Player SDK
Programmer Manual
(For Android)

Version 7.3.1.X

2016-01

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, should be given orally or in writing or should 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 capturing image in BMP or JPG format.

Chapter 2 Version Update

2.1 Version Description

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

***V** 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.

2.2 Version Information

2.2.1 Version 7.3.1.X

- Adds API of getting decoding type: getDecoderType.
- Adds API of setting Max. hardware decoding channel number: setMaxHardDecodePort。
- Adds API of setting synchronous playback group： setSycGroup。
- Edits the default display node as 6.
- Hardware decoding supported stream type adds H265, and MPEG4.

2.2.2 Version 7.3.0.X

Addition:

- Add hard decoding function: setHardDecode.
- Add wide-angle image correction function: setImageCorrection.
- Add fisheye correction functions: setFECDisplayEffect, getFECDisplayParam, setFECDisplayParam.
- Add pre-recording function: setPreRecordCallBack, setPreRecordFlag.

2.2.3 Version 7.1.0.X

Addition:

- This new version, all API related to the operation the file related to the operation do not system testing.
- Change play by the frame rate to the time stamp;
- Add API:
  setFileRefCB: file index callback function;
  setDecodeCB: decode callback function;
- Change:
  setVideoWindow doesn't support to add display region;
  setDisplayRegion support to add display region;
- Support Android 2.3 and above system(the current version test Android system version 2.3~4.2);

2.2.4 Version 7.0.2.0

Addition:

- Support Android 2.2 and above systems.

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_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_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 API Calling Reference

Initialize application

File Mode

getPort
setFileEndCB
setFileRefCB
openFile
setDecodeCB
setDisplayCB
play
stop
closeFile
freePort

Stream Mode

getPort
setStreamOpenMode
openStream
setDisplayBuf
setDecodeCB
setDisplayCB
Play
inputData
stop
closeStream
freePort

Fisheye Correction

getPort
openFile/openStream
play
setFECDisplayEffect
getFECDisplayParam
setFECDisplayParam
stop
closeFile/closeStream
freePort

End of application

Chapter 5 API Description

5.1 System Operation and Error Code Query

5.1.1 Getting SDK Version and Build Number getSdkVersion

int 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.

5.1.2 Getting Error Code getLastError

int 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.

5.1.3 Getting Valid Port Number getPort

int getPort();

Description:

Get a valid port number. Valid range of port number is [0,15].

Parameters:

nPort

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

Return:

true - API calling succeeds;

false - API calling fails.

5.1.4 Releasing Player Port freePort

boolean 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:

true - API calling succeeds;

false - API calling fails.

5.2 File Operation

5.2.5 Opening File openFile

boolean openFile(int nPort, String sFilePath);

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 stored locally.

5.2.6 Closing File closeFile

boolean closeFile(int 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.

5.3 Stream Operation

5.3.7 Setting Stream Input Mode setStreamOpenMode

```
boolean setStreamOpenMode(int nPort, 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 (play by time stamp)

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 should be called before playback starts.

5.3.8 Opening Stream openStream

```
boolean openStream(int nPort, byte[] pFileHeadBuf, int nSize, 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.

#define SOURCE_BUF_MAX 1024*100000

#define SOURCE_BUF_MIN 1024*50

Return:

true - API calling succeeds;

false - API calling fails.

5.3.9 Closing Stream closeStream

```
int PlayM4_CloseStream (int 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.

5.3.10 Inputting Stream Data inputData

```
boolean inputData(int nPort, byte[] pBuf, 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 true.

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

Notice:

Suggestions for data input failure handling:

- 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.
- For the data input failure caused by full buffer under STREAME_FILE mode, users should retry until input succeeds.

5.4 Playback Control

5.4.11 Startting Playback play

```
boolean play(int nPort, SurfaceHolder holder);
```

Description:

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

holder

[in] The handle of the display window.

Return:

true - API calling succeeds;

false - API calling fails.

5.4.12 Ending Playback stop

```
boolean stop(int nPort);
```

Description:

Stop playback.

Parameters:

nPort

[in] Valid port number of the player

Return:

true - API calling succeeds;

false - API calling fails.

