264_DVR_RealPlay的返回值
	[in]hWnd	用于停止相应窗口的解码播放;默认值 NULL 停止所有窗口的解码播放

返回值: 1 表示成功,0 表示失败。

说明:

3.4.3 设置数据回调 H264_DVR_SetRealDataCallBack

函数:
```
H264_DVR_API bool CALL_METHOD H264_DVR_SetRealDataCallBack(
long lRealHandle,
```

```
fRealDataCallBack cbRealData,
long dwUser
);
```

参数：

[in]	lRealHandle	预览播放句柄
[out]	cbRealData	实时数据回调
[in]	dwUser	回调函数参数

```
typedef int(CALL_METHOD *fRealDataCallBack) (
long lRealHandle,
long dwDataType,
unsigned char *pBuffer,
long lbufsize,
long dwUser
);
```

[out]	lRealHandle	播放句柄
[out]	dwDataType	数据类型—暂时不需要判断
[out]	pBuffer	回调数据
[out]	lbufsize	回调数据长度
[out]	dwUser	用户回调参数

返回值：true 表示成功，false 表示失败。具体错误码可以通过 H264_DVR_GetLastError 获取。

说明：设置实时监视数据回调，给用户提供设备流出的数据。

注：除了通用的接口，还有拓展接口 H264_DVR_SetRealDataCallBack_V2 可以获取帧信息，具体可以参考 netsdk.h 中的说明以及 Clientdemo 中的运用。

3.4.4 清除回调函数 H264_DVR_DelRealDataCallBack

函数：

```
H264_DVR_API bool CALL_METHOD H264_DVR_DelRealDataCallBack(
long lRealHandle,
```

```
    fRealDataCallBack cbRealData,
    long dwUser
    );
```

参数:

	[in]lRealHandle	播放句柄
	[out]cbRealData	实时回调数据
	[in]dwUser	用户参数

```
typedef int(CALL_METHOD *fRealDataCallBack) (
long lRealHandle,
long dwDataType,
unsigned char *pBuffer,
long lbufsize,
long dwUser
);
```

	[out]lRealHandle	播放句柄
	[out]dwDataType	数据类型—暂时不需要判断
	[out]pBuffer	回调数据
	[out]lbufsize	回调数据长度
	[out]dwUser	用户回调参数

返回值: true 表示成功，false 表示失败。具体错误码可以通过 H264_DVR_GetLastError 获取。

说明: 清除回调函数,该函数需要在H264_DVR_StopRealPlay前调用。

注：对应H264_DVR_SetRealDataCallBack_V2有H264_DVR_DelRealDataCallBack_V2。

3.5 强制 I 帧 H264_DVR_MakeKeyFrame

函数:

```
H264_DVR_API bool CALL_METHOD H264_DVR_MakeKeyFrame(
    long lLoginID,
    int nChannel,
```

```
int nStream
);
```

参数：

	[in]lLoginID	登录句柄
	[in]nChannel	通道号
	[in]nStream	码流类型--0 表示主码流，为表示子码流

返回值：true 表示成功，false 表示失败。具体错误码可以通过 H264_DVR_GetLastError 获取。

说明：支持强制 I 帧。

3.6 回放和下载

录像文件的查找

3.6.1 按文件名查询录像 H264_DVR_FindFile

函数：

```
H264_DVR_API long CALL_METHOD H264_DVR_FindFile(
long lLoginID,
H264_DVR_FINDINFO* lpFindInfo,
H264_DVR_FILE_DATA *lpFileData,
int lMaxCount,
int *findcount,
int waittime DEF_PARAM(5000)
);
```

参数：

	[in]lLoginID	登录句柄
	[in]lpFindInfo	查询条件-- H264_DVR_FINDINFO
	[out]lpFileData	查询结果-- H264_DVR_FILE_DATA
	[in]lMaxCount	查询的最大录像数量
	[out]findcount	查询到的录像数量

[in]waittime　　等待时间

返回值：　1 表示成功,0 表示失败。具体错误码可以通过 H264_DVR_GetLastError 获取。

说明：　在回放之前需要先调用本接口查询录像记录,当根据输入的时间段查询到的录像记录信息大于定义的缓冲区大小,则只返回缓冲所能存放的录像记录,可以根据需要继续查询。

3.6.2 按时间查找录像文件 H264_DVR_FindFileByTime

函数：
```
H264_DVR_API long CALL_METHOD H264_DVR_FindFileByTime(
long lLoginID,
SDK_SearchByTime* lpFindInfo,
SDK_SearchByTimeResult *lpFileData,
int waittime DEF_PARAM(10000)
);
```

参数：
[in]lLoginID　　登录句柄
[in]lpFindInfo　　查询条件-- SDK_SearchByTime
[out]lpFileData　　查询结果-- SDK_SearchByTimeResult
[in] waittime　　等待时间

返回值：　1 表示成功,0 表示失败。具体错误码可以通过 H264_DVR_GetLastError 获取。

说明：　按时间查找录像文件。

回放录像文件

3.6.3 按名字回放录像 H264_DVR_PlayBackByName

函数：
```
H264_DVR_API long CALL_METHOD H264_DVR_PlayBackByName(
long lLoginID,
H264_DVR_FILE_DATA *sPlayBackFile,
```

```
fDownLoadPosCallBack cbDownLoadPos,
fRealDataCallBack fDownLoadDataCallBack,
 long dwDataUser
);
```

参数：

[in]lLoginID	登录句柄
[in]sPlayBackFile	回访的文件参数-- H264_DVR_FILE_DATA
[out]cbDownLoadPos	进度回调，用户通知用户设备是否已经将数据发送完毕，回调中的lDownLoadSize=-1代表数据发送完毕。如果想实时显示进度，应该从码流里面获取时间来计算网络部分不分析码流，如果以当前接收数据大小/总大小来计算进度的话不是很准，应该以当前时间，根据开始时间和结束时间来计算进度。
[out]fDownLoadDataCallBack	回放数据回调
[in]dwDataUser	数据回调参数

```
typedef void(CALL_METHOD *fDownLoadPosCallBack) (
long lPlayHandle,
long lTotalSize,
long lDownLoadSize,
long dwUser
);
```

[out]lPlayHandle	回放句柄
[out]lTotalSize	总的数据长度
[out]lDownLoadSize	下载的数据长度
[out]dwUser	用户回调参数

```
typedef int(CALL_METHOD *fRealDataCallBack) (
long lRealHandle,
long dwDataType,
unsigned char *pBuffer,
```

```
long lbufsize,
long dwUser
);
```

[out]lRealHandle　　播放句柄

[out]dwDataType　　数据类型—暂时不需要判断

[out]pBuffer　　回调数据

[out]lbufsize　　回调数据长度

[out]dwUser　　用户回调参数

返回值：　非 0 表示成功，0 表示失败。具体错误码可以通过 H264_DVR_GetLastError 获取。

说明：　网络回放,需要说明的是，用户登录一台设备后，每通道同一时间只能播放一则录像,不能同时播放同一通道的多条记录。

3.6.4 按时间回放录像 H264_DVR_PlayBackByTime

函数：

```
H264_DVR_API long CALL_METHOD H264_DVR_PlayBackByTime(
long lLoginID,
H264_DVR_FINDINFO* lpFindInfo,
fDownLoadPosCallBack cbDownLoadPos,
fRealDataCallBack fDownLoadDataCallBack,
long dwDataUser
);
```

