网络 SDK 开发手册

VERSION 1.4.0.0 (Build 090926)

2009-01-10

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。

修订记录

日期	修订内容
2009.02.12	增加局域网内搜索设备接口 H264_DVR_SearchDevice
2009.02.18	增加语音对讲相关接口 H264_DVR_StartVoiceCom_MR， H264_DVR_VoiceComSendData，H264_DVR_StopVoiceCom，H264_DVR_SetTalkMode
2009.09.26	增加以下接口： H264_DVR_StartDVRRecord ， H264_DVR_StopDVRRecord ， H264_DVR_SetSystemDateTime ，H264_DVR_GetDVRWorkState， H264_DVR_ClickKey

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，如果登陆的软件是特殊的(不是 web)可以调用 **H264_DVR_LoginEx()**指定登陆的软件类型, 登录成功后返回一个全局唯一的句柄。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

A. 初始化

SDK 初始化	*H264_DVR_Init ()*

B. SDK 功能信息获取

设置报警消息回调	H264_DVR_SetDVRMessCallBack ()

C. 登录连接设备

登入设备	H264_DVR_Login () H264_DVR_LoginEx ()
报警消息订阅	H264_DVR_SetupAlarmChan ()

D. 设备功能操作与信息获取

系统参数配置	H264_DVR_GetDevConfig () H264_DVR_SetDevConfig ()
查询日志	H264_DVR_FindDVRLog ()
云台控制	H264_DVR_PTZControl () H264_DVR_PTZControlEx ()

E. 实时监视通道

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

F. 回放/下载通道

查询录像	H264_DVR_FindFile ()
回放及控制	H264_DVR_PlayBackByName H264_DVR_PlayBackControl() H264_DVR_StopPlayBack()
下载	H264_DVR_GetFileByName () 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 ()
断开连接	H264_DVR_Logout ()

I. 释放 SDK 资源

SDK 退出	H264_DVR_Cleanup ()

2 数据结构定义

2.1 客户端数据结构

```
//云台操作类型
typedef enum PTZ_ControlType
{
    TILT_UP = 0,          //上
    TILT_DOWN,            //下
    PAN_LEFT,             //左
    PAN_RIGHT,            //右
    PAN_LEFTTOP,          //左上
    PAN_LEFTDOWN,         //左下
    PAN_RIGTHTOP,         //右上
    PAN_RIGTHDOWN,        //右下
    ZOOM_IN,              //变倍大
    ZOOM_OUT,             //变倍小
    FOCUS_FAR,            //焦点后调
    FOCUS_NEAR,           //焦点前调
    IRIS_OPEN,            //光圈扩大
    IRIS_CLOSE,           //光圈缩小

    EXTPTZ_OPERATION_ALARM,         ///< 报警功能
    EXTPTZ_LAMP_ON,                 ///< 灯光开
    EXTPTZ_LAMP_OFF,                //灯光关
    EXTPTZ_POINT_SET_CONTROL,       //设置预置点
    EXTPTZ_POINT_DEL_CONTROL,       //清除预置点
    EXTPTZ_POINT_MOVE_CONTROL,      //转预置点
    EXTPTZ_STARTPANCRUISE,          //开始水平旋转
    EXTPTZ_STOPPANCRUISE,           //停止水平旋转
    EXTPTZ_SETLEFTBORDER,           //设置左边界
    EXTPTZ_SETRIGHTBORDER,          //设置右边界
    EXTPTZ_STARTLINESCAN,           //自动扫描开始
    EXTPTZ_CLOSELINESCAN,           //自动扫描开停止
    EXTPTZ_ADDTOLOOP,               //加入预置点到巡航  p1巡航线路  p2预置点值
    EXTPTZ_DELFROMLOOP,             //删除巡航中预置点  p1巡航线路  p2预置点值
    EXTPTZ_POINT_LOOP_CONTROL,      //开始巡航
    EXTPTZ_POINT_STOP_LOOP_CONTROL,//停止巡航
    EXTPTZ_CLOSELOOP,               //清除巡航  p1巡航线路
    EXTPTZ_FASTGOTO,                //快速定位
```

```
    EXTPTZ_AUXIOPEN,            //辅助开关，关闭在子命令中
    EXTPTZ_OPERATION_MENU,          //球机菜单操作，其中包括开，关，确定等等
    EXTPTZ_REVERSECOMM,         //镜头翻转
    EXTPTZ_OPERATION_RESET,     ///< 云台复位

    EXTPTZ_TOTAL,
};
```

错误类型代号，用于 GetLastError 函数的返回

```
typedef enum SDK_RET_CODE
{
    H264_DVR_NOERROR              = 0,      //没有错误
    H264_DVR_SUCCESS              = 1,      //返回成功
    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_DEV_VER_NOMATCH          = -11000, //收到数据不正确,可能版本不匹配
    H264_DVR_ERROR_GET_DATA       = -11001, //获取信息失败(包括配置,用户信息等)


    H264_DVR_OPEN_CHANNEL_ERROR       = -11200, //打开通道失败
    H264_DVR_CLOSE_CHANNEL_ERROR  = -11201, //关闭通道失败

    /// 用户管理部分错误码
    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, // 该用户以登陆
    H264_DVR_USER_NOT_LOGIN       = -11305, // 该用户没有登陆
    H264_DVR_CONNECT_DEVICE_ERROR   = -11306,   //可能设备不存在

    /// 配置管理相关错误码

    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, // 请求或者设置的配置不存在

    H264_DVR_CTRL_PAUSE_ERROR     = -11500, //暂停失败
};
```

☞ *报警事件类型*

```
enum SDK_EventCodeTypes
{
    SDK_EVENT_CODE_INIT = 0,
    SDK_EVENT_CODE_LOCAL_ALARM = 1,   //本地报警
    SDK_EVENT_CODE_NET_ALARM,         //网络报警
    SDK_EVENT_CODE_MANUAL_ALARM,      //手动报警
    SDK_EVENT_CODE_VIDEO_MOTION,      //动态检测
    SDK_EVENT_CODE_VIDEO_LOSS,        //视频丢失
    SDK_EVENT_CODE_VIDEO_BLIND,       //视频遮挡
    SDK_EVENT_CODE_VIDEO_TITLE,
    SDK_EVENT_CODE_VIDEO_SPLIT,
    SDK_EVENT_CODE_VIDEO_TOUR,
    SDK_EVENT_CODE_STORAGE_NOT_EXIST,
    SDK_EVENT_CODE_STORAGE_FAILURE,
    SDK_EVENT_CODE_LOW_SPACE,
    SDK_EVENT_CODE_NET_ABORT,
    SDK_EVENT_CODE_COMM,
    SDK_EVENT_CODE_STORAGE_READ_ERROR,
    SDK_EVENT_CODE_STORAGE_WRITE_ERROR,
    SDK_EVENT_CODE_NET_IPCONFLICT,
    SDK_EVENT_CODE_ALARM_EMERGENCY,
    SDK_EVENT_CODE_DEC_CONNECT,
    SDK_EVENT_CODE_ILLEGALLY_ENTERED,    //入侵报警
    SDK_EVENT_CODE_REMAINORMOVE,         //滞留或盗移报警
    SDK_EVENT_CODE_ILLEGALLY_BEHAVIOR,   //异常行为报警
    SDK_EVENT_CODE_FACE_RECOGNITION,     //人脸识别
    SDK_EVENT_CODE_LICENSE_RECOGNITION,  //车牌识别
    SDK_EVENT_CODE_NR,
};
//报警信息
typedef struct SDK_ALARM_INFO
{
```

```
    int nChannel;
    int iEvent;
    int iStatus;
    SDK_SYSTEM_TIME SysTime;
}SDK_AlarmInfo;
```

