Windows Player SDK
Programmer Manual

(For Windows 7/XP/2000/2003/Vista 64 bit)

Version 6.3.x.x

2013-01

Notices

The information in this documentation is subject to change without notice and does not represent any commitment on behalf of HIKVISION. HIKVISION disclaims any liability whatsoever for incorrect data that may appear in this documentation. The product(s) described in this documentation are furnished subject to a license and may only be used in accordance with the terms and conditions of such license.

This documentation is issued in strict confidence and is to be used only for the purposes for which it is supplied. It may not be reproduced in whole or in part, in any form, or by any means or be used for any other purpose without prior written consent of HIKVISION and then only on the condition that this notice is included in any such reproduction. No information as to the contents or subject matter of this documentation, or any part thereof, or arising directly or indirectly therefrom, shall be given orally or in writing or shall be communicated in any manner whatsoever to any third party being an individual, firm, or company or any employee thereof without the prior written consent of HIKVISION. Use of this product is subject to acceptance of the HIKVISION agreement required to use this product. HIKVISION reserves the right to make changes to its products as circumstances may warrant, without notice.

This documentation is provided "as-is," without warranty of any kind.

Please send any comments regarding the documentation to:
overseasbusiness@hikvision.com

Find out more about HIKVISION at www.hikvision.com

Contents

Chapter 1 Product Description

The Player SDK (hereby referred to as “The SDK” or “The player SDK”) is the secondary development kit for the decoding of HIKVISION DVR, DVS and IP devices, etc. The SDK supports video/ audio decoding from all the devices listed below:

DS-95xx/96xx series and DS-76xx series NVR;

DS-90xx series and DS-76xx series hybrid DVR;

DS-91xx series, DS-81xx/71xx/72xx series, DS-80xx/70xx series, DS-73xx series DVR; ATM DVR and mobile DVR;

DS-60xx series, DS-61xx series, DS-63xx series, DS-64xx series, DS-65xx series and DS-66xx series DVS, Decoder and Encoder;

DS-40xx/41xx/42xx/43xx series compression card;

IP devices: IP module, IP camera and IP Speed Dome, etc

The main functions of the Player SDK include real time live view of video stream, playback of recording files with control functions such as pause, step forward, step backward, etc; and the SDK can also get stream information such as file index, decoding frame info, resolution and frame rate, etc. The SDK also supports snapshot in BMP or JPG format.

Chapter 2 Version Update

Version Description

The naming rules of Player SDK version is described as following.

Main Version. Sub Version. Fix Version. Reserved Version

- ***Main version update:*** large-scale modification, re-construction or optimization of the SDK
- ***Sub version update:*** Additional functions/features added to the SDK
- ***Fix version update:*** partial changes or bug-fixing of the SDK
- ***Reserved version:*** reserved

Special Notice: If your CPU supports Hyper-Threading Technology, please use V3.4 or higher version of the SDK.

Version Information

Version 6.3.x.x

Update:

1. Update the decode library.
2. Modify the video playback mechanism according to the timestamp .

Addition:

1. Support the audio AAC.
2. Support reverse playback ,synchronous playback,image rotation,capture part image .
3. Add APIs ：
   PlayM4_ReversePlay()
   PlayM4_GetSystemTime()
   PlayM4_OpenStreamAdvanced
   PlayM4_SetRotateAngle
   PlayM4_SetSycGroup
   PlayM4_SetSycStartTime
   PlayM4_GetCropJPEG
   PlayM4_GetCropBMP
4. Support playback by SDP.

Version 6.2.2.4（2012.6）

Update:

1. Synchronous to Win32 baseline V6.3.0 version.
2. Add support for ST devices.
3. Fix the bug of the block of the audio playback

Version 6.2.1.3（2011.11）

1. Support basic function API
2. Support the vIdeo stream type :264,jpeg,MPEG4;
3. Support the Audio stream type:G711,G722,mpg.

Chapter 3 Error Code Definition

ID	Code	Description
PLAYM4_NOERROR	0	No error
PLAYM4_PARA_OVER	1	Illegal input parameter
PLAYM4_ORDER_ERROR	2	Calling reference error
PLAYM4_TIMER_ERROR	3	Set timer failure
PLAYM4_DEC_VIDEO_ERROR	4	Video decoding failure
PLAYM4_DEC_AUDIO_ERROR	5	Audio decoding failure
PLAYM4_ALLOC_MEMORY_ERROR	6	Memory allocation failure
PLAYM4_OPEN_FILE_ERROR	7	File operation failure
PLAYM4_CREATE_OBJ_ERROR	8	Create thread failure
PLAYM4_CREATE_DDRAW_ERROR	9	Create DirectDraw failure
PLAYM4_CREATE_OFFSCREEN_ERROR	10	Create offscreen failure
PLAYM4_BUF_OVER	11	Buffer overflow, input stream failure
PLAYM4_CREATE_SOUND_ERROR	12	Create sound device failure
PLAYM4_SET_VOLUME_ERROR	13	Set volume failure
PLAYM4_SUPPORT_FILE_ONLY	14	This API can only be called in file decoding mode
PLAYM4_SUPPORT_STREAM_ONLY	15	This API can only be called in stream decoding mode
PLAYM4_SYS_NOT_SUPPORT	16	System not support, the SDK can only work with CPU above Pentium 3
PLAYM4_FILEHEADER_UNKNOWN	17	Missing file header
PLAYM4_VERSION_INCORRECT	18	Version mismatch between encoder and decoder
PALYM4_INIT_DECODER_ERROR	19	Initialize decoder failure
PLAYM4_CHECK_FILE_ERROR	20	File too short or unrecognizable stream
PLAYM4_INIT_TIMER_ERROR	21	Initialize timer failure
PLAYM4_BLT_ERROR	22	BLT failure
PLAYM4_UPDATE_ERROR	23	Update overlay surface failure
PLAYM4_OPEN_FILE_ERROR_MULTI	24	Open video & audio stream failure
PLAYM4_OPEN_FILE_ERROR_VIDEO	25	Open video stream failure
PLAYM4_JPEG_COMPRESS_ERROR	26	JPEG compression failure
PLAYM4_EXTRACT_NOT_SUPPORT	27	File type not supported
PLAYM4_EXTRACT_DATA_ERROR	28	Data error
PLAYM4_SECRET_KEY_ERROR	29	Secret key error
PLAYM4_DECODE_KEYFRAME_ERROR	30	Key frame decoding failure
PLAYM4_NEED_MORE_DATA	31	Not enough data
PLAYM4_INVALID_PORT	32	Invalid port number
PLAYM4_NOT_FIND	33	Searching failed

PLAYM4_FAIL_UNKNOWN	99	Unknown error

Chapter 4 Display Description

The display part of the player mainly adopts DirectDraw technology. There are 2 ways to achieve video display:

1. Create off screen image with BLT.
2. Create OVERLAY surface.

The characters of these two methods are:

1. For the Off screen display mode, the merits are: each channel of the Multi-channel playback can be relatively independent and not influenced by each other. Shortcomings: the display performance is greatly affected by the display adapter. If the display adapter does not support zoom operation, the player will zoom via software if needed (while the display window and the original image size are different), which will cause high CPU usage. Therefore we provide an API PlayM4_GetCaps to test the display adapter's capability. Users can test with their display adapters to see if it supports BLT zooming, etc. Form 1 is a display adapter that we tested;
2. For the OVERLAY display mode, the merits are: Most display adapters support OVERLAY mode, and the OVERLAY mode supports hardware zooming. If the display adapter does not support off screen with zooming, then we can use overlay mode. Shortcoming: There can be only one OVERLAY surface for each display adapter, and thus only 1 channel display can use OVERLAY mode. If there is other program that occupies the OVERLAY surface, the player cannot use OVERLAY then; and if the OVERLAY surface is occupied by the player, the other programs cannot display in overlay mode, either.

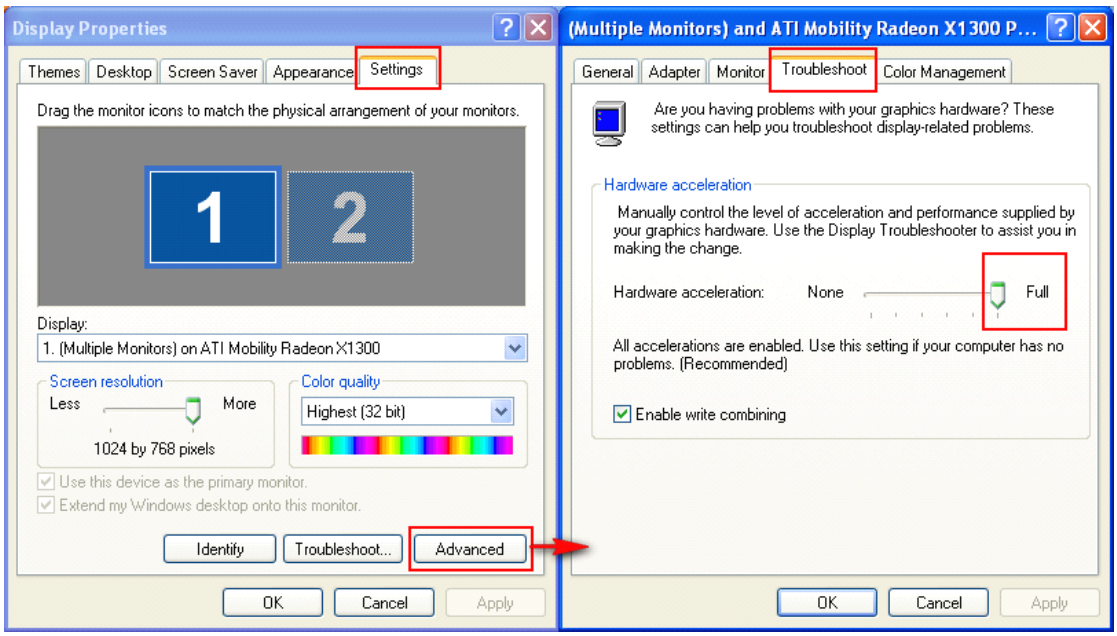
For the Windows 2003 OS users, please kindly Notice:

The Server Operating System such as Windows 2003 is not developed for multimedia display functions, and thus some of the video handling functions (i.e. hardware acceleration, directX) are disabled by default and need to be enabled manually.

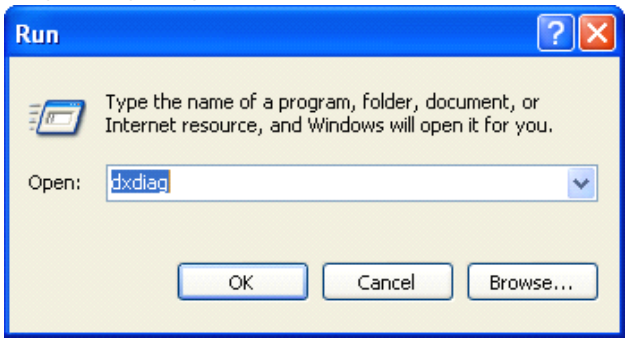
As most of the display functions in the SDK is depended on the BLT function of the display adapter (image zooming via hardware, when the original image size is not consistent with the display image size, we'll need this feature on the display adapter), if the display adapter (or its driver) does not support this feature, the SDK will switch to software zooming mode, and then the CPU usage will increase accordingly.

For Windows 2003 OS, please kindly follow the below operation steps:

1. Right click on Windows Desktop-> Select "Properties" ->Settings-> Advanced-> Troubleshoot->and drag "Hardware Acceleration" to "Full".



