VIVOTEK NETWORK DEVELOPMENT PLATFORM

AvSynchronizer
Version 4.4.0.54
2009/8/31

VIVOTEK may make changes to specifications and product descriptions at any time, without notice.

The following is trademarks of VIVOTEK Inc., and may be used to identify VIVOTEK products only: VIVOTEK. Other product and company names contained herein may be trademarks of their respective owners.

TABLE OF CONTENTS

1. Overview

1.1. Introduction

This document describes the properties and methods supported by the VIVOTEK AvSynchronizer module.

1.2. Getting Started with AvSynchornizer module

The main function of AvSynchronizer module is to display the audio and video from DataBroker or MediaDataBase module synchronously.

1.3. File Structure

FILE	DESCRIPTION
doc\VNDP_AvSynchronizer_API.pdf	This manual
lib\d_AvSynchronizer.lib	The dynamic linking library
lib\AvSynchronizer.dll	The dynamic runtime library
lib\AviConverter.dll	The dynamic runtime library for AVI conversion
inc\AvSynchronizer.h	Header file
inc\SrvTypeDef.h	Common definition file
inc\datapacketdef.h	Data packet definition file

2. PROGRAMMER'S GUIDE

2.1. Using AvSynchronizer Module

You can play audio/video from the network through the DataBroker or the audio/video from the MediaDataBase with AvSynchronizer module. And it is also possible to decode audio and video data directly by decoder channel. To convert media into AVI files, this module also provides an AVI channel to handle it.

The Linux platform is newly supported, but only decoder channel is ported. Those functions that are not available in Linux would be marked with underline in the following document.

Play audio/video from DataBroker

After initial the AvSynchronizer module, you might create a channel to play the audio/video from DataBroker. Using AvSynchronizer_CreateChannel to create the channel. Call AvSynchronizer_StartChannel to tell the Avsynchronizer to start play the audio/video. then input audio/video through AvSynchronizer_InputEncodedMediaFrame.

This channel is only available in Windows 32 / Windows CE platform now.

Play audio/video from raw data

After initial the AvSynchronizer module, you might create a playback channel to play the audio/video from MediaDatabase. Using AvSynchronizer_CreatePlaybackChannel to create the channel. The call AvSynchronizer_StartChannel to tell the Avsynchronizer to start play the audio/video and input audio/video through AvSynchronizer_InputPlaybackMediaFrame.

This channel is only available in Windows 32 platform now.

Decode audio/video and get the raw data

After initial the AvSynchronizer module, you could create a decoder channel to decode audio/video either from MediaDatabase or from on-line. Using AvSynchronizer_CreateDecoderChannel to create the channel. Then input audio/video through AvSynchronizer_InputToBeDecodedMediaFrame. The decoded data would be called back through LPDECODEFRAMECALLBACK. It is possible to decode audio and video in synchronous mode, the two functions AvSynchronizer_DecodeAudio and AvSynchronizer_DecodeVideo are now provided to satisfy the request.

Convert audio/video to AVI file

After initial the AvSynchronizer module, you could create an AVI channel to convert audio/video from MediaDatabase into AVI file. Using AvSynchronizer_CreateAVIChannel to create the channel. Start the channel by calling AvSynchronizer_StartAVIChannel before input packet. Then input audio/video through AvSynchronizer_InputAVIMedia. Use AvSynchronizer_ChooseVideoCompressor or AvSynchronizer_ChooseAudioCompressor to set the compressor for video and audio.

This channel is only available in Windows 32 platform now.

Convert live audio/video to AVI file

After initial the AvSynchronizer module, you could create a live AVI channel to convert live audio/video from MediaDatabase into AVI file. Here the live means the data comes from a live connection to video servers. For this kind of data, use the decoder callback for live channel to avoid the same frames are decoded twice (use the decoder channel or non-live AVI channel). It is ok to use the non- AVI channel to create AVI, but the performance would be worse. It is recommend to application to change a file when the video size is changed during AVI generation. Users could do this by closing the AVI file and reopen again when it receive the size change callback from this module. Using AvSynchronizer_CreateLiveAVIChannel to create the channel. Start the channel by calling AvSynchronizer_StartAVIChannel before input packet. Set a decoder callback LPDECODEFRAMECALLBACK_EX when for the live channel. Then input audio/video through AvSynchronizer_InputDecodedAVIMedia whenever the decoded frames are notified. Use AvSynchronizer_ChooseChannelVideoCompressor or AvSynchronizer_ChooseChannelAudioCompressor to set the compressor for video and audio.

This channel is only available in Windows 32 platform now.

3. Sample code

3.1. PlayRawFile

DESCRIPTION

Read a binary file that is saved from the AvCallback of DataBroker. Parse this file and input the parsed data to the playback channel of AvSynchronizer. This sample also demos the usage of Pause / Resume / Next frame / change display speed.

SAMPLE CODE

STEP 1. Prepare the callback functions

You have to prepare 2 callback functions for AvSynchronizer. One for display callback and the other is status callback.

```
//DisplayCallback is to draw the caption of graph if needed.
SCODE __stdcall AvSynchronizerDisplayCallBack(DWORD dwContext, TDISPLAYINFO*
ptDisplayInfo)
{
        //Draw the caption
}
///StatusCallback is used to inform AP the status of AvSynchronizer.
SCODE __stdcall AvSynchronizerStatusCallBack(DWORD dwContext, DWORD
dwStatusCode, DWORD dwParam1, DWORD dwParam2)
{
   //handles the status such as decode error, frame size changes, ...etc.
}
```

STEP 2. Initialize AvSynchronizer

```
scRet = AvSynchronizer_InitialEx(&hAvSync, AvSyncStatusCallback, AvSyncDisplayCallback,
hWnd, AUDIOOUT_FOCUS_NORMAL, 0, AVSYNCHRONIZER_VERSION, 0, RGB(0, 0, 0));
```

STEP 3. Create Playback channel

```
TPBCHOPTION tPBCHOption;
memset(&tPBCHOption, 0, sizeof(tPBCHOption));
tPBCHOption.dwFlags = PBCH_STATUSCB | PBCH_DISPLAYCB |
                      PBCH_STATUSCONTEXT | PBCH_DISPLAYCONTEXT |
                      PBCH_ALLBORDER | PBCH_BITMAP;
tPBCHOption.pfStatus  =AvShynchronizerStatusCallback;
tPBCHOption.pfDisplay = AvSynchronizerDisplayCallack;
```

```
tPBCHOption.dwStatusContext  = dwStatusContext;
tPBCHOption.dwDisplayContext = dwDisplayContext;
tPBCHOption.dwTopBorderSize    = 20;
tPBCHOption.dwLeftBorderSize   = 0;
tPBCHOption.dwBottomBorderSize = 0;
tPBCHOption.dwRightBorderSize  = 0;
tPBCHOption.hBMP = hBitmapHandle;
tPBCHOption.hDisplay = hDisplayWindow;
scRet = AvSynchronizer_CreatePlaybackChannel(hAvSync, &hPlaybackChannel,
            tPBCHOption);
```

STEP 4. Startt Playback channel

```
scRet = AvSynchronizer_StartChannel(hPlaybackChannel, dwStartFlag);
```

STEP 5.Input TMediaDataPacketInfo to playback channel

```
while (bEndOfFileIsFalse)
{
    scRet = AvSynchronizer_InputPlaybackMediaFrame(hPlaybackChannel,
            ptMediaDataPacket);
}
```

STEP 6. Wait until all frames are displayed then stop channel

```
TAVSYNCQUEUESTATUS tQueueStatus;
do{
            Sleep(16);
            memset(&tQueueStatus, 0, sizeof(tQueueStatus));
            // Get remaining audio and video elements in queue
            scRet = AvSynchronizer_GetRemainingQueueElementCount(m_
            hPlaybackChannel, &tAvSyncQueueStatus);
}while ((tQueueStatus.dwAudioQueueElements != tQueueStatus.dwAudioQueueSize) ||
        (tQueueStatus.dwVideoQueueElements != tQueueStatus.dwVideoQueueSize - 1));
scRet =AvSynchronizer_StopChannel(hPlaybackChannel);
```

STEP 7. Delete channel and release AvSynchronizer

scRet = AvSynchronizer_DeleteChannel(hAvSync, &hPlaybackChannel);

scRet = AvSynchronizer_Release(&hAvSync);

TIPS

1. remember to set media type of playback channel

2. Before calling NextFrame, please pause the channel

3. playback channel will block the input thread if the internal queue is full. User can disable this function by calling:

AvSynchronizer_SetChannelOption(hPlaybackChannel,

SETCH_NOT_BLOCK_WHEN_QUEUE_FULL, FALSE, 0);

3.2. PlayWithDataBroker

DESCRIPTION

This sample demos the usage of live channel when connect to camera by DataBroker.

SAMPLE CODE

STEP 1. Initialize AvSynchronizer

```
scRet = AvSynchronizer_InitialEx(&hAvSync, AvSyncStatusCallback, AvSyncDisplayCallback,
                    hWnd, AUDIOOUT_FOCUS_NORMAL, 0, AVSYNCHRONIZER_VERSION,
                    0, RGB(0, 0, 0));
```

STEP 2. Create LiveChannel channel

```
TCHANNELOPTION tCHOption;
memset(&tCHOption, 0, sizeof(tCHOption));
tCHOption.dwFlags = CH_STATUSCB | CH_DISPLAYCB | CH_STATUSCONTEXT
CH_DISPLAYCONTEXT | CH_ALLBORDER | CH_BITMAP;
tCHOption.pfStatus  = AvSynchronizerStatusCallback;
tCHOption.pfDisplay = AvSynchronizerDisplayCallback;
tCHOption.dwStatusContext = dwStatusContext;
tCHOption.dwDisplayContext = dwDisplayContext;
tCHOption.dwTopBorderSize    = 20;
tCHOption.dwLeftBorderSize   = 0;
tCHOption.dwBottomBorderSize = 0;
tCHOption.dwRightBorderSize  = 0;
tCHOption.hBMP     = hBMP;
tCHOption.hDisplay = hDisplay;
scRet = AvSynchronizer_CreateChannel(hAvSync, &hLiveChannel, tCHOption);
```

STEP 3. Start live channel

```
scRet = AvSynchronizer_StartChannel(hLiveChannel, DRAW_CONNECTING);
```

STEP 4. Set media type

Update the media type of live channel when DataBroker callback eOnConnectionInfo.

STEP 5. Input media frame in DataBroker AVCallback

```
DataBrokerAVCallback(DWORD dwContext,TMediaDataPacketInfo *ptPacket)
{
    scRet = AvSynchronizer_InputEncodedMediaFrame(hLiveChannel, ptPacket);
    if (scRet == AVSYNCHRONIZER_E_QUEUE_FULL)
            return DATABROKER_S_FRAME_NOT_HANDLED;
    ...
}
```

STEP 6. Stop channel

```
scRet =AvSynchronizer_StopChannel(hLiveChannel);
```

STEP 7. Delete channel and release AvSynchronizer

```
scRet = AvSynchronizer_DeleteChannel(hAvSync, &hLiveChannel);
scRet = AvSynchronizer_Release(&hAvSync);
```

TIPS

1. remember to set media type of playback channel

3.3. GetRawDataFromDevice

DESCRIPTION

This sample demos how to use decoder channel to get raw data, RGB or YUV, when connect to camera by DataBroker.

SAMPLE CODE

STEP 1. Prepare decode callbak

```
SCODE __stdcall AvSynchronizerFrameDecodeCallback(DWORD dwContext,
                                        TMediaType tFrameType, TFRAMEINFO *tFrameInfo)
{
    if (tFrameType == AVSYNCHRONIZER_MEDIATYPE_VIDEO_ONLY)
    {
            //do some operations on tFrameInfo->tVideoFrame.pbyPicture
    }
    else if (tFrameType == AVSYNCHRONIZER_MEDIATYPE_AUDIO_ONLY)
    {
            //do some operations on tFrameInfo->tAudioFrame.pbySound
    }
}
```

STEP 2. Initialize AvSynchronizer

```
scRet = AvSynchronizer_InitialEx(&hAvSync, NULL, NULL, NULL, 0, 0,
                                AVSYNCHRONIZER_VERSION, DECODER_CHANNEL_ONLY, 0);
```

STEP 3. create decoder channel

```
TDECCHOPTION tDecCHOption;
memset(&tDecCHOption, 0, sizeof(tDecCHOption));
tDecCHOption.pfDecodeFrame     = AvSynchronizerFrameDecodeCallback;
tDecCHOption.dwAvDecodeContext = dwDecodeFrameContext;
//please refer to EPIXELFORMAT
tDecCHOption.dwRawDataFormat   = ePixelFormat;
scRet = AvSynchronizer_CreateDecoderChannel(hAvSync, &hDecodeChannel,
            &tDecCHOption);
```

STEP 4. Option. Set output video format. After creating the decoder channel,

user can change the output data here.

scRet = AvSynchronizer_SetChannelOption(hDecodeChannel, SETCH_DECODE_DATA_FORMAT, ePixelFormat, 0);

STEP 5. Input media frame to decoder channel and the decoded raw data will callback to AvSynchronizerFrameDecodeCallback

scRet = AvSynchronizer_InputToBeDecodedMediaFrame(hDecodeChannel,
pMediaDataPacket);

STEP 6. Delete channel and release AvSynchronizer

scRet = AvSynchronizer_DeleteDecoderChannel(hAvSync, &hDecodeChannel);

scRet = AvSynchronizer_Release(&hAvSync);

3.4. DisplayAndGetRawDataWithLiveChannel

DESCRIPTION

Live channel of AvSynchronizer can support display and get raw data of a single frame. This sample shows how to do that.

SAMPLE CODE

STEP 1. Prepare decode, display and status callbak

```
SCODE __stdcall AvSynchronizerFrameDecodeCallbackEx(DWORD dwContext, TMediaType
tFrameType, TFRAMEINFOEX *tFrameInfoEx)
{
    if (tFrameType == AVSYNCHRONIZER_MEDIATYPE_VIDEO_ONLY)
    {
            //do some operations on tFrameInfo->tVideoFrame.pbyPicture
    }
    else if (tFrameType == AVSYNCHRONIZER_MEDIATYPE_AUDIO_ONLY)
    {
            //do some operations on tFrameInfo->tAudioFrame.pbySound
    }
}
//Display and status callback as usual
```

STEP 2. The usage of live channel please refer to PlayWithDataBroker

STEP 3. Set raw data format and related options

```
//set the output format
scRet = AvSynchronizer_SetChannelOption(hLiveChannel, SETCH_DECODE_DATA_
FORMAT, ePixelFormat, 0);
//enable output raw data
AvSynchronizer_SetChannelOption(m_hLiveChannel, SETCH_OUTPUT_VIDEO_FRAME,
(DWORD)TRUE, 0);
//set decode callback
AvSynchronizer_SetChannelOption(m_hLiveChannel, SETCH_DECODE_CBEX
dwDecodeFrameContext, (DWORD)AvSynchronizerFrameDecodeCallbackEx);
```

TIPS

1. The decode callback of live channel is DecodeCallbacEx rather than DecodeCallback. The only difference between them is the interface. DecodeCallbackEx uses TFRAMEINFOEX while DecodeCallback uses TFRAMEINFO.
2. User can get more than one formats for a single video frame.
3. If the data member, bContinueDecode, of TFRAMEINFOEX is TRUE, AvSynchronizer will callback the DecodeCallbackEx again with the format, dwNextFormat.

3.5. DisplayAndGetRawDataWithPlaybackChannel

DESCRIPTION

Playback channel of AvSynchronizer can support display and get raw data of a single frame. This sample is similar to DisplayAndGetRawDataWithLiveChannel but create playback channel.

SAMPLE CODE

Please reference the source code.

3.6. ConvertAVI

DESCRIPTION

Use playback channel to display a saved data and convert the data to an AVI file.

SAMPLE CODE

STEP 1. The way to use playback client, please refer to PlayRawFile

STEP 2. Create AVI channel

```
// Avi convertor need to use COM object
CoInitialize (NULL);
TAVICHOPTION tAviCHOption;
tAviCHOption.pfStatus  = pfAviStatusCallback;
tAviCHOption.pfCaption = pfAviCaptionCallback;
tAviCHOption.dwStatusContext  = dwAviStatusContext;
tAviCHOption.dwCaptionContext = dwAviCaptionContext;
// Create AVI channel
scRet = AvSynchronizer_CreateAVIChannel(hAvSync, &hAviChannel, &tAviCHOption);
```

STEP 3. Start AVI channel

```
TAVICHOPTION2 tAviCHOption2;
memset(&tAviCHOption2, 0, sizeof(tAviCHOption2));
// Use the first frame's width and height
tAviCHOption2.dwFlag = FIXED_VINFO_AT_FIRST_FRAME;
tAviCHOption2.bVideoEnable = TRUE;
tAviCHOption2.tAviVInfo.dwBitCount = DEFAULT_AVIBITCOUNT;
strcpy(tAviCHOption2.szFileName, "Playbak.avi");
tAviCHOption2.tBorder.dwHeader = 20;
tAviCHOption2.tBorder.dwFooter = 0;
tAviCHOption2.tBorder.dwLeft   = 0;
tAviCHOption2.tBorder.dwRight  = 0;

// Start AVI channel
scRet = AvSynchronizer_StartAVIChannel(hAviChannel, &tAviCHOption2);
```

STEP 4. Input media frame to Playback channel and Avi channel

scRet = AvSynchronizer_InputPlaybackMediaFrame(hPlaybackChannel, pMediaPacket);

scRet = AvSynchronizer_InputAVIMedia(hAviChannel, pMediaPacket);

STEP 5. Wait until all packets are input and stop Playback and Avi channel.

//wait all packets are displayed as previous sample shows

scRet = AvSynchronizer_StopChannel(hPlaybackChannel);

scRet = AvSynchronizer_StopAVIChannel(hAviChannel);

//release COM

CoUninitialize();

STEP 6. Delete channels and release AvSynchronizer

scRet = AvSynchronizer_DeleteChannel(hAvSync, &hPlaybackChannel);

scRet = AvSynchronizer_DeleteAVIChannel(hAvSync, &hAviChannel);

scRet = AvSynchronizer_Release(&hAvSync);

3.7. GetSnapShotAndSaveAsJPG

DESCRIPTION

This sample uses DataBroker to connect to camera, get a snapshot and save the picture as Jpeg file.

SAMPLE CODE

STEP 1. Initialize AvSynchronizer and use live channel to display A/V as sample PlayWithDataBroker.

STEP 2. Get snapshot and save the picture as Jpeg

```
TSNAPSHOT tSnapshot;
memset(&tSnapshot, 0, sizeof(tSnapshot));
scRet = AvSynchronizer_GetCurrentSnapShot(m_hLiveChannel, &tSnapshot, ((PF_JPEG <<
16) | SPECIFY_FMT) | ORIGINAL_SIZE);
//do some operations on the jpeg data pointed by tSnapshot.pDataStart
scRet = AvSynchronizer_FreePicture(&tSnapshot);
```

TIPS

1. Remember to free the buffer pointer by TSNAPSHOT.

3.8. GetDifferentRawFormatAtOnce

DESCRIPTION

This sample is similar to DisplayAndGetRawDataWithLiveChannel, but it retrieves more than one data format, RGB and YUV, from a single picture.

SAMPLE CODE

STEP 1. Prepare the decode callback function

```
SCODE __stdcall AvSynchronizerFrameDecodeCallbackEx(DWORD dwContext, TMediaType
tFrameType, TFRAMEINFOEX *tFrameInfoEx)
{
    if (tFrameType == AVSYNCHRONIZER_MEDIATYPE_VIDEO_ONLY)
    {
            //do some operations on tFrameInfo->tVideoFrame.pbyPicture
            if (tFrameInfoEx->dwCBCount == 1)
            {
                    if (dwNextDecodeFormat != 0)
                    {
                            //set the format that user wants
                            tFrameInfoEx->dwNextFormat = eNextPixelFormat;
                            tFrameInfoEx->bContinueDecode = TRUE;
                    }
            }
    else if (tFrameType == AVSYNCHRONIZER_MEDIATYPE_AUDIO_ONLY)
    {
            //do some operations on tFrameInfo->tAudioFrame.pbySound
    }
}
```

STEP 2. The usage of live channel and DataBroker, please refer to DisplayAndGetRawDataWithLiveChannel

TIPS

1. When user wants AvSynchronizer to return more video format, please set bContinueDecode to TRUE and the dwNextFormat

3.9. CustomDrawWithDisplayCallback

DESCRIPTION

This sample is similar to PlayWithDataBroker. There are only two differences. The first is the way to initialize AvSynchronizer and the other is the implementation of display callback.

