Player SDK
Programmer Manual
(For iOS)

Version 7.0.2.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 7.0.2.x

Addition:

1. Support MPEG2 decoding of Video;
2. Support G726,amr,mpeg,aac decoding of Audio;
3. Optimize the decoding of standard H.264;

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_FAIL_UNKNOWN	99	Unknown error

Chapter 4 Display Description

Notes:

1. DirectDraw is the image under the platform of windows display technology.
   It is not support under iOS platform:
   PlayM4_InitDDraw ,PlayM4_RealeseDDraw ,
   PlayM4_InitDDrawDevice,PlayM4_ReleaseDDrawDevice,
   PlayM4_SetDDrawDevice,PlayM4_SetDDrawDeviceEx,
   PlayM4_GetDDrawDeviceTotalNums,PlayM4_GetDDrawDeviceInfo,
   PlayM4_GetCaps,PlayM4_GetCapsEx,PlayM4_SetDDrawDeviceEx；
2. Not yet implemented iOS platform overlay rendering technology
   PlayM4_GetOverlayMode,PlayM4_SetOverlayFlipMode,PlayM4_SetOverlayMode,
   PlayM4_GetColorKey API not support；
3. Mechanism of iOS platform transfer message to the window is not yet implemented.
   PlayM4_SetEncChangeMsg,PlayM4_SetFileEndMsg,PlayM4_SetEncChangeMsg not support.
4. PlayM4_RefreshPlayEx,PlayM4_SetDeflash,PlayM4_InputVideoData,
   PlayM4_InputAudioData,PlayM4_SetPicQuality,PlayM4_GetPictureQuality,
   PlayM4_SetSourceBufCallBack,PlayM4_ResetSourceBufFlag,
   PlayM4_GetKeyFramePos,PlayM4_GetNextKeyFramePos,
   PlayM4_CheckDiscontinuousFrameNum,PlayM4_SetDisplayType,
   PlayM4_GetDisplayType,PlayM4_SetColor,PlayM4_GetColor,
   PlayM4_AdjustWaveAudio,PlayM4_SetAudioCallBack,
   PlayM4_OpenStreamEx,PlayM4_CloseStreamEx not support yet.
5. PlayM4_GetOriginalFrameCallBack,PlayM4_GetFileSpecialAttr Interface and baseline consistent, no inter support;
6. Some data structures such as PLAYM4_HWND, PLAYM4_HDC etc., there are different definitions in different platform, please refer to the specific header files.

Chapter 5 API Calling Reference

Initialize application

File Mode

PlayM4_GetPort
PlayM4_SetFileRefCallBack
PlayM4_OpenFile
PlayM4_Play
PlayM4_Stop
PlayM4_CloseFile
PlayM4_FreePort

Stream Mode

PlayM4_GetPort
PlayM4_SetStreamOpenMode
PlayM4_OpenStream
PlayM4_SetDisplayBuf
PlayM4_Play
PlayM4_InputData
PlayM4_Stop

End of application

Chapter 6 API Description

System Operation and Error Code Query

1.1 Get SDK Version and Build Number PlayM4_GetSdkVersion

unsigned int 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.

6.1 Get Error Code PlayM4_GetLastError

unsigned int PlayM4_GetLastError (int 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.

6.2 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:

--

Notice:

The interface does not support the iOS platform.

6.3 Initial DirectDraw Surface PlayM4_InitDDraw*

int PlayM4_InitDDraw (PLAYM4_HWND hWnd);

Description:

Initialize DirectDraw surface.

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.4 Release DirectDraw Surface PlayM4_RealeseDDraw*

int PlayM4_ReleaseDDraw ();

Description:

Release DirectDraw surface.

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.5 Set Timer Type PlayM4_SetTimerType

int PlayM4_SetTimerType (int nPort, unsigned int nTimerType, unsigned int 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:

1- API calling succeeds;

0- 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.

6.6 Get Timer Type PlayM4_GetTimerType

int PlayM4_GetTimerType (int nPort, unsigned int*pTimerType, unsigned int*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:

1- API calling succeeds;

0- API calling fails.

