# 网络视频解码器
# 客户端 SDK 开发手册

**(For Windows/Linux)**

**版本：5.1**

# 目 录

# 第 1 章 系统简介

## 1.1 概述

网络视频解码器可以将网络数字信号解为模拟信号，适用于模数结合系统。

**解码器 SDK 主要功能**

解码器 SDK 是专门为网络解码器系列产品提供的软件接口程序,以动态链接库的形式提供给应用软件开发者，并同时附有演示程序及其源码，能有效地缩短应用软件的开发周期。

**SDK 支持的系统**

Windows 下解码器 SDK：

Windows10/Windows8/Windows7/vista/xp/2000 以及 Windows Server 2008/2003 32 位。

Linux 下设备解码器 SDK：

32/64 位的 Linux 系统，版本要求 gcc 4.1 或者 4.1 以上。 测试过的系统：RedHat AS 5/6。

## 1.2 开发包组成

本开发包是 S 系列网络解码器开发手册,手册详细介绍了软件开发包中各个函数所实现的功能以及使用方法、各个函数之间的调用关系。特别强调，凡是接口中涉及到的所有结构体和宏定义都在DecCtrlClientTypes.h或GlobalTypes.h头文件有定义,上层可直接引用。

**解码器 SDK 包含核心功能库，网络库，搜索库，域名解析库，控制库等功能组件，我们提供 Windows 和 Linux 两个版本的 SDK。**

**Windows 下解码器 SDK：**

.\ 库文件\

| | |
|---|---|
| NVDSDK.dll | 开发包解码器核心动态库 |
| OsCore.dll | 开发包底层网络动态库 |
| NetAdmin.dll | 搜索设备动态库 |
| nslook.dll | 域名解析动态库 |
| DOME_PELCO_D.dll | DOME_PELCO_D 控制协议动态库 |
| DOME_Pelco_P.dll | DOME_Pelco_P 控制协议动态库 |
| DOME_PLUS.dll | DOME_PLUS 控制协议动态库 |
| NVDSDK.lib | |

.\ 头文件\

DecCtrlClientTypes.h
DecCtrlClient.h
GlobalTypes.h
ActionControl.h
RetValue.h
x_type.h

.\ 开发文档\

网络视频解码器开发包手册.chm
网络视频解码器开发包手册.pdf

**Linux 下解码器 SDK：**

.\ 库文件\

libnvdsdk.so　　开发包解码器核心动态库
libossdk.so　　开发包底层网络动态库

.\ 头文件\

DecCtrlClientTypes.h
DecCtrlClient.h
GlobalTypes.h
ActionControl.h
RetValue.h
x_type.h

.\ 开发文档\

网络视频解码器开发包手册.chm
网络视频解码器开发包手册.pdf

## 1.3 注意事项

注意事项

1. NVDSDK 采用异步方式实现，登录、连接视频、设置参数操作调用接口成功后还需等待消息或回调后才能执行其他操作。如调用 DEC_ClientLogon 接口成功后需等待系统消息 DEC_WCM_DEC_LOGON 后才能对此设备进行其他操作，否则操作可能失败。

2. NVDSDK 进行回调时会占用内部锁，所以在回调函数中尽量不要使用锁更不要阻塞回调函数，这样会导致死锁或设备掉线。

3. SDK 同时可支持同时登录 128 个设备，至于 push 流路数则受机器 CPU 限制，各不相同。

## 1.4 函数调用顺序说明

1. 初始化开发包

**int　__stdcall Dec_ClientStartup(unsigned int_uiMessage, HWND _hWnd);**

2. 设置开发包需要的一些参数

**int __stdcall DEC_ClientSetNotify(DecNotifyFun* _pNotify);**

3. 调用开发包所提供的其他函数

{……}

4. 释放开发包

**int __stdcall DEC_ClientCleanup()；**

# 第 2 章 版本说明

## 2.1 Version 4.0

1、本版NVDSDK继承了以前版本的所有功能，包括连接/断开视频；建立和取消音频预览；建立和取消对讲；建立和停止推流；解码器参数的设置：包括IP地址设置、485控制协议设置、DDNS设置、报警方式设置、显示Logo设置等；解码器的重启；解码器恢复出厂设置；并支持解码器内核程序、网页程序、Logo、控制协议的远程升级。

2、本版NVDSDK新增功能，包括拼控、日志、集群、预案。

## 2.2 Version 4.3

1、本版NVDSDK继承了以前版本的所有功能，包括连接/断开视频；建立和取消音频预览；建立和取消对讲；建立和停止推流；解码器参数的设置：包括IP地址设置、485控制协议设置、DDNS设置、报警方式设置、显示Logo设置等；解码器的重启；解码器恢复出厂设置；并支持解码器内核程序、网页程序、Logo、控制协议的远程升级、拼控、日志、集群、预案。

2、本版NVDSDK新增功能，包括屏控、本地输入通道、预案切换、SIP通道注册、重启外挂、抓拍、U盘播放等。

## 2.3 Version 5.1

1、本版NVDSDK继承了以前版本的所有功能，包括连接/断开视频；建立和取消音频预览；建立和取消对讲；建立和停止推流；解码器参数的设置：包括IP地址设置、485控制协议设置、DDNS设置、报警方式设置、显示Logo设置等；解码器的重启；解码器恢复出厂设置；并支持解码器内核程序、网页程序、Logo、控制协议的远程升级、拼控、日志、集群、预案。屏控、本地输入通道、预案切换、SIP通道注册、重启外挂、抓拍、U盘播放等。

2、本版NVDSDK新增功能，包括小间距屏设置等，本次提升了兼容性，安全性，内存CPU优化等。

# 第 3 章 函数说明

## 3.1 SDK 初始化

**int __stdcall DEC_ClientStartup(unsigned int_uiMessage, HWND _hWnd);**

功能： 启动 SDK，运行此函数后，就可以发送和接收命令了。

参数： _uiMessage 消息 ID，当有事件发生时，开发包以消息方式通知应用程序，此

为要发送的消息值

_hWnd　　窗口句柄，当有事件发生时，开发包以消息方式通知应用程序，此为接收消息的窗口句柄

返回：　0　　启动成功

-2　　开发包尚未初始化/初始化失败

-3　　传入非法参数

**int　__stdcall DEC_ClientSetSDKWorkMode (int　_iWorkMode);**

功能：设置 SDK 工作模式，如果不调用该接口，SDK 默认重量级工作。

参数：　_iWorkMode　　SDK 工作模式：0--重量级（登陆解码器内存占用较大），1--轻量级（登陆解码器内存占用较小）。

返回：　0　　正确，恒为 0

**int __stdcall DEC_ClientCleanup()**

功能：　释放 SDK，在系统退出前运行此函数，断开所有存在的连接。运行此函数后不能再调用 SDK 中其他接口

参数：　无

返回：　0　　正确停止

-2　　开发包初始化失败

说明：　手册中除特殊说明外，所有函数都必须在 Dec_ClientStartup 和 Dec_ClientCleanup 函数之间调用

**int __stdcall DEC_ClientGetSDKVersion(NVDSDK_VERSION *_ver)**

功能：　获得解码器 SDK 的版本号

参数：　_ver　　指向 NVDSDK_VERSION 的结构指针，返回的版本信息

返回：　0　　正确，恒为 0

**int　__stdcall　DEC_ClientSetComRecvNotify(DEC_COMRECV_NOTIFY　_comRecvNotify);**

功能：　设置回调函数

参数：　_comRecvNotify　　串口接收数据回调

返回：　0　　返回恒为 0

说明：　设置了该回调函数后,从解码器串口输入的数据,被传到客户端后通过该回调函数通知应用程序

回调：

**typedef void (__stdcall *DEC_COMRECV_NOTIFY)(LONG　_ulID, char*　_cBuf, int　_iLength);**

**参数：　_ulID　　解码器标识号**

**_cBuf　　接受数据缓存区**

**_iLength　　缓存区大小**

**int __stdcall DEC_ClientSetMsgCallback(pTDPostMessage _TDPostMsg);**

功能：　向 SDK 注册消息回调，不建议使用该接口

参数：　_TDPostMsg　　pTDPostMessage 类型的函数指针

返回：　0　　成功

<0　　失败，通过错误值查阅本文档错误码定义部分

**int __stdcall DEC_ClientSetMsgCallbackEx(pTDPostMessageEx _TDPostMsg);**
功能：　向 SDK 注册消息回调，不建议使用该接口
参数：　_TDPostMsg　　pTDPostMessageEx 类型的函数指针
返回：　0　　成功
　　<0　　失败，通过错误值查阅本文档错误码定义部分

**int __stdcall DEC_ClientSetNotify(DecNotifyFun* _pNotify);**
功能：　向 SDK 注册消息回调；
参数：　_pNotify　　DecNotifyFun 类型的结构体指针
返回：　0　　成功
　　<0　　失败，通过错误值查阅本文档错误码定义部分

消息回调函数：
**typedef int (*pTDPostMessage)(void *_hWnd, unsigned int _uiMsg, unsigned int _wParam, int _lParam);**

**typedef int (*pTDPostMessageEx)(void *_hWnd, unsigned int _uiMsg, unsigned int _wParam, int _lParam, void* _lpUserData);**

```
typedef struct _tagDecNotifyFun
{
    int iSize;
    ParamChangeNotify pParamChangeNotify;
    MainNotify pMainNotify;
}DecNotifyFun, *pDecNotifyFun;
```

**typedef int (*MainNotify)(unsigned long _lID, unsigned int _uiMsg, unsigned int _iChannel, int _iPos, void* _pvParam, int _iParamSize, void* _pvUserData);**

**typedef int (*ParamChangeNotify)(unsigned long _lID, unsigned int _uiMsg, unsigned int _iChannel, int _iPos, void* _pParam, int _iParamSize, void* _lpUserData);**

## 3.2 登录设备

连接 NVD 时首先需要登录，登录设备成功后即可以获得/设置设备上所有的属性设置。

**long __stdcall DEC_ClientLogon(char* _cIP, char* _cUserName, char* _cPassword, unsigned short _nPort = 3000);**
功能：　登录指定的网络视频解码器
参数：　_cIP　　IP 地址

_cUserName 用户名
_cPassword 口令
_nPort 该解码器所使用的通讯端口，默认为 3000
返回： >=0 操作成功，返回值为连接标识，提供给其他接口函数使用，当调用 DEC_ClientLogout 后，此标识失效，不能再标识当前设备
-2 开发包初始化失败
-3 传入非法参数
-10 达到最大连接数 128
说明： 登录操作成功后并不代表成功登录解码器，需要通过回调或者消息来获得登录状态登录后,会获得一个系统消息（如果设置了消息句柄）,可以从消息判断登录是否成功。如果设置了回调函数，也可以在回调函数内处理登录结果，建议使用消息机制。系统默认登录用户名：DEC，密码：DEC。

**long __stdcall DEC_ClientLogonEx(char* _cIP, char* _cUserName, char* _cPassword, unsigned short _nPort/* = 3000*/, void* _pvUserData/* = NULL*/);**
功能： 登录指定的网络视频解码器
参数： _cIP IP 地址
_cUserName 用户名
_cPassword 口令
_nPort 该解码器所使用的通讯端口，默认为 3000
_pvUserData 用户数据
返回： >=0 操作成功，返回值为连接标识，提供给其他接口函数使用，当调用 DEC_ClientLogout 后，此标识失效，不能再标识当前设备
-2 开发包初始化失败
-3 传入非法参数
-10 达到最大连接数 128
说明： 登录操作成功后并不代表成功登录解码器，需要通过回调或者消息来获得登录状态登录后,会获得一个系统消息（如果设置了消息句柄）,可以从消息判断登录是否成功。如果设置了回调函数，也可以在回调函数内处理登录结果，建议使用消息机制。系统默认登录用户名：DEC，密码：DEC。

**int __stdcall DEC_ClientGetLogonState(long _iID);**
功能： 获取用户登录状态
参数： _iID DEC_ClientLogon 返回的标识
返回： 0 登录成功
1 登录中
2 重新登录
-1 登录失败
-2 登录超时
-3 没有登录
-8 没有调用登录接口

**int __stdcall DEC_ClientLogout(long _ulID);**
功能： 注销用户登录

参数： _uIID DEC_ClientLogon 返回的标识
返回：

| | |
|---|---|
| 0 | 成功 |
| -1 | 指定的解码器不存在 |
| -2 | 开发包尚未初始化/初始化失败 |
| -8 | 没有登录成功 |

**int __stdcall DEC_ClientUserChangePassword(long _uIID, char *_cUser , char *_cOldPassword, char *_cNewPassword);**

功能： 修改解码器的登录密码
参数：

| | |
|---|---|
| _iID | 解码器标识号 |
| _cUser | 登录用户名 |
| _cOldPassword | 旧密码 |
| _cNewPassword | 新密码 |

返回：

| | |
|---|---|
| 0 | 成功 |
| <0 | 失败，通过错误值查阅本文档错误码定义部分 |

## 3.3 连接和断开视频

**int __stdcall DEC_ClientStartView(long _uIID, TNVSITEM *_pPara, int _iChannel = 0, int _iPos = 0);**

功能： 连接网络视频服务器，播放视频
参数：

| | |
|---|---|
| _uIID | 解码器标识号 |
| _pPara | 指向 TNVSITEM 的结构指针 |
| _iChannel | 当前选择通道 |
| _iPos | 当前选择画面 |

返回：

| | |
|---|---|
| 0 | 成功，开始播放视频 |
| -1 | 指定的解码器不存在 |
| -2 | 开发包尚未初始化/初始化失败 |
| -3 | 传入非法参数 |
| -8 | 没有登录 |
| -12 | 不允许操作，例如视频正在循环切换 |

说明： 此接口支持多次调用，将一路视频显示在多个不同的窗口。

**int __stdcall DEC_ClientStartViewEx(int _iID, TNVSITEMEX *_pPara, int _iChannel/* = 0*/, int _iPos/* = 0*/);**

功能： 连接网络视频服务器，播放视频。DEC_ClientStartView 接口的扩展
参数：

| | |
|---|---|
| _uIID | 解码器标识号 |
| _pPara | 指向 TNVSITEM 的结构指针 |
| _iChannel | 当前选择通道 |
| _iPos | 当前选择画面 |

返回：

| | |
|---|---|
| 0 | 成功，开始播放视频 |
| -1 | 指定的解码器不存在 |

-2 开发包尚未初始化/初始化失败
-3 传入非法参数
-8 没有登录
-12 不允许操作，例如视频正在循环切换

说明： 此接口支持多次调用，将一路视频显示在多个不同的窗口。

**int __stdcall DEC_ClientStartView_V1(int _iID, int _iChannel, int _iPos, int _iServerType, ServerItem* _puItem);**

功能： 连接网络视频服务器，播放视频。DEC_ClientStartView 和 DEC_ClientStartViewEx 接口的扩展

参数： _uIID 解码器标识号
_iChannel 当前选择通道
_iPos 当前选择画面
_iServerType 视频源服务器类型，0：私有，1：ONVIF，3：RTSP
_puItem 指向 ServerItem 的结构指针

返回： 0 成功，开始播放视频
-1 指定的解码器不存在
-2 开发包尚未初始化/初始化失败
-3 传入非法参数
-8 没有登录
-12 不允许操作，例如视频正在循环切换

说明： 此接口支持多次调用，将一路视频显示在多个不同的窗口。

**int __stdcall DEC_ClientStopView(long _uIID, int _iChannel = 0, int _iPos = 0,int _iShowLastPic=0);**

功能： 停止视频播放

参数： _uIID 解码器标识号
_iChannel 当前选择通道
_iPos 当前选择画面
_iShowLastPic 停止播放时显示最后一帧画面

返回： 0 成功，停止播放视频
-1 指定的解码器不存在
-2 开发包尚未初始化/初始化失败
-3 传入非法参数
-8 没有登录
-12 不允许操作，例如视频正在循环切换

**int __stdcall DEC_ClientGetViewItem(long _uIID, TNVSITEM *_pPara, int _iChannel = 0, int _iPos = 0);**

功能： 获取当前正在连接的 nvs 信息

参数： _uIID 解码器标识号
_pPara 指向 TNVSITEM 的结构指针
_iChannel 当前选择通道
_iPos 当前选择画面

返回： 1 成功，存在正在连接的 NVS
0 成功，不存在正在连接的 NVS
-1 指定的解码器不存在
-2 开发包尚未初始化/初始化失败
-3 传入非法参数
-8 没有登录

**int __stdcall DEC_ClientGetViewItemEx(int _iID, TNVSITEMEX *_pPara, int _iChannel/* = 0*/, int _iPos/* = 0*/);**
功能： 获取当前正在连接的 nvs 信息，DEC_ClientGetViewItem 接口的扩展
参数： _uIID 解码器标识号
_pPara 指向 TNVSITEM 的结构指针
_iChannel 当前选择通道
_iPos 当前选择画面
返回： 1 成功，存在正在连接的 NVS
0 成功，不存在正在连接的 NVS
-1 指定的解码器不存在
-2 开发包尚未初始化/初始化失败
-3 传入非法参数
-8 没有登录

**int __stdcall DEC_ClientGetViewItem_V1(int _iID, int _iChannel, int _iPos, int* _piServerType, ServerItem* _puItem);**
功能： 获取当前正在连接的 nvs 信息，DEC_ClientGetViewItem
和 DEC_ClientGetViewItemEx 接口的扩展
参数： _uIID 解码器标识号
_iChannel 当前选择通道
_iPos 当前选择画面
_piServerType 视频源服务器类型，0：私有，1：ONVIF，3：RTSP
_puItem 指向 ServerItem 的结构指针
返回： 1 成功，存在正在连接的 NVS
0 成功，不存在正在连接的 NVS
-1 指定的解码器不存在
-2 开发包尚未初始化/初始化失败
-3 传入非法参数
-8 没有登录

