LINUX NETSDK说明文档

（V 3. 0）

目录

NETSDK 提供的 API 定义

1. 初始化相关接口

IP_NET_DVR_Init(初始化)

函数原型：
LONG IP_NET_DVR_Init();
功能：
参数化DLL库，在调用所有函数之前要调用一次此函数。
参数：
0：表示成功
ERR_INIT_SOCKET_ERROR：表示初始化Socket失败
注意：
只能调用一次此API，直至调用IP_NET_DVR_Cleanup之后。

IP_NET_DVR_Cleanup(释放资源)

函数原型:
LONG IP_NET_DVR_Cleanup();
功能：
当不再使用DLL库时，调用此函数，用于清理DLL中使用的资源，调用此函数后。
返回值：
返回0表示成功
不会返回错误

注意：
退出应用程序时，请调用此API以便快速退出所有线程。

IP_NET_DVR_GetSDKVersion（获取 NETSDK 版本号）

函数原型：
LONG IP_NET_DVR_GetSDKVersion();
功能：
获取NETSDK的版本号
返回值：
返回32位整型数,如1110050101，前面6位表示发布日期，后面4位表示版本号。如1110050101即表示2011年10月5日发布的1.01版本

IP_NET_DVR_GetSDKBuildVersion（获取 NETSDK 编译版本号）

函数原型：
LONG IP_NET_DVR_GetSDKBuildVersion();
功能：
　　返回NETSDK的编译发布日期。
返回值：
　　返回编译发布日期。
　　如：2012011411表示发布版本的时候是2012年1月14日11点

IP_NET_DVR_SetStatusEventCallBack（设置状态告警事件回调函数）

函数原型：
LONG IP_NET_DVR_SetStatusEventCallBack(StatusEventCallBack fStatusEventCallBack,void * pUser);
功能：
　　设置设备状态变化、或者有接收到事件、告警时的回调函数。
参数：
　　fStatusEventCallBack：回调用函数，声明原型为：typedef LONG(CALLBACK *StatusEventCallBack)(LONG lUser,LONG nStateCode,char *pResponse,void *pUser);
　　其中参数意义为：
 lUser：标识是哪个设备
 nStateCode:事件类型,值有如下:

```
enum enumNetSatateEvent
{
    EVENT_CONNECTING,//连接中
    EVENT_CONNECTOK,//连接成功
    EVENT_CONNECTFAILED,//连接失败
    EVENT_SOCKETERROR,//sock失败
    EVENT_LOGINOK,//登录成功
    EVENT_LOGINFAILED,//登录失败
    EVENT_STARTAUDIOOK,//对讲启动成功
    EVENT_STARTAUDIOFAILED,//对讲启动失败
    EVENT_STOPAUDIOOK,//停止对讲成功
    EVENT_STOPAUDIOFAILED,//停止对讲失败
    EVENT_SENDPTZOK,//发送云台控制命令成功
    EVENT_SENDPTZFAILED,//发送云台控制命令失败
    EVENT_SENDAUXOK,//发送辅助通道成功
    EVENT_SENDAUXFAILED,//发送辅助通道失败
    EVENT_UPLOADOK,//上传文件成功
    EVENT_UPLOADFAILED,//上传文件失败
    EVENT_DOWNLOADOK,//下载成功
    EVENT_DOWNLOADFAILED,//下载失败
    EVENT_REMOVEOK,//删除文件成功
    EVENT_REMOVEFAILED,//删除文件失败
    EVENT_SENDPTZERROR,//云台操作失败
    EVENT_PTZPRESETINFO,//预置点信息
    EVNET_PTZNOPRESETINFO,//没有预置点
    EVENT_PTZALARM,//告警信息
    EVENT_RECVVIDEOPARAM,//视频参数返回
    EVENT_RECVAUDIOPARAM,//音频参数返回
    EVENT_CONNECTRTSPERROR,//请求实时视频失败
  EVENT_CONNECTRTSPOK,//读求媒体流成功
  EVENT_RTSPTHREADEXIT,//媒体流线程退出
```

```
EVENT_URLERROR,//解释连接媒体流URL时失败
EVENT_RECVVIDEOAUDIOPARAM,//收到音视频参数
EVENT_LOGIN_USERERROR,//登录用户或密码错误
EVENT_LOGOUT_FINISH, //登录线程已退出主循环
EVENT_LOGIN_RECONNECT,    //进行重新登录
EVENT_LOGIN_HEARTBEAT_LOST,//心跳丢失
EVENT_STARTAUDIO_ISBUSY,//设备对讲被占用
EVENT_STARTAUDIO_PARAMERROR,//对讲音频参数错误
                EVENT_STARTAUDIO_AUDIODDISABLED,//设备不支持对讲
                EVENT_CONNECT_RTSPSERVER_ERROR=38,    //连接错误，可能是IP或端口错误
EVENT_CREATE_RTSPCLIENT_ERROR, //创建rtsp客户端错误
EVENT_GET_RTSP_CMDOPTION_ERROR, //连接rtsp服务器，返回错误
EVENT_RTSP_AUTHERROR,  //认证失败，可能是用户或密码错误
            };
```

其中nStateCode= EVENT_PTZALARM时，pResponse中的code表示告警类型，定义如下

pResponse为结构体: ALARM_ITEM

```
typedef struct
{
    ALARM_TIME time;
    int code;
    int flag;
    int level;
    char data[128];
}ALARM_ITEM;
```

其中结构体ALARM_TIME为:

```
typedef struct
{
    int year;
    int month;
    int day;
    int wday;
    int hour;
    int minute;
    int second;
}ALARM_TIME;
```

1 : 设备下线
2 : 设备上线
3 : USB接上
4 : USB拔出
5 : SD卡1接上
6 : SD卡1拔出
7 : SD卡2接上
8 : SD卡2拔出
9 : USB存储空间不足
10: SD卡1存储空间不足
11: SD卡2存储空间不足
12: 视频丢失
13: 视频遮挡
14: 图像移动告警
15: IO由高到低触发
16: IO由低到高触发
17: 空间不足告警
18: 录像启动成功
19: 录像结束
20: 录像启动失败

当nStateCode= EVENT_RECVVIDEOPARAM 时pResponse为结构体：

```
typedef struct
```

```
{
  CHAR codec[256];//H264或是MPEG4，等，表示当前视频编码类型
  INT width;      //视频宽
  INT height;     //视频高
  INT colorbits;  //颜色位数，此处无效，仅用于内部
  INT framerate;  //参考帧率
  INT bitrate;    //参考码率
  BYTE vol_data[256];//扩展参数,解码时必要的数据
  INT vol_length;    //扩展参数长度
}VIDEO_PARAM;
```

当nStateCode= EVENT_RECVAUDIOPARAM 时pResponse为结构体：

```
typedef struct
{
  CHAR codec[256];   //音频类型，AAC或G711
  INT samplerate;    //参考比特率
  INT bitspersample; //参考比特率
  INT channels; //声道数量
  INT framerate;
  INT bitrate;
}AUDIO_PARAM;
```

当nStateCode= EVENT_RECVVIDEOAUDIOPARAM 时pResponse为结构体：

```
typedef struct  __STREAM_AV_PARAM
{
    unsigned char ProtocolName[32];  //==AV_FALG
    short  bHaveVideo;//0 表示没有视频参数
    short  bHaveAudio;//0 表示没有音频参数
    VIDEO_PARAM videoParam;//视频参数
    AUDIO_PARAM audioParam;//音频参数
}STREAM_AV_PARAM;
```

其中情况pResponse:相关信息描述
pUser：调用回调时，将为参数void *pUser

返回值：

返回0表示成功
不会返回错误

注意：

此回调函数不是线程安全的，对于线程安全问题，将由回调函数上处理。（即有可能多个线程同时调用此函数）

一般情况下，收到回调状态后，应该PostMessage到主线程，让主线程或专用线程来处理此消息，而不应该在回调登录上进行处理，除非是简单的状态变量设置。否则有可能会造成死锁问题。

IP_NET_DVR_SetAUXResponseCallBack（设置辅助通道消息回调函数）

函数原型：

LONG IP_NET_DVR_SetAUXResponseCallBack(AUXResponseCallBack fAUXCallBack,void * pUser);

功能：

设置辅助通道的有影响消息时的回调函数。

参数：

fAUXCallBack：当收到有辅助通道信息时，将会调用此回调函数。声明为：typedef LONG(CALLBACK *AUXResponseCallBack)(LONG lUser,LONG nType,char *pResponse,void *pUser);

其中，参数意义：

lUser：登陆设备时返回的参数，用于标识来自哪个设备

nType：表示消息类型，nType的最高字节为非0(即nType & 0xff000000!=0)则表示对应于消息执行失败，低24位（即Type=nType & 0xffffff）为真实消息类型，具体意义，见附表2

pUser：用户指针，与设置此回调时的pUser参数相同。

pUser：回调时，会将此指针作为参数void *pUser进行调用回调函数。

返回值：

0：表示成功

不会返回错误

注意：

此回调函数是线程不安全的，对于线程安全问题，将由回调函数上处理。（即有可能多个线程同时调用此函数）

在回调fAUXCallBack上，应该将先简单分析是什么类型，然后通过PostMessage交成主线程或是专用线程进行处理此回调，不应直接在回调上进行处理。否则有可能造成死锁等问题。

IP_NET_DVR_SetLogToFile(控制日志输入)

函数原型：

LONG IP_NET_DVR_SetLogToFile(DWORD bLogEnable,char *strLogDir,BOOL bAutoDel);

功能：

控制日志输出。

参数：

bLogEnable：当为0表示不输出日志，当为1表示输出本地日志，此时strLogDir为目录。

strLogDir：当启用了日志输出时，此值表示为日志文件目录

bAutoDel：非0表示，当日志文件多于10个时，将自动删除日志文件，否则表示不自动删除。(当前不支持此参数，等同于0)

返回值：

返回0表示成功

不会返回错误

IP_NET_DVR_SetAutoReconnect (设置是否自动重连)

函数原型：

LONG IP_NET_DVR_SetAutoReconnect(LONG lUserID,int bAutoReconnect);

功能：

设置登录出错时，是否自动重连。注意，库默认为自动重连。

参数：

lUserID：当为0时，表示新建连接将使用所设置的bAutoReconnect值

bAutoReconnect：0表示不自动重连，否则表示自动重连。

返回值：
0：表示成功
ERR_NOT_FIND_DEVICE:表示未找到lUserID对应的设备

2. 设备搜索发现相关接口

IP_NET_DVR_SetSearchInterval 设置搜索间隔时间

函数原型：
LONG IP_NET_DVR_SetSearchInterval(UINT nBroadcastInterval);

功能：
设置搜索广播包发送的时间间隔。本接口非必须调用接口，默认搜索时间间隔为3000毫秒。

参数：
nBroadcastInterval：搜索广播包发送的间隔时间，单位为毫秒

返回值：
0：表示成功

IP_NET_DVR_StartSearchIPC 开始搜索设备

函数原型：
LONG IP_NET_DVR_StartSearchIPC();

功能：
搜索局域网的设备

参数：
无

返回值：
0表示成功
ERR_START_THREADERROR:启动线程失败

注意：
1， 调用启动搜索IPC后，将要调用IP_NET_DVR_GetSerchIPCCount，用来取得当前搜索到的设备数量，当数据有变量时，再调用IP_NET_DVR_GetIPCInfoXML或IP_NET_DVR_GetIPCInfo进行读取IPC信息。

2，由于使用广播包进行搜索设备，所以有时搜索可能会慢一些，建议搜索时间在5分钟以上。

IP_NET_DVR_StopSearchIPC 停止搜索设备

函数原型：
LONG IP_NET_DVR_StopSearchIPC();

参数：
无

返回值：
0表示成功
不会返回失败

IP_NET_DVR_GetSearchIPCCount 取得当前已搜索的 IPC 数据

函数原型：
LONG IP_NET_DVR_GetSearchIPCCount ();

参数：
无

返回值：
返回当前已成功搜索的数量

IP_NET_DVR_GetIPCInfo 取得 IPC 的信息结构体格式

函数原型：
LONG IP_NET_DVR_GetIPCInfo(LONG index, IPC_ENTRY * pIPCInfo);

功能：
读取搜索到的第index+1个个设备信息

参数：
Index：要读取的第几个设备(从0开始编号),应该小于IP_NET_DVR_GetSearchIPCCount()返回值
pIPCInfo：信息保存指针，参见头文件结构体定义，

返回值：
0表示成功
ERR_PARAM_ERROR:参数错误，可能是pIPCInfo为空，或是index不在有效值内。

IP_NET_DVR_ModifyIPCByIndex 修改 IPC 信息结构体格式

函数原型：
LONG IP_NET_DVR_ModifyIPCByIndex (LONG index, IPC_ENTRY * pIPCInfo);
参数：
Index: 要修改的第几个设备(从0开始编号),应该小于IP_NET_DVR_GetSearchIPCCount()返回值
pIPCInfo:信息保存指针，参见头文件结构体定义，修改前端设备网络地址时，先调用IP_NET_DVR_GetIPCInfo读取该设备整个IPC_ENTRY结构体信息，修改结构体中网络相关字段后再调用本接口。

返回值：

0表示成功
ERR_PARAM_ERROR:参数错误，可能是pIPCInfo为空，或是index不在有效值内。

注意：
由于使用广播包进行修改，所以只能通过一直搜索，然后发现新设备时，通过系列号相同然后判断要关值是否为我们所修改的，如果与我们所修改的相同，则表示修改成功。

IP_NET_DVR_ModifyIPCBySN 修改 IPC 信息结构体格式

函数原型：
```
LONG __ stdcall IP_NET_DVR_ModifyIPCBySN(char *pSN, LANConfig     *p_lanCfg, MediaStreamConfig * pMediaStreamCfg);
```

参数：
pSN: 要修改的前端设备的序列号，一般来说针对已经搜索出来的前端设备进行修改，通过IP_NET_DVR_GetIPCInfo取到的前端设备信息中包含了序列号，该序列号对每个前端设备具有唯一性。
p_lanCfg:网络信息保存指针，参见头文件结构体定义，修改前端设备网络地址时，先调用IP_NET_DVR_GetIPCInfo读取该设备整个IPC_ENTRY结构体信息，修改结构体中网络相关字段后再调用本接口。
pMediaStreamCfg:媒体访问信息保存指针，参见头文件结构体定义，修改时，先调用IP_NET_DVR_GetIPCInfo读取该设备整个IPC_ENTRY结构体信息，修改结构体中streamCfg相关字段后再调用本接口。

返回值：
0,表示成功
ERR_PARAM_ERROR:参数错误，可能是p_lanCfg和pMediaStreamCfg同时为空
ERR_OUTOFF_MEMORY：内存不足

注意：
本接口通过UDP广播包进行修改，一次调用时不排除存在前端设备因为没有收到消息而修改失败的情况。

IP_NET_DVR_SearchIPCReleaseInfo 清空搜索结果

函数原型：
```
LONG IP_NET_DVR_SearchIPCReleaseInfo();
```

参数：
无

返回值：
0,表示成功

IP_NET_DVR_RebootIPCBySN 重启前端设备

函数原型：
```
LONG IP_NET_DVR_RebootIPCBySN(char *pSN);
```

参数：
pSN: 要重启的前端设备的序列号

返回值：

0,表示成功

ERR_PARAM_ERROR:参数错误

ERR_OUTOFF_MEMORY：内存不足

注意：

本接口通过UDP广播包进行修改，一次调用时不排除存在前端设备因为没有收到消息而失败的情况。

3. 登录相关接口

IP_NET_DVR_Login(连接设备)

