播放库使用说明书

Version:2.0.3.3

版本更新说明

v2.0.3.3　2015-09-29
增加抓 Y UV420 数据函数接口　HI_PLAYER_SnapYUVData

v2.0.3.2　2013-07-12
修改 1920*1080 抓拍崩溃问题

v2.0.3.1　2013-06-20
增大库文件中 Buffer 的大小

v2.0.2.9　2013-04-08
支持 1080P 文件播放

v2.0.2.6　2012-12-10
添加 D3D 显示支持，默认显示为 DDraw
添加设置显示获取模式接口 HI_PLAYER_GetDisplayMode 和 HI_PLAYER_SetDisplayMode
添加 D3D 模式下图像叠加开关 HI_PLAYER_SetDrawCallBack

v2.0.2.2　2011-07-22
取消录像接口 **HI_PLAYER_StartRecord** 和 **HI_PLAYER_StopRecord**，将录像接口移到网络库中，NetLib.dll 接口 **HI_NET_DEV_StartRecord** 和 **HI_NET_DEV_StopRecord** 为新录像接口。

v2.0.2.1　2011-07-02
添加录像类型 AVI，接口 HI_PLAYER_StartRecord 中第三参数指定为：PLAYER_FILE_AVI，HI_PLAYER_OpenFile 支持 AVI 文件播放。

v2.0.2.0　2011-05-14
添加显示区域电子放大接口：HI_PLAYER_DisplayAll

v2.0.1.8　2011-03-21
添加接口：
HI_PLAYER_SetStreamOpenMode 设置播放模式
HI_PLAYER_GetStreamOpenMode 获取播放模式
HI_PLAYER_OpenStream 设置音视频属性
HI_PLAYER_InputData 输入复合流接口
HI_PLAYER_ResetSourceBuffer 清空缓存
HI_PLAYER_GetBufferValue 获取缓存使用率
HI_PLAYER_GetCurrentPts 获取当前播放的时间戳
完善功能接口：
HI_PLAYER_Fast 快放，最多 X4
HI_PLAYER_Slow 慢放，最多 X4

v2.0.1.7　2011-02-21
修改 YUV 回调函数 HI_PLAYER_SetDecCallBack 接口，调用回调函数视频只回调出 YUV 数据，而不显示在窗口中；

v2.0.1.6　2011-01-20
修改抓拍接口控制接口 HI_PLAYER_SnapBMP 和 HI_PLAYER_SnapJPEG，修改后直接输入文件的路径+名称就可以生成图片。

v2.0.1.5　2010-12-1
添加解码控制接口 HI_PLAYER_PauseDecode 和 HI_PLAYER_ResumeDecode

一、功能说明

播放库主要功能

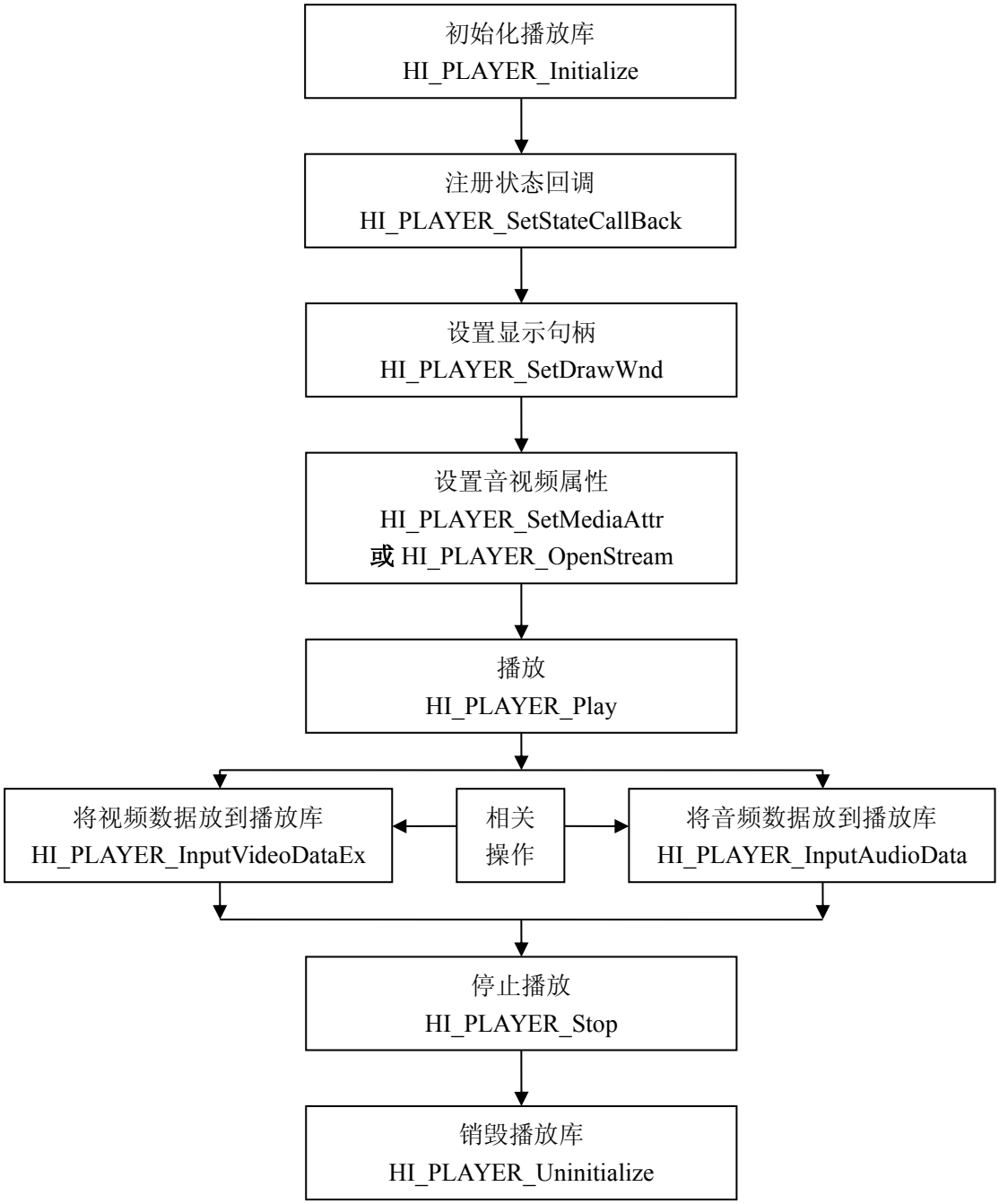
主要用于实时码流预览，录像文件的回放，播放控制如：暂停、单帧前进，获取码流信息，抓拍功能。