2. Click “Start”-> “Run” of Windows task bar, input “dxdiag” and enter DirectX diagnosing program.



3. Enable “DirectDraw” and “Direct3D” options in the “Display” column.

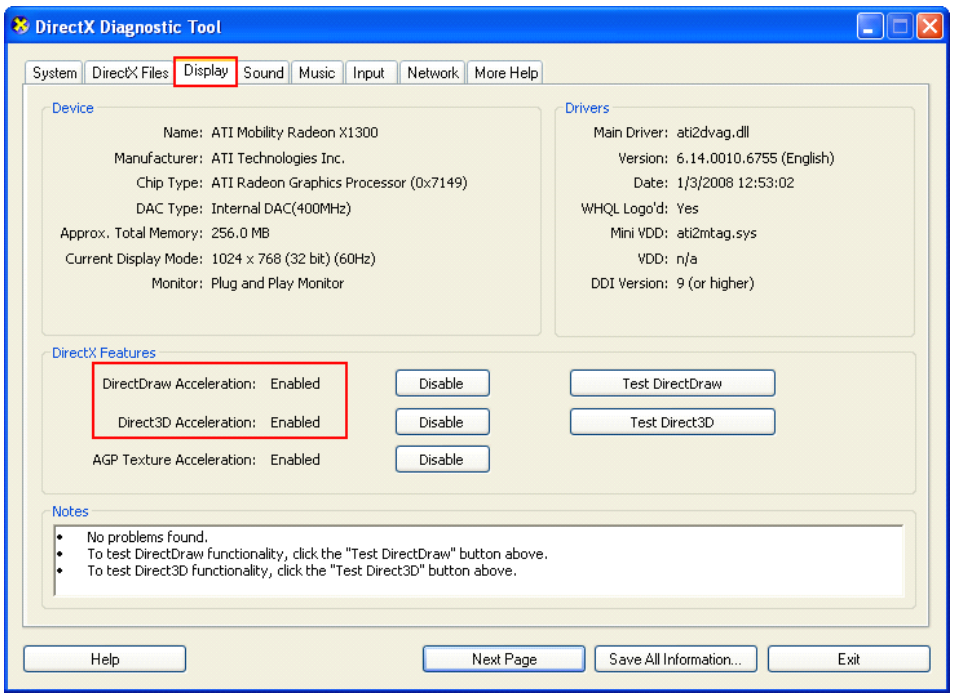


Table 1: These are some video adapter models which have been already tested (Windows2000)

Display Adapter Model	Memory (M)	Support Color Convertion	Support shrink	Support Stretch
ATI Rage128	32	YES	YES	YES
ATI Radeon LE	32	YES	YES	YES
ATI Radeon 7200	64	YES	YES	YES
nVidia TNT2 Model64	16 & 32	YES	YES	YES
nVidia TNT2 Pro	32	YES	YES	YES
Geforce2 Mx, Mx200, Mx400	32	YES	YES	YES
Geforce4 Mx420, Mx440	32	YES	YES	YES
**********				
Sis630	16	NO	NO	NO
Sis305	32	YES	NO	NO

Notice:

For the nVidia display adapters, users may need to update the driver to the latest version, otherwise some function may not be supported. If you find other display adapters cannot support some display functions, it is suggested to update the driver and test again.

Chapter 5 API Calling Reference

Initialize application

File Mode

PlayM4_GetPort
PlayM4_SetFileRefCallBack
PlayM4_OpenFile
PlayM4_Play
**(PlayM4_ReversePlay) for reverse playback*
PlayM4_Stop
PlayM4_CloseFile
PlayM4_FreePort

Stream Mode

PlayM4_GetPort
PlayM4_SetStreamOpenMode
PlayM4_OpenStream
PlayM4_SetDisplayBuf
PlayM4_Play
**(PlayM4_ReversePlay) for reverse playback*
PlayM4_InputData
PlayM4_Stop

End of application

Chapter 6 API Description

System Operation and Error Code Query

6.1 Get SDK Version and Build Number PlayM4_GetSdkVersion

DWORD PlayM4_GetSdkVersion();

Description:

Get SDK version and build number.

Parameters:

--

Return:

The higher 16 digits stands for the current build number, digits 9~16 stands for the main version and digit 1~8 is the sub version, e.g. return value 0x06040105 stands for Version1.5, Build 0604.

Notice:

For the debug version SDKs, there will only be difference in build number.

6.2 Get Error Code PlayM4_GetLastError

DWORD PlayM4_GetLastError (LONG nPort);

Description:

Get the error code.

Parameters:

nPort

[in] Valid port number of the player

Return:

The value specified the error code. For the error description, please refer to the table in Chapter 3.

6.3 Check Display Capabilities of the System PlayM4_GetCaps

int PlayM4_GetCaps ();

Description:

Check the display capabilities of the system information for the player.

Parameters:

--

Return:

Bit 0-8 respectively stands for the following info (a TRUE value of bitwise AND operation stands for 'support this function'):

SUPPORT_DDRAW

Support DIRECTDRAW. If not, the player SDK cannot function correctly.

SUPPORT_BLT

Support BLT operation. If not, the player SDK cannot function correctly.

SUPPORT_BLTFOURCC

Support color-space conversions during blit operations.

SUPPORT_BLTSHRINKX

Support arbitrary shrinking of a surface along the x-axis (horizontally). This flag is valid only for blit operations.

SUPPORT_BLTSHRINKY

Support arbitrary shrinking of a surface along the y-axis (vertically). This flag is valid only for blit operations.

SUPPORT_BLTSTRETCHX

Support arbitrary stretching of a surface along the x-axis (horizontally). This flag is valid only for blit operations.

SUPPORT_BLTSTRETCHY

Support arbitrary stretching of a surface along the y-axis (vertically). This flag is valid only for blit operations.

SUPPORT_SSE

Support SSE instruction set. If supports, It will get high performance.

SUPPORT_MMX

Support MMX instruction set.

Notice:

If all the above functions are supported by the display adapter, the CPU usage will be lowered greatly. Otherwise it is suggested to keep the size of display window the same as the decoded video.

I.e. If the decoded video is 352*288 (PAL), and the display adapter does not support BLT, then it is suggested to set the display window size as 352*288.

6.4 Initial DirectDraw Surface PlayM4_InitDDraw

BOOL PlayM4_InitDDraw (HWND hWnd);

Description:

Initialize DirectDraw surface.

Parameters:

hWnd

[in] The window handle of the application mainframe.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails. To get detailed error information, please call API PlayM4_GetLastError() to get the error definition.

Notice:

1. For SDK version above V1.1, users do not need to call this function.
2. Please kindly notice that for VB & DELPHI developing, please disable the WS_CLIPCHILDREN window style of the dialog box, otherwise the display image will be covered by the controls on the dialog box.

6.5 Release DirectDraw Surface PlayM4_RealeseDDraw

BOOL PlayM4_ReleaseDDraw ();

Description:

Release DirectDraw surface.

Parameters:

--

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

For the SDK version above V1.1, users do not need to call this function.

6.6 Set Timer Type PlayM4_SetTimerType

BOOL _stdcall PlayM4_SetTimerType (LONG nPort, DWORD nTimerType, DWORD nReserved);

Description:

Set the timer for the SDK.

Parameters:

nPort

[in] Valid port number of the player

nTimerType

[in] **TIMER_1** or **TIMER_2**, please refer to the **MACRO Definition** below.

nReserved

[in] reserved.

Macro Definition:

TIMER_1: Multi-media timer, support up to 16 for each process.

TIMER_2: No numeric limitation, but this timer is with lower precision, and is not recommended for high speed playback.

Channel 0-15 will be assigned to TIMER_1 while other channels will be assigned to TIMER_2 by default.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

This API shall be called before open operation. It is suggested to use TIMER_1 for file playback, and TIMER_2 for live view.

6.7 Get Timer Type PlayM4_GetTimerType

BOOL _stdcall PlayM4_GetTimerType (LONG nPort, DWORD *pTimerType, DWORD *pReserved);

Description:

Get the timer for player SDK.

Parameters:

nPort

[in] Valid port number of the player

pTimerType

[out] TIMER_1 or TIMER_2;

pReserved

[out] reserved.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.8 Get Valid Port Number PlayM4_GetPort

BOOL _stdcall PlayM4_GetPort (LONG* nPort);

Description:

Get a valid port number. Valid range of port number is [0, 499].

Parameters:

nPort

[in] [out] Long pointer of get the port number

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.9 Release Player Port PlayM4_FreePort

BOOL _stdcall PlayM4_FreePort (LONG nPort);

Description:

Release the port number which has been occupied

Parameters:

nPort

[in] Port number of the player

It is suggested to set the **nPort** as -1 after a successful port releasing.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

File Operation

6.10 Open File PlayM4_OpenFile

BOOL PlayM4_OpenFile (LONG nPort, LPSTR sFileName);

Description:

Open the file for playback.

Parameters:

nPort

[in] Valid port number of the player

sFileName

[in] The file name

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

- The file size ranges from 4KB to 4GB.
- Only support files locally stored on PC.

6.11 Close File PlayM4_CloseFile

BOOL PlayM4_CloseFile (LONG nPort);

Description:

Close the file that has been opened.

Parameters:

nPort

[in] Valid port number of the player

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Stream Operation

6.12 Set Stream Input Mode PlayM4_SetStreamOpenMode

BOOL PlayM4_SetStreamOpenMode (LONG nPort, DWORD nMode);

Description:

Set stream input mode.

Parameters:

nPort

[in] Valid port number of the player

nMode

[in] work in stream mode:

0: **STREAME_REALTIME** mode (default)

1: **STREAME_FILE** mode

STREAME_REALTIME mode gives priority to ensuring of real-time performance and preventing of data blocking problem, and is with strict data checking mechanism while **STREAME_FILE** is the contrary.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

This API shall be called before playback starts.

6.13 Get Stream Input Mode PlayM4_GetStreamOpenMode

LONG PlayM4_GetStreamOpenMode (LONG nPort);

Description:

Get stream input mode.

Parameters:

nPort

[in] Valid port number of the player

Return:

STREAME_REALTIME or STREAME_FILE

6.14 Open Stream PlayM4_OpenStream

BOOL PlayM4_OpenStream (LONG nPort, PBYTE pFileHeadBuf, DWORD nSize, DWORD nBufPoolSize);

Description:

Open the stream for playback (similar with open file).

Parameters:

nPort

[in] Valid port number of the player

pFileHeadBuf

[in] The file header which is got from the recording callback APIs of network client SDK or card SDK

nSize

[in] The data length of the file header.

nBufPoolSize

[in] Specify the size of the source buffer. It ranges from SOURCE_BUF_MIN to SOURCE_BUF_MAX. Users may encounter decoding failure if nBufPoolSize is too small. It is suggested that nBufPoolSize be no less than 200*1024 for SD (standard definition) devices, and no less than 600*1024 for HD (high definition) devices.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.15 Close Stream PlayM4_CloseStream

```
BOOL PlayM4_CloseStream (LONG nPort);
```

Description:

Close the stream which has been opened.

Parameters:

nPort

[in] Valid port number of the player

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.16 Input Stream Data PlayM4_InputData

```
BOOL PlayM4_InputData (LONG nPort, PBYTE pBuf, DWORD nSize);
```

Description:

Input stream data.

Parameters:

nPort

[in] Valid port number of the player

pBuf

[in] buffer address

nSize

[in] buffer size

Return:

If the data input succeeds, the return value is TRUE.

If the data input fails, the return value is FALSE. Stream data input shall starts after OpenStream, and a FALSE return is often caused by the case of full buffer.

Notice:

Suggestions for data input failure handling:

1. For the data input failure caused by full buffer under **STREAME_REALTIME**

mode, users can either discard the data (which will cause frame loss or incomplete decoding video) or retry after sleep of several milliseconds.