5.4.13 Playback Pause pause

boolean pause(int nPort, 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:

true - API calling succeeds;

false - API calling fails.

5.4.14 Fast Forward fast

boolean fast(int 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 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.

5.4.15 Slow Forward slow

boolean slow(int 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 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.

5.4.16 Playing Sound in Exclusive Mode playSound

boolean 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:

true - API calling succeeds;

false - API calling fails.

Notice:

- The audio display is disabled by default.
- playSound() and stopSound() should be used together in the application. If playSound() is called, then stopSound() should be called before application ends.
- When playing low frame rate file, please enable the audio before playing, otherwise it may cause audio playing exception.

5.4.17 Ending Sound in Exclusive Mode stopSound

boolean stopSound();

Description:

Stop the audio playback.

Parameters:

--

Return:

true - API calling succeeds;

false - API calling fails.

5.4.18 Setting Playback Position (Percentage) setPlayPos

boolean 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, ranging from 0% to 100%.

Return:

true - API calling succeeds;

false - API calling fails.

5.4.19 Getting Playback Position (Percentage) getPlayPos

float 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, ranging from 0% to 100%.

5.4.20 Setting Playback Time (ms) setPlayedTimeEx

boolean setPlayedTimeEx(int nPort, 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:

true - API calling succeeds;

false - API calling fails.

5.4.21 Getting Playback Time (ms) getPlayedTimeEx

long getPlayedTimeEx (int nPort);

Description:

Get the current playback position of the file in milliseconds.

Parameters:

nPort

[in] Valid port number of the player

Return:

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

5.4.22 Setting Playback Position (Frame) setCurrentFrameNum

boolean setCurrentFrameNum(int nPort, 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:

true - API calling succeeds;

false - API calling fails.

5.4.23 Getting Playback Position (Frame) getCurrentFrameNum

Int 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

5.5 Getting Playback or Decoding Information

5.5.24 Getting Time Duration of the File getFileTime

long 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

5.5.25 Getting System Time getSystemTime

Boolean getSystemTime(int nPort, MPSystemTime stSystemTime);

Description:

Get total file duration (unit: second).

Parameters:

nPort

[in] Valid port number of the player

MPSystemTime

[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:

The file's total duration in seconds

5.5.26 Get Frame Count of the File getFileTotalFrames

int 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.

5.5.27 Getting Current Frame Rate getCurrentFrameRate

int 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.

5.5.28 Getting Decoded Frame Count getPlayedFrames

int 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

5.5.29 Getting Original Image Size getPictureSize

boolean getPictureSize(int nPort, MPInteger stWidth, MPInteger stHeight);

Description:

Get the original image size of the video.

Parameters:

nPort

[in] Valid port number of the player

pWidth

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

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 should be called after playback starts to get the precise information. If the current resolution is 2CIF, the pic is 4CIF.

5.6 Decoding Operation & Control

5.6.30 Setting Frame Type setDecodeFrameType

boolean setDecodeFrameType(int nPort, 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:

true - API calling succeeds;

false - API calling fails.

5.6.31 Decoding callback setDecodeCB

boolean setDecodeCB(int nPort, PlayerDecodeCB decodeCB);

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:

public void onDecode(int nPort, byte[] data, int nDataLen, int nWidth, int nHeight, int nFrameTime, int nDataType, int Reserved);

Parameters:

nPort

Valid port number of the player

data

Pointer of the decoded video/audio buffer

nDataLen

Size of pBuf

nWidth

Image width in pixels or sound track number

nHeight

Image height in pixels or audio sample rate

nFrameTime

Time stamp in milliseconds

nDataType

Received data type as the Macro Definition below

Reserved

Reserved

Macro Definition:

T_AUDIO16 = 101	PCM Audio, sampling rate: 16 Khz, Mono, 16 bits
T_AUDIO8 = 100	PCM Audio, sampling rate: 8 Khz, Mono, 16 bits
T_RGB32 = 7	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 = 1	uyvy image, arranged in "U0-Y0-V0-Y1-U2-Y2-V2-Y3…." (starts from the bottom-left of the image) format (Reserved)
T_YV12 = 3	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 should be called before **Play** and will be invalid automatically after **Stop.**
- The decoding function provides no speed control and the decoding starts whenever the call back function returns. To use this function, user should understand mechanisms of video & audio display.