播放库文件说明

播放库	HsPlayer.h	头文件
	HIPlayer.lib	LIB 库文件
	HIPlayer.dll	DLL 库文件
	avcodec-54.dll	H.264 解码 dll
	avutil-51.dll	H.264 解码 dll
公用文件	hi_dataType.h	头文件

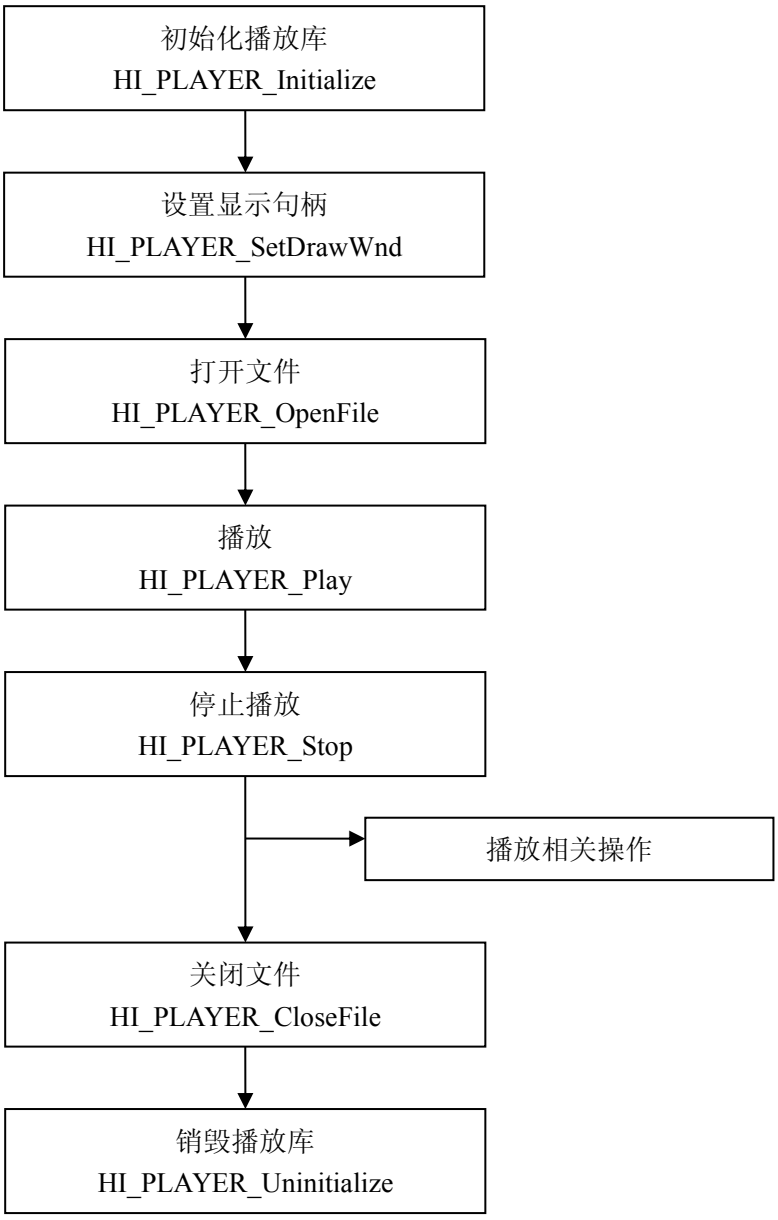
二、编程导引

实时流、文件流模式



相关操作包含有录像、抓拍、解码操作、音量控制等。

文件模式



播放相关操作为可选操作，有：播放、暂停、停止、抓拍、快放、慢放、单帧、获取播放时间、设置音量、设置播放位置等；详细接口请查阅文件播放操作部分相关函数。

三、数据类型定义说明

```
typedef unsigned char           HI_U8;
typedef unsigned char           HI_UCHAR;
typedef unsigned short          HI_U16;
typedef unsigned int            HI_U32;

typedef signed char             HI_S8;
typedef short                   HI_S16;
typedef int                     HI_S32;

#ifndef _M_IX86
typedef unsigned long long      HI_U64;
typedef long long               HI_S64;
#else
typedef __int64                 HI_U64;
typedef __int64                 HI_S64;
#endif

typedef char                    HI_CHAR;
typedef char*                   HI_PCHAR;

typedef float                   HI_FLOAT;
typedef double                  HI_DOUBLE;
typedef void                    HI_VOID;

typedef unsigned long           HI_SIZE_T;
typedef unsigned long           HI_LENGTH_T;


typedef enum {
    HI_FALSE        = 0,
    HI_TRUE         = 1,
} HI_BOOL;

#ifndef NULL
#define NULL                    0L
#endif
#define HI_NULL                 0L
#define HI_NULL_PTR             0L

#define HI_SUCCESS              0
#define HI_FAILURE              (-1)
```

四、函数说明

4.1 初始化销毁播放库

HI_PLAYER_Initialize

初始化播放库，获取播放库操作指针 pHandle

```
HRESULT    HI_PLAYER_Initialize (
        PLAYHANDLE*          pHandle
);
```

Parameters

pHandle

[IN] 播放库句柄，提供之后的操作使用

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_PLAYER_Uninitialize

初始化播放库

```
HRESULT    HI_PLAYER_Uninitialize (
        PLAYHANDLE           hHandle
);
```