SAMPLE CODE

STEP 1. Prepare the display callback function

The display callback function is similar to PlayWithDataBroker. But in this case, AvSynchronizer will callback a whole DC to AP. AP can draw what they want to draw.

STEP 2. Initialize AvSynchronizer and the following steps are the same as PlayWithDataBroker

```
scRet = AvSynchronizer_InitialEx(&hAvSync, AvSyncStatusCallback, AvSyncDisplayCallback,
hWnd, AUDIOOUT_FOCUS_NORMAL, 0, AVSYNCHRONIZER_VERSION, CAPTION_ON_
GRAPH, RGB(0, 0, 0));
```

TIPS

1. When turning on the flag, CAPTION_ON_GRAPH, it consumes more CPU power. This flag makes AvSynchronizer callback larger DC and increases the callback frequency.

3.10. DigitalZoom

DESCRIPTION

This sample uses live channel to show live streaming. It also demos the way to use digital zoom.

SAMPLE CODE

STEP 1. The usage of live channel is the same as PlayWithDataBroker

STEP 2. set digital zoom

```
TDIGITALZOOMPARAM tDZParam;
memset(&tDZParam, 0, sizeof(tDZParam));
DWORD dwCanvasBorderSize = 4;
DWORD dwZoomAreaBorderSize = 4;
tDZParam.bCanvasWin       = TRUE;
//the window handle that will show the whole picture
tDZParam.unCanvas.hTarget = hCanvasWnd;
tDZParam.rectCanvasBorder.top    = dwCanvasBorderSize;
tDZParam.rectCanvasBorder.left   = dwCanvasBorderSize;
tDZParam.rectCanvasBorder.bottom = dwCanvasBorderSize;
tDZParam.rectCanvasBorder.right  = dwCanvasBorderSize;
tDZParam.bZoomWin         = TRUE;

//the window handle that AvSynchronizer will draw digital zoom on it
tDZParam.unZoom.hZoomArea = hZoomasWnd;
tDZParam.rectZoomBorder.top    = dwZoomAreaBorderSize;
tDZParam.rectZoomBorder.left   = dwZoomAreaBorderSize;
tDZParam.rectZoomBorder.bottom = dwZoomAreaBorderSize;
tDZParam.rectZoomBorder.right  = dwZoomAreaBorderSize;
tDZParam.bZoomEnabled = TRUE;
tDZParam.bDisplayed   = TRUE;
DWORD dwDZFlags = DZ_CANVAS | DZ_CANVAS_DISPLAY | DZ_ZOOM_AREA |
DZ_ENABLE_ZOOM | DZ_CANVAS_BORDER | DZ_ZOOM_AREA_BORDER;
scRet = AvSynchronizer_SetChannelOption(m_hLiveChannel, SETCH_DIGITAL_ZOOM,
(DWORD)(&tDZParam), dwDZFlags);
```

3.11. Display32Channels

DESCRIPTION

This sample uses playback channel to show the streaming from a file.

SAMPLE CODE

STEP 1. The usage of live channel is the same as PlayRawFile. But it initializes 4 AvSynchronizers and each one contains 8 live channels

TIPS

1. Each AvSynchronizer has one internal thread to decode the streaming. If the AP is running on the CPU with multiple threads or cores, we suggest that AP can initialize more than on AvSynchronizer for better performance.

4. API Reference

This chapter contains the API function calls for the AvSynchronizer.

4.1. Enumeration

The enumeration used is depicted here.

EANTITEARING

This enumeration indicates if AvSynchronizer should turn on anti-tearing or not when drawing pictures.

```
typedef enum {
  ANTI_TEARING_DISABLED          = 1,
  ANTI_TEARING_ALWAYS            = 2,
  ANTI_TEARING_AUTO              = 3,
} EANTITEARING;
```

Values

ANTI_TEARING_DISABLED

Disable the anti-tearing(don't wait for verical blank)

ANTI_TEARING_ALWAYS

Always wait for vertical blank

ANTI_TEARING_AUTO

If the height of destination window is bigger than 240, it will wait for vertical blank.

Remarks

This enumeration is only available for Windows 32 platform.

Requirements

AvSynchronizer.h

EAUDIOFOCUSTYPE

This enumeration indicates the focus type of the DirectSound

```
typedef enum {
   AUDIOOUT_FOCUS_NORMAL          = 1,
   AUDIOOUT_FOCUS_STICKY          = 2,
   AUDIOOUT_FOCUS_GLOBAL          = 3,
} EAUDIOFOCUSTYPE;
```

Values

AUDIOOUT_FOCUS_NORMAL

The buffer is muted if user switches focus to another application.

AUDIOOUT_FOCUS_STICKY

If the user switches to another application not using DirectSound, the buffer is still audible. However, if the user switches to another DirectSound application, the buffer is muted.

AUDIOOUT_FOCUS_GLOBAL

The application can continue to play its buffers if the user switches focus to another application, even if the new application uses DirectSound.

Remarks

This enumeration is only available for Windows 32 platform.

Requirements

AvSynchronizer.h

EAVISTATUSCODE

This enumeration indicates status code for AVI conversion channel

```
typedef enum {
   AVI_WRITE_FRAME_TIME       = WM_USER + 100,
} EAVISTATUSCODE;
```

Values

AVI_WRITE_FRAME_TIME

Notify application the time value for the current processing media. This could let application shows the processing progress correctly.

Remarks

This enumeration is only available for Windows 32 platform.

Requirements

AvSynchronizer.h

EAVSYNCVIDEOMODE

This enumeration indicates the display mode.

```
typedef enum {
  AVSYNCHRONIZER_VIDEO_DDRAW    = 1,
  AVSYNCHRONIZER_VIDEO_GDI      = 2
} EAVSYBCINITFLAG;
```

Values

AVSYNCHRONIZER_VIDEO_DDRAW

the current display mode is DDraw

AVSYNCHRONIZER_VIDEO_GDI

the current display mode is GDI

Remarks

Requirements

AvSynchronizer.h

EAVSYBCINITFLAG

This enumeration indicates the initial mode. Used by AvSynchronizer_Initial or AvSynchronizer_InitialEx

```
typedef enum {
  DECODER_CHANNEL_ONLY      =0x00000001,
  USE_GDI_ONLY              =0x00000002,
  USE_DIRECTDRAW_ONLY       =0x00000004,
  CAPTION_ON_GRAPH          =0x00000008,
  BETTER_GDI_STRETCH        =0x00000010,
  ONE_PASS_DDRAW            =0x00000020,
  INDIVIDUAL_SURFACE        =0x00000040,
  ORG_SIZE_MODE             =0x00000080,
  FORCE_GDI                 =0x00000100,
  CHANGE_TIME_PRECISION     =0x00000200,
  FORCE_NON_YUV             =0x00000400,
  MEGA_PIXEL                =0x00000800,
  AV_DONT_SYNC              =0x80000000,
} EAVSYBCINITFLAG;
```

Values

DECODER_CHANNEL_ONLY

If this flag is set, then the AvSynchronizer_Initial function won't fail when the video initialize failed. And it will not check for the nullity of window handle passed.

USE_GDI_ONLY

If this flag is set, then the AvSynchronizer_Initial function will be forced to use GDI as display engine. GDI is slower but more compatible in different platform. This flag cannot be used with USE_DIRECTDRAW_ONLY flag.

USE_DIRECTDRAW_ONLY

If this flag is set, then the AvSynchronizer_Initial function will be forced to use DirectDraw as display engine. If the running environment can't support DirectDraw, the initial would be failed. This flag cannot be used with USE_GDI_ONLY flag.

CAPTION_ON_ GRAPH

If this flag is set, the LPDISPLAYCALLBACK will be called every frame. It will cause a worse performance.

BETTER_GDI_ STRETCH

If this flag is set, it will get more quality when using GDI with stretching but slower. If DirectDraw is used, this flag is ignored.

ONE_PASS_DDRAW

When in DirectDraw mode, some video card would have problem for the original two passes drawing mode (draw the YUV/BMP to a window size offscreen surface, and than show the surface to primary surface). Set this mode could solve this problem. But one pass mode would suffer the video shivering when other window covers the video partly. This flag will be of no effect when in GDI mode.

INDIVIDUAL_SURFACE

When in DirectDraw, decode the video directly to the YUV surface. In such case, the overall performance will be improved by about 20%.

ORG_SIZE_MODE

If AP needs to use original picture size mode to draw the video and caption/borders, it needs to specify this flag when initialize AVSynchronizer. And this flag could not be changed during runtime. Set this flag will need more memory so the default setting is off.

FORCE_GDI

Force the module to use GDI to draw the video. This allows the caller to switch between DirectDraw and GDI without release current AVSynchronizer object.

CHANGE_TIME_PRECISION

Change the time precision of OS, with this flag, the sleeping time of the thread in current application would be 1 fourth the original unit. This would get a faster system response time. But in such mode, the CPU would be consumed more than the normal case.

FORCE_NON_YUV

For some cards, the YUV operation will cause green screen. Set this flag will avoid the problem, but the system performance will worse than YUV mode.

MEGA_PIXEL

AvSynchronizer will allocate the internal buffer as 1280 x 1024 when initialization if this flag is set.

AV_DON'T_SYNC

If this flag is set, then the mechanism of synchronization will be disabled.

Remarks

If DECODER_CHANNEL_ONLY is set and the initialization of video was failed, then you can only create the decoder channel.

USE_DIRECTDRAW_ONLY, ONE_PASS_DDRAW, INDIVIDUAL_SURFACE is only available on Windows 32 platform.

Requirements

AvSynchronizer.h

ECHFLAG

This enumeration indicates which field of the structure TCHANNELOPTION is valid.

```
typedef enum {
  CH_STATUSCB              = 0x00000001,
  CH_DISPLAYCB             = 0x00000002,
  CH_ALLCALLBACK           = 0x00000003,
  CH_STATUSCONTEXT         = 0x00000004,
  CH_DISPLAYCONTEXT        = 0x00000008,
  CH_ALLCONTEXT            = 0x0000000c,
  CH_VOLUME                = 0x00000010,
  CH_LFBORDER              = 0x00000020,
  CH_RTBORDER              = 0x00000040,
  CH_TPBORDER              = 0x00000080,
  CH_BTBORDER              = 0x00000100,
  CH_ALLBORDER             = 0x000001e0,
  CH_BITMAP                = 0x00000200,
  CH_MOTION                = 0x00000400,
  CH_MOTIONRECT            = 0x00000800,
  CH_MOTIONRECTALERT       = 0x00001000,
  CH_VIDEOOUT              = 0x00002000,
  CH_AUDIOOUT              = 0x00004000,
  CH_ALL                   = 0xFFFFFFFF
} ECHFLAG;
```

Values

CH_STATUSCB
pfStatus is valid.

CH_DISPLAYCB
pfDisplay is valid.

CH_ALLCALLBACK
All callback is valid. pfStatus and pfDisplay.

CH_STATUSCONTEXT
dwStatusContext is valid.

CH_DISPLAYCONTEXT
dwDisplayContext is valid.

CH_ALLCONTEXT
All the context is valid. dwStatusContext and dwDisplayContext.

CH_VOLUME
dwVolume is valid,

CH_LFBORDER
dwLeftBorderSize is valid.

CH_RTBORDER
dwRightBorderSize is valid.

CH_TPBORDER
dwTopBorderSize is valid.

CH_BTBORDER
dwBottomBorderSize is valid.

CH_ALLBORDER
All the Border is valid. dwLeftBorderSize, dwRightBorderSize, dwRightBorderSize and dwBottomBorderSize.

CH_BITMAP
hBMP and bSaveBmp are valid.

CH_MOTION
bMotionAlert is valid.

CH_MOTIONRECT
dwMotionRect is valid.

CH_MOTIONRECTALERT
dwMotionRectAlert is valid.

CH_VIDEOOUT
bVideoOut is valid.

CH_AUDIOOUT
bAudioOut is valid.

CH_ALL
All fields are valid.

Remarks

CH_BITMAP option is only available for Windows 32 platform.

Requirements

AvSynchronizer.h

ECHANNELSTATUS

This enumeration lists the status code that appears when LPSTATUSCALLBACK get called.

```
typedef enum {
  AVSYNCHRONIZER_STATUSCODE_DECODE_ERROR          = 1,
  AVSYNCHRONIZER_STATUSCODE_MEMALLOC_ERROR        = 2,
  AVSYNCHRONIZER_STATUSCODE_FRAME_SIZE_CHG        = 3,
  AVSYNCHRONIZER_STATUSCODE_AUDIO_START           = 4,
  AVSYNCHRONIZER_STATUSCODE_AUDIO_STOP            = 5,
  AVSYNCHRONIZER_STATUSCODE_MOTION                = 6,
  AVSYNCHRONIZER_NEW_IMAGE                        = 7,
  AVSYNCHRONIZER_FIRST_IMAGE                      = 8,
  AVSYNCHRONIZER_STATUSCODE_VIDEO_MODE            = 9
} ECHANNELSTATUS;
```

Values

AVSYNCHRONIZER_STATUSCODE_DECODE_ERROR

Decode error. dwParam1 indicate the frame type (AUDIO_FARME or VIDEO_FRAME)

AVSYNCHRONIZER_STATUSCODE_MEMALLOC_ERROR

Memory allocation error happens during decoding.

AVSYNCHRONIZER_STATUSCODE_FRAME_SIZE_CHG

The decoded video size is changed. When the first frame is decoded it will issue a call to notify application. dwParam1 is width and dwParam2 is height of the new video size. Note: Please do not resize the window in the called back function. This would cause a deadlock. Use PostMessage() or timer to resize the window instead if needed.

AVSYNCHRONIZER_STATUSCODE_AUDIO_START

Called when audio only media type starts to play audio. This could let application start to render on screen.

AVSYNCHRONIZER_STATUSCODE_AUDIO_STOP

Called when audio only media type stop to play audio. These two codes let application could start and stop a timer to show multimedia effect when playing audio (like those in Media Player)

AVSYNCHRONIZER_STATUSCODE_MOTION

Motion is triggered. The dwParam1 is a DWORD array of 3 percentages, and dwParam2 is a DWORD array of 3 alert flags.

AVSYNCHRONIZER_NEW_IMAGE

Notify application that a new image is to be shown. The dwParam1 is a Boolean that records if the video contains signal or not.

AVSYNCHRONIZER_FIRST_IMAGE

Called back when a new I frame comes. It will be called only if the SETCH_RESUME_ PLAYING is set for a live channel.

AVSYNCHRONIZER_STATUSCODE_VIDEO_MODE

The status of current video out mode. param1 indicates the display mode. Please refer to EAVSYNCVIDEOMODE.

Remarks

Requirements

AvSynchronizer.h

EDIGITALZOOMFLAG

This enumeration specified the valid members in structure of TDIGITALZOOMPARAM structure parameters used when calling AvSynchronizer_SetChannelOption.

```
typedef enum {
  DZ_CANVAS                 = 0x00000001,
  DZ_CANVAS_BORDER          = 0x00000002,
  DZ_ZOOM_AREA              = 0x00000004,
  DZ_ZOOM_AREA_             = 0x00000008,
  BORDER                    = 0x00000010,
  DZ_ENABLE_ZOOM            = 0x00000020,
  DZ_CANVAS_DISPLAY
} EDIGITALZOOMFLAG;
```

Values

DZ_CANVAS

The bCanvasWin and unCanvas members are valid.

DZ_CANVAS_BORDER

The rectCanvasBorder member is valid.

DZ_ZOOM_AREA

The bZoomWin and unZoom members are valid.

DZ_ZOOM_AREA_BORDER

The rectZoomBorder member is valid.

DZ_ENABLE_ZOOM

The bZoomEnabled member is valid.

DZ_CANVAS_DISPLAY

The bDisplayed member is valid.

Remarks

Requirements

szAvSynchronizer.h

EDISPLAYCAP

This enumeration lists the video capability for the display card. It is used in AvSynchronizer_GetCapabilities.

```
typedef enum {
  AVSYNCHRONIZER_VIDEOSF_OFFSCREEN_YUY2          = 1,
  AVSYNCHRONIZER_VIDEOSF_OFFSCREEN_UYVY          = (1 << 1),
  AVSYNCHRONIZER_VIDEOSF_OFFSCREEN_RGB16         = (1 << 2),
  AVSYNCHRONIZER_VIDEOSF_OFFSCREEN_RGB24         = (1 << 3),
  AVSYNCHRONIZER_VIDEOSF_OFFSCREEN_RGB32         = (1 << 4),
  AVSYNCHRONIZER_VIDEOSF_OVERLAY_YUY2            = (1 << 5),
  AVSYNCHRONIZER_VIDEOSF_OVERLAY_UYVY            = (1 << 6),
  AVSYNCHRONIZER_VIDEOSF_OVERLAY_R5G6B5          = (1 << 7),
  AVSYNCHRONIZER_VIDEOSF_OVERLAY_RGB24           = (1 << 8),
  AVSYNCHRONIZER_VIDEOSF_OVERLAY_RGB32           = (1 << 9),
  AVSYNCHRONIZER_VIDEOCAB_BLTFOURCC              = (1 << 10),
  AVSYNCHRONIZER_VIDEOCAB_OVERLAYFOURCC          = (1 << 11),
  AVSYNCHRONIZER_VIDEOCAB_CKEY_DESTBLT           = (1 << 12),
  AVSYNCHRONIZER_VIDEOCAB_CKEY_DESTOVERLAY       = (1 << 13),
  AVSYNCHRONIZER_VIDEOCAB_CKEY_SRCBLT            = (1 << 14),
  AVSYNCHRONIZER_VIDEOCAB_CKEY_SRCOVERLAY        = (1 << 15),
  AVSYNCHRONIZER_VIDEOCAB_EMULATIONONLY          = (1 << 16),
  AVSYNCHRONIZER_VIDEOCAB_NO_HWSTRETCH           = (1 << 17),
  AVSYNCHRONIZER_VIDEOCAB_NO_                    = (1 << 18),
  OVERLAYHWSTRETCH                               = (1 << 19),
  AVSYNCHRONIZER_VIDEOCAP_NO_BLT_SYSMEM          = (1 << 20),
  AVSYNCHRONIZER_VIDEOCAP_NO_AGP                 = (1 << 21),
  AVSYNCHRONIZER_VIDEOSF_OVERLAY_R5G5B5          = (1 << 22),
  AVSYNCHRONIZER_VIDEOCAB_BLTSTRETCHY            = (1 << 23),
  AVSYNCHRONIZER_VIDEOCAB_BLTSTRETCHX            = (1 << 24),
  AVSYNCHRONIZER_VIDEOCAB_BLTSHRINKY             = (1 << 25),
  AVSYNCHRONIZER_VIDEOCAB_BLTSHRINKX             = (1 << 26),
  AVSYNCHRONIZER_VIDEOCAB_OVERLAYSTRETCHY        = (1 << 27),
  AVSYNCHRONIZER_VIDEOCAB_OVERLAYSTRETCHX        = (1 << 28),
  AVSYNCHRONIZER_VIDEOCAB_OVERLAYSHRINKY         = (1 << 29),
  AVSYNCHRONIZER_VIDEOCAB_OVERLAYSHRINKX
} EDISPLAYCAP;
```

Values

AVSYNCHRONIZER_VIDEOSF_OFFSCREEN_YUY2

Indicates that could create YUY2 off-screen surface.

AVSYNCHRONIZER_VIDEOSF_OFFSCREEN_UYVY

Indicates that could create UYVY off-screen surface

AVSYNCHRONIZER_VIDEOSF_OFFSCREEN_RGB16

Indicates that could create 16 bits RGB off-screen surface.

AVSYNCHRONIZER_VIDEOSF_OFFSCREEN_RGB24

Indicates that could create 24 bits RGB off-screen surface.

AVSYNCHRONIZER_VIDEOSF_OFFSCREEN_RGB32

Indicates that could create 32 bits RGB off-screen surface.

AVSYNCHRONIZER_VIDEOSF_OVERLAY_YUY2

Indicates that could create YUY2 overlay surface.

AVSYNCHRONIZER_VIDEOSF_OVERLAY_UYVY

Indicates that could create UYVY overlay surface.