## 3.4 音频预览

**int __stdcall DEC_ClientSoundCtrl(long _ullD, int _bOn, int _iChannel = 0, int _iPos = 0);**
功能： 启动/停止解码音频
参数： _uIID 解码器标识号

| | | |
|---|---|---|
| | _bOn | 启动/停止标识：1 表示启动，0 表示停止 |
| | _iChannel | 当前选择通道 |
| | _iPos | 当前选择画面 |
| 返回： | 0 | 成功 |
| | -1 | 指定的解码器不存在 |
| | -2 | 开发包尚未初始化/初始化失败 |
| | -3 | 传入非法参数 |
| | -8 | 没有登录 |
| | -12 | 不允许操作，四画面下仅允许打开一路音频 |

**int __stdcall DEC_ClientGetAudioStatus (long _ullD, int _iChannel = 0, int _iPos = 0);**

| | | |
|---|---|---|
| 功能： | 获取音频状态 | |
| 参数： | _ullD | 解码器标识号 |
| | _iChannel | 当前选择通道 |
| | _iPos | 当前选择画面 |
| 返回： | 0 | 音频关闭 |
| | 1 | 音频打开 |
| | -1 | 指定的解码器不存在 |
| | -2 | 开发包尚未初始化/初始化失败 |
| | -3 | 传入非法参数 |
| | -8 | 没有登录 |

**int __stdcall DEC_ClientRecvCommand(long _iID, int _iChannel, int _iPos, int _iCommand, void* _pBuffer, int _iBufferSize);**

| | | |
|---|---|---|
| 功能： | 通用命令获取接口 | |
| 参数： | _iID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| | _iCommand | 命令 ID |
| | | 16：获取音频状态 |
| | _pBuffer | 命令信息缓冲区，根据不同的命令 ID，传入不同的结构体变量 |
| | _iBufferSize | 缓冲区大小 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

## 3.5 对讲控制

**int __stdcall DEC_ClientTalkServer(long _ullD, int _bOn, int _iChannel = 0, int _iPos = 0);**

| | | |
|---|---|---|
| 功能： | 启动/停止跟 nvs 的对讲 | |
| 参数： | _ullD | 解码器标识号 |
| | _bOn | 启动/停止标识：1 表示启动，0 表示停止 |

| | | |
|---|---|---|
| | _iChannel | 当前选择通道 |
| | _iPos | 当前选择画面 |
| 返回： | 0 | 成功 |
| | -1 | 指定的解码器不存在 |
| | -2 | 开发包尚未初始化/初始化失败 |
| | -3 | 传入非法参数 |
| | -8 | 没有登录 |
| | -12 | 不允许操作，四画面下仅允许一路对讲，且在循环时不能对讲 |

**int __stdcall DEC_ClientGetTalkStatus (long _ulID, int _iChannel = 0, int _iPos = 0);**

| | | |
|---|---|---|
| 功能： | 获取对讲状态 | |
| 参数： | _ulID | 解码器标识号 |
| | _iChannel | 当前通道号 |
| | _iPos | 当前选择画面 |
| 返回： | 0 | 对讲关闭 |
| | 1 | 对讲打开 |
| | -1 | 指定的解码器不存在 |
| | -2 | 开发包尚未初始化/初始化失败 |
| | -3 | 传入非法参数 |
| | -8 | 没有登录 |

**int __stdcall DEC_ClientRecvCommand(long _iID, int _iChannel, int _iPos, int _iCommand, void* _pBuffer, int _iBufferSize);**

| | | |
|---|---|---|
| 功能： | 通用命令获取接口 | |
| 参数： | _iID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| | _iCommand | 命令 ID |
| | | 17：获取对讲状态 |
| | _pBuffer | 命令信息缓冲区，根据不同的命令 ID，传入不同的结构体变量 |
| | _iBufferSize | 缓冲区大小 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

## 3.6 解码器参数设置

**int __stdcall DEC_ClientGetDecoderPara(long _ulID, TDECPARAM *_pPara);**

| | | |
|---|---|---|
| 功能： | 获取解码器参数 | |
| 参数： | _ulID | 解码器标识号 |
| | _pPara | 指向 TDECPARAM 的结构体指针，用来获取解码器参数 |
| 返回： | 0 | 成功 |
| | -1 | 指定的解码器不存在 |
| | -2 | 开发包尚未初始化/初始化失败 |

-8　　没有登录

**int __stdcall DEC_ClientSetIP(long _ulID, char* _cNewIP, char* _cMask, char* _cGateway, char* _cDNS);**

功能：　设置解码器的 IP 地址、子网掩码、网关和 DNS

参数：　_ulID　　解码器标识号
　　_cNewIP　　新设置的解码器 IP
　　_cMask　　新设置的解码器子网掩码
　　_cGateway　　新设置的解码器网关
　　_cDNS　　新设置的解码器 DNS

返回：　0　　成功
　　-1　　指定的解码器不存在
　　-2　　开发包尚未初始化/初始化失败
　　-3　　传入非法参数(如果所有参数较上次都没有改变，也返回此值)
　　-8　　没有登录

**int __stdcall DEC_ClientSetDDNS(long _ulID, char* _cSvrIP, char* _cUserName, char* _cPassword, int _iPort)**

功能：　设置解码器的 DDNS IP、端口号、用户名和密码

参数：　_ulID　　解码器标识号
　　_cSvrIP　　设置的 DDNS IP 地址
　　_cUserName　　设置的 DDNS 用户名
　　_cPassword　　设置的 DDNS 密码
　　_iPort　　设置的 DDNS 端口号

返回：　0　　成功
　　-1　　指定的解码器不存在
　　-2　　开发包尚未初始化/初始化失败
　　-3　　传入非法参数
　　-8　　没有登录

说明：　如果没有设置 DDNS 的用户名和密码，则自动设置用户名为 0 ，密码为 0

**int __stdcall DEC_ClientSetCom(long _ulID, int _iType, int _iAddress, int _iBaudrate, int _iDatabit, int _iStopbit, int _iCheckbit, int _iFlowCtrl);**

功能：　设置解码器的 485 端口参数

参数：　_ulID　　解码器标识号
　　_iType　　485 控制协议类型：1 代表 PELCO_P 控制协议 0 代表透明通道
　　协议
　　_iAddress　　解码器 485 设置地址
　　_iBaudrate　　解码器 485 设置波特率
　　_iDatabit　　解码器 485 设置数据位
　　_iStopbit　　解码器 485 设置停止位
　　_iCheckbit　　解码器 485 设置校验位
　　_iFlowCtrl　　485 流控，默认为无流控，设置无效

返回： 0 成功
-1 指定的解码器不存在
-2 开发包尚未初始化/初始化失败
-3 传入非法参数
-8 没有登录

**int __stdcall DEC_ClientSetTVInfo(long _ulID, int _iIsPAL, int _iLanguage);**

功能： 设置视频参数

参数： _ulID 解码器标识号
_iIsPAL 视频制式：0 表示 PAL 制 1 表示 NTSC 制
_iLanguage 语言选择：0 表示中文 1 表示英文

返回： 0 成功
-1 指定的解码器不存在
-2 开发包尚未初始化/初始化失败
-3 传入非法参数
-8 没有登录

**int __stdcall DEC_ClientSetChannelMode(long _ulID, int _iChannel, int _iMode);**

功能： 设置通道显示画面类型

参数： _ulID 解码器标识号
_iChannel 当前通道号
_iMode 画面显示方式：1 表示单画面 4 表示四画面……

返回： 0 成功
-1 指定的解码器不存在
-2 开发包尚未初始化/初始化失败
-3 传入非法参数
-8 没有登录

**int __stdcall DEC_ClientSetVGASize(int _iID, int _iChannel, int _iSize);**

功能： 设置通道 VGA 分辨率

参数： _ulID 解码器标识号
_iChannel 当前通道号
_iSize VGA_800X600 = 1
VGA_1024X768 = 2,
VGA_1280X1024 = 3,
VGA_1280X720P_60 = 4,
VGA_1920X1080P_60 = 5,
VGA_1280X720P_50 = 6,
VGA_1920X1080P_50 = 7,
VGA_1920X1080I_60 = 8,
VGA_1920X1080I_50 = 9,
VGA_1366x768_60 = 10,
VGA_1440x900_60 = 11,
VGA_1280x800_60 = 12,

VGA_2560x1600_30 = 13,
VGA_3840x2160_30 = 14,
VGA_3840x2160_60 = 15,

返回： 0 成功
<0 失败，通过错误值查阅本文档错误码定义部分

**int __stdcall DEC_SetVolume(int _iID, int _iVolume);**

功能： 设置解码器的音量值

参数： _iID 解码器标识号
_iVolume 音量值，0-100

返回： 0 成功
<0 失败，通过错误值查阅本文档错误码定义部分

**int __stdcall DEC_GetVolume(int _iID, int *_piVolume);**

功能： 获取解码器的音量值

参数： _iID 解码器标识号
_piVolume 输出解码器当前音量值，0-100

返回： 0 成功
<0 失败，通过错误值查阅本文档错误码定义部分

**int __stdcall DEC_ClientSetCommonEnable(int _iID, int _iEnableID, int _iChannelNo, int _iEnableValue);**

功能： 设置通用使能接口，根据不同的使能 ID 设置不同使能

参数： _iID 解码器标识号
_iEnableID 使能 ID
0x13001 //断网保留最后一帧
0x13002 //预览偏好设置
_iChannel 当前通道号
_iEnableValue 使能值

返回： 0 成功
<0 失败，通过错误值查阅本文档错误码定义部分

**int __stdcall DEC_ClientGetCommonEnable(int _iID, int _iEnableID , int *_piEnableLastFrame, int *_piEnablePreference);**

功能： 获取断网保留最后一帧使能状态和预览偏好使能状态

参数： _iID 解码器标识号
_iEnableID 使能 ID
_piEnableLastFrame 输出断网保留最后一帧使能状态
_piEnablePreference 输出预览偏好设置使能状态

返回： 0 成功
<0 失败，通过错误值查阅本文档错误码定义部分

**int __stdcall DEC_ClientGetCommonEnableEx(int _iID, int _iEnableID , int _iChannelNo, int* _piEnableValue);**

功能： 获取通用使能状态

| 参数： | | |
|---|---|---|
| | _iID | 解码器标识号 |
| | _piVolume | 输出解码器当前音量值，0-100 |
| | _iEnableID | 使能 ID |
| | _iChannel | 当前通道号 |
| | _piEnableValue | 输出通用使能状态值 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

**int __stdcall DEC_SetSelectPicture(long _lID, int _iChannel, int _iPicture);**

功能： 选择画面

| 参数： | | |
|---|---|---|
| | _iID | 解码器标识号 |
| | _iChannel | 当前通道号 |
| | _iPicture | 选择画面号 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

**int __stdcall DEC_SetDZInfo(long _lID, DZ_INFO_PARAM* _pDzInfoParam);**

功能： 设置定制信息接口，特殊用途

| 参数： | | |
|---|---|---|
| | _iID | 解码器标识号 |
| | _pDzInfoParam | 定制信息结构体 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

**int __stdcall DEC_SetRegServer(long _lID, char* _pcRegSvrIP, int _iRegSvrPort, int _iEnable);**

功能： 平台服务器注册接口

| 参数： | | |
|---|---|---|
| | _iID | 解码器标识号 |
| | _pcRegSvrIP | 注册服务器 IP |
| | _iRegSvrPort | 注册服务器端口 |
| | _iEnable | 使能，不使能：0(默认值)　使能：1 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

**int __stdcall DEC_SetRegDevice(long _lID, char* _pcPUID, char* _pcPUName);**

功能： 设备注册接口

| 参数： | | |
|---|---|---|
| | _iID | 解码器标识号 |
| | _pcPUID | 设备 ID |
| | _pcPUName | 设备名称 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

**int __stdcall DEC_SetRegChannel(long _lID, int _iChannelNum, char* _pcPUID);**

功能： 通道注册接口

参数： _iID 解码器标识号
_iChannelNum 注册通道号
_pcPUID 通道注册 ID
返回： 0 成功
<0 失败，通过错误值查阅本文档错误码定义部分

**int __stdcall DEC_SetChannelSipConfig(long _lID**
**, int _iCmd, TSetSipVideoChannel* _lpBuf);**
功能： 设置 SIP 通道注册信息接口
参数： _iID 解码器标识号
_iCmd 传入 1 即可
_lpBuf TSetSipVideoChannel 结构体指针，包含 SIP 通道信息
返回： 0 成功
<0 失败，通过错误值查阅本文档错误码定义部分

**int __stdcall DEC_SetPlatformStart(long _lID, TPLATFORMINFO *_platformInfo);**
功能： 启动外挂
参数： _iID 解码器标识号
_platformInfo 外挂信息结构体 TPLATFORMINFO 的指针
返回： 0 成功
<0 失败，通过错误值查阅本文档错误码定义部分

**int __stdcall DEC_GetPlatformInfo(long _lID, TPLATFORMINFO *_platformInfo);**
功能： 获取设备当前所有外挂信息
参数： _iID 解码器标识号
_platformInfo 外挂信息结构体 TPLATFORMINFO 的指针
返回： 0 成功
<0 失败，通过错误值查阅本文档错误码定义部分

**int __stdcall DEC_GetRunningPlatformInfo(long _lID**
**, TPLATFORMINFO *_platformInfo);**
功能： 获取设备当前启用的外挂信息
参数： _iID 解码器标识号
_platformInfo 外挂信息结构体 TPLATFORMINFO 的指针
返回： 0 成功
<0 失败，通过错误值查阅本文档错误码定义部分

**int __stdcall DEC_ClientSendCommand(long _lID, int _iChannel, int _iPos, int _iCommand, void* _pBuffer, int _iBufferSize);**
功能： 通用命令发送接口
参数： _iID 解码器标识号
_iChannel 当前选择通道号
_iPos 当前选择画面
_iCommand 命令 ID

1：设备控制协议
2：校正编码卡偏屏偏色
3：上报写日志
4：获取用户数目
5：获取用户信息

| | | |
|---|---|---|
| | _pBuffer | 命令信息缓冲区，根据不同的命令 ID，传入不同的结构体变量 |
| | _iBufferSize | 缓冲区大小 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

**int __stdcall DEC_ClientRecvCommand(long _lID, int _iChannel, int _iPos, int _iCommand, void* _pBuffer, int _iBufferSize);**

功能： 通用命令获取接口

| | | |
|---|---|---|
| 参数： | _lID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| | _iCommand | 命令 ID |
| | | 6：获取用户数量 |
| | | 7：获取用户名和密码 |
| | | 8：获取设备错误信息 |
| | _pBuffer | 命令信息缓冲区，根据不同的命令 ID，传入不同的结构体变量 |
| | _iBufferSize | 缓冲区大小 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

**int __stdcall DEC_ClientSetDevConfig(long _ulID, int _iChannel, int _iPos, int _iCmd, void* _pvCmdBuf, int _iBuffSize);**

功能： 设备通用参数设置接口

| | | |
|---|---|---|
| 参数： | _lID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| | _iCmd | 命令 ID，详细内同查阅本文档设备参数命令码 |
| | _pvCmdBuf | 命令信息缓冲区，根据不同的命令 ID，传入不同的结构体变量 |
| | _iBuffSize | 缓冲区大小 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

**int __stdcall DEC_ClientGetDevConfig(long _ulID, int _iChannel, int _iPos, int _iCmd, void* _pvCmdBuf, int _iBuffSize);**

功能： 设备通用参数获取接口

| | | |
|---|---|---|
| 参数： | _lID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| | _iCmd | 命令 ID，详细内同查阅本文档设备参数命令码 |

| | | |
|---|---|---|
| | pvCmdBuf | 命令信息缓冲区，根据不同的命令 ID，传入不同的结构体变量 |
| | _iBuffSize | 缓冲区大小 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

## 3.7 Logo 叠加

**int __stdcall DEC_ClientSetLogo(long _ullD, TLOGOPARAM* _logoParam);**

| | | |
|---|---|---|
| 功能： | 设置叠加 Logo，该接口可以在视频源上叠加一个 Logo | |
| 参数： | _ullD | 解码器标识号 |
| | _logoParam | 指向 TLOGOPARAM 的指针，用于设置 Logo 显示和属性 |
| 返回： | 0 | 成功 |
| | -1 | 指定的解码器不存在 |
| | -2 | 开发包尚未初始化/初始化失败 |
| | -3 | 传入非法参数 |
| | -8 | 没有登录 |

**int __stdcall DEC_ClientGetLogo(long _ullD, TLOGOPARAM* _logoParam);**

| | | |
|---|---|---|
| 功能： | 获取 Logo 的参数 | |
| 参数： | _ullD | 解码器标识号 |
| | _logoParam | 指向 TLOGOPARAM 的指针，用于存放 Logo 属性 |
| 返回： | 0 | 成功 |
| | -1 | 指定的解码器不存在 |
| | -2 | 开发包尚未初始化/初始化失败 |
| | -8 | 没有登录 |

**int __stdcall DEC_ClientSetDevConfig(long _ullD, int _iChannel, int _iPos, int _iCmd, void* _pvCmdBuf, int _iBuffSize);**

| | | |
|---|---|---|
| 功能： | 设备通用参数设置接口 | |
| 参数： | _iID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| | _iCmd | CMD_DEC_CFG_LOGOPARAM |
| | _pvCmdBuf | 结构体 ChannelLogoParam 变量 |
| | _iBuffSize | sizeof(ChannelLogoParam) |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