2. For the data input failure caused by full buffer under **STREAME_FILE** mode, users shall retry until input succeeds.

6.17 Open Video/Audio Stream Separately PlayM4_OpenStreamEx

BOOL _stdcall PlayM4_OpenStreamEx (LONG nPort, PBYTE pFileHeadBuf, DWORD nSize, DWORD nBufPoolSize);

Description:

Open video and audio streams separately.

Parameters:

nPort

[in] Valid port number of the player

pFileHeadBuf

[in] The file header which is got from card

nSize

[in] The size of the file header

nBufPoolSize

[in] Set the size of the source buffer which ranges from SOURCE_BUF_MIN to SOURCE_BUF_MAX

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.18 Close Video/Audio Stream Separately PlayM4_CloseStreamEx

BOOL _stdcall PlayM4_CloseStreamEx (LONG nPort);

Description:

Close the stream which has been opened in video/audio-separate mode.

Parameters:

nPort

[in] Valid port number of the player

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.19 Open Video/Audio Stream Separately (Advanced Mode) PlayM4_OpenStreamAdvanced

BOOL _stdcall PlayM4_OpenStreamAdvanced(LONG nPort, int nProtocolType, PLAYM4_SESSION_INFO* pstSessionInfo, DWORD nBufPoolSize);

Description:

Open video and audio streams separately with no HIKVISION's 40-byte header

format. Currently this API supports open stream in SDP mode.

Parameters:

nPort

[in] Valid port number of the player

nProtocolType

[in] The protocol type

pstSessionInfo

[in] Pointer of PLAYM4_SESSION_INFO structure

nBufPoolSize

[in] Set the size of the source buffer which ranges from SOURCE_BUF_MIN to SOURCE_BUF_MAX

Description of nProtocolType and PLAYM4_SESSION_INFO:

```
#ifndef PLAYM4_SESSION_INFO_TAG
#define PLAYM4_SESSION_INFO_TAG
//nProtocolType
#define PLAYM4_PROTOCOL_RTSP      1
//nSessionInfoType
#define PLAYM4_SESSION_INFO_SDP 1

typedef struct _PLAYM4_SESSION_INFO_        //SessionInfo structure
{
        int             nSessionInfoType;   //Session Info type, e.x. SDP
        int             nSessionInfoLen;    //Data length of Session Info
unsigned char* pSessionInfoData;    //Session Info data
} PLAYM4_SESSION_INFO;
#endif
```

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.20 Input Video Data PlayM4_InputVideoData

```
BOOL _stdcall PlayM4_InputVideoData (LONG nPort, PBYTE pBuf, DWORD nSize);
```

Description:

Input video stream data got from card

Parameters:

nPort

[in] Valid port number of the player

pBuf

[in] Pointer of the buffer containing the data to be written to the source buffer

nSize

[in] Number of bytes to write to the source buffer

Return:

If the video input succeeds, the return value is TRUE.

If the video input fails, the return value is FALSE.

6.21 Input Audio Data PlayM4_InputAudioData

```
BOOL _stdcall PlayM4_InputAudioData (LONG nPort, PBYTE pBuf, DWORD nSize);
```

Description:

Input audio stream data.

Parameters:

nPort

[in] Valid port number of the player

pBuf

[in] Pointer of the buffer which contains the data to write to the source buffer

nSize

[in] buffer size

Return:

If the audio input succeeds, the return value is TRUE.

If the audio input fails, the return value is FALSE.

Notice:

Audio data shall be input after OpenStream. A FALSE return is often caused by case of a full buffer. It is suggested that user stops the thread of data input and retry again to prevent data loss.

Playback Control

6.22 Start Playback PlayM4_Play

BOOL PlayM4_Play (LONG nPort, HWND hWnd);

Description:

Start playback. The display image size will be automatically adjusted according to the hWnd window size. For full screen display, please magnify the hWnd window to full screen size.

If the video is already in playback status, calling this API will reset the playback speed to 1X.

Parameters:

nPort

[in] Valid port number of the player

hWnd

[in] The handle of the display window.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.23 Stop Playback PlayM4_Stop

BOOL PlayM4_Stop (LONG nPort);

Description:

Stop playback.

Parameters:

nPort

[in] Valid port number of the player

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.24 Pause Playback PlayM4_Pause

BOOL PlayM4_Pause (LONG nPort, DWORD nPause);

Description:

Pause or resume playback.

Parameters:

nPort

[in] Valid port number of the player

nPause

[in]

TRUE: pause

FALSE: resume the previous playback process

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.25 Fast Forward PlayM4_Fast

BOOL PlayM4_Fast (LONG nPort);

Description:

Fast forward. This API can be called up to 4 times continuously to increase the playback speed, which will be doubled after each API call. Call PlayM4_Play() to restore 1X playback from the current position.

Parameters:

nPort

[in] Valid port number of the player

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

HD video may not reach the fast forward speed set by the user due to limitation of decoding and display performance.

6.26 Slow Forward PlayM4_Slow

BOOL PlayM4_Slow (LONG nPort);

Description:

Slow forward. This API can be called up to 4 times continuously to decrease the playback speed, which will be lowered by half after each API call. Call PlayM4_Play() to restore 1X playback from the current position.

Parameters:

nPort

[in] Valid port number of the player

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.27 Step Forward PlayM4_OneByOne

BOOL PlayM4_OneByOne (LONG nPort);

Description:

Step Forward. Call PlayM4_Play() to restore 1X playback from the current position.

Parameters:

nPort

[in] Valid port number of the player

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.28 Step Backward PlayM4_OneByOneBack

BOOL PlayM4_OneByOneBack (LONG nPort);

Description:

Step backward. Reverse 1 frame after each API call. This API shall be called after file index is generated.

Parameters:

nPort

[in] Valid port number of the player

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.29 Play Sound in Exclusive Mode PlayM4_PlaySound

BOOL PlayM4_PlaySound (LONG nPort);

Description:

Open the audio. Only 1-ch audio playback can be enabled, and the audio on other channels will be switched off automatically.

Parameters:

nPort

[in] Valid port number of the player

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

1. The audio display is disabled by default.
2. PlayM4_PlaySound() and PlayM4_StopSound() shall be used together in the application. If PlayM4_PlaySound() is called, then PlayM4_StopSound() shall be called before application ends.

6.30 Stop Sound in Exclusive Mode PlayM4_StopSound

BOOL PlayM4_StopSound ();

Description:

Stop the audio playback.

Parameters:

--

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.31 Play Sound in Share Mode PlayM4_PlaySoundShare

BOOL PlayM4_PlaySoundShare (LONG nPort);

Description:

Open audio from a specified channel in share mode. It doesn't stop audio from other channels.

Parameters:

nPort

[in] Valid port number of the player

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

Creating of multiple audio devices is not supported on Windows 98 and earlier OS version. If the sound adapter has already been occupied, the API will return FALSE.

6.32 Stop Sound in Share Mode PlayM4_StopSoundShare

BOOL PlayM4_StopSoundShare (LONG nPort);

Description:

Stop audio playback from a specified channel.

Parameters:

nPort

[in] Valid port number of the player

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

PlayM4_PlaySoundShare() and PlayM4_StopSoundShare() shall be used together in the application. If PlayM4_PlaySoundShare() is called, then PlayM4_StopSoundShare() shall be called before application ends.

6.33 Set Volume PlayM4_SetVolume

BOOL PlayM4_SetVolume (LONG nPort, WORD nVolume);

Description:

Set the volume of the PC sound adapter. It may effect other applications related with PC sound devices.

Parameters:

nPort

[in] Valid port number of the player

nVolume

[in] New volume requested for this sound, range 0-0XFFFF.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

1. If this API is called before playback starts, the return value will be FALSE yet the volume setting will be saved as the initial volume when audio playback starts.
2. This API adjusts the audio output volume of the sound adapter and will effect the audio on all the related applications.

6.34 Get Volume PlayM4_GetVolume

WORD PlayM4_GetVolume (LONG nPort);

Description:

Get the volume level of the PC's audio adapter. It may effect other applications related with the system's display adapter.

Parameters:

nPort

[in] Valid port number of the player

Return:

Volume

6.35 Adjust Audio Wave PlayM4_AdjustWaveAudio

BOOL _stdcall PlayM4_AdjustWaveAudio (LONG nPort, LONG nCoefficient);

Description:

Adjust the wave data. This API can be called to adjust the volume of current channel only, while API PlayM4_SetVolume effects on the whole system.

Parameters:

nPort

[in] Valid port number of the player

nCoefficient

[in] The coefficient. The range from MIN_WAVE_COEF to MAX_WAVE_COEF, 0 means no adjust.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

This API will adjust the audio data directly and only effects on the selected channel. Calling of this API will lower the audio quality, and is not suggested to use unless the user wants to adjust each channel's volume separately.

6.36 Set Picture Quality PlayM4_SetPicQuality

BOOL PlayM4_SetPicQuality (LONG nPort, BOOL bHighQuality);

Description:

Set the image quality. High image quality settings leads to higher CPU usage, and

thus it is suggested to set low quality for multi-channel playback, and switch to high quality when a certain channel is enlarged during display.

Parameters:

nPort

[in] Valid port number of the player

bHighQuality

[in] is not 0- high quality; 0- low quality (default)

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.37 Get Picture Quality PlayM4_GetPictureQuality

BOOL PlayM4_GetPictureQuality (LONG nPort, BOOL *bHighQuality);

Description:

Get the quality of the image.

Parameters:

nPort

[in] Valid port number of the player

bHighQuality

[out] 1- high quality; 0- low quality

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.38 Set Display Video Parameters PlayM4_SetColor

BOOL _stdcall PlayM4_SetColor (LONG nPort, DWORD nRegionNum, int nBrightness, int nContrast, int nSaturation, int nHue);

Description:

Set video parameters of the pictures.

Parameters:

nPort

[in] Valid port number of the player

nRegionNum:

[in] Display region, please refer to PlayM4_SetDisplayRegion; if there is only one display region (the most common case) it should be set as 0.

nBrightness:

[in] Brightness, default: 64, range: from 0 to 128;

nContrast:

[in] Contrast, default: 64; range: from 0 to 128;

nSaturation:

[in] Saturation, default: 64; range: from 0 to 128;

nHue:

[in] Hue, default: 64; range: from 0 to 128;

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

If all the parameters are set as default level, the color settings will not be adjusted.

6.39 Get Display Video Parameters PlayM4_GetColor

BOOL _stdcall PlayM4_GetColor (LONG nPort, DWORD nRegionNum, int *pBrightness, int *pContrast, int *pSaturation, int *pHue);

Description:

Get the corresponding color value, parameters are as above.

Parameters:

nPort

[in] Valid port number of the player

nRegionNum:

[in] Display region, please refer to PlayM4_SetDisplayRegion; if there is only one display region (the most common case) it should be set as 0.

nBrightness:

[out] The brightness, default: 64, range: from 0 to 128;

nContrast:

[out] The contrast, default: 64; range: from 0 to 128;

nSaturation:

[out] The saturation, default: 64; range: from 0 to 128;

nHue:

[out] The hue, default: 64; range: from 0 to 128;

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.40 Set Image Sharpness PlayM4_SetImageSharpen

BOOL PlayM4_SetImageSharpen(LONG nPort,DWORD nLevel);

Description:

Set the sharpness level of the image.

Parameters:

nPort

[in] Valid port number of the player

nLevel

[in] Sharpness level, range: from 0(disable, by default) to 6 (highest).

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.41 Set Overlay Flip Mode PlayM4_SetOverlayFlipMode

BOOL PlayM4_SetOverlayFlipMode(LONG nPort,BOOL bTrue);

Description:

Set the overlay flip.

Parameters:

nPort

[in] Valid port number of the player

bTrue

[in] TRUE-flip; FALSE-disable flip

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

This API is previously used to enable pre-created buffer for overlay surface, and becomes invalid from V6.1.0.3.