2.1.1 设备信息结构

☞ *设备结构定义如下*

```
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
}H264_DVR_DEVICEINFO,*LPH264_DVR_DEVICEINFO;
```

2.1.2 时间信息

```
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;
```

2.1.3 录像文件信息

//查询条件结构体

```
typedef struct
{
    int nChannelN0;          //通道号
    int nFileType;           //录像类型
    H264_DVR_TIME startTime;   //开始时间
    H264_DVR_TIME endTime; //结束时间
    char szCard[32];         //卡号

}H264_DVR_FINDINFO;
```

//返回录像信息结构体

```
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
}H264_DVR_FILE_DATA;
```

☞ *串口协议信息*

```
struct SDK_COMMATTRI
{
    int iDataBits;    // 数据位取值为,6,7,8
    int iStopBits;    // 停止位
    int iParity;  // 校验位
    int iBaudRate;    // 实际波特率
};
// 串口配置
struct SDK_CONFIG_COMM_X
{
    char iProtocolName[32];// 串口协议:“Console”
    int iPortNo;     // 端口号
    SDK_COMMATTRI aCommAttri;      // 串口属性
};
```

☞云台协议

```
struct SDK_STR_CONFIG_PTZ
{
    char sProtocolName[NET_MAX_PTZ_PROTOCOL_LENGTH];  // 协议名称
    int ideviceNo;              // 云台设备地址编号
    int iNumberInMatrixs;       // 在矩阵中的统一编号
    int iPortNo;            // 串口端口号  [1, 4]
    SDK_COMMATTRI dstComm;          // 串口属性
};

//所有通道云台协议
struct SDK_STR_PTZCONFIG_ALL
{
    SDK_STR_CONFIG_PTZ ptzAll[NET_MAX_CHANNUM];
};
```

☞*用户管理功能数据结构*

权限列表

```
typedef struct _OPR_RIGHT
{
    string    name;
}OPR_RIGHT;

typedef struct _USER_INFO
{
    int       rigthNum;
    string    rights[NET_MAX_RIGTH_NUM];
    string    strGroupname;
    string    strmemo;
    string    strname;
    string    strpassWord;
    bool      reserved;     //是否保留
    bool      shareable;        //本用户是否允许复用1-复用，-不复用
}USER_INFO;

typedef struct _USER_GROUP_INFO
{
    int       rigthNum;
    string    memo;
    string    name;
    string    rights[NET_MAX_RIGTH_NUM]; //权限列表
}USER_GROUP_INFO;
```

```
//用户信息配置结构
typedef struct _USER_MANAGE_INFO
{
    int             rigthNum;
    OPR_RIGHT       rightList[NET_MAX_RIGTH_NUM];
    int             groupNum;
    USER_GROUP_INFO     groupList[NET_MAX_GROUP_NUM];
    int             userNum;
    USER_INFO       userList[NET_MAX_USER_NUM];
}USER_MANAGE_INFO;

//修改用户
typedef struct _CONF_MODIFYUSER
{
    std::string sUserName;
    USER_INFO User;
}CONF_MODIFYUSER;

//修改组
typedef struct _CONF_MODIFYGROUP
{
    std::string sGroupName;
    USER_GROUP_INFO Group;
}CONF_MODIFYGROUP;


/// 修改用户密码请求
struct _CONF_MODIFY_PSW
{
    std::string sUserName;
    std::string sPassword;
    std::string sNewPassword;
};
```

☞ *日志信息*

```
#define NET_MAX_RETURNED_LOGLIST  1024       //最多日志条数
/// 日志查询条件
struct SDK_LogSearchCondition
{
    int nType;    ///< 日志类型
    SDK_SYSTEM_TIME stBeginTime;  ///< 查询日志开始时间
    SDK_SYSTEM_TIME stEndTime;    ///< 查询日志结束时间
};
//日志返回信息
struct SDK_LogList
{
    int iNumLog;
    struct LogItem
    {
        char sType[24];  ///< 日志类型
        char sUser[32];  ///< 日志用户
        char sData[68];  ///< 日志数据
        SDK_SYSTEM_TIME stLogTime; ///< 日志时间
    } Logs[NET_MAX_RETURNED_LOGLIST];
};
```

☞ *查询硬盘信息的返回数据结构*

```
struct SDK_STORAGEDISK
{
    int    iPhysicalNo;
    int    iPartNumber;     // 分区数
    SDK_DriverInformation diPartitions[SDK_MAX_DRIVER_PER_DISK];
};
struct SDK_StorageDeviceInformationAll
{
    int iDiskNumber;
    SDK_STORAGEDISK vStorageDeviceInfoAll[SDK_MAX_DISK_PER_MACHINE];
};
```

☞ *网络监视*

```
typedef struct{
    int nChannel;//通道号
    int nStream;  //0表示主码流，为表示子码流
    int nMode;    //0：TCP方式,1：UDP方式,2：多播方式,3 - RTP方式，-音视频分开(TCP)
}H264_DVR_CLIENTINFO,*LPH264_DVR_CLIENTINFO;
```