6.7 Get Valid Port Number PlayM4_GetPort

int PlayM4_GetPort (int* nPort);

Description:

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

Parameters:

nPort

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

Return:

1- API calling succeeds;

0- API calling fails.

6.8 Release Player Port PlayM4_FreePort

int PlayM4_FreePort (int 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:

1- API calling succeeds;

0- API calling fails.

File Operation

6.9 Open File PlayM4_OpenFile

int PlayM4_OpenFile (int nPort, char*sFileName);

Description:

Open the file for playback.

Parameters:

nPort

[in] Valid port number of the player

sFileName

[in] The file name

Return:

1- API calling succeeds;

0- API calling fails.

Notice:

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

6.10 Close File PlayM4_CloseFile

int PlayM4_CloseFile (int nPort);

Description:

Close the file that has been opened.

Parameters:

nPort

[in] Valid port number of the player

Return:

1- API calling succeeds;

0- API calling fails.

Stream Operation

6.11 Set Stream Input Mode PlayM4_SetStreamOpenMode

int PlayM4_SetStreamOpenMode (int nPort, unsigned int 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:

1- API calling succeeds;

0- API calling fails.

Notice:

This API shall be called before playback starts.

6.12 Get Stream Input Mode PlayM4_GetStreamOpenMode

int PlayM4_GetStreamOpenMode (int nPort);

Description:

Get stream input mode.

Parameters:

nPort

[in] Valid port number of the player

Return:

STREAME_REALTIME or STREAME_FILE

6.13 Open Stream PlayM4_OpenStream

int PlayM4_OpenStream (int nPort, unsigned char* pFileHeadBuf, unsigned int nSize, unsigned int 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:

1- API calling succeeds;

0- API calling fails.

Notice:

```
#define SOURCE_BUF_MAX      1024*100000
#define SOURCE_BUF_MIN      1024*50
```

6.14 Close Stream PlayM4_CloseStream

```
int PlayM4_CloseStream (int nPort);
```

Description:

Close the stream which has been opened.

Parameters:

nPort

[in] Valid port number of the player

Return:

1- API calling succeeds;

0- API calling fails.