参数：

[in]lLoginID　　登录句柄

[in]lpFindInfo　　查询录像条件-- H264_DVR_FINDINFO

[out]cbDownLoadPos　　进度回调，用户通知用户设备是否已经将数据发送完毕，回调中的lDownLoadSize=-1代表数据发送完毕。

[out]fDownLoadDataCallBack　回放数据回调

[in]dwDataUser　　数据回调参数

```
typedef void(CALL_METHOD *fDownLoadPosCallBack) (
long lPlayHandle,
long lTotalSize,
long lDownLoadSize,
long dwUser
);
```

[out]lPlayHandle　　回放句柄

[out]lTotalSize　　总的数据长度

[out]lDownLoadSize　　下载的数据长度

[out]dwUser　　用户回调参数

```
typedef int(CALL_METHOD *fRealDataCallBack) (
long lRealHandle,
long dwDataType,
unsigned char *pBuffer,
long lbufsize,
long dwUser
);
```

[out]lRealHandle　　播放句柄

[out]dwDataType　　数据类型—暂时不需要判断

[out]pBuffer　　回调数据

[out]lbufsize　　回调数据长度

[out]dwUser　　用户回调参数

返回值：　非 0 表示成功，0 表示失败。具体错误码可以通过 H264_DVR_GetLastError 获取。

说明：　按时间回放录像文件。

3.6.5 停止回放录像 H264_DVR_StopPlayBack

函数： H264_DVR_API bool CALL_METHOD H264_DVR_StopPlayBack(
long lPlayHandle
);

参数： [in]lPlayHandle 回放句柄

返回值： true 表示成功，false 表示失败。具体错误码可以通过 H264_DVR_GetLastError 获取。

说明： 输入上一接口返回的播放 ID,调用本接口就可以停止控制。

3.6.6 回放控制 H264_DVR_PlayBackControl

函数： H264_DVR_API bool CALL_METHOD H264_DVR_PlayBackControl(
long lPlayHandle,
long lControlCode,
long lCtrlValue,
int itype DEF_PARAM(0)
);

参数：
[in]lPlayHandle 登录句柄
[in]lControlCode 控制命令,见enum SDK_PlayBackAction
[in]lCtrlValue 控制值
[in]itype 类型，见enum SDK_PLAY_BACK_SETTYPE

返回值： true 表示成功，false 表示失败。具体错误码可以通过 H264_DVR_GetLastError 获取。

说明： 回放控制，只有是智能回放定位才有效。

下载录像文件

3.6.7 按文件名下载录像文件 H264_DVR_GetFileByName

函数：

```
H264_DVR_API long CALL_METHOD H264_DVR_GetFileByName(
long lLoginID,
H264_DVR_FILE_DATA *sPlayBackFile,
char *sSavedFileName,
fDownLoadPosCallBack cbDownLoadPos DEF_0_PARAM,
long dwDataUser DEF_0_PARAM,
fRealDataCallBack fDownLoadDataCallBack DEF_0_PARAM
);
```

参数：

[in]lLoginID	登录句柄
[in]sPlayBackFile	下载的录像信息-- H264_DVR_FILE_DATA
[out]sSavedFileName	保存的文件路径
[out]cbDownLoadPos	下载的进度回调（可以为空，通过 H264_DVR_GetDownloadPos 获取下载进度）
[in]dwDataUser	回调函数参数
[out]fDownLoadDataCallBack	数据回调

```
typedef void(CALL_METHOD *fDownLoadPosCallBack) (
long lPlayHandle,
long lTotalSize,
long lDownLoadSize,
long dwUser
);
```

[out]lPlayHandle	回放句柄
[out]lTotalSize	总的数据长度
[out]lDownLoadSize	下载的数据长度
[out]dwUser	用户回调参数

```
typedef int(CALL_METHOD *fRealDataCallBack) (
long lRealHandle,
long dwDataType,
unsigned char *pBuffer,
long lbufsize,
long dwUser
);
```

[out]lRealHandle	播放句柄
[out]dwDataType	数据类型—暂时不需要判断
[out]pBuffer	回调数据
[out]lbufsize	回调数据长度
[out]dwUser	用户回调参数

返回值： 非 0 表示成功，0 表示失败。具体错误码可以通过 H264_DVR_GetLastError 获取。

说明： 按文件下载录像文件，通过查询到的文件信息下载。根据上面查询的记录，就可以将录像保存到指定的文件,下载进度回调与回放进度类似。

3.6.8 按时间下载录像文件 H264_DVR_GetFileByTime

函数：

```
H264_DVR_API long CALL_METHOD H264_DVR_GetFileByTime(
long lLoginID,
H264_DVR_FINDINFO* lpFindInfo,
char *sSavedFileDIR,
bool bMerge DEF_PARAM(0),
fDownLoadPosCallBack cbDownLoadPos DEF_0_PARAM,
long dwDataUser DEF_0_PARAM,
fRealDataCallBack fDownLoadDataCallBack DEF_0_PARAM);
```

参数： [in]lLoginID	登录句柄

[in]lpFindInfo 录像查询条件-- H264_DVR_FINDINFO

[in]sSavedFileDIR 录像文件保存路径

[in]bMerge 文件是否合并

[out]cbDownLoadPos 下载的进度回调

[in]dwDataUser 回调函数参数

[in]fDownLoadDataCallBack 数据回调

```
typedef void(CALL_METHOD *fDownLoadPosCallBack) (
long lPlayHandle,
long lTotalSize,
long lDownLoadSize,
long dwUser
);
```

[out]lPlayHandle 回放句柄

[out]lTotalSize 总的数据长度

[out]lDownLoadSize 下载的数据长度

[out]dwUser 用户回调参数

```
typedef int(CALL_METHOD *fRealDataCallBack) (
long lRealHandle,
long dwDataType,
unsigned char *pBuffer,
long lbufsize,
long dwUser
);
```

[out]lRealHandle 播放句柄

[out]dwDataType 数据类型—暂时不需要判断

[out]pBuffer 回调数据

[out]lbufsize 回调数据长度

[out]dwUser 用户回调参数

返回值： 非 0 表示成功，0 表示失败。具体错误码可以通过 H264_DVR_GetLastError 获取。

说明： 按时间下载录像文件，通过查询到的文件信息下载。

3.6.9 停止下载录像文件 H264_DVR_StopGetFile

函数：
```
H264_DVR_API bool CALL_METHOD H264_DVR_StopGetFile(
long lFileHandle
);
```

参数：
[in]lFileHandle 下载文件句柄

返回值： true 表示成功，false 表示失败。具体错误码可以通过 H264_DVR_GetLastError 获取。

说明： 根据需要可以等文件下载完了关闭下载,也可以下载到一部分停止下载;。

3.6.10 下载控制 H264_DVR_GetFileControl

函数：
```
H264_DVR_API bool CALL_METHOD H264_DVR_GetFileControl(
long lPlayHandle,
long lControlCode,
bool bDown DEF_PARAM(1)
);
```

参数：
[in]lPlayHandle 登录句柄
[in]lControlCode 控制命令,见 enum SDK_PlayBackAction
[in]bDown 是否位下载，默认为 1