5.6.32 Setting File end CallBack setFileEndCB

boolean setFileEndCB(int nPort, PlayerPlayEndCB playEndCB);

Description:

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

Parameters:

nPort

[in] Valid port number of the player

playEndCB

[in] Callback function described below

Description of the Callback Function:

public void onPlayEnd(int nPort);

Parameters:

nPort

Valid port number of the player

Return:

true - API calling succeeds;

false - API calling fails.

Notice:

For the callback functions, it must call the API before the Openfile.

5.6.33 Setting Decoding Secret Key setSecretKey

boolean setSecretKey(int nPort, int nKeyType, byte[] pSecretKey, int nKeyLen);

Description:

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

Parameters:

nPort

[in] Valid port number of the player

nKeyType

[in] Secret key type

pSecretKey

[in] Secret key string

nKeyLen

[in] Key length (unit: bit)

Return:

true - API calling succeeds;

false - API calling fails.

Notice:

The interface does not support the Android platform.

5.7 Display Operation

5.7.34 Setting/Adding Display Region setDisplayRegion

boolean setDisplayRegion(int nPort, int nRegionNum, MPRect pSrcRect, Surface 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.

MPRect pSrcRectc

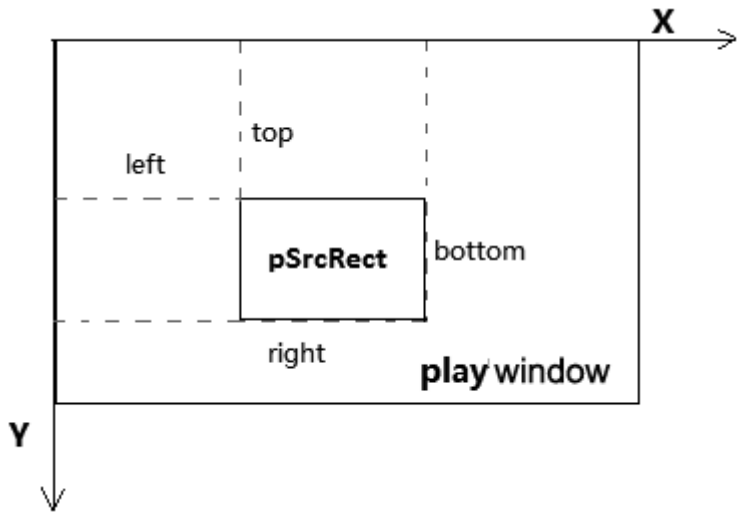
[out] Set the region to be displayed (this region should be inside the original image), i.e. if the resolution of original image is 352*288, the range of pSrcRect should not exceed (0, 0, 352, 288).

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

Note:

The MPRect structure parameters description is shown below.

```
public static class MPRect
{
public int    left;      X axis coordinate on the upper-left corner of the area
public int    right;     X axis coordinate on the lower-right corner of the area
public int    top;       Y axis coordinate on the upper-left corner of the area
public int    bottom;    Y axis coordinate on the lower-right corner of the area
    }
```



hDestWnd

[in] Set the display window. If the window hasn't been set, set hDestWnd as display window; If the window has already been set, and the display window is consistent with hDestWnd, neglect this parameter; If the window has already been set, and the display window is inconsistent with hDestWnd, change to hDestWnd dynamically.

bEnable

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

Return:

true - API calling succeeds;

false - API calling fails.

Remark:

- It can be used as partial zoom in when calling this API under the normal playing status for setting or adding region.
- When nRegionNum is 0, it indicates to work on the main window，the setting of hDestWnd and bEnable will be ignored. The Digital Zoom region range（pSrcRect）value must be more than 0，the value of right and bottom must be more than 16.

5.7.35 Setting video window setVideoWindow

boolean setVideoWindow(int nPort, int nRegionNum, SurfaceHolder holder);

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).

holder

[in] The handle of the display window.

Return:

true - API calling succeeds;

false - API calling fails.