Parameters

hHandle

[IN] 销毁播放库

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

4.2 数据回调

HI_PLAYER_SetDecCallBack

注册解码数据回调，用于实时预览

```
HRESULT    HI_PLAYER_SetDecCallBack (
        PLAYHANDLE                hHandle,
        HI_PLAYER_DecCallBack     CallBack,
        HI_VOID *                 pPara
);
```

Parameters

hHandle

[IN] 播放库句柄

CallBack

[IN] 解码数据回调

pPara

[IN] 用户数据

Callback Function

```
typedef   HRESULT    (*HI_PLAYER_DecCallBack)(
          PLAYHANDLE                    hHandle,
          const PLAYER_FRAME_INFO_S *pFrameInfo,
          HI_VOID*                      pDecParam
          );
```

Callback Function Parameters

hHandle

操作句柄

pFrameInfo

帧类型

```
typedef struct hi PLAYER_FRAME_INFO_S
{
    HI_U8* pY;          //解码后视频数据 Y  分量
    HI_U8* pU;          //解码后视频数据 U 分量
    HI_U8* pV;          //解码后视频数据 V 分量
    long nWidth;        //视频宽
    long nHeight;       //视频高
    long nType;         //data type:YUV420
    long nYPich;
    long nUVPich;
    HI_U64 u64Pts;
}
PLAYER_FRAME_INFO_S;
```

pDecParam

解码后视频数据

pPara

用户数据

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_PLAYER_SetStateCallBack

注册播放状态回调，用于实时预览和文件回放

```
HRESULT    HI_PLAYER_SetStateCallBack (
    PLAYHANDLE                  hHandle,
    HI_PLAYER_StateCallBack CallBack,
    HI_VOID *                   pPara
```

```
);
```

Parameters

hHandle

[IN] 播放库句柄

CallBack

[IN] 播放状态回调

pPara

[IN] 用户数据

Callback Function

```
typedef   HRESULT   (*HI_PLAYER_StateCallBack)(
          PLAYHANDLE              hHandle,
          PLAYER_STATE_ID_E       eStateID,
          HI_U32                  u32State,
          HI_VOID*                pPara
          );
```

Callback Function Parameters

hHandle

操作句柄

eStateID

状态类型

```
typedef enum hiPLAYER_STATE_ID_E
{
    PLAYER_STATE_PLAY = 0,      //播放
    PLAYER_STATE_REC,           //录像
    PLAYER_STATE_TALK,          //对讲
    PLAYER_STATE_BUTT
}PLAYER_STATE_ID_E;
```

u32State

对应类型的状态

pPara

用户数据

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

4.3 设置播放属性

HI_PLAYER_SetDrawWnd

设置显示句柄，用于实时预览和文件回放

```
HRESULT   HI_PLAYER_SetDrawWnd (
    PLAYHANDLE          hHandle,
```

```
    HI_VOID*            hWnd
);
```

Parameters

hHandle

[IN] 播放库句柄

hWnd

[IN] 显示句柄

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remarks

如果在播放时要将播放窗口换到另一个播放窗口，可以直接用改接口，只要将要显示的窗口句柄与相应的操作句柄关联起来即可。如果 pWnd 为空时，DDRAW 将销毁，即将不对视频进行显示；只有再次设置 pWnd 为非空时，才能再次显示。

HI_PLAYER_SetStreamOpenMode

设置播放流模式，文件流（HI_STREAM_FILE）和实时流（HI_STREAM_REALTIME）

```
HRESULT  HI_PLAYER_SetStreamOpenMode (
    PLAYHANDLE          hHandle,
    HI_U32              u32Mode
);
```

Parameters

hHandle

[IN] 播放库句柄

u32Mode

[IN] 流模式，文件流-HI_STREAM_FILE、实时流-HI_STREAM_REALTIME
默认为 HI_STREAM_REALTIME 模式。

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remarks

HI_STREAM_REALTIME 适用于播放网络实时流数据，输入后解码器会立即解码；HI_STREAM_FILE 用于文件流模式，适合用户把文件以流的方式，当 HI_PLAYER_InputData 返回 HI_FAILURE 时，表示缓存使用率已经使用满，要求用户要等待一段时候后重新输入。默认情况下不调用则视为 HI_STREAM_REALTIME 模式。

*在流开始播放前调用。

HI_PLAYER_GetStreamOpenMode

获取播放流模式

```
HI_U32    HI_PLAYER_GetStreamOpenMode (
        PLAYHANDLE          hHandle,
);
```

Parameters

hHandle

[IN] 播放库句柄

Return Values

返回值是 HI_STREAM_FILE--文件流、HI_STREAM_REALTIME--实时流。

HI_PLAYER_OpenStream

设置流播放的音视频属性，用于流模式。

```
HRESULT    HI_PLAYER_OpenStream (
        PLAYHANDLE          hHandle,
        HI_U8               *pFileHeadBuf,
        HI_U32              u32Size
);
```

Parameters

hHandle

[IN] 播放库句柄

pFileHeadBuf

[IN] 头文件数据。结构体 HI_S_SysHeader

u32Size

[IN] 设置头文件长度，sizeof(HI_S_SysHeader)

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remarks

HI_PLAYER_OpenStream 和 HI_PLAYER_SetMediaAttr 的音视频设置功能相同，前者只需将 HI_S_SysHeader 头文件结构体输入函数，就实现对音视频流属性的初始化；后者需自己配置音视频参数。具体用法如下：

HI_S_SysHeader fileHeader;

… //获取或设置头文件

HI_PLAYER_OpenStream(hHande, (HI_U8*)&fileHeader, sizeof(HI_S_SysHeader));

相似接口可查阅 HI_PLAYER_SetMediaAttr。

*在流开始播放前调用。

HI_PLAYER_SetMediaAttr

设置播放音视频属性（用于实时预览初始化，文件回放无此项）

```
HRESULT    HI_PLAYER_SetMediaAttr (
```

```
    PLAYHANDLE          hHandle,
    PLAYER_MEDIAATTR_TYPE_E     eAttrType,
    HI_VOID*            pAttr
);
```