返回值： true 表示成功，false 表示失败。具体错误码可以通过 H264_DVR_GetLastError 获取。

说明： 对已经打开的播放进行暂停和恢复控制。

3.6.11 获取下载进度 H264_DVR_GetDownloadPos

函数： H264_DVR_API int CALL_METHOD H264_DVR_GetDownloadPos(
long lFileHandle
);

参数： [in]lFileHandle 下载句柄

返回值： 大于等于 0 为下载进度，小于 0 为失败。具体错误码可以通过 H264_DVR_GetLastError 获取。

说明： 获得下载录像的当前位置,可以用于不需要实时显示下载进度的接口,与下载回调函数的功能类似。用于不打算通过回调计算进度,可定时调用本接口获取当前进度。

3.7 云台控制 H264_DVR_PTZControl

函数： H264_DVR_API bool CALL_METHOD H264_DVR_PTZControl(
long lLoginID,
int nChannelNo,
long lPTZCommand,
bool bStop DEF_PARAM(0),
long lSpeed DEF_PARAM(4)
);

参数: [in]lLoginID 登录句柄

[in]nChannelNo 控制的设备通道号

[out]lPTZCommand 控制类型。PTZ_ControlType

[in]bStop 是否是停止

[out]lSpeed	速度，默认 4

返回值: 成功返回 TRUE,失败返回 FALSE。具体错误码通过 H264_DVR_GetLastError 获取。

说明: 控制云台，但是必须在当前通道打开的情况下使用。

3.8 参数配置

3.8.1 获取设备配置 H264_DVR_GetDevConfig

函数:
```
H264_DVR_API long  CALL_METHOD H264_DVR_GetDevConfig(
long lLoginID,
unsigned long dwCommand,
int nChannelNO,
char * lpOutBuffer,
unsigned long dwOutBufferSize,
unsigned long* lpBytesReturned,
int waittime DEF_PARAM(1000)
 );
```

参数 :

[in]lLoginID	登录句柄
[in]dwCommand	配置类型 具体定义见数据结构定义中的 SDK_CONFIG_TYPE
[in]nChannelNO	配置通道号，-1表示所有通道
[out]lpOutBuffer	存放输出参数的缓冲区，根据不同的类型，输出不同的配置结构，具体见数据结构定义中各配置结构
[in]dwOutBufferSize	输入缓冲区的大小，(单位字节).
[out]lpBytesReturned	实际返回的缓冲区大小,对应配置结构的大小(单位字节)

	[in]waittime	等待时间

返回值： 大于 0 成功，小于 0 失败 （可根据错误类型查找）。具体错误码可以通过 H264_DVR_GetLastError 获取。

说明： 不同的获取功能对应不同的结构体和命令号。具体命令在配置信息结构 _SDK_CONFIG_TYPE。

3.8.2 设置设备配置 H264_DVR_SetDevConfig

函数：
```
H264_DVR_API long  CALL_METHOD H264_DVR_SetDevConfig(
long lLoginID,
unsigned long dwCommand,
int nChannelNO,
char * lpInBuffer,
unsigned long dwInBufferSize,
int waittime DEF_PARAM(1000)
);
```

参数：

[in]lLoginID	登录句柄
[in]dwCommand	配置类型，具体定义见数据结构定义中的 SDK_CONFIG_TYPE
[in]nChannelNO	配置通道号，-1表示所有通道
[in]lpInBuffer	存放输入参数的缓冲区，根据不同的类型，输入不同的配置结构，具体见数据结构定义中各配置结构
[in]dwInBufferSize	输入缓冲区的大小(单位字节).
[in]waittime	等待时间

返回值： 大于 0 成功，小于 0 失败 （可根据错误类型查找）。具体错误码可以通过 H264_DVR_GetLastError 获取。

说明： 不同的设置功能对应不同的结构体和命令号。具体命令在配置信息结构

_SDK_CONFIG_TYPE。

3.8.3 跨网段设置设备配置 H264_DVR_SetConfigOverNet

函数： H264_DVR_API long CALL_METHOD H264_DVR_SetConfigOverNet(
unsigned long dwCommand,
int nChannelNO,
char * lpInBuffer,
unsigned long dwInBufferSize,
int waittime DEF_PARAM(1000)
);

参数：

[in]dwCommand	配置类型, E_SDK_CONFIG_SYSNET
[in]nChannelNO	配置通道号，1临时保存,其他为永久保存
[in]lpInBuffer	存放输入参数的缓冲区， 结构体指针或地址 --SDK_CONFIG_NET_COMMON_V3
[in]dwInBufferSize	输入缓冲区的大小(单位字节).
[in]waittime	等待时间

返回值： 等于 0 表示成功，小于 0 表示失败。具体错误码可以通过 H264_DVR_GetLastError 获取。

说明： 跨网段设置设备配置。目前只支持对网络配置进行设置。

3.9 日志管理 H264_DVR_FindDVRLog

函数： H264_DVR_API bool CALL_METHOD H264_DVR_FindDVRLog(
long lLoginID,
SDK_LogSearchCondition *pFindParam,
SDK_LogList *pRetBuffer,
long lBufSize,
int waittime DEF_PARAM(2000)

);

参数：

[in]lLoginID	登录句柄
[in]pFindParam	日志查询条件
[in]pRetBuffer	返回日志信息
[in]lBufSize	日志返回长度
[in]waittime	等待时间

返回值： 成功返回 TRUE，失败返回 FALSE。具体错误码通过 H264_DVR_GetLastError 获取。

说明： 查询日志。

3.10 设备控制 H264_DVR_ControlDVR

函数：

```
H264_DVR_API bool CALL_METHOD H264_DVR_ControlDVR(
long lLoginID,
int type,
int waittime DEF_PARAM(2000)
);
```

参数：

[in]lLoginID	登录句柄
[in]type	0表示重启设备,1表示清除日志,2表示关机，3表示恢复记录日志，4表示停止记录日志，5表示手机对讲恢复关闭之前临时打开过的音频

返回值： 成功返回 TRUE，失败返回 FALSE。具体错误码通过 H264_DVR_GetLastError 获取。

说明： 重启/清除日志/关机/恢复记录日志/停止记录日志/”对讲关闭状态”恢复。

3.11 升级设备程序

3.11.1 本地升级 H264_DVR_Upgrade

函数：　H264_DVR_API long CALL_METHOD H264_DVR_Upgrade(

long lLoginID,

char *sFileName,

int nType DEF_0_PARAM,

fUpgradeCallBack cbUpgrade DEF_0_PARAM,

long dwUser DEF_0_PARAM

);

参数：	[in]lLoginID	登录句柄
	[in]sFileName	要升级的文件名
	[in]nType	要升级的文件类型
	[in]cbUpgrade	回调升级进度
	[in]dwUser	用户数据

typedef void(CALL_METHOD *fUpgradeCallBack) (

long lLoginID,

long lUpgradechannel,

int nTotalSize,

int nSendSize,

long dwUser

);

[out]lLoginID	登录句柄
[out]lUpgradechannel	升级通道
[out]nTotalSize	总数据长度，说明： nTotalSize = -1时, nSendSize:1-99返回升级进度 nTotalSize =0时,nSendSize =