Notice:

- Set the display area.
- The last parameter SurfaceHolder has life periods among which two period callback function:

  SurfaceCreated: call the API setVideoWindow(nPort, nRegionNum, holder) to create SurfaceHolder;

  SurfaceDestroyed: call the API setVideoWindow(nPort, nRegionNum, null) to destroy SurfaceHolder;

5.7.36 Setting Synchronous Playback Group setSycGroup

public boolean setSycGroup(int nPort, int nGroupIndex)

Description:

Set synchronous playback group

Parameters:

nPort

[in] Play library

nGroupIndex

[in] Synchronous playback group No., the value should be 0 to 4, hardware decoding is not supported.

Return:

true - API calling succeeds;

false - API calling fails.

Notice:

- It is valid to call this function repeatedly to add the specified nPort into the synchronous playback group.
- Call setSycGroup before enabling the play function.
- One nGroupIndex(Synchronous playback group No.)indicates one group, Max. 16 playback channel each group. If the channel number is over 16, the returned value will be false. Release the channel No. by calling PlayM4_FreePort to make it out of the group.
- Only the Hik code stream is supported.

5.8 Source Buffer Operation

**Source buffer is the buffer for the data to be decoded. The API of source buffer operation is available only under the openStream mode.*

5.8.37 Free Space Query getSourceBufferRemain

int getSourceBufferRemain(int nPort);

Description:

Get the free space information of the source buffer (the nPoolBufSize of openStream).

Parameters:

nPort

[in] Valid port number of the player

Return:

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

5.8.38 Clearing All Buffers resetSourceBuf

boolean resetSourceBuf(int nPort);

Description:

Clear the data in all the buffers.

Parameters:

nPort

[in] Valid port number of the player

nBufType

[in] Buffer type

Return:

true - API calling succeeds;

false - API calling fails.

5.8.39 Clearing Specified Buffer resetBuffer

boolean resetBuffer(int nPort, int nBufType);

Description:

Clear the data in a specified buffer.

Parameters:

nPort

[in] Valid port number of the player

nBufType

[in] Buffer type

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.

5.9 Decoding Buffer Operation

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

5.9.40 Setting Buffering Size setDisplayBuf

boolean setDisplayBuf(int nPort, 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:

true - API calling succeeds;

false - API calling fails.

Notice:

This function should be called between openStream and play.

5.9.41 Buffering Size Query getDisplayBuf

int 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.

5.10 Source Buffer & Decode Buffer Operation

5.10.42 Buffer Info Query getBufferValue

long getBufferValue (int nPort, int nBufType);

Description:

Get the data size of specified buffer.

Parameters:

nPort

[in] Valid port number of the player

nBufType

[in] Buffer type

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.

Remark:

Get player buffer size (frame or byte). BUF_VIDEO_RENDER can be used for estimating network delay time.

5.11 File Index

5.11.43 Setting File Index Callback setFileRefCB

boolean setFileRefCB(int nPort, PlayerFileRefCB fileRefCB);

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. All the Index-related APIs should be called after indexing, and the other APIs will not be affected.

Parameters:

nPort

[in] Valid port number of the player

fileRefCB

[in] pointer of callback function

Description of the Callback Function:

public void onFileRefDone(int nPort);

Parameters:

nPort

Valid port number of the player

Return:

true - API calling succeeds;

false - API calling fails.

5.12 Capture Image

5.12.44 Capturing Image Callback setDisplayCB

boolean setDisplayCB(int nPort, PlayerDisplayCB displayCB);

Description:

Register a callback function for image capturing.

Parameters:

nPort

[in] Valid port number of the player

DisplayCBFun

[in] Capturing image callback function

Description of the Callback Function:

public void onDisplay(int nPort, byte[] data, int nDataLen, int nWidth, int nHeight, int nFrameTime, int nDataType, int Reserved);

Parameters:

nPort

Valid port number of the player

data

Pointer of the image capturing buffer

nDataLen

Size of the image capturing buffer

nWidth

Width of the image (unit: pixels)

nHeight

Height of the image (unit: pixels)

nFrameTime

Time stamp (unit: ms)

nDataType

Received data type: T_YV12, T_RGB32, T_UYVY

nReserved

Reserved

Return:

true - API calling succeeds;

false - API calling fails.