**int __stdcall DEC_ClientGetDevConfig(long _ullD, int _iChannel, int _iPos, int _iCmd, void* _pvCmdBuf, int _iBuffSize);**

| | | |
|---|---|---|
| 功能： | 设备通用参数获取接口 | |
| 参数： | _iID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |

| | | |
|---|---|---|
| | _iPos | 当前选择画面 |
| | _iCmd | CMD_DEC_CFG_LOGOPARAM |
| | _pvCmdBuf | 结构体 ChannelLogoParam 变量 |
| | _iBuffSize | sizeof(ChannelLogoParam) |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

## 3.8 视频循环切换

**int __stdcall DEC_ClientStartLoop(long _ullD, int _iChannel = 0, int _iPos = 0);**

功能： 开始视频循环切换

| | | |
|---|---|---|
| 参数： | _ullD | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| 返回： | 0 | 成功 |
| | -1 | 指定的解码器不存在 |
| | -2 | 开发包尚未初始化/初始化失败 |
| | -3 | 传入非法参数 |
| | -8 | 没有登录 |
| | -12 | 不允许操作：例如当前没有可连接视频 |

**int __stdcall DEC_ClientStopLoop(long _ullD, int _iChannel = 0, int _iPos = 0);**

功能： 停止视频循环切换

| | | |
|---|---|---|
| 参数： | _ullD | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| 返回： | 0 | 成功 |
| | -1 | 指定的解码器不存在 |
| | -2 | 开发包尚未初始化/初始化失败 |
| | -3 | 传入非法参数 |
| | -8 | 没有登录 |

**int __stdcall DEC_ClientGetLoopStatus(long _ullD, int _iChannel = 0, int _iPos)**

功能： 获取当前视频循环切换状态

| | | |
|---|---|---|
| 参数： | _ullD | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| 返回： | 0 | 当前视频没有进行循环切换 |
| | 1 | 当前视频正在进行循环切换 |

**int __stdcall DEC_ClientGetLoopItemCount(long _ullD, int _iChannel = 0, int _iPos = 0);**

功能： 获取当前可连接 NVS 个数

参数： _ullD 解码器标识号
_iChannel 当前选择通道号
_iPos 当前选择画面
返回： 获取到的当前可连接的视频个数

**int __stdcall DEC_ClientGetLoopItem(long _ullD, int _iIndex, TNVSITEM *_pPara, int _iChannel = 0, int _iPos = 0);**
功能： 获取当前循环列表中 NVS 属性
参数： _ullD 解码器标识号
_iIndex NVS 标识号
_pPara 指向 TNVSITEM 的结构体指针，用来保存获得的 NVS 属性
_iChannel 当前选择通道号
_iPos 当前选择画面
返回： 0 成功
-1 指定的解码器不存在
-2 开发包尚未初始化/初始化失败
-3 传入非法参数
-8 没有登录

**int __stdcall DEC_ClientGetLoopItemEx(int _iID, int _iIndex, TNVSITEMEX* _pPara, int _iChannel/* = 0*/, int _iPos/* = 0*/);**
功能： 获取当前循环列表中 NVS 属性
参数： _ullD 解码器标识号
_iIndex NVS 标识号
_pPara 指向 TNVSITEMEX 的结构体指针，用来保存获得的 NVS 属性
_iChannel 当前选择通道号
_iPos 当前选择画面
返回： 0 成功
-1 指定的解码器不存在
-2 开发包尚未初始化/初始化失败
-3 传入非法参数
-8 没有登录

**int __stdcall DEC_ClientGetLoopItem_V1(int _iID, int _iChannel, int _iPos, int _iIndex, int* _piServerType, ServerItem* _puItem);**
功能： 获取当前循环列表中 NVS 属性
参数： _ullD 解码器标识号
_iChannel 当前选择通道号
_iPos 当前选择画面
_iIndex NVS 标识号
_piServerType 视频源服务器类型，0：私有，1：ONVIF，3：RTSP
_puItem 指向 ServerItem 的联合体指针，用来保存获得的 NVS 属性
返回： 0 成功
-1 指定的解码器不存在

| | | |
|---|---|---|
| | -2 | 开发包尚未初始化/初始化失败 |
| | -3 | 传入非法参数 |
| | -8 | 没有登录 |

**int __stdcall DEC_ClientSetLoopItem(long _ulID, int _iIndex, TNVSITEM *_pPara, int _iChannel = 0, int _iPos = 0);**

功能： 设置当前循环切换列表中的 NVS

| | | |
|---|---|---|
| 参数： | _ulID | 解码器标识号 |
| | _iIndex | 用来标识 NVS 的标号 |
| | _pPara | 指向 TNVSITEM 的结构指针，用来设置循环列表中的 NVS 属性 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| 返回： | 0 | 成功 |
| | -1 | 指定的解码器不存在 |
| | -2 | 开发包尚未初始化/初始化失败 |
| | -3 | 传入非法参数 |
| | -8 | 没有登录 |
| | -10 | 达到最大连接数：每个画面最多 64 个 NVS |
| | -12 | 不允许操作 |

**int __stdcall DEC_ClientSetLoopItemEx(int _iID, int _iIndex, TNVSITEMEX *_pPara, int _iChannel/* = 0*/, int _iPos/* = 0*/);**

功能： 设置当前循环切换列表中的 NVS

| | | |
|---|---|---|
| 参数： | _ulID | 解码器标识号 |
| | _iIndex | 用来标识 NVS 的标号 |
| | _pPara | 指向 TNVSITEMEX 的结构指针，用来设置循环列表中的 NVS 属性 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| 返回： | 0 | 成功 |
| | -1 | 指定的解码器不存在 |
| | -2 | 开发包尚未初始化/初始化失败 |
| | -3 | 传入非法参数 |
| | -8 | 没有登录 |
| | -10 | 达到最大连接数：每个画面最多 64 个 NVS |
| | -12 | 不允许操作 |

**int __stdcall DEC_ClientSetLoopItem_V1(int _iID, int _iChannel, int _iPos, int _iIndex, int _iServerType, ServerItem* _puItem);**

功能： 设置当前循环切换列表中的 NVS

| | | |
|---|---|---|
| 参数： | _ulID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |

| | | |
|---|---|---|
| | _iIndex | 用来标识 NVS 的标号 |
| | _iServerType | 视频源服务器类型，0：私有，1：ONVIF，3：RTSP |
| | _puItem | 指向 ServerItem 的结构指针，用来设置循环列表中的 NVS 属性 |
| 返回： | 0 | 成功 |
| | -1 | 指定的解码器不存在 |
| | -2 | 开发包尚未初始化/初始化失败 |
| | -3 | 传入非法参数 |
| | -8 | 没有登录 |
| | -10 | 达到最大连接数：每个画面最多 64 个 NVS |
| | -12 | 不允许操作 |

**int __stdcall DEC_ClientDelLoopItem(long _ulID, int _iIndex, int _iChannel = 0, int _iPos = 0);**

功能： 删除当前选择循环列表中的一项

| | | |
|---|---|---|
| 参数： | _ulID | 解码器标识号 |
| | _iIndex | 当前选择 NVS 标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| 返回： | 0 | 成功 |
| | -1 | 指定的解码器不存在 |
| | -2 | 开发包尚未初始化/初始化失败 |
| | -3 | 传入非法参数 |
| | -8 | 没有登录 |
| | -12 | 不允许操作 |

## 3.9 485 控制

**int __stdcall DEC_Client485Send(long _ulID,CONST BYTE *_pData, int _iSize, int _iChannel = 0, int _iPos = 0);**

功能： 发送 485 透明数据，用来控制设备

| | | |
|---|---|---|
| 参数： | _ulID | 解码器标识号 |
| | _pData | 字节型指针，保存发送控制码 |
| | _iSize | 发送控制码的个数 |
| | _iChannel | 当前选择通道 |
| | _iPos | 当前选择画面 |
| 返回： | 0 | 成功 |
| | -1 | 指定的解码器不存在 |
| | -2 | 开发包尚未初始化/初始化失败 |
| | -3 | 传入非法参数 |
| | -8 | 没有登录 |
| | -12 | 不允许操作 |

**int __stdcall DEC_ClientGetDevConfig(long _ulID, int _iChannel, int _iPos, int**

**_iCmd, void* _pvCmdBuf, int _iBuffSize);**

功能： 设备通用参数获取接口

| | | |
|---|---|---|
| 参数： | _iID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| | _iCmd | CMD_DEC_CFG_COM_PARAM |
| | _pvCmdBuf | 结构体 ComParam 变量 |
| | _iBuffSize | sizeof(ComParam) |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

## 3.10 获取云台控制协议

**int __stdcall DEC_ClientGetPTZprotocols(long _ulID, TDECPROTOCOL * _pProtocol);**

功能： 获取解码器支持的云台控制协议列表

| | | |
|---|---|---|
| 参数： | _ulID | 解码器标识号 |
| | _pProtocol | 指向 TDECPROTOCOL 的结构体指针，用于获取云台控制协议列表 |
| 返回： | 0 | 成功 |
| | -1 | 指定的解码器不存在 |
| | -2 | 开发包尚未初始化/初始化失败 |
| | -8 | 没有登录 |

**int __stdcall DEC_ClientGetDevConfig(long _ulID, int _iChannel, int _iPos, int _iCmd, void* _pvCmdBuf, int _iBuffSize);**

功能： 设备通用参数获取接口

| | | |
|---|---|---|
| 参数： | _iID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| | _iCmd | CMD_DEC_CFG_PROTOCOL |
| | _pvCmdBuf | 结构体 TDECPROTOCOL_EX 变量 |
| | _iBuffSize | sizeof(TDECPROTOCOL_EX) |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

## 3.11 NVD 系统设置

**int __stdcall DEC_ClientRebootDecoder(long _ulID);**

功能： 重启解码器

| | | |
|---|---|---|
| 参数： | _ulID | 解码器标识号 |

返回： 0 成功
-1 指定的解码器不存在
-2 开发包尚未初始化/初始化失败
-8 没有登录
说明： 该接口用来重新启动解码器，当进行某些设置后必须重新启动解码器才能生效时，可以调用该接口

**int __stdcall DEC_ClientResetDefault(long _ulID);**
功能： 恢复解码器出厂默认值
参数： _ulID 解码器标识号
返回： 0 成功
-1 指定的解码器不存在
-2 开发包尚未初始化/初始化失败
-8 没有登录
说明: 该接口要谨慎使用,恢复为出厂默认值以后,原来所有设置的值都将丢失,除了IP 地址和 MAC 地址保留不变外,其他所有属性均恢复为出厂默认值

## 3.12 远程升级

**int __stdcall DEC_ClientUpgrade(long _ulID, char *_cFileName);**
功能： 升级内核应用程序
参数： _ulID 解码器标识号
_cFileName 升级文件名：系统内核程序.bin 格式文件
返回： 0 成功
-1 指定的解码器不存在
-2 开发包尚未初始化/初始化失败
-7 文件格式非法
-8 没有登录
-11 正在升级
警告: 此功能一定要设备维护人员才能调用,否则会导致严重后果

**int __stdcall DEC_ClientUpgradeProtocol(long _ulID, char *_cFilePath, char *_cFileName);**
功能： 向解码器添加协议
参数： _ulID 解码器标识号
_cFilePath 升级文件路径
_cFileName 升级文件名称：.so 格式文件
返回： 0 成功
-1 指定的解码器不存在
-2 开发包尚未初始化/初始化失败
-7 文件格式非法
-8 没有登录
-11 正在升级

**int __stdcall DEC_ClientDeleteProtocol(long _ulID, char *_cProtocol);**

功能： 删除协议

| | | |
|---|---|---|
| 参数： | _ulID | 解码器标识号 |
| | _cProtocol | 要删除协议名称 |
| 返回： | 0 | 成功 |
| | -1 | 指定的解码器不存在 |
| | -2 | 开发包尚未初始化/初始化失败 |
| | -3 | 传入非法参数 |
| | -8 | 没有登录 |

**int __stdcall DEC_ClientUpgradeLogo(long _ulID, char *_cFileName);**

功能： 升级显示的 Logo

| | | |
|---|---|---|
| 参数： | _ulID | 解码器标识号 |
| | _cFileName | 升级 Logo 文件名：.bmp 格式文件，文件大小<=200*100, |
| 返回： | 0 | 成功 |
| | -1 | 指定的解码器不存在 |
| | -2 | 开发包尚未初始化/初始化失败 |
| | -7 | 文件格式非法 |
| | -8 | 没有登录 |
| | -11 | 正在升级 |
| | -15 | 位置越界 |

**int __stdcall DEC_ClientUpgradeWeb(long _ulID, char *_cFileName);**

功能： 升级网页

| | | |
|---|---|---|
| 参数： | _ulID | 解码器标识号 |
| | _cFileName | 升级文件名：.box 格式文件 |
| 返回： | 0 | 成功 |
| | -1 | 指定的解码器不存在 |
| | -2 | 开发包尚未初始化/初始化失败 |
| | -7 | 文件格式非法 |
| | -8 | 没有登录 |
| | -11 | 正在升级 |

警告: 此功能用于升级解码器内核 Web 程序.box,升级了非法网页将导致 Web 功能失灵

## 3.13 获取解码器版本号

**int __stdcall DEC_ClientGetVersion(long _ulID,char* _cVer);**

功能： 获取解码器版本号

| | | |
|---|---|---|
| 参数： | _ulID | 解码器标识号 |
| | _cVer | 接收解码器版本号信息 |
| 返回： | 0 | 成功 |
| | -1 | 指定的解码器不存在 |

| | | |
|---|---|---|
| | -2 | 开发包尚未初始化/初始化失败 |
| | -8 | 没有登录 |

## 3.14 获取解码器解码能力

**int __stdcall DEC_ClientGetCapability(long _ullD, char *_cCap);**

功能： 获取解码器解码能力

| | | |
|---|---|---|
| 参数： | _ullD | 解码器标识号 |
| | _cCap | 接受解码器解码能力,最长 32 个字节 |
| 返回： | 0 | 成功 |
| | -1 | 指定的解码器不存在 |
| | -2 | 开发包尚未初始化/初始化失败 |
| | -8 | 没有登录 |

## 3.15 PUSH 流

**int __stdcall DEC_ClientStartPushStream(long _ullD, int _iChannel, int _iPos, int _iEncodeMode, int _iVideoSize);**

功能： 通知解码器开始推送码流

| | | |
|---|---|---|
| 参数： | _ullD | 解码器标识号 |
| | _iChannel | 当前选择通道 |
| | _iPos | 当前选择画面 |
| | _iEncodeMode | 编码方式，见本文档中编码方式的解释 |
| | _iVideoSize | 视频大小，见本文档中视频大小的解释 |
| 返回： | >=0 | 成功，返回值为 PUSH ID，用于推流。 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

说明： 返回成功后，需要收到 DEC_WCM_DEC_PUSHSTREAM 消息后，才能真正的推数据。

**int __stdcall DEC_ClientStartPushStreamEx(long _ullD, int _iChannel, int _iPos, int _iEncodeMode, int _iVideoSize, char* _pcEncryptKey);**

功能： 通知解码器开始推送码流，DEC_ClientStartPushStream 接口的扩展

| | | |
|---|---|---|
| 参数： | _ullD | 解码器标识号 |
| | _iChannel | 当前选择通道 |
| | _iPos | 当前选择画面 |
| | _iEncodeMode | 编码方式，见本文档中编码方式的解释 |
| | _iVideoSize | 视频大小，见本文档中视频大小的解释 |
| | _pcEncryptKey | 视频解密密码 |
| 返回： | >=0 | 成功，返回值为 PUSH ID，用于推流。 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

说明： 返回成功后，需要收到 DEC_WCM_DEC_PUSHSTREAM 消息后，才能真正的推数据。

**int __stdcall DEC_ClientStartPushStream_V1(long _ullD, int _iChannel, int _iPos, START_PUSH_PARAM *_pStartPushParam);**

功能： 通知解码器开始推送码流， DEC_ClientStartPushStream 和 DEC_ClientStartPushStreamEx 接口的扩展

参数： _ullD 解码器标识号

_iChannel 当前选择通道

_iPos 当前选择画面

_pStartPushParam 指向 START_PUSH_PARAM 的结构体指针。

返回： >=0 成功，返回值为 PUSH ID，用于推流。

<0 失败，通过错误值查阅本文档错误码定义部分

说明： 返回成功后，需要收到 DEC_WCM_DEC_PUSHSTREAM 消息后，才能真正的推数据。

**int __stdcall DEC_ClientSendStream(long _ulPushID, int _iLen, char* _cStreambuf)；**

功能： 向解码器推送码流

参数： _ulPushID PUSH ID，由 DEC_ClientStartPushStream 调用成功后返回

_iLen 码流数据长度

_cStreambuf 码流缓冲区指针

返回： 0 成功

<0 失败，通过错误值查阅本文档错误码定义部分

**int __stdcall DEC_ClientStopPushStream(long _ulPushID)；**

功能： 通知解码器停止推送码流

参数： _ulPushID PUSH ID，由 DEC_ClientStartPushStream 调用成功后返回

返回： 0 成功

<0 失败，通过错误值查阅本文档错误码定义部分

**int __stdcall DEC_ClientStopPushStreamEx(long _ulPushID , int _iShowLastPic/*=0*/);**

功能： 通知解码器停止推送码流，DEC_ClientStopPushStream 接口的扩展

参数： _ulPushID PUSH ID，由 DEC_ClientStartPushStream 调用成功后返回

_iShowLastPic 标识是否保留最后一帧，默认不保留

返回： 0 成功

<0 失败，通过错误值查阅本文档错误码定义部分

**int __stdcall DEC_ClientGetStreamBufferState(long _ulPushID , int * _piStreamBufferState);**

功能： 向解码器推送流时，获取 SDK 缓冲区状态

参数： _ulPushID PUSH ID，由 DEC_ClientStartPushStream 调用成功后返回

_piStreamBufferState 输出 SDK 缓冲区状态

返回： 0 成功

<0 失败，通过错误值查阅本文档错误码定义部分

**int __stdcall DEC_ClientStreamControl(long _ulPushID, int _iCmd**

**, void* _pvCmdBuf, int _iCmdLen);**

功能： 流控操作

参数： _ulPushID PUSH ID，由 DEC_ClientStartPushStream 调用成功后返回

_iCmd 标识发送码流的操作指令，0：暂停，1：快放，2：慢放

*_pvCmdBuf 存储操作的指令值，目前仅支持以下三种操作：

<1>暂停指令：1：暂停， 0：恢复

<2>快放指令：0：正常 1：两倍速 2：四倍速 3：八倍速 4：十六倍速

<3>慢放指令：0：正常 1：两倍速 2：四倍速 3：八倍速 4：十六倍速

_iCmdLen 指令值存储区的长度

返回： 0 成功

<0 失败，通过错误值查阅本文档错误码定义部分

**int __stdcall DEC_ClientConfigStreamInfo(int _iID, int _iChannel, int _iPos, int _iCmd, void* _pvCmdBuf, int _iBuffSize);**