2.1.4 配置信息结构

H264_DVR_GetDevConfig、H264_DVR_SetDevConfig 的命令定义

```
typedef enum _SDK_CONFIG_TYPE
{
    E_SDK_CONFIG_NOTHING = 0,  //错误
    //用户管理部分
    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   //同增加用户
    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     //同增加组
    E_SDK_CONFIG_MODIFY_PSW,   //修改密码   _CONF_MODIFY_PSW


    //能力集部分
    E_SDK_CONFIG_ABILITY_SYSFUNC = 9,//支持的网络功能     SDK_SystemFunctio
    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,//动态检测块       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

    //配置部分
    E_SDK_CONFIG_SYSINFO = 17,     //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,           //网络设置       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_ALL
    E_SDK_CONFIG_MOTION,           //动态检测页面   SDK_MOTIONCONFIG
    E_SDK_CONFIG_SHELTER,       //视频遮挡          SDK_BLINDDETECTCONFIG_ALL
    E_SDK_CONFIG_VIDEO_LOSS,    //视频丢失,         SDK_VIDEOLOSSCONFIG_ALL
    E_SDK_CONFIG_ALARM_IN,      //报警输入          SDK_ALARM_INPUTCONFIG_ALL
    E_SDK_CONFIG_ALARM_OUT,     //报警输出
```

```
    E_SDK_CONFIG_DISK_MANAGER  //硬盘管理界面
    E_SDK_CONFIG_OUT_MODE,     //输出模式界面
    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
    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, //网络部分：黑/白名单   SDK_NetIPFilterConfig
    E_SDK_CONFIG_NET_DHCP,     //网络部分：DHCP
    E_SDK_CONFIG_NET_DDNS,     //网络部分：DDNS        SDK_NetDDNSConfigALL
    E_SDK_CONFIG_NET_EMAIL,    //网络部分：EMAIL       SDK_NetEmailConfig
    E_SDK_CONFIG_NET_MULTICAST, //网络部分:组播        SDK_NetMultiCastConfig
    E_SDK_CONFIG_NET_NTP,      //网络部分：  NTP       SDK_NetNTPConfig
    E_SDK_CONFIG_NET_PPPOE,    //网络部分：  PPPOE     SDK_NetPPPoEConfig
    E_SDK_CONFIG_NET_DNS,      //网络部分：  DNS       SDK_NetDNSConfig
    E_SDK_CONFIG_NET_FTPSERVER, //网络部分：FTP        SDK_FtpServerConfig
    E_SDK_CONFIG_SYS_TIME, //系统时间
    E_SDK_CONFIG_CLEAR_LOG,    //清除日志
    E_SDK_REBOOT_DEV,          //重启启动设备

    E_SDK_CONFIG_ABILITY_LANG,          //支持语言
    E_SDK_CONFIG_VIDEO_FORMAT,          //视频制式
    E_SDK_CONFIG_COMBINEENCODE,         //组合编码
    E_SDK_CONFIG_EXPORT,                //配置导出
    E_SDK_CONFIG_IMPORT,                //配置导入
    E_SDK_LOG_EXPORT,                   //日志导出
    E_SDK_CONFIG_COMBINEENCODEMODE,     //组合编码模式
    E_SDK_WORK_STATE,                   //运行状态
}SDK_CONFIG_TYPE;

/// 支持的DDNS类型
struct SDK_DDNSServiceFunction
{
    int  nTypeNum;
    char vDDNSType[NET_MAX_DDNS_TYPE][64];
};
/// 区域遮挡能力集
struct SDK_BlindDetectFunction
{
    int iBlindConverNum;    ///< 区域遮挡块数
};

/// 动检区域能力集
struct SDK_MotionDetectFunction
```

```
{
    int iGridRow;
    int iGridColumn;
};
/// 串口协议
struct SDK_COMMFUNC
{
    int nProNum; /// 协议个数
    char vCommProtocol[SDK_COM_TYPES][32];
};
/// 云台协议
struct SDK_PTZPROTOCOLFUNC
{
    int nProNum; /// 协议个数
    char vPTZProtocol[100][NET_MAX_PTZ_PROTOCOL_LENGTH];
};
/// 编码信息
struct SDK_EncodeInfo
{
    bool bEnable;           ///< 使能项
    int iStreamType;        ///< 码流类型，capture_channel_t
    bool bHaveAudio;        ///< 是否支持音频
    unsigned int uiCompression;     ///< capture_comp_t的掩码
    unsigned int uiResolution;      ///< capture_size_t的掩码
};
/// 编码能力
struct CONFIG_EncodeAbility
{
    int iMaxEncodePower;        ///< 支持的最大编码能力
    SDK_EncodeInfo vEncodeInfo[SDK_CHL_FUNCTION_NUM];  ///< 编码信息,最大四种码流
    SDK_EncodeInfo vCombEncInfo[SDK_CHL_FUNCTION_NUM]; ///< 组合编码信息,最大四种码流
};

///支持的系统功能
struct SDK_SystemFunction
{
    bool vEncodeFunction[SDK_ENCODE_FUNCTION_TYPE_NR]; ///< 编码功能EncodeFunctionTypes
    bool vAlarmFunction[SDK_ALARM_FUNCTION_TYPE_NR];   ///< 报警功能AlarmFucntionTypes
    bool vNetServerFunction[SDK_NET_SERVER_TYPES_NR];  ///< 网络服务功能NetServerTypes
    bool vPreviewFunction[SDK_PREVIEW_TYPES_NR];       ///< 预览功能PreviewTypes
};

///< 自动维护设置
```

```
struct SDK_AutoMaintainConfig
{
    int iAutoRebootDay;                  ///< 自动重启设置日期
    int iAutoRebootHour;             ///< 重启整点时间 [0, 23]
    int iAutoDeleteFilesDays;        ///< 自动删除文件时间[0, 30]
};
//硬盘信息
struct SDK_STORAGEDISK
{
    int      iPhysicalNo;
    int      iPartNumber;       // 分区数
    SDK_DriverInformation diPartitions[SDK_MAX_DRIVER_PER_DISK];
};
struct SDK_StorageDeviceInformationAll
{
    int iDiskNumber;
    SDK_STORAGEDISK vStorageDeviceInfoAll[SDK_MAX_DISK_PER_MACHINE];
};

// 云台联动类型
enum PtzLinkTypes
{
    PTZ_LINK_NONE,          // 不需要联动
    PTZ_LINK_PRESET,        // 转至预置点
    PTZ_LINK_TOUR,          // 巡航
    PTZ_LINK_PATTERN        // 轨迹
};

// 云台联动结构
struct SDK_PtzLinkConfig
{
  int iType;       // 联动的类型
  int iValue;      // 联动的类型对应的值
};

// 联动操作
struct SDK_EventHandler
{
  unsigned int     dwRecord;              // 录象掩码
  unsigned int     iRecordLatch;          // 录像延时：10～300 sec
  unsigned int     dwTour;                    // 轮巡掩码
  unsigned int     dwSnapShot;                // 抓图掩码
  unsigned int     dwAlarmOut;                // 报警输出通道掩码
```

```
    unsigned int     dwMatrix;              // 矩阵掩码
    int         iEventLatch;            // 联动开始延时时间，s 为单位
    int         iAOLatch;               // 报警输出延时：10～300 sec
    SDK_PtzLinkConfig PtzLink[NET_MAX_CHANNUM];      // 云台联动项
    SDK_CONFIG_WORKSHEET schedule;      // 录像时间段
    bool    bRecordEn;                  // 录像使能
    bool    bTourEn;                // 轮巡使能
    bool    bSnapEn;                // 抓图使能
    bool    bAlarmOutEn;            // 报警使能
    bool    bPtzEn;                     // 云台联动使能
    bool    bTip;                       // 屏幕提示使能
    bool    bMail;                      // 发送邮件
    bool    bMessage;               // 发送消息到报警中心
    bool    bBeep;                      // 蜂鸣
    bool    bVoice;                     // 语音提示
    bool    bFTP;                       // 启动 FTP 传输
    bool    bMatrixEn;                  // 矩阵使能
    bool    bLog;                       // 日志使能，目前只有在 WTN 动态检测中使用
    bool    bMessagetoNet;              // 消息上传给网络使能
};

///< 遮挡检测配置
struct SDK_BLINDDETECTCONFIG
{
    bool bEnable;       ///< 遮挡检测开启
    int       iLevel;           ///< 灵敏度：～
    SDK_EventHandler hEvent;    ///< 遮挡检测联动
};

///< 报警输入配置
struct SDK_ALARM_INPUTCONFIG
{
    bool bEnable;       ///< 报警输入开关
    int       iSensorType;  ///< 传感器类型常开or 常闭
    SDK_EventHandler hEvent;    ///< 报警联动
};
///< 所有通道的报警输入配置
struct SDK_ALARM_INPUTCONFIG_ALL
{
    SDK_ALARM_INPUTCONFIG vAlarmConfigAll[NET_MAX_CHANNUM];
};

/// 全通道遮挡检测配置
```

```
struct SDK_BLINDDETECTCONFIG_ALL
{
    SDK_BLINDDETECTCONFIG vBlindDetectAll[NET_MAX_CHANNUM];
};
///< 动态检测设置
struct SDK_MOTIONCONFIG
{
    bool bEnable;                               // 动态检测开启
    int iLevel;                                 // 灵敏度
    unsigned int mRegion[NET_MD_REGION_ROW];            // 区域，每一行使用一个二进制串