Parameters

hHandle

[IN] 播放库句柄

eAttrType

[IN] 设置的类型

```
typedef enum hiPLAYER_MEDIAATTR_TYPE_E
{
    PLAYER_ATTR_VIDEO_STREAM = 0,       //视频流
    PLAYER_ATTR_VIDEO_OUTPUT = 1,       //视频输出
    PLAYER_ATTR_AUDIO_STREAM = 2,       //音频流
    PLAYER_ATTR_AUDIO_ENCODE = 3,       //音频压缩
    PLAYER_BUTT_ATTR_TYPE
}PLAYER_MEDIAATTR_TYPE_E;
```

pAttr

[IN] 属性参数

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remarks

HI_PLAYER_SetMediaAttr 函数用于设置播放器属性，比如视频流属性、音频属性、音频压缩属性等，该函数使用在 HI_PLAYER_Play 之前，其用法如下：

1、 设置视频属性：

```
HI_PLAYER_SetMediaAttr ( pHsPlayerHandle,                   //句柄
                         PLAYER_ATTR_VIDEO_STREAM,  //设置视频标志
                         &stVStreamAttr );                  //视频属性
```

stVStreamAttr 定义为 PLAYER_ATTR_VIDEO_STREAM_S

```
typedef struct hiPLAYER_ATTR_VIDEO_STREAM_S
{
    PLAYER_VIDEO_FORMAT_E eVEncode;     //视频压缩格式，如 H264 等
    long        lHeight;                //视频高
    long        lWidth;                 //视频宽
    long        lBitRate;               //最大码率
    long        lFrameRate;             //帧率
} PLAYER_ATTR_VIDEO_STREAM_S;
```

注：lBitRate 为视频单帧醉倒的码率，如 2*1024*1024，即 2M 码率，码率一定要大于实际的码率。如 1.5M 的码率，我们设为 2M；lFrameRate 帧率一般设置为 25 帧。

用法：

```
m_struAVAttr.struVAttr.lWidth     = s32PicWidth;
struAVAttr.struVAttr.lHeight      = s32PicHeight;
struAVAttr.struVAttr.eVEncode     = PLAYERSDK_VENC_FORMAT_H264;
struAVAttr.struVAttr.lBitRate     = 20000000;
struAVAttr.struVAttr.lFrameRate   = 25;
```

2、 设置音频属性：

```
HI_PLAYER_SetMediaAttr ( pHsPlayerHandle,                  //句柄
                         PLAYER_ATTR_AUDIO_STREAM,  //设置音频标志
                         &stAStreamAttr);                  //音频属性
```

stAStreamAttr 定义为 PLAYER_ATTR_AUDIO_S

```
typedef struct hiPLAYER_ATTR_AUDIO_S
{
    PLAYER_AUDIO_FORMAT_E eAEncode;     //音频格式
    long          lSamplesPerSec;       //audio's samples per second
    long          lBitsPerSample;       //bits per sample
    long          lBitRate;             //audio's bit rate
    long          lBlockAlign;          //对齐
    long          lChannels;            //通道，单通道双通道
    long          lFrameFlag;           //帧标志
    long          length;               //音频大小
    void          *pReserved;
} PLAYER_ATTR_AUDIO_S;
```

注：lBitsPerSample = lBitRate /lSamplesPerSec。如果音频格式为 G711A，lBitRate 为 16000；如果为 AMR、G711 则 lBitRate 为 64000。

用法：

G726 音频属性：

```
struAVAttr.struAAttr.eAEncode = PLAYERSDK_AUDIO_CODEC_FORMAT_G726;
struAVAttr.struAAttr.lBitRate          = 16000;
struAVAttr.struAAttr.lSamplesPerSec    = 8000;
struAVAttr.struAAttr.lBitsPerSample    = 2;
struAVAttr.struAAttr.lBlockAlign       = 1;
struAVAttr.struAAttr.lChannels         = 1;
struAVAttr.struAAttr.length            = 0;
struAVAttr.struAAttr.lFrameFlag        = 0;
struAVAttr.struAAttr.pReserved         = NULL;
```

G711 音频属性：

```
struAVAttr.struAAttr.eAEncode = PLAYERSDK_AUDIO_CODEC_FORMAT_G711A;
struAVAttr.struAAttr.lBitRate          = 64000;
struAVAttr.struAAttr.lSamplesPerSec    = 8000;
struAVAttr.struAAttr.lBitsPerSample    = 8;
struAVAttr.struAAttr.lBlockAlign       = 1;
struAVAttr.struAAttr.lChannels         = 1;
struAVAttr.struAAttr.length            = 0;
```

```
struAVAttr.struAAttr.lFrameFlag          = 0;
struAVAttr.struAAttr.pReserved           = NULL;
```

3、 设置对讲音频属性：

```
HI_PLAYER_SetMediaAttr ( pHsPlayer,
                         PLAYER_ATTR_AUDIO_ENCODE,
                         &stAStreamAttr);
```

注：stAStreamAttr 属性同音频属性。

*在流开始播放前调用。

HI_PLAYER_GetMediaAttr

获取播放音视频属性，用于实时预览和文件回放

```
HRESULT   HI_PLAYER_GetMediaAttr (
    PLAYHANDLE          hHandle,
    PLAYER_MEDIAATTR_TYPE_E     eAttrType,
    HI_VOID*            pAttr
);
```

Parameters

hHandle

[IN] 播放库句柄

eAttrType

[IN] 设置的类型

```
typedef enum hiPLAYER_MEDIAATTR_TYPE_E
{
    PLAYER_ATTR_VIDEO_STREAM = 0,        //视频流
    PLAYER_ATTR_VIDEO_OUTPUT = 1,        //视频输出
    PLAYER_ATTR_AUDIO_STREAM = 2,        //音频流
    PLAYER_ATTR_AUDIO_ENCODE = 3,        //音频压缩
    PLAYER_BUTT_ATTR_TYPE
}PLAYER_MEDIAATTR_TYPE_E;
```

pAttr

[OUT] 属性参数

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

4.4 播放操作

HI_PLAYER_Play

播放，用于实时预览和文件回放

```
HRESULT   HI_PLAYER_Play (
    PLAYHANDLE          hHandle
);
```