函数原型：

LONG IP_NET_DVR_Login(char * szDVRIP,WORD wDVRPort,char *sUserName , char *sPassword , LPIP_NET_DVR_DEVICEINFO lpDeviceInfo);

功能：

登录通信通道返回唯一标识，用于后面进行云台控制，设备配置、控制使用。

参数:

szDVRIP：连接的IP地址(如192.168.0.10)

wDVRPort：通信端口（如8091）

sUserName：登录用户名

sPassword：登录密码

lpDeviceInfo：传空即可，保存此参数，主要用于将来扩展及兼容性

返回值：

成功返回非0，失败返回0，返回非0时，将值将会在后面的调用中使用（参数名为lUserID都使用此值）

返回失败可能原因有：

1、 szDVRIP为空

2、 wDVRPort不在1-65535之间（即非法端口）。

3、 此IP和端口已经登录过一次，重复登录。

注意：

同一设备可以登录多次

调用登录后，内部会启用一个线程进行连接、登录。默认情况下连接失败会一直重连，如果用户想自己控制是否要重连，则先调用IP_NET_DVR_SetAutoReconnect(0,0)

登录状态将会通过状态回调函数(IP_NET_DVR_SetStatusEventCallBack设置)通知。

可能返回状态nStateCode事件类型值有如下:

EVENT_CONNECTING,//连接中

EVENT_CONNECTOK,//连接成功,表示连接辅助通过成功，但还没有认证

EVENT_CONNECTFAILED,//连接失败

EVENT_SOCKETERROR,//sock失败

EVENT_LOGINOK,//登录成功,只有到这一步才表示认证成功了

EVENT_LOGINFAILED=5,//登录失败

EVENT_LOGIN_USERERROR=31,//登录用户或密码错误

EVENT_LOGOUT_FINISH, //登录线程已退出主循环

EVENT_LOGIN_RECONNECT, //进行重新登录 EVENT_LOGIN_HEARTBEAT_LOST,//心跳丢失

IP_NET_DVR_Reconnect (重新连接)

函数原型：

```
LONG IP_NET_DVR_Reconnect(LONG lUserID);
```

功能：

重新连接指定设备

参数：

lUserID：设备登录控制句柄

返回值：

0：重连成功（或是当前已经在登录中）
ERR_NOT_FIND_DEVICE：句柄错误
ERR_START_THREADERROR：启动线程失败
ERR_DEV_NOT_LOGIN:还没有调用登录

注意：

只有登录的设备处于退出连接时有效。不允许在状态回调线程里调用此API函数，否则会造成死锁（如果要调用，如果收到EVENT_LOGOUT_FINISH后要调用时，则应该使用另一个线程上调用，可以PostMessage到主线程，让主线程调用此函数）
内部默认会自动重连，除非用户设置不自动重连。

IP_NET_DVR_Logout(断开设备)

函数原型：

LONG IP_NET_DVR_Logout(LONG lUserID);

功能：

注销登录设备

参数：

lUserID：为调用IP_NET_DVR_Login的返回值

返回值：

0：登出成功
ERR_NOT_FIND_DEVICE：句柄错误

注意：

1，一般正常使用过程中，调用了IP_NET_DVR_Login后，应该调用一次IP_NET_DVR_Logout。
2，调用此函数后，会等待所有线程退出，同时会释放相关资源。

IP_NET_DVR_LogoutAll(断开所有设备)

函数原型：

LONG IP_NET_DVR_LogoutAll();

功能：

注销所有登录设备

参数：

无

返回值：
0：登出成功

注意：
如果要注销所有已登录的设备(但保持媒体流连接)，则调用此函数，以加快退出所有线程。

4. 配置获取/设置相关接口

获取配置接口

说明：
配置获取函数调用时仅仅发网络请求，不会阻塞等待结果，通过辅助通道回调函数异步返回结果，需要调用者在IP_NET_DVR_SetAUXResponseCallBack设置的回调函数中针对相应的ActionCode消息回应中携带的XML进行解析和处理。XML的解析，也提供有相应的解析接口。

IP_NET_DVR_GetDVRConfig（读取设备配置）

函数原型：
LONG IP_NET_DVR_GetDVRConfig(LONG lUserID,DWORD dwCommand,LONG lChannel,LPVOID lpOutBuffer,DWORD dwOutBufferSize,LPDWORD lpBytesReturned);

功能：
读取设备配置内容的底层接口，通过XML方式进行呈现。调用者可以不需要关注本接口，直接使用下面介绍的接口，在回调函数中调用XML解析接口来获取相应的配置

参数：
lUserID：登陆设备时返回的值
dwCommand：配置对应信息，见附表2
lChannel：为保兼容性以及以后扩展预留，设为0即可
lpOutBuffer：为保兼容性以及以后扩展预留，传空串（注意不是空指针）即可
dwOutBufferSize：为保兼容性以及以后扩展预留，设为0即可
lpBytesReturned：为保兼容性以及以后扩展预留，设为0即可

返回值：
返回0表示成功
ERR_NOT_FIND_DEVICE:句柄错误
ERR_DEV_NOT_CONNECTED:设备未连接
ERR_DEV_NOT_LOGIN：设备未登录成功
ERR_OUTOFF_MEMORY:内存不足

注意：
调用此函数后，返回结果都通过辅助通道回调函数返回，如果成功，则回调函数的错误标记位0，消息类型（注意后面24位的值）与dwCommand相同。

获取报警配置

LONG IP_NET_DVR_GET_AlarmConfig(LONG lUserID);

```
LONG IP_NET_DVR_GET_MotionDetectAlarm(LONG lUserID);
LONG IP_NET_DVR_GET_PersonDetectAlarm(LONG lUserID);
LONG IP_NET_DVR_GET_InputAlarm(LONG lUserID);
LONG IP_NET_DVR_GET_VideoLostAlarm(LONG lUserID);
LONG IP_NET_DVR_GET_VideoCoverAlarm(LONG lUserID);
LONG IP_NET_DVR_GET_StorageFullAlarm(LONG lUserID);
```

获取系统配置

```
LONG IP_NET_DVR_GET_SystemConfig(LONG lUserID);
LONG IP_NET_DVR_GET_PtzConfig(LONG lUserID);
LONG IP_NET_DVR_GET_UserConfig(LONG lUserID);
LONG IP_NET_DVR_GET_SyslogConfig(LONG lUserID);
LONG IP_NET_DVR_GET_TimeConfig(LONG lUserID);
LONG IP_NET_DVR_GET_MiscConfig(LONG lUserID);
```

获取音视频配置

```
LONG IP_NET_DVR_GET_MediaConfig(LONG lUserID);
LONG IP_NET_DVR_GET_VideoConfig(LONG lUserID);
LONG IP_NET_DVR_GET_VideoCaptureConfig(LONG lUserID);
LONG IP_NET_DVR_GET_VideoEncodeConfig(LONG lUserID);
LONG IP_NET_DVR_GET_VideoOverlayConfig(LONG lUserID);
LONG IP_NET_DVR_GET_VideoMaskConfig(LONG lUserID);
LONG IP_NET_DVR_GET_VideoROIConfig(LONG lUserID);
LONG IP_NET_DVR_GET_AudioConfig(LONG lUserID);
LONG IP_NET_DVR_GET_AudioEncode(LONG lUserID);
LONG IP_NET_DVR_GET_AudioCapture(LONG lUserID);
```

获取录像配置

```
LONG IP_NET_DVR_GET_RecordConfig(LONG lUserID);
```

获取媒体流访问配置

```
LONG IP_NET_DVR_GET_MediaStreamConfig(LONG lUserID);
```

获取 IPVS 平台配置

```
LONG IP_NET_DVR_GET_PlatformConfig(LONG lUserID);
```

获取 GB28181 平台配置

```
LONG IP_NET_DVR_GET_GB28181Config(LONG lUserID);
```

获取网络配置

```
LONG IP_NET_DVR_GET_NetworkConfig(LONG lUserID);
LONG IP_NET_DVR_GET_NetworkLANConfig(LONG lUserID);
LONG IP_NET_DVR_GET_NetworkWIFIConfig(LONG lUserID);
LONG IP_NET_DVR_GET_NetworkADSLConfig(LONG lUserID);
LONG IP_NET_DVR_GET_NetworkDDNSConfig(LONG lUserID);
LONG IP_NET_DVR_GET_NetworkUPNPConfig(LONG lUserID);
LONG IP_NET_DVR_GET_NetworkP2PConfig(LONG lUserID);
```

获取 FTP/EMAIL 服务器配置

```
LONG IP_NET_DVR_GET_ServerConfig(LONG lUserID);
LONG IP_NET_DVR_GET_FtpServerConfig(LONG lUserID);
LONG IP_NET_DVR_GET_SmtpServerConfig(LONG lUserID);
```

设置配置

说明：

设置配置前，必须将将数据填充完整，否则可能会导致前端设备解析不到正确的配置而工作异常。为确保配置数据正确，可以先获取一遍前端设备的配置，然后修改需要修改的项目，再调用相应接口来进行配置。

IP_NET_DVR_SetDVRConfig（设置设备配置）

函数原型：

```
LONG IP_NET_DVR_SetDVRConfig(LONG lUserID,DWORD dwCommand,LONG lChannel,LPVOID pXml,DWORD dwInBufferSize);
```

功能：

设置设备配置内容的底层接口，需要自行封装XML文本来传入参数。调用者可以不需要关注本接口，直接使用下面介绍的具体配置接口，填充结构体数据即可。

参数：

lUserID：登陆设备时返回的值

dwCommand：配置对应信息，见附表2

lChannel：为保兼容性以及以后扩展预留，设为0即可

pXml：配置XML文本内容

dwInBufferSize：文本内容的长度，即为strlen(pXml)

返回值：

返回0表示成功

ERR_NOT_FIND_DEVICE:句柄错误

ERR_DEV_NOT_CONNECTED:设备未连接

ERR_DEV_NOT_LOGIN：设备未登录成功

ERR_OUTOFF_MEMORY:内存不足

注意：

调用此函数后，返回结果都通过辅助通道回调函数返回，如果成功，则回调函数的错误标记位0，消息类型（注意后面24位的值）与dwCommand相同。

设置报警配置

```
LONG IP_NET_DVR_SET_AlarmConfig(LONG lUserID, AlarmConfig *pAlarmCfg);
LONG IP_NET_DVR_SET_InputAlarmConfig(LONG lUserID, InputAlarm *pInputAlm);
LONG IP_NET_DVR_SET_MotionDetectAlarm(LONG lUserID, MotionDetectAlarm *pMDAlm);
LONG IP_NET_DVR_SET_PersonDetectAlarm(LONG lUserID, PdAlarm *pPDAlm);
LONG IP_NET_DVR_SET_VlAlarmConfig(LONG lUserID, VideoLostAlarm *pVideoLost);
LONG IP_NET_DVR_SET_VCAlarmConfig(LONG lUserID, VideoCoverAlarm *pVideoCover);
LONG IP_NET_DVR_SET_OutputAlarmConfig(LONG lUserID, OutPutAlarm *p_config);
```

设置系统配置

```
LONG IP_NET_DVR_SET_MiscConfig(LONG lUserID, MiscConfig *pCfg);
LONG IP_NET_DVR_SET_UserConfig(LONG lUserID, UserConfig *pUserCfg);
LONG IP_NET_DVR_SET_TimeConfig(LONG lUserID, TimeConfig *pTimeCfg);
```

设置音视频配置

```
LONG IP_NET_DVR_SET_MediaConfig(LONG lUserID, MediaConfig *pConfig);
LONG IP_NET_DVR_SET_VideoCaptureConfig(LONG lUserID, VideoCapture *pCaptureCfg);
LONG IP_NET_DVR_SET_VideoEncodeConfig(LONG lUserID, VideoEncode *pEncodeCfg);
LONG IP_NET_DVR_SET_VideoOSDConfig(LONG lUserID, VideoOverlay *pCfg);
LONG IP_NET_DVR_SET_VideoUserOSDConfig(LONG lUserID, VideoUserOverlay *p_config);
LONG IP_NET_DVR_SET_VideoMaskConfig(LONG lUserID, VideoMaskConfig *pCfg);
LONG IP_NET_DVR_SET_VideoConfig(LONG lUserID, VideoConfig *pVideoCfg);
LONG IP_NET_DVR_SET_AudioConfig(LONG lUserID, AudioConfig *pAudioCfg);
```

设置媒体访问配置

```
LONG IP_NET_DVR_SET_MediaStreamConfig(LONG lUserID, MediaStreamConfig *config);
```

设置 IPVS 平台配置

```
LONG IP_NET_DVR_SET_PlatformConfig(LONG lUserID, PlatformConfig *pPlatformCfg);
```

设置 GB28181 平台配置

```
LONG IP_NET_DVR_SET_GB28181Config(LONG lUserID, GB28181Config *pPlatformCfg);
```

设置网络配置

```
LONG IP_NET_DVR_SET_NetworkConfig(LONG lUserID, NetworkConfigNew *config);
LONG IP_NET_DVR_SET_NetworkLANConfig(LONG lUserID, LANConfig *config);
LONG IP_NET_DVR_SET_NetworkWIFIConfig(LONG lUserID, WIFIConfig *config);
```