    SDK_EventHandler hEvent;                    // 动态检测联动
};
///< 视频丢失
struct SDK_VIDEOLOSSCONFIG
{
    bool bEnable;           ///< 使能
    SDK_EventHandler hEvent;    ///< 处理参数
};
/// 所有通道的视频丢失结构
struct SDK_VIDEOLOSSCONFIG_ALL
{
    SDK_VIDEOLOSSCONFIG vGenericEventConfig[NET_MAX_CHANNUM];
};

/// 录像模式种类
enum SDK_RecordModeTypes
{
    SDK_RECORD_MODE_CLOSED,     ///< 关闭录像
    SDK_RECORD_MODE_MANUAL,     ///< 手动录像
    SDK_RECORD_MODE_CONFIG,     ///< 按配置录像
    SDK_RECORD_MODE_NR,
};

///< 录像设置
struct SDK_RECORDCONFIG
{
    int iPreRecord;             ///< 预录时间，为零时表示关闭
    bool bRedundancy;       ///< 冗余开关
    bool bSnapShot;             ///< 快照开关
    int iPacketLength;      ///< 录像打包长度（分钟）[1, 255]
    int iRecordMode;        ///< 录像模式，见SDK_RecordModeTypes
    SDK_CONFIG_WORKSHEET wcWorkSheet;               ///< 录像时间段
    unsigned int typeMask[NET_N_WEEKS][NET_N_TSECT];        ///< 录像类型掩码
```

```
};

//录像设置结构体
struct SDK_RECORDCONFIG_ALL
{
    SDK_RECORDCONFIG vRecordConfigAll[NET_MAX_CHANNUM];
};

//普通配置页结构体
typedef struct _SDK_CONFIG_NORMAL
{
    NEW_NET_TIME sysTime;       ///< 系统时间
    int iLocalNo;              ///< 本机编号:[0, 998]
    int iOverWrite;             ///< 硬盘满时处理 "OverWrite", "StopRecord"
    int iSnapInterval;          ///< 定时抓图的时间间隔，以秒为单位
    char sMachineName[64]; ///< 机器名
    int iVideoStartOutPut; ///< 输出模式*/
    int iAutoLogout;            ///< 本地菜单自动注销(分钟) [0, 120]
    int iVideoFormat;       ///< 视频制式:“PAL”， “NTSC”， “SECAM”
    int iLanguage;              ///< 语言选择:“English”, “SimpChinese”, “TradChinese”, “Italian”,
“Spanish”， “Japanese”， “Russian”， “French”， “German”
    int iDateFormat;        ///< 日期格式:“YYMMDD”， “MMDDYY”， “DDMMYY”
    int iDateSeparator;         ///< 日期分割符:“.”， “-”， “/”
    int iTimeFormat;        ///< 时间格式:“12”， “24”
    int iDSTRule;           ///< 夏令时规则
    int iWorkDay;           ///< 工作日
    DSTPoint dDSTStart;
    DSTPoint dDSTEnd;
}SDK_CONFIG_NORMAL;

// 编码设置
struct SDK_CONFIG_ENCODE
{
    SDK_MEDIA_FORMAT dstMainFmt[SDK_ENCODE_TYPE_NUM];       //  主码流格式
    SDK_MEDIA_FORMAT dstExtraFmt[SDK_EXTRATYPES]; //  辅码流格式
    SDK_MEDIA_FORMAT dstSnapFmt[SDK_ENCODE_TYPE_NUM];       //  抓图格式
};
struct  SDK_EncodeConfigAll
{
    SDK_CONFIG_ENCODE vEncodeConfigAll[NET_MAX_CHANNUM];
};

// 组合编码设置
```

```
struct  SDK_CombineEncodeConfigAll
{
    SDK_CONFIG_ENCODE vEncodeConfigAll[NET_MAX_COMBINE_NUM];
};
////!普通网络设置
struct SDK_CONFIG_NET_COMMON
{
    char HostName[NET_NAME_PASSWORD_LEN]; //!主机名
    CONFIG_IPAddress HostIP;              //!主机IP
    CONFIG_IPAddress Submask;             //!子网掩码
    CONFIG_IPAddress Gateway;             //!网关IP
    int HttpPort;                         //!HTTP服务端口
    int TCPPort;                          //!TCP侦听端口
    int SSLPort;                          //!SSL侦听端口
    int UDPPort;                          //!UDP侦听端口
    int MaxConn;                          //!最大连接数
    int MonMode;                          //!监视协议{"TCP","UDP","MCAST",…}
    int MaxBps;                       //!限定码流值
    int TransferPlan;                 //!传输策略//char TransferPlan[NET_NAME_PASSWORD_LEN];
    bool bUseHSDownLoad;                  //!是否启用高速录像下载
};


// 云台设置
struct SDK_STR_CONFIG_PTZ
{
    char sProtocolName[NET_MAX_PTZ_PROTOCOL_LENGTH];    // 协议名称
    int  ideviceNo;                   // 云台设备地址编号
    int  iNumberInMatrixs;       // 在矩阵中的统一编号
    int iPortNo;                 // 串口端口号 [1, 4]
    SDK_COMMATTRI dstComm;            // 串口属性
};
//所有通道云台协议
struct SDK_STR_PTZCONFIG_ALL
{
    SDK_STR_CONFIG_PTZ ptzAll[NET_MAX_CHANNUM];
};
// 串口配置
struct SDK_CONFIG_COMM_X
{
    char iProtocolName[32];// 串口协议:“Console”
    int iPortNo;       // 端口号
    SDK_COMMATTRI aCommAttri;         // 串口属性
```

```
};
struct SDK_CommConfigAll
{
    SDK_CONFIG_COMM_X vCommConfig[SDK_COM_TYPES];
};


///< IP权限设置
struct SDK_NetIPFilterConfig
{
    bool Enable;      ///< 是否开启
    CONFIG_IPAddress BannedList[NET_MAX_FILTERIP_NUM];      ///< 黑名单列表
    CONFIG_IPAddress TrustList[NET_MAX_FILTERIP_NUM];       ///< 白名单列表
};


///< 组播设置
struct SDK_NetMultiCastConfig
{
    bool Enable;      ///< 是否开启
    SDK_RemoteServerConfig Server;       ///< 组播服务器
};

///< pppoe设置
struct SDK_NetPPPoEConfig
{
    bool Enable;  ///< 是否开启
    SDK_RemoteServerConfig Server;       ///< PPPOE服务器
    CONFIG_IPAddress addr;      ///< 拨号后获得的IP地址
};

///< DDNS设置
struct SDK_NetDDNSConfig
{
    bool Enable;  ///< 是否开启
    char DDNSKey[NET_NAME_PASSWORD_LEN]; ///< DDNS类型名称
    char HostName[NET_NAME_PASSWORD_LEN];///< 主机名
    SDK_RemoteServerConfig Server;           ///< DDNS服务器
};

///< DDNS设置
struct SDK_NetDDNSConfigALL
{
```

```
    SDK_NetDDNSConfig ddnsConfig[5];
};

///< ftp设置
struct SDK_FtpServerConfig {
    ///< 服务器使能
    bool Enable;
    ///< FTP服务器
    SDK_RemoteServerConfig Server;
    ///< 备用服务器IP
    CONFIG_IPAddress SpareIP;
    ///< 远程目录
    char RemotePathName[NET_MAX_PATH_LENGTH];
    ///< 文件最大长度
    int  FileMaxLen;
    ///< 上传时段
    SDK_TIMESECTION  UpLoadPeriod[NET_N_MIN_TSECT];
};


///< NTP设置
struct SDK_NetNTPConfig
{
    ///< 是否开启
    bool Enable;
    ///< NTP服务器
    SDK_RemoteServerConfig Server;
    ///< 更新周期
    int UpdatePeriod;
    ///< 时区
    int TimeZone;
};
#define  NET_MAX_EMAIL_TITLE_LEN 64
#define  NET_MAX_EMAIL_RECIEVERS  5
#define  NET_EMAIL_ADDR_LEN  32

///< EMAIL设置
struct SDK_NetEmailConfig
{
    ///< 是否开启
    bool Enable;
    ///< smtp 服务器地址使用字符串形式填充
    ///< 可以填ip,也可以填域名
```

```
    SDK_RemoteServerConfig Server;
    bool bUseSSL;
    ///< 发送地址
    char SendAddr[NET_EMAIL_ADDR_LEN];
    ///< 接收人地址
    char Recievers[NET_MAX_EMAIL_RECIEVERS][NET_EMAIL_ADDR_LEN];
    ///< 邮件主题
    char Title[NET_MAX_EMAIL_TITLE_LEN];
    ///< email有效时间
    SDK_TIMESECTION Schedule[NET_N_MIN_TSECT];
};


///< DNS设置
struct SDK_NetDNSConfig
{
    CONFIG_IPAddress        PrimaryDNS;
    CONFIG_IPAddress        SecondaryDNS;
};


/// 音频输入格式，语音对讲用
struct SDK_AudioInFormatConfig
{
    int iBitRate;     ///< 码流大小，kbps为单位
    int iSampleRate;    ///< 采样率，Hz为单位
    int iSampleBit;     ///< 采样的位深
    int iEncodeType;    ///< 编码方式，参照AudioEncodeTypes定义
};

/// 告警状态
struct SDK_DVR_ALARMSTATE
{
    int iVideoMotion; ///< 移动侦测状态,用掩码表示通道号,bit0代表通道一,以此类推1：有告警0：无告警
    int iVideoBlind; ///< 视频遮挡状态,用掩码表示通道号,bit0代表通道一,以此类推1：有告警0：无告警
    int iVideoLoss;     ///< 视频丢失状态,用掩码表示通道号,bit0代表通道一,以此类推1：有告警0：无告警
    int iAlarmIn; ///< 告警输入状态,用掩码表示通道号,bit0代表通道一,以此类推1：有告警0：无告警
    int iAlarmOut;      ///< 告警输出状态,用掩码表示通道号,bit0代表通道一,以此类推1：有告警0：无告警
};

// 通道状态
struct SDK_DVR_CHANNELSTATE
{
```

```
    bool bRecord; ///< 是否正在录像
    int iBitrate; ///< 当前码率
};

// 设备工作状态信息
struct SDK_DVR_WORKSTATE
{
    SDK_DVR_CHANNELSTATE vChnState[NET_MAX_CHANNUM];
    SDK_DVR_ALARMSTATE vAlarmState;
};
```