Notice:

- The callback should 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.

5.12.45 Capturing BMP Image getBMP

boolean getBMP(int nPort, byte[] pBmp, int nBufSize, MPInteger pBmpSize);

Description:

Capture image in BMP format

Parameters:

nPort

[in] Valid port number of the player

pBitmap

[in] Address assigned by users for storing BMP image, the file size should be no less than 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.

5.12.46 Capturing JPEG Image getJPEG

boolean getJPEG(int nPort, byte[] pJpeg, int nBufSize, MPInteger pJpegSize);

Description:

Capture image in JPEG format

Parameters:

nPort

[in] Valid port number of the player

pJpeg

[in] Address assigned by users for storing JPEG image, the file size should be no less than 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 captured image resolution will be automatically cropped to multiple of 16. E.g. original image of 176*120 will be cropped to 176*112.

5.13 Hard Decoding

5.13.47 Setting Hard Decoding Priority setHardDecode

boolean setHardDecode (int nPort, int bEnable);

Description:

Set to prioritize hard decoding or not.

Parameters:

nPort

[in] Valid port number of the player

bEnable

[in] Open/set or close the hard decoding priority.

Return:

true - API calling succeeds;

false - API calling fails.

Notice:

- System Version: Android 4.1 (API16) or above.
- This hard decoding API should be set before Play and after Open.
- It supports files or real-time stream of standard H.264, H265, and MPEG4 encoding.
- It supports resolutions of CIF, 4CIF, 720P, VGA, 1080P, UXGA, etc. At present, the maximum resolution supported is UXGA.
- There is little difference between these resolutions in terms of performance. 1080P resolution of 30FPS can be played in double-speed.
- When enabling hard decoding, the following APIs are not supported: getJPG, getBMP, setImageCorrection, getPictureSize, setDisplayRegion, setVideoWindow, getDisplayBuf, setDisplayBuf, setDisplayCB, setFECDisplayEffect, setFECDisplayParam, resetBuffer(RENDER_BUFFER).
- The last frame will remain after the hardware decoding playback being stopped.

5.13.48 Setting Max. Hardware Decoding Channel Number setMaxHardDecodePort

public boolean setMaxHardDecodePort(int nCount);

Description:

Set the max. hardware decoding channel number

Parameters:

nCount

[in] Max. hardware decoding channel number

Return:

true - API calling succeeds;

false - API calling fails.

Notice:

The Max. hardware decoding channel number should be in the range of 0 to 16. Call this API before enabling hardware decoding setHardDecode.

5.13.49 Getting Current Decoding Type getDecoderType

public int getDecoderType(int nPort);

Description:

Get current decoding type

Parameters:

nPort

[in] Display channel No.

Return:

true - API calling succeeds;

false - API calling fails.

Notice:

Call this API when the video is playback normally after enabling the play function (play).

5.14 Pre-recording

5.14.50 Setting Pre-recording Data Callback setPreRecordCallBack

boolean setPreRecordCallBack(int nPort, PlayerPreRecordCB PreRecordCB);

Description:

Set pre-recording data callback.

Parameters:

nPort

[in] Valid port number of the player

PreRecordCB

[in] Pre-recording callback function.

pUser

[in] User pointer.

Return:

true - API calling succeeds;

false - API calling fails.

5.14.51 Setting Pre-recording Switch setPreRecordFlag

boolean setPreRecordFlag(int nPort, int bFlag);

Description:

Set pre-recording switch.

Parameters:

nPort

[in] Valid port number of the player

bFlag

[in] Enable or disable.

Return:

true - API calling succeeds;

false - API calling fails.

5.15 Fisheye

5.15.52 Setting Image Correction setImageCorrection

boolean setImageCorrection(int nPort, int bEnable);

Description:

Set to enable the wide-angle image correction or not.

Parameters:

nPort

[in] Valid port number of the player

bEnable

[in] Open/set or close the image correction.

Return:

true - API calling succeeds;

false - API calling fails.