XML 配置解析接口

说明：

配置获取的结果，SDK通过辅助通道回调函数异步返回，需要调用者在IP_NET_DVR_SetAUXResponseCallBack设置的回调函数中针对相应的ActionCode消息回应中携带的XML进行解析和处理。XML的解析，可以调用本章提供的接口来解析成结构体。

```
LONG IP_NET_DVR_GetNetworkCfgByXml(NetworkConfigNew *pNetworkCfg, char *xmlBuf);

LONG IP_NET_DVR_Network_getLANCfgByXml(LANConfig *lanCfg,char *xmlBuf);
LONG IP_NET_DVR_Network_getWIFICfgByXml(WIFIConfig *wifiCfg,char *xmlBuf);
LONG IP_NET_DVR_Network_getADSLCfgByXml(ADSLConfigNew *adslCfg,char *xmlBuf);
LONG IP_NET_DVR_Network_getDDNSCfgByXml(DDNSConfig *ddnsCfg,char *xmlBuf);
LONG IP_NET_DVR_Network_getUPNPCfgByXml(UPNPConfig *upnpCfg,char *xmlBuf);
LONG IP_NET_DVR_Network_getP2PCfgByXml(P2PConfig *p2pCfg,char *xmlBuf);

LONG IP_NET_DVR_Server_getFtpsByXml(ServerConfig *pServerCfg, char *xmlBuf);
LONG IP_NET_DVR_Server_getSmtpsByXml(ServerConfig *pServerCfg, char *xmlBuf);
LONG IP_NET_DVR_GetServerCfgByXml(ServerConfig *pServerCfg, char *xmlBuf);


LONG IP_NET_DVR_GetRecordCfgByXml(RecordConfig *pRecordCfg, char *xmlBuf);


LONG IP_NET_DVR_Media_getAudioByXml(AudioConfig *pAudioCfg,  char *xmlBuf);
LONG IP_NET_DVR_Media_getVideoByXml(VideoConfig *pVideoCfg, char *xmlBuf);
LONG IP_NET_DVR_GetMediaCfgByXml(MediaConfig *pMediaCfg, char *xmlBuf);

LONG IP_NET_DVR_Media_getVideoCaptureByXml(VideoCapture *pVideoCapture, char *xmlBuf);
LONG IP_NET_DVR_Media_getVideoOverlayByXml(VideoOverlay *pVideoOverlay, char *xmlBuf);
LONG IP_NET_DVR_Media_getVideoEncodeByXml(VideoEncode *pVideoEncode, char *xmlBuf);
LONG IP_NET_DVR_Media_getJpegEncodeByXml(JpegEncodeCfg *pJpegCfg, char *xmlBuf);
LONG IP_NET_DVR_Media_getVideoMaskByXml(VideoMaskConfig *pVideoMask, char *xmlBuf);

LONG IP_NET_DVR_GetMediaStreamCfgByXml(MediaStreamConfig *pMediaStream, char *xmlBuf);
LONG IP_NET_DVR_GetPlatformCfgByXml(PlatformConfig *pPlatform, char *xmlBuf);
LONG IP_NET_DVR_GetGB28181CfgByXml(GB28181Config *pPlatformCfg, char *xmlBuf);


LONG IP_NET_DVR_Alarm_getInputByXml(InputAlarm *pInputAlm, char *xmlBuf);
LONG IP_NET_DVR_Alarm_getMotionDetectByXml(MotionDetectAlarm *pMDAlm, char *xmlBuf);
LONG IP_NET_DVR_Alarm_getPersonDetectByXml(PdAlarm *pPDAlm, char *xmlBuf);
LONG IP_NET_DVR_Alarm_getVideoLostByXml(VideoLostAlarm *pVideoLost, char *xmlBuf);
```

```
LONG IP_NET_DVR_Alarm_getVideoCoverByXml(VideoCoverAlarm *pVideoCover, char *xmlBuf);
LONG IP_NET_DVR_Alarm_getStorageFullByXml(StorageFullAlarm *pSFAlm, char *xmlBuf);
LONG IP_NET_DVR_GetAlarmConfigByXml(AlarmConfig *pAlarmCfg, char *xmlBuf);

LONG IP_NET_DVR_InputAlarm_getAlarmChannelCfgByXml(AlarmChannel *pAlarmChannel, char *xmlBuf);


LONG IP_NET_DVR_System_getPTZCfgByXml(PTZConfig *pPtzCfg, char *xmlBuf);
LONG IP_NET_DVR_System_getTimeCfgByXml(TimeConfig *pTimeCfg, char *xmlBuf);
LONG IP_NET_DVR_System_getUserCfgByXml(UserConfig *pUserCfg, char *xmlBuf);
LONG IP_NET_DVR_System_getLogCfgByXml(SyslogConfig *pSyslogCfg, char *xmlBuf);
LONG IP_NET_DVR_System_getMiscCfgByXml(MiscConfig *pMiscCfg, char *xmlBuf);
LONG IP_NET_DVR_GetSystemConfigByXml(SystemConfig *pSystemCfg, char *xmlBuf);
LONG IP_NET_DVR_GetSystemVersionByXml(SYSTEM_VERSION_DATA *pSystemVer, char *xmlBuf);


LONG IP_NET_DVR_System_getPTZCommonCfgByXml(PTZCommonConfig *pPtzCommonCfg, char *xmlBuf);
LONG IP_NET_DVR_System_getPTZAdvanceCfgByXml(PTZAdvanceConfig *pPtzAdvanceCfg, char *xmlBuf);
```

XML 配置生成接口

说明：

本章接口用于将结构体转换成XML文本，用于在设置配置时调用IP_NET_DVR_SetDVRConfig下发给前端设备。返回的XML文本的内存指针由动态malloc分配出u来，用完后需要调用者手动free，否则会造成内存泄漏。

```
char* IP_NET_DVR_XMLGET_SystemConfig(SystemConfig *pSystemCfg);
char* IP_NET_DVR_XMLGET_UserConfig(UserConfig *pUserCfg);
char* IP_NET_DVR_XMLGET_TimeConfig(TimeConfig *pTimeCfg);
char* IP_NET_DVR_XMLGET_SyslogConfig(SyslogConfig *pSyslogCfg);
char* IP_NET_DVR_XMLGET_SysMiscConfig(MiscConfig *pMiscCfg);
char* IP_NET_DVR_XMLGET_PTZConfig(PTZConfig *pPtzCfg);

char* IP_NET_DVR_XMLGET_AlarmConfig(AlarmConfig *pAlarmCfg);
char* IP_NET_DVR_XMLGET_InputAlarmConfig(InputAlarm *pInputAlm);
char* IP_NET_DVR_XMLGET_MDAlarmConfig(MotionDetectAlarm *pMDAlm);
char* IP_NET_DVR_XMLGET_PDAlarmConfig(PdAlarm *pPDAlm);
char* IP_NET_DVR_XMLGET_VlAlarmConfig(VideoLostAlarm *pVideoLost);
char* IP_NET_DVR_XMLGET_VCAlarmConfig(VideoCoverAlarm *pVideoCover);
char* IP_NET_DVR_XMLGET_SFAlarmConfig(StorageFullAlarm *pSFAlm);


char* IP_NET_DVR_XMLGET_MediaStreamConfig(MediaStreamConfig *mediaStreamCfg);
char* IP_NET_DVR_XMLGET_PlatformConfig(PlatformConfig *platformCfg);
char* IP_NET_DVR_XMLGET_GB28181Config(GB28181Config *platformCfg);


char* IP_NET_DVR_XMLGET_RecordConfig(RecordConfig *recordCfg);


char* IP_NET_DVR_XMLGET_NetworkConfig(NetworkConfigNew *networkCfg);
char* IP_NET_DVR_XMLGET_NetworkLANConfig(LANConfig *lanCfg);
char* IP_NET_DVR_XMLGET_NetworkWIFIConfig(WIFIConfig *wifiCfg);
char* IP_NET_DVR_XMLGET_NetworkADSLConfig(ADSLConfigNew *adslCfg);
char* IP_NET_DVR_XMLGET_NetworkDDNSConfig(DDNSConfig *ddnsCfg);
char* IP_NET_DVR_XMLGET_NetworkUPNPConfig(UPNPConfig *upnpCfg);
char* IP_NET_DVR_XMLGET_NetworkP2PConfig(P2PConfig *pCfg);
```

```
char* IP_NET_DVR_XMLGET_ServerConfig(ServerConfig *serverCfg);
char* IP_NET_DVR_XMLGET_FtpConfig(FtpServerList *fptServerList);
char* IP_NET_DVR_XMLGET_SmtpConfig(SmtpServerList *smtpServerList);


char* IP_NET_DVR_XMLGET_MediaConfig(MediaConfig *mediaCfg);
char* IP_NET_DVR_XMLGET_AudioConfig(AudioConfig *audioCfg);
char* IP_NET_DVR_XMLGET_VideoConfig(VideoConfig *videoCfg);

char* IP_NET_DVR_XMLGET_AudioCaptureConfig(AudioCapture* audioCfg);
char* IP_NET_DVR_XMLGET_AudioEncodeConfig(AudioEncode *audioCfg);


char* IP_NET_DVR_XMLGET_VideoOverlayConfig(VideoOverlay *pCfg);
char* IP_NET_DVR_XMLGET_VideoMaskConfig(VideoMaskConfig *pCfg);
char* IP_NET_DVR_XMLGET_VideoCaptureConfig(VideoCapture*pCfg);
char* IP_NET_DVR_XMLGET_VideoEncodeConfig(VideoEncode*pCfg);
```

5. 预览相关接口

注意：不建议使用NETSDK的预览接口，而是在登录前端设备成功后，通过标准的RTSP协议来跟前端设备进行交互。因为：

1、 预览接口通过RTSP协议与前端设备进行交互，使用开源代码live555实现。使用包含live555的NETSDK lib库才支持本节接口
2、 NETSDK预览接口RTSP性能一般
3、 如果调用者已经使用了live555开源代码，如果使用包含live555的NETSDK lib库，链接时会发生符号表冲突问题

IP_NET_DVR_SetRealDataCallBack（设置收流回调函数）

函数原型：

```
LONG IP_NET_DVR_SetRealDataCallBack(fRealDataCallBack cbRealDataCallBack, void *pUser);
```

功能：

设置接收媒体流的回调用函数，本函数所设置的回调会在请求流时传入的回调函数为NULL时起作用，即当作所有通道的默认回调函数。

参数：

cbRealDataCallBack：收到流时的回调函数

原型：`typedef LONG(CALLBACK *fRealDataCallBack)(LONG lRealHandle,DWORD dwDataType,BYTE *pBuffer,DWORD dwBufSize,LPFRAME_EXTDATA  pExtData);`

其中参数意义为:

lRealHandle用于标记是哪个视频流，即为本函数的返回值，
dwDataType=0表示视频，1表示音频,2为解码参数
pBuffer媒体数据指针
当dwDataType= 2 时，pBuffer结构体及其意义为

```
typedef struct  __STREAM_AV_PARAM
{
    unsigned char ProtocolName[32];  //==AV_FALG
    short  bHaveVideo;//0 表示没有视频参数
    short  bHaveAudio;//0 表示没有音频参数
    VIDEO_PARAM videoParam;//视频参数
    AUDIO_PARAM audioParam;//音频参数
}STREAM_AV_PARAM;
```

其中:

```
typedef struct
```

```
{
    CHAR codec[256];//H264或是MPEG4，等，表示当前视频编码类型
    INT width;      //视频宽
    INT height;     //视频高
    INT colorbits;  //颜色位数，此处无效，仅用于内部
    INT framerate;  //参考帧率
    INT bitrate;    //参考码率
    BYTE vol_data[256];//扩展参数,解码时必要的数据
    INT vol_length;    //扩展参数长度
}VIDEO_PARAM;

typedef struct
{
    CHAR codec[256];   //音频类型，AAC或G711
    INT samplerate;    //参考比特率
    INT bitspersample; //参考比特率
    INT channels; //声道数量
    INT framerate;
    INT bitrate;
}AUDIO_PARAM;
```

dwBufSize媒体数据长度

pExtData扩展参数，当dwDataType=0,1时结构及其意义为,当dwDataType=2时,此指针为空

```
typedef struct
{
    int bIsKey;//是否为关键帧
    double    timestamp;//时间截
    void    *pUserData;//用户数据指针，本为函数中pUser所设指针
}FRAME_EXTDATA,* LPFRAME_EXTDATA;
```

pUser：用于收到媒体数据时，当作参数进行回调仅用户使用。

返回值：

返回0表示成功

不会返回错误

注意：

本函数所设置的回调会在请求流时传入的回调函数为NULL时起作用，即当作所有通道的默认回调函数。如果每次请求流时都传了非空的回调函数，则此函数可以不调用。

IP_NET_DVR_RealPlay（连接媒体收流）

函数原型：

```
LONG IP_NET_DVR_RealPlay(LONG lUserID,LPIP_NET_DVR_CLIENTINFO lpClientInfo,fRealDataCallBack
cbRealDataCallBack,LPUSRE_VIDEOINFO pUser,BOOL bBlocked);
```

功能：

启动接收视频流

参数：

lUserID：登陆设备时返回的值，用于收流时读取IP地址。

lpClientInfo：为了保持与其它厂商SDK兼容而预留，传入NULL

cbRealDataCallBack：收到流时的回调函数

原型：

```
typedef LONG(CALLBACK *fRealDataCallBack)(LONG lRealHandle,DWORD dwDataType,BYTE *pBuffer,DWORD
dwBufSize,LPFRAME_EXTDATA  pExtData);
```

其中参数意义为:

lRealHandle用于标记是哪个视频流，即为本函数的返回值，

dwDataType=0表示视频，1表示音频,2为解码参数

pBuffer媒体数据指针

当dwDataType= 2 时，pBuffer结构体及其意义为

```
typedef struct  __STREAM_AV_PARAM
{
    unsigned char ProtocolName[32];  //==AV_FALG
    short  bHaveVideo;//0 表示没有视频参数
    short  bHaveAudio;//0 表示没有音频参数
    VIDEO_PARAM videoParam;//视频参数
    AUDIO_PARAM audioParam;//音频参数
}STREAM_AV_PARAM;
```

其中:

```
typedef struct
{
    CHAR codec[256];//H264或是MPEG4，等，表示当前视频编码类型
    INT width;      //视频宽
    INT height;     //视频高
    INT colorbits;  //颜色位数，此处无效，仅用于内部
    INT framerate;  //参考帧率
    INT bitrate;    //参考码率
    BYTE vol_data[256];//扩展参数,解码时必要的数据
    INT vol_length;    //扩展参数长度
}VIDEO_PARAM;

typedef struct
{
    CHAR codec[256];   //音频类型，AAC或G711
    INT samplerate;    //参考比特率
    INT bitspersample; //参考比特率
    INT channels; //声道数量
    INT framerate;
    INT bitrate;
}AUDIO_PARAM;
```

dwBufSize媒体数据长度

pExtData扩展参数，当dwDataType=0,1时结构及其意义为,当dwDataType=2时,此指针为空

```
typedef struct
{
    int bIsKey;//是否为关键帧
    double   timestamp;//时间截
    void    *pUserData;//用户数据指针，本为函数中pUser所设指针
}FRAME_EXTDATA,* LPFRAME_EXTDATA;
```

pUser：连接相关参数，结构定义为：

```
typedef struct
{
 int  nVideoPort;//连接的视频端口
 int  bIsTcp;//是否使用TCP连接
 int  nVideoChannle;//视频通道，为32位整数，高16表示类型，可以是0，1，0表示IPC，1表示DVS，低16位表示
   //视频通道号,当类型为0时，可以是0，1，0表示主码流，1表示子码流
   //当类型为1时，可以是0，1，2，3，4，5，分别表示通道1，2，3，4，和CIF输出。
 void *pUserData;//用户指针，用于回调时作为最后一参数
}USRE_VIDEOINFO,* LPUSRE_VIDEOINFO;
```

bBlocked：为兼容性保留参数

返回值：

返回0表示失败，否则返回唯一标识符

注意：

回调函数调用不能占太长时间（即，调用回调函数时，不能很久才返回）。

回调函数cbRealDataCallBack不是线程安全的。（即有可能多个线程同时调用此函数）

在状态回调函数上，可能会收到以下状态：

EVENT_RECVVIDEOPARAM=24,//收到视频编码参数

```
EVENT_RECVAUDIOPARAM,//收到音频编码参数
EVENT_CONNECTRTSPERROR,//连接失败
EVENT_CONNECTRTSPOK,//连接成功
EVENT_RTSPTHREADEXIT,//连接线程退出
EVENT_URLERROR,//URL解释错误
EVENT_RECVVIDEOAUDIOPARAM,//同时收到音视频参数。
EVENT_CONNECT_RTSPSERVER_ERROR=38,    //连接错误，可能是IP或端口错误
EVENT_CREATE_RTSPCLIENT_ERROR, //创建rtsp客户端错误
EVENT_GET_RTSP_CMDOPTION_ERROR, //连接rtsp服务器，返回错误
EVENT_RTSP_AUTHERROR,  //认证失败，可能是用户或密码错误
```

当nStateCode= EVENT_RECVVIDEOPARAM 时pResponse为结构体：

```
typedef struct
{
  CHAR codec[256];//H264或是MPEG4，等，表示当前视频编码类型
  INT width;      //视频宽
  INT height;     //视频高
  INT colorbits;  //颜色位数，此处无效，仅用于内部
  INT framerate;  //参考帧率
  INT bitrate;    //参考码率
  BYTE vol_data[256];//扩展参数,解码时必要的数据
  INT vol_length;    //扩展参数长度
}VIDEO_PARAM;
```

当nStateCode= EVENT_RECVAUDIOPARAM 时pResponse为结构体：

```
typedef struct
{
  CHAR codec[256];   //音频类型，AAC或G711
  INT samplerate;    //参考比特率
  INT bitspersample; //参考比特率
  INT channels; //声道数量
  INT framerate;
  INT bitrate;
}AUDIO_PARAM;
```