6.42 Set Image Rotation PlayM4_SetRotateAngle

BOOL PlayM4_SetRotateAngle(LONG nPort,DWORD nRegionNum, DWORD dwType)

Description:

Set rotate angle for the image.

Parameters:

nPort

[in] Valid port number of the player

nRegionNum

[in] region number

dwType

[in] Please refer to the following macro definition:

```
#define R_ANGLE_0     0xffffffff     // No rotation
#define R_ANGLE_L90   0              //Rotate left by 90 degrees
#define R_ANGLE_R90   1              // Rotate right by 90 degrees
#define R_ANGLE_180   2              // Rotate 180 degrees
```

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.43 Set Playback Position (Percentage) PlayM4_SetPlayPos

BOOL PlayM4_SetPlayPos (LONG nPort, float fRelativePos);

Description:

Locate the relative playback position of the file (percentage). To get precise locating performance, please create file index before executing this operation, otherwise it can only achieve rough locating.

Parameters:

nPort

[in] Valid port number of the player

fRelativePos

[in] The percent of the file relative position. It is from 0% to 100%.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.44 Get Playback Position (Percentage) PlayM4_GetPlayPos

```
float PlayM4_GetPlayPos (LONG nPort);
```

Description:

Get the relative playback position of file (percentage).

Parameters:

nPort

[in] Valid port number of the player

Return:

The percentage of the file's relative playback position, ranges from 0% to 100%.

6.45 Set Playback Position (ms) PlayM4_SetPlayedTimeEx

```
BOOL PlayM4_SetPlayedTimeEx (LONG nPort, DWORD nTime);
```

Description:

Set the new position in the file in milliseconds for playback. To get precise locating performance, users need to create file index before executing this operation, otherwise it can only achieve rough locating.

Parameters:

nPort

[in] Valid port number of the player

nTime

[in] Start time for playback

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.46 Get Playback Position (ms) PlayM4_GetPlayedTimeEx

```
DWORD PlayM4_GetPlayedTimeEx (LONG nPort);
```

Description:

Get the current playback position of the file in milliseconds.

nPort

[in] Valid port number of the player

Return:

The current playback position of the file (unit: millisecond)

6.47 Set Playback Position (Frame) PlayM4_SetCurrentFrameNum

BOOL PlayM4_SetCurrentFrameNum(LONG nPort, DWORD nFrameNum);

Description:

Specifies the playback position in the file (unit: frames) from the beginning. To get precise locating performance, users need to create file index before executing this operation, otherwise it can only achieve rough locating.

Parameters:

nPort

[in] Valid port number of the player

nFrameNum

[in] The starting frame number for playback

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.48 Get Playback Position (Frame) PlayM4_GetCurrentFrameNum

DWORD PlayM4_GetCurrentFrameNum(LONG nPort);

Description:

Get the current playback position in the file in frame number from the beginning

Parameters:

nPort

[in] Valid port number of the player

Return:

The current frame number for playback

6.49 Image De-flashing PlayM4_SetDeflash

BOOL _stdcall PlayM4_SetDeflash(LONG nPort, BOOL bDeflash);

Description:

This API reduces the image flashing (refreshing) problem under scenario of static image with noise.

Parameters:

nPort

[in] Valid port number of the player

bDeflash

[in] TRUE- enable de-flashing; FALSE- disable de-flashing

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Get Playback or Decoding Information

6.50 Get Time Duration of the File PlayM4_GetFileTime

DWORD PlayM4_GetFileTime (LONG nPort);

Description:

Get total file duration (unit: second).

Parameters:

nPort

[in] Valid port number of the player

Return:

The file's total duration in seconds

6.51 Get Frame Count of the File PlayM4_GetFileTotalFrames

DWORD PlayM4_GetFileTotalFrames (LONG nPort);

Description:

Get the total frame number of the file.

Parameters:

nPort

[in] Valid port number of the player

Return:

The total frame count of the file

6.52 Get Current Frame Rate PlayM4_GetCurrentFrameRate

DWORD PlayM4_GetCurrentFrameRate (LONG nPort);

Description:

Get the current frame rate.

Parameters:

nPort

[in] Valid port number of the player

Return:

The current frame rate

If the frame rate is lower than 1fps, this API will return 0.

6.53 Get Played Time PlayM4_GetPlayedTime

DWORD PlayM4_GetPlayedTime (LONG nPort);

Description:

Get played time count for the current file (unit: second)

Parameters:

nPort

[in] Valid port number of the player

Return:

The played time of the current file, counted from the beginning of the file.

6.54 Get Decoded Frame Count PlayM4_GetPlayedFrames

DWORD PlayM4_GetPlayedFrames (LONG nPort);

Description:

Get the decoded frame number.

Parameters:

nPort

[in] Valid port number of the player

Return:

The decoded frame number of the file

6.55 Get Original Image Size PlayM4_GetPictureSize

BOOL PlayM4_GetPictureSize(LONG nPort,LONG *pWidth,LONG *pHeight);

Description:

Get the original image size of the video.

Parameters:

nPort

[in] Valid port number of the player

LONG *pWidth

[out] original image width, i.e. for CIF/PAL video, the image width is 352

LONG *pHeight

[out] original image height, i.e. for CIF/PAL video, the image height is 288

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

This API gets the size of the latest decoded image. Therefore it shall be called after playback starts to get the precise information. It is suggested to call this API together with PlayM4_SetEncTypeChangeCallback() or PlayM4_SetEncChangeMsg().

6.56 Get File Header Length PlayM4_GetFileHeadLength

DWORD PlayM4_GetFileHeadLength ();

Description:

Get the length of the file header or info header for data exchange purpose.

Parameters:

--

Return:

The size of the file header.

Sample:

CFile m_TestFile;

```
void Start ()
{
    //Get file header length
    DWORD nLength= PlayM4_GetFileHeadLength ();
    PBYTE pFileHead=new BYTE[nLength];
    //Open File
    m_TestFile.Open ("test.mp4 ", CFile: : modeRead, NULL);
    m_TestFile.Read (pFileHead, nLength);
    //Set Stream Mode
    PlayM4_SetStreamOpenMode (0, STREAME_FILE);
    //Open Stream
    if (!PlayM4_OpenStream (0, pFileHead, nLength, 1024*100))
    {
        m_strPlayFileName="";
        MessageBox ("Open File Failure");
    }
    //Playback
    m_bPlaying = PlayM4_Play ( 0, m_hWnd);
    delete []pFileHead;
}
//////////////////////////////////////////////////////////////////////////
void InputData ()
{
    BYTE pBuf[4096];
    m_TestFile.Read (pBuf, sizeof (pBuf));
    while (!PlayM4_InputData (0, pBuf, sizeof (pBuf)))
    {
        if (!m_bPlaying)
            break;//If the playback has already been stopped, exit
        TRACE ("SLEEEP \n");
        Sleep (5);
    }
}
```

6.57 Get Global Timestamp PlayM4_GetSpecialData

DWORD PlayM4_GetSpecialData(LONG nPort);

Description:

Get the global timestamp of current frame.

Parameters:

nPort

[in] Valid port number of the player

Return:

If the function succeeds, the return value is a compressed data for global time (unit: second).

```
#define GET_YEAR(_time_)        (((_time_)>>26) + 2000)
#define GET_MONTH(_time_)       (((_time_)>>22) & 15)
#define GET_DAY(_time_)         (((_time_)>>17) & 31)
#define GET_HOUR(_time_)        (((_time_)>>12) & 31)
#define GET_MINUTE(_time_)      (((_time_)>>6)  & 63)
#define GET_SECOND(_time_)      (((_time_)>>0)  & 63)
```

FALSE - API calling fails.

Decoding Operation & Control

6.58 Set Callback Stream Type PlayM4_SetDecCBStream

BOOL _stdcall PlayM4_SetDecCBStream (LONG nPort, DWORD nStream);

Description:

Set stream type for decoding callback

Parameters:

nPort

[in] Valid port number of the player

nStream

[in]1. video stream; 2.audio stream; 3.video & audio stream

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.59 Set Frame Type PlayM4_SetDecodeFrameType

BOOL PlayM4_SetDecodeFrameType(LONG nPort,DWORD nFrameType);

Description:

Set frame type for video frame decoding

Parameters:

nPort

[in] Valid port number of the player

nFrameType

[in]

#define DECODE_NORMAIL	0 -Decode video frames
#define DECODE_KEY_FRAME	1- Decode key frames
#define DECODE_NONE	2- Do not decode video frames

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.60 Decoder Callback PlayM4_SetDecCallBack

BOOL PlayM4_SetDecCallBack (LONG nPort, void (CALLBACK* DecCBFun) (long nPort, char * pBuf, long nSize, FRAME_INFO * pFrameInfo, long nReserved1, long nReserved2));

Description:

Register a callback function for user-controllable display.

Parameters:

nPort

[in] Valid port number of the player

DecCBFun

[in] Pointer of the decoder callback function, can't be set as NULL.

Description of the Callback Function:

nPort

Valid port number of the player

pBuf

Pointer of the decoded video/audio buffer

nSize

Size of pBuf

pFrameInfo

Pointer of FRAME_INFO structure

nReserved1

Reserved

nReserved2

Reserved

Description of structure FRAME_INFO:

```
typedef struct{
  long nWidth;        //Image width in pixels or sound track number
  long nHeight;       // Image height in pixels or audio bit rate
  long nStamp;        //Time stamp in milliseconds
  long nType;         // Received data type as the Macro Definition below
  long nFrameRate;    //Encoding frame rate or audio sampling rate
}FRAME_INFO;
```

Macro Definition:

T_AUDIO16	PCM Audio, sampling rate: 16 Khz, Mono, 16 bits
T_RGB32	RGB 32 image, 4 bytes per pixel arranged in 'B-G-R-0…' similar as bitmap (starts from the bottom-left of the image) (Reserved)
T_UYVY	uyvy image, arranged in "U0-Y0-V0-Y1-U2-Y2-V2-Y3…." (starts from the bottom-left of the image) format (Reserved)
T_YV12	yv12 image arranged in "Y0-Y1-……""V0-V1….""U0-U1-….." format

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

- Currently only T_YV12 format raw video data is supported.
- This API shall be called before **PlayM4_Play** and will be invalid automatically after **PlayM4_Stop.**
- The decoding function provides no speed control, and the decoding starts whenever the call back function returns. To use this function, user shall understand mechanisms of video & audio display. Please refer to DirectX development package about the relative knowledge.

6.61 Callback with User Data PlayM4_SetDecCallBackMend

BOOL _stdcall PlayM4_SetDecCallBackMend (LONG nPort, void (CALLBACK* DecCBFun) (long nPort, char * pBuf, long nSize, FRAME_INFO * pFrameInfo, long nUser, long nReserved2), long nUser);

Description:

Register a callback function to replace the displayed part. It is controlled by user. It is called back before **PlayM4_Play** and will be invalid automatically when **PlayM4_Stop**. Also it needs to be reset before recalling back **PlayM4_Play**. Please be aware that the decoded part does not control speed, and as user returns from call back function, the decoder will decode the next part of data. To use this function, user must understand video display and audio play. Please refer to directx development package about the relative knowledge.

Parameters:

nPort

[in] Valid port number of the player

DecCBFun

[in] Pointer of the decoder callback function, can't be set as NULL.