6.15 Input Stream Data PlayM4_InputData

```
int PlayM4_InputData (int nPort, unsigned char* pBuf, unsigned int 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 1.

If the data input fails, the return value is 0. If the error code is 11 is due to the internal buffer full.

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.

6.16 Open Video/Audio Stream Separately PlayM4_OpenStreamEx*

int PlayM4_OpenStreamEx (int nPort, unsigned char* pFileHeadBuf, unsigned int nSize, unsigned int nBufPoolSize);

Description:

Open video and audio streams separately.

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.17 Close Video/Audio Stream Separately PlayM4_CloseStreamEx*

int PlayM4_CloseStreamEx (int nPort);

Description:

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

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.18 Input Video Data PlayM4_InputVideoData*

int PlayM4_InputVideoData (int nPort, unsigned char* pBuf, unsigned int nSize);

Description:

Input video stream data got from card

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.19 Input Audio Data PlayM4_InputAudioData*

int PlayM4_InputAudioData (int nPort, unsigned char* pBuf, unsigned int nSize);

Description:

Input audio stream data.

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

Playback Control

6.20 Start Playback PlayM4_Play

int PlayM4_Play (int nPort, PLAYM4_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:

1- API calling succeeds;

0- API calling fails.

6.21 Stop Playback PlayM4_Stop

int PlayM4_Stop (int nPort);

Description:

Stop playback.

Parameters:

nPort

[in] Valid port number of the player

Return:

1- API calling succeeds;

0- API calling fails.

6.22 Pause Playback PlayM4_Pause

int PlayM4_Pause (int nPort, unsigned int nPause);

Description:

Pause or resume playback.

Parameters:

nPort

[in] Valid port number of the player

nPause

[in]

1: pause

0: resume the previous playback process

Return:

1- API calling succeeds;

0- API calling fails.

6.23 Fast Play PlayM4_Fast

int PlayM4_Fast (int nPort);

Description:

Fast play. 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:

1- API calling succeeds;

0- 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.

6.24 Slow Play PlayM4_Slow

int PlayM4_Slow (int nPort);

Description:

Slow play. 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:

1- API calling succeeds;

0- API calling fails.

6.25 Step Forward PlayM4_OneByOne

int PlayM4_OneByOne (int 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:

1- API calling succeeds;

0- API calling fails.

6.26 Step Backward PlayM4_OneByOneBack

int PlayM4_OneByOneBack (int 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:

1- API calling succeeds;

0- API calling fails.

6.27 Play Sound in Exclusive Mode PlayM4_PlaySound

int PlayM4_PlaySound (int 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:

1- API calling succeeds;

0- 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.

6.28 Stop Sound in Exclusive Mode PlayM4_StopSound

int PlayM4_StopSound ();

Description:

Stop the audio playback.

Parameters:

--

Return:

1- API calling succeeds;

0- API calling fails.

6.29 Play Sound in Share Mode PlayM4_PlaySoundShare

int PlayM4_PlaySoundShare (int 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:

1- API calling succeeds;

0- 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.

6.30 Stop Sound in Share Mode PlayM4_StopSoundShare

int PlayM4_StopSoundShare (int nPort);

Description:

Stop audio playback from a specified channel.

Parameters:

nPort

[in] Valid port number of the player

Return:

1- API calling succeeds;

0- 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.

6.31 Set Volume PlayM4_SetVolume

int PlayM4_SetVolume (int nPort, unsigned short 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:

1- API calling succeeds;

0- 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.

6.32 Get Volume PlayM4_GetVolume

unsigned short PlayM4_GetVolume (int 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

6.33 Adjust Audio Wave PlayM4_AdjustWaveAudio*

int PlayM4_AdjustWaveAudio (int nPort, int 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:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.34 Audio synchronization PlayM4_SyncToAudio*

int PlayM4_SyncToAudio(int nPort, int bSyncToAudio);

Description:

Audio synchronization. This API can be called to synchronization the video and the audio.

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.35 Set Picture Quality PlayM4_SetPicQuality*

int PlayM4_SetPicQuality (int nPort, int 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:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.36 Get Picture Quality PlayM4_GetPictureQuality*

int PlayM4_GetPictureQuality (int nPort, int *bHighQuality);

Description:

Get the quality of the image.

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.37 Set Display Video Parameters PlayM4_SetColor*

int PlayM4_SetColor (int nPort, unsigned int nRegionNum, int nBrightness, int nContrast, int nSaturation, int nHue);

Description:

Set video parameters of the pictures.

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.38 Get Display Video Parameters PlayM4_GetColor*

int PlayM4_GetColor (int nPort, unsigned int nRegionNum, int *pBrightness, int *pContrast, int *pSaturation, int *pHue);

Description:

Get the corresponding color value, parameters are as above.

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.39 Set play mode PlayM4_SetPlayMode*

int PlayM4_SetPlayMode(int nPort,int bNormal);

Description:

Get the corresponding color value, parameters are as above.

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.40 Set Playback Position (Percentage) PlayM4_SetPlayPos

int PlayM4_SetPlayPos (int 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:

1- API calling succeeds;

0- API calling fails.

6.41 Get Playback Position (Percentage) PlayM4_GetPlayPos

```
float PlayM4_GetPlayPos (int 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%.

6.42 Set Playback Position (ms) PlayM4_SetPlayedTimeEx

```
int PlayM4_SetPlayedTimeEx (int nPort, unsigned int 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:

1- API calling succeeds;

0- API calling fails.

6.43 Get Playback Position (ms) PlayM4_GetPlayedTimeEx

```
unsigned int PlayM4_GetPlayedTimeEx (int 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)

6.44 Set Playback Position (Frame) PlayM4_SetCurrentFrameNum

int PlayM4_SetCurrentFrameNum (int nPort, unsigned int 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:

1- API calling succeeds;

0- API calling fails.

