VXG Player SDK for iOS
Programmer's Guide

VXG Inc.,

May 17, 2017

Legal Disclaimer

Information in this document is provided in connection with VXG products. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document. Except as provided in VXG's license agreement for such products, VXG assumes no liability whatsoever and VXG disclaims any express or implied warranty, relating to sale and/or use of VXG products including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright or other intellectual property right.

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

Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.

Content

1. Overview

VXG Player SDK consists of a set of resources for fast and convenient development of mobile applications for viewing various media streams like RTMP, HLS, RTSP, RTP, MMS, WebM, FLV, MP4, TS, and other network video formats and playback files with following formats: AVI, MOV, MKV, FLV, AVI, 3GP, 3G2, ASF, WMV, MP4, M4V, TP, TS, MTP, M2T, etc. The core of the SDK is a library for application development.

Key Features:

Hardware acceleration – a new hardware accelerated decoder for HD video.

Multi-core decoding – support of the multiple processor cores for decoding.

Multi-channel support – simultaneous connection to multiple resources or multiple video channels and simultaneous video decoding.

Video integration with any Activity – is based on SurfaceView and can be integrated with any Activity.

Hardware pre and post video processing – hardware de-interlacing and various pre and post video processing using OpenGL shaders.

Custom and standard notifications – notifies application about connection, disconnection and other events. It is possible to add custom events.

Smart and online thumbnails – quick and simple API to get thumbnails for local files and network streams.

Low latency for network stream – special API to control playback latency.

Record streams – special API to record streams into mp4 file.

Audio and Subtitle control – special API to control audio and subtitle tracks during playback.

Audio pitch correction on changed rate – the filter added for correcting the intonation of an audio signal without affecting other aspects of its sound when playback rate has been applied.

2. How to Use

2.1 iOS version

The SDK works with iOS version 6 or newer. (Lower version 5.0 can be customized and provided by request as well).

2.2 Folders and files

The SDK package consists of following files and folders:

bin (Sample application package)

libs (Library files to be linked to the application and headers files)
libMediaPlayerSDK.a
include/M3U8.h
include/MediaPlayer.h
include/MediaPlayerConfig.h
include/Thumbnailer.h
include/ThumbnailerConfig.h

src (Sample to test VXG Player SDK)

doc (Documentation including this document)

2.3 Development tools

Our build environment is XCode. Please import the project to XCode for building the sample application.

2.4 Integration with an application

2.4.1 Integration dynamically (without modifying resources)

Step 1: Create new player
Step 2: Call player contentView to get UIView
Step 3: Insert UIView to parent view

Step 4: Configuration
Step 5: Open player

Here is an example:

```
player = [[MediaPlayer alloc] init:self.view.bounds];
UIView *frameView = [player contentView];
[self.view insertSubview:frameView atIndex:0];
MediaPlayerConfig* config = [[MediaPlayerConfig alloc] init];
config.connectionUrl = URL;
config.decodingType =1; // HW – 1 //SW – 0
config.numberOfCPUCores = 2;
[player Open:config callback:self];
```

3. Media Player

3.1 API Reference

There are following API providers in SDK: content provider, decoder provider and render provider:

Provider name	Provider acronym	Description
Pipeline Provider	PLP_	Controls pipeline and all components
Content Provider	CP_	Connects to server, downloads data and controls connection
Video Decoder Provider	VDP_	s/w or h/w video decoding
Audio Decoder Provider	ADP_	s/w or h/w video decoding
Video renderer Provider	VRP_	Video renderer
Audio renderer Provider	ARP_	Audio renderer