2.1.5 网络键盘键值定义

```
/// 按键值
enum SDK_NetKeyBoardValue
{
    SDK_NET_KEY_0, SDK_NET_KEY_1, SDK_NET_KEY_2, SDK_NET_KEY_3, SDK_NET_KEY_4, SDK_NET_KEY_5,
SDK_NET_KEY_6, SDK_NET_KEY_7, SDK_NET_KEY_8, SDK_NET_KEY_9,
    SDK_NET_KEY_10, SDK_NET_KEY_11, SDK_NET_KEY_12, SDK_NET_KEY_13, SDK_NET_KEY_14, SDK_NET_KEY_15,
SDK_NET_KEY_16, SDK_NET_KEY_10PLUS,
    SDK_NET_KEY_UP = 20,      // 上或者云台向上
    SDK_NET_KEY_DOWN,         // 下或者云台向下
    SDK_NET_KEY_LEFT,         // 左或者云台向左
    SDK_NET_KEY_RIGHT,        // 右或者云台向右
    SDK_NET_KEY_SHIFT,
    SDK_NET_KEY_PGUP,         // 上一页
    SDK_NET_KEY_PGDN,         // 下一页
    SDK_NET_KEY_RET,          // 确认
    SDK_NET_KEY_ESC,          // 取消或退出
    SDK_NET_KEY_FUNC,         // 切换输入法
    SDK_NET_KEY_PLAY,         // 播放/暂停
    SDK_NET_KEY_BACK,         // 倒放
    SDK_NET_KEY_STOP,         // 停止
    SDK_NET_KEY_FAST,         // 快放
    SDK_NET_KEY_SLOW,         // 慢放
    SDK_NET_KEY_NEXT,         // 下一个文件
    SDK_NET_KEY_PREV,         // 上一个文件
    SDK_NET_KEY_REC = 40,     // 录像设置
    SDK_NET_KEY_SEARCH,       // 录像查询
    SDK_NET_KEY_INFO,         // 系统信息
    SDK_NET_KEY_ALARM,        // 告警输出
    SDK_NET_KEY_ADDR,         // 遥控器地址设置
```

```
    SDK_NET_KEY_BACKUP,       // 备份
    SDK_NET_KEY_SPLIT,        // 画面分割模式切换，每按一次切换到下一个风格模式
    SDK_NET_KEY_SPLIT1,       // 单画面
    SDK_NET_KEY_SPLIT4,       // 四画面
    SDK_NET_KEY_SPLIT8,       // 八画面
    SDK_NET_KEY_SPLIT9,       // 九画面
    SDK_NET_KEY_SPLIT16,      // 16画面
    SDK_NET_KEY_SHUT,         // 关机
    SDK_NET_KEY_MENU,         // 菜单
    SDK_NET_KEY_PTZ = 60,     // 进入云台控制模式
    SDK_NET_KEY_TELE,         // 变倍减
    SDK_NET_KEY_WIDE,         // 变倍加
    SDK_NET_KEY_IRIS_SMALL,   // 光圈增
    SDK_NET_KEY_IRIS_LARGE,   // 光圈减
    SDK_NET_KEY_FOCUS_NEAR,   // 聚焦远
    SDK_NET_KEY_FOCUS_FAR,    // 聚焦近
    SDK_NET_KEY_BRUSH,        // 雨刷
    SDK_NET_KEY_LIGHT,        // 灯光
    SDK_NET_KEY_SPRESET,      // 设置预置点
    SDK_NET_KEY_GPRESET,      // 转至预置点
    SDK_NET_KEY_DPRESET,      // 清除预置点
    SDK_NET_KEY_PATTERN,      // 模式
    SDK_NET_KEY_SCANON,       // 自动扫描开始
    SDK_NET_KEY_SCANOFF,      // 自动扫描结束
    SDK_NET_KEY_AUTOTOUR,     // 自动巡航
    SDK_NET_KEY_AUTOPANON,    // 线扫开始
    SDK_NET_KEY_AUTOPANOFF,   // 线扫结束
};

/// 按键状态
enum SDK_NetKeyBoardState
{
    SDK_NET_KEYBOARD_KEYDOWN,    // 按键按下
    SDK_NET_KEYBOARD_KEYUP,      // 按键松开
};

struct SDK_NetKeyBoardData
{
    int iValue;                  // 见 SDK_NetKeyBoardValue
    int iState;                  // 见 SDK_NetKeyBoardState
};
```