AVSYNCHRONIZER_VIDEOSF_OVERLAY_R5G6B5

Indicates that could create 16 bits RGB (565) overlay surface.

AVSYNCHRONIZER_VIDEOSF_OVERLAY_R5G5B5

Indicates that could create 15 bits RGB (555) overlay surface.

AVSYNCHRONIZER_VIDEOSF_OVERLAY_RGB24

Indicates that could create 24 bits RGB overlay surface.

AVSYNCHRONIZER_VIDEOSF_OVERLAY_RGB32

Indicates that could create 32 bits RGB overlay surface.

AVSYNCHRONIZER_VIDEOCAB_BLTFOURCC

Indicates that display hardware is capable of color-space conversions during blit operations.

AVSYNCHRONIZER_VIDEOCAB_OVERLAYFOURCC

Indicates that overlay hardware is capable of color-space conversions during overlay operations.

AVSYNCHRONIZER_VIDEOCAB_CKEY_DESTBLT

Supports transparent blitting with a color key that identifies the replaceable bits of the destination surface for RGB colors.

AVSYNCHRONIZER_VIDEOCAB_CKEY_DESTOVERLAY

Supports overlaying with color keying of the replaceable bits of the destination surface being overlaid for RGB colors.

AVSYNCHRONIZER_VIDEOCAB_CKEY_SRCBLT

Supports transparent blitting using the color key for the source with this surface for RGB colors.

AVSYNCHRONIZER_VIDEOCAB_CKEY_SRCOVERLAY

Supports overlaying using the color key for the source with this overlay surface for RGB colors.

AVSYNCHRONIZER_VIDEOCAB_EMULATIONONLY

Indicates that there is no hardware support.

AVSYNCHRONIZER_VIDEOCAB_NO_HWSTRETCH

Indicates that display hardware is not capable of stretching during blit operations.

AVSYNCHRONIZER_VIDEOCAB_NO_OVERLAYHWSTRETCH
Indicates that overlay hardware is not capable of stretching.

AVSYNCHRONIZER_VIDEOCAB_NO_BLT_SYSTEM
Indicates that display hardware is not capable of blitting to or from system memory.

AVSYNCHRONIZER_VIDEOCAB_NO_AGP
Indicates that the display driver doesn't supports surfaces in non-local video memory.

AVSYNCHRONIZER_VIDEOCAB_BLTSTRETCHY
Supports arbitrary stretching of a surface along the y-axis (vertically).

AVSYNCHRONIZER_VIDEOCAB_BLTSTRETCHX
Supports arbitrary stretching of a surface along the x-axis (horizontally).

AVSYNCHRONIZER_VIDEOCAB_BLTSHRINKY
Supports arbitrary shrinking of a surface along the y-axis (vertically).

AVSYNCHRONIZER_VIDEOCAB_BLTSHRINKX
Supports arbitrary shrinking of a surface along the x-axis (horizontally).

AVSYNCHRONIZER_VIDEOCAB_OVERLAYSTRETCHY
Supports arbitrary stretching of a surface along the y-axis (vertically).

AVSYNCHRONIZER_VIDEOCAB_OVERLAYSTRETCHX
Supports arbitrary stretching of a surface along the x-axis (horizontally).

AVSYNCHRONIZER_VIDEOCAB_OVERLAYSHRINKY
Supports arbitrary shrinking of a surface along the y-axis (vertically).

AVSYNCHRONIZER_VIDEOCAB_OVERLAYSHRINKX
Supports arbitrary shrinking of a surface along the x-axis (horizontally).

Remarks

Requirements

AvSynchronizer.h

EDISPLAYINFOFLAG

This enumeration indicates the type of this callback. Used by LPDISPLAYCALLBACK

```
typedef enum {
  DIF_PIC_SIZE              = 1,
  DIF_CONNECTING            = 2,
  DIF_SHOW_BLANK            = 4
  DIF_SHOW_                 = 8
  AUDIOONLY                 = 16
  DIF_IGNORE_BORDER         = 32
  DIF_RESUMING              = 64
  DIF_TELLME_RECT
} EDISPLAYINFOFLAG;
```

Values

DIF_PIC_SIZE

If this flag is set, the caption you draw will be scaled when the picture scaled.

DIF_CONNECTING

If this flag is set, then you can show the texts or graphics that you want to show during the time connecting to server.

DIF_SHOW_BLANK

If AP does no specify the bitmap to be displayed before media data is input, it could ask AVSynchronizer to call back when the window needs to be redrawn.

DIF_SHOW_AUDIOONLY

Called back each second when the channel is playing with audio only data. In such case, AP could display anything it likes to be shown to let users know that it's now audio only mode. When return from the callback, if the AP needs only the border part, it could return AVSYNCHRONIZER_S_NO_CONTENT, AVSynchronizer will skip the area not belong to border.

DIF_IGNORE_BORDER

For audio only media type channel, the callback needs AP to fill data in the window, but the border is ignored. This flag just let AP knows that it should not draw the border in current time.

DIF_RESUMING

For RTSP channel, callback to let application shows resuming message. It will continue to call until I frame reaches.

DIF_TELLME_RECT

The callee must tell the callback about the rect and color valid for caption.

Remarks

Requirements

AvSynchronizer.h

EFRAMETYPE

This enumeration indicates the type of frame (audio / video).

```
typedef enum {
  AUDIO_FRAME              = 2,
  VIDEO_FRAME              = 1,
} EFRAMETYPE;
```

Values

AUDIO_FRAME

An Audio frame.

VIDEO_FARME

A Video frame.

Remarks

Requirements

AvSynchronizer.h

EGRAPHALIGN

This enumeration specified if the alignment for horizontal and vertical direction when displaying video if users don't want the video to be stretched. It's used when calling AvSynchronizer_SetChannelOption.

```
typedef enum {
  ALIGN_HORI_CENTER     =0x0,
  ALIGN_HORI_LEFT       =0x1,
  ALIGN_HORI_RIGHT      =0x2,
  ALIGN_VERT_CENTER     =0x0,
  ALIGN_VERT_TOP        =0x4,
  ALIGN_VERT_BOTTOM     =0x8,
} EGRAPHALIGN;
```

Values

ALIGN_HORI_CENTER

Align the video to center in horizontal direction. If the video width is smaller than the width of the display area, Users will see the left and right black (or any color set when calling AvSynchronizer_InitialEx) area. If the video width is larger, then only the center part (horizontally) of the video will be shown.

ALIGN_HORI_LEFT

Align the video to left in horizontal direction. No mater video is larger or smaller, the left side of the video is align with the left side of the display area.

ALIGN_HORI_RIGHT

Align the video to left in horizontal direction. No mater video is larger or smaller, the right side of the video is align with the right side of the display area.

ALIGN_VERT_CENTER

Align the video to center in vertical direction. If the video height is smaller than the height of the display area, Users will see the top and bottom black (or any color set when calling AvSynchronizer_InitialEx) area. If the video height is larger, then only the center part (vertically) of the video will be shown.

ALIGN_VERT_TOP

Align the video to top in vertical direction. No mater video is larger or smaller, the top side of the video is align with the top side of the display area.

ALIGN_VERT_BOTTOM

Align the video to top in vertical direction. No mater video is larger or smaller, the bottom side of the video is align with the bottom side of the display area.

Remarks

Requirements

AvSynchronizer.h

EMediaCodecType

This enumeration indicates the media codec type. Please refer to Data Packet Parser document for detail definition

EMEDIATYPE (TMediaType)

This enumeration indicates the Media type of the channel. Used by AvSynchronizer_UpdateChannelSettings and AvSynchronizer_UpdatePlaybackChannelSettings.

```
typedef enum {
  AVSYNCHRONIZER_MEDIATYPE_VIDEO_ONLY        = 0x0001,
  AVSYNCHRONIZER_MEDIATYPE_AUDIO_ONLY        = 0x0002,
  AVSYNCHRONIZER_MEDIATYPE_AUDIO_VIDEO       = 0x0003
} EMEDIATYPE;
```

Values

AVSYNCHRONIZER_MEDIATYPE_AUDIO_ONLY

The channel is audio only.

AVSYNCHRONIZER_MEDIATYPE_VIDEO_ONLY

The channel is video only.

AVSYNCHRONIZER_MEDIATYPE_AUDIO_VIDEO

The channel has both audio and video.

Remarks

Requirements

AvSynchronizer.h

EMOTIONALERTCOLOR

This enumeration indicates the color of the Motion detection rectangle

```
typedef enum {
  RECT_RED                  = 0,
  RECT_GREEN                = 1,
  RECT_BLUE                 = 2,
  RECT_GRAY                 = 3
} EMOTIONALERTCOLOR;
```

Values

RECT_RED

Indicate to draw red rectangle.

RECT_GREEN

Indicate to draw green rectangle.

RECT_BLUE

Indicate to draw blue rectangle.

RECT_GRAY

Indicate to draw gray rectangle.

Remarks

Besides these color, it is possible to use RGB() macro to define your own color, the color field must be set the MSB to 1 and set the low 3 bytes to hold the RGB value.

Requirements

AvSynchronizer.h

EPBCHFLAG

This enumeration indicates which field of the structure TPBCHOPTION is valid.

```
typedef enum {
  PBCH_STATUSCB                 = 0x00000001,
  PBCH_DISPLAYCB                = 0x00000002,
  PBCH_ALLCALLBACK              = 0x10000003,
  PBCH_STATUSCONTEXT            = 0x00000004,
  PBCH_DISPLAYCONTEXT           = 0x00000008,
  PBCH_ALLCONTEXT               = 0x2000000c,
  PBCH_VOLUME                   = 0x00000010,
  PBCH_LFBORDER                 = 0x00000020,
  PBCH_RTBORDER                 = 0x00000040,
  PBCH_TPBORDER                 = 0x00000080,
  PBCH_BTBORDER                 = 0x00000100,
  PBCH_ALLBORDER                = 0x000001e0,
  PBCH_BITMAP                   = 0x00000200,
  PBCH_MOTION                   = 0x00000400,
  PBCH_MOTIONRECT               = 0x00000800,
  PBCH_MOTIONRECTALERT          = 0x00001000,
  PBCH_VIDEOOUT                 = 0x00002000,
  PBCH_AUDIOOUT                 = 0x00004000,
  PBCH_NEXTFRAME                = 0x08000000,
  PBCH_INPUTCB                  = 0x10000000,
  PBCH_INPUTCONTEXT             = 0x20000000,
  PBCH_BLOCKING                 = 0x40000000,
  PBCH_ALL                      = 0xFFFFFFFF
} EPBCHFLAG;
```

Values

PBCH_STATUSCB

pfStatus field is valid.

PBCH_DISPLAYCB

pfDisplay field is valid.

PBCH_ALLCALLBACK

All the callback is valid. pfStaus, pfDisplay and pfFrameRequest.

PBCH_STATUSCONTEXT

dwStatusContext is valid.

PBCH_DISPLAYCONTEXT

dwDisplayContext is valid.

PBCH_ALLCONTEXT

All the context is valid. dwStatusContext, dwDisplayContext and dwFrameRequestContext.

PBCH_VOLUME
dwVolume is valid,

PBCH_LFBORDER
dwLeftBorderSize is valid.

PBCH_RTBORDER
dwRightBorderSize is valid.

PBCH_TPBORDER
dwTopBorderSize is valid.

PBCH_BTBORDER
dwBottomBorderSize is valid.

PBCH_ALLBORDER
All the Border is valid. dwLeftBorderSize, dwRightBorderSize, dwRightBorderSize and dwBottomBorderSize.

PBCH_BITMAP
hBMP and bSaveBmp are valid.

PBCH_MOTION
bMotionAlert is valid.

PBCH_MOTIONRECT
dwMotionRect is valid.

PBCH_MOTIONRECTALERT
dwMotionRectAlert is valid.

PBCH_VIDEOOUT
bVideoOut is valid.

PBCH_AUDIOOUT
bAudioOut is valid.

PBCH_NEXTFRAME
If this flag is set and in the pause mode, it will display frame by frame or it will return error.

PBCH_INPUTCB
pfFrameRequest is valid.

PBCH_INPUTCONTEXT
dwFrameRequestContext is valid.

PBCH_BLOCKING
bNonBlocking is valid.

PBCH_ALL
All fields are valid.

Remarks

Requirements

AvSynchronizer.h

EPIXELFORMAT

This enumeration indicates the output format. Used by AvSynchronizer_CreateDecoderChannel.

```
typedef enum {
  PF_YUY2                 = 1,
  PF_RGB16565             = 2,
  PF_RGB24                = 3,
  PF_RGB32                = 4,
  PF_BMP16565             = 5,
  PF_BMP24                = 6,
  PF_BMP32                = 7,
  PF_JPEG                 = 8,
  PF_RGB16555             = 9,
  PF_BMP16555             = 10,
  PF_YUV                  = 11
  PF_BGR16565             = 12
  PF_BGR24                = 13
  PF_BGR32                = 14
  PF_BGR16555             = 15
  PF_YV12                 = 16
} EPIXELFORMAT;
```

Values

PF_YUY2

Output YUY2 format.

PF_RGB16565

Output RGB16 format. This is raw data. The RGB mask for the data is R(5), G(6), B(5)

PF_RGB24

Output RGB24 format. This is raw data.

PF_RGB32

Output RGB32 format. This is raw data.

PF_BMP16565

Output 16 bit Bitmap file with header. The RGB mask for the data is R(5), G(6), B(5)

PF_BMP24

Output 24Bit Bitmap with file header.

PF_BMP32

Output 32Bit Bitmap with file header.

PF_JPEG

The JPEG frame data. The module will need to encode the decoded data into JPEG if the original video type is not JPEG, so the performance of this type would be worse than other format.

PF_RGB16565

Output RGB16 format. This is raw data. The RGB mask for the data is R(5), G(5), B(5)

PF_BMP16565

Output 16 bit Bitmap file with header. The RGB mask for the data is R(5), G(5), B(5)

PF_YUV

Output YUV (4:2:0) format. The Y data is from 0 ~ W*H - 1, U is from W*H ~ W*H * 1.25, and V is from W*H * 1.25 ~ W*H*1.5. Where W, H is the width and height of the frame.

PF_BGR16565

Output BGR16565 format.

PF_BGR24

Output BGR24 format.

PF_BGR32

Output BGR32 format.

PF_BGR16555

Output BGR16555 format.

PF_YV12

Output YV12 format.

Remarks

PF_ BGR16565, PF_ BGR24, PF_ BGR32, and PF_ BGR16555 are only available for Linux platform currently.

Requirements

AvSynchronizer.h

ESETCHOPTION

This enumeration specified the parameters used when calling AvSynchronizer_SetChannelOption.

```
typedef enum {
  SETCH_DEBLOCKING                    = 0x00000001,
  SETCH_BLANK_BORDER_CB               = 0x00000002,
  SETCH_GRAPH_STRETCH                 = 0x00000003,
  SETCH_GRAPH_ALIGN                   = 0x00000004,
  SETCH_NOTIFY_MOTION                 = 0x00000005,
  SETCH_RESERVED_HEADER               = 0x00000006,
  SETCH_LEFT_BORDER                   = 0x00000007,
  SETCH_RIGHT_BORDER                  = 0x00000008,
  SETCH_TOP_BORDER                    = 0x00000009,
  SETCH_BOTTOM_BORDER                 = 0x0000000A,
  SETCH_DIGITAL_ZOOM                  = 0x0000000B,
  SETCH_REDRAW_DIGITAL_ZOOM           = 0x0000000C,
  SETCH_MEDIA_TYPE                    = 0x0000000D,
  SETCH_OUTPUT_VIDEO_FRAME            = 0x0000000E,
  SETCH_OUTPUT_AUDIO_FRAME            = 0x0000000F,
  SETCH_DECODE_DATA_FORMAT            = 0x00000010,
  SETCH_DECODE_JPEG_QUALITY           = 0x00000011,
  SETCH_DECODE_CB                     = 0x00000012,
  SETCH_DECODE_VIDEO_BUFFER           = 0x00000013,
  SETCH_DECODE_AUDIO_BUFFER           = 0x00000014,
  SETCH_RESUME_PLAYING                = 0x00000015,
  SETCH_DECODE_CBEX                   = 0x00000016,
  SETCH_DISPLAY_PERIOD                = 0x00000017,
  SETCH_AVI_CAPTION_CB_CONFORM        = 0x00000018,
  SETCH_BITMAP_FILE                   = 0x00000019,
  SETCH_BITMAP_HANDLE                 = 0x0000001A,
  SETCH_CLEAR_QUEUE                   = 0x0000001B,
  SETCH_BACKWARD_I_MODE               = 0x0000001C,
  SETCH_FASTER_CAPTION_ONGRAPH        = 0x0000001D,
  SETCH_ROTATE_IMAGE                  = 0x0000001E,
  SETCH_EMPTY_QUEUE                   = 0x0000001F,
  SETCH_DISPLAY_REDRAW                = 0x00000020,
  SETCH_FRAME_BUFF_STRIDE             = 0x00000021,
  SETCH_GETAV_TIME                    = 0x00000022,
  SETCH_DEINTERLACE                   = 0x00000023,
  SETCH_VIDEO_FRAME_BEFORE_DISPLAY    = 0x00000024,
  SETCH_AV_NOT_SYNC                   = 0x00000025,
  SETCH_AVI_MAX_FILE_SIZE             = 0x00000026,
  SETCH_AVI_FILE_TIME_INTERVAL        = 0x00000027,
  SETCH_NOT_FORCE_SHOW_FRAME          = 0x00000028,
  SETCH_NOT_BLOCK_WHEN_QUEUE_FULL     = 0x00000029,
  SETCH_ANTI_TEARING                  = 0x0000002a
} ESETCHOPTION;
```


Values

SETCH_DEBLOCKING

Enable or disable the de-blocking capability of the decoder. Enabled de-blocking would

gain better quality but will suffer a much worse performance. When setting with this option, the param1 is the Boolean value for enabling (TRUE) or disabling (FALSE). The default value is FALSE.

SETCH_BLANK_BORDER_CB

Enable or disable the blank border callback. The param1 is Boolean value: TRUE to enable border callback, FALSE to disable the callback. This is for some applications that need to display their own pattern rather than bitmap before channel display video. The default is FALSE.

SETCH_GRAPH_STRETCH

Should the channel stretch video when displaying? The param1 is TRUE to stretch, FALSE not. The default value is TRUE. If it is set to FALSE, SETCH_GRAPH_ALIGN could be used to specify the alignment in both horizontal and vertical direction.

SETCH_GRAPH_ALIGN

Specify the alignment for horizontal and vertical direction. The param1 is the alignment for horizontal direction, and param2 is the alignment setting for vertical direction. The value range could be found in EGRAPHALIGN. The default alignment is both center in the tow direction.

SETCH_NOTIFY_MOTION

Specify if AVSynchronizer should notify AP that motion is triggered. The param1 is the Boolean value: TRUE to enable the notification, FALSE to disable the notification. Default value is FALSE.

SETCH_RESERVED_HEADER

Set the reserved header for decoder channel. This value is only used for notifying decoded video frame. The param1 is the wanted reserved size in bytes.

SETCH_LEFT_BORDER

Set the left border for playback channel or normal channel. The param1 is the wanted left border size in pixel.

SETCH_RIGHT_BORDER

Set the right border for playback channel or normal channel. The param1 is the wanted left border size in pixel.

SETCH_TOP_BORDER

Set the top border for playback channel or normal channel. The param1 is the wanted left border size in pixel.

SETCH_BOTTOM_BORDER

Set the bottom border for playback channel or normal channel. The param1 is the wanted left border size in pixel.

SETCH_DIGITAL_ZOOM

Set the digital zoom parameters for normal or playback channel. The param1 is a pointer for the TDIGITALZOOMPARAM structure and param2 is the flag to specify the valid members in the structure. Note: digital zoom is now valid for GDI mode only.

SETCH_REDRAW_DIGITAL_ZOOM

Force to redraw the digital editing window. The video for this channel is not redrawn.

The parameters are not used.

SETCH_MEDIA_TYPE

Update the media type of a channel dynamically. It is very important to change to correct type. For example, if the new type is changed to A/V but no audio is input, the video will be blocked there. The param1 is the new media type for the channel. It's only available for normal and playback channel.

SETCH_OUTPUT_VIDEO_FRAME

For decoder channel, it tells the decoder to decode the video only, but not output or callback the decoded video frame data. This is useful for fast forward because in such case, only some frames need to be shown but due to the nature of MPEG4, all frames must be encoded. For normal or playback channel, this is to set the output of video frame during displaying. Application could request to get the decoded video frame in any available format no matter GDI or DirectDraw are used. The output mode is disabled default for normal and playback channel. The param1 is a Boolean to specify if users want to output the frame.