Parameters

hHandle

[IN] 播放库句柄

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remarks

如果是流播放，开始该函数时必须先用 HI_PLAYER_SetMediaAttr 对播放流进行初始化，初始化函数的使用请查阅 4.3 章 HI_PLAYER_SetMediaAttr 函数。

HI_PLAYER_Stop

停止播放，用于实时预览和文件回放

```
HRESULT    HI_PLAYER_Stop (
        PLAYHANDLE          hHandle
);
```

Parameters

hHandle

[IN] 播放库句柄

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

4.5 音视频流操作

HI_PLAYER_InputData

以符合流的方式输入到播放缓冲中，用于流播放模式

```
HRESULT    HI_PLAYER_InputData (
        PLAYHANDLE          hHandle,
        HI_U8 *             pBuf,
        HI_S32              s32Size,
);
```

Parameters

hHandle

[IN] 播放库句柄

pBuf

[IN] 流数据缓冲地址

s32Size

[IN] 数据大小，最大为 1280*768*2

Return Values

成功返回 HI_SUCCESS，如果返回 HI_FAILURE，表示缓存已经到达上限，需要等待一会才能继续输入。

Remarks

需要在流开始后输入才有效。HI_PLAYER_InputData 用于复合流的输入，不区分音视频，包含数据头；而 HI_PLAYER_InputVideoDataEx 和 HI_PLAYER_InputAudioData 是音视频分开的方式输入，pBuf 不包含帧头。

相似接口请查阅 HI_PLAYER_InputVideoDataEx、HI_PLAYER_InputAudioData。

HI_PLAYER_InputVideoData

将视频流放到播放库进行处理，（旧版本使用）

```
HRESULT    HI_PLAYER_InputVideoData (
        PLAYHANDLE          hHandle,
        HI_U8 *             pBuf,
        HI_S32              s32Size,
        HI_U64              u64TimeStamp
);
```

Parameters

hHandle

[IN] 播放库句柄

pBuf

[IN] 视频流数据（不包含帧头）

s32Size

[IN] 数据大小

u64TimeStamp

[IN] 时间戳

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_PLAYER_InputVideoDataEx

将视频流放到播放库进行处理，（新版本使用）

```
HRESULT    HI_PLAYER_InputVideoDataEx (
        PLAYHANDLE          hHandle,
        HI_U8 *             pBuf,
        HI_S32              s32Size,
        HI_S32              s32KeyFrame,
        HI_U64              u64TimeStamp
);
```

Parameters

hHandle

[IN] 播放库句柄

pBuf

[IN] 视频流数据（不包含帧头）

s32Size

[IN] 数据大小

s32KeyFrame

[IN] 视频关键帧

u64TimeStamp

[IN] 时间戳

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remarks

HI_PLAYER_InputVideoData 使用在回调数据不是完整帧的情况，由播放库对数据进行组帧；在新的网络库中回调的的视频流都是已组好的完整帧，只需要调用 HI_PLAYER_InputVideoDataEx 函数即可，播放库就不再进行组帧。

网络库回调出来的整帧数据时带帧头的，pBuf 数据不包含帧头，所以数据要去掉帧头：

```
HI_PLAYER_InputVideoDataEx( pHsPlayer,
                            pu8Buffer+sizeof(HI_S_AVFrame),
                            pstruVideo->u32AVFrameLen,
                            s32KeyFrame,
                            pstruVideo->u32AVFramePTS);
```

音频输入也要去掉帧头。

HI_PLAYER_InputAudioData

将音频流放到播放库进行处理

```
HRESULT    HI_PLAYER_InputAudioData (
    PLAYHANDLE          hHandle,
    HI_U8 *             pBuf,
    HI_S32              s32Size,
    HI_U64              u64TimeStamp
);
```

Parameters

hHandle

[IN] 播放库句柄

pBuf

[IN] 视频流数据（不包含帧头）

s32Size

[IN] 数据大小

u64TimeStamp

[IN] 时间戳

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

4.6 对讲

HI_PLAYER_StartTalk

获取 PC 端音频压缩数据

```
HRESULT    HI_PLAYER_StartTalk (
        PLAYHANDLE              hHandle,
        HI_PLAYER_TalkCallBack CallBack,
        HI_VOID*                pPara
);
```

Parameters

hHandle

[IN] 播放库句柄

CallBack

[IN] 对讲数据回调

pPara

[IN] 用户数据

Callback Function

```
typedef    HRESULT    (*HI_PLAYER_TalkCallBack)(
                PLAYHANDLE      hHandle,
                HI_U8*          pBuf,
                HI_S32          s32Size,
                HI_U64          u64TimeStamp,
                HI_VOID*        pPara
                );
```

Callback Function Parameters

hHandle

操作句柄

pBuf

音频数据

s32Size

音频数据大小

u64TimeStamp

时间戳

pPara

用户数据

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remarks

获取音频压缩数据的格式：

```
HI_PLAYER_SetMediaAttr ( pHsPlayer,
                         PLAYER_ATTR_AUDIO_ENCODE,
                         &stAStreamAttr);
```

通过调用网络库的 HI_NET_DEV_SendVoiceData 将对讲的数据发送给摄像机，具体请参阅 HI_NET_DEV_SendVoiceData 函数使用说明。

HI_PLAYER_StopTalk

关闭 PC 获取的音频数据

```
HRESULT    HI_PLAYER_StopTalk (
    PLAYHANDLE          hHandle
);
```

Parameters

hHandle

[IN] 播放库句柄

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

4.7 抓拍

HI_PLAYER_SnapBMP

保存为 BMP 数据，用于实时预览和文件回放

```
HRESULT    HI_PLAYER_SnapBMP (
    PLAYHANDLE          hHandle,
    HI_CHAR *           pSnapPath
);
```