3.2 Notifications

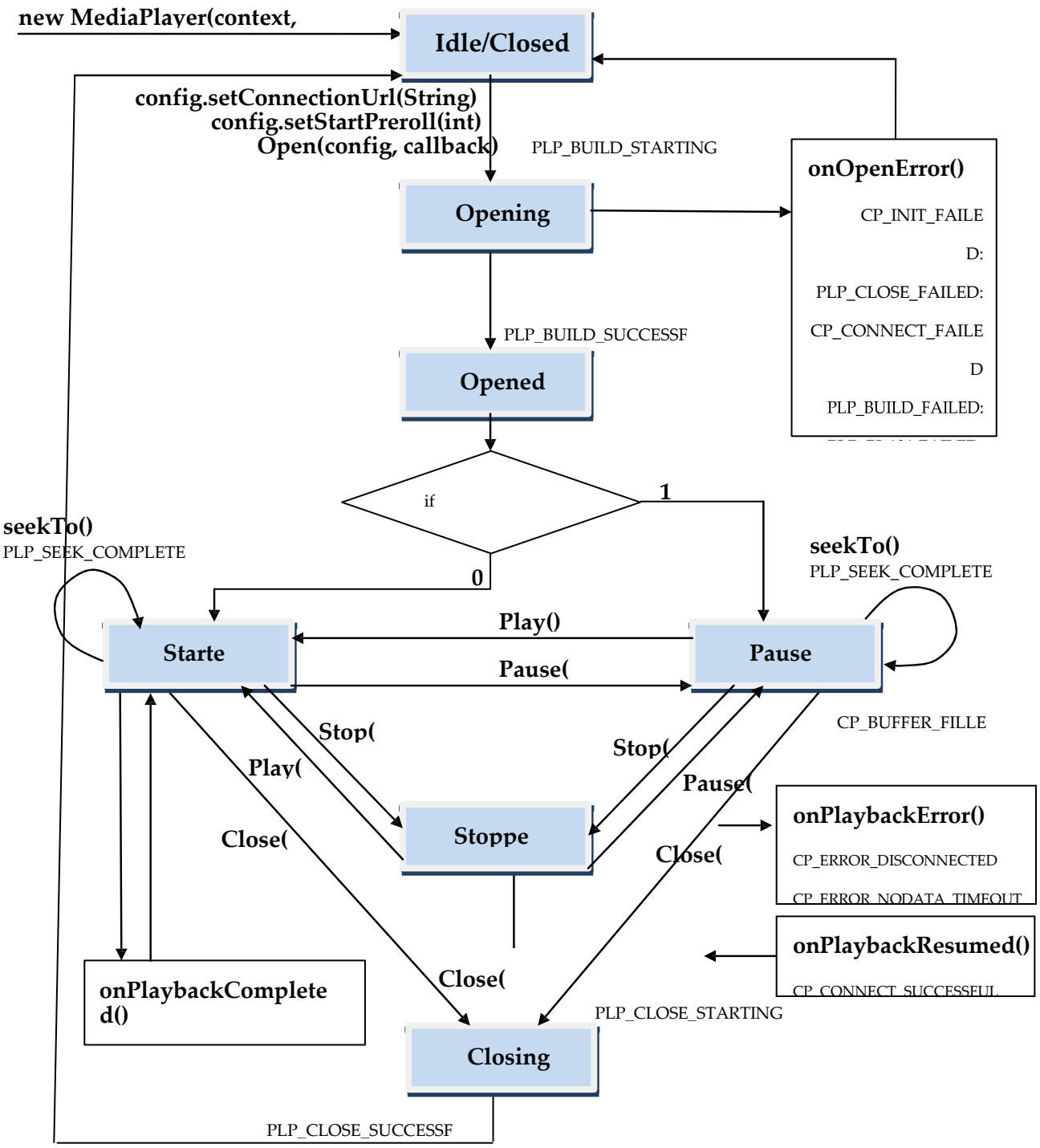
Providers notifies about results, errors and notifications using “MediaPlayerCallback” callback. All messages are synchronous and provider waits until the application handles a message.

Value	Name	Type	Description
1	PLP_BUILD_STARTING	NOTIFICATION	PLP notifies that pipeline is started to build
2	PLP_BUILD_SUCCESSFUL	RESULT	Pipeline has been built successfully
3	PLP_BUILD_FAILED	RESULT	Pipeline cannot be built
4	PLP_PLAY_STARTING	NOTIFICATION	Pipeline is going to start
5	PLP_PLAY_SUCCESSFUL	RESULT	Pipeline has been ran successfully after Open (autostart)
6	PLP_PLAY_FAILED	RESULT	Error on pipeline starting
7	PLP_CLOSE_STARTING	NOTIFICATION	Pipeline is going to stop
8	PLP_CLOSE_SUCCESSFUL	RESULT	Pipeline has been closed successfully
9	PLP_CLOSE_FAILED	RESULT	Error on pipeline closing
10	PLP_ERROR	ERROR	Pipeline is disconnected due inner error
12	PLP_EOS	NOTIFICATION	End-of-stream notification
14	PLP_PLAY_PLAY	NOTIFICATION	Pipeline has been run successfully
15	PLP_PLAY_PAUSE	NOTIFICATION	Pipeline has been paused successfully
16	PLP_PLAY_STOP	NOTIFICATION	Pipeline has been stopped successfully
17	PLP_SEEK_COMPLETED	NOTIFICATION	Seek operation has been completed