Description of the Callback Function:

nPort:

Valid port number of the player

pBuf:

Pointer of the decoded video/audio buffer

nSize:

Size of pBuf

pFrameInfo:

Pointer of FRAME_INFO structure

nUser:

User data

nReserved2:

Reserved

Description of structure FRAME_INFO:

```
typedef struct{
  long nWidth;        //Image width in pixels or sound track number
  long nHeight;       // Image height in pixels or audio bit rate
  long nStamp;        //Time stamp in milliseconds
  long nType;         //Received data type as the Macro Definition below
  long nFrameRate;    //Encoding frame rate or audio sampling rate
}FRAME_INFO;
```

Macro Definition:

T_AUDIO16	PCM Audio, sampling rate: 16 Khz, Mono, 16 bits
T_RGB32	RGB 32 image, 4 bytes per pixel arranged in 'B-G-R-0…' similar as bitmap (starts from the bottom-left of the image) (Reserved)

T_UYVY	uyvy image, arranged in "U0-Y0-V0-Y1-U2-Y2-V2-Y3…." (starts from the bottom-left of the image) format (Reserved)
T_YV12	yv12 image arranged in "Y0-Y1-……""V0-V1….""U0-U1-….." format

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

- Currently only T_YV12 format raw video data is supported.
- This API shall be called before **PlayM4_Play** and will be invalid automatically after **PlayM4_Stop.**
- The decoding function provides no speed control, and the decoding starts whenever the call back function returns. To use this function, user shall understand mechanisms of video & audio display. Please refer to DirectX development package about the relative knowledge.

6.62 Callback with Data & Length PlayM4_SetDecCallBackEx

BOOL PlayM4_SetDecCallBackEx(LONG nPort,void (CALLBACK* DecCBFun)(long nPort,char * pBuf,long nSize,FRAME_INFO * pFrameInfo, long nReserved1,long nReserved2), char* pDest, long nDestSize);

Description:

PlayM4_SetDecCallBackEx() decodes and displays the data, and send the decoding data to the user via callback mode, while PlayM4_SetDecCallBack() decodes but not displays.

The parameter pDest and nDestSize can be set as NULL and 0.

Parameters:

nPort

[in] Valid port number of the player

DecCBFun

[in] Pointer of the decoder callback function, can't be set as NULL.

pDest:

Target data, shall be set as NULL

nDestSize:

Target data size, shall be set as 0

Description of the Callback Function:

nPort:

Valid port number of the player

pBuf:

Pointer of the decoded video/audio buffer

nSize:

Size of pBuf

pFrameInfo:

Pointer of FRAME_INFO structure

nReserved1:

Reserved

nReserved2:

Reserved

Description of structure FRAME_INFO:

```
typedef struct{
  long nWidth;          //Image width in pixels or sound track number
  long nHeight;         // Image height in pixels or audio bit rate
  long nStamp;          //Time stamp in milliseconds
  long nType;           //Received data type as the Macro Definition below
  long nFrameRate;      //Encoding frame rate or audio sampling rate
}FRAME_INFO;
```

Macro Definition:

T_AUDIO16	PCM Audio, sampling rate: 16 Khz, Mono, 16 bits
T_RGB32	RGB 32 image, 4 bytes per pixel arranged in 'B-G-R-0…' similar as bitmap (starts from the bottom-left of the image) (Reserved)
T_UYVY	uyvy image, arranged in "U0-Y0-V0-Y1-U2-Y2-V2-Y3…." (starts from the bottom-left of the image) format (Reserved)
T_YV12	yv12 image arranged in "Y0-Y1-……""V0-V1….""U0-U1-….." format

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.63 Callback with User Data & Data Length

PlayM4_SetDecCallBackExMend

```
BOOL PlayM4_SetDecCallBackExMend(LONG nPort, void (CALLBACK* DecCBFun)(long nPort, char* pBuf, long nSize, FRAME_INFO* pFrameInfo, long nUser, long nReserved2), char* pDest, long nDestSize, long nUser);
```

Description:

PlayM4_SetDecCallBackExMend() decodes and displays the data, and send the decoding data to the user via callback mode, while PlayM4_SetDecCallBackMend() only decodes but not displays.

The parameter pDest and nDestSize can be set as NULL and 0.

Parameters:

nPort

[in] Valid port number of the player

DecCBFun

[in] Pointer of the decoder callback function, can't be set as NULL.

pDest:

Target data, shall be set as NULL

nDestSize:

Target data size, shall be set as 0

nUser:

User parameter

Description of the Callback Function:

nPort:

Valid port number of the player

pBuf:

Pointer of the decoded video/audio buffer

nSize:

Size of pBuf

pFrameInfo:

Pointer of FRAME_INFO structure

nUser:

User data

nReserved2:

Reserved

Description of structure FRAME_INFO:

```
typedef struct{
  long nWidth;        //Image width in pixels or sound track number
  long nHeight;       // Image height in pixels or audio bit rate
  long nStamp;        //Time stamp in milliseconds
  long nType;         //Received data type as the Macro Definition below
  long nFrameRate;    //Encoding frame rate or audio sampling rate
}FRAME_INFO;
```

Macro Definition:

T_AUDIO16	PCM Audio, sampling rate: 16 Khz, Mono, 16 bits
T_RGB32	RGB 32 image, 4 bytes per pixel arranged in 'B-G-R-0…' similar as bitmap (starts from the bottom-left of the image) (Reserved)
T_UYVY	uyvy image, arranged in "U0-Y0-V0-Y1-U2-Y2-V2-Y3…." (starts from the bottom-left of the image) format (Reserved)
T_YV12	yv12 image arranged in "Y0-Y1-……""V0-V1….""U0-U1-….." format

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.64 Set Audio Callback PlayM4_SetAudioCallBack

Notice:

This is a reserved function for future functional expanding and is invalid yet.

```
BOOL _stdcall PlayM4_SetAudioCallBack (LONG nPort, void (_stdcall * funAudio) (long nPort, char * pAudioBuf, long nSize, long nStamp, long nType, long nUser), long nUser);
```

Description:

Register a callback function to refer to the wave format data that have been

decoded out.

Parameters:

nPort

[in] Valid port number of the player

funAudio

the callback function pointer.

nUser

user data.

Description of the Callback Function:

void _stdcall Audio (long nPort, char pAudioBuf, long nSize, long nStamp, long nType, long nUser)*

nPort

Port

pAudioBuf

Wave data

nSize wave

Data size

nStamp

Time stamp (ms)

nType

Audio type: T_AUDIO16: 16 KHz, mono, 16 bit per sampling

nUser

User data

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.65 Set File End Message PlayM4_SetFileEndMsg

BOOL PlayM4_SetFileEndMsg (LONG nPort, HWND hWnd, UINT nMsg);

Description:

Register a windows message which will be post when the file reaches its end. For player SDK version above V2.4, when the file is end, the decoding thread will not stop by itself, and the users will need to call **PlayM4_Stop (nPort) after** received this message to stop the playback**.**

Parameters:

nPort

[in] Valid port number of the player

HWnd

[in] The handle to the window to receive this message.

nMsg

[in] The message is defined by an application. When received this message, it means that the file which is playing is end.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

API PlayM4_SetFileEndMsg and API PlayM4_SetFileEndCallback shall not be called at the same time.

6.66 Set File End Callback PlayM4_SetFileEndCallback

BOOL_stdcall PlayM4_SetFileEndCallback (LONG nPort, void (CALLBACK *FileEndCallback) (long nPort, void *pUser), void *pUser);

Description:

Set callback for decoding file end. This API should be called before PlayM4_OpenSteam () or PlayM4_OpenFile ()

Parameters:

nPort

[in] Valid port number of the player

FileEndCallback

[in] Callback function described below

pUser

[in] Parameter of callback function

Callback function Description:

*void (CALLBACK *FileEndCallback) (long nPort, void *pUser)*

Parameters:

nPort:

Port number

pUser:

User defined parameters, as the last parameter pUser of PlayM4_SetFileEndCallback ().

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

1. For the callback functions, as VB does not support multi-thread, thus problems may occur when calling API defined in VB under VC environment. Please refer to: **Microsoft Knowledge Base Article - Q198607 "PRB: Access Violation in VB Run-Time Using AddressOf "** for detail information.
2. PlayM4_SetFileEndMsg and PlayM4_SetFileEndCallback shall not be called at the same time.

6.67 Notify on Resolution Changing PlayM4_SetEncChangeMsg

BOOL _stdcall PlayM4_SetEncChangeMsg (LONG nPort, HWND hWnd, UINT nMsg);

Description:

Set a notify message when there is change in the encoding resolution

Parameters:

nPort

[in] Valid port number of the player

hWnd

[in] handle of the message window

nMsg

[in] user will receive this message on changing of encoding resolution, the parameter 'WParam' in this message is the returned nPort

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

PlayM4_SetEncChangeMsg and PlayM4_SetEncTypeChangeCallBack shall not be called at the same time.

6.68 Set Resolution Change Callback PlayM4_SetEncTypeChangeCallBack

```
BOOL _stdcall PlayM4_SetEncTypeChangeCallBack (LONG nPort, void (CALLBACK *funEncChange)(long nPort, long nUser), long nUser);
```

Description:

Set a callback function to notify change of encoding resolution

Parameters:

nPort

[in] Valid port number of the player

funEncChange

Callback function;

nUser

User data

Callback Function Descripton:

*void (CALLBACK *funEncChange)(long nPort, long nUser)*

nPort

Valid port number of the player

nUser

User data

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

1. This API shall be called before PlayM4_OpenFile.
2. PlayM4_SetEncChangeMsg and PlayM4_SetEncTypeChangeCallBack shall not be called at the same time.

6.69 Throw B Frame(s) PlayM4_ThrowBFrameNum

BOOL PlayM4_ThrowBFrameNum (LONG nPort, DWORD nNum);

Description:

Set the number of B frame(s) to be skipped during decoding. Skipping of B frame(s) can reduce CPU usage. If there is no B frames in the stream, this function will not take effect. This mechanism can be applied to fast forward or multi-channel playback modes to reduce the PC load.

Parameters:

nPort

[in] Valid port number of the player

nNum

[in] The number of B-frame that is not decoded. It ranges from 0 to 2.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.70 Check Frame Continuity PlayM4_CheckDiscontinuousFrameNum

BOOL _stdcall PlayM4_CheckDiscontinuousFrameNum (LONG nPort, BOOL bCheck);

Description:

Check the continuity of decoding frames. If discontinuity of frames is detected, the decoder will jump to the next I frame.

Parameters:

nPort

[in] Valid port number of the player

bCheck

[in]

TRUE- enable continuity checking and jump to the next I frame when discontinuity detected;

FALSE- disable continuity checking

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

This function is valid in since SDK V6.1.1.17.

6.71 Set Decryption Key PlayM4_SetSecretKey

BOOL __stdcall PlayM4_SetSecretKey (LONG nPort, LONG lKeyType, char *pSecretKey, LONG lKeyLen);

Description:

If a secret key has been set for encryption purpose during the encoding process, then this API shall be called before PlayM4_OpenSteam or PlayM4_OpenFile for decryption.

Parameters:

nPort

[in] Valid port number of the player

lKeyType

[in] secret key type

pSecretKey

[in] secret key string

lKeyLen

[in] key length (unit: bit)

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Display Operation

6.72 Set Overlay Mode and Color Key PlayM4_SetOverlayMode

BOOL PlayM4_SetOverlayMode (LONG nPort, BOOL bOverlay, COLORREF colorKey);

Description:

Set display surface and color key for overlay mode.

Parameters:

nPort

[in] Valid port number of the player

bOverlay

[in]

TRUE- Set the display mode as OVERLAY by default, and try other mode (off-screen) only if OVERLAY fails

FALSE- Do not use OVERLAY mode

colorKey

[in] If bOverlay is set as TRUE, user shall set the color key for overlay surface. User shall paint this color on the display region of overlay to see the image.

This parameter is a DWORD value which shall be represented in 0x00bbggrr mode, the highest digits are set as 0 and the last 3 bytes stands for the value of R/G/B.