		H264_DVR_NOENOUGH_MEMORY- H264_DVR_INVALID_WIFI_DRIVE升级错误具体码，其他就是发送进度
	[out]nSendSize	发送数据长度，说明： nSendSize = -1 说明升级完成 nSendSize = -2 说明升级出错
	[out]dwUser	用户数据

返回值： 非 0 表示成功，0 表示失败。具体错误码可以通过 H264_DVR_GetLastError 获取。

说明： 设置对前端设备网络升级程序。设置远程程序的升级,返回程序升级句柄。

3.11.2 获取升级状态 H264_DVR_GetUpgradeState

函数： H264_DVR_API int CALL_METHOD H264_DVR_GetUpgradeState(
long lUpgradeHandle
);

参数：	[in]lUpgradeHandle	升级句柄，返回值：1 成功，2 正在升级 3 失败

返回值： 1 表示成功,0 表示失败。具体错误码可以通过 H264_DVR_GetLastError 获取。

说明： 无

3.11.3 释放升级句柄 H264_DVR_CloseUpgradeHandle

函数： H264_DVR_API long CALL_METHOD H264_DVR_CloseUpgradeHandle(
long lUpgradeHandle
);

参数：	[in]lUpgradeHandle	升级句柄

返回值： 1 表示成功,0 表示失败。具体错误码可以通过 H264_DVR_GetLastError 获取。

说明：　　停止升级。

3.11.4 云升级 H264_DVR_Upgrade_Cloud

函数：　　H264_DVR_API long CALL_METHOD H264_DVR_Upgrade_Cloud(

long lLoginID,

SDK_CloudUpgradeVersion *sUpgradeVer,

int nType DEF_0_PARAM,

fUpgradeCallBack cbUpgrade DEF_0_PARAM,

long dwUser DEF_0_PARAM

);

参数：　　[in]lLoginID　　登录句柄

[in]sUpgradeVer　　升级的文件信息

[in]nType　　类型

[out]cbUpgrade　　回调的升级信息

[in]dwUser　　用户参数

typedef void(CALL_METHOD *fUpgradeCallBack) (

long lLoginID,

long lUpgradechannel,

int nTotalSize,

int nSendSize,

long dwUser

);

[out]lLoginID　　登录句柄

[out]lUpgradechannel　　升级通道

[out]nTotalSize　　总数据长度，说明：

nTotalSize = -1时, nSendSize:1-99返

		回升级进度 nTotalSize =0时,nSendSize = H264_DVR_NOENOUGH_MEMORY-H264_DVR_INVALID_WIFI_DRIVE升级错误具体码，其他就是发送进度。云升级增加了nTotalSize=-2时,nSendSize:0-100=下载进度,没有发送进度
	[out]nSendSize	发送数据长度，说明： nSendSize = -1 说明升级完成 nSendSize = -2 说明升级出错
	[out]dwUser	用户数据

返回值： 等于 0 表示成功，小于 0 表示失败。具体错误码可以通过 H264_DVR_GetLastError 获取。

说明： 进行在线升级。

3.11.5 停止云升级 H264_DVR_StopUpgrade_Cloud

函数：
```
H264_DVR_API long CALL_METHOD H264_DVR_StopUpgrade_Cloud(
long lHandle
);
```

参数： [in]lHandle 云升级句柄

返回值： 等于 0 表示成功，小于 0 表示失败。具体错误码可以通过 H264_DVR_GetLastError 获取。

说明： 停止云升级。

3.12 语音对讲

3.12.1 开始对讲 H264_DVR_StartVoiceCom_MR

函数：　H264_DVR_API long CALL_METHOD H264_DVR_StartVoiceCom_MR(

long lLoginID,

pfAudioDataCallBack pVcb,

long dwDataUser

);

参数：　[in]lLoginID　登录句柄

[out]pVcb　从设备接收到的语音对讲数据回调

[in]dwDataUser　回调函数参数

typedef void (CALL_METHOD *pfAudioDataCallBack)(

long lVoiceHandle,

char *pDataBuf,

long dwBufSize,

char byAudioFlag,

long dwUser

);

[out]lVoiceHandle　对讲句柄

[out]pDataBuf　回调数据

[out]dwBufSize　数据大小

[out]byAudioFlag　标志位

[out]dwUser　用户参数

返回值：　非0表示成功,0表示失败。具体错误码可以通过H264_DVR_GetLastError获取。

说明：　开始对讲的过程。

3.12.2 发送对讲数据 H264_DVR_VoiceComSendData

函数： H264_DVR_API bool CALL_METHOD H264_DVR_VoiceComSendData(
long lVoiceHandle,
char *pSendBuf,
long lBufSize
);

参数：

	[in]lVoiceHandle	对讲句柄
	[in]pSendBuf	发送的语音数据
	[in]lBufSize	发送的数据大小

返回值： 成功返回 TRUE,失败返回 FALSE。具体错误码通过 H264_DVR_GetLastError 获取。

说明： 发送对讲数据。

3.12.3 停止对讲 H264_DVR_StopVoiceCom

函数： H264_DVR_API bool CALL_METHOD H264_DVR_StopVoiceCom(
long lVoiceHandle
);

参数：

	[in]lVoiceHandle	对讲句柄

返回值： 成功返回 TRUE,失败返回 FALSE。具体错误码通过 H264_DVR_GetLastError 获取。

说明： 停止对讲。

3.12.4 设置对讲音频编码方式 H264_DVR_SetTalkMode

函数： H264_DVR_API bool CALL_METHOD H264_DVR_SetTalkMode(
long lLoginID,

SDK_AudioInFormatConfig* pTalkMode

);

参数： [in]lLoginID 登录句柄

[in]pTalkMode 对讲模式结构体

返回值： 成功返回TRUE，失败返回FALSE。具体错误码通过H264_DVR_GetLastError 获取。

说明： 设置对讲音频编码方式，用户可以不设置，默认为G711A编码。

3.13 录像模式设置

3.13.1 手动录像 H264_DVR_StartDVRRecord

函数： H264_DVR_API bool CALL_METHOD H264_DVR_StartDVRRecord(

long lLoginID,

int nChannelNo,

long lRecordType

);

参数： [in]lLoginID 登录句柄

[in]nChannelNo 通道号,-1代表全通道，0-n代表单个通道

[in]lRecordType 录像类型 参见：SDK_RecordModeTypes

返回值： 成功返回TRUE，失败返回FALSE。具体错误码通过H264_DVR_GetLastError 获取。

说明： 该接口是为了方便手动开启录像而增加，也可以通过系统配置设置接口（H264_DVR_SetDevConfig）设置录像为手动模式。

3.13.2 关闭录像 H264_DVR_StopDVRRecord

函数： H264_DVR_API bool CALL_METHOD H264_DVR_StopDVRRecord(

long lLoginID,

```
int nChannelNo
);
```

参数：

[in]lLoginID	登录句柄
[in]nChannelNo	通道号,-1代表全通道,0-n代表单个通道

返回值：成功返回TRUE，失败返回FALSE。具体错误码通过H264_DVR_GetLastError 获取。

说明：该接口是为了方便手动关闭录像而增加，也可以通过系统配置设置接口（H264_DVR_SetDevConfig）设置录像为关闭模式

3.14 设置系统时间 H264_DVR_SetSystemDateTime

函数：

```
H264_DVR_API bool CALL_METHOD H264_DVR_SetSystemDateTime(
long lLoginID,
SDK_SYSTEM_TIME *pSysTime,
bool nType DEF_0_PARAM
);
```