SETCH_OUTPUT_AUDIO_FRAME

For decoder channel, it tells the decoder whether to decode audio data, if it is set to false, the audio input data will be skipped. For normal or playback channel, this is to set the output of audio frame during playing. The output mode is disabled default for normal and playback channel. The param1 is a Boolean to specify if users want to output the frame.

SETCH_DECODE_DATA_FORMAT

For decoder channel, it tells the decoder what video format to output, this value could be changed at runtime. The param1 is the new format. For normal and playback channel, this is to specify what format to output during playing. If the option is not set, no media will be called back.

SETCH_DECODE_JPEG_QUALITY

Update the jpeg encode quality factor, the value range is 1-125. The param1 is the new quality value. This option is applied to normal, playback, and decoder channel.

SETCH_DECODE_CB

Setup the decoder callback for normal and playback channel. If the callback is empty, no callback will be made during playing. The param1 is the context for the callback, and param2 is the function pointer for the callback.

SETCH_DECODE_VIDEO_BUFFER

Set up the video decoding buffer for normal and playback channel. The buffer is used when callback is made. If no buffer is specified, internal buffer will be used. The param1 is the size of the buffer and param2 is the starting address of the buffer. Note, if the given buffer is too small, no callback will be made. The buffer needed for each video format is different, but a 704x576x4 + proper reserved header (1024 for example) buffer size could fit most case. Note, if the graph is translate to Jpeg, the buffer size above might sometimes not enough due to complex graph content. But that's quite extreme case.

SETCH_DECODE_AUDIO_BUFFER

Set up the audio decoding buffer for normal and playback channel. The buffer is

used when callback is made. If no buffer is specified, internal buffer will be used. The param1 is the size of the buffer and param2 is the starting address of the buffer. Note, if the given buffer is too small, no callback will be made. The buffer needed for each audio depends on the audio codec the channel is using, currently the maximum possible size for one frame is 2048 bytes, some more space for packet header is needed (for example 1024 bytes).

SETCH_RESUME_PLAYING

Tell a channel to wait for vide I frame again. This is usually used in RTSP channel. When this option is set, the channel will wait for I frame and before that frame comes, all decoding and displaying stop. The param1 is the flag to be set. Set True to enable resuming and False to stop resuming state. It only applies to live channel

SETCH_DECODE_CBEX

Set up an extension version of decoded data callback. This callback could replace the normal version. It gives application more information about the format of the decoding data. And If application likes, it could let the module decode for more than one output type. To accomplish this, the application could return a resuming flag and the new wanted type. For example, if the application sets the decoded type to BMP24, and it also needs YUV. When callback function is invoked, it could return resuming flag on and set next decoding type to YUV. After YUV is called, return with resuming flag False to decode next frame. The param1 is the context for the callback, and param2 is the address of the callback. When this and the normal callback are both set, this one is taken. This option applies to live, playback, and decoder channel.

SETCH_DISPLAY_PERIOD

Set up the display period for live and playback channel. The period is counted by frame number. For example, if the period is set to 5, the channel will show 1 frame per 5 decoded frames. This could somehow decrease the system loading. The param1 is the period number to be set, 0 or 1 means to show every frame decoded. The param2 is a flag to indicate if the live/playback channels' decode data callbacks are affected by this number. If it is set to be True, the callback will also be called back per period number. If the param2 is false, the callback will be called for every frame.

SETCH_AVI_CAPTION_CB_CONFORM

To set if the DrawTime member of TDISPLAYINFO structure in AVI channel's LPDISPLAYCALLBACK contains a time_t value or a "struct tm" pointer. In version before 4.3.0.0, the data member contains a time_t value. And this is not the same as other channels. To maintain backward compatibility, if this option is not set, the member is still a time_t value. But if this option is set, the value will contain a pointer to "struct tm". The time_t value is moved to a new member of dwTimet. The param1 contains Boolean value. True means the DrawTime is a pointer to "struct tm". False means DrawTime is a time_t value. Only applies to AVI channel.

SETCH_BITMAP_FILE

Set the bitmap to be shown in a channel when off line. The bitmap is assigned by a file name. This module will load the bitmap from the file and maintain it automatically. Multiple channels will share the same bitmap if the content of the file is the same. This option and the next one are exclusive. And the update channel setting will control the display of this bitmap. The param1 contains the file name (For Windows CE, the file name is in Unicode. For other platform, the name is in char). Only applies to live and

playback channel.

SETCH_BITMAP_HANDLE

Update a channel's display bitmap handle. This option and the above one are exclusive. The module will keep the bitmap and maintain it automatically. This means, after the call, application is safe to free the bitmap. The param1 contains the handle to the bitmap (HBITMAP). Only applies to live and playback channel.

SETCH_CLEAR_QUEUE

Set the way to handle if the video queue is full. The param1 is the flag to indicate the way to handle. If param1 is 0, the setting is doing nothing. If param1 is 1, the queue will be cleared only when I frame reaches and queue is full. If param1 is 2, the queue will be cleared when it is full and the I frame flag will be reset. The channel will accept packet only when next I frame reaches. Only applies to normal channel.

SETCH_BACKWARD_I_MODE

In this mode, the channel will show each I frame for one second. If the format is non-JPEG, the frame will show in reserved sequence. If param1 is true, the backward mode will turn on.

SETCH_FASTER_CAPTION_ONGRAPH

For some cards, get direct draw dc and painting on graph would be slowing. Setting this flag, the module will use a faster way to handle caption on graph, but the AP must also prepare to handle this type of callback. The drawing area is maximum 640x480 in such mode, and AP must also specify the drawing area before return. To be transparent, mask color must be given.

SETCH_ROTATE_IMAGE

Setting the rotated degree to be 90, 180, or 270 degree in clockwise. This option only works for Windows CE, other platform does not support. param1 is the degree could be.

SETCH_EMPTY_QUEUE

Clear the queue currently playing (both audio and video), this is workable for AP that support switch to time point

SETCH_DISPLAY_REDRAW

Redraw the current channel.

SETCH_FRAME_BUFF_STRIDE

Set the stride of the buffer used for retrieving the decoded video frame, dwParam1 is width stride in pixel unit, not in bytes, only works for decoder channel. Set value to 0 would make the scan line continuous in memory

SETCH_GETAV_TIME

Get the current time (only second) of the audio and video. The time is the last video and audio frame been played. param1 is the address to hold video time, return 0 if audio only and param2 is the address to hold audio time, return 0 if video only.

SETCH_DEINTERLACE

Enable deinterlace, param1 is true for enable and false for disable

SETCH_VIDEO_FRAME_BEFORE_DISPLAY

Callback the decoded frame before display. AP can do some operations on this frame. param1 is the context of this callback function and param2 is the callbakc function pointer, LPOUTPUTVIDEOFRAMEBEFOREDISPLAY.

SETCH_AV_NOT_SYNC

If param1 is TRUE, A/V will not sync and go on their best speed

SETCH_AVI_MAX_FILE_SIZE

Set max avi file size, this is for avi channel. This setting should be called before AvSynchronizer_StartAVIChannel. param1 is the max file size of avi file. If the file size reaches the max value, AvSychronizer will generate another avi file whose name is to append a number(from 1, 2, 3,...) to current avi file name.

SETCH_AVI_FILE_TIME_INTERVAL

Set max avi file length, only for avi channel. This setting should be called before AvSynchronizer_StartAVIChannel. param1 is the time interval of avi file and its unit is minute. If the time length reaches the max value, AvSychronizer will generate another avi file whose name is to append a number(from 1, 2, 3,...) to current avi file name. param2 is TRUE means using time length to divide the avi file. And param2 is FALSE to disable this function

SETCH_NOT_FORCE_SHOW_FRAME

Playback channel will force the current video to show if the queue is full. param1 = TRUE to disable this behavior.

SETCH_NOT_BLOCK_WHEN_QUEUE_FULL

In Playback channel, if AP inputs packets to AvSynchronizer and the internal queue fulls, Playback channel will block the calling thread until internal queue has free space. User can turn off this mechanism by setting param1 to TRUE or turn on it again by setting param1 = FALSE. This mechanism turn on by default.

SETCH_ANTI_TEARING

Set the anti_tearing policy, please refer to EANTITEARING. The default value is ANTI_TEARING_DISABLED.

Remarks

Requirements

AvSynchronizer.h

ESNAPSHOTFLAG

This enumeration specified if capturing image in original video size. Used by AvSynchronizer_GetCurrentSnapShot.

```
typedef enum {
  VISIBLE_SIZE              = 0x0,
  ORIGINAL_SIZE             = 0x1,
  SOURCE_IMAGE              = 0x2,
  SPECIFY_FMT               = 0x4,
  GIVEN_BUFFER              = 0x8
} ESNAPSHOTFLAG;
```

Values

VISIBLE_SIZE

Capture the image in the visible size.

ORIGINAL_SIZE

Capture the image in the original video size.

SOURCE_IMAGE

Get the current source encoded image data. This is helpful when the source is JPEG and users want to get JPEG graph.

SPECIFY_FMT

Get the data according to specified, the high word of the flag is the format of the request 'fmt'. The available format is listed as EPIXELFORMAT

GIVEN_BUFFER

The buffer pDataStart of TSNAPSHOT is given by caller, in such case the dwdataSize must specify the buffer size. When return, dwdataSize will be modified to the real size of the buffer if caller needs to reuse the buffer, the buffer size value most be saved before calling this function

Remarks

Requirements

AvSynchronizer.h

ESTARTAVICHFLAG

This enumeration specified the flags in TAVICHOPTION2.

```
typedef enum {
  FIXED_VINFO_AT_FIRST_FRAME      = (1<<2),
  FIXED_AINFO_AT_FIRST_FRAME      = (1<<3),
} ESTARTAVICHFLAG;
```

Values

FIXED_VINFO_AT_FIRST_FRAME

This flag means that the video size of the output AVI file would be determined by the first of the video frame. All the subsequence frames would be stretched to match this size. If this flag is not set, users need to give the wanted video size when start the channel.

FIXED_AINFO_AT_FIRST_FRAME

This flag means that the audio sample frequency of the output AVI file would be determined by the first of the audio frame. All the subsequence frames would be transform to this frequency. If this flag is not set, users need to give the wanted audio sampling frequency when start the channel.

Remarks

This enumeration is only available for Windows 32 platform.

Requirements

AvSynchronizer.h

ESTARTCHANNELFLAG

This enumeration indicates if needed to show something during the time connecting to server. Used by AvSynchronizer_StartChannel

```
typedef enum {
  DRAW_CONNECTING            = 0x00000001,
} ESTARTCHANNELFLAG;
```

Values

DRAW_CONNECTING

Set this flag to notify that you want to draw something during the time connecting to the server.

Remarks

During the time connecting to the server, if DRAW_CONNECTING is set, then the LPDISPLAYCALLBACK will be called. You can show some texts or graphics by using the DC.

Requirements

AvSynchronizer.h

EUPCHOPTION

This enumeration indicates which field of the structure TUPDATECHANNELOPTION or TUPDATEPBCHANNELOPTION is valid.

```
typedef enum {
  UPCH_VIDEOOUT              = 0x00000001,
  UPCH_AUDIOOUT              = 0x00000002,
  UPCH_MOTIONALERT           = 0x00000004,
  UPCH_MOTIONRECT            = 0x00000008,
  UPCH_MOTIONRECTALERT       = 0x00000010,
  UPCH_VOLUME                = 0x00000020,
  UPCH_MEDIATYPE             = 0x00000040,
  UPCH_TIMEITV_CHG           = 0x00000080,
  UPCH_SETSPEED              = 0x00000100,
  UPCH_PAUSE                 = 0x00000200,
  UPCH_SHOW_BMP              = 0x00000400,
  UPCH_WIN_HANDLE            = 0x00000800,
  UPCH_RESTORE_WINPROC       = 0x00001000,
  UPCH_IGNORE_BORDER         = 0x00002000,
  UPCH_ALL                   = 0xFFFFFFFF
} EUPCHOPTION;
```

Values

UPCH_VIDEOOUT
bVideoOut is valid.

UPCH_AUDIOOUT
bAudioOut is valid.

UPCH_MOTIONALERT
bMotionAlert is valid.

UPCH_MOTIONRECT
dwMotionRect is valid.

UPCH_MOTIONRECTALERT
dwMotionRectAlert is valid.

UPCH_VOLUME
dwVolume is valid.

UPCH_MEDIATYPE
tMediaType is valid. This flag is only for TUPDATEPBCHANNELOPTION structure.

UPCH_TIMEITV_CHG
Set this flag to indicate AvSynchronizer object that time interval change.

UPCH_SETSPEED
dwFast and dwSlow is valid.

UPCH_PAUSE

bPasue is valid.

UPCH_SHOW_BMP

Set this flag to show the specified Bitmap on the channel.

UPCH_WIN_HANDLE

Change the display window handle of a channel.

UPCH_RESTORE_WINPROC

Restore the window message handling proc. Must update this before set the channel a new display window handle.

UPCH_IGNORE_BORDER

Do not show the border when update graph. The video frame will fill the whole window when this is set.

UPCH_ALL

All fields are valid.

Remarks

Requirements

AvSynchronizer.h

TsdrAudioCodec

This enumeration indicates the audio codec types.

```
typedef enum {
  eACodecNone              = 0x0000,
  eACodecG7221             = 0x0100,
  eACodecG729A             = 0x0200
  eACodecAAC               = 0x0400
  eACodeGAMR               =0x0800
} TsdrAudioCodec;
```

Values

eACodecNone

no audio codec

eACodecG7221

G.722.1

eACodecG729A

G.729A

eACodecAAC

AAC (stereo)

eACodecGAMR

GAMR, used in RTSP IP camera and server

Remarks

Requirements

SrvTypeDef.h

TsdrVideoCodec

This enumeration indicates the video codec types.

```
typedef enum {
  eVCodecNone              = 0x0000,
  eVCodecMJPEG             = 0x0001,
  eVCodecH263              = 0x0002,
  eVCodecMPEG4             = 0x0004
} TsdrVideoCodec;
```

Values

eVCodecNone

no video codec

eVCodecMJPEG

Motion JPEG

eVCodecH263

H.263

eVCodecMPEG4

MPEG-4

Remarks

Requirements

SrvTypeDef.h

4.2. Callback Function

The Callback function is depicted here.

LPDECODEFRAMECALLBACK

This callback function is used to notify the application the decoded audio or video frame. The timestamp for the original frame is reserved in the structure. This callback is used in decoder, normal and playback channel.

Syntax

```
typedef SCODE
(__stdcall * LPDECODEFRAMECALLBACK)(    DWORD dwContext,
                                        TFrameType tFrametype,
                                        TFRAMEINFO* ptFrameInfo     );
```

Parameters

dwContext

[in] Specify the application-defined value that is passed, along with the returned handle, to this function.

tFrametype

[in] indicate this frame is audio or video.

ptFrameInfo

[in] according to the audio frame or video frame pointer to the different structure.

Return Values

Returns S_OK if success, or an error value otherwise.

Remarks

The buffer will be used for further decoding, so when called back, application needs to copy the frame to its own buffer for future use.

Requirements

AvSynchronizer.h

LPDECODEFRAMECALLBACK_EX

This extended callback function is used to notify the application the decoded audio or video frame. The timestamp for the original frame is reserved in the structure. This callback is used in decoder, normal and playback channel. It is to be used to replace LPDECODEFRAMECALLBACK

Syntax

```
typedef SCODE
(__stdcall *  PDECODEFRAMECALLBACK_EX)(   DWORD dwContext,
                                          TFrameType tFrametype,
                                          TFRAMEINFOEX* ptFrameInfoEx   );
```

Parameters

dwContext

[in] Specify the application-defined value that is passed, along with the returned handle, to this function.

tFrametype

[in] indicate this frame is audio or video.

ptFrameInfoEx

[in] according to the audio frame or video frame pointer to the different structure.

Return Values

Returns S_OK if success, or an error value otherwise.

Remarks

The buffer will be used for further decoding, so when called back, application needs to copy the frame to its own buffer for future use.

Requirements

AvSynchronizer.h

LPDISPLAYCALLBACK

This callback function is used to notify the application before displaying.

Syntax

```
typedef SCODE
(__stdcall * LPDISPLAYCALLBACK)(        DWORD dwContext,
                                        TDISPLAYINFO * ptDisplayInfos     );
```

Parameters

dwContext

[in] Specify the application-defined value that is passed, along with the returned handle, to this function.

ptDisplayInfos

[in] address of the structure contains some information. See

Return Values

Returns S_OK if success, or an error value otherwise.

Remarks

Because the callback is called from internal thread, it is not proper to call to many UI functions in this callback. To many UI calls might cause un-predicted result.

Requirements

AvSynchronizer.h

LPOUTPUTVIDEOFRAMEBEFOREDISPLAY

This callback function is used to callback the decoded frame before display..

Syntax

```
typedef SCODE
(__stdcall * LPOUTPUTVIDEOFRAMEBEFOREDISPLAY)(   DWORD dwContext,
                                                 TVIDEOINFO * ptVideoInfo   );
```

Parameters

dwContext

[in] Specify the application-defined value that is passed, along with the returned handle, to this function.

ptVideoInfo

[in] address of TVIDEOINFO

Return Values

Returns S_OK if success, or an error value otherwise.

Remarks

Because the callback is called from internal thread, it is not proper to call to many UI functions in this callback. To many UI calls might cause un-predicted result.

Requirements

AvSynchronizer.h

LPINPUTFRAMEREQUESTCALLBACK

Reserved. This callback function is not implemented yet.

Syntax

```
typedef SCODE
(__stdcall * LPINPUTFRAMEREQUESTCALLBACK)(   DWORD dwContext,
                                             HANDLE hChannel      );
```

Parameters

dwContext

[in] Specify the application-defined value that is passed, along with the returned handle, to this function.

hChannel

[in] the Handle of the channel.

Return Values

Return NO_MORE_DATA if the channel has no more data to be played, return S_OK otherwise.

Remarks

This function is used for nonblocking mode. The application should call the AvSynchronizer_InputPlaybackMediaFrame to push the packet into internal queue.

Requirements

AvSynchronizer.h

LPSTATUSCALLBACK

This callback function is used to notify the application the status of the AvSynchronizer.

Syntax

```
typedef SCODE
(__stdcall * LPSTATUSCALLBACK)(     DWORD dwContext,
                                    DWORD dwStatusCode,
                                    DWORD dwParam1,
                                    DWORD dwParam2          );
```

Parameters

dwContext

[in] Specify the application-defined value that is passed, along with the returned handle, to this function.

dwStatusCode

[in] This is the status code that indicates why the callback function is being called. This parameter can be one of the values listed in ECHANNELSTATUS.

dwParam1

[in] additional status information.

dwParam2

[in] additional status information.

Return Values

Returns S_OK if success, or an error value otherwise.

Remarks

Because the callback is called from internal thread, it is not proper to call to many UI functions in this callback. To many UI calls might cause un-predicted result.

Requirements

AvSynchronizer.h

4.3. Data Structure

The data structure is depicted here.

TAFRAMEINFO

This structure is used by LPDECODEFRAMECALLBACK function.

```
typedef struct {
  DWORD                    dwTime;
  DWORD                    dwSize;
  BYTE *                   pbySound;
  DWORD                    dwTimeStampSec;
  DWORD                    dwTimeStampMiliSec;
  TMediaDataPacketInfo     tPacketInfo;
} TAFRAMEINFO, * PTAFRAMEINFO;
```

Members

dwTime

Specify total time of the frame in millisecond. This time is the length of time that needs to play this piece of sound data. Currently, for eACodecG7221 the value is 20ms, and for eACodecG729A the value is 10ms

dwSize

Specify the size of the frame in bytes.

pbySound

a pointer that store the data of the frame.

dwTimeStampSec

second of the time stamp for this audio piece.

dwTimeStampMilliSec

millisecond of the time stamp for this audio piece. With dwTimeStampSec

tPacketInfo

The packet info copied from encoded audio frame. Only those fields related to audio are valid.

Remarks

Requirements

Avsynchronizer.h

TAUDIOOPTION

This structure is an alias for WAVEFORMATEX structure in Win32 SDK. Please refer to the definition in related document.