As in color keying mode all the other colors penetrate through this key color and displays on overlay surface, it is suggested to set a non-common color in the scenario as the color key.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

Overlay mode is widely supported by different display adapters and it can be applied to get good-quality image and lower CPU usage if the display adapter does not support BLT and color-conversion. However, there is only 1 overlay surface for each display adapter, and if it is occupied by other Apps (i.e. common player Apps, live view of compression card), then the player cannot use overlay mode and will switch to off-screen mode automatically. This API will not return FALSE in this case.

6.73 Display Mode Query PlayM4_GetOverlayMode

LONG PlayM4_GetOverlayMode (LONG nPort);

Description:

Check if the player is in OVERLAY mode or not.

Parameters:

nPort

[in] Valid port number of the player

Return:

-1: API calling failure

0: Off-screen mode

1: Overlay mode

6.74 Get Overlay Color Key PlayM4_GetColorKey

COLORREF PlayM4_GetColorKey (LONG nPort);

Description:

Get the color key of the OVERLAY surface.

Parameters:

nPort

[in] Valid port number of the player

Return:

The value of the color key- a DWORD value represented in 0x00bbggrr mode, in which the last 3 bytes stands for the value of R/G/B.

6.75 Set/Add Display Region PlayM4_SetDisplayRegion

BOOL _stdcall PlayM4_SetDisplayRegion (LONG nPort, DWORD nRegionNum, RECT *pSrcRect, HWND hDestWnd, BOOL bEnable);

Description:

Set or add display regions, also applies to partial enlargement.

Parameters:

nPort

[in] Valid port number of the player

nRegionNum

[in] Display region number from 0 to (MAX_DISPLAY_WND-1). If nRegionNum is set as 0, it stands for setting of the main display window (window set in PlayM4_Play) and hDestWnd and bEnable settings will not be handled in this mode.

pSrcRect

[in] Set the region to be displayed (this region shall be inside the original image), i.e. if the resolution of original image is 352*288, the range of pSrcRect shall not exceed (0, 0, 352, 288).

If pSrcRect is set as NULL, the whole image will be displayed.

hDestWnd

[in] Set the display window. If the window for this region has already been set or opend, then this parameter will be neglected.

bEnable

[in] Open /set or close the display region.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.76 Refresh Display PlayM4_RefreshPlay

BOOL PlayM4_RefreshPlay (LONG nPort) ;

Description:

Refresh/retrieve image display after the refreshing of display window under PAUSE status. This API is only valid under pause and step forward status, otherwise it will return directly.

Parameters:

nPort

[in] Valid port number of the player

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.77 Refresh Display for Multiple Regions PlayM4_RefreshPlayEx

```
BOOL _stdcall PlayM4_RefreshPlayEx (LONG nPort, DWORD nRegionNum);
```

Description:

Refresh display for multiple display regions.

Parameters:

nPort

[in] Valid port number of the player

nRegionNum

[in]display region serial number.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.78 Set Normal/Quarter Display Mode PlayM4_SetDisplayType

```
BOOL PlayM4_SetDisplayType (LONG nPort, LONG nType);
```

Description:

Switch between DISPLAY_NORMAL and DISPLAY_QUARTER modes. DISPLAY_QUARTER mode can reduce the work load of display adapter in the cost of image quality, and is suitable to apply in small-window-size viewing mode. For normal display or single-channel viewing, usually it's better to apply DISPLAY_NORMAL mode.

Parameters:

nPort

[in] Valid port number of the player

nType

[in]

DISPLAY_NORMAL Send the decoded data to display adapter normally

DISPLAY_QUARTER Transmit 1/4 resolution data to display adapter

DISPLAY_YC_SCALE YC extension

DISPLAY_NOTEARING No display tearing effect

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

The parameter nType can not be set both 1 and 2,the other can be any combination. The parameter nType is representation by bit,the 9 is 1 and 8 at the same time by setting.

6.79 Normal/Quarter Display Mode Query PlayM4_GetDisplayType

long PlayM4_GetDisplayType (LONG nPort);

Description:

Check the current display mode.

Parameters:

nPort

[in] Valid port number of the player

Return:

DISPLAY_NORMAL or DISPLAY_QUARTER

Notice:

The nType is a combination.

Source Buffer Operation

** Source buffer is the buffer for the data to be decoded*

6.80 Free Space Query PlayM4_GetSourceBufferRemain

DWORD PlayM4_GetSourceBufferRemain (LONG nPort);

Description:

Get the free space information of the source buffer.

Parameters:

nPort

[in] Valid port number of the player

Return:

Free space of the source buffer (unit: Byte).

6.81 Threshold Setting & Callback PlayM4_SetSourceBufCallBack

BOOL PlayM4_SetSourceBufCallBack (LONG nPort, DWORD nThreShold, void (CALLBACK * SourceBufCallBack) (long nPort, DWORD nBufSize, DWORD dwUser, void*pResvered), DWORD dwUser, void *pReserved);

Description:

Set a threshold for the free space of source buffer, and register a notification callback when the free space falls below the threshold.

PlayM4_ResetSourceBufFlag() shall be called after each triggering of threshold notification callback to re-enable this callback notification again.

Parameters:

nPort

[in] Valid port number of the player

nThreShold

[in] Threshold for the free space of source buffer

SourceBufCallBack

[in] pointer of the callback function

dwUser

[in] user data

pResvered

[in] reserved

Description of the Callback Function:

*void CALLBACK SourceBufCallBack (long nPort, DWORD nBufSize, DWORD dwUser, void*pContext);*

Parameters:

nPort

Valid port number of the player

nBufSize

Number of bytes remained of the source buffer.

dwUser

user data .

pResvered

reserved.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

PlayM4_ResetSourceBufFlag() shall be called after each triggering of threshold notification callback to re-enable this callback notification again.

6.82 Reset Threshold Callback PlayM4_ResetSourceBufFlag

BOOL PlayM4_ResetSourceBufFlag (LONG nPort);

Description:

Re-enable the callback of Source Buffer Threshold. Every time the source buffer threshold notification is triggered, this API shall be called afterwards to reset the buffer threshold detecting status and re-enable callback notification again.

Parameters:

nPort

[in] Valid port number of the player

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Decode Buffer Operation

**Decode buffer is the buffer for the decoded data*

6.83 Set Buffering Size PlayM4_SetDisplayBuf

BOOL PlayM4_SetDisplayBuf (LONG nPort, DWORD nNum);

Description:

Set the buffer size for playback (buffer of decoded images). This buffer is directly related with the fluency and real-time performance of playback. Larger buffer size usually means higher fluency yet longer time delay, and thus it is suggested to set larger buffer size for OpenFile mode (if the system memory is large enough). The default buffer size would be 15 (frames) for Open File mode, which is, about 0.6 sec under cases of 25fps; and 10 (frames) for Open Stream mode.

Parameters:

nPort

[in] Valid port number of the player

nNum

[in] The number of frames to be buffered, range: from MIN_DIS_FRAMES to MAX_DIS_FRAMES

MIN_DIS_FRAMES

The minimum number of image frames to be buffered.

MAX_DIS_FRAMES

The maximum number of image frames to be buffered.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

This function shall be called between PlayM4_OpenStream and PlayM4_Play.

6.84 Buffering Size Query PlayM4_GetDisplayBuf

```
DWORD PlayM4_GetDisplayBuf (LONG nPort);
```

Description:

Get the buffering size (unit: frame number) of the playback buffer.

Parameters:

nPort

[in] Valid port number of the player

Return:

The maximum number of image frames to be buffered.

Source Buffer & Decode Buffer Operation

6.85 Clear All Buffer PlayM4_ResetSourceBuffer

```
BOOL PlayM4_ResetSourceBuffer (LONG nPort);
```

Description:

Clear the data in all the buffers.

Parameters:

nPort

[in] Valid port number of the player

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.86 Clear Specified Buffer PlayM4_ResetBuffer

```
BOOL _stdcall PlayM4_ResetBuffer (LONG nPort, DWORD nBufType);
```

Description:

Clear the data in a specified buffer.

Parameters:

nPort

[in] Valid port number of the player

nBufType

[in]

BUF_VIDEO_SRC: Video source buffer, valid for stream mode.

BUF_AUDIO_SRC: Audio source buffer, valid for video/audio separate stream mode.

BUF_VIDEO_RENDER: Video decode buffer.

BUF_AUDIO_RENDER: Audio decode buffer.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.87 Buffer Info Query PlayM4_GetBufferValue

DWORD _stdcall PlayM4_GetBufferValue (LONG nPort, DWORD nBufType);

Description:

Get the data size of specified buffer.

Parameters:

nPort

[in] Valid port number of the player

nBufType

[in]

BUF_VIDEO_SRC: Video source buffer, valid for stream mode (unit: Byte)

BUF_AUDIO_SRC: Audio source buffer, valid for video/audio separate stream mode (unit: Byte)

BUF_VIDEO_RENDER: Video display buffer total num(unit: Frame)

BUF_AUDIO_RENDER: Audio render buffer total num(unit: Frame)

BUF_VIDEO_DECODED:data for video decode buffer (**0~BUF_VIDEO_RENDER**)

BUF_AUDIO_DECODED:data for audio decode buffer (unit: Frame,)

(0~BUF_AUDIO_DECODED)

Return:

Current data length in the specified buffer

File Index

6.88 Set File Index Callback PlayM4_SetFileRefCallBack

BOOL PlayM4_SetFileRefCallBack (LONG nPort,
void (_stdcall *pFileRefDone) (DWORD nPort, DWORD nUser), DWORD nUser);

Description:

Register a callback function to notify the user when the key frame index for the file is created. If the callback is not triggered, it indicates errors in the file.

The file index is used for fast locating in the file. The indexing speed is about 40MB per second. All the Index-related APIs shall be called after indexing, and the other APIs will not be affected.

Parameters:

nPort

[in] Valid port number of the player

SetFileRefCallBack

[in] pointer of callback function

nUser

[in] user data

Description of the Callback Function:

void FileRefDone (DWORD nPort, DWORD nUser)

Parameters:

nPort

Valid port number of the player

nUser

User data

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.89 Locate Previous Key Frame PlayM4_GetKeyFramePos

BOOL PlayM4_GetKeyFramePos (LONG nPort, DWORD nValue, DWORD nType, PFRAME_POS pFramePos);

Description:

Locate the nearest key frame before the designated position. As the decoding process shall start from a key frame, and any data before the first key frame will be discarded, users shall start clip/playback from a key frame (it doesn't matter much if it ends with key frame or not, and the maximum frame loss number at the file end position is 3).

Parameters:

nPort

[in] Valid port number of the player

nValue

[in] User-specified position

nType

[in] How the position is presented, either BY_FRAMENUM or BY_FRAMETIME

pFramePos

[out] Pointer of FRAME_POS structure

Structure Description:

```
typedef struct{
    long nFilePos;          //Nearest Key Frame Position in File
    long nFrameNum;         // Frame number of nearest key frame
    long nFrameTime;        //The time stamp of nearest key frame (unit: ms).
}FRAME_POS, *PFRAME_POS;
```

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

PlayM4_GetKeyFramePos and PlayM4_GetNextKeyFramePosition can be called to achieve video clip function after the file index has been successfully created. The clip precision is related with key frame interval of the original file.

6.90 Locate Next Key Frame PlayM4_GetNextKeyFramePos

BOOL PlayM4_GetNextKeyFramePos (LONG nPort, DWORD nValue, DWORD nType, PFRAME_POS pFramePos);

Description:

Get the position of a key frame after the designated position.

Parameters:

nPort

[in] Valid port number of the player

nValue

[in] User-specified position

nType

[in] How the position is presented, either BY_FRAMENUM or BY_FRAMETIME

pFramePos

[out] Pointer of FRAME_POS structure

Structure Description:

```
typedef struct{
    long nFilePos;          //Nearest Key Frame Position in File
    long nFrameNum;         // Frame number of nearest key frame
    long nFrameTime;        //The time stamp of nearest key frame (unit: ms).
}FRAME_POS, *PFRAME_POS;
```

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

PlayM4_GetKeyFramePos and PlayM4_GetNextKeyFramePosition can be called to achieve video clip function after the file index has been successfully created. The clip precision is related with key frame interval of the original file.