3 接口定义

3.1 SDK 初始化

1. `H264_DVR_API long H264_DVR_GetLastError();;`

- 函数说明：返回函数执行失败代码，当调用下面的接口失败时，可以用该函数获取失败的代码，具体错误代码参见*错误类型代号说明*
- 参数说明：
- 返回值：返回
- 相关函数：

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

2. `H264_DVR_API long H264_DVR_Init(fDisConnect cbDisConnect, unsigned long dwUser);`

- 函数说明：初始化 SDK, 在所有的 SDK 函数之前调用
- 参数说明：

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 0 时禁止回调

[in]dwUser

用户数据

回调函数参数说明：

lLoginID

`H264_DVR_Login` 的返回值

pchDVRIP

设备 IP

nDVRPort

端口

dwUser

用户数据，就是上面输入的用户数据

- 返回值：成功返回 TRUE，不成功返回 FALSE
- 相关函数：`H264_DVR_Cleanup`

3. `CLIENT_API void H264_DVR_Cleanup ();`

- 函数说明：清空 SDK，释放占用的资源，在所有的 SDK 函数之后调用。
- 参数：无
- 返回值：无
- 相关函数：H264_DVR_Init
- 典型应用：在应用程序关闭时调用

3.2 报警状态获取

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

- H264_DVR_API bool H264_DVR_SetDVRMessCallBack(fMessCallBack cbAlarmcallback, unsigned long lUser);
- 函数说明：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消息订阅接口 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
- 参数说明：

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

回调函数参数说明：

lLoginID

H264_DVR_Login 的返回值

pBuf

具体信心见 SDK_AlarmInfo

dwBufLen

pBuf 的长度，(单位字节)

dwUser

回调的用户数据，就是上面输入的用户数据

- 返回值：TRUE 回调函数执行正确，FALSE 执行错误
- 相关函数：H264_DVR_SetupAlarmChan、H264_DVR_CloseAlarmChan

4. H264_DVR_API long H264_DVR_SetupAlarmChan(long lLoginID);

- 函数说明：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。
- 参数说明：

[in]lLoginID

H264_DVR_Login 的返回值

- 返回值：成功返回 TRUE，失败返回 FALSE
- 相关函数：H264_DVR_SetDVRMessCallBack，H264_DVR_CloseAlarmChan

典型应用：在设备连接后调用本函数进行消息订阅.

5. H264_DVR_API bool H264_DVR_CloseAlarmChan(long lLoginID);

- 函数说明：停止对某个设备侦听消息
- 参数说明：

[in]lLoginID

H264_DVR_Login 返回值

- 返回值：成功返回 TRUE，失败返回 FALSE
- 相关函数：H264_DVR_SetupAlarmChan
- 典型应用：参见 demo 程序

3.3 设备注册

6. H264_DVR_API long H264_DVR_Login (char *sDVRIP, unsigned short wDVRPort, char *sUserName, char *sPassword, LPH264_DVR_DEVICEINFO lpDeviceInfo, int *error);

- 函数说明：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。
- 参数说明：

[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
(当函数返回成功时,该参数的值无意义)，返回登录错误码：

- 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。
- 相关函数：H264_DVR_Logout
- 典型应用：在初始化后就可以调用本接口注册到指定的设备，成功后将返回设备句柄，给相关的函数调用

7. H264_DVR_API long H264_DVR_LoginEx(char *sDVRIP, unsigned short wDVRPort, char *sUserName, char *sPassword, LPH264_DVR_DEVICEINFO lpDeviceInfo, int nType, int *error);

- 函数说明：注册用户到设备的扩展接口，支持一个用户指定登陆的客户端类型
- 参数说明：增加扩展参数

[in] nType
设备支持的能力，值为 2 表示主动侦听模式下的用户登陆。（车载 dvr 登录）

```
enum LoginType
{
    LOGIN_TYPE_GUI,          ///< 本地GUI登陆
    LOGIN_TYPE_CONSOLE,      ///< 控制台登陆
    LOGIN_TYPE_WEB,          ///< WEB登陆
    LOGIN_TYPE_SNS,          ///< SNS登陆
    LOGIN_TYPE_MOBIL,        ///< 移动终端登陆
```

```
        LOGIN_TYPE_NETKEYBOARD, ///< 网络键盘登陆
        LOGIN_TYPE_SERVER,      ///< 中心服务器登陆
        LOGIN_TYPE_AUTOSEARCH,  ///< IP自动搜索工具登陆
        LOGIN_TYPE_UPGRADE,     ///< 升级工具登陆
        LOGIN_TYPE_MEGAEYE,     ///< 全球眼登陆
        LOGIN_TYPE_NR,          ///< 登陆类型
    };
```

- 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
- 相关函数：H264_DVR_Logout
- 典型应用：升级工具等的登陆。

8. `H264_DVR_API long H264_DVR_Logout(long lLoginID)`

- 函数说明：注销设备用户
- 参数说明：

  [in]lLoginID

  H264_DVR_Login 的返回值
- 返回值：成功返回 TRUE，失败返回 FALSE
- 相关函数：H264_DVR_Login
- 典型应用：当需要设备主动断开时调用;

3.4 实时监视

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

- 函数说明：启动实时监视
- 参数说明：

  [in]lLoginID

  H264_DVR_Login 的返回值

  [in] lpClientInfo

  实时监视参数
- 返回值：失败返回 0，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。
- 相关函数：H264_DVR_StopRealPlay，H264_DVR_SetRealDataCallBack
- 典型应用：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 ，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;

10. `H264_DVR_API bool H264_DVR_StopRealPlay(long lRealHandle);`

- 函数说明：停止实时监视
- 参数说明：

  [in]lRealHandle

  H264_DVR_RealPlay 的返回值

- 返回值：成功返回 TRUE，失败返回 FALSE
- 相关函数：H264_DVR_RealPlay
- 典型应用：关闭实时监视

```
//原始数据回调原形
   typedef  int(__stdcall  *fRealDataCallBack)  (long  lRealHandle,  long
   dwDataType, unsigned char *pBuffer,long lbufsize,long dwUser);
```

11. `H264_DVR_API bool H264_DVR_SetRealDataCallBack(long lRealHandle,fRealDataCallBack cbRealData, long dwUser);`

- 函数说明：设置实时监视数据回调，给用户提供设备流出的数据
- 参数说明：

  [in]lRealHandle

  H264_DVR_RealPlay 的返回值

  cbRealData

  回调函数,用于传出设备流出的实时数据

  [in]dwUser

  用户数据

回调函数参数说明：

lRealHandle

H264_DVR_RealPlay 的返回值

dwDataType

暂时可以不需要判断

pBuffer

回调数据,根据数据类型的不同每次回调不同的长度的数据,除类型 0, 其他数据类型都是按帧,每次回调一帧数据,

dwBufSize

回调数据的长度, (单位字节).

dwUser

用户数据，就是上面输入的用户数据

- 返回值：成功返回 TRUE，失败返回 FALSE
- 相关函数：H264_DVR_RealPlay、H264_DVR_StopRealPlay

3.5 回放和下载

12. `H264_DVR_API long H264_DVR_FindFile(long lLoginID, H264_DVR_FINDINFO* lpFindInfo, H264_DVR_FILE_DATA *lpFileData, int lMaxCount, int *findcount, int waittime = 2000);`