Remarks

This structure is only available for Windows 32 / Windows CE platform currently.

Requirements

Mmreg.h

TAVIAINFO

This structure is used in TAVICHOPTION2

```
typedef struct {
  DWORD                 dwChannels;
  DWORD                 dwSamplesPerSec;
  DWORD                 dwBitsPerSample;
} TAVIAINFO;
```

Members

dwChannels

Specify the number of audio channel in bit stream. Except AAC, all other audio data should specify 1 now. Assign 2 for AAC.

dwSamplesPerSec

Specify the sample per second for the audio bit stream. Currently only 8000 and 16000 are supported.

dwBitsPerSample

Specify the bits per sample. Currently, please assign it to 16.

Remarks

This structure is only available for Windows 32 platform currently.

Requirements

Avsynchronizer.h

TAVICHOPTION

This structure is used by AvSynchronizer_CreateAVIChannel function. It specifies the callback functions for the channel.

```
typedef struct {
  LPSTATUSCALLBACK          pfStatus;
  LPDISPLAYCALLBACK         pfCaption;
  DWORD                     dwStatusContext;
  DWORD                     dwCaptionContext;
  DWORD                     dwBitCount;
} TAVICHOPTION;
```

Members

pfStatus

This is the callback to notify current status of the conversion. Usually, it will be used to update the progress indicator.

pfCaption

Use to let application draw the caption and border. If application does not need that, set this to NULL.

dwStatusContext

This is the context associated with the status callback.

dwCaptionContext

This is the context associated with the caption callback.

dwBitCount

Reserved. It's not used in current version.

Remarks

This structure is only available for Windows 32 platform currently.

Requirements

Avsynchronizer.h

TAVICHOPTION2

This structure is used for AvSynchronizer_StartAVIChannel function.

```
typedef struct {
  TAVIVINFO                 tAviVInfo;
  TAVIAINFO                 tAviAInfo;
  DWORD                     dwFrameRate;
  BOOL                      bAudioEnable;
  BOOL                      bVideoEnable;
  char                      szFileName[MAX_PATH];
  TBORDERSIZE               tBorder;
  DWORD                     dwFlag;
} TAVICHOPTION2;
```

Members

tAviVInfo

Specify the settings for video stream in AVI file.

tAviAInfo

Specify the settings for audio stream in AVI file.

dwFrameRate

Specify the frame rate for video. This is the wanted frame rate, the channel will adjust the frame rate from the input video stream to conform to the wanted frame rate.

bAudioEnable

Specify whether the channel should generate the audio stream in AVI file.

bVideoEnable

Specify whether the channel should generate the video stream in AVI file.

szFileName

Specify the name of the AVI file. The content of the file will be destroyed when generating AVI file.

tBorder

Specify the border settings for this channel. If no border is needed, specify 0 for all the member.

dwFlag

Specify the extra setting for this channel. The possible values are defined in ESTARTAVICHFLAG.

Remarks

This structure is only available for Windows 32 platform currently.

Requirements

Avsynchronizer.h

TAVIVINFO

This structure is used in TAVICHOPTION2.

```
typedef struct {
  DWORD                 dwWidth;
  DWORD                 dwHeight;
  DWORD                 dwBitCount;
} TAVIVINFO;
```

Members

dwWidth

Specify the width of the video in AVI file. This member could be 0 if the FIXED_VINFO_AT_FIRST_FRAME flag is set in TAVICHOPTION2 structure.

dwHeight

Specify the height of the video in AVI file. This member could be 0 if the FIXED_VINFO_AT_FIRST_FRAME flag is set in TAVICHOPTION2 structure.

dwBitCount

Specify the bitcount of the output file. This value should be always set to 24 now.

Remarks

This structure is only available for Windows 32 platform currently.

Requirements

Avsynchronizer.h

TAVSYNCQUEUESTATUS

This structure is used when calling AvSynchronizer_GetRemainingQueueElementCount.

```
typedef struct {
  DWORD                    dwAudioQueueElements;
  DWORD                    dwVideoQueueElements;
  DWORD                    dwAudioQueueSize;
  DWORD                    dwVideoQueueSize
} TAVIVINFO;
```

Members

dwAudioQueueElements

The remaining queue elements in audio buffer queue

dwVideoQueueElements

The remaining queue elements in video buffer queue

dwAudioQueueSize

The size of audio buffer queue

dwVideoQueueSize

the size of video buffer queue

Remarks

There are two kinds of queues in AvSynchronizer. One is buffer queue and the other is data queue. Buffer queue is the free queue that can be used when user input media frame. If dwAudioQueueElements/dwVideoQueueElements is zero, AP will get queue full message when inputing media frame.

This structure is only available for Windows 32 platform currently.

Requirements

Avsynchronizer.h

TBORDERSIZE

This structure specifies the border setting for the video in an AVI channel.

```
typedef struct {
  DWORD                 dwHeader;
  DWORD                 dwFooter;
  DWORD                 dwLeft;
  DWORD                 dwRight;
} TBORDERSIZE;
```

Members

dwHeader

The height of the header

dwFooter

The height for footer

dwLeft

Specify the width of the left border.

dwRight

Specify the width of right border.

Remarks

This structure is only available for Windows 32 / Windows CE platform currently.

Requirements

Avsynchronizer.h

TCHANNELOPTION

This structure collects the settings of the channel. When creating channel, fill this structure to setup the channel.

```
typedef struct {
  DWORD                 dwFlags;
  LPSTATUSCALLBACK      pfStatus;
  LPDISPLAYCALLBACK     pfDisplay;
  DWORD                 dwStatusContext;
  DWORD                 dwDisplayContext
  DWORD                 dwVolume;
  DWORD                 dwLeftBorderSize;
  DWORD                 dwRightBorderSize;
  DWORD                 dwTopBorderSize;
  DWORD                 dwBottomBorderSize;
  HBITMAP               hBMP;
  HWND                  hVideoOut;
  HWND                  hDisplay;
  BOOL                  bSaveBmp;
  BOOL                  bVideoOut;
  BOOL                  bAudioOut;
  BOOL                  bMotionAlert;
  DWORD                 dwMotionRectAler
  DWORD                 dwMotionRect;
  BOOL                  bOrgSizeCallback;
  DWORD                 dwOrgSizeLeftBorder;
  DWORD                 dwOrgSizeTopBorder;
  DWORD                 dwOrgSizeRightBorder;
  DWORD                 dwOrgSizeBottomBorder;
} TCHANNELOPTION, *PTCHANNELOPTION;
```


Members

dwFlags

Indicate which field of this structure is valid. It can be one or the combination of the values of ECHFLAG.

pfStatus

Specify the pointer to the Status callback function. To be used to notify the application the status of AvSynchronizer.

pfDisplay

Specify the pointer to the display callback function. To be used to notify the application to display something.

dwStatusContext

Specify the application-defined value that is passed, along with the returned handle, to status callback functions.

dwDisplayContext

Specify the application-defined value that is passed, along with the returned handle, to

display callback functions.

dwVolume

Specify the audio volume. The valid value is 0 ~ 5000. 5000 is the loudest.

dwLeftBorderSize

Specify the size of the left border.

dwRightBorderSize

Specify the size of the right border.

dwTopBorderSize

Specify the size of the top border.

dwBottomBorderSize

Specify the size of the bottom border.

hBMP

A handle of a bitmap.

hVideoOut

Window handle used for the application. Set to the calling application's top-level window handle (not a handle for any child windows created by the top-level window).

hDisplay

The handle of the window which the video of this channel will show on it.

bSaveBmp

Indicate that if the AvSynchronizer module needs to keep the bitmap. Application could be safely delete the bitmap right after create the channel if this flag is specified.

bVideoOut

Indicate if show the video of this channel.

bAudioOut

Indicate if this channel audible.

bMotionAlert

Indicate if drawing the rectangle of the motion detection window.

dwMotionRectAlert

Specify the color of the rectangle of the motion detection window when motion alert happened. The value is one of the values in EMOTIONALERTCOLOR enumeration. If you want to use the color that are not list in the EMOTIONALERTCOLOR, then you can set the MSB to be 1 and the rest of the DWORD is the RGB value.

dwMotionRect

Specify the color of the rectangle of the motion detection window. The value is one of the values listed in EMOTIONALERTCOLOR enumeration. If you don't want to show the rectangle of the motion detection window when motion alert didn't happen, set 0xFFFFFFFF as it's value. If you want to use the color that are not list in the EMOTIONALERTCOLOR, then you can set the MSB to be 1 and the rest of the DWORD is the RGB value.

bOrgSizeCallback

If this is TRUE, this module will use the real size of the video plus the border size you specified below to draw the video and caption. The generated graph will be scaled onto the display window. If False, only the video will be scaled, and the border will be the same as you specified when shown. Note: To set up this value, the ORG_SIZE_ MODE must be turned on when initializing AVSynchronizer.

dwOrgSizeLeftBorder

Specify the size of the left border. This is valid only when bOrgSizeCallback is TRUE.

dwOrgSizeTopBorder

Specify the size of the top border. This is valid only when bOrgSizeCallback is TRUE.

dwOrgSizeRightBorder

Specify the size of the right border. This is valid only when bOrgSizeCallback is TRUE.

dwOrgSizeBottomBorder

Specify the size of the bottom border. This is valid only when bOrgSizeCallback is TRUE.

Remarks

hBMP is only available on Windows 32 platform.

Requirements

Avsynchronizer.h

TDECCHOPTION

This structure is used by AvSynchronizer_CreateDecoderChannel function.

```
typedef struct {
  LPDECODEFRAMECALLBACK          pfDecodeFrame;
  DWORD                          dwAvDecodeContext;
  DWORD                          dwRawDataFormat;
  DWORD                          dwJpegQualityFactor;
} TDECCHOPTION, * PTDECCHOPTION;
```

Members

pfDecodeFrame

Specify the pointer to the audio/video output callback function.

dwAvDecodeContext

Specify the application-defined value that is passed, along with the returned handle, to audio/video output callback functions.

dwRawDataFormat

Indicate the format of the output data. The value is define in EPIXELFORMAT.

dwJpegQualityFactor

The encoding quality for JPEG output frame. The range is 1-125, the smaller the better quality, but would have worse performance because it requires more time to generate the JPEG frame.

Remarks

Requirements

Avsynchronizer.h

TCURRENTDISPLAYINFO

This structure is used by AvSynchronizer_GetCurrentDisplay function.

```
typedef struct {
  DWORD                                 dwDisplayWidth;
  DWORD                                 dwDisplayHeight;
  DWORD                                 dwTimet;
  DWORD                                 dwFlag;
} TCURRENTDISPLAYINFO, *
PTCURRENTDISPLAYINFO;
```

Members

dwDisplayWidth

Indicate the display width .

dwDisplayHeight

Indicate the display height.

dwTimet

This is the time_t value for the DrawTime.

dwFlag

The flag tells if the channel is resuming or audio only. .

Remarks

Requirements

Avsynchronizer.h

TDIGITALZOOMPARAM

This structure is used in AvSynchronizer_SetChannelOption for the option SETCH_DIGITAL_ZOOM.

```
typedef struct {
  BOOL                  bCanvasWin;
  union
  {
    HWND                hTarget;
    RECT                rectTarget;
  }                     unCanvas;
  RECT                  rectCanvasBorder;
  BOOL                  bZoomWin;
  union
  {
    HWND                hZoomArea;
    RECT                rectZoomArea;
  }                     unZoom;
  RECT                  rectZoomBorder;
  BOOL                  bZoomEnabled;
  BOOL                  bDisplayed;
} TDISPLAYINFO, *
PDISPLAYINFO;
```

Members

bCanvasWin

Determine which element in unCanvas is correct. If the value is True hTarget is vlaid, else rectTarget is valid.

hTarget

The canvas window that will hold the full image, users could set the zoom area in this canvas so that the video for the channel will be zoomed image for the selected part. The drawing will be stuck to the window, so the image will be moved when window moved.

rectTarget

This is the rectangle to draw the full image. The coordinate should be related to the upper-left corner of screen. The application should be take in charge of the moving of the drawing area when window move. Or the image might be drawn outside the application window.

rectCanvasBorder

This is the border size for the canvas. The image will be drawn inside the window or rectangle with left, top, right and bottom offset recorded in this structure. The default is 0.

bZoomWin

Determine which element in unZoom is correct. If the value is True hZoomArea is vlaid, else rectZoomAreais valid.

hZoomArea

The child window of canvas window that shows the selected zoomed area. With this window, users could change the position or resize the zoom area.

rectZoomArea

This is the rectangle for zoomed area. The coordinate is related to upper-left corner of screen. The rectangle must be within the rectTarget or the rectangle of hTarget or the zooming will not be taken effect.

rectZoomBorder

This is the border size for the zoom area. The zoom area will be actually subtracted with left, top, right and bottom offset recorded in this structure. The default is 0.

bZoomEnabled

Is the digital zoom turned on for this channel? If the value is false, any previous setup value will be kept, but the channel will be shown with full image. This value provides a quick way to turn off digital zoom.

bDisplayed

Should the module try to paint the image in canvas area? Because painting will consume some CPU power, it is recommended to design a UI that will normally hide the canvas and zoom area window. Whenever the windows are shown, turn this value on, and turn it off right after the windows hide.

Remarks

Requirements

Avsynchronizer.h

TDISPLAYINFO

This structure is used by LPDISPLAYCALLBACK function.

```
typedef struct {
  HDC                                hDC;
  POINT                              ptOrigin;
  DWORD                              dwWidth;
  DWORD                              dwHeight;
  DWORD                              DrawTime;
  DWORD                              dwCallTimes;
  DWORD                              dwFlag;
  DWORD                              dwTimet;
  BOOL                               bDrawOneMem;
  DWORD                              dwActiveBlock;
  TONEDISPBLOCK                      atBlocks[MAX_DISP_BLOCK];
} TDISPLAYINFO, * PDISPLAYINFO;
```

Members

hDC

The hDC to draw with.

ptOrigin

The Original point of the draw able area.

dwWidth

Specify the width of the draw able area.

dwHeight

Specify the height of the draw able area.

DrawTime

The address of the struct tm. Contains the time of the frame. If for AVI channel, and the SETCH_AVI_CAPTION_CB_CONFORM is not set, this value is the time_t value. Caller could use gmtime to convert it to the struct tm pointer

dwCallTimes

Specify how many times this callback was called. This is only valid for connecting mode.

dwFlag

Specify the mode this callback is. See EDISPLAYINFOFLAG.

dwTimet

This is the time_t value for the DrawTime.

bDrawOneMem

If this variable is true, the following

dwActiveBlock

Callee must fill this variable to tell the caller how many blocks are correct.

TFRAMEINFO

This structure is used by LPDECODEFRAMECALLBACK function.

```
typedef struct {
  union {
  TVFRAMEINFO                    tVideoFrame;
  TAFRAMEINFO                    tAudioFrame;
  }
} TVFRAMEINFO, * PVTFRAMEINFO;
```

Members

tVideoFrame

See TVFRAMEINFO.

tAudioFrame

See TAFRAMEINFO.

Remarks

Requirements

Avsynchronizer.h

TFRAMEINFOEX

This structure is used by LPDECODEFRAMECALLBACK_EX function. For audio data, all other member except tF is effective when this callback is called.

```
typedef struct {
  TFRAMEINFO                    tF,
  DWORD                         dwCBCount;
  DWORD                         dwVideoFormat;
  DWORD                         dwNextFormat;
  BOOL                          bContinueDecode;
  BYTE                          abyReserved[64];
} TVFRAMEINFO, * PVTFRAMEINFO;
```

Members

tF

This is the TFRAMEINFO structure for the video or audio frame.

dwCBCount

This is the count for the call times of a single frame. It starts from 1. It's only usable for video frame.

dwVideoFormat

The video format of current called back data. It's only usable for video frame.

dwNextFormat

This is usable when application needs more decoded data format for the same frame. And it is only used if the bContinueDecode is set to be true. It's only usable for video frame.

bContinueDecode

The value is set to false when called back. If the application needs more data format of the same frame, set this parameter to true and set the next wanted format in dwNextFormat to ask the module to send out other format. It's only usable for video frame.

abyReserved

These bytes are reserved for future used.

Remarks

Requirements

Avsynchronizer.h

TMediaDataPacketInfo

This structure is used by AvSynchronizer_InputEncodedMediaFrame and AvSynchronizer_InputPlaybackMediaFrame function. Please refer to VNDP_Data_Packet_Parser.doc for the definition of this data structure

Remarks

Requirements

datapacketdef.h

TONEDISPBLOCK

This structure stores the information of a display block

```
typedef struct {
  SIZE                    szSrc;
  POINT                   ptDest;
  COLORREF                clrMask;
} TONEDISPBLOCK;
```

Members

szSrc

This indicates the height and width of a rectangle.

ptDest

The destination point (x, y) to paint the text and its size is the same as szSrc.

clrMask

The mask of color that should be treated as transparent.

Remarks

Requirements

Avsynchronizer.h

TPBCHOPTION

This structure collects the settings of the playback channel. When creating playback channel, fill this structure to setup the channel.

```
typedef struct {
  DWORD                          dwFlags;
  LPSTATUSCALLBACK               pfStatus;
  LPDISPLAYCALLBACK              pfDisplay;
  DWORD                          dwStatusContext;
  DWORD                          dwDisplayContext;
  DWORD                          dwVolume;
  DWORD                          dwLeftBorderSize;
  DWORD                          dwRightBorderSize;
  DWORD                          dwTopBorderSize;
  DWORD                          dwBottomBorderSize;
  HBITMAP                        hBMP;
  HWND                           hVideoOut;
  HWND                           hDisplay;
  BOOL                           bSaveBmp;
  BOOL                           bVideoOut;
  BOOL                           bAudioOut;
  BOOL                           bMotionAlert;
  DWORD                          dwMotionRectAlert;
  DWORD                          dwMotionRect;
  BOOL                           bOrgSizeCallback;
  DWORD                          dwOrgSizeLeftBorder;
  DWORD                          dwOrgSizeTopBorder;
  DWORD                          dwOrgSizeRightBorder;
  DWORD                          dwOrgSizeBottomBorder;
  LPINPUTFRAMEREQUESTCALLBACK    pfFrameRequest;
  BOOL                           bNonBlocking;
  DWORD                          dwFrameRequestContext;
} TPBCHOPTION, *PTPBCHOPTION;
```

Members

dwFlags

Indicate which field of this structure is valid. It can be one or the combination of the values of EPBCHFLAG.

pfStatus

Specify the pointer to the Status callback function. To be used to notify the application the status of AvSynchronizer.

pfDisplay

Specify the pointer to the display callback function. To be used to notify the application to display something.

dwStatusContext

Specify the application-defined value that is passed, along with the returned handle, to status callback functions.

dwDisplayContext

Specify the application-defined value that is passed, along with the returned handle, to display callback functions.

dwVolume

Specify the audio volume. The valid value is 0 ~ 5000.

dwLeftBorderSize

Specify the size of the left border.

dwRightBorderSize

Specify the size of the right border.

dwTopBorderSize

Specify the size of the top border.

dwBottomBorderSize

Specify the size of the bottom border.

hBMP

A handle of a bitmap.

hVideoOut

Window handle used for the application. Set to the calling application's top-level window handle (not a handle for any child windows created by the top-level window).

hDisplay

Window handle of the window which the video of this channel will show on it.

bSaveBmp

This value is ignored now, all bitmap will be parsed and saved internally.

bVideoOut

Indicate if show the video of this channel.

bAudioOut

Indicate if this channel audible.

bMotionAlert

Indicate if drawing the rectangle of the motion detection window.

dwMotionRectAlert

Specify the color of the rectangle of the motion detection window when motion alert happened. The value is one of the values in EMOTIONALERTCOLOR enumeration. If you want to use the color that are not list in the EMOTIONALERTCOLOR, then you can set the MSB to be 1 and the rest of the DWORD is the RGB value.

dwMotionRect

Specify the color of the rectangle of the motion detection window. The value is one of the values in EMOTIONALERTCOLOR enumeration. If you don't want to show the rectangle of the motion detection window when motion alert didn't happen, set 0xFFFFFFF as it's value. If you want to use the color that are not list in the EMOTIONALERTCOLOR, then you can set the MSB to be 1 and the rest of the

DWORD is the RGB value.

bOrgSizeCallback

If this is TRUE, this module will use the real size of the video plus the border size you specified below to draw the video and caption. The generated graph will be scaled onto the display window. If False, only the video will be scaled, and the border will be the same as you specified when shown. Note: To set up this value, the ORG_SIZE_ MODE must be turned on when initializing AVSynchronizer.

dwOrgSizeLeftBorder

Specify the size of the left border. This is valid only when bOrgSizeCallback is TRUE.

dwOrgSizeTopBorder

Specify the size of the top border. This is valid only when bOrgSizeCallback is TRUE.

dwOrgSizeRightBorder

Specify the size of the right border. This is valid only when bOrgSizeCallback is TRUE.

dwOrgSizeBottomBorder

Specify the size of the bottom border. This is valid only when bOrgSizeCallback is TRUE.

pfFrameRequest

Reserved. This is not implemented yet.

bNonBlocking

Reserved. This is not implemented yet.

dwFrameRequestContext

Reserved. This is not implemented yet.