5.15.53 Setting Fisheye Correction Type setFECDisplayEffect

boolean setFECDisplayEffect(int nPort, int nRegionNum, MPVR_DISPLAY_EFFECT enDisplayEffect);

Description:

Set the fisheye correction type.

Parameters:

nPort

[in] Valid port number of the player

nRegionNum

[in] Display region number.

enDisplayEffect

[in] Fisheye correction type.

VR_ET_NULL = 0x100, no correction;

VR_ET_FISH_PTZ_CEILING = 0x101, correction for ceiling mounted fisheye;

VR_ET_FISH_PTZ_FLOOR = 0x102, correction for floor mounted fisheye;

VR_ET_FISH_PTZ_WALL = 0x103, correction for wall mounted fisheye;

VR_ET_FISH_PANORAMA_CEILING360 = 0x104, correction for ceiling mounted fisheye 1P;

VR_ET_FISH_PANORAMA_CEILING180 = 0x105, correction for ceiling mounted fisheye 2P;

VR_ET_FISH_PANORAMA_FLOOR360 = 0x106, correction for floor mounted fisheye 1P;

VR_ET_FISH_PANORAMA_FLOOR180 = 0x107, correction for floor mounted fisheye 2P;

VR_ET_FISH_LATITUDE_WALL = 0x108, correction for wide angle wall mounted

fisheye;

VR_ET_REDBLUE_3D = 0x109, correction for red and blue 3D fisheye.

Return:

true - API calling succeeds;

false - API calling fails.

Notice:

- The fisheye correction must be operated in the display region which enables playing.
- The fisheye correction can be operated after APIs of setDisplayRegion and setVideoWindow set the new region.
- The fisheye correction is operated on the original image.

5.15.54 Getting Fisheye Correction Parameter

getFECDisplayParam

boolean getFECDisplayParam(int nPort, int nRegionNum, MPVR_FISH_PARAM stFishParam);

Description:

Get the fisheye correction parameter.

Parameters:

nPort

[in] Valid port number of the player

nRegionNum

[in] Display region number.

stFishParam

[in] Fisheye correction parameter.

```
public static class MPVR_FISH_PARAM{
        public float    xLeft;      //The X coordinate on the far left (min).
        public float    xRight;     //The X coordinate on the far right (max).
        public float    yTop;       //The Y coordinate on the top (min).
        public float    yBottom;    //The Y coordinate on the top (min).
        public float    angle;      //180°correction central angle.
        public float    zoom;       //PTZ correction: zoom
        public float    PTZX;       //X coordinate of PTZ correction center.
        public float    PTZY;       //Y coordinate of PTZ correction center.
}
```

Return:

true - API calling succeeds;

false - API calling fails.

5.15.55 Setting Fisheye Correction Parameter

setFECDisplayParam

boolean setFECDisplayParam(int nPort, int nRegionNum, MPVR_FISH_PARAM stFishParam);

Description:

Set the fisheye correction parameter.

Parameters:

nPort

[in] Valid port number of the player

nRegionNum

[in] Display region number.

stFishParam

[in] Fisheye correction parameter.

```
public static class MPVR_FISH_PARAM{
    public float xLeft;      //The X coordinate on the far left (min). Range: [0.0,1.0].
    public float xRight; //The X coordinate on the far right (max). Range: [0.0,1.0].
    public float yTop;       //The Y coordinate on the top (min). Range: [0.0,1.0].
    public float yBottom; //The Y coordinate on the top (min). Range: [0.0,1.0].
    public float angle;     //180°correction central angle. Range: [0.0,360.0].
    public float zoom;     //PTZ correction: zoom. Range: (0.0,1.0).
    public float PTZX;     //X coordinate of PTZ correction center. Range: [0.0,1.0]
    public float PTZY;     //Y coordinate of PTZ correction center. Range: [0.0,1.0]
}
```

Return:

true - API calling succeeds;

false - API calling fails.

Notice:

- The parameters should meet the following conditions:

  xLeft < **xRight**;

  yTop < **yBottom**;

  The PTZ correction center should be in the circle of radius 0.4 and centered in (0.5,0.5).

- The excessive parameter setting may cause image over correction and incorrect display.