功能： 配置码流信息接口，此接口不建议使用

参数： _ullD 解码器标识号

_iChannel 当前选择通道

_iPos 当前选择画面

_iCmd 配置码流命令， 1：重新配置视频头信息，2：清除缓冲区数据

*_pvCmdBuf 缓冲区

_iBuffSize 缓冲区长度

返回： 0 成功

<0 失败，通过错误值查阅本文档错误码定义部分

## 3.16 报警管理

**int __stdcall DEC_ClientSetAlarmOut(long _ullD, int _iAlarmOut, int _iAlarmMode);**

功能： 设置解码器报警方式

参数： _ullD 解码器标识号

_iAlarmOut 解码器报警方式：0 不设置报警方式 1 端口报警 2 视频丢失报警 3 端口报警+视频丢失报警 4 视频移动报警 5 端口报警+视频移动报警 6 视频丢失+视频移动报警 7 端口+视频丢失+视频移动报警

_iAlarmMode 报警端口输出状态：0 关闭 1 打开

返回： 0 成功

-1 指定的解码器不存在

-2 开发包尚未初始化/初始化失败

-3 传入非法参数

-8 没有登录

**int __stdcall DEC_ClientSetDevConfig(long _ullD, int _iChannel, int _iPos, int _iCmd, void* _pvCmdBuf, int _iBuffSize);**

功能： 设备通用参数设置接口

参数： _iID 解码器标识号

| | | |
|---|---|---|
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| | _iCmd | 命令 ID，详细内同查阅本文档设备参数命令码 |
| | _pvCmdBuf | 命令信息缓冲区，根据不同的命令 ID，传入不同的结构体变量 |
| | _iBuffSize | 缓冲区大小 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

**int __stdcall DEC_ClientGetDevConfig(long _ulID, int _iChannel, int _iPos, int _iCmd, void* _pvCmdBuf, int _iBuffSize);**

功能： 设备通用参数获取接口

| | | |
|---|---|---|
| 参数： | _iID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| | _iCmd | 命令 ID，详细内同查阅本文档设备参数命令码 |
| | _pvCmdBuf | 命令信息缓冲区，根据不同的命令 ID，传入不同的结构体变量 |
| | _iBuffSize | 缓冲区大小 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

## 3.17 拼控管理

**int __stdcall DEC_ClientSetDevConfig(long _ulID, int _iChannel, int _iPos, int _iCmd, void* _pvCmdBuf, int _iBuffSize);**

功能： 设备通用参数设置接口

| | | |
|---|---|---|
| 参数： | _iID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| | _iCmd | 命令 ID，详细内同查阅本文档设备参数命令码 |
| | _pvCmdBuf | 命令信息缓冲区，根据不同的命令 ID，传入不同的结构体变量 |
| | _iBuffSize | 缓冲区大小 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

**int __stdcall DEC_ClientGetDevConfig(long _ulID, int _iChannel, int _iPos, int _iCmd, void* _pvCmdBuf, int _iBuffSize);**

功能： 设备通用参数获取接口

| | | |
|---|---|---|
| 参数： | _iID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| | _iCmd | 命令 ID，详细内同查阅本文档设备参数命令码 |
| | _pvCmdBuf | 命令信息缓冲区，根据不同的命令 ID，传入不同的结构体变量 |
| | _iBuffSize | 缓冲区大小 |

返回： 0 成功
<0 失败，通过错误值查阅本文档错误码定义部分

## 3.18 集群管理

**int __stdcall DEC_ClientSetClusterConfig(long _iID, int _iCommand , void *_lpInBuffer, int _iInBufferSize);**

功能： 集群参数设置接口

参数： _iID 解码器标识号
_iCommand 集群命令 ID
0：合并集群
1：解散集群
2：设置集群别名
8：设置集群内主控卡 IP 地址
10：集群搜索
_lpInBuffer 命令信息缓冲区，根据不同的命令 ID，传入不同的结构体变量
_iInBufferSize 缓冲区大小

返回： 0 成功
<0 失败，通过错误值查阅本文档错误码定义部分；

**int __stdcall DEC_ClientGetClusterConfig(long _iID, int _iCommand , void *_lpInBuffer, int _iInBufferSize);**

功能： 集群参数获取接口

参数： _iID 解码器标识号
_iCommand 集群命令 ID
2：获取集群别名
3：获取集群信息
4：获取集群内主控卡版本信息
5：获取集群内工作卡（解码卡和编码卡）版本信息
6：获取集群内主控卡状态信息
7：获取集群内工作卡（解码卡和编码卡）状态信息
8：获取集群内主控卡 IP 地址
9：获取集群内设备信息
10：获取集群搜索结果
11：获取集群内设备状态信息
12：获取集群内主控卡的 MAC 地址
_lpInBuffer 命令信息缓冲区，根据不同的命令 ID，传入不同的结构体变量
_iInBufferSize 缓冲区大小

返回： 0 成功
<0 失败，通过错误值查阅本文档错误码定义部分；

## 3.19 预案管理

**int __stdcall DEC_ClientSetDevConfig(long _ullD, int _iChannel, int _iPos, int _iCmd, void* _pvCmdBuf, int _iBuffSize);**

功能： 设备通用参数设置接口

| | | |
|---|---|---|
| 参数： | _iID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| | _iCmd | 命令 ID，详细内同查阅本文档设备参数命令码 |
| | _pvCmdBuf | 命令信息缓冲区，根据不同的命令 ID，传入不同的结构体变量 |
| | _iBuffSize | 缓冲区大小 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

**int __stdcall DEC_ClientGetDevConfig(long _ullD, int _iChannel, int _iPos, int _iCmd, void* _pvCmdBuf, int _iBuffSize);**

功能： 设备通用参数获取接口

| | | |
|---|---|---|
| 参数： | _iID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| | _iCmd | 命令 ID，详细内同查阅本文档设备参数命令码 |
| | _pvCmdBuf | 命令信息缓冲区，根据不同的命令 ID，传入不同的结构体变量 |
| | _iBuffSize | 缓冲区大小 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

## 3.20 日志管理

**int __stdcall DEC_ClientNetLogQuery(long _ullD, PDecLogQuery _logQuery, int _iSize);**

功能： 查询日志

| | | |
|---|---|---|
| 参数： | _ullD | 解码器标识号 |
| | _logQuery | 日志查询结构体指针 |
| | _iSize | 结构体大小 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

**int __stdcall DEC_ClientNetLogGetLogfile(long _ullD, int _iLogIndex, PDecLogData _pLogInfo, int _iSize);**

功能： 获取日志内容

| | | |
|---|---|---|
| 参数： | _ullD | 解码器标识号 |
| | _iLogIndex | 日志索引，范围 0-19，每次最多查 20 条 |

| | | |
|---|---|---|
| | _pLogInfo | 日志信息结构体指针 |
| | _iSize | 结构体大小 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

**int __stdcall DEC_ClientNetLogGetLogCount(long _ullD, int *_iTotalCount, int *_iCurrentCount);**

功能： 获取日志条数

| | | |
|---|---|---|
| 参数： | _ullD | 解码器标识号 |
| | _iTotalCount | 输出设备当前日志总条数 |
| | _iCurrentCount | 输出本次查询日志条数 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

## 3.21 屏控管理

**int __stdcall DEC_ClientSetDevConfig(long _ullD, int _iChannel, int _iPos, int _iCmd, void* _pvCmdBuf, int _iBuffSize);**

功能： 设备通用参数设置接口

| | | |
|---|---|---|
| 参数： | _iID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| | _iCmd | 命令 ID，详细内同查阅本文档设备参数命令码 |
| | _pvCmdBuf | 命令信息缓冲区，根据不同的命令 ID，传入不同的结构体变量 |
| | _iBuffSize | 缓冲区大小 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

**int __stdcall DEC_ClientGetDevConfig(long _ullD, int _iChannel, int _iPos, int _iCmd, void* _pvCmdBuf, int _iBuffSize);**

功能： 设备通用参数获取接口

| | | |
|---|---|---|
| 参数： | _iID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| | _iCmd | 命令 ID，详细内同查阅本文档设备参数命令码 |
| | _pvCmdBuf | 命令信息缓冲区，根据不同的命令 ID，传入不同的结构体变量 |
| | _iBuffSize | 缓冲区大小 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

## 3.22 预案切换管理

**int __stdcall DEC_ClientSetDevConfig(long _ulID, int _iChannel, int _iPos, int _iCmd, void* _pvCmdBuf, int _iBuffSize);**

功能： 设备通用参数设置接口

| | | |
|---|---|---|
| 参数： | _iID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| | _iCmd | 命令 ID，详细内同查阅本文档设备参数命令码 |
| | _pvCmdBuf | 命令信息缓冲区，根据不同的命令 ID，传入不同的结构体变量 |
| | _iBuffSize | 缓冲区大小 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

**int __stdcall DEC_ClientGetDevConfig(long _ulID, int _iChannel, int _iPos, int _iCmd, void* _pvCmdBuf, int _iBuffSize);**

功能： 设备通用参数获取接口

| | | |
|---|---|---|
| 参数： | _iID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| | _iCmd | 命令 ID，详细内同查阅本文档设备参数命令码 |
| | _pvCmdBuf | 命令信息缓冲区，根据不同的命令 ID，传入不同的结构体变量 |
| | _iBuffSize | 缓冲区大小 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

## 3.23 U 盘管理

**int __stdcall DEC_ClientSetDevConfig(long _ulID, int _iChannel, int _iPos, int _iCmd, void* _pvCmdBuf, int _iBuffSize);**

功能： 设备通用参数设置接口

| | | |
|---|---|---|
| 参数： | _iID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| | _iCmd | 命令 ID，详细内同查阅本文档设备参数命令码 |
| | _pvCmdBuf | 命令信息缓冲区，根据不同的命令 ID，传入不同的结构体变量 |
| | _iBuffSize | 缓冲区大小 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

**int __stdcall DEC_ClientGetDevConfig(long _ulID, int _iChannel, int _iPos, int _iCmd, void* _pvCmdBuf, int _iBuffSize);**

功能： 设备通用参数获取接口

| | | |
|---|---|---|
| 参数： | _iID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| | _iCmd | 命令 ID，详细内同查阅本文档设备参数命令码 |
| | _pvCmdBuf | 命令信息缓冲区，根据不同的命令 ID，传入不同的结构体变量 |
| | _iBuffSize | 缓冲区大小 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

**int __stdcall DEC_ClientSendCommand(long _iID, int _iChannel, int _iPos, int _iCommand, void* _pBuffer, int _iBufferSize);**

功能： 通用命令发送接口

| | | |
|---|---|---|
| 参数： | _iID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| | _iCommand | 命令 ID<br>13：获取 U 盘视频列表<br>14：发送 U 盘命令 |
| | _pBuffer | 命令信息缓冲区，根据不同的命令 ID，传入不同的结构体变量 |
| | _iBufferSize | 缓冲区大小 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

**int __stdcall DEC_ClientRecvCommand(long _iID, int _iChannel, int _iPos, int _iCommand, void* _pBuffer, int _iBufferSize);**

功能： 通用命令获取接口

| | | |
|---|---|---|
| 参数： | _iID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| | _iCommand | 命令 ID<br>13：获取 U 盘视频列表<br>14：获取命令结果 |
| | _pBuffer | 命令信息缓冲区，根据不同的命令 ID，传入不同的结构体变量 |
| | _iBufferSize | 缓冲区大小 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

## 3.24 抓拍

**int __stdcall DEC_ClientSendCommand(long _iID, int _iChannel, int _iPos, int _iCommand, void* _pBuffer, int _iBufferSize);**

功能： 通用命令发送接口

| | | |
|---|---|---|
| 参数： | _iID | 解码器标识号 |
| | _iChannel | 当前选择通道号 |
| | _iPos | 当前选择画面 |
| | _iCommand | 命令 ID |
| | | 9：发送抓拍命令 |
| | | 10：发送获取抓拍文件名命令 |
| | | 15：发送删除抓拍文件命令 |
| | _pBuffer | 命令信息缓冲区，根据不同的命令 ID，传入不同的结构体变量 |
| | _iBufferSize | 缓冲区大小 |
| 返回： | 0 | 成功 |
| | <0 | 失败，通过错误值查阅本文档错误码定义部分 |

# 第 4 章 消息说明

SDK 内回调函数可以和消息同时存在，也可以选择使用，使用回调函数的方法和消息类似，在这里只介绍消息的使用。

## 4.1 系统消息

**消息注册方法：**

启动开发包时传入消息 ID 来注册系统消息。Dec_ClientStartup(unsigned int_uiMessage, HWND _hWnd)，系统消息注册后会得到以下消息通知。消息类以 WParamLo 来区分。

**消息列表：**

**DEC_WCM_DEC_LOGON:** 登录状态消息，用户调用登录接口 Dec_ClientLogon 后，登录结果产生后，通过该消息通知应用程序。

消息参数： LParam，解码器的标识号

WParamHi，登录状态，具体状态可能为以下状态：

DEC_LOGON_SUCCESS，登录成功；

DEC_LOGON_FAILED，登录失败，可能被解码器拒绝；

DEC_LOGON_TIMEOUT，登录超时；NVD 不存在或网络连接不通。

**DEC_WCM_NVS_STARTVIEW:** 开始连接 NVS 视频消息，当连接到 NVS 视频时产生。

消息参数： LParam，解码器的标识号

WParamHi，用来获得通道号和画面号

**DEC_WCM_NVS_STOPVIEW:** 断开 NVS 消息，当断开 NVS 视频时产生。

消息参数： LParam，解码器的标识号

WParamHi，用来获得通道号和画面号

**DEC_WCM_DEC_TALK:** 对讲消息,当开启对讲或者关闭对讲时产生。
消息参数： LParam，解码器的标识号
WParamHi，用来获得通道号和画面号

**DEC_WCM_DEC_UPGRADE:** 升级解码器时产生的消息
消息参数： LParam，解码器的标识号
WParamHi，升级状态 0--成功

**DEC_WCM_DEC_UPGRADELOGO:** 升级 Logo 时产生的消息
消息参数： LParam，解码器的标识号
WParamHi，升级状态 0--成功

**DEC_WCM_DEC_UPGRADEP:** 升级控制协议产生的消息
消息参数： LParam，解码器的标识号
WParamHi，升级状态 0--成功

**DEC_WCM_ERR_ORDER:** 命令层发生错误产生的消息
消息参数： LParam，解码器的标识号

**DEC_WCM_DEC_PARASET:** 参数发生改变时产生的消息

**DEC_WCM_DEC_UPGRADEWEB:** 升级网页时产生的消息
消息参数： LParam，解码器的标识号
WParamHi，升级状态 0--成功

**DEC_WCM_DEC_UPGRADEPROCESS:** 解码器升级进度产生的消息
消息参数： LParam，解码器的标识号
WParamHi，升级百分比

**DEC_WCM_DEC_AUDIOSTATUS:** 音频状态消息
消息参数： LParam，解码器的标识号
WParamHi，用来获得通道号和画面号

**DEC_WCM_DEC_PUSHSTREAM:** 开始推送码流消息
消息参数： LParam，用来获得对应的 PUSH ID
WParamHi，解码器回复的结果

**DEC_WCM_ERR_PUSHSTREAM:** Push stream 连接意外断开

**DEC_WCM_ERR_DATANET:** 数据通道发生网络异常

**DEC_WCM_ERR_USERPASS:** 修改密码结果

**DEC_WCM_DETECT_CHANN_STATE:** 解码器各个通道的状态

**DEC_WCM_ALARM_NOTIFY:**上报警情消息

**DEC_WCM_PARASET_CREATEVVO:** 创建拼控输出通道消息

**DEC_WCM_AUTOTEST_INFO:** 自动化调试
**DEC_WCM_QUERY_LOG_FINISH:** 日志查询

**DEC_WCM_NVS_STARTVIEW_FAILED:** 连接 NVS 失败消息

**DEC_WCM_SEARCH_IPC:** 搜索发现 IPC 消息

**DEC_WCM_SEARCH_NVD:** 搜索发现解码器消息

**DEC_WCM_CLUSTER_STATE:** 集群状态消息

**DEC_WCM_CLUSTER_CONTROL_CARD_STATE:** 集群主控卡状态消息

**DEC_WCM_CLUSTER_WORK_CARD_STATE:** 集群工作卡状态消息

**DEC_WCM_EXPORT_CONFIG_FINISH:** 导出配置文件完成消息

**DEC_WCM_ENCODE_CHANNEL_STATE:** 编码通道状态消息

**DEC_WCM_LOG_WRITE:** 写日志消息

**DEC_WCM_EXCEPTION:** 异常消息

**DEC_WCM_DEC_USER_NUM:** 获取用户数量消息

**DEC_WCM_DEC_USER_INFO:** 获取用户信息消息

**DEC_WCM_DEC_LAST_ERROR:** 设备错误码消息

**DEC_WCM_DEC_PICSNAP**：抓拍结果消息。

**DEC_WCM_DEC_GETPICFILENAME**：抓拍结果消息。

**DEC_WCM_DEC_SCREENCTLSET**：屏控设置结果消息。

**DEC_WCM_DEC_REBOOTTRADE**：重启外挂结果消息。

**DEC_WCM_DEC_GETUDISK_VIDEOLISTS**：U 盘视频列表更新消息。

**DEC_WCM_DEC_VIDEO_PLAYSET**：U 盘播放命令结果消息。

## 4.2 参数改变消息

**设备参数更改消息：**

当 NVD 里的任何参数被其他客户端或自己修改，SDK 都会产生该消息来通知应用程序，以达到实时更新的目的。设备的属性在任何时候都可以通过相应接口获得。

**消息注册方法：**

启动开发包时调用注册消息回调接口 DEC_ClientSetNotify(DecNotifyFun* _pNotify)向 SDK 注册回调，参数改变消息通过回调函数返给上层用户。