当nStateCode= EVENT_RECVVIDEOAUDIOPARAM 时pResponse为结构体：

```
typedef struct  __STREAM_AV_PARAM
{
    unsigned char ProtocolName[32];  //==AV_FALG
    short  bHaveVideo;//0 表示没有视频参数
    short  bHaveAudio;//0 表示没有音频参数
    VIDEO_PARAM videoParam;//视频参数
    AUDIO_PARAM audioParam;//音频参数
}STREAM_AV_PARAM;
```

IP_NET_DVR_RealPlayEx（连接媒体收流扩展接口）

函数原型：

```
LONG IP_NET_DVR_RealPlayEx(char * serverip,char *user,char *pass,fRealDataCallBack cbRealDataCallBack,LPUSRE_VIDEOINFO pUser,BOOL bBlocked);
```

功能：

启动接收视频流，与IP_NET_DVR_RealPlay不同之处是，此函数，可以不连接辅助通道的情况下接收视频流。

参数：

serverip：设备IP地址。
user：设备登录用户
pass：设备登录密码
cbRealDataCallBack：收到流时的回调函数
原型：typedef LONG(CALLBACK *fRealDataCallBack)(LONG lRealHandle,DWORD dwDataType,BYTE *pBuffer,DWORD dwBufSize,LPFRAME_EXTDATA pExtData);
其中参数意义为:

lRealHandle用于标记是哪个视频流，即为本函数的返回值，
dwDataType=0表示视频，1表示音频,2为解码参数
pBuffer媒体数据指针
当dwDataType= 2 时，pBuffer结构体及其意义为

```
typedef struct  __STREAM_AV_PARAM
{
    unsigned char ProtocolName[32];  //==AV_FALG
    short  bHaveVideo;//0 表示没有视频参数
    short  bHaveAudio;//0 表示没有音频参数
    VIDEO_PARAM videoParam;//视频参数
    AUDIO_PARAM audioParam;//音频参数
}STREAM_AV_PARAM;
```

其中:

```
typedef struct
{
    CHAR codec[256];//H264或是MPEG4，等，表示当前视频编码类型
    INT width;      //视频宽
    INT height;     //视频高
    INT colorbits;  //颜色位数，此处无效，仅用于内部
    INT framerate;  //参考帧率
    INT bitrate;    //参考码率
    BYTE vol_data[256];//扩展参数,解码时必要的数据
    INT vol_length;    //扩展参数长度
}VIDEO_PARAM;

typedef struct
{
    CHAR codec[256];   //音频类型，AAC或G711
    INT samplerate;    //参考比特率
    INT bitspersample; //参考比特率
    INT channels; //声道数量
    INT framerate;
    INT bitrate;
}AUDIO_PARAM;
```

dwBufSize媒体数据长度
pExtData扩展参数，当dwDataType=0,1时结构及其意义为,当dwDataType=2时,此指针为空

```
typedef struct
{
    int bIsKey;//是否为关键帧
    double   timestamp;//时间截
    void    *pUserData;//用户数据指针，本为函数中pUser所设指针
}FRAME_EXTDATA,* LPFRAME_EXTDATA;
```

pUser：连接相关参数，结构定义为：

```
typedef struct
{
 int  nVideoPort;//连接的视频端口
 int  bIsTcp;//是否使用TCP连接
 int  nVideoChannle;//视频通道，为32位整数，高16表示类型，可以是0，1，0表示IPC，1表示DVS，低16位表示
   //视频通道号,当类型为0时，可以是0，1，0表示主码流，1表示子码流
   //当类型为1时，可以是0，1，2，3，4，5，分别表示通道1，2，3，4，和CIF输出。
 void *pUserData;//用户指针，用于回调时作为最后一参数
```

```
        }USRE_VIDEOINFO,* LPUSRE_VIDEOINFO;
```

bBlocked：为兼容性保留参数

返回值：

返回0表示失败，否则返回媒体流句柄

注意：

回调函数调用不能占太长时间（即，调用回调函数时，不能很久才返回）。

回调函数cbRealDataCallBack不是线程安全的。（即有可能多个线程同时调用此函数）

在状态回调函数上，可能会收到以下状态：

```
EVENT_RECVVIDEOPARAM=24,//收到视频编码参数
EVENT_RECVAUDIOPARAM,//收到音频编码参数
EVENT_CONNECTRTSPERROR,//连接失败
EVENT_CONNECTRTSPOK,//连接成功
EVENT_RTSPTHREADEXIT,//连接线程退出
EVENT_URLERROR,//URL解释错误
EVENT_RECVVIDEOAUDIOPARAM,//同时收到音视频参数。
EVENT_CONNECT_RTSPSERVER_ERROR=38,    //连接错误，可能是IP或端口错误
EVENT_CREATE_RTSPCLIENT_ERROR, //创建rtsp客户端错误
EVENT_GET_RTSP_CMDOPTION_ERROR, //连接rtsp服务器，返回错误
EVENT_RTSP_AUTHERROR,  //认证失败，可能是用户或密码错误
```

当nStateCode= EVENT_RECVVIDEOPARAM 时pResponse为结构体：

```
typedef struct
{
  CHAR codec[256];//H264或是MPEG4，等，表示当前视频编码类型
  INT width;      //视频宽
  INT height;     //视频高
  INT colorbits;  //颜色位数，此处无效，仅用于内部
  INT framerate;  //参考帧率
  INT bitrate;    //参考码率
  BYTE vol_data[256];//扩展参数,解码时必要的数据
  INT vol_length;    //扩展参数长度
}VIDEO_PARAM;
```

当nStateCode= EVENT_RECVAUDIOPARAM 时pResponse为结构体：

```
typedef struct
{
    CHAR codec[256];   //音频类型，AAC或G711
    INT samplerate;    //参考比特率
    INT bitspersample; //参考比特率
    INT channels; //声道数量
    INT framerate;
    INT bitrate;
}AUDIO_PARAM;
```

当nStateCode= EVENT_RECVVIDEOAUDIOPARAM 时pResponse为结构体：

```
typedef struct  __STREAM_AV_PARAM
{
    unsigned char ProtocolName[32];  //==AV_FALG
    short  bHaveVideo;//0 表示没有视频参数
    short  bHaveAudio;//0 表示没有音频参数
    VIDEO_PARAM videoParam;//视频参数
    AUDIO_PARAM audioParam;//音频参数
}STREAM_AV_PARAM;
```

IP_NET_DVR_StopAllRealPlay(停止所有媒体流)

函数原型：
LONG IP_NET_DVR_StopAllRealPlay();
功能：
　　停止收流
参数：
　　无
返回值：
　　返回0表示成功

注意：
　　如果要停止所有媒体流，则调用此函数，以加快退出线程及释放相关资源。

IP_NET_DVR_StopRealPlay(停止收流)

函数原型：
LONG IP_NET_DVR_StopRealPlay(LONG lRealHandle);
功能：
　　停止收流
参数：
　　lRealHandle：调用IP_NET_DVR_RealPlayEx或IP_NET_DVR_RealPlay返回的非0值
返回值：
　　返回0表示成功
　　ERR_NOT_FIND_STREAM:句柄错误

注意：
　　调用此函数，将会等待收流线程结束才会返回，在极端情况下（如网络不好等情况），可能会占用一定的时间。经测试多数情况下都少于100毫秒。

IP_NET_DVR_GetVideoParam（获取视频参数）

函数原型：
LONG IP_NET_DVR_GetVideoParam(LONG lRealHandle,VIDEO_PARAM *pParam);
功能：
　　读取视频流中的视频参数结构
参数：
　　lRealHandle：调用IP_NET_DVR_RealPlayEx或IP_NET_DVR_RealPlay返回的非0值
　　pParam：视频参数输出

返回值：
　　返回0表示成功
　　ERR_NOT_FIND_STREAM:没有找到媒体流句柄

注意：
　　只有在成功登录，且状态回调函数中收到EVENT_RECVAUDIOPARAM调用才有效。
　　VIDEO_PARAM 定义为：

```
typedef struct
{
  CHAR codec[256];//H264或是MPEG4，等，表示当前视频编码类型
  INT width;      //视频宽
  INT height;     //视频高
  INT colorbits;  //颜色位数，此处无效，仅用于内部
  INT framerate;  //参考帧率
  INT bitrate;    //参考码率
  BYTE vol_data[256];//扩展参数,解码时必要的数据
  INT vol_length;    //扩展参数长度
}VIDEO_PARAM;
```

IP_NET_DVR_GetAudioParam（获取音频参数）

函数原型：
LONG IP_NET_DVR_GetAudioParam(LONG lRealHandle,AUDIO_PARAM *pParam);
功能：
读取视频流中的音频参数结构
参数：
lRealHandle：调用IP_NET_DVR_RealPlayEx或IP_NET_DVR_RealPlay返回的非0值
pParam：音频参数输出
返回值：
返回0表示成功
ERR_NOT_FIND_STREAM:没有找到媒体流句柄

注意：
只有在成功登录，且状态回调函数中收到EVENT_RECVVIDEOPARAM调用才有效。

AUDIO_PARAM定义如下:

```
typedef struct
{
  CHAR codec[256];
  INT samplerate;
  INT bitspersample;
  INT channels;
  INT framerate;
  INT bitrate;
}AUDIO_PARAM;
```

IP_NET_DVR_GetDeviceAbility（获取设置能力）

函数原型：
LONG IP_NET_DVR_GetDeviceAbility(LONG lUserID);
功能：
获取设备的参数集，读取成功后将由辅助通道消息回调函数返回
参数：
lUserID：登陆设备时返回的值
返回值：
返回0表示成功
ERR_NOT_FIND_DEVICE:句柄错误
ERR_DEV_NOT_CONNECTED:设备未连接
ERR_DEV_NOT_LOGIN：设备未登录成功

ERR_OUTOFF_MEMORY:内存不足

注意：
调用此函数后，返回结果都通过辅助通道回调函数返回，如果成功，则回调函数的错误标记位0，消息类型1020,返回Xml为:

<RESPONSE_PARAM SystemConfigString=" + audio_support + schedule_record + ftpemail_storage + picture_capture + ptz_control + gpio_input + gpio_output + zh_cn + en_us + zh_tw + front_replay + media_capabiltiy" />

audio_support //支持音频
schedule_record //支持定时录像
ftpemail_storage //支持 ftp 和 email
picture_capture //支持抓图
ptz_control //支持云台
gpio_input //支持 IO 输入
gpio_output //支持 IO 输出
zh_cn //设备配置界面是否支持中文
en_us //设备配置界面是否支持英文
zh_tw //设备配置界面是否支持繁体
front_replay //支持前端在线回放
media_capabiltiy //可以查询设备支持的分辨率
wireless_accesspoint //支持 WIFI 作为客户端功能
wireless_station //支持 WIFI 作为服务端功能
network_storage //支持网络存储
ptz_action //支持云台联动
ircut_setting //支持 IRCUT 功能
evdo_support //支持 EVDO 3G 功能
wcdma_support //支持 WCDMA 3G 功能
tdscdma_support //支持 TDSCDMA 3G 功能

6. 录像回放相关接口

IP_NET_DVR_GetReplayAblity 取得是否支持在线回放前端文件

函数原型：
LONG IP_NET_DVR_GetReplayAblity(LONG lUserID);

参数：
lUserID：登录设备的返回值，唯一标识

返回值：
ERR_NOT_FIND_DEVICE:句柄错误
返回-1，表示当前状态未知，因为还没有登录成功或是还没有获取到能力集，可以调用IP_NET_DVR_GetDeviceAbility进行判断。
0：不支持在线回放，只能是下载回来后再回放
1：支持在线回放

IP_NET_DVR_GetReplay_OneDay_Distribute 获取某一天的录像分布

函数原型：

```
LONG IP_NET_DVR_GetReplay_OneDay_Distribute(LONG lUserID, int year, int month, int day);
```

参数：

lUserID：登录设备的返回值，唯一标识

year,month,day:指定查询摄像机的本地时间年月日

返回值：

ERR_NOT_FIND_DEVICE:句柄错误

返回-1，表示当前状态未知，因为还没有登录成功

说明：

该API调用后，不会阻塞等待结果，通过辅助通道回调函数异步返回结果，需要调用者在IP_NET_DVR_SetAUXResponseCallBack设置的回调函数中对ActionCode =CMD_GET_RECORD_FILE_LIST且SearchMode=10的消息回应XML进行解析和处理，录像分布为这一天以1分钟为单位的录像分布字符串（6X24=1440字节），每字节含义

A:定时录像

B:报警录像（目前IPC只有在关闭定时录像、开启报警录像的配置时，才会上报报警录像的分布。后续IPC会更新功能，支持在定时录像的情况下也记录报警事件、上报相应时段的报警类别）

C:无录像

D:移动侦测

E:IO报警

F:人形识别

G:车牌识别

H:人脸识别

I:视频遮挡

例如：

<RESPONSE_PARAM SearchMode="1">
<RECORD_DISTRIBUTE
Distribute="AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABABAAAABBBABBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBABBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBAAAAAAAAABAABBBABAABBBBBBBBBBBBBABABBBBABABABBA
BAABAAAABAAABABAAAAAAAAAAAABAAAAAAAAABBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBABABBBBBBBBBBBBBBBBBBBBBBBBBCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC"
/>
</RESPONSE_PARAM>

IP_NET_DVR_GetReplay_Dates_Distribute 获取有录像的日期分布

函数原型：

```
LONG IP_NET_DVR_GetReplay_Dates_Distribute(LONG lUserID);
```

参数：

lUserID：登录设备的返回值，唯一标识

返回值：

ERR_NOT_FIND_DEVICE:句柄错误

返回-1，表示当前状态未知，因为还没有登录成功

说明：

该API调用后，不会阻塞等待结果，而是通过异步消息返回结果，需要调用者在IP_NET_DVR_SetAUXResponseCallBack设置的回调函数中对ActionCode =CMD_GET_RECORD_FILE_LIST且SearchMode=10的消息回应XML进行解析和处理，XML中返回了有录像的所有日期，每条记录包含了相应日期零点从1970年1月1日零点零分零秒开始的时间秒数以及年月日字符串，例如：

```
<RESPONSE_PARAM SearchMode="10" DateCount="12">
  <DATE UtcTime="1645286400" DateStr="20220220"/>
  <DATE UtcTime="1645372800" DateStr="20220221"/>
  <DATE UtcTime="1645459200" DateStr="20220222"/>
  <DATE UtcTime="1645545600" DateStr="20220223"/>
  <DATE UtcTime="1645632000" DateStr="20220224"/>
  <DATE UtcTime="1645718400" DateStr="20220225"/>
  <DATE UtcTime="1645804800" DateStr="20220226"/>
  <DATE UtcTime="1645891200" DateStr="20220227"/>
  <DATE UtcTime="1645977600" DateStr="20220228"/>
  <DATE UtcTime="1646064000" DateStr="20220301"/>
  <DATE UtcTime="1646150400" DateStr="20220302"/>
  <DATE UtcTime="1646236800" DateStr="20220303"/>
</RESPONSE_PARAM>
```

IP_NET_DVR_ControlReplay 前端 IPC 文件回放控制

函数原型：

```
LONG IP_NET_DVR_ControlReplay(LONG lUserID,LONG Action,LONG param);
```

功能：对回放文件进行快进等操作控制

参数：

lUserID：登录时返回的句柄

Action:播放操作类型,具体值如下

```
enum REPLAY_IPC_ACTION
{
ACTION_PLAY=0,//播放
ACTION_PAUSE,//暂停
ACTION_RESUME,//恢复播放
ACTION_FAST,//快进
ACTION_SLOW,//慢放
ACTION_SEEK,//定位
ACTION_FRAMESKIP,//单帧
ACTION_STOP,//停止
};
```