6.45 Get Playback Position (Frame) PlayM4_GetCurrentFrameNum

unsigned int PlayM4_GetCurrentFrameNum (int 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

6.46 Image De-flashing PlayM4_SetDeflash*

int PlayM4_SetDeflash (int nPort, int bDeflash);

Description:

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

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

Get Playback or Decoding Information

6.47 Get Time Duration of the File PlayM4_GetFileTime

unsigned int PlayM4_GetFileTime (int 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

6.48 Get start and end time of the File PlayM4_GetFileTimeEx

Int PlayM4_GetFileTimeEx(int nPort, unsigned int* pStart, unsigned int* pStop, unsigned int* pRev);

Description:

Get start and end time of the file(unit: second).

Parameters:

nPort

[in] Valid port number of the player

pStart

[out] the start time of the file.

pStop

[out] the stop time of the file.

pRev

[out] the reserved.

Return:

1- API calling succeeds;

0- API calling fails.

6.49 Get Frame Count of the File PlayM4_GetFileTotalFrames

unsigned int PlayM4_GetFileTotalFrames (int 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

6.50 Get Current Frame Rate PlayM4_GetCurrentFrameRate

unsigned int PlayM4_GetCurrentFrameRate (int 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.

6.51 Get Current Frame Rate PlayM4_GetCurrentFrameRateEx

int PlayM4_GetCurrentFrameRateEx(int nPort, float* pfFrameRate);

Description:

Get the current frame rate.

Parameters:

nPort

[in] Valid port number of the player

pfFrameRate

[out] the current frame rate.

Return:

1- API calling succeeds;

0- API calling fails.

6.52 Get Played Time PlayM4_GetPlayedTime

unsigned int PlayM4_GetPlayedTime (int 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.

6.53 Get Decoded Frame Count PlayM4_GetPlayedFrames

unsigned int PlayM4_GetPlayedFrames (int nPort);

Description:

Get the decoded frame number.

Parameters:

nPort

[in] Valid port number of the player

Return:

The decoded frame number of the file

6.54 Get Original Image Size PlayM4_GetPictureSize

int PlayM4_GetPictureSize(int nPort,int *pWidth,int *pHeight);

Description:

Get the original image size of the video.

Parameters:

nPort

[in] Valid port number of the player

int *pWidth

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

int *pHeight

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

Return:

1- API calling succeeds;

0- 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().

6.55 Get File Header Length PlayM4_GetFileHeadLength

unsigned int 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
    unsigned intnLength= PlayM4_GetFileHeadLength ();
    unsigned char* 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);
        }
    }
```

6.56 Get Global Timestamp PlayM4_GetSpecialData

unsigned int PlayM4_GetSpecialData(int 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.

6.57 Set check the water mark PlayM4_SetCheckWatermarkCallBack*

unsigned int PlayM4_SetCheckWatermarkCallBack(int nPort,void(CALLBACK* funCheckWatermark)(int nPort,WATERMARK_INFO* pWatermarkInfo,unsigned int nUser),unsigned int nUser);

Description:

Set the water mark info of current frame.

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.58 Get the Abs frame number PlayM4_GetAbsFrameNum*

unsigned int PlayM4_GetAbsFrameNum(int nPort);

Description:

Get the Abs frame number.

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

Decoding Operation & Control

6.59 Set Callback Stream Type PlayM4_SetDecCBStream

int PlayM4_SetDecCBStream (int nPort, unsigned int 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:

1- API calling succeeds;

0- API calling fails.

6.60 Set Frame Type PlayM4_SetDecodeFrameType

Int PlayM4_SetDecodeFrameType(int nPort,unsigned int 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:

1- API calling succeeds;

0- API calling fails.

6.61 Decoder Callback PlayM4_SetDecCallBack

int PlayM4_SetDecCallBack (int nPort, void (CALLBACK* DecCBFun) (int nPort, char * pBuf, int nSize, FRAME_INFO * pFrameInfo, int nReserved1, int 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{
  int nWidth;          //Image width in pixels or sound track number
  int nHeight;         // Image height in pixels or audio bit rate
  int nStamp;          //Time stamp in milliseconds
  int nType;           // Received data type as the Macro Definition below
  int 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:

1- API calling succeeds;

0- 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 User Data PlayM4_SetDecCallBackMend

int PlayM4_SetDecCallBackMend (int nPort, void (CALLBACK*DecCBFun) (int nPort, char * pBuf, int nSize, FRAME_INFO * pFrameInfo, int nUser, int nReserved2), int 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{
  int nWidth;          //Image width in pixels or sound track number
  int nHeight;         // Image height in pixels or audio bit rate
  int nStamp;          //Time stamp in milliseconds
  int nType;           //Received data type as the Macro Definition below
  int 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:

1- API calling succeeds;

0- 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.

6.63 Callback with Data & Length PlayM4_SetDecCallBackEx

Int PlayM4_SetDecCallBackEx(int nPort,void (CALLBACK* DecCBFun)(int nPort,char * pBuf,int nSize,FRAME_INFO * pFrameInfo, int nReserved1,int nReserved2), char* pDest, int 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:

1 - API calling succeeds;

0 - API calling fails.

6.64 Callback with User Data & Data Length PlayM4_SetDecCallBackExMend