**消息列表：**

```
#define  DEC_PARACHANGE_DNSABSTRACT              1
#define  DEC_PARACHANGE_TVINFO                   2
#define  DEC_PARACHANGE_RS485TYPE                3
#define  DEC_PARACHANGE_ALARMOUT                 4
#define  DEC_PARACHANGE_CHANNELINFO              5
#define  DEC_PARACHANGE_SETLOOPITEM              6
#define  DEC_PARACHANGE_DELLOOPITEM              7
#define  DEC_PARACHANGE_STARTLOOP                8
#define  DEC_PARACHANGE_STOPTLOOP                9
#define  DEC_PARACHANGE_GETPTZPROTOCOLS          10
#define  DEC_PARACHANGE_PLATFORMRUN              11
#define  DEC_PARACHANGE_VGA                      12
#define  DEC_PARACHANGE_MANAGERINFO              13
#define  DEC_PARACHANGE_COMMONINFO               14
#define  DEC_PARACHANGE_COMMONINFO_CHANNEL       15
#define  DEC_PARACHANGE_DZCOMMON                 16
#define  DEC_PARACHANGE_SIP_VIDEOCHANNEL         17
#define  DEC_PARACHANGE_VIDEOPARAM               18
#define  DEC_PARACHANGE_VOLUME                   19
#define  DEC_PARACHANGE_ALARM_LINKSWITCH         20
#define  DEC_PARACHANGE_CREATEFREEVO             21
#define  DEC_PARACHANGE_ALARM_SCHEDULE           22
#define  DEC_PARACHANGE_ALARMLINK                23
#define  DEC_PARACHANGE_ALM_IN_LHP               24
#define  DEC_PARACHANGE_ALM_OUT_LHP              25
#define  DEC_PARACHANGE_ALARM_INPORT             26
#define  DEC_PARACHANGE_ALARM_OUTPORT            27
#define  DEC_PARACHANGE_SCHEDULE_ENABLE          28
#define  DEC_PARACHANGE_TIMEZONE                 29
#define  DEC_PARACHANGE_DISPLAY                  30
#define  DEC_PARACHANGE_CHANGEAREA               31
```

```
#define DEC_PARACHANGE_CREATEVVO 32
#define DEC_PARACHANGE_SELECT_PIC 33
#define DEC_PARACHANGE_PREFERENCE 34
#define DEC_PARACHANGE_RESEVELASTFRAME 35
#define DEC_PARACHANGE_LOGOPARAM 36
#define DEC_PARACHANGE_DEVICEABSTRACT 37
#define DEC_PARACHANGE_CLUSTER_ALIAS 38
#define DEC_PARACHANGE_CLUSTER_CLUSTERINFO 39
#define DEC_PARACHANGE_CLUSTER_DEVINFO 40
#define DEC_PARACHANGE_CLUSTER_CONTRAL_CARD_INFO 41
#define DEC_PARACHANGE_CLUSTER_WORK_CARD_INFO 42
#define DEC_PARACHANGE_MERAGE_CLUSTER 43
#define DEC_PARACHANGE_DEMERAGE_CLUSTER 44
#define DEC_PARACHANGE_FLOAT_WINDOW 45
#define DEC_PARACHANGE_PLAN 46
#define DEC_PARACHANGE_PLAN_ALIAS 47
#define DEC_PARACHANGE_OSD 48
#define DEC_PARACHANGE_CLUSTER_IP 49
#define DEC_PARACHANGE_APPLY_PLAN 50
#define DEC_PARACHANGE_DEVCOMMONNAME 51
#define DEC_PARACHANGE_DETECT_CHANN_STATE 52
#define DEC_PARACHANGE_SHOWOFFLINECHN 53
#define DEC_PARACHANGE_DEVICEMACADDR 54
#define DEC_PARACHANGE_PLATFORM_LIST 55
#define DEC_PARACHANGE_CTRL_MAC 56
#define  DEC_PARACHANGE_SINGLE_PIC 57
#define DEC_PARACHANGE_AUTOTEST_SYSTEMTYPE 58
#define DEC_PARACHANGE_AUTOTEST_DEVICEPRODUCTER 59
#define DEC_PARACHANGE_SCREEN_REGION 60
#define DEC_PARACHANGE_SCREEN_BASICPARA 61
#define DEC_PARACHANGE_SCREEN_OUTPUT_MODE 62
#define DEC_PARACHANGE_PICLEVEL_RELATION 63
#define DEC_PARACHANGE_SYNCLOOPCTRL 64
#define DEC_PARACHANGE_SYNCLOOPTIME 65
#define DEC_PARACHANGE_ENCODERPAPA 66
#define DEC_PARACHANGE_HTTPPORT 67
#define DEC_PARACHANGE_SHOWICON 68
#define  DEC_PARACHANGE_ALARM_EFFECT_LIGHT 69
#define DEC_PARACHANGE_ALARM_EFFECT_REGION 70
#define DEC_PARACHANGE_ALARM_EFFECT_TEXT 71
#define DEC_PARACHANGE_HD_DISPLAY 72
#define DEC_PARACHANGE_PLAN_LOOP_ITEM 73
#define DEC_PARACHANGE_PLAN_LOOP 74
#define DEC_PARACHANGE_PLAN_LOOP_ITEM_DEL 75
```

```
#define DEC_PARACHANGE_LANPARAM_WORKMODE 76
#define DEC_PARACHANGE_SIP_VIDEO_CHANNEL_EX_V2 77
#define DEC_PARACHANGE_CHAN_MAX_WIN 78
#define DEC_PARACHANGE_CHAN_LAYOUTLIST 79
#define DEC_PARACHANGE_CHAN_RESOLUTIONLIST 80
#define DEC_PARACHANGE_GETSCREENCTLPROTOCOLS 81
#define DEC_PARACHANGE_SCREENCRLPROTOCOL 82
#define DEC_PARACHANGE_SET_CHANNEL_VIDEO 83
#define DEC_PARACHANGE_CURVIDEO 84
#define DEC_PARACHANGE_VIDEO_PLAYMODE 85
#define DEC_PARACHANGE_PLAN_LOOP_ITEM_LIST 86
#define DEC_PARACHANGE_STARTPLANLOOP 87
#define DEC_PARACHANGE_STOPPLANLOOP 88
#define DEC_PARACHANGE_TELNET 89
```

# 第 5 章 常量及数据结构

## 5.1 常量定义

**//解码器参数限制**

S2、S3 系列解码器限制：

```
#define    DEC_MAX_CHANNEL_NUM    8          //每个解码器最多通道数
#define    DEC_MAX_PICTURE_NUM    4          //每个解码通道最多画面数
```

S4 系列解码器限制：

```
#define    DEC_MAX_CHANNEL_NUM_NEW          16    //每个解码器最多通道数，
#define    DEC_MAX_VIRTUAL_CHANNEL_NUM      8     //每个解码器最多虚拟通道数
#define    DEC_MAX_SPLIT_SCREEN_NUM         16    //大屏最大的分屏个数
```

S5 系列解码器限制：

```
#define    DEC_MAX_FLOAT_WINDOW_NUM         4     //每个通道最多开窗个数。
#define    DEC_MAX_PICTURE_NUM              64    //每个解码通道最多画面个数
#define    DEC_MAX_WINDOW_TOTAL_NUM         4
#define    DEC_MAX_PHYSICAL_CHANNEL_NUM     96    //最大物理通道数
#define    DEC_MAX_SPLIT_CHANNEL_NUM        48    //最大拼接通道数
#define    DEC_MAX_CHANNEL_TOTAL_NUM        144   //最大通道数。
#define    DEC_MAX_ENCODE_CHANNEL_NUM       48    //最大内部编码通道数
```

通用参数限制：

```
#define    DEC_MAX_NVSITEM_NUM    64         //每个画面最多 NVS 数量
#define    DEC_MAX_DECODER        128        //最多连接解码器数
#define DEC_MAX_COM_NUM           2          // S4 及其以下版本限制:最大串口数。
#define DEC_MAX_COM_NUM_NEW       8          // S5 系列解码器限制：最大串口数。
#define DEC_MAX_USER_NUM          32         //最大用户数量
#define NVS_MAX_CHANNEL_NUM       160        // NVS 最大通道数
#define DEC_MAX_CHANNELID_LEN     32         //最大通道 ID 长度
#define DEC_NO_CHANNEL_PARAM      -1         //通道无关
#define DEC_NO_PIC_PARAM          -1         //画面无关
```

长度限制：

```
#define DEC_LEN_8          8
#define DEC_LEN_16         16
#define DEC_LEN_32         32
#define DEC_LEN_64         64
#define DEC_LEN_128        128
#define DEC_LEN_256        256

#define DEC_MAX_UDISK_ALL_CHANNEL_VIDEO    10
#define DEC_MAX_UDISK_CHANNEL_VIDEO        20
#define DEC_MAX_VIDEOLIST_SIZE             100
#define DEC_MAX_RESOLUTION_NUM             32
```

```
#define DEC_MAX_LAYOUT_NUM                  32
#define DEC_MAX_PLAN_LOOP_ITEM_NUM          16
```

## 5.2 数据结构定义

**//解码器登录状态**

```
enum
{
    DEC_LOGON_RETRY          = 2,
    DEC_LOGON_SUCCESS        = 0,
    DEC_LOGON_ING            = 1,
    DEC_LOGON_FAILED         = -1,
    DEC_LOGON_TIMEOUT        = -2,
    DEC_NOT_LOGON            = -3,
};
```

**//解码器参数定义**

```
typedef struct
{
    char                m_ip[16];                   //解码器 IP 地址
    char                m_submask[16];              //解码器子网掩码
    char                m_gateway[16];              //解码器的网关地址
    char                m_dns[16];                  //DNS 地址
    char                m_cMac[18];                 //物理地址
    char                m_ddnsIP[16];               //解析服务器 IP 地址
    char                m_ddnsUser[16];             //解析服务器用户名
    char                m_ddnsPass[16];             //解析服务器密码
    unsigned  short     m_ddnsPort;             //解析服务器端口
    unsigned  char      m_ispal;                //0:PAL；1:NTSC
    unsigned  char      m_language;             //0:中文；1:英文
    unsigned  char      m_rs485type;            //485 协议 ID，0:透明通道，1:Pelco-P
    unsigned  char      m_rs485address;         //485 地址（取值范围：0-255）
    int                 m_rs485baudrate;  //485 波特率 2400,4800,9600，19200，38400，57600
    unsigned char       m_rs485databit;             //485 数据位，默认为 8
    unsigned char       m_rs485stopbit;             //485 停止位，默认为 1
    unsigned char       m_rs485checkbit;
    //485 校验位，四种可选：0 无校验；1 奇校验；2 偶校验；3 固定为 1；4 固定为 0
    unsigned char m_rs485flowcontrol;           //485 流控，默认为无流控，设置无效
    int             m_Alarmout;                 //报警输出联动关系，&0x01:端口报
    警，&0x02:丢失报警，&0x04:移动报警，1 表示联动输出，0 表示不联动输出。
    int             m_AlarmMode;            //报警输出模式：0 低电平触发;1 高电平触发
    int             m_iChannel;             //解码通道数，0-8
    unsigned char m_nChanMode[8];           //每个通道的工作模式（1 单画面；4 四画面）
```

```
    int             m_iVGASize[DEC_MAX_CHANNEL_NUM];
#ifdef __WIN__
    char            m_cRegSvrIP[32];
    int             m_iRegSvrPort;
    int             m_iRegSvrEnable;
    char            m_cPUID[21];
    char            m_cPUName[DEC_LEN_32+1];
    int             m_iVolume;
    int             m_iSelPic[2];
    DZ_INFO_PARAM       m_dzInfoParam;
    TGetSipVideoChannel m_SipVideoChannel;
    STR_VideoParam      m_VideoParam[DEC_MAX_CHANNEL_NUM];
    int                 m_iRegChannelNum[2];
    char                m_cRegChannelPUID[2][DEC_LEN_32+1];
#endif
}TDECPARAM;

//视频通道参数定义
typedef struct
{
    union
    {
        char            m_cRTSPUrl[RTSP_URL_LEN+1];         //RTSP 地址
        struct
        {
            char        m_serName[33];  //服务器名称(暂时保留)
            char        m_serIP[33];    //服务器地址，可以是 IP 地址，也可以是域名；
            char        m_serProxy[16]; //服务器上级代理
        };
    };

    unsigned  char    m_serChan;            //服务器通道
    unsigned  char    m_serStream;          //服务器码流类型
    unsigned  char    m_serNetmode;
    //传输协议 1：TCP，2：UDP，3：多播；（解码器连接视频只能用 TCP 连接）
    unsigned  char    m_bUseddns;           //通过解析服务器连接
    unsigned  short   m_serPort;            //服务器端口
    int               m_bisView;            //当前是否在连接状态 (暂时不使用)
    char              m_username[20];       //用户名
    char              m_password[20];       //密码
    int               m_holdtime;           //保持时间，取值范围：10----1000 秒
    char              m_deviceType[16];     //设备控制协议（需要翻译到的协议）
    unsigned char     m_iAddress;           //NVS 设备地址（取值范围：0-255）
} TNVSITEM;
```

```
//视频通道参数扩展定义
#define MAX_ENCYPTPWD_LEN     16
typedef struct __tagTNVSITEMEX
{
    int    m_iStructLen; //如果以后要扩展，可以根据这个长度来判断扩展字段的内容
    TNVSITEM          m_stOldItem;
    char m_cEncyptPWD[MAX_ENCYPTPWD_LEN + 1];    //加密密码
    int    m_iDevPlatType;  //设备平台类型：0，私有协议；1，Onvif；3，RTSP 流；
    void*             m_pvReserved;                              //为了便于扩展
    char              m_cMultIP[33];                             //组播地址协议
    int               m_iMultPort;                               //组播端口号
}TNVSITEMEX, *LPTNVSITEMEX;

//服务器支持的协议列表定义
typedef struct
{
    int         m_iCount;                 //服务器支持的协议类型数
    char        m_cProtocol[128][16];     //各协议名称，最多支持 128 个协议，每条协议名
                                          称最长 15 字节；
} TDECPROTOCOL;

#define MAX_PROTOCOL_NUM                128       //DEC 最多支持 128 个协议
#define MAX_PROTOCOL_NAME_LEN32         32        //设备支持的协议名称的最大长度
typedef struct
{
    int         m_iCount;                 //协议个数
    char        m_cProtocol[MAX_PROTOCOL_NUM][MAX_PROTOCOL_NAME_LEN32]
                  //各协议名称，最多支持 128 个协议，每条协议名称最长 31 字节；
} TDECPROTOCOL_EX;

//SDK 版本号定义
typedef struct
{
    unsigned    short     m_ulMajorVersion;
    unsigned    short     m_ulMinorVersion;
    unsigned    short     m_ulBuilder;
    char*                 m_cVerInfo;
}NVDSDK_VERSION;

//OSD 叠加 Logo 属性定义
typedef struct
{
    unsigned    char    m_enable;                 //是否叠加 Logo
```

```
    int                 m_iPosX;                //叠加 Logo 横坐标
    int                 m_iPosY;                //叠加 Logo 纵坐标
    unsigned  char  m_iAlpha;                   //透明度(0~100)，为完全不透明
    int                 m_iTransparentColor;    //背景色，RGB 格式(0x00000000~0x00ffffff)
}TLOGOPARAM;


typedef struct tagChannelLogoParam
{
    int                     iSize;
    TLOGOPARAM      tLogoParam;
}ChannelLogoParam, *pChannelLogoParam;

//SIP 协议相关
typedef struct
{
    int         m_iChannelNo;
    char        m_cChannelID[DEC_MAX_CHANNELID_LEN+1];
    int         m_iLevel;
    int         m_iPtzTime;
} TSetSipVideoChannel;

typedef struct
{
    int         m_iBeginChannelNo;
    int         m_iChannelCount;
    char
          m_cChannelID[DEC_MAX_CHANNEL_NUM][DEC_MAX_CHANNELID_LEN+1];
    int         m_iLevel[DEC_MAX_CHANNEL_NUM];
    int         m_iPtzTime[DEC_MAX_CHANNEL_NUM];
} TGetSipVideoChannel;

typedef struct tagSipVideoChannel
{
    int         iSize;
    char        cChannelID[DEC_MAX_CHANNELID_LEN+1];
    int         iLevel;
    int         iPtzTime;
}SipVideoChannel, *PSipVideoChannel;

typedef struct tagGetSipVideoChannel_Ex
{
    int         m_iBuffSize;
    int         m_iBeginChannelNo;
```

```
    int         m_iChannelCount;
    char
    m_cChannelID[DEC_MAX_CHANNEL_NUM_NEW][DEC_MAX_CHANNELID_LEN+1
];
    int         m_iLevel[DEC_MAX_CHANNEL_NUM_NEW];
    int         m_iPtzTime[DEC_MAX_CHANNEL_NUM_NEW];
}GetSipVideoChannel_Ex, *pGetSipVideoChannel_Ex;


//获取通道数和每个通道的画面数
typedef struct tagChnNumAndMode
{
    int             iBufSize;               //结构体大小
    int             iLocalChnNum;               //本地通道数
    int             iVOChnNum;              //拼控通道数
    int             iLocalChnMode[DEC_MAX_CHANNEL_NUM_NEW];
    int             iVOChnMode[DEC_MAX_VIRTUAL_CHANNEL_NUM];
}ChnNumAndMode, *pChnNumAndMode;

typedef struct tagDecTimezone
{
    int iTimezone;
}DecTimezone, *PDecTimezone;

typedef struct tagDecScreenDisplay
{
    int iDisplay;
}DecScreenDisplay, *PDecScreenDisplay;

//设备支持平台信息定义
typedef struct
{
    int         m_iCount;                   //支持的平台数目
    char        m_cPlatformName[10][32];    //各个平台的名称
}TPLATFORMINFO;;

//Encodemode: 编码方式
enum
{
    ENCODE_H264         = 0,            //H.264 编码算法
    ENCODE_MJPEG        = 1,            //MJPEG 编码算法
    ENCODE_BUTT         = 2,
};
```