参数：

[in]lLoginID	登录句柄
[in]pSysTime	系统时间 ,参见：SDK_SYSTEM_TIME
[in]nType	系统时间类型(true-新的系统时间)

返回值：成功返回TRUE，失败返回FALSE。具体错误码通过H264_DVR_GetLastError 获取。

说明：设置设备系统时间。

3.15 布防式报警

3.15.1 报警状态获取 H264_DVR_SetDVRMessCallBack

函数： H264_DVR_API bool CALL_METHOD H264_DVR_SetDVRMessCallBack(
fMessCallBack cbAlarmcallback,
unsigned long lUser
);

参数： [out]cbAlarmcallback 消息回调函数,可以回调设备的状态,如报警状态可以通过此回调获取;当设置为0时表示禁止回调

[in]lUser 用户数据

```
typedef bool (CALL_METHOD *fMessCallBack)(
long lLoginID,
char *pBuf,
unsigned long dwBufLen,
long dwUser,
int nType
);
```

[out]lLoginID 登录句柄

[out]pBuf 回调数据--SDK_AlarmInfo

[out]dwBufLen 回调数据长度

[out]dwUser 回调函数参数

[out]nType 类型--ALARM_TYPE

返回值： 成功返回TRUE，失败返回FALSE。具体错误码通过H264_DVR_GetLastError 获取。

说明： 设置设备消息回调函数，用来得到设备当前状态信息,与调用顺序无关，SDK默认不回调，此回调函数必须先调用报警回调上传通道接口H264_DVR_SetupAlarmChan才有效，同时需要说明的是针对目前定义的报警，

是每秒回调设备当前的报警信息。

3.15.2 设置报警回调上传通道 H264_DVR_SetupAlarmChan

函数：　H264_DVR_API long CALL_METHOD H264_DVR_SetupAlarmChan(
long lLoginID
);

参数：　[in]lLoginID　登录句柄

返回值：　1表示成功，0表示失败。具体错误码通过H264_DVR_GetLastError 获取。

说明：　开始对某个设备订阅消息,用来设置是否需要对设备消息回调,得到的消息从H264_DVR_SetDVRMessCallBack的设置值回调出来。

3.15.3 关闭报警回调上传通道 H264_DVR_CloseAlarmChan

函数：　H264_DVR_API bool CALL_METHOD H264_DVR_CloseAlarmChan(
long lLoginID
);

参数：　[in]lLoginID　登录句柄

返回值：　1表示成功，0表示失败。具体错误码通过H264_DVR_GetLastError 获取。

说明：　停止对某个设备侦听消息。

3.16 监听式报警

3.16.1 启动报警中心监听 H264_DVR_StartAlarmCenterListen

函数：　H264_DVR_API bool CALL_METHOD H264_DVR_StartAlarmCenterListen(
int nPort,
fMessCallBack cbAlarmCenter,
unsigned long dwDataUser
);

参数：

[in]nPort	监听端口号
[out]cbAlarmCenter	数据回调
[in]dwDataUser	回调参数

```
typedef bool (CALL_METHOD *fMessCallBack)(
long lLoginID,
char *pBuf,
unsigned long dwBufLen,
long dwUser,
int nType
);
```

[out]lLoginID	登录句柄
[out]pBuf	回调数据--SDK_AlarmInfo
[out]dwBufLen	回调数据长度
[out]dwUser	回调函数参数
[out]nType	类型--ALARM_TYPE

返回值：成功返回TRUE，失败返回FALSE。具体错误码通过H264_DVR_GetLastError 获取。

说明：开启报警中心前，需要调用接口（H264_DVR_SetDevConfig）来设置监听设备报警的电脑地址。对应配置命令 E_SDK_CONFIG_ALARM_CENTER，结构体 SDK_NetAlarmServerConfigAll。

3.16.2 关闭报警中心监听 H264_DVR_StopAlarmCenterListen

函数：

```
H264_DVR_API bool CALL_METHOD H264_DVR_StopAlarmCenterListen();
```

参数：无

返回值：无

说明：关闭报警中心监听。

3.17 获取设备的运行状态信息 H264_DVR_GetDVRWorkState

函数： H264_DVR_API bool CALL_METHOD H264_DVR_GetDVRWorkState(

long lLoginID,

SDK_DVR_WORKSTATE *pWorkState

);

参数： [in]lLoginID 登录句柄

[out]pWorkState 设备的工作状态-- SDK_DVR_WORKSTATE

返回值： 成功返回 TRUE,失败返回 FALSE。具体错误码通过 H264_DVR_GetLastError 获取。

说明： 获取设备工作状态信息。

3.18 网络报警 H264_DVR_SendNetAlarmMsg

函数： H264_DVR_API bool CALL_METHOD H264_DVR_SendNetAlarmMsg(

long lLoginID,

SDK_NetAlarmInfo *pAlarmInfo

);

参数： [in]lLoginID 登录句柄

[in]pAlarmInfo 网络报警参数-- SDK_NetAlarmInfo

返回值： 成功返回 TRUE,失败返回 FALSE。具体错误码通过 H264_DVR_GetLastError 获取。

说明： 发送网络报警信息。

3.19 磁盘管理 H264_DVR_StorageManage

函数： H264_DVR_API int CALL_METHOD H264_DVR_StorageManage(

long lLoginID,

SDK_StorageDeviceControl *pStorageCtl

);

参数：　[in]lLoginID　登录句柄

[in]pStorageCtl　操作参数--SDK_StorageDeviceControl

返回值：　1 表示成功，小于等于 0 表示失败。具体错误码通过 H264_DVR_GetLastError 获取。

说明：　磁盘管理。

3.20 设备端抓图 H264_DVR_CatchPic

函数：　H264_DVR_API bool CALL_METHOD H264_DVR_CatchPic(

long lLoginID,

int nChannel,

char *sFileName,

int nType DEF_0_PARAM

);

参数：　[in] lLoginID　登录句柄

[in]nChannel　控制的设备通道号

[in] sFileName　要保存的图片名，全路径

返回值：　成功返回TRUE，失败返回FALSE。具体错误码通过H264_DVR_GetLastError 获取。

说明：　1.需要设备配置里面有抓图配置选项该接口才有效；

2.如果满足1，默认抓出来的分辨率是D1，如果需要抓跟视频分辨率一样的图片，就需要修改编码设置里的抓图分辨率，如果编码设置没有抓图分辨率选项，则需要定制支持该项的程序。

3.21 透明串口

3.21.1 创建透明串口通道 H264_DVR_OpenTransComChannel

函数：　H264_DVR_API bool CALL_METHOD H264_DVR_OpenTransComChannel(

```
long lLoginID,
TransComChannel *TransInfo,
fTransComCallBack cbTransCom,
unsigned long lUser
);
```

参数：

[in]lLoginID	登录句柄
[in]TransInfo	串口参数，具体参考 TransComChannel
[in]cbTransCom	回调

```
typedef void (CALL_METHOD *fTransComCallBack) (
long lLoginID,
long lTransComType,
char *pBuffer,
unsigned long dwBufSize,
unsigned long dwUser
);
```