- 函数说明：查询录像文件
- 参数说明：

  [in]lLoginID

H264_DVR_Login 的返回值

[in] lpFindInfo

查询条件 H264_DVR_FINDINFO

[out]nriFileinfo

返回的录像文件信息，是一个 H264_DVR_FILE_DATA 结构数组

[in]maxlen

nriFileinfo 缓冲的最大长度;（单位字节，建议在

100-200*sizeof(H264_DVR_FILE_DATA)之间）

[out]filecount

返回的文件个数,属于输出参数最大只能查到缓冲满为止的录像记录;

[in]waittime

等待时间

- 返回值：成功返回 TRUE，失败返回 FALSE
- 相关函数：H264_DVR_Login、H264_DVR_PlayBackByName、H264_DVR_StopPlayBack，H264_DVR_PlayBackControl，H264_DVR_GetFileByName
- 典型应用：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

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

13. H264_DVR_API long H264_DVR_PlayBackByName(long lLoginID, H264_DVR_FILE_DATA *sPlayBackFile, fDownLoadPosCallBack cbDownLoadPos, fRealDataCallBack fDownLoadDataCallBack, long dwDataUser);

- 函数说明：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。
- 参数说明：

[in]lLoginID

H264_DVR_Login 的返回值

[in] sPlayBackFile

录像文件信息，由 H264_DVR_FindFile 返回

[in] cbDownLoadPos

进度回调函数

[in] fDownLoadDataCallBack

原始数据回调函数

[in]dwUserData

用户自定义数据

回调函数说明：

lPlayHandle

H264_DVR_PlayBackByName 的返回值

dwTotalSize

指本次播放总大小，单位为 KB

dwDownLoadSize

指已经播放的大小，单位为 KB，当其值为-1 时表示本次回放结束

dwUser

用户数据，就是上面输入的用户数据

- 返回值：成功返回网络回放 ID，失败返回 0
- 相关函数:H264_DVR_Login、H264_DVR_FindFile，H264_DVR_StopPlayBack，H264_DVR_PlayBackControl

14. H264_DVR_API bool H264_DVR_StopPlayBack(long lPlayHandle);

- 函数说明：网络回放停止
- 参数说明：

[in]lPlayHandle

回放句柄,如 H264_DVR_PlayBackByName 的返回值

- 返回值：成功返回 TRUE，失败返回 FALSE
- 相关函数：H264_DVR_PlayBackByName
- 典型应用：输入上一接口返回的播放 ID,调用本接口就可以停止控制。

15. H264_DVR_API long H264_DVR_GetFileByName(long lLoginID,H264_DVR_FILE_DATA *sPlayBackFile,char *sSavedFileName, fDownLoadPosCallBack cbDownLoadPos = NULL, long dwDataUser = NULL);

- 函数说明：按文件下载录像文件，通过查询到的文件信息下载
- 参数说明：

[in]lLoginID

H264_DVR_Login 的返回值

[in] sPlayBackFile

录像文件信息指针

[in]sSavedFileName

要保存的录像文件名，全路径

cbDownLoadPos

下载进度回调函数，可以为空，用户自己调用 H264_DVR_GetDownloadPos 得到进度

[in]dwUserData

下载进度回调用户自定义数据

下载进度回调函数参数说明：参见 H264_DVR_PlayBackByName

- 返回值：成功返回下载 ID，失败返回 0
- 相关函数：H264_DVR_StopGetFile、H264_DVR_GetDownloadPos
- 典型应用：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

16. H264_DVR_API bool H264_DVR_StopGetFile(long lFileHandle);

- 函数说明：停止下载录像文件
- 参数说明：

[in]lFileHandle

H264_DVR_GetFileByName 的返回值

- 返回值：成功返回 TRUE，失败返回 FALSE
- 相关函数：H264_DVR_GetFileByName、H264_DVR_GetDownloadPos
- 典型应用：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;

17. H264_DVR_API int H264_DVR_GetDownloadPos(long lFileHandle);

- 函数说明：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
- 参数说明：

[in]lFileHandle

H264_DVR_GetFileByName 的返回值

- 返回值：成功返回 pos(百分比)
- 相关函数：H264_DVR_GetFileByName、H264_DVR_StopGetFile
- 典型应用：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;

3.6 回放控制

18. H264_DVR_API bool H264_DVR_PlayBackControl(long lPlayHandle, long lControlCode,long lCtrlValue);

- 函数说明：网络回放暂停与恢复以及进度控制
- 参数说明：

[in]lPlayHandle

回放句柄,如 H264_DVR_GetFileByName 的返回值

[in] lControlCode

控制类型

```
enum SEDK_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*/
    };
```

- 返回值：成功返回 TRUE，失败返回 FALSE
- 相关函数：H264_DVR_PlayBackByName、H264_DVR_StopPlayBack
- 典型应用：对已经打开的播放进行暂停和恢复控制

3.7 云台控制

19. `H264_DVR_API bool H264_DVR_PTZControl(long lLoginID,int nChannelNo, long lPTZCommand, bool bStop = false, long lSpeed = 4)`

- 函数说明：云台控制
- 参数说明:

  [in]lLoginID

  H264_DVR_Login 的返回值

  [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。
- 相关函数：H264_DVR_Login，H264_DVR_RealPlay
- 典型应用：控制云台，但是必须在当前通道打开的情况下使用。

3.8 系统配置

20. `H264_DVR_API long H264_DVR_GetDevConfig(long lLoginID, unsigned long dwCommand, char * lpOutBuffer, unsigned long dwOutBufferSize, unsigned long* lpBytesReturned,int waittime = 1000);`

- 函数说明: 获取设备配置。
- 参数说明 :

  [in]lLoginID

H264_DVR_Login 的返回值

[in]dwCommand

配置类型 具体定义见数据结构定义中的SDK_CONFIG_TYPE

[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：TRUE 表示成功，FALSE 表示失败。

21. H264_DVR_API long H264_DVR_SetDevConfig(long lLoginID, unsigned long dwCommand, char * lpInBuffer, unsigned long dwInBufferSize, int waittime = 1000);

- 函数说明：获取设备配置。
- 参数说明 ：

[in]lLoginID

H264_DVR_Login 的返回值

[in]dwCommand

配置类型 具体定义见数据结构定义中的SDK_CONFIG_TYPE

[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：TRUE 表示成功，FALSE 表示失败。

3.9 日志管理

22. H264_DVR_API bool H264_DVR_FindDVRLog(long lLoginID, SDK_LogSearchCondition *pFindParam, SDK_LogList *pRetBuffer, long lBufSize, int waittime = 2000);

- 函数说明：查询日志
- 参数说明：

[in]lLoginID

H264_DVR_Login 的返回值

[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

3.10 远程控制

23. H264_DVR_API bool H264_DVR_ControlDVR(long lLoginID, int type, int waittime = 2000)

- 函数说明：重启和清除日志
- 参数说明：
  [in]lLoginID
  H264_DVR_Login 的返回值
  [in] type
  0 重启设备，1 清除日志
- 返回值：成功返回 TRUE，失败返回 FALSE
- 相关函数：

```
typedef void(__stdcall *fUpgradeCallBack) (long lLoginID, long
  lUpgradechannel, int nTotalSize, int nSendSize, long dwUser);
```

24. `H264_DVR_API long H264_DVR_Upgrade(long lLoginID, char *sFileName, int nType = 0, fUpgradeCallBack cbUpgrade = NULL, long dwUser = 0);;`