**//videosize: 视频大小**

```
enum
{
    VIDEO_QCIF      = 0,        //Quarter Cif   172*144,
    VIDEO_HCIF      = 1,        //Half Cif      352*144
    VIDEO_FCIF      = 2,        //Full Cif      352*288
    VIDEO_HD1       = 3,        //Half D1       704*288
    VIDEO_FD1       = 4,        //Full D1       704*576
    VIDEO_QVGA          = 6,    //QVGA          320x240
    VIDEO_VGA           = 7,    //VGA           640*480
    VIDEO_HVGA          = 8,    //HVGA          640*240
    VIDEO_HD_720P       = 9,    //720p          1280*720
    VIDEO_HD_1080P      = 11,   //1080P         1920*1080
    VIDEO_HD_QXGA       = 13,       //300W          2048*1536
    VZ_4M               = 0x0210,   //400W          2560*1440
    VZ_4MB              = 0x0220,   //400WB         2592*1520
    VZ_QSXGA        = 0x500,    //500W(3)       2560*2048
    //走廊模式
    F720P_9_16      = 0x100009,     //720x 1280
    F1080P_9_16     = 0x10000C,     //1080x 1920
    F4M_9_16        = 0x100210,     //1440x2560
    F4MB_9_16       = 0x100220,     //1520x2592
    VIDEO_BUTT
};
```

**//VGA 分辨率**

```
enum
{
    VGA_800X600     = 1,    //800*600 60HZ
    VGA_1024X768    = 2,    //1024*768 60HZ
    VGA_1280X1024   = 3,    //1280*1024 60HZ
    VGA_1280X720P_60,
    VGA_1920X1080P_60,
    VGA_1280X720P_50,
    VGA_1920X1080P_50,
    VGA_1920X1080I_60,
    VGA_1920X1080I_50,
    VGA_1366x768_60,
    VGA_1440x900_60,
    VGA_1280x800_60,
    VGA_2560x1600_30,
    VGA_3840x2160_30,
    VGA_3840x2160_60,
    VGA_BUTT,
```

```
};

//视频源服务器类型
#define   DEVTYPE_NORMAL          0
#define   DEVTYPE_ONVIF           1
#define   DEVTYPE_PUSH            2
#define   DEVTYPE_RTSP            3
#define   DEVTYPE_ENCODE          4

//流控类型
#define PUSH_STREAM_CMD_TYPE     0
#define PUSH_STREAM_CMD_PAUSE    0
#define PUSH_STREAM_CMD_FAST     1
#define PUSH_STREAM_CMD_SLOW     2

//push 流控制暂停状态
#define PUSH_REAL_TIME_STREAM_STATE     0x100    //实时流
#define PUSH_PLAYBACK_STREAM_PLAY       0        //回放流播放
#define PUSH_PLAYBACK_STREAM_PAUSE      1        //回放流暂停

//缓冲区状态
#define       RET_BUFFER_IS_ALREADY_FULL      (-10)
//    缓冲区已经满了，数据没有放入缓冲区，数据需要重新发送
#define       RET_BUFFER_FULL                 (-11)
//    缓冲区满，数据不需要重新发送
#define       RET_BUFFER_WILL_BE_FULL         (-18)
//    即将满，降低送入数据的频率
#define       RET_BUFFER_WILL_BE_EMPTY        (-19)
//    即将空，提高送入数据的频率
#define      RET_BUFFER_IS_OK                 (-20)
//   正常可以放数据
#define       RET_BUFFER_IS_EMPTY             (-21)
//    缓冲区空
#define       RET_PAUSE_STATE                 (-30)
//    暂停状态

//push 流速度类型
#define       PUSH_SLOW_SPEED       0
#define       PUSH_NORMAL_SPEED     1
#define       PUSH_FAST_SPEED       2

//push 流添加 StartPush 结构体
#define   MAX_ENCRYPT_KEY         16
#define   VIDEO_HEADER_SIZE       88
```

```
typedef struct
{
    int     m_iSize;                                    //结构体大小
    char    m_cFileHeader[VIDEO_HEADER_SIZE];  //视频头信息
    char    m_cEncryptKey[MAX_ENCRYPT_KEY + 4];
     //前端设备视频解密密码，为空则表示不加密，至多 16 个字符且保证四字节
}START_PUSH_PARAM;

//配置码流信息
#define     MIN_CONF_CMD                        0
#define     CONF_CMD_VIDEOHEAD                  MIN_CONF_CMD + 1
#define     CONF_CMD_AUDIOVIDEO                 MIN_CONF_CMD + 2
#define     MAX_CONF_CMD                        MIN_CONF_CMD + 2

//升级文件结果状态
#define  RET_UPGRADE_FINISH      0
#define  RET_UPGRADE_FAILED      -1
#define  RET_UPGRADE_ERROR       2

//置顶置底
#define SET_TOP     1
#define SET_DOWN    0

//设备参数命令码
#define CMD_DEC_CFG_PROTOCOL                0       //PTZ 控制协议
#define CMD_DEC_CFG_ALARM_LINK_SWITCH       1       //报警联动切换
#define CMD_DEC_CFG_CREATE_VVO              2       //创建拼控输出通道 VVO
#define CMD_DEC_CFG_SPLIT_SCREEN        3       //自定义画面分割方式
#define CMD_DEC_CFG_CHANN_STATE         4       //获取解码器各个通道的状态
#define CMD_DEC_CFG_DISP_VONUM          5       //控制输出设备显示物理编号
#define CMD_DEC_CFG_CHANN_NUM       6   //获取解码器本地通道数和虚拟通道数
#define CMD_DEC_CFG_ALARM_SCHEDULE      7       //设置智能分析报警布防模板
#define CMD_DEC_CFG_ALARM_LINK          8       //设置智能分析联动
#define CMD_DEC_CFG_ALARM_NOTIFY        9       //发送警情
#define CMD_DEC_CFG_ALARM_IN_OUT        10      //设置报警输入输出
#define CMD_DEC_CFG_SCHEDULE_ENABLE     11      //设置智能分析布防使能
#define CMD_DEC_CFG_VGA_SIZE            12      //获取 VGA 大小
#define CMD_DEC_CFG_SIP_VIDEO_CHANNEL   13      //获取 SIP 相关参数
#define CMD_DEC_CFG_VIDEO_PARAM         14      //获取视频参数
#define CMD_DEC_CFG_SYSTEM_TIME         15      //设置系统时间
#define CMD_DEC_CFG_DEV_INFO            16      //获取设备信息
#define CMD_DEC_CFG_TIMEZONE            17      //时区
#define CMD_DEC_CFG_CHANNEL_REGINFO         18      //通道注册信息
#define CMD_DEC_CFG_SCREEN_DISPLAY          19      //画面显示/隐藏
```

```
#define CMD_DEC_CFG_SCREEN_CHANGEAREA 20 //画面自定义大小/区域
#define CMD_DEC_CFG_FLOAT_WINDOW 22 //开窗
#define CMD_DEC_CFG_SAVE_PLAN 23 //设置预案
#define CMD_DEC_CFG_APPLY_PLAN 24 //应用预案
#define CMD_DEC_CFG_PLANALIAS 25 //预案别名
#define CMD_DEC_CFG_BASE_OFFSET 26 //起始编号
#define CMD_DEC_CFG_LOGOPARAM 27 //logo 参数
#define CMD_DEC_CFG_DECOSD 28 //OSD 叠加操作
#define CMD_DEC_CFG_PICNUM 29 //画面数量
#define CMD_DEC_CFG_WINDOW_SEQUENCE 30 //窗口次序
#define CMD_DEC_CFG_REGSERVER 31 //平台服务器注册
#define CMD_DEC_CFG_PUPARAM 32 //设备注册
#define CMD_DEC_CFG_DZ_INFO 33 //定制信息
#define CMD_DEC_CFG_SELECT_PIC 34 //选择画面
#define CMD_DEC_CFG_CHANNEL_REGINFO_EX 35 //通道注册信息
#define CMD_DEC_CFG_SIP_VIDEO_CHANNEL_EX 36 //SIP 通道注册信息
#define CMD_DEC_CFG_DEVCOMMONNAME 37 //通道别名
#define CMD_DEC_CFG_EXPORT_CONFIG 38 //导出配置
#define CMD_DEC_CFG_SHOW_OFFLINE_CHAN 39 //是否显示不在线通道
#define CMD_DEC_CFG_PLATFORM_LIST 40 //平台列表
#define CMD_DEC_CFG_COM_PARAM 41 //串口参数
#define CMD_DEC_CFG_SINGLEPIC 42 //双击放大/缩小控制
#define CMD_DEC_CFG_AUTOTEST_SYSTEMTYPE 43
#define CMD_DEC_CFG_AUTOTEST_DEVICEPRODUCTER 44
#define CMD_DEC_CFG_SCREEN_REGION 45
#define CMD_DEC_CFG_SCREEN_BASICPARA 46
#define CMD_DEC_CFG_SCREEN_OUTPUT_MODE 47
#define CMD_DEC_CFG_PICLEVEL_RELATION 48
#define CMD_DEC_CFG_SYNCLOOPCTRL 49
#define CMD_DEC_CFG_SYNCLOOPTIME 50
#define CMD_DEC_CFG_ENCODERPAPA 51
#define CMD_DEC_CFG_HTTPPORT 52
#define CMD_DEC_CFG_SHOWICON 53
#define CMD_DEC_CFG_ALARM_EFFECT_LIGHT 54
#define CMD_DEC_CFG_ALARM_EFFECT_REGION 55
#define CMD_DEC_CFG_ALARM_EFFECT_TEXT 56
#define CMD_DEC_CFG_PLANLOOPITEM 57
#define CMD_DEC_CFG_PLANLOOP 58
#define CMD_DEC_CFG_PLANLOOPITEMDEL 59
#define CMD_DEC_CFG_LANPARAM_WORKMODE 60
#define CMD_DEC_CFG_SIP_VIDEO_CHANNEL_EX_V2 61
#define CMD_DEC_CFG_CHANNEL_MAX_WIN 62
#define CMD_DEC_CFG_CHANNEL_LAYOUTLIST 63
#define CMD_DEC_CFG_CHANNEL_RESOLUTIONLIST 64
```

```
#define CMD_DEC_CFG_GETSCREENCTLPROTOCOLS        65
#define CMD_DEC_CFG_SCREENCRL_PROTOCOL           66
#define CMD_DEC_CFG_SET_CHANNEL_VIDEO            67
#define CMD_DEC_CFG_CURVIDEO                     68
#define CMD_DEC_CFG_VIDEO_PLAYMODE               69
#define CMD_DEC_CFG_GETPLANLOOPSTATUS            70
#define CMD_DEC_CFG_START_LOOP                   71
#define CMD_DEC_CFG_STOP_LOOP                    72
#define CMD_DEC_CFG_ALL_CHANNEL_VIDEO            73
```

**//报警**

```
define DEC_ALARM_TYPE_BASE          128
#define DEC_ALARMTYPE_VIDEOLOST     128 + 0
#define DEC_ALARMTYPE_NET_PORT      128 + 1
#define DEC_ALARMTYPE_MOVE          128 + 2
#define DEC_ALARMTYPE_LOCAL_PORT    128 + 7
#define DEC_MAX_ALARM_TYPE          128 + 8

#define ALARM_LINK_SWITCH_ENBALE        1
#define ALARM_LINK_SWITCH_DISENBALE     0
//报警联动切换
typedef struct tagAlarmLinkSwitch
{
    int   iBufSize;      //结构体大小
    int   iAlarmType;  //报警类型：0，视频丢失；1，端口报警；2，移动侦测；3，视频遮挡；4，智能分析；5，音频丢失；6,温湿度报警；
                        //128 之后解码器使用，模式=128 + iType（上一行表示的数值）；
                        //128，视频丢失；
                        //129，网络端口报警；
                        //130，移动侦测；
                        //131～159，预留
                        //160，本地端口报警；"
    int   iPortNo;          //端口号 仅当报警类型为本地端口报警时，该字段有效
    int   iChannelNo;                   //通道号
    int   iPos;                         //画面号
    char cSerName[DEC_LEN_32];          //前端设备名称
    char cSerIP[DEC_LEN_16];            //前端设备 IP 地址
    char cSerProxy[DEC_LEN_16];         //前端设备代理 IP
    int   iSerChan;                     //前端设备通道号
    int   iSerStreamNo;               //主副码流    0,主码流；1，副码流
    int   iSerNetmode;                //服务器网络模式 1，TCP；2，UDP；3，多播
    int    iUseddns;                  //是否使用 ddns   0，IP；1，域名；3，主动模式
    int   iSerPort;                   //服务器端口号
    char cUserName[DEC_LEN_32];         //前端设备登录用户名
```

```
    char cPassword[DEC_LEN_32];         //前端设备登录密码
    int   iHoldTime;                    //停留时间     10-999
    char cEncrypt[DEC_LEN_32];          //前端设备视频解密密码
    int   iSerType;//前端设备类型：0，私有协议；1，Onvif；3，rtsp 流；4，内部编码通道
    char cRtspUrl[DEC_LEN_256];
    int   iLinkSwitchEnable;
}AlarmLinkSwitch, *pAlarmLinkSwitch;

//创建拼控输出通道 VVO
typedef struct tagCreateVVOInfo
{
    int iBufSize;                                   //结构体大小
    int iVVOChannelNo;                              //拼控通道号
    int iRows;                                      //拼控屏行数
    int iCols;                                      //拼控屏列数
    int iInputChannel[DEC_MAX_PHYSICAL_CHANNEL_NUM];
            //各分屏绑定的输出通道号，没绑定则用 0x7fffffff 表示
}CreateVVOInfo, *pCreateVVOInfo;

//画面分割参数，使用万分比表示
typedef struct tagScreenPara
{
    int iBufSize;           //结构体大小
    int   iX;               //起始点横坐标
    int   iY;               //起始点纵坐标
    int   iWidth;           //画面宽
    int   iHeight;          //画面高
}ScreenPara, *pScreenPara;
//自定义画面分割方式
typedef struct tagSplitScreen
{
    int             iBufSize;                                   //结构体大小
    int             iVVOChannelNo;                              //拼控通道号
    int             iPicNum;                                    //画面数
    ScreenPara      tSplitScreenPara[DEC_MAX_PICTURE_NUM];      //画面参数
}SplitScreen, *pSplitScreen;

//解码器各个通道的状态
#define CHANEEL_ON_LINE         1
#define CHANNEL_OFF_LINE    0

#define DECODE_CHANNEL          0
#define ENCODE_CHANNEL          1
typedef struct tagLocalChannelState
```

```
{
    int             iBufSize;                   //输入参数：结构体大小
    int             iChannelNo;                 //输入参数：通道号
    int             iState;                     //输出参数：在线状态, 0-不在线   1-在线
    int             iType;                      //输入参数：通道类型, 0-解码通道  1-编码通
道
}LocalChannelState, *pLocalChannelState;

#define DEC_MAX_DAYS   7
#define DEC_MAX_TIMESEGMENT       4
//Schedle time
typedef struct tagDEC_SCHEDTIME
{
    int         iStartHour;
    int         iStartMin;
    int         iStopHour;
    int         iStopMin;
    int         iEnable;
}DEC_SCHEDTIME, *PDEC_SCHEDTIME;

//时间段
typedef struct tagAlarmScheduleParam
{
    int                 iBuffSize;
    int                 iChannelNo;   //标识进行智能分析的通道
    int                 iAlarmType;   //128，视频丢失；129，网络端口报警；130，移动侦
测；131～134，预留；135，本地端口报警；
    int                 iWeekday;     //星期日到星期六（0-6）
    int                 iParam1;      //iType=4 时表示规则 ID
    int                 iParam2;      //iType= 4 时表示事件类型
    DEC_SCHEDTIME       timeSeg[DEC_MAX_DAYS][DEC_MAX_TIMESEGMENT];
    void*               pvReserved;
}AlarmScheduleParam, *PAlarmScheduleParam;

typedef struct tagAlarmLink
{
    int   iBuffSize;
    int iChannelNo;             //取值范围据设备类型而定
    int   iAlarmType;           //128，视频丢失；129，网络端口报警；130，移动侦测；
131～134，预留；135，本地端口报警；
    int   iAlarmTypeParam;      //取值由 iAlarmType 而定，假如 iAlarmType 为智能分析，
则 iAlarmTypeParam 表示 iRuleID
    char cReserved[DEC_LEN_32];
    int   iLinkType;            //联动类型,0，联动声音提示；1，联动屏幕显示；2，联动
```