Remarks

Requirements

Avsynchronizer.h

TSNAPSHOT

This structure is used by AvSynchronizer_GetCurrentSnapShot function.

```
typedef struct {
  DWORD                 dwdataSize;
  DWORD                 dwWidth;
  DWORD                 dwHeight;
  BYTE *                pDataStart;
} TSNAPSHOT, * PTSNAPSHOT;
```

Members

dwdataSize

Indicate the size of the snapshot in BYTE.

dwWidth

Indicate the width of the snapshot in pixel.

dwHeight

Indicate the Height of the snapshot in pixel

pDataStart

A pointer to the buffer the snapshot store in.

Remarks

Requirements

Avsynchronizer.h

TUPDATECHANNELOPTION

This structure collects the settings, which can change at run time of the channel. When you want to update the channel settings, fill this structure to setup the channel.

```
typedef struct {
  DWORD                 dwFlags;
  BOOL                  bVideoOut;
  BOOL                  bAudioOut;
  BOOL                  bMotionAlert;
  DWORD                 dwMotionRectAlert;
  BOOL                  bIgnoreBorder;
  DWORD                 dwMotionRect;
  DWORD                 dwVolume;
  TMediaType            tMediaType;
  HWND                  hWnd;
} TUPDATECHANNELOPTION, *PTUPDATECHANNELOPTION;
```

Members

dwFlags

Indicate which field of this structure is valid. It can be one or the combination of the values of EUPCHOPTION.

bVideoOut

Indicate if show the video of this channel.

bAudioOut

Indicate if this channel audible.

bMotionAlert

Indicate if drawing the rectangle of the motion detection window.

dwMotionRectAlert

Specify the color of the rectangle of the motion detection window when motion alert happened. The value is one of the values in EMOTIONALERTCOLOR enumeration. If you want to use the color that are not list in the EMOTIONALERTCOLOR, then you can set the MSB to be 1 and the rest of the DWORD is the RGB value.

bIgnoreBorder

Should the channel ignore the border setting when displaying video. TRUE means to ignore, FALSE means not.

dwMotionRect

Specify the color of the rectangle of the motion detection window. The value is one of the values in EMOTIONALERTCOLOR enumeration. If you don't want to show the rectangle of the motion detection window when motion alert didn't happen, set –1 as its value. If you want to use the color that are not list in the EMOTIONALERTCOLOR, then you can set the MSB to be 1 and the rest of the DWORD is the RGB value.

dwVolume

Specify the audio volume. The valid value is 0 ~ 5000.

tMediaType

Specify the media type of this channel. The value is one in EMEDIATYPE enumeration.

hWnd

The new window this channel attach to. Before setting new window, the UPCH_ RESTORE_WINPROC must be called.

Remarks

Requirements

Avsynchronizer.h

TUPDATEPBCHANNELOPTION

This structure collects the settings, which can change at run time of the playback channel. When you want to update playback channel settings, fill this structure to setup the playback channel.

```
typedef struct {
  DWORD                 dwFlags;
  BOOL                  bVideoOut;
  BOOL                  bAudioOut;
  BOOL                  bMotionAlert;
  BOOL                  bIgnoreBorder;
  DWORD                 dwMotionRectAlert;
  DWORD                 dwMotionRect;
  DWORD                 dwVolume;
  TMediaType            tMediaType;
  HWND                  hWnd
  DWORD                 dwFast;
  DWORD                 dwSlow;
  BOOL                  bPause;
} TUPDATEPBCHANNELOPTION,
*PTUPDATEPBCHANNELOPTION;
```

Members

dwFlags

Indicate which field of this structure is valid. It can be one or the combination of the values of EUPCHOPTION.

bVideoOut

Indicate if show the video of this channel.

bAudioOut

Indicate if this channel audible.

bMotionAlert

Indicate if drawing the rectangle of the motion detection window.

bIgnoreBorder

Should the channel ignore the border setting when displaying video. TRUE means to ignore, FALSE means not.

dwMotionRectAlert

Specify the color of the rectangle of the motion detection window when motion alert happened. The value is one of the values in EMOTIONALERTCOLOR enumeration.

dwMotionRect

Specify the color of the rectangle of the motion detection window. The value is one of the values in EMOTIONALERTCOLOR enumeration. If you don't want to show the rectangle of the motion detection window when motion alert didn't happen, set 0xFFFFFFFF as it's value.

dwVolume

Specify the audio volume. The valid value is 0 ~ 5000.

tMediaType

Specify the media type of this playback channel. The value is one in the EMEDIATYPE enumeration.

dwFast

Specify the multiple of speed for playing. This can't be zero if the dwSlow is not zero.

dwSlow

Specify the divisor of speed for playing. If this value is set to be 0 it will be play in full speed (as fast as possible).

bPause

Indicate if pause the channel. TRUE for pause. Set it FALSE to resume the channel.

hWnd

The new window this channel attach to. Before setting new window, the UPCH_ RESTORE_WINPROC must be called.

Remarks

Requirements

Avsynchronizer.h

TVFRAMEINFO

This structure is used by LPDECODEFRAMECALLBACK function.

```
typedef struct {
  DWORD                          dwWidth;
  DWORD                          dwHeight;
  DWORD                          dwSize;
  BYTE *                         pbyPicture;
  BYTE*                          pbyHeader;
  DWORD                          dwReserved;
  TMediaDataPacketInfo           tPacketInfo;
} TVFRAMEINFO, * PVTFRAMEINFO;
```

Members

dwWidth

Specify the width of the frame.

dwHeight

Specify the height of the frame.

dwSize

Specify the size of the frame.

pbyPicture

a pointer that store the data of the frame.

pbyHeader

This pointer is the starting address of the real buffer allocated. It's useful when users need to use the output data to generate another packet. In such case, the dwReserved is the reserved space between pbyHeader and pbyPicture. Users need to use AvSynchronizer_SetChannelOption to specify the reserved space they need. If not, this pointer is the same as pbyPicture is.

dwReserved

This is the difference in bytes between pbyHeader and pbyPicture.

tPacketInfo

The packet that contains this decoded video frame. In this structure: pbyBuff, dwOffset, dwBitstreamSize, dwAudioSamplingFreq, byAudioChannelNum, and bAudioDI are not valid, other fields are the same as the input packet of AvSynchronizer_InputToBeDecodedMediaFrame.

Remarks

Requirements

Avsynchronizer.h

TVIDEOOPTION

This structure is used to hold the video codec information for AVI channel.

```
typedef struct {
  DWORD                 dwfccType;
  DWORD                 dwHandler;
  LONG                  lKey;
  LONG                  lDataRate;
  LONG                  lQuality;
  LPBITMAPINFO          lpbiOut;
} TVIDEOOPTION;
```

Members

dwfccType

Type of compressor used. Currently only ICTYPE_VIDEO (VIDC) is supported. You could check the value for each codec by calling AvSynchronizer_ ChooseVideoCompressor.

dwHandler

It's four-character code of the compressor. Specify NULL to indicate the data is not to be recompressed. Specify "DIB" to indicate the data is an uncompressed, full frame. You can use this member to specify which compressor is selected by default when the dialog box is displayed

lKey

Key-frame rate. Specify an integer to indicate the frequency that key frames are to occur in the compressed sequence or zero to not use key frames.

lDataRate

Data rate, in kilobytes per second.

lQuality

Specify a quality setting of 1 to 10,000.

lpbiOut

Pointer to a BITMAPINFO structure containing the image output format.

Remarks

Requirements

Avsynchronizer.h

TVIDEOINFO

This structure is used to hold the video frame information from LPOUTPUTVIDEOFRAMEBEFOREDISPLAY.

```
typedef struct {
  DWORD                 dwStride;
  DWORD                 dwMaxHeight;
  DWORD                 dwWidth;
  DWORD                 dwHeight;
  BYTE *                pbyBuf;
} TVIDEOINFO;
```

Members

dwStride

The stride of video buffer.

dwMaxHeight

The maximum height of video buffer

dwWidth

Width of video

dwHeight

Height of video

pbyBuf

pointer to video buffer

Remarks

Requirements

Avsynchronizer.h

4.4. API Definition

The API definition is depicted here.

AvSynchronizer_ChooseAudioCompressor

Call this function to show up the audio compressor selection window for AVSynchronizer handle. This window belongs to windows API, so the user interface and the language is determined by the operation system. The settings and the per-channel settings will be used according to which one is more recently when channel conversion starts. For example if this one is called after the AvSynchronizer_ChooseChannelAudioCompressor, the channel will take the settings in AVShychronizer handle.

Syntax

```
SCODE AvSynchronizer_ChooseAudioCompressor (
    HANDLE hAVSync,
    HWND hWnd,
    const char * pszTitle,
    TAVIAINFO* pAudioInfo );
```

Parameters

hAVSync,

[in] the handle created by AvSynchronizer_Initial or AvSynchronizer_InitialEx.

hWnd

[in] the handle of the window that would be assigned as the parent of the video compressor choosing window.

pszTitle

[in] the title of the choosing window.

pAudioInfo

[in/out] the initial setting for the choosing dialog. It also contains the selected codec when return.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_NOT_IMPLEMENT

The AviConverter.dll is not found in the DLL search path.

Remarks

Note: to reduce the overhead for the programs that do not need AVI converter, the actual AVI converter operation code is implemented in AviConverter.dll. So for the programs that need AVI converter functions, please make sure the DLL exists in your working directory.

This function is only available for Windows 32 platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_ChooseChannelAudioCompressor

Call this function to show up the audio compressor selection window for channel. This window belongs to windows API, so the user interface and the language is determined by the operation system. The settings are related to the specified channel. So application could set different settings for different channels. The settings and the settings in Avsynchronizer will be used according to which one is more recently when channel conversion starts. For example if this one is called after the AvSynchronizer_ChooseAudioCompressor, the channel will take the settings in itself.

Syntax

```
SCODE
AvSynchronizer_ChooseChannelAudioCompressor (
                                              HANDLE hAVIChannel,
                                              HWND hWnd,
                                              const char * pszTitle,
                                              TAVIAINFO* pAudioInfo    );
```

Parameters

hAVIChannel,

[in] the AVI channel created by calling AvSynchronizer_CreateAVIChannel or AvSynchronizer_CreateLiveAVIChannel.

hWnd

[in] the handle of the window that would be assigned as the parent of the video compressor choosing window.

pszTitle

[in] the title of the choosing window.

pAudioInfo

[in/out] the initial setting for the choosing dialog. It also contains the selected codec when return.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_NOT_IMPLEMENT

The AviConverter.dll is not found in the DLL search path.

Remarks

Note: to reduce the overhead for the programs that do not need AVI converter, the actual AVI converter operation code is implemented in AviConverter.dll. So for the programs that need AVI converter functions, please make sure the DLL exists in your working directory.

This function is only available for Windows 32 platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_ChooseChannelVideoCompressor

Call this function to show up the video compressor selection window for channel. This window belongs to Windows API. So the user interface and the language are determined by the operation system. The settings are related to the specified channel. So application could set different settings for different channels. The settings and the settings in Avsynchronizer will be used according to which one is more recently when channel conversion starts. For example if this one is called after the AvSynchronizer_ChooseVideoCompressor, the channel will take the settings in itself.

Syntax

```
SCODE
AvSynchronizer_ChooseVideoCompressor (  HANDLE hAVIChannel,
                                        HWND hWnd,
                                        const char * pszTitle,
                                        TAVIVINFO * pVideoInfo      );
```

Parameters

hAVIChannel,

[in] the AVI channel created by calling AvSynchronizer_CreateAVIChannel.

hWnd

[in] the handle of the window that would be assigned as the parent of the video compressor choosing window.

pszTitle

[in] the title of the choosing window.

pVideoInfo

[in/out] the initial setting for the choosing dialog. It also contains the selected codec when return.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_NOT_IMPLEMENT

The AviConverter.dll is not found in the DLL search path.

Remarks

Note: to reduce the overhead for the programs that do not need AVI converter, the actual AVI converter operation code is implemented in AviConverter.dll. So for the programs that need AVI converter functions, please make sure the DLL exists in your working directory.

This function is only available for Windows 32 platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_ChooseVideoCompressor

Call this function to show up the video compressor selection window for AVSynchronizer handle. This window belongs to Windows API. So the user interface and the language are determined by the operation system. The settings and the per-channel settings will be used according to which one is more recently when channel conversion starts. For example if this one is called after the AvSynchronizer_ChooseChannelVideoCompressor, the channel will take the settings in AVShychronizer handle.

Syntax

```
SCODE
AvSynchronizer_
ChooseVideoCompressor (
                        HANDLE hAVIChannel,
                        HWND hWnd,
                        const char * pszTitle,
                        TAVIVINFO * pVideoInfo );
```

Parameters

hAVIChannel,

[in] the AVI channel created by calling AvSynchronizer_CreateAVIChannel or AvSynchronizer_CreateLiveAVIChannel.

hWnd

[in] the handle of the window that would be assigned as the parent of the video compressor choosing window.

pszTitle

[in] the title of the choosing window.

pVideoInfo

[in/out] the initial setting for the choosing dialog. It also contains the selected codec when return.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_NOT_IMPLEMENT

The AviConverter.dll is not found in the DLL search path.

Remarks

Note: to reduce the overhead for the programs that do not need AVI converter, the actual AVI converter operation code is implemented in AviConverter.dll. So for the programs that need AVI converter functions, please make sure the DLL exists in your working directory.

This function is only available for Windows 32 platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_CreateAVIChannel

Create a channel that could be used to convert audio/video stream into AVI file. It's optional to compress the audio and video frames by the compressors installed on system that runs the conversion program. Please refer to AvSynchronizer_ChooseVideoCompressor, AvSynchronizer_ChooseAudioCompressor, AvSynchronizer_SetVideoCompressorParam, AvSynchronizer_SetAudioCompressorParam. When using this channel, please input only the data in the same time interval if they are audio/video. Once the interval end or the conversion stopped, please delete the channel by calling AvSynchronizer_DeleteAVIChannel. Create a new channel for new time interval. When generating AVI file, users must call AvSynchronizer_InputAVIMedia to input un-decoded audio or video packets.

Syntax

```
SCODE                                HANDLE hAvSync,
AvSynchronizer_CreateAVIChannel (    HANDLE *phAVIChannel,
                                     TAVICHOPTION * ptAviChOption        );
```

Parameters

hAvSync

[in] the value of the handle returned by AvSynchronizer_Initial.

phAVIChannel

[out] the pointer to receive the handle of this channel.

ptAviChOption

[in] the channel option.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

AVSYNCHRONIZER_E_NOT_IMPLEMENT

The AviConverter.dll is not found in the DLL search path.

AVICONVERTER_E_OUT_OF_MEMORY

Memory allocation error happens when creating channel.

Remarks

Note: to reduce the overhead for the programs that do not need AVI converter, the actual AVI converter operation code is implemented in AviConverter.dll. So for the programs that need AVI converter functions, please make sure the DLL exists in your working directory.

This function is only available for Windows 32 platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_CreateChannel

Create a channel to display video or play sound.

Syntax

```
SCODE                          HANDLE hAvSync,
AvSynchronizer_CreateChannel ( HANDLE* phChannel,
                               TCHANNELOPTION tChannelOptions );
```

Parameters

hAvSync

[in] the handle of the AvSynchronizer, which created by AvSynchronizer_Initial.

***phChannel**

[out] the pointer to receive the handle of this channel.

tChannelOptionsl

[in] a structure contains the settings of the channel. See TCHANNELOPTION.

Return Values

AVSYNCHRONIZER_S_OK

Create the channel successfully.

AVSYNCHRONIZER_E_FAIL

Fail to create the channel.

AVSYNCHRONIZER_E_INVALID_ ARG

There are one or more arguments invalid.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

AVSYNCHRONIZER_E_OUT_OF_MEMORY

There is not enough memory.

AVSYNCHRONIZER_E_CHANNEL_EXCEEDED

You can't create more than 16 channels.

AVSYNCHRONIZER_E_DECODER _CHANNEL_ONLY

You can only create decoder channel only.

Remarks

This function is only available for Windows 32 platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_DeleteChannel, AvSynchronizer_CreatePlaybackChannel, TCHANNELOPTION.

AvSynchronizer_CreateDecoderChannel

Create a decoder channel. A decoder channel could decode audio and video at the same time.

Syntax

```
SCODE                                   HANDLE hAvSync,
AvSynchronizer_CreateDecoderChannel (   HANDLE *phDecChannel,
                                        TDecCHOption *ptDecChannelOption    );
```

Parameters

hAvSync

[in] the handle of the AvSynchronizer, which created by AvSynchronizer_Initial.

***phDecChannel**

[out] the pointer to receive the handle of this channel.

ptDecChannelOptions

[in] a structure contains the settings of the channel. See TDECCHOPTION.

Return Values

AVSYNCHRONIZER_S_OK

Create the channel successfully.

AVSYNCHRONIZER_E_FAIL

Fail to create the channel.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

Remarks

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_CreateLiveAVIChannel

Create a channel that could be used to convert live audio/video stream into AVI file. It's optional to compress the audio and video frames by the compressors installed on system that runs the conversion program. Please refer to AvSynchronizer_ChooseVideoCompressor, AvSynchronizer_ChooseAudioCompressor, AvSynchronizer_SetVideoCompressorParam, AvSynchronizer_SetAudioCompressorParam. When using this channel, please input only the data in the same time interval if they are audio/video. Once the interval end or the conversion stopped, please delete the channel by calling AvSynchronizer_DeleteAVIChannel. Create a new channel for new time interval. This channel is different from the AVI channel because it accept only decoded data. When generating AVI file, users must call AvSynchronizer_InputDecodedAVIMedia to input audio or video frames.

Syntax

```
SCODE                                   HANDLE hAvSync,
AvSynchronizer_CreateLiveAVIChannel (   HANDLE *phAVIChannel,
                                        TAVICHOPTION * ptAviChOption    );
```

Parameters

hAvSync

[in] the value of the handle returned by AvSynchronizer_Initial.

phAVIChannel

[out] the pointer to receive the handle of this channel.

ptAviChOption

[in] the channel option.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

AVSYNCHRONIZER_E_NOT_IMPLEMENT

The AviConverter.dll is not found in the DLL search path.

AVICONVERTER_E_OUT_OF_MEMORY

Memory allocation error happens when creating channel.

Remarks

Note: to reduce the overhead for the programs that do not need AVI converter, the actual AVI converter operation code is implemented in AviConverter.dll. So for the programs that need AVI converter functions, please make sure the DLL exists in your working directory.

This function is only available for Windows 32 platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_CreatePlaybackChannel

Create a channel to display video or play sound from the database.

Syntax

SCODE **AvSynchronizer_CreatePlaybackChannel (**	**HANDLE** hAvSync, **HANDLE *** phPlaybackChannel, **TPBCHOPTION** tPbChOption **);**

Parameters

hAvSync

[in] the handle of the AvSynchronizer, which created by AvSynchronizer_Initial.

***phPlaybackChannel**

[out] the pointer to receive the handle of this channel.

tPbChOption

[in] a structure contains the settings of the channel. See TPBCHOPTION.

Return Values

AVSYNCHRONIZER_S_OK

Create the channel successfully.

AVSYNCHRONIZER_E_FAIL

Fail to create the channel.

AVSYNCHRONIZER_E_INVALID_ ARG

There are one or more arguments invalid.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

AVSYNCHRONIZER_E_OUT_OF_MEMORY

There is not enough memory.

AVSYNCHRONIZER_E_CHANNEL_EXCEEDED

You can't create more than 16 channels.

AVSYNCHRONIZER_E_DECODER _CHANNEL_ONLY

You can only create decoder channel only.