[out]lLoginID	登录句柄
[out]lTransComType	串口类型，见 SERIAL_TYPE
[out]pBuffer	回调的数据缓冲
[out]dwBufSize	回调的数据长度
[out]dwUser	用户数据

返回值： 成功返回 TRUE,失败返回 FALSE。具体错误码通过 H264_DVR_GetLastError 获取。

说明： 创建透明串口通道。

3.21.2 通过串口向设备写数据 H264_DVR_SerialWrite

函数： H264_DVR_API bool CALL_METHOD H264_DVR_SerialWrite(
long lLoginID,
SERIAL_TYPE nType,
char *pBuffer,
int nBufLen
);

参数：

[in] lLoginID	登录句柄
[in] nType	类型详见SERIAL_TYPE
[int] pBuffer	数据缓冲区
[in] nBufLen	数据缓冲的长度

返回值： 成功返回TRUE，失败返回FALSE。具体错误码通过H264_DVR_GetLastError 获取。

说明： 通过串口向设备写数据。

3.21.3 通过串口从设备读数据 H264_DVR_SerialRead

函数： H264_DVR_API bool CALL_METHOD H264_DVR_SerialRead(
long lLoginID,
SERIAL_TYPE nType,
char *pBuffer,
int nBufLen,
int *pReadLen
);

参数：

[in] lLoginID	登录句柄
[in] nType	类型详见 SERIAL_TYPE
[out] pBuffer	读取数据后缓冲区

	[in] nBufLen	缓冲区内的数据长度
	[out] pReadLen	实际接收到的长度

返回值： 成功返回TRUE，失败返回FALSE。具体错误码通过H264_DVR_GetLastError 获取。

说明： 通过串口从设备读数据。

3.21.4 关闭透明串口通道 H264_DVR_CloseTransComChannel

函数：
```
H264_DVR_API bool CALL_METHOD H264_DVR_CloseTransComChannel(
long lLoginID,
SERIAL_TYPE nType
);
```

参数：	[in]lLoginID	登录句柄
	[in]nType	串口类型，具体参考SERIAL_TYPE

返回值： 成功返回TRUE，失败返回FALSE。具体错误码通过H264_DVR_GetLastError 获取。

说明： 关闭透明串口通道。

3.22 客户端录像

3.22.1 开始本地录像 H264_DVR_StartLocalRecord

函数：
```
H264_DVR_API bool CALL_METHOD H264_DVR_StartLocalRecord(
long lRealHandle,
char* szSaveFileName,
long type=0
);
```

参数：	[in]lRealHandle	播放句柄（H264_DVR_RealPlay的返回

		值）
	[in]szSaveFileName	保存路径
	[in]type	录像类型：（0：文件名后缀为.h264；2：文件名后缀.avi），默认为0；

返回值： 成功返回TRUE，失败返回FALSE。具体错误码通过H264_DVR_GetLastError 获取。

说明： 对预览进行 pc 端录像。

3.22.2 关闭本地录像 H264_DVR_StopLocalPlay

函数：
```
H264_DVR_API bool CALL_METHOD H264_DVR_StopLocalRecord(
long lRealHandle
);
```

参数：	[in]lRealHandle	播放句柄（H264_DVR_RealPlay的返回值）

返回值： 成功返回TRUE，失败返回FALSE。具体错误码通过H264_DVR_GetLastError 获取。

说明： 停止pc端录像

3.23 客户端音频

3.23.1 打开视频通道的音频 H264_DVR_OpenSound

函数：
```
H264_DVR_API bool CALL_METHOD H264_DVR_OpenSound(
long lHandle
);
```

参数：	[in]lHandle	H264_DVR_RealPlay 或 H264_DVR_StartLocalPlay 或 H264_DVR_PlayBackByName 或

H264_DVR_PlayBackByTimeEx 的返回值

返回值： 成功返回TRUE，失败返回FALSE。具体错误码通过H264_DVR_GetLastError 获取。

说明： 打开视频通道的的音频。

3.23.2 关闭视频通道的音频 H264_DVR_CloseSound

函数： H264_DVR_API bool CALL_METHOD H264_DVR_CloseSound(
long lHandle
);

参数： [in]lHandle H264_DVR_RealPlay 或 H264_DVR_StartLocalPlay 或 H264_DVR_PlayBackByName 或 H264_DVR_PlayBackByTimeEx 的返回值

返回值： 成功返回TRUE，失败返回FALSE。具体错误码通过H264_DVR_GetLastError 获取。

说明： 关闭视频通道的的音频。

3.24 播放定位

3.24.1 获取播放位置（百分比）H264_DVR_GetPlayPos

函数： H264_DVR_API float CALL_METHOD H264_DVR_GetPlayPos(
long lPlayHandle
);

参数： [in] lPlayHandle H264_DVR_StartLocalPlay 或

H264_DVR_PlayBackByName 或 H264_DVR_PlayBackByTimeEx 的返回值

返回值： 播放百分比

说明： 获取回放或本地播放的播放进度，回放时传窗口句柄该接口才有效。

3.24.2 设置播放位置（百分比）H264_DVR_SetPlayPos

函数：
```
H264_DVR_API bool CALL_METHOD H264_DVR_SetPlayPos(
long lPlayHandle,
float fRelativPos
);
```

参数：

[in] lPlayHandle	H264_DVR_StartLocalPlay 或 H264_DVR_PlayBackByName 或 H264_DVR_PlayBackByTimeEx 的返回值
[in]fRelativPos	播放百分比

返回值： 成功返回TRUE，失败返回FALSE。具体错误码通过H264_DVR_GetLastError 获取。

说明： 设置回放或本地播放的播放进度，回放时传窗口句柄该接口才有效。

3.25 设置信息帧回调 H264_DVR_SetInfoFrameCallBack

函数：
```
H264_DVR_API bool CALL_METHOD H264_DVR_SetInfoFrameCallBack(
long lPlayHandle,
InfoFramCallBack callback,
long user
);
```

参数：　[in] lPlayHandle　　H264_DVR_RealPlay 或 H264_DVR_StartLocalPlay 或 H264_DVR_PlayBackByName 或 H264_DVR_PlayBackByTimeEx 的返回值

[in]callback　　回调函数

[in] user　　用户自定义数据

```
typedef void (CALL_METHOD  *InfoFramCallBack)(
long lPlayHand,
long nType,
LPCSTR pBuf,
long nSize,
long nUser
);
```

[out]lPlayHand　　播放句柄

[out]nType　　数据类型—暂时不需要判断

[out]pBuf　　回调数据

[out]nSize　　回调数据长度

[out]nUser　　用户回调参数

返回值：　成功返回TRUE，失败返回FALSE。具体错误码通过H264_DVR_GetLastError 获取。

说明：　设置信息帧回调。

3.26 客户端视频颜色

3.26.1 获取播放视频颜色信息 H264_DVR_LocalGetColor

函数：
```
H264_DVR_API bool CALL_METHOD H264_DVR_LocalGetColor(
long lHandle,
DWORD nRegionNum,
LONG *pBrightness,
LONG *pContrast,
LONG *pSaturation,
LONG *pHue
);
```