Parameters

hHandle

[IN] 播放库句柄

pSnapPath

[IN] 文件路径+文件名称

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_PLAYER_SnapJPEG

保存为 JPG 数据，具体用法同 HI_PLAYER_SnapBMP，用于实时预览和文件回放

```
HRESULT    HI_PLAYER_SnapJPEG (
```

```
        PLAYHANDLE          hHandle,
        HI_CHAR *           pSnapPath,
        HI_S32              s32QValue
    );
```

Parameters

hHandle

[IN] 播放库句柄

pSnapPath

[IN] 文件路径+文件名称

s32Qvalue

[IN] JPG 图像质量 100 为最好

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_PLAYER_SnapYUVData

抓解码后的 YUV 数据

```
HRESULT HI_PLAYER_SnapYUVData (
        PLAYHANDLE lHandle,
        PLAYER_YUV_INFO_S * pYUVInfo,
        HI_S32 nYUVType,
        HI_S32 *pNeedSize
        );
```

Parameters

lHandle

[IN] 操作句柄

pYUVInfo

[IN][OUT] 存放 YUV 数据。

nYUVType

[IN] 指定抓取的 YUV 类型，扩展用，恒为 0,目前只支持 YUV420 格式。

pNeedSize

[OUT] 存放 YUV 所需内存的大小，不能为空。

Return Values

成功返回 HI_SUCCESS，失败返回 HI_FAILURE。

如果 pYUVInfo 为空，函数返回 HI_FAILURE,pNeedSize 返回 YUV 数据所需内存的大小。

4.8 声音操作

HI_PLAYER_SetMute

设置静音/监听，用于实时预览和文件回放

```
HRESULT   HI_PLAYER_SetMute (
    PLAYHANDLE                  hHandle,
    PLAYER_AUDIO_DIRECT_E       eAudioDirect,
    HI_BOOL                     bMute
);
```

Parameters

hHandle

[IN] 播放库句柄

eAudioDirect

[IN] 输入输出

```
typedef enum hiPLAYER_AUDIO_DIRECT_E
{
    PLAYER_AUDIO_OUT = 0,       //输出音频
    PLAYER_AUDIO_IN = 1         //输入音频，如麦克
}PLAYER_AUDIO_DIRECT_E;
```

bMute

[IN] 开关，TRUE-静音，FALSE-监听

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_PLAYER_GetMute

获取静音/监听状态，用于实时预览和文件回放

```
HRESULT   HI_PLAYER_GetMute (
    PLAYHANDLE                  hHandle,
    PLAYER_AUDIO_DIRECT_E       eAudioDirect,
    HI_BOOL                     *pMute
);
```

Parameters

hHandle

[IN] 播放库句柄

eAudioDirect

[IN] 输入输出

```
typedef enum hiPLAYER_AUDIO_DIRECT_E
{
    PLAYER_AUDIO_OUT = 0,       //输出音频
    PLAYER_AUDIO_IN = 1         //输入音频，如麦克
}PLAYER_AUDIO_DIRECT_E;
```

pMute

[OUT] 开关，TRUE-静音，FALSE-监听

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_PLAYER_SetVolume

设置声音大小，用于实时预览和文件回放

```
HRESULT    HI_PLAYER_SetVolume (
        PLAYHANDLE                  hHandle,
        PLAYER_AUDIO_DIRECT_E       eAudioDirect,
        HI_S32                      s32LVolume,
        HI_S32                      s32RVolume
);
```

Parameters

hHandle

[IN] 播放库句柄

eAudioDirect

[IN] 输入输出

```
typedef enum hiPLAYER_AUDIO_DIRECT_E
{
    PLAYER_AUDIO_OUT = 0,       //输出音频
    PLAYER_AUDIO_IN = 1         //输入音频，如麦克
}PLAYER_AUDIO_DIRECT_E;
```

s32LVolume

[IN] 左声道声音大小，范围[0，0xFFFF]

s32RVolume

[IN] 右声道声音大小，范围[0，0xFFFF]

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_PLAYER_GetVolume

获取声音大小，用于实时预览和文件回放

```
HRESULT    HI_PLAYER_GetVolume (
        PLAYHANDLE                  hHandle,
        PLAYER_AUDIO_DIRECT_E       eAudioDirect,
        HI_S32*                     pLVolume,
        HI_S32*                     pRVolume
);
```

Parameters

hHandle

[IN] 播放库句柄

eAudioDirect

[IN] 输入输出

```
typedef enum hiPLAYER_AUDIO_DIRECT_E
{
    PLAYER_AUDIO_OUT = 0,       //输出音频
    PLAYER_AUDIO_IN = 1         //输入音频，如麦克
}PLAYER_AUDIO_DIRECT_E;
```

pLVolume

[OUT] 左声道声音大小，范围[0，0xFFFF]

pRVolume

[OUT] 右声道声音大小，范围[0，0xFFFF]

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

4.9 文件播放操作

HI_PLAYER_OpenFile

打开本地文件，文件播放操作

```
HRESULT   HI_PLAYER_OpenFile (
    PLAYHANDLE          hHandle,
    HI_U8 *             pFileName
);
```

Parameters

hHandle

[IN] 播放库句柄

pFileName

[IN] 文件路径

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remarks

打开文件后，用 HI_PLAYER_Play，HI_PLAYER_Stop 进行播放、停止等操作。

HI_PLAYER_CloseFile

关闭本地文件播放，文件播放操作

```
HRESULT   HI_PLAYER_CloseFile (
    PLAYHANDLE          hHandle
);
```

Parameters

hHandle

[IN] 播放库句柄

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_PLAYER_Pause

暂停播放，该函数适用于文件播放或文件流模式

```
HRESULT    HI_PLAYER_Pause (
        PLAYHANDLE          hHandle
);
```