Remarks

This function is only available for Windows 32 / Windows CE platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_CreateChannel, AvSynchronizer_DeleteChannel, TPBCHOPTION

AvSynchronizer_DecodeAudio

Decode audio in synchronous mode. The buffer should be given by caller, and if the buffer size is not enough, error will be returned without decode the audio.

Syntax

SCODE **AvSynchronizer_DecodeAudio (**	**HANDLE** hDecChannel, **TMediaDataPacketInfo*** pMediaDataPacket, **TAFRAMEINFO*** pfAudioFrameInfo, **DWORD*** pdwSize,);

Parameters

hDecChannel

[in] the decoder handle created by AvSynchronizer_CreateDecoderChannel.

pMediaDataPacket

[in] the pointer of the media packet that hold the to be decoded audio data.

pfAudioFrameInfo

[in/out] the pointer of the structure TAFRAMEINFO that holds the decoded audio frame. The pbySound must points to the buffer to hold the decoded audio. And the dwSize of this structure hold the size of the buffer. The minimum size of the buffer is (Size of a decoded frame) * (number of frame in a packet). The former depends on the codec type and the maximum value now is 2048. And the later is recorded in member dwFrameNumber of pMediaDataPacket.

pdwSize

[out] It holds the total size of the decoded audio.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_E_INVALID_DATA

The checksum of the packet is not correct.

AVSYNCHRONIZER_E_OUT_OF_MEMORY

There is not enough memory.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_BUFFER_SIZE_NOTENOUGH

The given buffer size is not enough to hold the decoded data.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

AVSYNCHRONIZER_E_DECODE_ERROR

The audio is decoded error.

Remarks

Note, if you turn off the output of audio by setting SETCH_OUTPUT_AUDIO_FRAME to be false, the call to this function will return AVSYNCHRONIZER_S_OK, but no data will be decoded, and no data will be returned.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_DecodeVideo

Decode video in synchronous mode. The buffer should be given by caller, and if the buffer size is not enough, error will be returned without decode the video.

Syntax

SCODE **AvSynchronizer_DecodeVideo (**	**HANDLE** hDecChannel, **TMediaDataPacketInfo*** pMediaDataPacket, **TVFRAMEINFO*** pfVideoFrameInfo, **DWORD*** pdwSize, **);**

Parameters

hDecChannel

[in] the decoder handle created by AvSynchronizer_CreateDecoderChannel.

pMediaDataPacket

[in] the pointer of the media packet that hold the to be decoded video data.

pfVideoFrameInfo

[in/out] the pointer of the structure TVFRAMEINFO that holds the decoded video frame. The pbyHeader must points to the buffer to hold the decoded audio. And the dwSize of this structure hold the size of the buffer. The minimum size of the buffer is (Width of the video) * (Height of the video) * (Bytes per pixel) for most video format. For Jpeg, it would relate to the compressing factor. Usually set the buffer size to 704x576x4 would fit most case. The buffer size must also add with the reserved buffer size set by SETCH_RESERVED_HEADER option with AvSynchronizer_SetChannelOption.

pdwSize

[out] It holds the total size of the decoded video.

Return Values

AVSYNCHRONIZER_S_OK

Decode the channel successfully.

AVSYNCHRONIZER_E_INVALID_DATA

The checksum of the packet is not correct.

AVSYNCHRONIZER_E_OUT_OF_MEMORY

There is not enough memory.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_BUFFER_SIZE_NOTENOUGH

The given buffer size is not enough to hold the decoded data.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

AVSYNCHRONIZER_E_DECODE_ERROR

The audio is decoded error.

Remarks

Note, if you turn off the output of video by setting SETCH_OUTPUT_VIDEO_FRAME to be false, the call to this function will return AVSYNCHRONIZER_S_OK, but no data will be returned. The video is still decoded if the stream type is H263 or MPEG4. For Jpeg, the video decoding process is also skipped.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_DecodeVideoDecodeOnly

This function is almost the same as AvSynchronizer_DecodeVideo, but this function only decodes and returns information and size of the frame. It does not retrieve the decoded frame content. Use AvSynchronizer_GetDecodedVideoFrame to get frame content.

Syntax

SCODE **AvSynchronizer_DecodeVideoDecodeOnly** **(**	**HANDLE** hDecChannel, **TMediaDataPacketInfo*** pMediaPacket, **TVFRAMEINFO*** pfVideoFrameInfo, **DWORD*** pdwSize, **);**

Parameters

hDecChannel

[in] the decoder handle created by AvSynchronizer_CreateDecoderChannel.

pMediaDataPacket

[in] the pointer of the media packet that hold the to be decoded video data.

pfVideoFrameInfo

[in/out] the pointer of the structure TVFRAMEINFO that holds the decoded video frame. The pbyHeader must points to the buffer to hold the decoded audio. And the dwSize of this structure hold the size of the buffer. The minimum size of the buffer is (Width of the video) * (Height of the video) * (Bytes per pixel) for most video format. For Jpeg, it would relate to the compressing factor. Usually set the buffer size to 704x576x4 would fit most case. The buffer size must also add with the reserved buffer size set by SETCH_RESERVED_HEADER option with AvSynchronizer_SetChannelOption.

pdwSize

[out] It holds the total size of the decoded video.

Return Values

AVSYNCHRONIZER_S_OK

Decode the channel successfully.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

AVSYNCHRONIZER_E_DECODE_ERROR

The audio is decoded error.

Remarks

This function can reduce the overhead when users want to decode the frame but do not retrieve the frame.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_DeleteAVIChannel

Delete the AVI channel whenever conversion is done. This function release all the resource associated with the channel.

Syntax

```
SCODE AvSynchronizer_DeleteAVIChannel (     HANDLE hAvSync,
                                            HANDLE *phAVIChannel,     );
```

Parameters

hAvSync

[in] the handle created by AvSynchronizer_Initial or AvSynchronizer_InitialEx.

phAVIChannel

[in/out] the pointer of the handle to be deleted and the content of the pointer would be cleared to NULL.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_S_ALREADY_DELETE

The channel has been removed.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

AVSYNCHRONIZER_E_NOT_IMPLEMENT

The AviConverter.dll is not found in the DLL search path.

Remarks

Note: to reduce the overhead for the programs that do not need AVI converter, the actual AVI converter operation code is implemented in AviConverter.dll. So for the programs that need AVI converter functions, please make sure the DLL exists in your working directory.

This function is only available for Windows 32 platform currently.

Requirements

AvSynchronizer.h, AviConverter.dll

See Also

AvSynchronizer_DeleteChannel

Delete the channel created by AvSynchronizer_CreateChannel or AvSynchronizer_CreatePlaybackChannel.

Syntax

```
SCODE AvSynchronizer_DeleteChannel  (        HANDLE hAvSync,
                                             HANDLE * pChannel        );
```

Parameters

hAvSync

[in] the handle of the AvSynchronizer, which created by AvSynchronizer_Initial.

***phChannel**

[in] the address of the handle of channel created by AvSynchronizer_CreateChannel or AvSynchronizer_CreatePlaybackChannel.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_E_FAIL

Fail to delete the channel.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

AVSYNCHRONIZER_E_CHANNEL_NOT_FOUND

The channel you specified does not exist.

Remarks

This function is only available for Windows 32 / Windows CE platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_CreateChannel, **AvSynchronizer_CreatePlaybackChannel**

AvSynchronizer_DeleteDecoderChannel

Delete the decode channel.

Syntax

```
SCODE AvSynchronizer_DeleteDecoderChannel (     HANDLE hAvSync,
                                                HANDLE *pDecChannel    );
```

Parameters

hAvSync

[in] the handle of the AvSynchronizer, which created by AvSynchronizer_Initial.

***phDecChannel**

[in] the address of the handle created by AvSynchronizer_CreateDecoderChannel.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_E_FAIL

Fail delete the channel.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

Remarks

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_FreePicture

Call this function to free the snapshot created by AvSynchronizer_GetCurrentSnapShot.

Syntax

```
SCODE AvSynchronizer_FreePicture (      PTSNAPSHOT ptSnapShot      );
```

Parameters

ptSnapShot

[in] the address of the TSNAPSHOT structure.

Return Values

AVSYNCHRONIZER_S_OK

Free snapshot successfully.

AVSYNCHRONIZER_E_FAIL

Fail to free snapshot.

Remarks

This function is only available for Windows 32 / Windows CE platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_GetAudioCompressorParam

Retrieve the current audio compressor setting in the AVSynchronizer handle.

Syntax

```
SCODE
AvSynchronizer_GetAudioCompressorParam (    HANDLE hAVSync,
                                            TAUDIOOPTION* ptOption      );
```

Parameters

hAVSync

[in] the value of the handle returned by AvSynchronizer_Initial.

ptAudioOption

[out] the setting for the audio compressor.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_NOT_IMPLEMENT

The AviConverter.dll is not found in the DLL search path.

Remarks

Note: to reduce the overhead for the programs that do not need AVI converter, the actual AVI converter operation code is implemented in AviConverter.dll. So for the programs that need AVI converter functions, please make sure the DLL exists in your working directory.

This function is only available for Windows 32 / Windows CE platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_GetChannelAudioCompressorParam

Retrieve the current audio compressor setting in the channel.

Syntax

```
SCODE
AvSynchronizer_GetChannelAudioCompressorParam (   HANDLE hAVIChannel,
                                                  TAUDIOOPTION* ptOption   );
```

Parameters

hAVIChannel,

[in] the AVI channel created by calling AvSynchronizer_CreateAVIChannel or AvSynchronizer_CreateLiveAVIChannel.

ptAudioOption

[out] the setting for the audio compressor.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_NOT_IMPLEMENT

The AviConverter.dll is not found in the DLL search path.

Remarks

Note: to reduce the overhead for the programs that do not need AVI converter, the actual AVI converter operation code is implemented in AviConverter.dll. So for the programs that need AVI converter functions, please make sure the DLL exists in your working directory.

This function is only available for Windows 32 / Windows CE platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_GetChannelVideoCompressorParam

Retrieve the current video compressor settings in the channel.

Syntax

SCODE	**HANDLE** hAVIChannel,
AvSynchronizer_GetVideoCompressorParam (	**TVIDEOOPTION** * ptOption **);**

Parameters

hAVIChannel,

[in] the AVI channel created by calling AvSynchronizer_CreateAVIChannel or AvSynchronizer_CreateLiveAVIChannel.

pVideoInfo

[out] the setting for the video compressor.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_NOT_IMPLEMENT

The AviConverter.dll is not found in the DLL search path.

Remarks

Note: to reduce the overhead for the programs that do not need AVI converter, the actual AVI converter operation code is implemented in AviConverter.dll. So for the programs that need AVI converter functions, please make sure the DLL exists in your working directory.

This function is only available for Windows 32 / Windows CE platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_GetCapabilities

Get the capability of the video card.

Syntax

```
SCODE AvSynchronizer_GetCapabilities (  HANDLE hAvSync,
                                        DWORD * pdwCapabilities,
                                        DWORD dwScreen                    );
```

Parameters

hAvSync

[in] the handle of the AvSynchronizer, which created by AvSynchronizer_Initial.

***pdwCapabilities**

[out] the address of the DWORD to receive the value of capabilities. The capabilities are defined in EDISPLAYCAP.

dwReserverd

[in] This is the reserved field that should be 0 now.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_E_FAIL

Fail delete the channel.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

Remarks

This function is only available for Windows 32 / Windows CE platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_GetCurrentSnapShot

Call this function to get a snapshot.

Syntax

SCODE AvSynchronizer_GetCurrentSnapShot (	**HANDLE** hChannel, **PTSNAPSHOT** ptSnapShot, **DWORD** dwFlag **);**

Parameters

hChannel

[in] the handle of the channel, which created by AvSynchronizer_CreateChannel or AvSynchronizer_CreatePlaybackChannel.

ptSnapShot

[out] the address of the TSNAPSHOT structure.

dwFlag

[in] indicate if get the original video size image. See ESNAPSHOTFLAG.

Return Values

AVSYNCHRONIZER_S_OK

Get snapshot successfully.

AVSYNCHRONIZER_E_FAIL

Fail to get snapshot.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

Remarks

When you don't need the snapshot, you must call AvSynchronizer_FreePicture to free the snapshot.

This function is only available for Windows 32 / Windows CE platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_FreePicture, TSNAPSHOT, ESNAPSHOTFLAG.

AvSynchronizer_GetCurrentDisplay

Call this function to get a snapshot of current display area.

Syntax

```
SCODE                                   HANDLE hChannel,
AvSynchronizer_GetCurrentDisplay (      HDC hDC,
                                        TCURRENTDISPLAYINFO *ptDspInfo      );
```

Parameters

hChannel

[in] the handle of the channel, which created by AvSynchronizer_CreateChannel or AvSynchronizer_CreatePlaybackChannel.

HDC

[int/out] The DC to draw the current display area.

ptDspInfo

[in/out] The structure to store current display information. See TCURRENTDISPLAYINFO.

Return Values

AVSYNCHRONIZER_S_OK

Get snapshot successfully.

AVSYNCHRONIZER_E_FAIL

Fail to get current display.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

Remarks

This function is only available for Windows 32 / Windows CE platform currently.

Requirements

AvSynchronizer.h

AvSynchronizer_GetDecodedVideoFrame

Get the previous decoded frame as the specified output format.

Syntax

```
SCODE
AvSynchronizer_GetDecodedVideoFrame
(
    HANDLE hDecChannel,
    TMediaDataPacketInfo * pPacket,
    TVFRAMEINFO *pfVideoFrameInfo,
    DWORD dwRawDataFormat,
    DWORD dwReservedHeaderLen,
    DWORD *pdwSize                    );
```

Parameters

hDecChannel

[in] the decoder handle created by AvSynchronizer_CreateDecoderChannel.

pMediaDataPacket

[out] the pointer of the media packet that hold the video frame information to be output.

pfVideoFrameInfo

[in/out] the pointer of the structure TVFRAMEINFO that holds the decoded video frame. The pbyHeader must points to the buffer to hold the decoded audio. And the dwSize of this structure hold the size of the buffer. The minimum size of the buffer is (Width of the video) * (Height of the video) * (Bytes per pixel) for most video format. For Jpeg, it would relate to the compressing factor. Usually set the buffer size to 704x576x4 would fit most case. The buffer size must also add with the reserved buffer size set by SETCH_RESERVED_HEADER option with AvSynchronizer_SetChannelOption.

dwRawDataFormat

[in]the video format of the output

dwReservedHeaderLen

[in]the reserved header length

pdwSize

[out] It holds the total size of the decoded video.

Return Values

AVSYNCHRONIZER_S_OK

Decode the channel successfully.

AVSYNCHRONIZER_E_INVALID_DATA

The checksum of the packet is not correct.

AVSYNCHRONIZER_E_OUT_OF_MEMORY

There is not enough memory.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_BUFFER_SIZE_NOTENOUGH

The given buffer size is not enough to hold the decoded data.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

AVSYNCHRONIZER_E_DECODE_ERROR

The audio is decoded error.

Remarks

Use this function to get the previous frame in various formats repeatedly. AvSynchronizer_DecodeVideo can only get a given frame with specified format once.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_DecodeVideoDecodeOnly, AvSynchronizer_DecodeVideo

AvSynchronizer_GetFlags

Call this function to retrieve the flags for AVSynchronizer.

Syntax

```
SCODE AvSynchronizer_GetFlags (     HANDLE hAvSync
                                    DWORD *pdwFlags                    );
```

Parameters

hAvSync

[in] the AvSynchronizer object, returned by AvSynchronizer_Initial.

pdwFlags

[out] the buffer to hold the returned flags.

Return Values

AVSYNCHRONIZER_S_OK

Retrieve flags successfully.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

Remarks

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_Initial, AvSynchronizer_InitialEx.

AvSynchronizer_GetVersionInfo

Get the version info about the AvSynchronizer module.

Syntax

```
SCODE AvSynchronizer_ GetVersionInfo (    BYTE * pbyMajor,
                                          BYTE * pbyMinor,
                                          BYTE * pbyBuild,
                                          BYTE * pbyRevision          );
```

Parameters

***pbyMajor**

[out] the address of a BYTE to receive the Major number of the version.

***pbyMinor**

[out] the address of a BYTE to receive the Minor number of the version.

***pbyBuildr**

[out] the address of a BYTE to receive the Build number of the version.

***pbyRevision**

[out] the address of a BYTE to receive the Revision number of the version.

Return Values

AVSYNCHRONIZER_S_OK

Get version info successfully.

AVSYNCHRONIZER_E_FAIL

Fail to get version info.

Remarks

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_GetRemainingQueueElementCount

Call this function to get the queue status of AvSynchronizer.

Syntax

SCODE AvSynchronizer_GetRemainingQueueElementCount (	**HANDLE** hChannel, **TAVSYNCQUEUESTATUS** *ptQueueStatus **);**

Parameters

hChannel

[in] he handle of the channel, which created by AvSynchronizer_CreateChannel or AvSynchronizer_CreatePlaybackChannel.

ptQueueStatus

[out] the structure holds the status of queue.

Return Values

AVSYNCHRONIZER_S_OK

Retrieve queue status successfully.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

Remarks

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_Initial, AvSynchronizer_InitialEx.

AvSynchronizer_GetVideoCompressorParam

Retrieve the current video compressor setting in the AVSynchronizer handle.

Syntax

```
SCODE
AvSynchronizer_GetVideoCompressorParam    HANDLE hAVIChannel,
(                                         TVIDEOOPTION * ptVideoOption    );
```

Parameters

hAvSync

[in] the AvSynchronizer object, returned by AvSynchronizer_Initial.

pVideoInfo

[out] the setting for the video compressor.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_NOT_IMPLEMENT

The AviConverter.dll is not found in the DLL search path.

Remarks

Note: to reduce the overhead for the programs that do not need AVI converter, the actual AVI converter operation code is implemented in AviConverter.dll. So for the programs that need AVI converter functions, please make sure the DLL exists in your working directory.

This function is only available for Windows 32 / Windows CE platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_Initial

This function initializes the AvSynchronizer Module. You must call either this function or AvSynchronizer_InitialEx before using this module.

Syntax

```
SCODE AvSynchronizer_Initial (  HANDLE * phAvSync,
                                LPSTATUSCALLBACK pfStatus,
                                LPDISPLAYCALLBACK pfDisplay,
                                HWND hWnd,
                                DWORD dwAudioFocusType,
                                long lSyncParameter,
                                DWORD dwVersion,
                                DWORD dwFlag                    );
```

Parameters

***phAvSync**

[out] the pointer to receive the handle of this AvSynchronizer object.

pfStatus

[in] The pointer to the LPSTATUSCALLBACK. To be used to notify the application the status of AvSynchronizer.

pfDisplay

[in] The pointer to the LPDISPLAYCALLBACK. To be used to notify the application to display something.

hWnd

[in] Window handle used for the application. Set to the calling application's top-level window handle (not a handle for any child windows created by the top-level window).

dwAudioFocusType

[in] indicate the focus type of the DirectSound buffer. Its value is one in EAUDIOFOCUSTYPE enumeration.

lSyncParameter

[in] a long as a parameter to tune the audio and video synchronization.

dwVersion

[in] the version of the AvSynchronizer. Assign the value AVSYNCHRONIZER_VERSION to this parameter.

dwFlag

[in] See EAVSYBCINITFLAG.

Return Values

AVSYNCHRONIZER_S_OK

Initializes this module ok.

AVSYNCHRONIZER_S_NO_AUDIO_PLAY

Audio initializes failed. Audio input will be ignored.

AVSYNCHRONIZER_E_FAIL
Initial this module fail.

AVSYNCHRONIZER_E_INVALID_VERSION
You are using an incompatible version.

AVSYNCHRONIZER_E_INVALID_ ARG
There are one or more arguments invalid.

AVSYNCHRONIZER_E_OUT_OF_MEMORY
There are not enough memory.

AVSYNCHRONIZER_E_CREATE_EVENT_FAILED
Create event fail

Remarks

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_Release, AvSynchronizer_InitialEx, LPSTATUSCALLBACK, LPDISPLAYCALLBACK

AvSynchronizer_InitialEx

This function initializes the AvSynchronizer Module. You must call either this function or AvSynchronizer_Initial before using this module.