参数：

[in] lHandle	H264_DVR_RealPlay 或 H264_DVR_StartLocalPlay 或 H264_DVR_PlayBackByName 或 H264_DVR_PlayBackByTimeEx 的返回值
[in]nRegionNum	区域（暂时没有：可设为0）
[out]pBrightness	亮度
[out]pContrast	对比度
[out]pSaturation	饱和度
[out]pHue	色度

返回值：成功返回TRUE，失败返回FALSE。具体错误码通过H264_DVR_GetLastError 获取。

说明：获取播放视频颜色信息。

3.26.2 设置播放视频颜色信息 H264_DVR_LocalSetColor

函数： H264_DVR_API bool CALL_METHOD H264_DVR_LocalSetColor(

long lHandle,

DWORD nRegionNum,

LONG nBrightness,

LONG nContrast,

LONG nSaturation,

LONG nHue

);

参数：

	[in] lHandle	H264_DVR_RealPlay 或 H264_DVR_StartLocalPlay 或 H264_DVR_PlayBackByName 或 H264_DVR_PlayBackByTimeEx 的返回值
	[in]nRegionNum	区域（暂时没有：可设为0）
	[in]nBrightness	亮度
	[in]nContrast	对比度
	[in]nSaturation	饱和度
	[in]nHue	色度

返回值： 成功返回TRUE，失败返回FALSE。具体错误码通过H264_DVR_GetLastError 获取。

说明： 设置播放视频颜色信息。

3.27 播放客户端本地文件

3.27.1 播放本地文件 H264_DVR_StartLocalPlay

函数： H264_DVR_API long CALL_METHOD H264_DVR_StartLocalPlay(

```
char*pFileName,
void* hWnd,
fPlayDrawCallBack drawCallBack=0,
long user=0
);
```

参数：

[in]pFileName	播放文件名
[in]hWnd	播放窗口句柄
[in]drawCallBack	回调函数（不用可以设为NULL）
[in]user	用户自定义数据

```
typedef void (CALL_METHOD * fPlayDrawCallBack)(
long lPlayHand,
HDC hDc,
long nUser
);
```

[out]lPlayHand	播放句柄
[out]hDc	
[out]nUser	回调参数

返回值：失败返回 0，成功返回播放 ID(本地播放句柄)，将作为相关函数的参数。

说明：播放本地.h264 视频文件。

3.27.2 关闭本地播放 H264_DVR_StopLocalPlay

函数：

```
H264_DVR_API bool CALL_METHOD H264_DVR_StopLocalPlay(
long lPlayHandle
);
```

参数：[in] lPlayHandle　　H264_DVR_StartLocalPlay 返回值

返回值：成功返回TRUE，失败返回FALSE。具体错误码通过H264_DVR_GetLastError 获取。

说明：关闭本地播放。

3.27.3 本地文件播放结束回调 H264_DVR_SetFileEndCallBack

函数： H264_DVR_API bool CALL_METHOD H264_DVR_SetFileEndCallBack(
long lPlayHandle,
fLocalPlayFileCallBack callBack,
long user
);

参数： [in]lPlayHandle H264_DVR_StartLocalPlay返回值
[in]callBack 结束回调
[in]user 用户自定义数据

typedef void (CALL_METHOD * fLocalPlayFileCallBack)(
long lPlayHand,
long nUser
);

[out]lPlayHand 播放句柄
[out]nUser 回调参数

返回值： 成功返回TRUE，失败返回FALSE。具体错误码通过H264_DVR_GetLastError 获取。

说明： 本地文件播放结束回调。

3.27.4 本地文件播放控制 H264_DVR_LocalPlayCtrl

函数： H264_DVR_API bool CALL_METHOD H264_DVR_LocalPlayCtrl(
long lPlayHandle,
SDK_LoalPlayAction action,
long lCtrlValue
);

参数： [in]lPlayHandle H264_DVR_StartLocalPlay 返回值

	[in]action	参见：SDK_LoalPlayAction
	[in]lCtrlValue	快放（1,2,3,4 级别），和慢放（1,2,3,4 级别）

返回值： 成功返回TRUE，失败返回FALSE。具体错误码通过H264_DVR_GetLastError 获取。

说明： 播放控制（播放，停止，恢复，快发，慢放）。

3.28 检测子连接断开 H264_DVR_SetSubDisconnectCallBack

函数：
```
H264_DVR_API long CALL_METHOD H264_DVR_SetSubDisconnectCallBack(
fSubDisConnectCallBack callBack,
DWORD userData
);
```

参数：

	[in]callBack	子连接断线回调(参见SubConnType)
	[in]userData	用户数据

```
typedef void (CALL_METHOD *fSubDisConnectCallBack)(
long lLoginID,
SubConnType type,
long nChannel,
long dwUser
);
```

	[out]lLoginID	登录句柄
	[out]type	数据类型-- SubConnType
	[out]nChannel	通道号
	[out]dwUser	用户参数

返回值： 1表示成功，0表示失败。

说明： 检测子连接异常断开。

3.29 设置保活时间及断线检测时间 H264_DVR_SetKeepLifeTime

函数： H264_DVR_API long CALL_METHOD H264_DVR_SetKeepLifeTime(
long lLoginID,
unsigned int perKeeplifeTime,
unsigned int detectDisconTime
);

参数：

[in]lLoginID	登录句柄
[in]perKeeplifeTime	发送心跳包的间隔时间(单位秒)
[in]detectDisconTime	设备断线时间(单位秒)

返回值： 1表示成功，0表示失败。

说明： 设置保活时间,perKeeplifeTime(心跳间隔):默认1秒,detectDisconTime(断线检测时间):默认60秒。

3.30 搜索局域网内设备 H264_DVR_SearchDevice

函数： H264_DVR_API bool CALL_METHOD H264_DVR_SearchDevice(
char* szBuf,
int nBufLen,
int* pRetLen,
int nSearchTime
);

参数：

[out]szBuf	接收缓冲
[in]nBufLen	接收缓冲大小, sizeof(SDK_CONFIG_NET_COMMON_V2)*n
[in]pRetLen	返回的大小
[in]nSearchTime	等待时间

返回值： 成功返回TRUE，失败返回FALSE。具体错误码通过H264_DVR_GetLastError 获

取。

说明：　　搜索局域网内设备。

4 错误码枚举

错误码名称	错误返回值	说明
H264_DVR_NOERROR	0	没有错误
H264_DVR_SUCCESS	1	返回成功
H264_DVR_SDK_NOTVALID	-10000	非法请求
H264_DVR_NO_INIT	-10001	SDK 未经初始化
H264_DVR_ILLEGAL_PARAM	-10002	用户参数不合法
H264_DVR_INVALID_HANDLE	-10003	句柄无效
H264_DVR_SDK_UNINIT_ERROR	-10004	SDK 清理出错
H264_DVR_SDK_TIMEOUT	-10005	等待超时
H264_DVR_SDK_MEMORY_ERROR	-10006	内存错误，创建内存失败
H264_DVR_SDK_NET_ERROR	-10007	网络错误
H264_DVR_SDK_OPEN_FILE_ERROR	-10008	打开文件失败
H264_DVR_SDK_UNKNOWNERROR	-10009	未知错误
H264_DVR_DEV_VER_NOMATCH	-11000	收到数据不正确，可能版本不匹配
H264_DVR_SDK_NOTSUPPORT	-11001	版本不支持