6.91 Get File Index Info PlayM4_GetRefValue

BOOL _stdcall PlayM4_GetRefValue (LONG nPort, BYTE *pBuffer, DWORD *pSize);

Description:

Get the file index information. User must call this after the file index has been created.

Parameters:

nPort

[in] Valid port number of the player

pBuffer

[in] The buffer of index information

pSize

[in/out] Input the pBuffer size and output the index size, i.e. on first time calling, users can set *pSize=0; pBuffer=NULL;* and get the buffer size needed from the return valud of pSize, then re-call this API after assigning enough buffer size.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.92 Set File Index PlayM4_SetRefValue

BOOL _stdcall PlayM4_SetRefValue (LONG nPort, BYTE *pBuffer, DWORD nSize);

Description:

If the file index has already been created, user can input it directly via this API and does not need to call PlayM4_SetFileRefCallBack.

Parameters:

nPort

[in] Valid port number of the player

pBuffer

[in]The file index information

nSize

[in]The size of the index information

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

This API shall be called after PlayM4_OpenFile, and please make sure that the length and content of the file index is correct.

Multi-screen Playback

Notice:

The Following APIs in this section are specially added for multi display adapters support. Only operation systems with Windows98, Windows2000 and Windows 2000 support multi display adapters and need installing DirectX6.0 or more advanced edition. If user needn't environment of supporting multi display adapters, these interfaces are dismissal. With regards to multi display adapters, please consult correlative file "Multiple-Monitor Systems" of Microsoft sdk.

SDK version above V6.1.1.0 is self-adaptive to multi-screen display, and users do not need to call the APIs in this section.

6.93 Enum the Display Devices in the System PlayM4_InitDDrawDevice

BOOL PlayM4_InitDDrawDevice();

Description:

Enumerate display devices of system

Parameters:

--

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.94 Release Display Device Resource PlayM4_ReleaseDDrawDevice

void PlayM4_ReleaseDDrawDevice ();

Description:

Release resource distributed in the course of enumerating display devices

6.95 Get Display Adapter Number PlayM4_GetDDrawDeviceTotalNums

DWORD PlayM4_GetDDrawDeviceTotalNums ();

Description:

Get the total number of display device that are attached to the desktop.

Return:

If return 0, it indicates that there is only main displayed device in system. If 1, it indicates that there are many video adapters in system but only one is tied to windows desktop. If return other value, it indicates the number of video adapter tied to desktop of system. In the system with many video adapter, user can designate any video adapter as main display device via setting display attribution.

6.96 Assign Display Adapter for the Monitor PlayM4_SetDDrawDevice

BOOL PlayM4_SetDDrawDevice (LONG nPort, DWORD nDeviceNum);

Description:

Assign display adapter for the monitor. Notice: The display region should be set consistently.

Parameters:

nPort

[in] Valid port number of the player

nDeviceNum

[in] The number of the display device. It ranges from 0 to the return value of PlayM4_GetDDrawDeviceTotalNums. If 0, it means the primary display device.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.97 Assign Display Adapter for Multiple Monitors PlayM4_SetDDrawDevice_EX

BOOL _stdcall PlayM4_SetDDrawDeviceEx (LONG nPort, DWORD nRegionNum, DWORD nDeviceNum);

Description:

Set display adapter used in play window, as 62. It is an added parameter set for PlayM4_SetDisplayRegion.

Parameters:

nPort

[in] Valid port number of the player

nRegionNum

[in]display region serial No.

[in]The video adapter No.

nDeviceNum

[in] The number of the display device. It ranges from 0 to the return value of PlayM4_GetDDrawDeviceTotalNums. If 0, it means the primary display device.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.98 Get the Display Adapter and Monitor Info PlayM4_GetDDrawDeviceInfo

BOOL PlayM4_GetDDrawDeviceInfo (DWORD nDeviceNum, LPSTR lpDriverDescription, DWORD nDespLen, LPSTR lpDriverName , DWORD nNameLen, HMONITOR *hhMonitor);

Description:

Get the information of the display device and monitor specified by nDeviceNum.

Parameters:

nDeviceNum

[in]The number of the display device. If it is 0, It means the primary display device.

nDespLen

[in]The number of bytes of the lpDriverDescription that has been allocated.

nNameLen

[in] The number of bytes of the lpDriverName that has been allocated.

lpDriverDescription

[out] Address of a string that contains the driver description.

lpDriverName

[out] Address of a string that contains the driver name.

hhMonitor

[out] Handle of the monitor associated with the enumerated DirectDraw object. This parameter is NULL when the enumerated DirectDraw object is for the primary device, a nondisplay device (such as a 3-D accelerator with no 2-D capabilities), or devices not attached to the desktop. For more information, see the function of GetMonitorInfo (Windows API). Notice: the type of HMONITOR is defined in the header file “windef.h” (_WIN32_WINNT >= 0x0500). Please download and install the new Platform sdk. (http: //www.microsoft.com/msdownload/platformsdk/sdkupdate/)

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.99 Get System Info of Display Devices PlayM4_GetCapsEx

```
int  PlayM4_GetCapsEx (DWORD nDDrawDeviceNum);
```

Description:

Get some capabilities of your system.

Parameters:

nDeviceNum

[in]The number of the display device. If it is 0, It means the primary display device.

Return:

SUPPORT_DDRAW

Support DIRECTDRAW. If not, player can’t work.

SUPPORT_BLT

Support BLT operation. If not, player can’t work.

SUPPORT_BLTFOURCC

Display hardware is capable of color-space conversions during blit operations.

SUPPORT_BLTSHRINKX

Supports arbitrary shrinking along the x-axis (horizontally), this flag is valid only for BLT operations.

SUPPORT_BLTSHRINKY

Supports arbitrary shrinking along the y-axis (vertically), this flag is valid only for BLT operations.

SUPPORT_BLTSTRETCHX

Supports arbitrary stretching of a surface along the x-axis (horizontally), this flag is valid only for blit operations.

SUPPORT_BLTSTRETCHY

Supports arbitrary stretching of a surface along the y-axis (vertically), this flag is valid only for blit operations.

SUPPORT_SSE

Supports SSE instruction set, if supports, It will get high performance.

SUPPORT_MMX

Supports MMX instruction set.

Snapshot

6.100 Snapshot Callback PlayM4_SetDisplayCallBack

BOOL PlayM4_SetDisplayCallBack (LONG nPort, void (CALLBACK* DisplayCBFun) (long nPort, char * pBuf, long nSize, long nWidth, long nHeight, long nStamp, long nType, long nReserved));

Description:

Register a callback function for image capturing.

Parameters:

nPort

[in] Valid port number of the player

DisplayCBFun

[in] Snapshot callback function

Description of the Callback Function:

nPort

Valid port number of the player

pBuf

Pointer of the snapshot buffer

nSize

Size of the snapshot buffer

nWidth

Width of the image (unit: pixels)

nHeight

Height of the image (unit: pixels)

nStamp

Time stamp (unit: ms)

nType

Received data type: T_YV12, T_RGB32, T_UYVY, please refer to the macro definition of PlayM4_SetDecCallback() for detail information.

nReserved

Reserved;

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

The callback shall return ASAP, as the callback is triggered in clock thread and any time-consuming operation will affect the clock pulse and cause display problems. Users can stop the callback by setting DisplayCDFun as NULL.

This API can be called at any time, and it will remain valid until application ends.

6.101 Snapshot with User Data PlayM4_SetDisplayCallBack

BOOL PlayM4_SetDisplayCallBackEx(LONG nPort,void (CALLBACK* DisplayCBFun)(DISPLAY_INFO

*pstDisplayInfo), long nUser);

Description:

Register a callback function for image capturing, this API is an extended version compared to PlayM4_SetDisplayCallback() which contains user data.

Parameters:

nPort

[in] Valid port number of the player

DisplayCBFun

[in] Snapshot callback function

nUser

[in] User data

Description of the Callback Function:

nPort

[in] Valid port number of the player

pstDisplayInfo

Pointer of DISPLAY_INFO structure

Structure Definition:

```
typedef struct
{
    long    nPort;      //Valid port number of the player
    char * pBuf;        //Pointer of the snapshot buffer
    long    nBufLen;    //Size of the snapshot buffer
    long    nWidth;     // Width of the image (unit: pixels)
    long    nHeight;    // Height of the image (unit: pixels)
    long    nStamp;     // Time stamp (unit: ms)
    long    nType;      //Received data type: T_YV12, T_RGB32, T_UYVY, please refer to the macro definition of PlayM4_SetDecCallback() for detail information.
    long    nUser;      //User Data
}DISPLAY_INFO
```

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

The callback shall return ASAP, as the callback is triggered in clock thread and any time-consuming operation will affect the clock pulse and cause display problems. Users can stop the callback by setting DisplayCDFun as NULL.

This API can be called at any time, and it will remain valid until application ends.

6.102 Convert YV12 Image to BMP File PlayM4_ConvertToBmpFile

BOOL PlayM4_ConvertToBmpFile (char * pBuf, long nSize, long nWidth, long nHeight, long nType, char *sFileName);

Description:

Convert the YV12 image data (from either DecCallback or DisplayCallback) to a BMP file.

Parameters:

pBuf

Pointer of the snapshot buffer

nSize

snapshot size

nWidth

width of the image (unit: pixels)

nHeight

height of the image (unit: pixels)

nType

Received data type: T_YV12, T_RGB32, T_UYVY, please refer to the macro definition of PlayM4_SetDecCallback() for detail information.

sFileName

The BMP file name for saving

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

The YV12-RGB converting process takes CPU resource from PC.

6.103 Convert YV12 Image to JPEG File PlayM4_ConvertToJpegFile

BOOL _stdcall PlayM4_ConvertToJpegFile (char *pBuf, long nSize, long nWidth, long nHeight, long nType, char *sFileName);

Description:

Convert the YV12 image to a jpeg file.

Parameters:

pBuf

Pointer of the snapshot buffer

nSize

snapshot size

nWidth

width of the image (unit: pixels)

nHeight

height of the image (unit: pixels)

nType

Received data type: T_YV12, T_RGB32, T_UYVY, please refer to the macro definition of PlayM4_SetDecCallback() for detail information.

sFileName

The JPEG file name for saving

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

The converting process takes CPU resource of the PC. If the image width or height is not a multiple of 16, the snapshot image resolution will be automatically cropped to multiple of 16. i.e. original image of 176*120 will be cropped to 176*112.

6.104 BMP Snapshot PlayM4_GetBMP

BOOL _stdcall PlayM4_GetBMP (LONG nPort, PBYTE pBitmap, DWORD nBufSize, DWORD*pBmpSize);

Description:

Get a snapshot in BMP format

Parameters:

nPort

[in] Valid port number of the player

pBitmap

[in] Address assigned by users for storing BMP snapshot, the file size shall be no less then the bmp file size, which is sizeof (BITMAPFILEHEADER) + sizeof (BITMAPINFOHEADER) + w * h * 4, in which w and h stand for the width and height of the image.

nBufSize

[in] Buffer size

pBmpSize

[out] Actual BMP size captured

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.105 JPEG Snapshot PlayM4_GetJPEG

BOOL _stdcall PlayM4_GetJPEG (LONG nPort, PBYTE pJpeg, DWORD nBufSize, DWORD*pJpegSize);

Description:

Get a snapshot in JPEG format

Parameters:

nPort

[in] Valid port number of the player

pJpeg

[in] [in] Address assigned by users for storing JPEG snapshot, the file size shall be no less then the JPEG file size, suggested value: w * h * 3/2, in which w and h stand for the width and height of the image.

nBufSize

[in] assigned buffer size

pBmpSize

[out]Actual Jpeg image size captured.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

If the image width or height is not a multiple of 16, the snapshot image resolution will be automatically cropped to multiple of 16. i.e. original image of 176*120 will be cropped to 176*112.