Syntax

```
SCODE AvSynchronizer_InitialEx (  HANDLE * phAvSync,
                                  LPSTATUSCALLBACK pfStatus,
                                  LPDISPLAYCALLBACK pfDisplay,
                                  HWND hWnd,
                                  DWORD dwAudioFocusType,
                                  long lSyncParameter,
                                  DWORD dwVersion,
                                  DWORD dwFlag,
                                  DWORD dwBlankColor          );
```

Parameters

***phAvSync**

[out] the pointer to receive the handle of this AvSynchronizer object.

pfStatus

[in] The pointer to the LPSTATUSCALLBACK. To be used to notify the application the status of AvSynchronizer.

pfDisplay

[in] The pointer to the LPDISPLAYCALLBACK. To be used to notify the application to display something.

hWnd

[in] Window handle used for the application. Set to the calling application's top-level window handle (not a handle for any child windows created by the top-level window).

dwAudioFocusType

[in] indicate the focus type of the DirectSound buffer. Its value is one in EAUDIOFOCUSTYPE enumeration.

lSyncParameter

[in] a long as a parameter to tune the audio and video synchronization.

dwVersion

[in] the version of the AvSynchronizer. Assign the value AVSYNCHRONIZER_VERSION to this parameter.

dwFlag

[in] See EAVSYBCINITFLAG.

dwBlankColor

[in] Specified the color to fill when there is no video to display.

Return Values

AVSYNCHRONIZER_S_OK
Initializes this module ok.

AVSYNCHRONIZER_S_NO_AUDIO_PLAY
Audio initializes failed. Audio input will be ignored.

AVSYNCHRONIZER_E_FAIL
Initial this module fail.

AVSYNCHRONIZER_E_INVALID_VERSION
You are using an incompatible version.

AVSYNCHRONIZER_E_INVALID_ ARG
There are one or more arguments invalid.

AVSYNCHRONIZER_E_OUT_OF_MEMORY
There are not enough memory.

AVSYNCHRONIZER_E_CREATE_EVENT_FAILED
Create event fail

Remarks

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_Release, AvSynchronizer_Initial, LPSTATUSCALLBACK, LPDISPLAYCALLBACK

AvSynchronizer_InputAVIMedia

Input the media frames into the AVI channel. The channel will decode the frame and insert them into the AVI files. Compression would be taken if specified.

Syntax

```
SCODE AvSynchronizer_InputAVIMedia (   HANDLE hAVIChannel,
                                       TmediaDataPacketInfo *pPacketInfo      );
```

Parameters

hAVIChannel,

[in] the AVI channel created by calling AvSynchronizer_CreateAVIChannel.

ptMediaDataPacketInfo

[in] the data packet received from network or retrieved from database.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

AVSYNCHRONIZER_E_NOT_IMPLEMENT

The AviConverter.dll is not found in the DLL search path.

AVSYNCHRONIZER_E_INVALID_DATA

The checksum of the packet is not correct.

AVICONVERTER_E_OUT_OF_MEMORY

Memory allocation error happens when generating AVI file.

AVSYNCHRONIZER_E_DECODE_ERROR

Media frame decoded error.

Remarks

Note: to reduce the overhead for the programs that do not need AVI converter, the actual AVI converter operation code is implemented in AviConverter.dll. So for the programs that need AVI converter functions, please make sure the DLL exists in your working directory.

This function is only available for Windows 32 platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_InputDecodedAVIMedia

Input the media frames into the AVI channel. The channel will decode the frame and insert them into the AVI files. Compression would be taken if specified.

Syntax

```
SCODE AvSynchronizer_InputDecodedAVIMedia (     HANDLE hAVIChannel,,
                                                TMediaType eFrameType,
                                                TFRAMEINFO * ptFrameInfo     );
```

Parameters

hAVIChannel,

[in] the AVI channel created by calling AvSynchronizer_CreateLiveAVIChannel.

eFrameType

[in] The media type for current frame.

ptFrameInfo

[in] The decoded video or audio frame data.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

AVSYNCHRONIZER_E_NOT_IMPLEMENT

The AviConverter.dll is not found in the DLL search path.

AVSYNCHRONIZER_E_INVALID_DATA

The checksum of the packet is not correct.

AVICONVERTER_E_OUT_OF_MEMORY

Memory allocation error happens when generating AVI file.

AVSYNCHRONIZER_E_DECODE_ERROR

Media frame decoded error.

Remarks

Note: to reduce the overhead for the programs that do not need AVI converter, the actual AVI converter operation code is implemented in AviConverter.dll. So for the programs that need AVI converter functions, please make sure the DLL exists in your working directory.

This function is only available for Windows 32 platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_InputEncodedMediaFrame

Call this function to input data into the channel.

Syntax

```
SCODE                                   HANDLE hChannel,
AvSynchronizer_InputEncodedMediaFrame (  TMediaDataPacketInfo  *pPacket    );
```

Parameters

hChannel

[in] the handle of the channel, which created by AvSynchronizer_CreateChannel.

***pMediaDataPacket**

[in] the address of the structure of data packet generated from DataBroker module. See TMediaDataPacketInfo.

Return Values

AVSYNCHRONIZER_S_OK

Input data to the channel successfully.

AVSYNCHRONIZER_E_FAIL

Fail to input data to the channel.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

AVSYNCHRONIZER_E_INVALID_DATA

The data you input is not the valid data from VIVOTEK.

AVSYNCHRONIZER_E_FRAME_DROPPED

There is one frame dropped.

AVSYNCHRONIZER_E_MEDIA _TYPE_NEEDED

You need to setup the media type before the input the data packet.

Remarks

This function is only available for Windows 32 / Windows CE platform currently.

Requirements

AvSynchronizer.h , datapacketdef.h

See Also

AvSynchronizer_InputPlaybackMediaFrame, TMediaDataPacketInfo

AvSynchronizer_InputPlaybackMediaFrame

Call this function to input data into the playback channel.

Syntax

```
SCODE
AvSynchronizer_InputPlaybackMediaFrame (    HANDLE hChannel,
                                            TMediaDataPacketInfo  *pPacket    );
```

Parameters

hChannel

[in] the handle of the channel, which created by AvSynchronizer_CreatePlaybackChannel.

***pMediaDataPacket**

[in] the address of the structure of data packet generated from DataBroker module. See TMediaDataPacketInfo.

Return Values

AVSYNCHRONIZER_S_OK

Input data to the channel successfully.

AVSYNCHRONIZER_E_FAIL

Fail to input data to the channel.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

AVSYNCHRONIZER_E_INVALID_DATA

The data you input is not the valid data from VIVOTEK.

AVSYNCHRONIZER_E_FRAME_DROPPED

There is one frame dropped.

Remarks

This function is only available for Windows 32 / Windows CE platform currently.

Requirements

AvSynchronizer.h , datapacketdef.h

See Also

AvSynchronizer_InputEncodedMediaFrame, TMediaDataPacketInfo.

AvSynchronizer_InputToBeDecodedMediaFrame

Input data to the decode channel.

Syntax

```
SCODE
AvSynchronizer_InputToBeDecodedMediaFrame(   HANDLE hDecChannel
                                             TMediaDataPacketInfo *pPacket   );
```

Parameters

hDecChannel

[in] the handle of the channel, which created by AvSynchronizer_CreateDecoderChannel.

***pMediaDataPacket**

[in] the address of the structure of data packet generated from DataBroker module. See TMediaDataPacketInfo.

Return Values

AVSYNCHRONIZER_S_OK

Input data to the channel successfully.

AVSYNCHRONIZER_E_FAIL

Fail to input data to the channel.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

Remarks

Requirements

AvSynchronizer.h, datapacketdef.h

See Also

AvSynchronizer_Release

Call this function to release the AvSynchronizer object.

Syntax

```
SCODE AvSynchronizer_Release (      HANDLE *phAvSync                    );
```

Parameters

***phAvSync**

[in] the address of the pointer to the AvSynchronizer object, returned by AvSynchronizer_Initial.

Return Values

AVSYNCHRONIZER_S_OK

Release the object successfully.

AVSYNCHRONIZER_E_FAIL

Failed to release the object.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

Remarks

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_Initial

AvSynchronizer_SetAudioCompressorParam

Set the audio compressor parameters for AVSynchronizer handle. This function allows program to set the compressor without invoking the selection dialog. The settings and the per-channel settings will be used according to which one is more recently when channel conversion starts. For example if this one is called after the AvSynchronizer_SetChannelAudioCompressorParam, the channel will take the settings in AVShychronizer handle.

Syntax

```
SCODE
AvSynchronizer_SetAudioCompressorParam (    HANDLE hAVSync,
                                            TAUDIOOPTION* ptOption    );
```

Parameters

hAvSync

[in] the AvSynchronizer object, returned by AvSynchronizer_Initial.

ptAudioOption

[in] the setting for the audio compressor. The setting could be kept by application in its own database and retrieved whenever starts.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_NOT_IMPLEMENT

The AviConverter.dll is not found in the DLL search path.

Remarks

Note: to reduce the overhead for the programs that do not need AVI converter, the actual AVI converter operation code is implemented in AviConverter.dll. So for the programs that need AVI converter functions, please make sure the DLL exists in your working directory.

This function is only available for Windows 32 / Windows CE platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_SetChannelAudioCompressorParam

Set the audio compressor parameters. This function allows program to set the compressor without invoking the selection dialog. If the channel is used without calling this function, the video would be generated uncompressed in the AVI file. The settings and the settings in Avsynchronizer will be used according to which one is more recently when channel conversion starts. For example if this one is called after the AvSynchronizer_SetAudioCompressorParam, the channel will take the settings in itself.

Syntax

```
SCODE
AvSynchronizer_SetChannelAudioCompressorParam (   HANDLE hAVIChannel,
                                                  TAUDIOOPTION* ptOption   );
```

Parameters

hAVIChannel,

[in] the AVI channel created by calling AvSynchronizer_CreateAVIChannel or AvSynchronizer_CreateLiveAVIChannel.

ptAudioOption

[in] the setting for the audio compressor. The setting could be kept by application in its own database and retrieved whenever starts.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_NOT_IMPLEMENT

The AviConverter.dll is not found in the DLL search path.

Remarks

Note: to reduce the overhead for the programs that do not need AVI converter, the actual AVI converter operation code is implemented in AviConverter.dll. So for the programs that need AVI converter functions, please make sure the DLL exists in your working directory.

This function is only available for Windows 32 / Windows CE platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_SetChannelOption

Set an option for the channel. This function could be applied to all the three channels but if the target channel does not support the option, AVSYNCHRONIZER_E_INVALID_ARG will be returned.

Syntax

SCODE AvSynchronizer_ SetChannelOption (	**HANDLE** hChannel, **DWORD** dwOption, **DWORD** dwParam1, **DWORD** dwParam2 **);**

Parameters

hChannel

[in] the handle of the channel that is created by AvSynchronizer_CreateChannel, AvSynchronizer_CreatePlaybackChannel, or AvSynchronizer_CreateDecoderChannel.

dwOption

[in] the value of the option defined in ESETCHOPTION

dwParam1

[in] the value of the first parameter, the meaning of this parameter depends on the value of dwOption.

dwParam2

[in] the value of the second parameter, the meaning of this parameter depends on the value of dwOption.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

Remarks

Note: to reduce the overhead for the programs that do not need AVI converter, the actual AVI converter operation code is implemented in AviConverter.dll. So for the programs that need AVI converter functions, please make sure the DLL exists in your working directory.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_SetChannelVideoCompressorParam

Set the video compressor parameters for this channel. This function allows program to set the compressor without invoking the selection dialog. If the channel is used without calling this function, the video would be generated uncompressed in the AVI file. The settings and the settings in Avsynchronizer will be used according to which one is more recently when channel conversion starts. For example if this one is called after the AvSynchronizer_SetVideoCompressorParam, the channel will take the settings in itself.

Syntax

```
SCODE
AvSynchronizer_SetChannelVideoCompressorParam
(
    HANDLE hAVIChannel,
    TVIDEOOPTION * ptOption
);
```

Parameters

hAVIChannel,

[in] the AVI channel created by calling AvSynchronizer_CreateAVIChannel.

pVideoInfo

[in] the setting for the video compressor. The setting could be kept by application in its own database and retrieved whenever starts.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_NOT_IMPLEMENT

The AviConverter.dll is not found in the DLL search path.

Remarks

Note: to reduce the overhead for the programs that do not need AVI converter, the actual AVI converter operation code is implemented in AviConverter.dll. So for the programs that need AVI converter functions, please make sure the DLL exists in your working directory.

This function is only available for Windows 32 / Windows CE platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_SetFlags

Call this function to change the flags for AVSynchronizer at runtime.

Syntax

```
SCODE AvSynchronizer_SetFlags (    HANDLE hAvSync
                                   DWORD dwFlags                    );
```

Parameters

hAvSync

[in] the AvSynchronizer object, returned by AvSynchronizer_Initial.

dwFlags

[in] the flags to be set, the possible values for this flag is defined EAVSYBCINITFLAG except DECODER_CHANNEL_ONLY, USE_DIRECTDRAW_ONLY and ORG_SIZE_MODE. It's possible to change the drawing mode from DirectDraw to GDI at runtime, but not vice versa.

Return Values

AVSYNCHRONIZER_S_OK

Change flags successfully.

AVSYNCHRONIZER_E_COULDNT_SWITCH_TWOPASS

When changing for one pass to two passes mode, the needs resource could not be acquired. It's safe for AP to continue running if this error is returned. But the flag is not changed.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

AVSYNCHRONIZER_E_INVALID_ARG

The given parameters are not correct.

Remarks

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_Initial, AvSynchronizer_InitialEx.

AvSynchronizer_SetVideoCompressorParam

Set the video compressor parameters for AVSynchronizer handle. This function allows program to set the compressor without invoking the selection dialog. If the channel is used without calling this function, the video would be generated uncompressed in the AVI file. The settings and the per-channel settings will be used according to which one is more recently when channel conversion starts. For example if this one is called after the AvSynchronizer_GetChannelVideoCompressorParam, the channel will take the settings in AVShychronizer handle.

Syntax

```
SCODE
AvSynchronizer_SetVideoCompressorParam (    HANDLE hAVIChannel,
                                            TVIDEOOPTION * ptOption      );
```

Parameters

hAvSync

[in] the AvSynchronizer object, returned by AvSynchronizer_Initial.

pVideoInfo

[in] the setting for the video compressor. The setting could be kept by application in its own database and retrieved whenever starts.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_NOT_IMPLEMENT

The AviConverter.dll is not found in the DLL search path.

Remarks

Note: to reduce the overhead for the programs that do not need AVI converter, the actual AVI converter operation code is implemented in AviConverter.dll. So for the programs that need AVI converter functions, please make sure the DLL exists in your working directory.

This function is only available for Windows 32 / Windows CE platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_StartAVIChannel

Start the AVI channel. This initializes some parameters such as the output file name and media type, etc….

Syntax

```
SCODE AvSynchronizer_StartAVIChannel (    HANDLE hAVIChannel,
                                          TAVICHOPTION2 * ptAviOption2    );
```

Parameters

hAVIChannel

[in] the channel handle created by calling AvSynchronizer_CreateAVIChannel.

ptAviChOption2

[int] the setting for the channel.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

AVSYNCHRONIZER_E_NOT_IMPLEMENT

The AviConverter.dll is not found in the DLL search path.

AVICONVERTER_E_INVALID_FILENAME

The file name member in TAVICHOPTION2 structure is empty.

AVSYNCHRONIZER_E_OPEN_AVI_FILE

Unable to open the AVI file, the path may not exist, the user does not have the permission to open file or the disk is full.

AVSYNCHRONIZER_E_CREATE_AVI_V_STREAM

Unable to create video stream in AVI file, may the system resource is not enough.

AVSYNCHRONIZER_E_CREATE_AVI_A_STREAM

Unable to create audio stream in AVI file, may the system resource is not enough.

Remarks

Note: to reduce the overhead for the programs that do not need AVI converter, the actual AVI converter operation code is implemented in AviConverter.dll. So for the programs that need AVI converter functions, please make sure the DLL exists in your working directory.

This function is only available for Windows 32 platform currently.

Requirements

AvSynchronizer.h, AviConverter.dll

See Also

AvSynchronizer_StartChannel

Call this function to start the channel. You can't see the video before calling this function.

Syntax

```
SCODE AvSynchronizer_StartChannel (   HANDLE hChannel
                                      DWORD dwFlag                    );
```

Parameters

hChannel

[in] the handle of the channel, which created by AvSynchronizer_CreateChannel or AvSynchronizer_CreatePlaybackChannel.

dwFlag

[in] See ESTARTCHANNELFLAG.

Return Values

AVSYNCHRONIZER_S_OK

Start the channel successfully.

AVSYNCHRONIZER_E_FAIL

Fail to start the channel.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

Remarks

This function is only available for Windows 32 / Windows CE platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_CreateChannel, AvSynchronizer_CreatePlaybackChannel, AvSynchronizer_StopChannel

AvSynchronizer_StopAVIChannel

Stop the AVI channel and close the AVI file.

Syntax

```
SCODE AvSynchronizer_StopAVIChannel (     HANDLE hAVIChannel                    );
```

Parameters

hAVIChannel

[in] the handle of the channel to be stopped. Please ensure that no more packet is input to the channel after calling this function.

Return Values

AVSYNCHRONIZER_S_OK

Delete the channel successfully.

AVSYNCHRONIZER_E_INVALID_ARG

The given option is not supported by this channel.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

AVSYNCHRONIZER_E_NOT_IMPLEMENT

The AviConverter.dll is not found in the DLL search path.

AVSYNCHRONIZER_E_ENCODE_AVI_AUDIO

Encode audio packet failed.

AVSYNCHRONIZER_E_WRITE_AVI_FILE

Write media data to AVI file error.

Remarks

Note: to reduce the overhead for the programs that do not need AVI converter, the actual AVI converter operation code is implemented in AviConverter.dll. So for the programs that need AVI converter functions, please make sure the DLL exists in your working directory.

This function is only available for Windows 32 platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_StopChannel

Call this function to stop the channel. Once you call this to stop a channel, you must call AvSynchronizer_StartChannel if you want to restart the channel again.

Syntax

```
SCODE AvSynchronizer_StopChannel (    HANDLE hChannel                    );
```

Parameters

hChannel

[in] the handle of the channel, which created by AvSynchronizer_CreateChannel or AvSynchronizer_CreatePlaybackChannel.

Return Values

AVSYNCHRONIZER_S_OK

Stop the channel successfully.

AVSYNCHRONIZER_E_FAIL

Fail to stop the channel.

Remarks

This function is only available for Windows 32 / Windows CE platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_CreateChannel, AvSynchronizer_CreatePlaybackChannel, AvSynchronizer_StartChannel.

AvSynchronizer_UpdateChannelSettings

Call this function to change some settings of the channel.

Syntax

SCODE **AvSynchronizer_UpdateChannelSettings (**	**HANDLE** hChannel, **TUPDATECHANNELOPTION** tUpdateOptions **);**

Parameters

hChannel

[in] the handle of the channel, which created by AvSynchronizer_CreateChannel.

tUpdateChannelOptions

[in] a structure contains some settings of the channel. See TUPDATECHANNELOPTION.

Return Values

AVSYNCHRONIZER_S_OK

Update the channel successfully.

AVSYNCHRONIZER_E_FAIL

Fail to update the channel.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

AVSYNCHRONIZER_E_NOT_APPLICABLE

This action is not acceptible.

Remarks

This function is only available for Windows 32 / Windows CE platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_UpdatePlaybackChannelSettings, TUPDATECHANNELOPTION

AvSynchronizer_UpdatePlaybackChannelSettings

Call this function to change some settings of the playback channel.

Syntax

```
SCODE
AvSynchronizer_UpdatePlaybackChannelSettings
(
    HANDLE hChannel,
    TUPDATEPBCHANNELOPTION tUpdateOptions
);
```

Parameters

hChannel

[in] the handle of the channel, which created by AvSynchronizer_CreatePlaybackChannel.

tUpdatePBChannelOptions

[in] a structure contains the settings of the channel. See TUPDATEPBCHANNELOPTION.

Return Values

AVSYNCHRONIZER_S_OK

Update the playback channel successfully.

AVSYNCHRONIZER_E_FAIL

Fail to update the playback channel.

AVSYNCHRONIZER_E_INVALID_HANDLE

The handle can't be NULL.

Remarks

This function is only available for Windows 32 / Windows CE platform currently.

Requirements

AvSynchronizer.h

See Also

AvSynchronizer_UpdateChannelSettings, TUPDATEPBCHANNELOPTION.