H264_DVR_ANAS_EXIST	-11130	NAS 地址已存在
H264_DVR_ANAS_ALIVE	-11131	路径被使用，无法操作
H264_DVR_ANAS_FULL	-11132	NAS 已达到支持的最大值
H264_DVR_OPEN_CHANNEL_ERROR	-11200	打开通道失败,可能检测到设备已经不在线
H264_DVR_CLOSE_CHANNEL_ERROR	-11201	关闭通道失败
H264_DVR_SUB_CONNECT_ERROR	-11202	建立媒体子连接失败,网络出错或者设备可能不在线
H264_DVR_SUB_CONNECT_SEND_ERROR	-11203	媒体子连接通讯失败,可能检测到设备已经不在线
H264_DVR_NATCONNET_REACHED_MAX	-11204	Nat 视频链接达到最大，不允许新的 Nat 视频链接
H264_DVR_NOTSUPPORT	-11205	版本不支持
H264_DVR_NOTVALID	-11206	请求非法,主连接可能已断开
H264_DVR_TCPCONNET_REACHED_MAX	-11207	Tcp 视频链接达到最大，不允许新的 Tcp 视频链接
H264_DVR_OPENEDPREVIEW	-11208	该通道已经打开预览(通道的打开关闭需要一一对应,打开几次需要关闭几次;不一致会打开提示该错误;预防客户端开发逻辑上的不合理设计增加该错误值)
H264_DVR_NOPOWER	-11300	无权限
H264_DVR_PASSWORD_NOT_VALID	-11301	账号密码不对

H264_DVR_LOGIN_USER_NOEXIST	-11302	用户不存在
H264_DVR_USER_LOCKED	-11303	该用户被锁定
H264_DVR_USER_IN_BLACKLIST	-11304	该用户不允许访问(在黑名单中)
H264_DVR_USER_HAS_USED	-11305	该用户已登录
H264_DVR_USER_NOT_LOGIN	-11306	该用户没有登录
H264_DVR_CONNECT_DEVICE_ERROR	-11307	可能设备不存在
H264_DVR_ACCOUNT_INPUT_NOT_VALID	-11308	用户管理输入不合法
H264_DVR_ACCOUNT_OVERLAP	-11309	索引重复
H264_DVR_ACCOUNT_OBJECT_NONE	-11310	不存在对象，用于查询时
H264_DVR_ACCOUNT_OBJECT_NOT_VALID	-11311	不存在对象
H264_DVR_ACCOUNT_OBJECT_IN_USE	-11312	对象正在使用
H264_DVR_ACCOUNT_SUBSET_OVERLAP	-11313	子集超范围(如组的权限超过权限表，用户权限超出组的权限范围等等)
H264_DVR_ACCOUNT_PWD_NOT_VALID	-11314	密码不正确
H264_DVR_ACCOUNT_PWD_NO	-11315	密码不匹配

T_MATCH		
H264_DVR_ACCOUNT_RESERVED	-11316	保留账号
H264_DVR_ACCOUNT_SYS_MAINTAIN	-11317	系统维护中，不可登录
H264_DVR_EE_DVR_PASSWORD_NOT_VALID2	-11318	账号密码不对
H264_DVR_OPT_RESTART	-11400	保存配置后需要重启应用程序
H264_DVR_OPT_REBOOT	-11401	需要重启系统
H264_DVR_OPT_FILE_ERROR	-11402	写文件出错
H264_DVR_OPT_CAPS_ERROR	-11403	配置特性不支持
H264_DVR_OPT_VALIDATE_ERROR	-11404	配置校验失败
H264_DVR_OPT_CONFIG_NOT_EXIST	-11405	请求或设置的配置不存在
H264_DVR_CTRL_PAUSE_ERROR	-11500	暂停失败
H264_DVR_SDK_NOTFOUND	-11501	查找失败，没有找到对应文件
H264_DVR_CFG_NOT_ENABLE	-11502	配置未启用
H264_DVR_DECORD_FAIL	-11503	解码失败
H264_DVR_SOCKET_ERROR	-11600	创建套接字失败
H264_DVR_SOCKET_CONNECT	-11601	连接套接字失败
H264_DVR_SOCKET_DOMAIN	-11602	域名解析失败

H264_DVR_SOCKET_SEND	-11603	发送数据失败
H264_DVR_ARSP_NO_DEVICE	-11604	没有获取到设备信息，设备应该不在线
H264_DVR_ARSP_BUSING	-11605	ARSP 服务繁忙
H264_DVR_ARSP_BUSING_SELECT	-11606	ARSP 服务繁忙，selcet 失败
H264_DVR_ARSP_BUSING_RECVICE	-11607	ARSP 服务繁忙，receive 失败
H264_DVR_CONNECTSERVER_ERROR	-11608	连接服务器失败
H264_DVR_CONNECT_AGNET	-11609	代理
H264_DVR_CONNECT_NAT	-11610	穿透
H264_DVR_CONNECT_FAILED	-11611	连接失败
H264_DVR_CONNECT_FULL	-11612	服务器连接数已满
H264_DVR_CLOUD_LOGIN_ERR	-11613	云登陆具体的错误码,说明:当登陆接口的 error=-11613 时，通过 H264_DVR_DEVICEINFO 成员 sCloudErrCode 获取错误码
H264_DVR_NO_CONNECT_FRONT	-11614	前端设备未连接或者连接的前端设备分辨率未知
H264_DVR_LOGIN_FULL	-11615	登录句柄已达到最大值，无法再登录
H264_DVR_ARSP_USER_NOEXIST	-11619	用户不存在

H264_DVR_ARSP_PASSWORD_ERROR	-11620	账号密码不对
H264_DVR_ARSP_QUERY_ERROR	-11621	查询失败
H264_DVR_PIRATESOFTWARE	-11700	设备盗版
H264_DVR_AUTH_TIMEOUT	-11800	鉴权超时
H264_DVR_AUTH_FILE_FAILED	-11801	鉴权文件失败
H264_DVR_GAIN_LIST_TIMEOUT	-11802	获取服务器列表超时
H264_DVR_AUTH_CODE_ERR	-11803	鉴权码错误
H264_DVR_NOENOUGH_MEMORY	-11804	内存不足
H264_DVR_INVALID_FORMAT	-11805	升级文件格式不对
H264_DVR_UPDATE_PART_FAIL	-11806	某个分区升级失败
H264_DVR_INVALID_HARDWARE	-11807	硬件型号不匹配
H264_DVR_INVALID_VENDOR	-11808	客户信息不匹配
H264_DVR_INVALID_COMPALIBLE	-11809	升级程序的兼容版本号比设备现有的小，不允许设备升级回老程序
H264_DVR_INVALID_VERSION	-11810	非法的版本
H264_DVR_INVALID_WIFI_DRIVE	-11811	升级程序里 wifi 驱动和设备当前在使用的 wifi 网卡不匹配

H264_DVR_INVALID_CUR_FLASH	-11812	升级程序不支持设备使用的 Flash
H264_DVR_NAT_INIT_TIMEOUT	-12000	云登录初始化超时用于区别一般超时情况

5 示例功能实现

请参看 ClientDemo 程序和 DEMO 说明.doc。