Param:控制参数

当为ACTION_FAST，ACTION_SLOW时，此值为播放倍数

当为ACTION_SEEK时，表示要播放到第几秒。

返回值：

成功返回0

ERR_NOT_REPLAY_MODE_ERROR:当前不是处于回放模式

ERR_NOT_FIND_DEVICE:句柄错误

ERR_DEV_NOT_CONNECTED:设备未连接

ERR_DEV_NOT_LOGIN：设备未登录成功

ERR_OUTOFF_MEMORY:内存不足

ERR_PLAY_ACTION_ERROR: Action错误

IP_NET_DVR_SetReplayDataCallBack 设置前端文件回放流回调

函数原型：

```
LONG IP_NET_DVR_SetReplayDataCallBack(LONG lUserID,fRealDataCallBack cbRealDataCallBack,DWORD dwUser);
```

功能：

设置前端回放文件时，媒体流回调函数

参数：

lUserID：登录时返回的句柄

cbRealDataCallBack：收到流时的回调函数

原型：typedef LONG(CALLBACK *fRealDataCallBack)(LONG lRealHandle,DWORD dwDataType,BYTE *pBuffer,DWORD dwBufSize,LPFRAME_EXTDATA pExtData);

其中参数意义为:

lRealHandle用于标记是哪个视频流，即为本函数的返回值，

dwDataType=0表示视频，1表示音频,2为解码参数

pBuffer媒体数据指针

当dwDataType= 2 时，pBuffer结构体及其意义为

```
typedef struct  __STREAM_AV_PARAM
{
    unsigned char ProtocolName[32];  //==AV_FALG
    short  bHaveVideo;//0 表示没有视频参数
    short  bHaveAudio;//0 表示没有音频参数
    VIDEO_PARAM videoParam;//视频参数
    AUDIO_PARAM audioParam;//音频参数
}STREAM_AV_PARAM;
```

其中:

```
typedef struct
{
    CHAR codec[256];//H264或是MPEG4，等，表示当前视频编码类型
    INT width;      //视频宽
    INT height;     //视频高
    INT colorbits;  //颜色位数，此处无效，仅用于内部
    INT framerate;  //参考帧率
    INT bitrate;    //参考码率
    BYTE vol_data[256];//扩展参数,解码时必要的数据
    INT vol_length;    //扩展参数长度
}VIDEO_PARAM;

typedef struct
{
    CHAR codec[256];   //音频类型，AAC或G711
    INT samplerate;    //参考比特率
    INT bitspersample; //参考比特率
    INT channels; //声道数量
    INT framerate;
    INT bitrate;
}AUDIO_PARAM;
```

dwBufSize媒体数据长度

pExtData扩展参数，当dwDataType=0,1时结构及其意义为,当dwDataType=2时,此指针为空

```
typedef struct
{
    int bIsKey;//是否为关键帧
    double    timestamp;//时间截
    void    *pUserData;//用户数据指针，本为函数中pUser所设指针
}FRAME_EXTDATA,* LPFRAME_EXTDATA;
```

返回值：
0表示成功
不会返回错误

IP_NET_DVR_PlayDeviceFile 播放前端文件

函数原型：
```
LONG IP_NET_DVR_PlayDeviceFile(LONG lUserID,char * filenme);
```

功能：
播放指定的前端文件（暂未实现，预留）

参数：
lUserID：登录时返回的句柄
filenme：要播放的设备上存储的文件名（可以通过查询得到）

返回值：
ERR_NOT_FIND_DEVICE:句柄错误
ERR_NOT_ALLOW_REPLAY_ERROR：不支持回放或是未读取未获取是否支持回放。
ERR_DEV_NOT_CONNECTED:设备未连接
ERR_DEV_NOT_LOGIN：设备未登录成功

注意：
调用此函数前，应该先查询是否支持前端回放，另外，应该先调用IP_NET_DVR_SetReplayDataCallBack设置回放流回调函数。

7. 文件下载相关接口

IP_NET_DVR_GetFileByName（下载文件）

函数原型：
```
LONG IP_NET_DVR_GetFileByName(LONG lUserID,LONG nFileType,char *sDVRFileName,char *sSavedFileName);
```
功能：
下载指定文件
参数：
lUserID：登陆设备时返回的值
nFileType：要下载的文件类型,0表示日志文件，1表示录像文件
sDVRFileName：要下载的文件名
sSavedFileName:要保存到本地的文件名
返回值：
0：表示已提交下载任务,对于下载进度可以调用IP_NET_DVR_GetDownloadPos查询，下载是否成功将由状态回调函数返回。
ERR_NOT_FIND_DEVICE：句柄错误
ERR_PARAM_ERROR：表示sDVRFileName或sSavedFileName为空
ERR_DEV_NOT_CONNECTED：表示设备未连接
ERR_DEV_NOT_LOGIN：表示设备未成功登录
ERR_ISUPLOADING_ERROR：表示正在上传中
ERR_ISDOWNLOADING_ERROR：表示正在下载中

ERR_IS_STARTAUDIO_ERROR：表示正在对讲中，不允许下载

注意：

1,同一个设备只能上传或下载一个文件。

2,如果返回0，调用此函数后，在状态回调函数上可能收到状态有：

EVENT_DOWNLOADOK=16, //下载完成

EVENT_DOWNLOADFAILED, //下载失败

IP_NET_DVR_StopGetFile（停止下载）

函数原型：

LONG IP_NET_DVR_StopGetFile(LONG lUserID);

功能：

停止下载文件

参数：

lUserID：登陆设备时返回的值

返回值：

0：表示成功

ERR_NOT_FIND_DEVICE:句柄错误

ERR_ISFINISH_ERROR：表示当前没有在下载文件或是下载已完成

IP_NET_DVR_GetDownloadPos（获取下载进度）

函数原型：

LONG IP_NET_DVR_GetDownloadPos(LONG lUserID);

功能：

取得下载进度

参数：

lUserID：登陆设备时返回的值

返回值：

101：已下载完成

ERR_NOT_DOWNLOAD_MODE_ERROR：当前不是在下载中

大于等于0:当前下载进度

8. 云台控制相关接口

IP_NET_DVR_PTZControl（基本云台控制）

函数原型：

LONG IP_NET_DVR_PTZControl(LONG lUser,DWORD dwPTZCommand,DWORD nTspeed,DWORD nPpeed);

功能：

进行简单云台控制。

参数：

lUserID：登陆设备时返回的值

dwPTZCommand：云台操作类型,可能值如下

```
enum PTZ_CMD_TYPE
```

```
{
  LIGHT_PWRON=2,// 2 接通灯光电源
  WIPER_PWRON,// 3 接通雨刷开关
  FAN_PWRON,// 4 接通风扇开关
  HEATER_PWRON,// 5 接通加热器开关
  AUX_PWRON1,// 6 接通辅助设备开关
  AUX_PWRON2,// 7 接通辅助设备开关
  ZOOM_IN= 11,// 焦距变大(倍率变大)
  ZOOM_OUT, //12 焦距变小(倍率变小)
  FOCUS_NEAR, //13 焦点前调
  FOCUS_FAR, //14 焦点后调
  IRIS_OPEN, //15 光圈扩大
  IRIS_CLOSE,// 16 光圈缩小
  TILT_UP,// 21 云台上仰
  TILT_DOWN, //22 云台下俯
  PAN_LEFT,// 23 云台左转
  PAN_RIGHT,// 24 云台右转
  UP_LEFT,// 25 云台上仰和左转
  UP_RIGHT,// 26 云台上仰和右转
  DOWN_LEFT, //27 云台下俯和左转
  DOWN_RIGHT,// 28 云台下俯和右转
  PAN_AUTO,// 29 云台左右自动扫描
  STOPACTION// 30 停止云台动作
};
```

nTspeed：如果是控制云台转动，则表示水平速度

nPpeed：如果是控制云台转动表示垂直速度

返回值：

返回0表示成功

ERR_NOT_FIND_DEVICE:句柄错误

ERR_DEV_NOT_CONNECTED:设备未连接

ERR_DEV_NOT_LOGIN：设备未登录成功

ERR_PTZCMD_ACTION_ERROR:云台命令错误

ERR_OUTOFF_MEMORY:内存不足

注意：

调用返回0表示此命令已增加到发送队列，对于命令是否发送成功，则要等待EVENT_SENDPTZOK,EVENT_SENDPTZFAILED这两个事件进行判断。。操作成功后，将会在状态回调函数上收到一个EVENT_PTZPRESETINFO事件

IP_NET_DVR_PTZPreset（预置点控制）

函数原型：

```
LONG IP_NET_DVR_PTZPreset(LONG lUser,DWORD dwPTZPresetCmd,DWORD dwPresetIndex);
```

功能：

云台的预置点的基本操作

参数：

lUserID：登陆设备时返回的值

dwPTZPresetCmd：操作类型，可以是

```
enum PTZ_PRESET_TYPE
{
    SET_PRESET= 8 ,//设置
    CLE_PRESET= 9,//删除
    GOTO_PRESET= 39//查询
};
```

dwPresetIndex：要操作的预置点编号，可以是0-255

返回值：

返回0表示成功

ERR_NOT_FIND_DEVICE:句柄错误

ERR_DEV_NOT_CONNECTED:设备未连接
ERR_DEV_NOT_LOGIN：设备未登录成功
ERR_OUTOFF_MEMORY:内存不足
ERR_PTZCMD_ACTION_ERROR: dwPTZPresetCmd错误
ERR_PARAM_ERROR: dwPresetIndex错误

注意：
调用返回0表示此命令已增加到发送队列，对于命令是否发送成功，则要等待EVENT_SENDPTZOK,EVENT_SENDPTZFAILED这两个事件进行判断。操作成功后，将会在状态回调函数上收到一个EVENT_PTZPRESETINFO事件

IP_NET_DVR_PTZControlEx（云台控制扩展接口）

函数原型：
```
LONG IP_NET_DVR_PTZControlEx(LONG lUser,char *pXml);
```
功能：

参数：
lUserID：登陆设备时返回的值
pXml：云台操作XML的格式，具体格式及对应功能见附表一
返回值：
返回0表示成功
ERR_NOT_FIND_DEVICE:句柄错误
ERR_DEV_NOT_CONNECTED:设备未连接
ERR_DEV_NOT_LOGIN：设备未登录成功
ERR_OUTOFF_MEMORY:内存不足
ERR_PARAM_ERROR: pXml为空

注意：
调用返回0表示此命令已增加到发送队列，对于命令是否发送成功，则要等待EVENT_SENDPTZOK,EVENT_SENDPTZFAILED这两个事件进行判断。操作成功后，将会在状态回调函数上收到一个EVENT_PTZPRESETINFO事件

9. 广播/对讲控制相关接口

IP_NET_DVR_StartVoiceCom（启用对讲）

函数原型：
```
LONG IP_NET_DVR_StartVoiceCom(LONG lUserID,DWORD dwVoiceChan,BOOL bNeedCBNoEncData,fVoiceDataCallBack
cbVoiceDataCallBack,void *pUser);
```
功能：
启用对讲
参数：
lUserID：登陆设备时返回的值
dwVoiceChan：为保兼容性预留，传入0
bNeedCBNoEncData：为保有兼容性预留传入1
cbVoiceDataCallBack：为保有兼容性预留传入NULL
pUser：为保有兼容性预留传入NULL

返回值：
返回0表示成功
ERR_NOT_FIND_DEVICE:句柄错误

ERR_IS_STARTAUDIO_ERROR:当前设备处于对讲状态
ERR_DEV_NOT_CONNECTED:设备未连接
ERR_DEV_NOT_LOGIN：设备未登录成功

注意：
调用此函数，返回成功，仅表示已请向设备请求对讲，对于能否启动成功，还要在状态回调函数等待
EVENT_STARTAUDIOOK=6,//启动对讲成功
EVENT_STARTAUDIOFAILED=7,// 启动对讲失败
EVENT_STARTAUDIO_ISBUSY //设备对讲被占用
EVENT_STARTAUDIO_PARAMERROR //不支持此音频参数对讲
EVENT_STARTAUDIO_AUDIODDISABLED //不支持对讲

注意：
调用此函数后，只是启动了设备对讲，只有调用IP_NET_DVR_StartTalk初始化音频采集，则时调用IP_NET_DVR_AddTalk 将设备加入广播列表，才对真正进行对讲.

IP_NET_DVR_StopVoiceCom（停止对讲）

函数原型：
LONG IP_NET_DVR_StopVoiceCom(LONG lVoiceComHandle);
功能：
停止对讲
参数：
lUserID：登陆设备时返回的值
返回值：
返回0表示成功
ERR_NOT_FIND_DEVICE:句柄错误
ERR_AUDIO_NOT_START:设备不是正在对讲状态

注意：
返回0后，还要在状态回调函数中，等待以下两个状态，才能判断是否停止成功
EVENT_STOPAUDIOOK,
EVENT_STOPAUDIOFAILED,

IP_NET_DVR_InputAudioData 送入对讲/广播数据

函数原型：
LONG IP_NET_DVR_InputAudioData(LONG lUserID, const char* pBuffer, int nSize);

参数：
lUserID:登录设备时返回的唯一标记值
pBuffer:音频数据指针
nSize:音频数据长度

返回值：
0表示成功
ERR_NOT_FIND_DEVICE：句柄错误

IP_NET_DVR_StartTalk 初始化音频广播参数

函数原型：
```
LONG IP_NET_DVR_StartTalk(int audiotype, int samplerate, int bitspersample, int channels);
```

参数：
audiotype:表示音频类型，1表示AAC，0表示G711
samplerate:采样率，一般为16000
bitspersample：一般为16
channels：声音数，一般为2
返回值：
0,表示成功
ERR_OPEN_AUDIOCAPTURE_FAIL：打开音频采集设备失败，可能是声卡采集被占用
ERR_START_AUDIOCAPTURE_FAIL:启动音频采集失败，可能是参数错误.

注意：
如果当前已启动对讲，则会先关掉当前对讲，重新使用指定的参数启用对讲。

IP_NET_DVR_StopTalk 释放音频广播相关资源

函数原型：
```
LONG IP_NET_DVR_StopTalk();
```

参数：
无

返回值：
0表示成功
不会返回错误
注意：
如果有设置处于广播时，会停止所有设备对讲。

IP_NET_DVR_AddTalk 将设备加入广播列表

函数原型：
```
LONG IP_NET_DVR_AddTalk(LONG lUserID);
```

参数：
lUserID:登录设备时返回的控制句柄

返回值：
0表示成功
ERR_NOT_FIND_DEVICE:句柄错误
ERR_NOT_STARTTALK_MODE_ERROR：当前未初始化广播参数，要先调用IP_NET_DVR_StartTalk设置相关参数
ERR_AUDIO_PARAM_ERROR:当前设备未启动对讲，或是参数不相同。要先调用IP_NET_DVR_StartVoiceCom才能将设备加入广播列表
ERR_NOT_STARTTALK_MODE_ERROR:当前未初始化音频广播参数,要先调用IP_NET_DVR_StartTalk进行初始化

注意:
要先调用IP_NET_DVR_StartVoiceCom启用设备的对讲功能,如果参数与调用IP_NET_DVR_StartTalk相同,则可以将设备加入广播列表.