```
Int PlayM4_SetDecCallBackExMend(int nPort, void (CALLBACK* DecCBFun)(int nPort, char* pBuf, int nSize, FRAME_INFO* pFrameInfo, int nUser, int nReserved2), char* pDest, int nDestSize, int 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:

1- API calling succeeds;

0- API calling fails.

6.65 Set Audio Callback PlayM4_SetAudioCallBack*

Notice:

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

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

Description:

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

Parameters:

--

Return:

The interface does not support the iOS platform.

6.66 Set File End Message PlayM4_SetFileEndMsg*

int PlayM4_SetFileEndMsg (int nPort, PLAYM4_HWND hWnd, unsigned int 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:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.67 Set File End Callback PlayM4_SetFileEndCallback

Int PlayM4_SetFileEndCallback (int nPort, void (CALLBACK *FileEndCallback) (int 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) (int nPort, void *pUser)*

Parameters:

nPort:

Port number

pUser:

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

Return:

1- API calling succeeds;

0- 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.

6.68 Notify on Resolution Changing PlayM4_SetEncChangeMsg*

int PlayM4_SetEncChangeMsg (int nPort, PLAYM4_HWND hWnd, unsigned int nMsg);

Description:

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

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.69 Set Resolution Change Callback PlayM4_SetEncTypeChangeCallBack

int PlayM4_SetEncTypeChangeCallBack (int nPort, void (CALLBACK *funEncChange) (int nPort, int nUser), int 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) (int nPort, int nUser)*

nPort

Valid port number of the player

nUser

User data

Return:

1- API calling succeeds;

0- 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.

6.70 Throw B Frame(s) PlayM4_ThrowBFrameNum

int PlayM4_ThrowBFrameNum (int nPort, unsigned int 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:

1- API calling succeeds;

0- API calling fails.

6.71 Throw B Frame(s) CallBack PlayM4_GetThrowBFrameCallBack*

int PlayM4_GetThrowBFrameCallBack(int nPort,void(CALLBACK* funThrowBFrame)(int nPort,unsigned int nBFrame,unsigned int nUser),unsigned int nUser);

Description:

Get the number of B frame(s) to be skipped during decoding.

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.72 Check Frame Continuity PlayM4_CheckDiscontinuousFrameNum*

int PlayM4_CheckDiscontinuousFrameNum (int nPort, int bCheck);

Description:

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

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.73 Set Decryption Key PlayM4_SetSecretKey

```
int PlayM4_SetSecretKey(int nPort, int lKeyType, char *pSecretKey, int 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:

1- API calling succeeds;

0- API calling fails.

Display Operation

6.74 Set Overlay Mode and Color Key PlayM4_SetOverlayMode*

int PlayM4_SetOverlayMode (int nPort, int bOverlay, COLORREF colorKey) ;

Description:

Set display surface and color key for overlay mode.

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.75 Display Mode Query PlayM4_GetOverlayMode*

int PlayM4_GetOverlayMode (int nPort);

Description:

Check if the player is in OVERLAY mode or not.

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.76 Get Overlay Color Key PlayM4_GetColorKey*

COLORREF PlayM4_GetColorKey (int nPort);

Description:

Get the color key of the OVERLAY surface.

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.77 Set Image Sharpness PlayM4_SetImageSharpen*

Int PlayM4_SetImageSharpen(int nPort,unsigned int nLevel);

Description:

Set the sharpness level of the image.

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

1.1 Set Overlay Flip Mode PlayM4_SetOverlayFlipMode*

Int PlayM4_SetOverlayFlipMode(int nPort,int bTrue);

Description:

Set the overlay flip.

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.78 Set/Add Display Region PlayM4_SetDisplayRegion

int PlayM4_SetDisplayRegion (int nPort, unsigned int nRegionNum, RECT *pSrcRect, PLAYM4_HWND hDestWnd, int 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:

1- API calling succeeds;

0- API calling fails.

6.79 Set video window PlayM4_SetVideoWindow

int PlayM4_SetVideoWindow(int nPort, unsigned int nRegionNum, PLAYM4_HWND hWnd);

Description:

Set the video window.

Parameters:

nPort

[in] Valid port number of the player

nRegionNum

[in] Display region number from 0 to (MAX_DISPLAY_WND-1).

hWnd

[in] The handle of the display window.

Return:

--

Notice:

The interface does not support the iOS platform.

6.80 Refresh Display PlayM4_RefreshPlay

int PlayM4_RefreshPlay (int 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:

1- API calling succeeds;

0- API calling fails.

6.81 Refresh Display for Multiple Regions PlayM4_RefreshPlayEx*

int PlayM4_RefreshPlayEx (int nPort, unsigned int nRegionNum);

Description:

Refresh display for multiple display regions.

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.82 Set Normal/Quarter Display Mode PlayM4_SetDisplayType*

int PlayM4_SetDisplayType (int nPort, int 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:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.83 Normal/Quarter Display Mode Query PlayM4_GetDisplayType*

int PlayM4_GetDisplayType (int nPort);

Description:

Check the current display mode.

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

Source Buffer Operation

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

6.84 Free Space Query PlayM4_GetSourceBufferRemain

unsigned int PlayM4_GetSourceBufferRemain (int nPort);

Description:

Get the remain space information of the source buffer.

Parameters:

nPort

[in] Valid port number of the player

Return:

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

6.85 Threshold Setting & Callback PlayM4_SetSourceBufCallBack

int PlayM4_SetSourceBufCallBack (int nPort, unsigned int nThreShold, void (CALLBACK * SourceBufCallBack) (int nPort, unsigned int nBufSize, unsigned int dwUser, void*pResvered), unsigned int 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] The threshold for the free space of source buffer.

SourceBufCallBack

[in] pointer of callback function

dwUser

[in] user data

pReserved

[in] *Reserved;*

Description of the Callback Function:

Parameters:

nPort

Valid port number of the player

nBufSize

Size of the free space of source buffer

dwUser

User data

pResvered

Reserved

Return:

1- API calling succeeds;

0- API calling fails.

Notice:

The interface does not support the iOS platform.

6.86 Reset Threshold Callback PlayM4_ResetSourceBufFlag*

int PlayM4_ResetSourceBufFlag (int 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:

--

Return:

--

Notice:

The interface does not support the iOS platform.

Decode Buffer Operation

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

6.87 Set Buffering Size PlayM4_SetDisplayBuf

int PlayM4_SetDisplayBuf (int nPort, unsigned int 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 inter 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 10 (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:

1- API calling succeeds;

0- API calling fails.

Notice:

This function shall be called between PlayM4_OpenStream and PlayM4_Play.

6.88 Buffering Size Query PlayM4_GetDisplayBuf

```
unsigned int PlayM4_GetDisplayBuf (int 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

6.89 Clear All Buffer PlayM4_ResetSourceBuffer

```
int PlayM4_ResetSourceBuffer (int nPort);
```

Description:

Clear the data in all the buffers.

Parameters:

nPort

[in] Valid port number of the player

Return:

1- API calling succeeds;

0- API calling fails.