6.106 Set JPEG Snapshot Quality PlayM4_SetJpegQuality

BOOL _stdcall PlayM4_SetJpegQuality(long nQuality);

Description:

Set the quality of JPEG snapshots.

Parameters:

nQuality

[in] the quality of JPEG file, range: from 0<lowest quality> to 100 <highest quality> (80 by default).

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

1. This API shall be called before JPEG snapshot, and is suggested to be called before PlayM4_OpenFile().
2. Suggested quality level: [75, 90].

6.107 JPEG Snapshot in Cropped Region PlayM4_GetCropJPEG

BOOL PlayM4_GetCropJPEG(long nPort,CROP_PIC_INFO *pstPicInfo)

Description:

Get a snapshot of cropped region in JPEG format

Parameters:

nPort

[in] Valid port number of the player

pstPicInfo

[in] Pointer of CROP_PIC_INFO structure which describes the cropped area

Structure Description of CROP_PIC_INFO:

```
typedef struct
{
BYTE *pDataBuf;          //Buffer of the snapshot data
DWORD dwPicSize;         //Actual snapshot size
DWORD dwBufSize;         //Buffer size
DWORD dwPicWidth;        //Width of the snapshot
```

DWORD dwPicHeight; ***//Height of the snapshot***
DWORD dwReserve; ***//Reserved***
*RECT *pCropRect;* ***// cropped region***
}CROP_PIC_INFO;

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

If the image width or height is not a multiple of 16, the snapshot image resolution will be automatically cropped to multiple of 16. i.e. original image of 176*120 will be cropped to 176*112.

6.108 BMP Snapshot in Cropped region PlayM4_GetCropBMP

BOOL PlayM4_GetCropBMP(long nPort,CROP_PIC_INFO *pstPicInfo)

Description:

Get a snapshot of cropped region in JPEG format

Parameters:

nPort

[in] Valid port number of the player

pstPicInfo

[in] Pointer of CROP_PIC_INFO structure which describes the cropped area

Structure Description of CROP_PIC_INFO:

typedef struct
{
*BYTE *pDataBuf;* ***//Buffer of the snapshot data***
DWORD dwPicSize; ***//Actual snapshot size***
DWORD dwBufSize; ***//Buffer size***
DWORD dwPicWidth; ***//Width of the snapshot***
DWORD dwPicHeight; ***//Height of the snapshot***
DWORD dwReserve; ***//Reserved***
*RECT *pCropRect;* ***// Crop region***
}CROP_PIC_INFO;

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

If the image width or height is not a multiple of 16, the snapshot image resolution will be automatically cropped to multiple of 16. i.e. original image of 176*120 will be cropped to 176*112.

Others

6.109 DirectDraw Surface Callback PlayM4_RegisterDrawFun

BOOL _stdcall PlayM4_RegisterDrawFun (LONG nPort, void (CALLBACK* DrawFun) (long nPort, HDC hDc, LONG nUser), LONG nUser);

Description:

Register a callback function to get the device context of DirectDraw off-screen surface, and get self- control of the display on this DC.

Parameters:

nPort

[in] Valid port number of the player

DrawFun

[in] pointer of the callback function

nUser

[in] User data

Description of the Callback Function:

voio CALLBACK DrawFun (long nPort, HDC hDc, LONG nUser);

Parameters:

nPort

[out] Valid port number of the player

hDc

[out] The off-screen surface DC.

nUser

[out] The user data.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

This API is invalid for the overlay surface, users can draw on the window directly under overlay mode and get penetrated via overlay color key.

6.110 DirectDraw Surface Callback PlayM4_RigisterDrawFun

BOOL _stdcall PlayM4_RigisterDrawFun (LONG nPort, void (CALLBACK* DrawFun) (long nPort, HDC hDc, LONG nUser), LONG nUser);

Description:

This API functions the same as 6.105 PlayM4_RegisterDrawFun, and is reserved for downward-compatibility.

Parameters:

nPort

[in] Valid port number of the player

DrawFun

[in] pointer of the callback function

nUser

[in] User data

Description of the Callback Function:

void CALLBACK DrawFun (long nPort, HDC hDc, LONG nUser);

Parameters:

nPort

[out] Valid port number of the player

hDc

[out] The off-screen surface DC.

nUser

[out] The user data.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

This API is invalid for the overlay surface. Users can draw on the window directly under overlay mode and get penetrated via overlay color key.

6.111 Set Data Verify Callback PlayM4_SetVerifyCallBack

```
BOOL _stdcall PlayM4_SetVerifyCallBack (LONG nPort, DWORD nBeginTime, DWORD nEndTime, void (_stdcall* funVerify) (long nPort, FRAME_POS * pFilePos, DWORD bIsVideo, DWORD nUser), DWORD nUser);
```

Notice:

This API is reserved for future functional expanding and is not valid yet.

Description:

Register a callback function when data verify error is detected. Can be used as watermark or data-loss detect.

Parameters:

nPort

[in] Valid port number of the player

nBeginTime

verify start time (ms);

nEndTime

verify stop time (ms);

funVerify

the callback function pointer;

nUser

user data.

Description of the Callback Function:

```
void _stdcall Verify (long nPort, FRAME_POS * pFilePos, DWORD bIsVideo, DWORD nUser);
```

nPort

port

pFilePos

file position.

bIsVideo

1-video data, 0-audio data

nUser

user data.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

1. This API shall be called before PlayM4_OpenFile().
2. This API shall be registered before file index creating as the verify is executed during indexing process.

6.112 Get Original Frame Callback PlayM4_GetOriginalFrameCallBack

```
BOOL _stdcall PlayM4_GetOriginalFrameCallBack (LONG nPort, BOOL bIsChange, BOOL bNormalSpeed, long nStartFrameNum, long nStartStamp, long nFileHeader, void (CALLBACK *funGetOrignalFrame) (long nPort, FRAME_TYPE *frameType, long nUser), long nUser);
```

Notice:

This API is reserved and not valid yet.

Description:

Create callback function to get the original frame data. You can change the time stamp and no. of each frame. The API is used after the file is opened. Used to combine two files.

Parameters:

nPort

[in] Valid port number of the player

bIsChange

[in] Change the parameters of each frame or not

bNormalSpeed

[in] Get the original frame at normal speed or not

nStartFrameNum

[in] If the user wants to change the original frame number, this is the start frame number of new file.

nStartStamp

[in] If the user wants to change the original frame time stamp, this is the start time stamp of the new file.

nFileHeader

[in] The version information of the file header, if the version is not matched, it will return failure.

Description of callback function:

void (CALLBACK *funGetOrignalFrame) (long nPort, FRAME_TYPE *frameType, long nUser)

nPort:

Channel number;

frameType:

data frame information

```
typedef struct{
    char *pDataBuf;                 //the start address of data frame
    long nSize;                     //frame size
    long nFrameNum;                 //frame number
    BOOL bIsAudio;                  //is audio frame or not
long nReserved;
}FRAME_TYPE;
```

nUser:

user data.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.113 Get File Attributes PlayM4_GetFileSpecialAttr

BOOL _stdcall PlayM4_GetFileSpecialAttr (LONG nPort, DWORD *pTimeStamp, DWORD *pFileNum, DWORD *nFileHeader);

Notice:

This API is reserved and not valid yet.

Description:

Get the file last frame number and time stamp. Used after the file is opened and combine with front file.

Parameters:

nPort

[in] Valid port number of the player

pTimeStamp: [out] the file end time stamp;

pFileNum: [out] the file end frame number;

nfileHeader: [out] the file header information.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.114 Jump to the Next Key Frame on Error PlayM4_PlaySkipErrorData

BOOL PlayM4_PlaySkipErrorData(LONG nport, BOOL bSkip);

Description:

Users can enable this function to avoid blurring or other display problems caused by decoding data error.

Parameters:

nPort

[in] Valid port number of the player

bSkip

[in]

TRUE: jump to the next key frame if there is error in video data;

FALSE: continue decoding from the next frame if there is error in video data

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Reverse Playback

Notice:

- Reverse playback is an expanded function which is supported from V6.3.0 SDK.
- Valid for HIKVISION baseline devices only
- Reverse playback under 1280*960: key frame interval shall not exceed 100
- Reverse playback above 1280*960: key frame interval shall not exceed 25
- Reverse playback supports up to 2X playback speed.
- Reverse playback under D1 resolution: up to 8X playback speed
- Reverse playback above D1 resolution: up to 2X playback speed
- Invalid for files with error in frame No. or timestamp
- OpenFile mode: call API PlayM4_ReversePlay() to switch between normal/reverse playback mode. The default start position for reverse playback is the start of file.
- OpenStream mode:GOPs shall be sent reversely.
- OpenStream mode: After sending of the last GOP, which is the end of reverse playback, PlayM4_InputData(port, NULL, -1) shall be called to ensure the decoding of the last GOP.
- Step backward is not supported in stream mode

6.115 Start Reverse Playback PlayM4_ReversePlay

```
BOOL PlayM4_ReversePlay(LONG nPort);
```

Description:

Start reverse playback.

Parameters:

nPort

[in] Valid port number of the player

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Notice:

This function shall be called after PlayM4_Play().

This function shall be called after PlayM4_SetFileRefCallBack().

6.116 Get Global Timestamp of the Frame PlayM4_GetSystemTime

```
BOOL PlayM4_GetSystemTime(LONG nPort, PLAYM4_SYSTEM_TIME *pstSystemTime);
```

Description:

Get the global timestamp of the current decoding frame.

Parameters:

nPort

[in] Valid port number of the player

pstSystemTime

[out] Structure PLAYM4_SYSTEM_TIME

```
PLAYM4_SYSTEM_TIME {
    DWORD dwYear;       //year
    DWORD dwMon;        //month
    DWORD dwDay;        //date
    DWORD dwHour;       //hour
    DWORD dwMin;        //minute
    DWORD dwSec;        //second
    DWORD dwMs;         //millisecond
}
```

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

6.117 Set the Start Frame of Reverse Stream PlayM4_SetPlayedTimeEx

```
BOOL PlayM4_SetPlayedTimeEx(LONG nPort,DWORD nTime);
```

Description:

Set the starting frame of reverse playback under OpenStream mode.

Parameters:

nPort

[in] Valid port number of the player

nTime

[in] The starting frame of reverse playback, which is the relative timestamp compared to the starting I frame in the 1st reverse playback GOP, i.e. nTime = 0 stands for starting from the 1st I frame in the 1st GOP; and nTime = 40 under 25fps encoding resolution stands for starting from the following frame of the 1st I frame (as 40ms is the frame interval under 25fps mode) in the 1st GOP.

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.

Synchronized Playback

6.118 Synchronized Playback PlayM4_SetSycGroup

BOOL PlayM4_SetSycGroup(LONG nPort, DWORD dwGroupIndex)

Description:

This API shall be called before PlayM4_Play and each dwGroupIndex is corresponding to 1 synchronizing group. Users can call this API multiple times to add target nPort to the synchronizing group and achieve synchronizing playback. Each synchronizing group supports up to 16 nPort channels, otherwise the synchronization will fail and API PlayM4_FreePort shall be called to release the target nPort from the synchronizing group.

Parameters:

nPort

[in] Valid port number of the player that needs to be added to the synchronizing group

dwGroupIndex

[in] Index number for the synchronizing group

Return:

TRUE - API calling succeeds;

FALSE - API calling fails.