输出端口；3，联动录像；4，联动 PTZ；5，联动抓拍；

int iLinkParam1; //联动参数,取值由 iLinkType 而定：iLinkType=0,1 时，iParam1 表示使能 iEnable。0，不使能；1，使能。

//iLinkType=2,3,5,6 时，iParam1 表示按位使能 iEnableByBits，从最低位至最高位每一位表示一个音视频通道/输出端口的使能。

//iLinkType=4 时，iParam1 表示待联动的通道号 iLinkChannel,取值范围据设备类型而定；

//iParam2 表示待联动的类型 iLinkType：0，不联动该通道 PTZ，1 预置位，2 轨迹，3 路径；iParam3 表示 PTZ 号 iNo，

//根据 iParam2 的类型分别表示预置位号，轨迹号和路径号

int iLinkParam2; //联动参数

int iLinkParam3; //联动参数

}AlarmLink, *PAlarmLink;

```
#define DEC_MAX_PORT_NUM            64
#define MAX_ALARM_IN_OUT_TYPE    4

//设置报警输入是高电平触发还是低电平触发
#define DEC_ALARM_IN_LHP        0
//设置报警输出是高电平触发还是低电平触发
#define DEC_ALARM_OUT_LHP      1
//设置报警输入端口使能
#define DEC_ALARM_IN_PORT       2
//设置报警输出端口使能
#define DEC_ALARM_OUT_PORT    3

typedef struct tagAlarmInAndOut
{
    int   iBuffSize;
    int   iType;          //0：报警输入高低电平设置；1：报警输出高低电平设置；2：报警输入端口使能；3：报警输出端口使能
    int   iPortNo;        //输入输出端口号
    int   iPara1;         //与 iType 取值有关
    int   iPara2;         //保留
}AlarmInAndOut, *PAlarmInAndOut;

typedef struct tagAlarmNotify
{
    int   iBuffSize;
    int   iAlarmType;  //128，视频丢失；129，网络端口报警；130，移动侦测；131～134，预留；135，本地端口报警；
    int   iChannelNo;  //通道号,iType = 160，表示解码器本地端口号
    int   iState;         //1，报警；0，消警
}AlarmNotify, *pAlarmNotify;
```

```
typedef struct tagScheduleEnable
{
    int   iBuffSize;
    int iChannelNo;
    int   iAlarmType;        //128，视频丢失；129，网络端口报警；130，移动侦测；131～
134，预留；135，本地端口报警；
    int   iEnable;           //是否使能,0：不使能 1：使能
    int   iParam1;           //参数 1
    int   iParam2;           //参数 2
    int   iParam3;           //参数 3
}ScheduleEnable, *pScheduleEnable;

typedef struct tagVGASize
{
    int   iBuffSize;
    int iChannelNo;
    int iVGASize;
}VGASize, *pVGASize;

typedef struct
{
    int iSize;
    int iChanNum;
    int iAlarmInNum;
    int iAlarmOutNum;
    int iComNum;
    int iSplitChanNum;
    int iEncChanNum;
    int iUsbInterfaceNum;
    int iNetCardNum;
}tDevInfo, *ptDevInfo;

typedef struct tagChannelRegInfoEx
{
    int         iBuffsize;
    char        cRegChannelPUID[DEC_LEN_32+1];
}ChannelRegInfoEx, *pChannelRegInfoEx;

#define DEC_MAX_MAIN_FUNC_TYPE                  8
#define DEC_MAX_SUB_FUNC_TYPE                   8
typedef struct _tagDecAbilityLevel
{
    int         iSize;
    int         iMainFuncType;              //main function type
```

```
    int         iSubFuncType;               //sub function type
    char        cParam[DEC_LEN_256];        //Capability Description
} DecAbilityLevel, *pDecAbilityLevel;

//日志
#define MAX_PAGE_LOG_SIZE 20
typedef struct  _tagDecLogQuery
{
    int             iChannelNo;             //通道号
    int             iLogType;               //日志类型
    int             iLanguage;              //语言类型
    NVS_FILE_TIME  struStartTime;           //开始时间
    NVS_FILE_TIME  struStopTime;            //结束时间
    int             iPageSize;              //页大小
    int             iPageNo;                //页编号
}DecLogQuery, *PDecLogQuery;

#define MAX_LOG_LEN     130
typedef struct    _tagDecLogData
{
    int                 iChannel;
    int                 iLogType;
    NVS_FILE_TIME   struStartTime;      /* File start time */
    char                cLogContent[MAX_LOG_LEN];
}DecLogData,*PDecLogData;

//集群
#define MIN_CLUSTER_OP                                  0
#define CLUSTER_MERAGE                          MIN_CLUSTER_OP
#define CLUSTER_DEMERAGE                        (MIN_CLUSTER_OP+1)
#define CLUSTER_ALIAS                           (MIN_CLUSTER_OP+2)
#define CLUSTER_CLUSTERINFO                     (MIN_CLUSTER_OP+3)
#define CLUSTER_CONTROL_CARD_VERSIONINFO  (MIN_CLUSTER_OP+4)
#define CLUSTER_WORK_CARD_VERSIONINFO     (MIN_CLUSTER_OP+5)
#define CLUSTER_CONTROL_CARD_STATE              (MIN_CLUSTER_OP+6)
#define CLUSTER_WORK_CARD_STATE                 (MIN_CLUSTER_OP+7)
#define CLUSTER_IP                              (MIN_CLUSTER_OP+8)
#define CLUSTER_DEVICE_INFO                     (MIN_CLUSTER_OP+9)
#define CLUSTER_SEARCH                          (MIN_CLUSTER_OP+10)
#define CLUSTER_DEVICE_STATE                    (MIN_CLUSTER_OP+11)
#define CLUSTER_CTRL_MAC                        (MIN_CLUSTER_OP+12)
#define MAX_CLUSTER_OP                          (MIN_CLUSTER_OP+13)

typedef struct tagClusterMerage
```

```
{
    int         iSize;
    char        cIP[LEN_64];
    int         iPort;
    char        cUserName[LEN_32];
    char        cPassword [LEN_32];
    char        cClusterId [LEN_64];
}ClusterMerage, *pClusterMerage;

typedef struct tagClusterAlias
{
    int         iSize;
    char        cAlias[LEN_64];
}ClusterAlias, *pClusterAlias;

typedef struct tagClusterInfo
{
    int         iSize;
    char        cClusterId[LEN_64];
    int         iDevNum;
}ClusterInfo, *pClusterInfo;

#define MAX_CLUSTER_DEVICE_COUNT    4
#define MAX_CONTROL_CARD_COUNT      2
#define MAX_CONTROL_NET_COUNT       2
#define MAX_WORK_CARD_COUNT         16

#define CARD_TYPE_CONTROL    1
#define CARD_TYPE_DECODE     2
#define CARD_TYPE_ENCODE     3

typedef struct tagClusterControlCardInfo
{
    int     iSize;
    int     iDevId;
    int     iCardId;
    char    cMasterVersion[LEN_64];
    char    cProxyVersion[LEN_64];
    char    cStreamVersion[LEN_64];
    int     iProductModel;
}ClusterControlCardInfo, *pClusterControlCardInfo;

typedef struct tagClusterWorkCardInfo
{
```

```
    int     iSize;
    int     iDevId;
    int     iCardId;
    int     iCardType;
    char    cVersion[LEN_64];
    int     iEncChnNo;      //编码通道通道号，编码卡使用
}ClusterWorkCardInfo, *pClusterWorkCardInfo;

typedef struct tagClusterDeviceInfo
{
    int     iSize;
    int     iDevId;
    int     iCtrlCardNum;
    int     iSlotdNum;
    int     iAlarmInNum;
    int     iAlarmOutNum;
    int     iComNum;
}ClusterDeviceInfo, *pClusterDeviceInfo;

typedef struct tagClusterCardState
{
    int     iSize;
    int     iDevId;
    int     iCardId;
    int     iState;
    int     iType;
    int     iMaxCaptility;
    int     iCurrentCaptility;
    int     iProgress;
}ClusterCardState, *pClusterCardState;

#define SEEK_DEVICE_IPC    1
#define SEEK_DEVICE_NVD    2

#define SEEK_TYPE_IP       0
#define SEEK_TYPE_DNS   1
#define SEEK_TYPE_DMS  2
typedef struct tagClusterSearchParam
{
    int iSize;
    int iSeekDevice; // SEEK_DEVICE_IPC、SEEK_DEVICE_NVD
    int iSeekType; //SEEK_TYPE_IP、SEEK_TYPE_DNS、SEEK_TYPE_DMS
}ClusterSearchParam, *PClusterSearchParam;
```

```
typedef struct tagClusterSearchResult
{
	int iSize;
	char cMac[LEN_64];
	char cIP[LEN_64];
	char cMask[LEN_64];
	char cGateWay[LEN_64];
	char cDNS[LEN_64];
	int iChannelNum;
	int iServerPort;
	int iClientPort;
	int iDeviceType;
	int iHttpPort;
	int iProductModel;
	char cFactoryID[LEN_64];
	char cKernelVersion[LEN_64];
	char cOcxVersion[LEN_64];
	char cClusterId[LEN_64];
	char cClusterAlias[LEN_64];
	int iDevNum;
	int iSeekType;//SEEK_TYPE_IP、SEEK_TYPE_DNS、SEEK_TYPE_DMS
}ClusterSearchResult, *PClusterSearchResult;

#define CHANGEIP_TYPE_SINGLE 0
#define CHANGEIP_TYPE_ALL   1
typedef struct tagClusterChangeIP
{
	int	iSize;
	int	iDevId;
	int	iCardId;
	char	cIP[LEN_16];
	char	cSubMask[LEN_16];
	char	cGateway[LEN_16];
	char	cDns[LEN_16];
	int	iType; // CHANGEIP_TYPE_SINGLE ,CHANGEIP_TYPE_ALL
}ClusterIP, *PClusterChangeIP;

typedef struct tagDecCtrlMac
{
	int iSize;
	int iDevId;
	int iCardId;
	int iNetId;
	char cPcMac[LEN_64];
```

```
}DecCtrlMac, *PDecCtrlMac;

//开窗漫游
typedef struct tagFloatWindow
{
	int	iSize;
	int	iEnable;
	ScreenPara tArea;
}FloatWindow, *pFloatWindow;

#define WINDOW_SEQUENCE_TOP		1
#define WINDOW_SEQUENCE_BOTTOM	0

typedef struct tagWindowSequence
{
	int iSize;
	int iSequence; // WINDOW_SEQUENCE_TOP 、WINDOW_SEQUENCE_BOTTOM
}WindowSequence, *PWindowSequence;

//预案
#define MAX_PLAN_COUNT 16
typedef struct tagDecPlan
{
	int	iSize;
	int	 iPlanId;
	char	cSaveTime[LEN_64];
}DecPlan, *pDecPlan;

typedef struct tagApplyPlan
{
	int	iSize;
	int	 iPlanId;
}ApplyPlan, *pApplyPlan;

typedef struct tagPlanAlias
{
	int	iSize;
	int	iPlanId;
	char	cAlias[LEN_64];
}PlanAlias, *pPlanAlias;

//字符叠加
#define ALIGN_ABOUT_BWLOW_MIDDLE	0		//上下居中
#define ALIGN_ABOUT			1		//上对齐
```

```
#define ALIGN_BWLOW                     2       //下对齐

#define ALIGN_LEFT_RIGHT_MIDDLE         0       //左右对齐
#define ALIGN_LEFT                      1       //左对齐
#define ALIGN_RIGHT                             2               //右对齐

#define SHOW_STATIC                             0               //静态显示
#define SHOW_DYNAMIC_LEFT_RIGHT         1               //左右移动动态显示
#define SHOW_DYNAMIC_ABOUT_BWLOW        2               //上下移动动态显示

typedef struct tagDecOsd
{
    int             iSize;
    char            cOsd[LEN_256];
    int             iColor;
    int             iBackColor;
    int             iDiaphaneity;//0-100
    int             iFontSize; // 1-5
    ScreenPara      tArea;
    int             iAboutBelowAlign;
    int             iLeftRightAlign;
    int             iDynamic;
    int             iEnbale;
}DecOsd, *pDecOsd;

//大屏和开窗的起始编号
typedef struct tagBaseOffset
{
    int     iSize;
    int     iSplitScreen;
    int     iFloatWindow;
}BaseOffset, *pBaseOffset;
#define DEFAULT_SPLIT_SCREEN_OFFSET         10000
#define DEFAULT_OPEN_WINDOW_OFFSET          10000

//视频源服务器信息
#define MAX_SERVER_ITEM_COUNT     8192
typedef struct tagServerCommonInfo
{
    char    cServerIp[DEC_LEN_32];
    int     iServerPort;
    int     iServerChannelNo;
    int     iServerStreamNo;
    int     iServerNetmode;  //0:TCP, 1:UDP, 2:MUT
```

```
    int     iHoldTime;
    char    cServerUserName[DEC_LEN_64];
    char    cServerPassword[DEC_LEN_64];
    char    cServerPtzProtocol[DEC_LEN_32];
    int     iServerPtzAddress;
} ServerCommonInfo, *pServerCommonInfo;

typedef struct tagNormalServer
{
    int                 iSize;              //input para,the size of struct
    ServerCommonInfo    tCommonInfo;
    int                 iConnectMode;       //0:ip, 1:DDNS, 3:DSM
    char                cServerName[DEC_LEN_128];
    char                cServerProxy[DEC_LEN_32];
    char                cVideoDecryptPassword[DEC_LEN_32];
} NormalServer, *pNormalServer;

typedef struct tagOnvifServer
{
    int                 iSize;          //input para,the size of struct
    ServerCommonInfo    tCommonInfo;
} OnvifServer,*pOnvifServer;

typedef struct tagRtspServer
{
    int                 iSize;          //input para,the size of struct
    ServerCommonInfo    tCommonInfo;
    char                cRtspUrl[DEC_LEN_256];
    char                cMultIP[DEC_LEN_32];
    int                 iMultPort;
} RtspServer, *pRtspServer;

typedef struct tagEncodeChannel
{
    int                 iSize;          //input para,the size of struct
    ServerCommonInfo    tCommonInfo;
} EncodeChannel, *pEncodeChannel;

typedef union tagServerItem
{
    int             iSize;
    NormalServer    tNormal;
    OnvifServer     tOnvif;
    RtspServer      tRtsp;
```

```
    EncodeChannel      tEncode;
} ServerItem, *pServerItem;

//S5 解码器
typedef struct tagDecPuParam
{
    int    iSize;
    char cID[LEN_64];
    char cUsername[LEN_64];
}DecPuParam, *PDecPuParam;

typedef struct tagDecDzInfo
{
    int                    iSize;
    DZ_INFO_PARAM      tInfo;
}DecDzInfo, *PDecDzInfo;

typedef struct tagDecSelectPic
{
    int iSize;
    int iPicNo;
}DecSelectPic, *PDecSelectPic;

typedef struct tagPushStreamResponse
{
    int    iSize;
    long lPushId;
    int    result;
}PushStreamResponse, *PPushStreamResponse;

#define MAX_DEC_CHANNEL_TYPE      3
typedef struct tagDevCommonName
{
    int    iSize;
    int    iChannelType;
    char cChannelName[DEC_LEN_64];
} DevCommonName, *pDevCommonName;

typedef struct tagChannelPicNum
{
    int    iSize;
    int    iPicNum;
    int    iMode; // iPicNum == 20    1:20A 2: 20B
} ChannelPicNum, *pChannelPicNum;
```

```
typedef struct tagShowOfflineChan
{
    int     iSize;
    int     iEnable; //1----显示    0:----不显示
} ShowOfflineChan, *pShowOfflineChan;

typedef struct tagComParam
{
    int             iSize;
    int             iCom;
    unsigned char   ucType;         //485 协议 ID，0：透明通道，1:Pelco-P，...
    unsigned char   ucAddress;      //485 地址（取值范围：0-255）
    int             iBaudrate;      //485 波特率 2400，4800，9600
    unsigned char   ucDatabit;      //485 数据位，默认为 8
    unsigned char   ucStopbit;      //485 停止位，默认为 1
    unsigned char ucCheckbit; //485 校验位，四种可选，0：无校验；1：奇校验；2：偶校验；
    unsigned char   ucFlowcontrol;  //485 流控，默认为无流控，设置无效
} ComParam, *PComParam;
```

**//通用命令接口命令 ID**

```
#define DEC_COMMAND_PTZ_CONTROL         1       //设备控制协议
#define DEC_COMMAND_PROOF_ADJUST        2       //校正编码卡偏屏偏色
#define DEC_COMMAND_LOGWRITE            3       //cgi report log
#define DEC_COMMAND_USERNUM             4       //获取用户数目
#define DEC_COMMAND_USERINFO            5       //获取用户信息
#define DEC_COMMAND_GETUSERNUM          6       //获取用户数量
#define DEC_COMMAND_GETUSERINFO         7       //获取用户名和密码
#define DEC_COMMAND_GETLASTERROR        8       //获取设备错误信息
```