Parameters

hHandle

[IN] 播放库句柄

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_PLAYER_SetPlayPos

设置文件播放的位置，该函数使用在播放文件时

```
HRESULT    HI_PLAYER_SetPlayPos (
        PLAYHANDLE          hHandle,
        HI_S32              s32Pos
);
```

Parameters

hHandle

[IN] 播放库句柄

s32Pos

[IN] 播放位置，范围[0，100]

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_PLAYER_GetPlayPos

获取文件播放的位置，该函数使用在播放文件时

```
HRESULT    HI_PLAYER_GetPlayPos (
        PLAYHANDLE          hHandle,
        HI_S32*             pPos
```

```
);
```

Parameters

hHandle

[IN] 播放库句柄

pPos

[OUT] 播放位置，范围[0，100]

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_PLAYER_SetRate

设置播放速度，该函数适用于文件播放或文件流模式，速度不为常速音频不解码

```
HRESULT    HI_PLAYER_SetRate (
    PLAYHANDLE          hHandle,
    HI_S32              rate
);
```

Parameters

hHandle

[IN] 播放库句柄

rate

[IN] 播放速度，范围[-3，8]，大于 7 为按 I 帧播放

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_PLAYER_GetRate

设置播放速度，该函数适用于文件播放或文件流模式，速度不为常速音频不解码

```
HRESULT    HI_PLAYER_GetRate (
    PLAYHANDLE          hHandle,
    HI_S32*             rate
);
```

Parameters

hHandle

[IN] 播放库句柄

rate

[OUT] 播放速度

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_PLAYER_Fast

设置播放速度，该函数适用于文件播放或文件流模式，速度不为常速音频不解码

```
HRESULT    HI_PLAYER_Fast (
        PLAYHANDLE          hHandle,
);
```

Parameters

hHandle

[IN] 播放库句柄

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remaks

每调用一次速度将快一倍，最多可以调用 4 次，超过 4 次将恢复到正常播放速度，直接恢复正常速度可以使用 HI_PLAYER_Play()。如果直接调整速度值可以调用相似接口 HI_PLAYER_SetRate()；

HI_PLAYER_Slow

设置播放速度，该函数适用于文件播放或文件流模式，速度不为常速音频不解码

```
HRESULT    HI_PLAYER_Slow (
        PLAYHANDLE          hHandle,
);
```

Parameters

hHandle

[IN] 播放库句柄

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remaks

每调用一次速度将慢一倍，最多可以调用 4 次，超过 4 次将恢复到正常播放速度，直接恢复正常速度可以使用 HI_PLAYER_Play()。如果直接调整速度值可以调用相似接口 HI_PLAYER_SetRate()；

HI_PLAYER_OneByOne

单帧播放，该函数适用于文件播放或文件流模式，单帧播放要求在暂停情况下实现

```
HRESULT    HI_PLAYER_OneByOne (
        PLAYHANDLE          hHandle
);
```

Parameters

hHandle

[IN] 播放库句柄

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_PLAYER_GetFileAttr

获取文件播放属性，该函数使用在播放文件时

```
HRESULT    HI_PLAYER_GetFileAttr (
    PLAYHANDLE              hHandle,
    PLAYER_ATTR_FILE_S *    pFileAttr
);
```

Parameters

hHandle

[IN] 播放库句柄

pFileAttr

[OUT] 播放文件属性

```
typedef struct hiPLAYER_ATTR_FILE_S
{
    unsigned char u8FormatName[256];        //文件格式
    unsigned char u8FileName[512];          //文件名
    PLAYER_DURATION_S struDuration;         //文件时间信息
    unsigned int u32StreamNum;              //数据类型（0 视频、1 音频、2 混合）
    int s32VideoIndex;                      //video stream's index
    int s32AudioIndex;                      //audio stream's index
    HI_BOOL bSeekEnable;                    //是否支持调节进度
}
PLAYER_ATTR_FILE_S;
```

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_PLAYER_GetState

获取播放、录像、对讲状态

```
HRESULT    HI_PLAYER_GetState (
    PLAYHANDLE              hHandle,
    PLAYER_STATE_ID_E       eStateID,
    HI_U32*                 pState
);
```

Parameters

hHandle

[IN] 播放库句柄

eStateID

[IN] 状态类型

```
typedef enum hiPLAYER_STATE_ID_E
{
    PLAYER_STATE_PLAY = 0,      //播放
    PLAYER_STATE_REC,           //录像
    PLAYER_STATE_TALK,          //对讲
    PLAYER_STATE_BUTT
}PLAYER_STATE_ID_E;
```

pState

[OUT] 状态值

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_PLAYER_GetPlayTime

单帧播放，该函数使用在播放文件时，单帧播放要求在暂停情况下实现

```
HRESULT   HI_PLAYER_GetPlayTime (
    PLAYHANDLE          hHandle,
    HI_S32 *            pTime
);
```

Parameters

hHandle

[IN] 播放库句柄

pTime

[OUT] 当前文件播放时间

Return Values

返回当前文件总时间，如果没有播放，返回 0。

4.10 解码控制

HI_PLAYER_PauseDecode

停止解码，用于实时预览

```
HRESULT   HI_PLAYER_PauseDecode (
    PLAYHANDLE          hHandle,
);
```

Parameters

hHandle

[IN] 播放库句柄

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_PLAYER_ResumeDecode

恢复解码，用于实时预览

```
HRESULT    HI_PLAYER_ResumeDecode (
        PLAYHANDLE          hHandle,
);
```

Parameters

hHandle

[IN] 播放库句柄

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

4.11 流播放缓冲操作

HI_PLAYER_GetBufferValue

获取缓存使用率，范围：0-10000

```
HI_S32    HI_PLAYER_GetBufferValue (
        PLAYHANDLE          hHandle,
);
```

Parameters

hHandle

[IN] 播放库句柄

Return Values

返回缓存使用率，0-表示未使用，6000-表示使用了 60%。在文件流模式下，最大值为 9000，在实时流模式下最大值为 6000。

HI_PLAYER_ResetSourceBuffer

清空缓存区，需在流播放模式下使用

```
HRESULT    HI_PLAYER_ResetSourceBuffer (
        PLAYHANDLE          hHandle,
);
```