IP_NET_DVR_RemoveTalk 停止向指定设备广播

函数原型：
LONG IP_NET_DVR_RemoveTalk(LONG lUserID);

参数：
lUserID:登录设备时返回的唯一标记值

返回值：
0表示成功
ERR_NOT_FIND_DEVICE：句柄错误
ERR_AUDIO_NOT_START：当前设备不是处于对讲状态

10. 系统控制相关接口

IP_NET_DVR_FormatDisk（格式化存储设备）

函数原型：
LONG IP_NET_DVR_FormatDisk(LONG lUserID,LONG lDiskNumber);
功能：
格式化存储设备
参数：
lUserID：登陆设备时返回的值
lDiskNumber:0表示sd1,1表示sd2,2表示usb
返回值：
返回0表示成功
ERR_NOT_FIND_DEVICE:句柄错误
ERR_DEV_NOT_CONNECTED:设备未连接
ERR_DEV_NOT_LOGIN：设备未登录成功
ERR_OUTOFF_MEMORY:内存不足

注意：
调用返回0表示此命令已增加到待处理队列.在辅助通道回调函数中将会有一个消息类型为1017的消息回调。返回XML中如果flag=0表示成功

IP_NET_DVR_Upgrade（升级固件）

函数原型：
LONG IP_NET_DVR_Upgrade(LONG lUserID,char *sFileName);
功能：

参数：
lUserID：登陆设备时返回的值

sFileName：本地电脑中的固件文件，使用绝对路径
返回值：
返回0表示已处理请求。
ERR_NOT_FIND_DEVICE:句柄错误
ERR_LOC_FILE_ERROR:文件sFileName错误
ERR_ISDOWNLOADING_ERROR:正在下载文件，不允许升级
ERR_ISUPLOADING_ERROR:正在上传，不允许升级
ERR_NOT_FIND_DEVICE:句柄错误
ERR_DEV_NOT_CONNECTED:设备未连接
ERR_DEV_NOT_LOGIN：设备未登录成功
ERR_OUTOFF_MEMORY:内存不足

注意：
1、调用返回0表示此命令已增加到待处理队列.
2、当上传升级文件失败时，将由事件StateEventCallBack回调,ID=EVENT_UPLOADFAILED通知.
3、当上传升级文件失败时，将由事件StateEventCallBack回调,ID=EVENT_UPLOADOK通知.
4、上传完成后，将会重新启动设备，此时会重新连接，一般判断升级成不成功，就判断升级前后版本号是否一致，如果不一致表示成功。所以升级前一般先读取版本号.

IP_NET_DVR_GetUpgradeProgress（获取升级固件进度）

函数原型：
LONG IP_NET_DVR_GetUpgradeProgress(LONG lUserID);
功能：
取得升级固件上传进度
参数：
lUserID：登陆设备时返回的值
返回值：
ERR_NOT_FIND_DEVICE:句柄错误
ERR_LOC_FILE_ERROR:文件sFileName错误
返回：101，表示当前没有在升级，或是升级已文件上传完。
返回大于等于0表示当前进度百分比，小于0表示失败。
注意：
只有在上升状态时才有效，否则会返回负值。

IP_NET_DVR_GetUpgradeState（获取固件升级状态）

函数原型：
LONG IP_NET_DVR_GetUpgradeState(LONG lUserID);
功能：
获取固件升级状态
参数：
lUserID：登陆设备时返回的值
返回值：
0：表示处于升级状态
ERR_NOT_FIND_DEVICE:设备
101：表示当前已停止升级或升级完成
负值：表示当前不是处理升级模式

IP_NET_DVR_CloseUpgradeHandle（停止升级）

函数原型：
LONG IP_NET_DVR_CloseUpgradeHandle(LONG lUserID);
功能：
　　停止升级固件
参数：
　　lUserID：登陆设备时返回的值
返回值：
　　返回0表示成功
　　ERR_NOT_FIND_DEVICE:句柄错误
　　ERR_ISFINISH_ERROR:升级文件已上传完，或是未启动升级

IP_NET_DVR_FindDVRLogFile（搜索设备日志文件）

函数原型：
LONG IP_NET_DVR_FindDVRLogFile(LONG lUserID);
功能：
　　搜索日志文件
参数：
　　lUserID：登陆设备时返回的值
返回值：
　　返回0表示成功
　　ERR_NOT_FIND_DEVICE:句柄错误
　　ERR_DEV_NOT_CONNECTED:设备未连接
　　ERR_DEV_NOT_LOGIN：设备未登录成功
　　ERR_OUTOFF_MEMORY:内存不足

注意：
　　调用后，搜索返回结果，将在辅助通道回调函数（由IP_NET_DVR_SetAUXResponseCallBack设置），类型为CMD_GET_LOGFILE_LIST(1025)返回

IP_NET_DVR_RestoreConfig（恢复出厂设置）

函数原型：
LONG IP_NET_DVR_RestoreConfig(LONG lUserID);
功能：
　　恢复默认配置
参数：
　　lUserID：登陆设备时返回的值
返回值：
　　返回0表示成功
　　ERR_NOT_FIND_DEVICE:句柄错误
　　ERR_DEV_NOT_CONNECTED:设备未连接
　　ERR_DEV_NOT_LOGIN：设备未登录成功
　　ERR_OUTOFF_MEMORY:内存不足

IP_NET_DVR_GetConfigFile（下载配置文件）

函数原型：
LONG IP_NET_DVR_GetConfigFile(LONG lUserID,char *sFileName);
功能：
    下载配置文件
参数：
    lUserID：登陆设备时返回的值
返回值：
    返回0表示成功
    ERR_NOT_FIND_DEVICE:句柄错误
    ERR_DEV_NOT_CONNECTED:设备未连接
    ERR_DEV_NOT_LOGIN：设备未登录成功
    ERR_OUTOFF_MEMORY:内存不足

IP_NET_DVR_SetConfigFile（上传配置文件并替换）

函数原型：
LONG IP_NET_DVR_SetConfigFile(LONG lUserID,char *sFileName);
功能：
    上传配置文件并替换
参数：
    lUserID：登陆设备时返回的值
返回值：
    返回0表示成功
    ERR_NOT_FIND_DEVICE:句柄错误
    ERR_DEV_NOT_CONNECTED:设备未连接
    ERR_DEV_NOT_LOGIN：设备未登录成功
    ERR_OUTOFF_MEMORY:内存不足

IP_NET_DVR_RebootDVR（重启设备）

函数原型：
LONG IP_NET_DVR_RebootDVR(LONG lUserID);
功能：
    重启设备
参数：
    lUserID：登陆设备时返回的值
返回值：
    返回0表示成功
    ERR_NOT_FIND_DEVICE:句柄错误
    ERR_DEV_NOT_CONNECTED:设备未连接
    ERR_DEV_NOT_LOGIN：设备未登录成功
    ERR_OUTOFF_MEMORY:内存不足

IP_NET_DVR_SystemControl(系统控制)

函数原型：
LONG IP_NET_DVR_SystemControl(LONG lUserIDDWORD dwCommand ,LONG flag,char * pXml);
功能：
对设备进行高级控制。
参数：
lUserID：登陆设备时返回的值
dwCommand：配置对应信息，见附表2系统控制部分
flag：传0即可
pXml：操作相关参数的XML格式

返回值：
返回0表示成功
ERR_NOT_FIND_DEVICE:句柄错误
ERR_DEV_NOT_CONNECTED:设备未连接
ERR_DEV_NOT_LOGIN：设备未登录成功
ERR_OUTOFF_MEMORY:内存不足

注意：
调用此函数后，返回结果都通过辅助通道回调函数返回，如果成功，则回调函数的错误标记位0，消息类型（注意后面24位的值）与dwCommand相同。

IP_NET_DVR_CreateIFrame 产生 I 帧

函数原型：
LONG IP_NET_DVR_CreateIFrame(ULONG lUserId,int bIsSubStream);

参数：
lUserID:登录设备时返回的唯一标记值
bIsSubStream：0 表示主码流，1 表示子码流

返回值：
0表示成功
ERR_NOT_FIND_DEVICE：句柄错误
ERR_DEV_NOT_CONNECTED:设备未连接
ERR_DEV_NOT_LOGIN：设备未登录成功

注意：
此函数，并不会判断是否申请了媒体流。

IP_NET_DVR_SetUserData 保存用户数据到设备

函数原型：
LONG IP_NET_DVR_SetUserData(ULONG lUserId,char * pBuffer,int len);

参数：
lUserID:登录设备时返回的唯一标记值

pBuffer：要保存的数据，只能是字符串,对于二进制数据或是带有不可打印字符等字符时，要先使用%02d进行转成字符串，然后读取出来再转成二进制数据。
len:数据长度

返回值：
0:表示成功加入队列,保存结果将由辅助通道回调函数返回
-1: pBuffer为NULL
ERR_NOT_FIND_DEVICE:表示没有设备登录句柄错误
ERR_DEV_NOT_CONNECTED:表示设备还没连接上
ERR_DEV_NOT_LOGIN:设备还没有认证成功
ERR_OUTOFF_MEMORY:内存不足

读取成功后，将由IP_NET_DVR_SetAUXResponseCallBack所设置的回调函数上，接收到一个nType=1034的事件

IP_NET_DVR_GetUserData 读取用户保存的数据

函数原型：
```
LONG IP_NET_DVR_GetUserData(ULONG lUserId,void * pOutBuffer,int* nInOutLen);
```
参数：
lUserID:登录设备时返回的唯一标记值
void * pOutBuffer：数据保存位置,如果pOutBuffer=NULL，则不等待立即返回
int* nInOutLen：调用前为pOutBuffer可用大小，调用后为实际写入数据字节。如果* nInOutLen<=0则不等待立即返回

返回值：
0表示成功加入发送队列

注意：
读取成功后，将由IP_NET_DVR_SetAUXResponseCallBack所设置的回调函数上，接收到一个nType=1034的事件，里面的XML格式为
<RESPONSE_PARAM UserDevData="616263646566"/>其中，UserDevData即为用户所设置的数据经过%02x进行转换的结果。实际使用时，还要将此串进行第二位，按16进制进行转成为一个字符，例如616263646566转换后是串 abcdef,即abcdef才是用户实际设置的数据。

使用 NETSDK 的基本步骤

使用 NETSDK 注意事项

为比较好的使用此 SDK，应该注意以下事项：

1、 对于所有回调函数，应该只进行简单分析，然后将处理交给其它线程，不应该在回调函数上直接处理。
2、 对于辅助通道操作（如云台，配置设备，前端设备系统控制）相关系统返回成功，都只是表示此任务已向设备请求，至于结果都会通过状态回调函数或是辅助通道消息回调函数进行通知。
3、 使用 SDK 前应该先调用一次 IP_NET_DVR_Init,为快速退出 SDK 相关线程并释放资源，在不再使用时，应该调用 IP_NET_DVR_Cleanup，而不应该对每个设备调用 IP_NET_DVR_StopRealPlay 和 IP_NET_DVR_Logout.如果只是停止所有流，可以调用 IP_NET_DVR_StopAllRealPlay,如果只登出所有设备，可以调用 IP_NET_DVR_LogoutAll.
4、 对于 C#等语言(非 C++)，函数字义时，应该将 char * 全部转换成指针(C#为 IntPtr)，而不应该转成 string.
5、 对于媒体流解码时，如果用户自己解码，则要注意，视频流的后面 16 个字节要去掉（具体规则可以参见 IP_NET_DVR_SetRealDataCallBack），且应该进行丢帧处理，否则有可能会出现花屏或是解码库报错。

实现播放视频流步骤

1、调用 IP_NET_DVR_Init 初始化 NETSDK
2、调用 IP_NET_DVR_SetStatusEventCallBack 设置状态改变回调函数
3、调用 IP_NET_DVR_SetRealDataCallBack 设置收到媒体数据回调函数（如果第 4 步中回调函数不为空，则此步可以省略）
4、调用 IP_NET_DVR_RealPlayEx 请求视频流
5、当在状态回调函数中收到 EVENT_RECVVIDEOPARAM 和 EVENT_RECVAUDIOPARAM 事件时，表示已经收到视频/音频编码参数。6、当收到媒体数据时，调用者区分视频/音频数据传出去进行处理，不要阻塞回调函数。
7、调用 IP_NET_DVR_StopRealPlay 停止收流（如果只停某一路流时调用）
10、调用 IP_NET_DVR_Cleanup 释放 NETSDK 控件资源

注意：
退出应用程序时，请调用 IP_NET_DVR_Cleanup 进行释放。

实现辅助通道连接进行云台控制、系统控制、设备配置等

1、调用 IP_NET_DVR_Init 初始化 NETSDK
2、调用 IP_NET_DVR_SetStatusEventCallBack 设置状态改变回调函数
3、调用 IP_NET_DVR_SetAUXResponseCallBack 设置辅助通信消息回调函数
4、调用 IP_NET_DVR_Login 登录设置

5、当在状态回调函数中，收到 EVENT_LOGINOK 事件时，表示登录成功。
6、登录成功后即可调用相当函数，如基本云台控制函数 IP_NET_DVR_PTZControl 进行控制。
7、调用 IP_NET_DVR_Logout 注销登录。
8、调用 IP_NET_DVR_Cleanup 释放 NETSDK 控件资源

注意事项：

对于云台控制，都会在状态回调函数上返回一个 EVENT_PTZPRESETINFO 事件，对于读取配置、修改配置、系统控制等操作时，将会在辅助通道消息回调函数上返回处理结果:LONG OnAUXResponseCallBack(LONG lUser,LONG nType,char *pResponse,void *pUser);其中，参数意义：

lUser：登陆设备时返回的参数，用于标识来自哪个设备

nType：表示消息类型，nType的最高字节为状态标记位，(即nType & 0xff000000!=0)，当标记位为非0时，表示操作失败。

低24位（即Type=nType & 0xffffff）为真实类型，不同值表示不同操作，详情见附表2

pUser：用户指针，与设置此回调时的pUser参数相同。

实现音频对讲基本流程

1、调用 IP_NET_DVR_SetStatusEventCallBack 设置状态回调函数
2、调用登录设备函数 m_nLastCmdId=IP_NET_DVR_Login
3、调用启动设备对讲函数 IP_NET_DVR_StartVoiceCom(LONG lUserID, int AudioType, int iSampleRate, int iChannel);
如果送入是编码后的 G711U/G711A 数据，需要与前端设备音频编码类型保持一致，否则前端设备播放不正常。
如果送入 16 位的 PCM 数据，AudioType 需要填写为 AudioType_PCM。建议直接送入 16 位 PCM 数据。
4、在状态回调函数上等待以下事件
EVENT_STARTAUDIOOK,//对讲启动成功
EVENT_STARTAUDIOFAILED,//对讲启动失败
如果为EVENT_STARTAUDIOOK则，调用IP_NET_DVR_AddTalk(m_nLastCmdId)
5、调用IP_NET_DVR_InputAudioData(LONG lUserID, const char* pBuffer, int nSize)送入音频数据，重复调用。
6、调用IP_NET_DVR_StopVoiceCom停止并释放资源

云台控制步骤

1、调用 IP_NET_DVR_SetStatusEventCallBack 设置状态回调函数
2、调用 IP_NET_DVR_SetAUXResponseCallBack 设置辅助通道消息回调函数
3、调用登录设备函数 m_nLastCmdId=IP_NET_DVR_Login
4、当在状态回调函数中，收到 EVENT_LOGINOK 事件时，表示登录成功。
5、调用云台控制命令(IP_NET_DVR_PTZControl 或 IP_NET_DVR_PTZControlEx)

附录一 云台控制 XML 命令定义

1. 云台控制说明

本文主要定义 IPC 和辅助通道（云台）控件之间云台控制的 XML 命令。由于市场上存在多种云台命令，我司在 Camera 底层实现多种协议，并把这些协议命令抽象定义成通用的 XML 格式，上层应用可使用通用的云台 XML 命令访问 Camera 连接的云台设备，以保持上层应用程序的一致性。