6.90 Clear Specified Buffer PlayM4_ResetBuffer

```
int PlayM4_ResetBuffer (int nPort, unsigned int 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:

1- API calling succeeds;

0- API calling fails.

6.91 Buffer Info Query PlayM4_GetBufferValue

unsigned int PlayM4_GetBufferValue (int nPort, unsigned int 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: remained data for video decode buffer (unit: Frame)

BUF_AUDIO_RENDER: remained data for audio decode buffer (unit: Frame, every 40ms audio is divided as a frame)

Return:

Current data length in the specified buffer

6.92 Set get user data Call back PlayM4_SetGetUserDataCallBack*

Int PlayM4_SetGetUserDataCallBack(int nPort,void(CALLBACK* funGetUserData)(int nPort,unsigned char *pUserBuf,unsigned int nBufLen,unsigned int nUser),unsigned int nUser);

Description:

Set get user data call back..

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

File Index

6.93 Set File Index Callback PlayM4_SetFileRefCallBack

int PlayM4_SetFileRefCallBack (int nPort, void (_stdcall *pFileRefDone) (unsigned int nPort, unsigned int nUser), unsigned int 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 (unsigned int nPort, unsigned int nUser)

Parameters:

nPort

Valid port number of the player

nUser

User data

Return:

1- API calling succeeds;

0- API calling fails.

6.94 Locate Previous Key Frame PlayM4_GetKeyFramePos*

int PlayM4_GetKeyFramePos(int nPort, unsigned int nValue, unsigned int 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:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.95 Locate Next Key Frame PlayM4_GetNextKeyFramePos*

int PlayM4_GetNextKeyFramePos (int nPort, unsigned int nValue, unsigned int nType, PFRAME_POS pFramePos);

Description:

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

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.96 Get File Index Info PlayM4_GetRefValue

int PlayM4_GetRefValue (int nPort, BYTE *pBuffer, unsigned int*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:

1- API calling succeeds;

0- API calling fails.

6.97 Set File Index PlayM4_SetRefValue

int PlayM4_SetRefValue (int nPort, BYTE *pBuffer, unsigned int 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:

1- API calling succeeds;

0- 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.

6.98 Enum the Display Devices in the System PlayM4_InitDDrawDevice*

int PlayM4_InitDDrawDevice();

Description:

Enumerate display devices of system

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.99 Release Display Device Resource PlayM4_ReleaseDDrawDevice*

void PlayM4_ReleaseDDrawDevice ();

Description:

Release resource distributed in the course of enumerating display devices

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.100 Get Display Adapter Number PlayM4_GetDDrawDeviceTotalNums*

unsigned int PlayM4_GetDDrawDeviceTotalNums ();

Description:

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

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.101 Assign Display Adapter for the Monitor PlayM4_SetDDrawDevice*

int PlayM4_SetDDrawDevice (int nPort, unsigned int nDeviceNum);

Description:

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

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.102 Assign Display Adapter for Multiple Monitors PlayM4_SetDDrawDeviceEX*

int PlayM4_SetDDrawDeviceEx (int nPort, unsigned int nRegionNum, unsigned int nDeviceNum);

Description:

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

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.103 Get the Display Adapter and Monitor Info PlayM4_GetDDrawDeviceInfo*

int PlayM4_GetDDrawDeviceInfo (unsigned int nDeviceNum, char*lpDriverDescription, unsigned int nDespLen, char*lpDriverName , unsigned int nNameLen, HMONITOR *hhMonitor);

Description:

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

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

6.104 Get System Info of Display Devices PlayM4_GetCapsEx*

int PlayM4_GetCapsEx (unsigned int nDDrawDeviceNum);

Description:

Get some capabilities of your system.

Parameters:

--

Return:

--

Notice:

The interface does not support the iOS platform.

Snapshot

6.105 Snapshot Callback PlayM4_SetDisplayCallBack

int PlayM4_SetDisplayCallBack (int nPort, void (CALLBACK* DisplayCBFun) (int nPort, char * pBuf, int nSize, int nWidth, int nHeight, int nStamp, int nType, int 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:

1- API calling succeeds;

0- 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.

6.106 Snapshot with User DataPlayM4_SetDisplayCallBackEx

Int PlayM4_SetDisplayCallBackEx(int nPort,void (CALLBACK* DisplayCBFun)(DISPLAY_INFO *pstDisplayInfo), int 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:

1- API calling succeeds;

0- 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.

6.107 Convert YV12 Image to BMP File PlayM4_ConvertToBmpFile

```
int PlayM4_ConvertToBmpFile (char * pBuf, int nSize, int nWidth, int nHeight, int 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:

1- API calling succeeds;

0- API calling fails.