Parameters

hHandle

[IN] 播放库句柄

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

4.12 其他相关操作

HI_PLAYER_SetPlayerBufNumber

设置网络延时和播放流畅度可以通过此接口来进行调节，用于实时预览

```
HRESULT    HI_PLAYER_SetPlayerBufNumber (
        PLAYHANDLE          hHandle,
        HI_U32              u32BufNumber
);
```

Parameters

hHandle

[IN] 播放库句柄

u32BufNumber

[IN] 所要设置的单视频播放时缓冲区最大的帧数，取值范围（高清[0-20]，普通[0-50]），SDK 默认的帧缓冲区大小为 0

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remarks

设置网络延时和播放流畅度可以通过此接口来进行调节。s32RBNum 值越大，播放的流畅性越好，相对延时就大；s32RBNum 值越小，播放的延时就小，但是当网络不太顺畅的时候，会有丢帧现象，影响播放的流畅性。

如果要做到音视频同步，请结合 HI_PLAYER_SetAORBParam 使用。

HI_PLAYER_SetAORBParam

缓冲的音频帧数，用于实时预览

```
HRESULT    HI_PLAYER_SetAORBParam (
        PLAYHANDLE          hHandle,
        HI_S32              s32MaxFrameRate,
        HI_S32              s32CurFrameRate
);
```

Parameters

hHandle

[IN] 播放库句柄

s32MaxFrameRate

[IN] 最大帧率

s32CurFrameRate

[IN] 当前帧率

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remarks

该函数结合 HI_PLAYER_SetPlayerBufNumber 函数使用，当设置缓冲帧数后，要通过最大帧率与当前的帧率计算出缓冲的音频帧数

HI_PLAYER_SetAutoAdjust

设置播放画面的显示比例

```
HRESULT    HI_PLAYER_SetAutoAdjust (
        PLAYHANDLE          hHandle,
);
```

Parameters

hHandle

[IN] 播放库句柄

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_PLAYER_GetAutoAdjust

获取播放画面的显示比例

```
HI_BOOL    HI_PLAYER_GetAutoAdjust (
        PLAYHANDLE          hHandle
);
```

Parameters

hHandle

[IN] 播放库句柄

Return Values

HI_SUCCESS 表示当前显示为自动调节状态，HI_ FAILURE 表示非自动调节状态。

HI_PLAYER_GetCurrentPts

获取播放画面的显示比例，用于流播放模式

```
HRESULT    HI_PLAYER_GetCurrentPts (
        PLAYHANDLE          hHandle,
        HI_U64              *pCurPts
);
```

Parameters

hHandle

[IN] 播放库句柄

pCurPts

[IN] 当前播放的时间戳

Return Values

HI_SUCCESS 表示成功，失败返回错误代码。

HI_PLAYER_DisplayAll

显示区域电子放大

```
HI_S32    HI_PLAYER_DisplayAll (
        PLAYHANDLE      hHandle,
        HI_S32          s32Left,
        HI_S32          s32Top,
        HI_S32          s32Right,
        HI_S32          s32Bottom,
        HI_BOOL         bDisplayAll
);
```

Parameters

lHandle

[IN] 操作句柄

s32Left

[IN] 相对于显示屏幕的左上角坐标（x）

s32Top

[IN] 相对于显示屏幕的左上角坐标（y）

s32Right

[IN] 相对于显示屏幕的右下角坐标（x）

s32Bottom

[IN] 相对于显示屏幕的右下角坐标（y）

bDisplayAll

[IN] 是否显示整个图像，HI_TRUE—显示全部，HI_FALSE—使用区域放大功能
默认值 HI_TRUE，设置显示区域必须使用 HI_FALSE；

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remark

函数的功能在 SDK 内部显示动态电子放大，输入的坐标是相对窗口显示坐标。

HI_PLAYER_SetDisplayMode

设置显示模式

```
HI_S32    HI_PLAYER_SetDisplayMode (
```

```
        PLAYHANDLE                  hHandle,
        PLAYER_DISPLAYMODE_E        eDisplayMode
    );
```

Parameters

lHandle

[IN] 操作句柄

eDisplayMode

[IN] 显示模式

```
typedef enum hiPLAYER_DISPLAYMODE_E
{
    PLAYER_DPY_D3D = 0,     //D3D 模式
    PLAYER_DPY_DDRAW,       //Ddraw 模式
}PLAYER_DISPLAYMODE_E;
```

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_PLAYER_GetDisplayMode

获取当前显示模式

```
HI_S32    HI_PLAYER_GetDisplayMode (
    PLAYHANDLE                  hHandle,
    PLAYER_DISPLAYMODE_E        *pDisplayMode
);
```

Parameters

lHandle

[IN] 操作句柄

pDisplayMode

[OUT] 显示模式

```
typedef enum hiPLAYER_DISPLAYMODE_E
{
    PLAYER_DPY_D3D = 0,     //D3D 模式
    PLAYER_DPY_DDRAW,       //Ddraw 模式
}PLAYER_DISPLAYMODE_E;
```

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

HI_PLAYER_SetDrawCallBack

设置 D3D 显示模式下叠加标志

```
HI_S32    HI_PLAYER_SetDrawCallBack (
    PLAYHANDLE    hHandle,
```

```
    HI_BOOL          bDrawCallback
);
```

Parameters

lHandle

[IN] 操作句柄

bDrawCallback

[IN] 启动 D3D 叠加标志

Return Values

成功返回 HI_SUCCESS，失败返回错误代码。

Remark

播放库为了适应在 WIN7 以及以上版本的需求，修改了默认显示方式，设置成 D3D 并保留了 Ddraw 模式。如果当前显示模式为 D3D 模式，需要从回调函数 HI_PLAYER_SetPostDrawCallBack 中调用画图句柄进行图像叠加（比如画报警框，叠加文字等），需要将 D3D 模式打开，此函数功能为打开回调标志。