**//PTZ 控制**

```
typedef struct tagPtzControlPara
{
    int   iSize;
    int   iActionType;  //上、下、左、右、变倍、预置位等
    int   iControlType; //0，普通云台控制；1，电子云台 e-PTZ.缺省为 0
    int   iParam1;      //根据命令类型这两个参数的含义也不一样，水平速度或预置位号…
    int   iParam2;      //垂直速度…
} PtzControlPara, *pPtzControlPara;
```

**//示证调节**

```
#define     PROOFADJUST_OPERTYPE_MANU     1
#define     PROOFADJUST_OPERTYPE_AUTO     2
```

```
#define         PROOFADJUST_OUT_DEV_HDMI        1
#define         PROOFADJUST_OUT_DEV_VGA         2
#define         PROOFADJUST_TYPE_RESO           1
#define         PROOFADJUST_TYPE_COLOR          2
typedef struct tagProofAdjust
{
    int         iSize;
    int         iOperateMethod;         //1-手动， 2-自动
    int         iVoId;                  //1-HDMI， 2-VGA
    int         iType;                  //1-视频分辨率  2-视频颜色
} ProofAdjust, *pProofAdjust;

#define SHOW_LAST_FRAME 1

#define DELETE_LOOOP_ITEM_EXCEPT            0
#define ADD_LOOP_ITEM_EXCEPT                1

#define AUDIO_STATUS_CLOSE                  0
#define AUDIO_STATUS_OPEN                   1

#define TALK_STATUS_CLOSE       0
#define TALK_STATUS_OPEN        1

//ErrorID
#define DEC_LASTERROR_ALARMLINK     0x13000
//（报警联动）当前正在联动，不允许修改设置
#define DEC_LASTERROR_MODIFY_IP     0x13100     //ip 地址修改失败

typedef struct tagDecLastError
{
    int         iSize;
    int         iErrorID;
    char        cErrorInfo[DEC_LEN_64];
} DecLastError, *pDecLastError;



typedef struct tagDecSinglePic
{
    int         iSize;
    int         iPicNum;
    int         iState;     //单画面状态:0-非单画面模式, 1-单画面
} DecSinglePic, *pDecSinglePic;

typedef struct tagDecScreenRegion
```

```
{
    int     iSize;
    int     iEnable;    //使能
    int     iXPos;          //起始横坐标
    int     iYPos;          //起始纵坐标
    int     iPixelWidth;  //像素宽度
    int     iPixelHeight;  //像素高度
} DecScreenRegion, *pDecScreenRegion;

typedef struct tagDecScreenBasicPara
{
    int     iSize;
    int     iBrightness;   //亮度，0～100
    int     iContrast;     //对比度，0～100
    int     iSaturation;   //饱和度，0～100
    int     iHue;          //色度，0～100
} DecScreenBasicPara, *pDecScreenBasicPara;

#define SCREEN_OUTPUT_MODE_HDMI             0
#define SCREEN_OUTPUT_MODE_DVI              1
#define SCREEN_OUTPUT_MODE_ADAPTATION       2
typedef struct tagDecScreenOutputMode
{
    int     iSize;
    int     iMode;      //0:HDMI 1:DVI 2:自适应
} DecScreenOutputMode, *pDecScreenOutputMode;

typedef struct tagDecPicLevelRelation
{
    int     iSize;
    int     iPicCount;
    int     iLevelArray[DEC_MAX_WINDOW_TOTAL_NUM];     //层级关系:下标为画面号，层级关系从最底层开始为 0，往上依次加 1。
            //其中 iLevelArray[0]是第 0 画面的层级，iLevelArray[1]是第 1 画面的层级……iLevelArray[N]是第 N 画面的层级
} DecPicLevelRelation, *pDecPicLevelRelation;

typedef struct tagDecSyncLoopCtrl
{
    int     iSize;
    int     iLoopEnable;  //0：停止切换， 1：开始切换
} DecSyncLoopCtrl, *pDecSyncLoopCtrl;

typedef struct tagDecSyncLoopTime
```

```
{
    int     iSize;
    int     iHoldTime;      //停留时间：10-999
} DecSyncLoopTime, *pDecSyncLoopTime;

typedef struct tagDecEncodePara
{
    int     iSize;
    int     iWidth;         //视频宽
    int     iHeight;        //视频高
    int     iFrameRate;     //帧率，有效值：1～25，50，60
    int     iBitRate;       //码率，单位为 KBytes/s，如 1024kbps 的码率
} DecEncodePara, *pDecEncodePara;

typedef struct tagDecHttpPort
{
    int     iSize;
    int     iPort;          //HTTP 端口号
    int     iHttpsPort;     //HTTPS 端口号
    int     iRtsPort;       //端口号
    int     iSchedulePort;//排期端口号
} DecHttpPort, *pDecHttpPort;

typedef struct tagDecShowIcon
{
    int     iSize;
    int     iIconType;      //图标类型,1-655350-预留,1-画面底端红色报警闪烁
    int     iDisp;          //是否显示,0 不显示 1 显示
} DecShowIcon, *pDecShowIcon;

typedef struct tagDecAlarmEffectLight
{
    int     iSize;
    int     iEnable;            //特效使能：1--使能，0--关闭
    int     iDuration;          //特效持续时间   单位：秒

    int     iColor;             //颜色：32 位中取低 24 位表示颜色 rgb, 代表数字
方式 bgr。[((DWORD)(((BYTE)(r)|((WORD)((BYTE)(g))<<8))|(((DWORD)(BYTE)(b))<<16)))]

    int     iFlashtime;         //闪烁间隔,  单位：毫秒
} DecAlarmEffectLight, *pDecAlarmEffectLight;

#define MAX_EFFECTREGION_POINT_COUNT        8
typedef struct tagDecAlarmEffectRegion
```

```
{
    int     iSize;
    int     iEnable;                //特效使能：1--使能，0--关闭
    int     iDuration;                  //特效持续时间    单位：秒
    int     iWidth;                     //边框宽度
    int     iColor;                     //颜色：32 位中取低 24 位表示颜色 rgb, 代表数字方式 bgr。[((DWORD)(((BYTE)(r)|((WORD)((BYTE)(g))<<8))|(((DWORD)(BYTE)(b))<<16)))]

    int     iPointCount;            //区域顶点数，最大 8 个点
    POINT   tPoints[MAX_EFFECTREGION_POINT_COUNT];  //点坐标, 相对于视频分辨率的万分比坐标
} DecAlarmEffectRegion, *pDecAlarmEffectRegion;

typedef struct tagDecAlarmEffectText
{
    int     iSize;
    int     iEnable;                //特效使能：1--使能，0--关闭
    int     iDuration;                  //特效持续时间    单位：秒
    int     iFontSize;                  //字体大小，范围 1-5，代表 5 个级别的字体大小
    int     iColor;                     //颜色：32 位中取低 24 位表示颜色 rgb, 代表数字方式 bgr。[((DWORD)(((BYTE)(r)|((WORD)((BYTE)(g))<<8))|(((DWORD)(BYTE)(b))<<16)))]

    char cText[LEN_64];             //文本内容    最大 64 字节
    int     iTextAlign;                 //对齐方式：0--用户自定义, 1--左上角, 2--左下角, 3--右上角, 4--右下角, 5 居中
    POINT   tPoint;                     //字符叠加左上角坐标，相对于视频分辨率的万分比坐标
    int     iTextWidth;                 //文本区域宽度，实际分辨率

    int     iTextHeight;            //文本区域高度，实际分辨率
    int     iLeft;                      //左边距，iTextAlign 为 1, 2 时有效

    int     iTop;                       //上边距，iTextAlign 为 1, 3 时有效

    int     iRight;                     //右边距，iTextAlign 为 3, 4 时有效

    int     iBottom;                //下边距，iTextAlign 为 2, 4 时有效

} DecAlarmEffectText, *pDecAlarmEffectText;

#define DEC_MAX_PLAN_LOOP_ITEM_NUM MAX_PLAN_COUNT
typedef struct tagPlanLoopItem
{
    int     iIndex;         //切换序列号
```

```
    int         iPlanNo; //预案编号
    int         iHoldTime;    //停留时间  30-9999
}PlanLoopItem,*pPlanLoopItem;

typedef struct tagPlanLoopList
{
    int       iCount;          //切换列表数目
    PlanLoopItem tPlanLoopItem[DEC_MAX_PLAN_LOOP_ITEM_NUM];
}PlanLoopList,*pPlanLoopList;

typedef struct tagPlanLoop
{
    int       iEnable;   //使能
    int       iCount;       //数目
    int         iIndex;         //当前切换序列号
    int         iPlanNo; //当前预案编号
}PlanLoop,*pPlanLoop;

typedef struct tagPlanLoopItemDel
{
    int         iIndex;         //要删除的切换序列号
}PlanLoopItemDel,*pPlanLoopItemDel;

//设备网络工作模式
typedef    struct tagLanParamWorkMode
{
    int                 iWorkMode;              //工作模式 0，备份；1，多址
    int                 iMainLanNo;             //主网卡编号 0，Lan1；1，Lan2
}LanParamWorkMode, *pLanParamWorkMode;

typedef struct tagSipVideoChannelEx
{
    char       cChannelID[DEC_MAX_CHANNELID_LEN+1];
    int       iLevel;
    int       iPtzTime;
    int         iConnectMode;
    int         iTCPConnectType;
}SipVideoChannelEx, *pSipVideoChannelEx;

typedef struct tagDecChanMaxWin
{
    int       iMaxWin;
}DecChanMaxWin, *pDecChanMaxWin;
```

```
#define    DEC_MAX_LAYOUT_NUM 32
typedef struct tagDecChanLayoutList
{
    int        iLayoutNum;
    int         iLayout[DEC_MAX_LAYOUT_NUM];

}DecChanLayoutList, *pDecChanLayoutList;

#define    DEC_MAX_RESOLUTION_NUM 32
typedef struct tagDecChanResolutionList
{
    int        iResolutionNum;
    int         iResolution[DEC_MAX_RESOLUTION_NUM];
}DecChanResolutionList, *pDecChanResolutionList;

typedef struct tagPicSnap
{
    ServerItem uServerItem;
    int serType;                //0，Tiandy；1，Onvif；2，push 流；3，rtsp；4，解码器内部编
码通道；5，本地输入通道
                                //1000 开始是厂家对接特殊类型  1001，LG； 1002，松下；
}PicSnap,*pPicSnap;

typedef struct tagPicSnapResult
{
    int        iRet;            //0，成功；1，失败；
    int        iSnapID;         //抓拍 ID
}PicSnapResult, *pPicSnapResult;

typedef struct tagPicSnapID
{
    int        iSnapID;//抓拍 ID
}PicSnapID, *pPicSnapID;

typedef struct tagPicSnapFileName
{
    int        iSnapID;
    int        iState;                  //0，成功；1，失败；2，抓拍中
    char       cPicFileName[LEN_256];   //抓拍图片文件名
}PicSnapFileName, *pPicSnapFileName;

typedef struct tagScreenCtlProtocols
{
    int         iCount;                                          //协议个数
```

```
    char        cProtocol[MAX_PROTOCOL_NUM][MAX_PROTOCOL_NAME_LEN32+1];
    //各协议名称，最多支持 128 个协议，每条协议名称最长 32 字节；
}ScreenCtlProtocols, *pScreenCtlProtocols;

typedef struct tagScreenCtlProtocol
{
    int         iScreenNo;          //拼接屏的编号，指的是在 IE 上显示的屏幕编号

    int         iAddressNo;         //屏幕地址码 0x11~0xAA
    int         iComNo;             //串口编号
    char        cProtocol[MAX_PROTOCOL_NAME_LEN32+1];
}ScreenCtlProtocol, *pScreenCtlProtocol;

typedef struct tagScreenCtl
{
    int         iScreenNo;          //拼接屏的编号，指的是在 IE 上显示的屏幕编号

    int         iActionType;    //屏控命令：1：开关机；2：拼接屏选择；3：对比度；4：亮度；
5：色饱和度；6：清晰度
    int         iParam;             //命令具体参数：（1）开关机命令 0-关机，1-开机（2）拼接
屏选择命令时 0-AV，
                                    //1-S-VIDEO，2-VGA，3-DVI，4-HDMI（3）四度控制（对
比度、亮度、色饱和度、清晰度）命令 0-减少，命令 1-增加
}ScreenCtl, *pScreenCtl;

typedef struct tagScreenCtlResult
{
    int         iScreenNo;
    int         iActionType;
    int         iParam;
    int         iResult;            //0：预留，1：成功，2：失败
}ScreenCtlResult, *pScreenCtlResult;

typedef struct tagRebootTradeResult
{
    int           iResult;      //0：预留，1：成功，2：失败
}RebootTradeResult,*pRebootTradeResult;

#define DEC_MAX_VIDEOLIST_SIZE                  100

typedef struct tagUDiskVideoList
{
    int           iCount;
    char          cVideoName[DEC_MAX_VIDEOLIST_SIZE][LEN_64];//视频名称
```

```
}UDiskVideoList, *pUDiskVideoList;

typedef struct tagUDiskCurVideo
{
    int         iCurVideoState;          //视频状态
    char        cVideoName[LEN_64]; //视频名称
}UDiskCurVideo, *pUDiskCurVideo;


#define DEC_MAX_UDISK_CHANNEL_VIDEO          20
typedef struct tagUDiskSetChannelVideo
{
    int         iVideoNum; //视频个数
    char        cVideoName[DEC_MAX_UDISK_CHANNEL_VIDEO][LEN_64]; // 视 频 名 称

    int        iChannelNo;
    int        iPos;
}UDiskSetChannelVideo, *pUDiskSetChannelVideo;

#define DEC_MAX_UDISK_ALL_CHANNEL_VIDEO        10
typedef struct tagUDiskAllChannelVideo
{
    int      iCount;
    UDiskSetChannelVideo
tUDiskSetChannelVideo[DEC_MAX_UDISK_ALL_CHANNEL_VIDEO];
}UDiskAllChannelVideo, *pUDiskAllChannelVideo;


typedef struct tagVideoPlayMode
{
    int         iPlayMode; //播放模式类型视频播放类型：1：循环播放 2：顺序播放
}VideoPlayMode, *pVideoPlayMode;

typedef struct tagVideoPlayCmd
{
    int         iPlayCmd;   //播放命令 0：停止播放 1：开始播放
}VideoPlayCmd, *pVideoPlayCmd;

typedef struct tagVideoPlayCmdResult
{
    int                 iPlayCmd;
    int                 iResult;        //0：预留，1：成功，2：失败
}VideoPlayCmdResult, *pVideoPlayCmdResult;
```

```
typedef struct tagPlanLoopStatus
{
    int             iStatus;//0：开始切换 1：停止切换
}PlanLoopStatus, *pPlanLoopStatus;

typedef struct tagLoopParam
{
    int             iType;  //0,视频切换列表；1， 预案切换列表
}LoopParam, *pLoopParam;

typedef struct tagSoundCtrlResult
{
    int iAudioStatus;//0:AUDIO_STATUS_CLOSE  1:AUDIO_STATUS_OPEN  -1:
AUDIO_STATUS_FAILD
    int iFailReason;//0：预留；1：对讲已打开，与本地音频预览存在互斥
}SoundCtrlResult;

typedef struct tagTalkServResult
{
    int iTalkStatus;//0:TALK_STATUS_CLOSE 1:TALK_STATUS_OPEN
    -1:TALK_STATUS_FAILD
    int iFailReason;//0：预留；1：本地音频预览已打开，与对讲存在互斥
}TalkServResult;
```

## 5.3 错误码定义

```
#define   ERR_SUCCESS          0          //成功
#define   ERR_INVALIDID        -1         //指定的解码器不存在
#define   ERR_UNINIT           -2         //开发包尚未初始化/初始化失败
#define   ERR_INVALIDPARA      -3         //传入非法参数
#define   ERR_OUTMEMERY        -4         //内存分配错误
#define   ERR_NOACCOUNT        -5         //用户名错误
#define   ERR_NOPASS           -6         //密码错误
#define   ERR_INVALIDFILE          -7     //文件格式非法
#define   ERR_NOTLOGON             -8     //没有登录
#define   ERR_LOGING               -9     //已登录或正在登录
#define   ERR_MAXDEC               -10    //达到最大连接数
#define   ERR_UPGRADING            -11    //正在升级
#define   ERR_ISVIEW               1      //存在正在连接的 NVS
#define   ERR_FORBID               -12    //不允许操作
#define   ERR_RELOOPITEM           -13    //向循环列表中添加重复的连接
```

```
#define    ERR_RECONNECT                -14     //重复连接 NVS
#define    ERR_INVALIDPOS               -15     //位置越界
#define    ERR_NOCONNECTED              -16     //还没有建立连接
#define    ERR_POS_X_Y                  -17     //坐标非法
#define    ERR_NULL_PTR                 -18     //空指针
#define    ERR_UNDEFINED_BEHAVIOR       -19     //不支持的行为
#define    ERR_INVALID_BUFF_SIZE        -20     //不合法的缓冲区大小
#define    ERR_INVALID_ARRAY_INDEX      -21     //不合法的数组下标
#define    ERROR_INVALID_PARAM          -22     //参数不合法
#define    ERR_REPEAT_SET               -23     //重复设置
#define    ERR_BUILD_PROTOCOL           -24     //封装协议失败
#define    ERR_SEND_PROTOCOL            -25     //发送协议失败
#define    ERR_NO_IDLE_ITEM             -26     //没有空闲的条目
#define    ERR_INVALID_FILEHEADER       -27     //非法的文件头
#define ERR_LIGHTMODE_NOTSUPPORT        -28     //不支持轻量级模式
#define ERR_OUT_OF_MEMERY               -29     //内存不足
#define    ERR_SYSTEM                   -100    //系统错误
```