Notice:

The YV12-RGB converting process takes CPU resource from PC.Image capture 2CIF resolution after resolution of 4CIF.The original image resolution of 2CIF file snapshot after the resolution of 4CIF

6.108 Convert YV12 Image to JPEG File PlayM4_ConvertToJpegFile

int PlayM4_ConvertToJpegFile (char *pBuf, int nSize, int nWidth, int nHeight, int 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:

1- API calling succeeds;

0- 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. The original image resolution of 2CIF file snapshot after the resolution of 4CIF

6.109 BMP Snapshot PlayM4_GetBMP

Int PlayM4_GetBMP (int nPort, unsigned char* pBitmap, unsigned int nBufSize, unsigned short *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:

1- API calling succeeds;

0- API calling fails.

Notes:

The original image resolution of 2CIF file snapshot after the resolution of 4CIF.

6.110 JPEG Snapshot PlayM4_GetJPEG

int PlayM4_GetJPEG (int nPort, unsigned char* pJpeg, unsigned int nBufSize, unsigned short *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:

1- API calling succeeds;

0- 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. The original image resolution of 2CIF file snapshot after the resolution of 4CIF.

6.111 Set JPEG Snapshot Quality PlayM4_SetJpegQuality

int PlayM4_SetJpegQuality (int 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:

1- API calling succeeds;

0- 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].

Others

6.112 DirectDraw Surface Callback PlayM4_RegisterDrawFun

Int PlayM4_RegisterDrawFun (int nPort, void (CALLBACK* DrawFun) (int nPort, PLAYM4_HDC hDc, int nUser), int 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:

void CALLBACK DrawFun (int nPort, PLAYM4_HDC ***hDc, int nUser);***

Parameters:

nPort

[out] Valid port number of the player

hDc

[out] The off-screen surface DC.

nUser

[out] The user data.

Return:

1- API calling succeeds;

0- 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.

6.113 DirectDraw Surface Callback PlayM4_RigisterDrawFun

Int PlayM4_RigisterDrawFun (int nPort, void (CALLBACK* DrawFun) (int nPort, PLAYM4_HDC hDc, int nUser), int 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:

void CALLBACK DrawFun (int nPort, PLAYM4_HDC ***hDc, int nUser);***

Parameters:

nPort

[out] Valid port number of the player

hDc

[out] The off-screen surface DC.

nUser

[out] The user data.

Return:

1- API calling succeeds;

0- 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.

PlayM4_RegisterDrawFun spelling mistakes

6.114 Set Data Verify Callback PlayM4_SetVerifyCallBack*

int PlayM4_SetVerifyCallBack (int nPort, unsigned int nBeginTime, unsigned int nEndTime, void (_stdcall* funVerify) (int nPort, FRAME_POS * pFilePos, unsigned int bIsVideo, unsigned int nUser), unsigned int 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:

--

Return:

The interface does not support the iOS platform.

6.115 Get Original Frame Callback PlayM4_GetOriginalFrameCallBack*

int PlayM4_GetOriginalFrameCallBack (int nPort, int bIsChange, int bNormalSpeed, int nStartFrameNum, int nStartStamp, int nFileHeader, void (CALLBACK *funGetOrignalFrame) (int nPort, FRAME_TYPE *frameType, int nUser), int 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) (int nPort, FRAME_TYPE *frameType, int nUser)

nPort:

Channel number;

frameType:

data frame information

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

nUser:

user data.

Return:

1- API calling succeeds;

0- API calling fails.

6.116 Get File Attributes PlayM4_GetFileSpecialAttr*

int PlayM4_GetFileSpecialAttr (int nPort, unsigned int*pTimeStamp, unsigned int*pFileNum , unsigned int*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:

1- API calling succeeds;

0- API calling fails.

6.117 Jump to the Next Key Frame on Error PlayM4_SkipErrorData

int PlayM4_PlaySkipErrorData(int nport, int 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]

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

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

Return:

1- API calling succeeds;

0- API calling fails.