- 函数说明：设置对前端设备网络升级程序
- 参数说明：

[in]lLoginID

H264_DVR_Login 的返回值

[in] sFileName

要升级的文件名

[in] nType

要升级的文件类型

```
enum UpgradeTypes
{
    UPGRADE_TYPES_SYSTEM,   ///< 升级系统
    UPGRADE_TYPES_NR,
};
```

//回调说明

fUpgradeCallBack

回调升级进度,其中 lUpgradechannel 为升级句柄

nTotalSize

为升级文件的总长度,(单位字节)

nSendSize

为已升级的长度,(单位字节)

[in]dwUser

用户数据

- 返回值：成功返回升级句柄 ID，失败返回 0
- 相关函数：H264_DVR_GetUpgradeState，H264_DVR_CloseUpgradeHandle
- 典型应用：设置远程程序的升级,返回程序升级句柄

25. `H264_DVR_API long H264_DVR_CloseUpgradeHandle(long lUpgradeHandle);`

- 函数说明：停止升级
- 参数说明：

[in]lUpgradeID

升级句柄 ID

- 返回值：成功返回 TRUE，失败返回 FALSE
- 相关函数：H264_DVR_Upgrade
- 典型应用：停止升级

26. `H264_DVR_API int H264_DVR_GetUpgradeState(long lUpgradeHandle)`

- 函数说明：获取升级状态
- 参数说明：

  0- *[in]lUpgradeID*

  升级句柄 ID

  返回值：1 成功，2 正在升级 3 失败
- 相关函数: H264_DVR_Upgrade，H264_DVR_CloseUpgradeHand

27. `H264_DVR_API bool H264_DVR_SearchDevice(char* szBuf, int nBufLen, int* pRetLen, int nSearchTime);`

- 函数说明：搜索局域网内设备信息
- 参数说明：

  0- *[in]* szBuf

  接收搜索到的设备信息缓冲

  1- *[in]* nBufLen

  接收搜索到的设备信息缓冲的长度

  2- *[in]* pRetLen

  实际搜索到的设备信息的长度，用来判断缓冲大小是否合适

  3- *[in]* nSearchTime

  指定设备搜索的总时间，超过时间则认为搜索失败

  返回值：1 成功，0 失败
- 相关函数:

3.11 语音对讲

```
// 语音对讲的音频数据回调函数原形
typedef void (__stdcall *pfAudioDataCallBack)(long lVoiceHandle, char *pDataBuf,
                                              long dwBufSize, char byAudioFlag, long dwUser);
```

28. `H264_DVR_API long H264_DVR_StartVoiceCom_MR (long lLoginID, pfAudioDataCallBack pVcb, Long dwDataUser);`

- 函数说明：开启语音对讲，负责数据转发
- 参数说明：

  0- *[in]* lLoginID

  设备登陆句柄：H264_DVR_Login 返回值

  1- *[in]* pVcb

  从设备接收到的语音对讲数据回调

  2- *[in]* dwDataUser

  接收数据对象

  返回值：> 0 对讲句柄 <= 0 失败
- 相关函数:

  H264_DVR_VoiceComSendData H264_DVR_StopVoiceCom H264_DVR_SetTalkMode

29. H264_DVR_API bool H264_DVR_VoiceComSendData (long lVoiceHandle, char *pSendBuf, long lBufSize);

- 函数说明：转发 PC 采集到的语音对讲数据
- 参数说明：

  0- *[in]* lVoiceHandle

  对讲句柄：H264_DVR_StartVoiceCom_MR 返回值

  1- *[in]* pSendBuf

  从 PC 采集到的音频数据

  2- *[in]* lBufSize

  音频数据长度
- 返回值：1 成功 0 失败
- 相关函数：

  H264_DVR_StartVoiceCom_MR H264_DVR_StopVoiceCom H264_DVR_SetTalkMode

30. H264_DVR_API bool H264_DVR_StopVoiceCom (long lVoiceHandle);

- 函数说明：停止语音对讲
- 参数说明：

0- *[in]* lVoiceHandle

对讲句柄：H264_DVR_StartVoiceCom_MR 返回值

- 返回值：1 成功　0 失败
- 相关函数：

H264_DVR_StartVoiceCom_MR　H264_DVR_VoiceComSendData　H264_DVR_SetTalkMode

31. H264_DVR_API bool H264_DVR_SetTalkMode (long lLoginID, SDK_AudioInFormatConfig* pTalkMode);

- 函数说明：指定设备的语音对讲模式
- 参数说明：

0- *[in]* lLoginID

设备登陆句柄：H264_DVR_Login 返回值

1- *[in]* pTalkMode

对讲模式结构体 参见：SDK_AudioInFormatConfig

- 返回值：1 成功　0 失败
- 相关函数：

H264_DVR_StartVoiceCom_MR　H264_DVR_VoiceComSendData　H264_DVR_StopVoiceCom

3.12 录像模式设置

32. H264_DVR_API bool H264_DVR_StartDVRRecord(long lLoginID, int nChannelNo ,long lRecordType);

- 函数说明：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
- 参数说明：

0- *[in]* lLoginID

设备登陆句柄：H264_DVR_Login 返回值

1- *[in]* nChannelNo

通道号，-1 所有通道

2- [in] lRecordType

录像类型 参见：SDK_RecordModeTypes

- 返回值：1 成功　0 失败

- 相关函数：

H264_DVR_StopDVRRecord　H264_DVR_GetDevConfig H264_DVR_SetDevConfig

33. H264_DVR_API bool H264_DVR_StopDVRRecord(long lLoginID, int nChannelNo);

- 函数说明：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
- 参数说明：

0- *[in]* lLoginID

设备登陆句柄：H264_DVR_Login 返回值

1- *[in]* nChannelNo

通道号，-1 所有通道

- 返回值：1 成功　0 失败
- 相关函数：

H264_DVR_StartDVRRecord　H264_DVR_GetDevConfig H264_DVR_SetDevConfig

3.13 设置系统时间

34. H264_DVR_API bool H264_DVR_SetSystemDateTime (long lLoginID, SDK_SYSTEM_TIME *pSysTime);

- 函数说明：设置系统时间
- 参数说明：

0- *[in]* lLoginID

设备登陆句柄：H264_DVR_Login 返回值

1- *[in]* pSysTime

系统时间 参见：SDK_SYSTEM_TIME

- 返回值：1 成功　0 失败

3.14 获取设置运行状态信息

35. H264_DVR_API bool H264_DVR_GetDVRWorkState(long lLoginID, SDK_DVR_WORKSTATE *pWorkState);

- 函数说明：获取设备工作状态信息
- 参数说明：

0- *[in]* lLoginID

设备登陆句柄：H264_DVR_Login 返回值

1- *[in]* pWorkState

运行状态 参见：SDK_DVR_WORKSTATE

- 返回值：1 成功　0 失败

3.15 网络键盘

36. H264_DVR_API bool H264_DVR_ClickKey (long lLoginID, SDK_NetKeyBoardData *pKeyBoardData);

- 函数说明：发送网络键盘按键消息
- 参数说明：

0- *[in]* lLoginID

设备登陆句柄：H264_DVR_Login 返回值

1- *[in]* pKeyBoardData

按键信息 参见：SDK_NetKeyBoardData

- 返回值：1 成功　0 失败

4 示例功能实现

请参看 ClientDemo 程序和” DEMO 说明.doc。