通过调用NETSDK中的API函数 IP_NET_DVR_PTZControlEx(LONG lUser,char *pXml); 进行执行对应云台命令。

2. XML 命令

我司定义 XML 命令主要参照 PELCO-P，由此扩展到多种云台。由于不同厂家的不同设备协议的细微差别，我司不保证支持所有云台设备。
发送的命令，都是以其它方式<xml></xml>节点，里面包括一个<cmd></cmd>表示不同的命令，每个命令具体查看 2.1

2.1 普通命令

2.1.1 上（UP）

```
<xml>
<cmd>up</cmd>
<panspeed>5</panspeed>
<tiltspeed>5</tiltspeed>
</xml>
```

panspeed: 取值为 1 至 10,值越大，云台水平方向转动得越快，此值对 up 命令无效

tiltspeed:取值为 1 至 10,值越大，云台垂直方向转动得越快

2.1.2 下（down）

```
<xml>
<cmd>down</cmd>
<panspeed>5</panspeed>
<tiltspeed>5</tiltspeed>
</xml>
```

panspeed: 取值为 1 至 10,值越大，云台水平方向转动得越快

tiltspeed:取值为 1 至 10,值越大，云台垂直方向转动得越快

2.1.3 左（left）

```
<xml>
<cmd>left</cmd>
<panspeed>5</panspeed>
<tiltspeed>5</tiltspeed>
</xml>
```

panspeed: 取值为 1 至 10,值越大，云台水平方向转动得越快

tiltspeed:取值为 1 至 10,值越大，云台垂直方向转动得越快

2.1.4 右（right）

```
<xml>
<cmd>right</cmd>
<panspeed>5</panspeed>
<tiltspeed>5</tiltspeed>
</xml>
```

panspeed: 取值为 1 至 10,值越大，云台水平方向转动得越快

tiltspeed:取值为 1 至 10,值越大，云台垂直方向转动得越快

2.1.5 左上

```
<xml>
<cmd>left_up</cmd>
<panspeed>5</panspeed>
<tiltspeed>5</tiltspeed>
</xml>
```

panspeed: 取值为 1 至 10,值越大，云台水平方向转动得越快

tiltspeed:取值为 1 至 10,值越大，云台垂直方向转动得越快

2.1.6 左下

```
<xml>
<cmd>left_down</cmd>
<panspeed>5</panspeed>
<tiltspeed>5</tiltspeed>
</xml>
```

panspeed: 取值为 1 至 10,值越大，云台水平方向转动得越快

tiltspeed:取值为 1 至 10,值越大，云台垂直方向转动得越快

2.1.7 右上

```
<xml>
<cmd>right_up</cmd>
<panspeed>5</panspeed>
<tiltspeed>5</tiltspeed>
</xml>
```

panspeed: 取值为 1 至 10,值越大，云台水平方向转动得越快

tiltspeed:取值为 1 至 10,值越大，云台垂直方向转动得越快

2.1.8 右下

```
<xml>
<cmd>right_down</cmd>
<panspeed>5</panspeed>
<tiltspeed>5</tiltspeed>
</xml>
```

panspeed: 取值为 1 至 10,值越大，云台水平方向转动得越快

tiltspeed:取值为 1 至 10,值越大，云台垂直方向转动得越快

2.1.9 停止 （stop）

```
<xml>
<cmd>stop/cmd>
</xml>
```

2.1.10 视场大（zoomwide）

```
<xml>
<cmd>zoomwide</cmd>
</xml>
```

2.1.11 视场小（zoomtele）

```
<xml>
<cmd>zoomtele</cmd>
</xml>
```

2.1.12 自动聚焦（FocusAutoOn）

```
<xml>
<cmd>FocusAutoOn</cmd>
</xml>
```

2.1.13 手动聚焦（FocusAutoOff）

```
<xml>
<cmd> FocusAutoOff</cmd>
</xml>
```

2.1.14 聚焦-（focusnear）

```
<xml>
<cmd>FocusNearAutoOff</cmd>
</xml>
```

说明：要执行聚焦-之前，要先设置为手动聚焦模式。

2.1.15 聚焦+（focusfar）

```
<xml>
<cmd>FocusFarAutoOff</cmd>
</xml>
```

说明：要执行聚焦+之前，要先设置为手动聚焦模式。

2.1.16 自动光圈（IrisAutoOn）

```
<xml>
<cmd>IrisAutoOn</cmd>
</xml>
```

2.1.17 手动光圈（IrisAutoOff）

```
<xml>
<cmd>IrisAutoOff</cmd>
</xml>
```

2.1.18 光圈-（irisclose）

```
<xml>
<cmd>IrisCloseAutoOff</cmd>
</xml>
```

说明：要执行光圈-之前，要先设置为手动光圈模式。

2.1.19 光圈+（irisopen）

```
<xml>
<cmd>IrisOpenAutoOff</cmd>
</xml>
```

说明：要执行光圈＋之前，要先设置为手动光圈模式。

2.2 巡视和线扫

2.2.1 巡视（Orbit）

```
<xml>
<cmd>Orbit</cmd>
</xml>
```

2.2.2 线扫起点（ScanBegin）

```
<xml>
<cmd>ScanBegin</cmd>
</xml>
```

2.2.3 线扫终点（ScanEnd）

```
<xml>
<cmd>ScanEnd</cmd>
</xml>
```

2.2.4 线扫开（ScanOn）

```
<xml>
<cmd>ScanOn</cmd>
</xml>
```

2.2.5 线扫关（ScanOff）

```
<xml>
<cmd>ScanOff</cmd>
</xml>
```

2.3 预置点命令

2.3.1 添加预制点

```
<xml>
<cmd>setpreset</cmd>
<preset>presetnum</preset>
<r>
<b>bhour:bmin</b>
<e>ehour:emin</e>
</r>
</xml>
```

说明：

1) presetnum 为 1－66 内的整数

2) <r></r>为设置在设定的时间段内，5 分钟没有云台操作，则自动回复到该预制点。如无需此功能，可没有这一段. 中为开始时刻，<e></e>中为结束时刻
3) 只有 1－8 预制点可以设置定时归位功能

2.3.2 删除预制点

```
<xml>
<cmd>clearpreset</cmd>
<preset>presetnum</preset>
</xml>
```

说明：presetnum 为 1－66 内的整数

2.3.3 查询预制点

```
<xml>
<cmd>callpreset</cmd>
<preset>presetnum</preset>
</xml>
```

说明：presetnum 为 1－66 内的整数（67 至 255 可以调用查询实现某些特定功能，视具体球机而定）

2.3.4 添加定时巡视设置

```
<xml>
<cmd>SetOrbit</cmd>
<preset>presetnum</preset>
<r>
<b>bhour:bmin</b>
<e>ehour:emin</e>
</r>
</xml>
```

说明：

1） 该功能设置在设定的时间段内，5 分钟没有云台操作，则自动进行巡视。
2） 若该段时间内有 2.1 所设置的定时归位预制点，则先归位到预制点，再进行巡视。
3） presetnum 可以为 1－8 之间的整数
4） 00:00中为开始时刻，<e>23:59</e>中为结束时刻
5） 设置的时间段不能与其他 presetnum 所设置的重叠，也不能与下一节中的定时线扫中的设置重叠。

注意：

时分，都用二位表示。

2.3.5 添加定时线扫设置

```
<xml>
<cmd>SetScan</cmd>
<preset>presetnum</preset>
<r>
<b>bhour:bmin</b>
<e>ehour:emin</e>
</r>
</xml>
```

说明：

1） 功能设置在设定的时间段内，5 分钟没有云台操作，则自动进行线扫。

2） 若该段时间内有 2.1 所设置的定时归位预制点，则先归位到预制点，再进行线扫。

3） presetnum 可以为 1－8 之间的整数

4） 中为开始时刻，<e></e>中为结束时刻

5） 设置的时间段不能与其他 presetnum 所设置的重叠，也不能与上一节中的定时巡视中的设置重叠。

注意：

时分，都用二位表示

2.3.6 删除定时巡视设置

```
<xml>
<cmd>DelOrbit</cmd>
<preset>presetnum</preset>
</xml>
```

说明：presetnum 为 1－8 之间的整数

2.3.7 删除定时线扫设置

```
<xml>
<cmd>DelScan</cmd>
<preset>presetnum</preset>
</xml>
```

说明：presetnum 为 1－8 之间的整数

2.4 高级命令

2.4.1 黑白模式

```
<xml>
<cmd>BLCOff</cmd>
```

</xml>

2.4.2 彩色模式

<xml>
<cmd>BLCOn</cmd>
</xml>

2.4.3 镜像关

<xml>
<cmd>MirrorOff </cmd>
</xml>

2.4.4 镜像开

<xml>
<cmd>MirrorOn</cmd>
</xml>

2.4.5 图像冻结关

<xml>
<cmd>FreezeOff</cmd>
</xml>

2.4.6 图像冻结开

<xml>
<cmd>FreezeOn</cmd>
</xml>

2.4.7 数字变倍关

<xml>
<cmd>StepsZOff</cmd>
</xml>

2.4.8 数字变倍开

```
<xml>
<cmd>StepsZOn</cmd>
</xml>
```

2.4.9 屏显关

```
<xml>
<cmd>ScreenOff</cmd>
</xml>
```

2.4.10 屏显开

```
<xml>
<cmd>ScreenOn</cmd>
</xml>
```

2.4.11 低照度关

```
<xml>
<cmd>Min.SenOff</cmd>
</xml>
```

2.4.12 低照度开

```
<xml>
<cmd>Min.SenOn</cmd>
</xml>
```

2.4.13 背光补偿关

```
<xml>
<cmd>ILLOff</cmd>
```

</xml>

2.4.14 背光补偿开

<xml>
<cmd>ILLOn</cmd>
</xml>

2.4.15 摄像机复位

<xml>
<cmd>CamReset</cmd>
</xml>

2.4.16 白平衡自动

<xml>
<cmd>AWBOn</cmd>
</xml>

2.4.17 白平衡手动

<xml>
<cmd>AWBOff</cmd>
</xml>

2.4.18 白平衡 R

<xml>
<cmd>RWB</cmd>
</xml>

说明：执行白平衡 R 之前要先设置到白平衡手动状态

2.4.19 白平衡 B

<xml>
<cmd>BWB</cmd>

</xml>

说明：执行白平衡 R 之前要先设置到白平衡手动状态

2.4.20 白平衡 M

<xml>
<cmd>MWB</cmd>
</xml>

说明：执行白平衡 R 之前要先设置到白平衡手动状态

2.4.21 白平衡 G

<xml>
<cmd>GWB</cmd>
</xml>

说明：执行白平衡 R 之前要先设置到白平衡手动状态

2.4.22 初始设置

<xml>
<cmd>InitSet</cmd>
</xml>

2.4.23 控制菜单

<xml>
<cmd>Menu</cmd>
</xml>

透明传输数据

<xml>
<cmd>trans</cmd>
<data>data="xxx" /* 二进制字符， 0xa0 为 a0 */
</xml>

附录二　辅助通道高级功能接口说明

1、辅助通道实现的功能

辅助通道控件除了可以实现云台控制以及反向音频之外，还能实现复杂的系统控制功能，包括摄像头配置参数的查询，设置，以及其他包括录像和日志文件查询，文件上传（配置文件，系统固件升级），文件下载（配置文件，日志文件，录像文件），删除，系统重启，系统参数（功能表，版本信息，网络参数），存储设备格式化等复杂功能。简而言之，目前网页上能够完成的功能，在客户端上通过调用辅助通道控件的响应接口，基本都能实现。

2、辅助通道功能的调用

要配置IPC，必须先定义一个回调用函数LONG CALLBACK OnAUXResponse(LONG lUser,LONG nType,char *pResponse,void *pUser);然后调用IP_NET_DVR_SetAUXResponseCallBack(OnAUXResponse,NULL);当对设备进行读取和配置时，将使用异步回调返回。
其中,lUser表示对应的设备句柄，nType&0xff000000如果不为0，则表示操作结果是否成功，ntype&0x00ffffff为对应命令号。对于不同的配置，所使用的命令不相同。

读取配置时，将使用函数 IP_NET_DVR_GetDVRConfig API函数进行读取，其中dwCommand即为对应宏（宏定义特点是宏名GET），对设备里进配置时，将调用 IP_NET_DVR_SetDVRConfig API函数配置，其中dwCommand即为对应宏(宏定义特别是宏名带有SET),对系统进行控制，则调用 IP_NET_DVR_SystemControl 函数实现。

#define　CMD_SYSTEM_CONFIG_BASE 200

以下，仅假定 m_nLastCmdId 为设备登录时返回的控制句柄。

3、系统配置信息读取&设置命令

调用 IP_NET_DVR_GetDVRConfig 或者 IP_NET_DVR_SetDVRConfig 命令后，将会在 OnAuxResponse(回调函数)收到一条对应的命令，其中它的 cmdId 的前一个字节，即 cmdId&0xff000000 如果为 0 则表示命令成功，如果为非为则表示失败，同时 cmdId&0xffffff 为相关命令值。
IP_NET_DVR_GetDVRConfig 和 IP_NET_DVR_SetDVRConfig 需要输入具体的命令序号，以及设置的具体 XML 文本。
NETSDK 提供 API 将 XML 与结构体之间的转换。

获取配置流程：调用 IP_NET_DVR_GetDVRConfig，或者具体某项配置的 API，然后在回调中处理接收到的 XML 文本，调用 API 将 XML 转换成结构体。
设置配置流程：填写 XML 或者调用 API 将结构体转换成 XML 后调用 IP_NET_DVR_SetDVRConfig；或者直接调用 API 来设置具体某项配置，参数为结构体。

4、系统控制命令

```
#define CMD_SYSTEM_MANAGE_BASE  1000
```

调用 IP_NET_DVR_SystemControl 命令后，将会在 OnAuxResponse(回调函数)收到一条对应的命令，其它 cmdId 的前一个字节，即 cmdId&0xff000000 如果为 0 则表示命令成功，如果为非为则表示失败，同时 cmdId&0xffffff 为相关命令值。

读取版本信息(1012 命令)

```
#define CMD_SYSTEM_MANAGE_BASE  1000
#define CMD_GET_SYSTEM_VERSION_INFO  (CMD_SYSTEM_MANAGE_BASE + 12)
```

调用方法：

```
IP_NET_DVR_GetDVRConfig(m_nLastCmdId, CMD_GET_SYSTEM_VERSION_INFO,0,"", pXml,0);
```

返回 XML：

```
    Linux 2.6.18_arm_v5t_le #1 PREEMPT Thu Nov 18 11:30:21 GMT+8 2010
```

恢复默认配置(1002 命令)

```
#define   CMD_CONFIG_RESTORE    (CMD_SYSTEM_MANAGE_BASE + 2)
```

调用方法：

```
IP_NET_DVR_SystemControl(m_nLastCmdId, CMD_CONFIG_RESTORE,0,” ”)
```

返回，错误标记为 0 则表示成功

修改系统时间(1005 命令)

```
#define   CMD_SET_SYSTEM_TIME   (CMD_SYSTEM_MANAGE_BASE + 5)
```

调用方法：

```
IP_NET_DVR_SystemControl(m_nLastCmdId, CMD_CONFIG_RESTORE,0,xmldata)
```

其中 xmldata 格式为：

```
<REQUEST_PARAM
Time="20120424 17:45:01"    //时间格式为 yyyyMMdd HH:mm:ss
TimeZone="4800"   //时区，时区 a 对应的计算公式为(12+a)*60,其中,a 为-12 至+12
/>
```

返回，错误标记为 0 则表示成功

注意：
此控制命令只有交时间设成手动修改时才有效。

重启系统(1007 命令)

#define CMD_SYSTEM_REBOOT (CMD_SYSTEM_MANAGE_BASE + 7)

调用方法：

IP_NET_DVR_SystemControl(m_nLastCmdId, CMD_SYSTEM_REBOOT,0,"")

返回:
错误标记为 0 则表示成功

注意：
建议使用 IP_NET_DVR_RebootDVR

获取存储设备信息(1014 命令)

#define CMD_GET_STORAGE_INFO (CMD_SYSTEM_MANAGE_BASE + 14)

调用方法：
IP_NET_DVR_SystemControl(m_nLastCmdId, CMD_GET_STORAGE_INFO,0,"")

返回 Xml 为:

<RESPONSE_PARAM>
<DeviceInfo
DevName="sd1" //可能是 sd1,sd2,usb，分别表示不同存储设备
Mounted="0" //是否已加载
Total="0" //可用存储空间大小，如果为 0 则表示此设备不存在
Used="0" //已使用空间大小
Free="0" //可用空间大小
Percent="0" //已使用空间比例
/>
<DeviceInfo
DevName="sd2"
Mounted="0" //是否已加载
Total="0" //可用存储空间大小，如果为 0 则表示此设备不存在
Used="0" //已使用空间大小
Free="0" //可用空间大小
Percent="0" //已使用空间比例
 />
<DeviceInfo
DevName="usb"
Mounted="0" //是否已加载
Total="0" //可用存储空间大小，如果为 0 则表示此设备不存在
Used="0" //已使用空间大小
Free="0" //可用空间大小
Percent="0" //已使用空间比例

/>
</RESPONSE_PARAM>

格式化前端存储设备(1017 命令)

#define CMD_STORAGE_DEVICE_FORMAT (CMD_SYSTEM_MANAGE_BASE + 17)

调用方法：

IP_NET_DVR_SystemControl(m_nLastCmdId, CMD_STORAGE_DEVICE_FORMAT,0, devType)

devType 可能是 sd1,sd2,usb

返回，错误标记为 0 则表示成功

卸载前端存储设备(1018 命令)

#define CMD_STORAGE_DEVICE_UNMOUNT (CMD_SYSTEM_MANAGE_BASE + 18)

调用方法：

IP_NET_DVR_SystemControl(m_nLastCmdId, CMD_STORAGE_DEVICE_UNMOUNT,0,devType)

devType 可能是 sd1,sd2,usb

返回，错误标记为 0 则表示成功

获取设备系列号信息(1019 命令)

#define CMD_GET_SERIALNUMBER (CMD_SYSTEM_MANAGE_BASE + 19)

调用方法：

IP_NET_DVR_SystemControl(m_nLastCmdId, CMD_GET_SERIALNUMBER,0,"")

返回 Xml 为:

<RESPONSE_PARAM SerialNumber="4756736557496768" />

其中系列为：475673655749676

读取系统能力集(1020 命令)

#define CMD_GET_SYSTEMCONTROLSTRING (CMD_SYSTEM_MANAGE_BASE + 20)

调用方法：

IP_NET_DVR_SystemControl(m_nLastCmdId, CMD_GET_SYSTEMCONTROLSTRING,0,"")

返回 Xml 为:

<RESPONSE_PARAM SystemConfigString=" + audio_support + schedule_record + ftpemail_storage + picture_capture + ptz_control + gpio_input + gpio_output + zh_cn + en_us + zh_tw + front_replay + media_capabiltiy" />

```
audio_support  //支持音频
schedule_record  //支持定时录像
ftpemail_storage  //支持 ftp 和 email
picture_capture  //支持抓图
ptz_control   //支持云台
gpio_input    //支持 IO 输入
gpio_output   //支持 IO 输出
zh_cn   //设备配置界面是否支持中文
en_us   //设备配置界面是否支持英文
zh_tw   //设备配置界面是否支持繁体
front_replay       //支持前端在线回放
media_capabiltiy //可以查询设备支持的分辨率
wireless_accesspoint    //支持 WIFI 作为客户端功能
wireless_station //支持 WIFI 作为服务端功能
network_storage //支持网络存储
ptz_action    //支持云台联动
ircut_setting      //支持 IRCUT 功能
evdo_support  //支持 EVDO 3G 功能
wcdma_support      //支持 WCDMA 3G 功能
tdscdma_support //支持 TDSCDMA 3G 功能
```

设备录像文件搜索（图片或录像文件）(1021 命令)

#define CMD_GET_RECORD_FILE_LIST (CMD_SYSTEM_MANAGE_BASE + 21)

调用方法：

IP_NET_DVR_SystemControl(m_nLastCmdId, CMD_GET_RECORD_FILE_LIST,0,searchxml)

其中 searchxml 格式为：

```
<REQUEST_PARAM
    RecordMode="MOTION"     //见下面
StartTime="20090101 0:00:00"    //搜索开始时间
EndTime="20090102 0:00:00" //搜索结束时间，注意开始时间到结束时间最多为 7 天
    MediaType="PICTURE"     // 见下表
    StreamIndex="1"  //流 1 为主码流，2 为子码流
    MinSize="-1"       //搜索文件不小于，当为-1 为无限制，单位字节
    MaxSize="-1"       //搜索文件不大于，当为-1 为无限制，单位字节
```

```
    Page="1"   //搜索第几页，当为多页时，可以使用此值控制搜索第几页，每页 20 条记录
/>
```

RecordMode 可能值及意义为
- MANUAL:手动触发录像（不支持）
- MOTION:移动侦测录像
- SCHEDULE:按时间表录像
- ALARM:IO 口告警录像

MediaType 可能值及意义为
- AUDIOVIDEO: 音频视频
- VIDEO:仅视频
- AUDIO：仅音频（不支持）
- PICTURE:拍照

返回 Xml 为:

```
<RESPONSE_PARAM>
<RECORD_FILE
FilePath="/mnt/mmc0/motion/2009/04/01/103222-av-1.avi" //文件名
FileLength="14832236"       //文件大小
/>
<RECORD_FILE
FilePath="/mnt/mmc0/motion/2009/04/01/103107-av-1.avi"
FileLength="14463472"
/>
</RESPONSE_PARAM>
```

文件上传（配置文件和固件文件）(1022 命令)

```
#define    CMD_UPLOAD_FILE   (CMD_SYSTEM_MANAGE_BASE + 22)
```

调用方法：

IP_NET_DVR_SystemControl(m_nLastCmdId, CMD_UPLOAD_FILE,0,uploadxml)

其中 uploadxml 格式为：

```
        <REQUEST_PARAM
         FileType = "/mnt/mmc0/motion/2009/04/01/103222-av-1.avi"  //0 为配置文件，1 为固件文件
         FilePath ="本地文件路径"   //放在计算机上的文件,注意本地文件路径名，如果带特殊字符，则要进行 XML 转义,主要可
    能有 ",& 字符
        />
```

注意：

不建议使用此命令，直接调用 IP_NET_DVR_Upgrade IP_NET_DVR_SetConfigFile 等相关函数

文件下载(1023 命令)

#define CMD_DOWNLOAD_FILE (CMD_SYSTEM_MANAGE_BASE + 23)

调用方法：

IP_NET_DVR_SystemControl(m_nLastCmdId, CMD_DOWNLOAD_FILE,0,downloadxml)

其中 downloadxml 为：

```
<REQUEST_PARAM
FileName = "/mnt/mmc0/motion/2009/04/01/103222-av-1.avi" //存储在设备上的文件路径，搜索文件时返回
StartPos="0"  //开始下载位置，当前只能为 0
/>
```

注意：

不建议使用此命令,直接调用提供的下载 API 函数， IP_NET_DVR_GetFileByName 等相关函数

删除文件(1024 命令)

#define CMD_REMOVE_FILE (CMD_SYSTEM_MANAGE_BASE + 24)

调用方法：

IP_NET_DVR_SystemControl(m_nLastCmdId, CMD_REMOVE_FILE,0,removexml)

其中 removexml 为：

```
<REQUEST_PARAM
FileName = "/mnt/mmc0/motion/2009/04/01/103222-av-1.avi" //存储在设备上的文件路径，搜索文件时返回
/>
```

搜索日志文件(1025 命令)

#define CMD_GET_LOGFILE_LIST (CMD_SYSTEM_MANAGE_BASE + 25)

调用方法：

IP_NET_DVR_SystemControl(m_nLastCmdId, CMD_GET_LOGFILE_LIST,0, "")

返回 XML 格式为:

```
<RESPONSE_PARAM>
<LOG_FILE
FilePath="/mnt/nand/20090401.log"
/>
<LOG_FILE
FilePath="/mnt/nand/20010101.log"
/>
```

```
<LOG_FILE
FilePath="/mnt/nand/20001231.log"
/>
</RESPONSE_PARAM>
```

读取设备类型(1030 命令)

```
#define CMD_GET_IPC_TYPE     (CMD_SYSTEM_MANAGE_BASE + 30)
```

调用方法：

IP_NET_DVR_SystemControl(m_nLastCmdId, CMD_GET_IPC_TYPE,0, "")

返回 XML 格式为:

```
<RESPONSE_PARAM
IPC_Type="100"     //可能值有 100，101，102，103
/>
```

强制设备产生关键帧(1032 命令)

```
#define  CMD_SET_KEYFRAME (CMD_SYSTEM_MANAGE_BASE  +  32)
```

调用方法：

IP_NET_DVR_SystemControl(m_nLastCmdId, CMD_SET_KEYFRAME,0, Createiframxml)

其中 Createiframxml 格式为:

```
    <REQUEST_PARAM
    Camera="0"    //固定为 0
    Stream="0"    //0 表示主码流，1 表示子码流
    />
```

注意：

不建议直接调节此命令，可以调用 IP_NET_DVR_CreateIFrame 代替

设置用户存储数据 (1033 命令)

```
#define  CMD_SET_USER_DEV_DATA (CMD_SYSTEM_MANAGE_BASE  +  33)
```

调用方法：

IP_NET_DVR_SystemControl(m_nLastCmdId, CMD_SET_USER_DEV_DATA,0, Createiframxml)

其中 Createiframxml 格式为:

```
<REQUEST_PARAM
UserDevData="0102000012301230123012301230123" //用户存储的数据
/>
```

建议：

保存数据时，将数据使用 sprintf(str,"%02x",c);进行转换成一串数据，然后进行保存，再次读取时，将其它转换回来。

可直接调用 IP_NET_DVR_SetUserData 来实现此功能

读取用户存储数据 (1034 命令)

```
#define CMD_GET_USER_DEV_DATA (CMD_SYSTEM_MANAGE_BASE + 34)
```

调用方法：

IP_NET_DVR_SystemControl(m_nLastCmdId, CMD_GET_USER_DEV_DATA,0, Createiframxml)

其中 Createiframxml 格式为:

```
<REQUEST_PARAM
Camera="0"
Stream="0"    //0 表示主码流，1 表示子码流
/>
```

注意：

可直接调用 IP_NET_DVR_GetUserData 来实现此功能

IO 口控制(1042 命令)

```
#define CMD_ALARM_OUTPUT_CONTROL (CMD_SYSTEM_MANAGE_BASE + 42) //added by johnnyling 20100909
```

调用方法：

IP_NET_DVR_SystemControl(m_nLastCmdId, CMD_ALARM_OUTPUT_CONTROL,0, Createiframxml)

其中 Createiframxml 格式为:

```
<REQUEST_PARAM
ChannelNo ="IO 通道号，从 1 开始“
Output ="0 或者 1"
/>
```

目前只支持一个 IO 口。

注意：后期 IPC 版本支持此命令

图片抓拍 (1043 命令)

```
#define CMD_SNAP_JPEG_PICTURE (CMD_SYSTEM_MANAGE_BASE + 43) //added by johnnyling 20120330
```

调用方法：

IP_NET_DVR_SystemControl(m_nLastCmdId, CMD_SNAP_JPEG_PICTURE,0, Createiframxml)
其中 Createiframxml 格式为:

```
<REQUEST_PARAM
Stream="0 或者 1 “
Quality="50-100"
PictureId="XXX"
/>
```

PictureId 可以不传。

返回 XML 格式为:

```
<RESPONSE_PARAM>
JpgFile="文件名"
</RESPONSE_PARAM>
```

拍照是异步操作，返回应答并不表示拍照完成。
拍照完成后，IPC 会通过事件通过客户端，事件编号：ALARM_CODE_JPEG_CAPTURED

完整事件编号定义如下：

```
typedef enum {
    ALARM_CODE_BEGIN=0,
    ALARM_CODE_LINKDOWN=1,
    ALARM_CODE_LINKUP,
    ALARM_CODE_USB_PLUG,
    ALARM_CODE_USB_UNPLUG,
    ALARM_CODE_SD0_PLUG,
    ALARM_CODE_SD0_UNPLUG,
    ALARM_CODE_SD1_PLUG,
    ALARM_CODE_SD1_UNPLUG,
    ALARM_CODE_USB_FREESPACE_LOW,
    ALARM_CODE_SD0_FREESPACE_LOW,
    ALARM_CODE_SD1_FREESPACE_LOW,
    ALARM_CODE_VIDEO_LOST,
    ALARM_CODE_VIDEO_COVERD,
    ALARM_CODE_MOTION_DETECT,
    ALARM_CODE_GPIO3_HIGH2LOW,
    ALARM_CODE_GPIO3_LOW2HIGH,
    ALARM_CODE_STORAGE_FREESPACE_LOW,
    ALARM_CODE_RECORD_START,
    ALARM_CODE_RECORD_FINISHED,
    ALARM_CODE_RECORD_FAILED,
    ALARM_CODE_GPS_INFO,
    ALARM_CODE_EMERGENCY_CALL,
    ALARM_CODE_JPEG_CAPTURED,
    ALARM_CODE_RS485_DATA,
    ALARM_CODE_SAME_IP,
    ALARM_CODE_HW130_PIR,
    ALARM_CODE_END
}AjAlarmCode;
```

事件中的 data 字段，就是完整的拍照文件名。 客户端/NETSDK 需要通过文件下载接口，将该照片下载下来，然后再通过回调方式通知应用层。

获取 IPC 系统时间(1048 命令)

```
#define  CMD_GET_SYSTEM_TIME (CMD_SYSTEM_MANAGE_BASE  +  48)  //added  by  johnnyling  20121127
```

调用方法：
IP_NET_DVR_SystemControl(m_nLastCmdId, CMD_GET_SYSTEM_TIME,0, NULL)

返回 XML 格式为:

```
 <RESPONSE_PARAM
 TimeZone="时区"  Year="年"  Month="月"  Day="日"  Hour="时"  Minute="分"  Second="秒"
 />,
```

获取支持的 PTZ 协议列表（1049 命令）

```
#define CMD_GET_PTZ_CAPABILITY (CMD_SYSTEM_MANAGE_BASE + 49) //added by johnnyling 20121127
```

调用方法：
IP_NET_DVR_SystemControl(m_nLastCmdId, CMD_GET_PTZ_CAPABILITY,0, NULL)

返回 XML 格式为:

```
<RESPONSE_PARAM>
<PTZ_PROTOCOL Name="PECO-P" Type="PECO-P">
<PTZ_PROTOCOL Name="PECO-D" Type="PECO-D">
</RESPONSE_PARAM>
```

5、相关宏定义对照表

参见以下头文件：
cmd_def.h： 命令号
media_cfg.h，data_struct.h：数据结构
function_list.h：能力集定义
NetSDKDLL.h：API 定义