101	CP_CONNECT_STARTING	NOTIFICATION	CP is initialized and is going to start connection
102	CP_CONNECT_SUCCESSFUL	RESULT	CP has been connected successfully
103	CP_CONNECT_FAILED	RESULT	CP notifies that connection is failed
104	CP_ INTERRUPTED	RESULT	CP notifies that connection with server is interrupted by close function
105	CP_ERROR_DISCONNECTED	NOTIFICATION	CP notifies that connection with server is lost
106	CP_STOPPED	NOTIFICATION	CP has been stopped
107	CP _ INIT_FAILED	RESULT	CP notifies that there is an error on initialization
108	CP_RECORD_STARTED	NOTIFICATION	CP notifies that recording started and new file has been created. Call *player*.RecordGetFileName(1) to get name of file.
109	CP_RECORD_STOPPED	NOTIFICATION	CP notifies that recording has stopped and the file has been finished. Call *player*.RecordGetFileName(0) to get name of file.
110	CP_RECORD_CLOSED	NOTIFICATION	CP notifies that recording is closed.
111	CP_BUFFER_FILLED	NOTIFICATION	CP notifies about pre-buffering process is completed.
112	CP_ERROR_NODATA_TIMEOUT	NOTIFICATION	CP notifies that no data had came for DataReceiveTimeout period.
113	CP_SOURCE_AUDIO_DISCONTINUITY	NOTIFICATION	CP notifies that there is audio discontinue (difference in PTS between contiguous samples is more than 100ms
114	CP_SOURCE_VIDEO_DISCONTINUITY	NOTIFICATION	CP notifies that there is video discontinue (difference in PTS between contiguous video frames is more than 100 ms)
115	CP_START_BUFFERING	NOTIFICATION	Buffering is started if data in buffer reach the defined threshold
116	CP_STOP_BUFFERING	NOTIFICATION	Buffering is stopped and playback continues
201	VDP_ STOPPED	NOTIFICATION	VDP has been stopped
202	VDP_ INIT_FAILED	RESULT	VDP notifies that there is an error on initialization
300	VRP_STOPPED	NOTIFICATION	VRP has been stopped
301	VRP_INIT_FAILED	RESULT	VRP notifies that there is an error on initialization

302	VRP_NEED_SURFACE	NOTIFICATION	VRP notifies that it is going to allocate surface
305	VRP_FIRSTFRAME	NOTIFICATION	VRP notifies that first frame is rendered
306	VRP_LASTFRAME	NOTIFICATION	VRP notifies that last video frame rendered before end of file
308	VRP_FIRSTFRAME_AFTER_PAUSE	NOTIFICATION	VRP notifies that first frame is rendered after change state from PAUSE to PLAY
400	ADP_STOPPED	RESULT	ADP has been stopped
401	ADP_ INIT_FAILED	RESULT	ADP notifies that there is an error on initialization
500	ARP_STOPPED	NOTIFICATION	ARP has been stopped
501	ARP _INIT_FAILED	NOTIFICATION	ARP notifies that there is an error on initialization
502	ARP_LASTFRAME	NOTIFICATION	ARP notifies that last audio sample is rendered
600	CRP_STOPPED	NOTIFICATION	CRP has been stopped
701	SDP_STOPPED_THREAD	NOTIFICATION	SDP has been stopped
702	SDP_FAILED_INIT	NOTIFICATION	SDP initialization failed

3.3 State diagram



Application registers single **callback** function via **Open (config, callback)** call. State diagram separates notifications into 3 groups:

- **onOpenError()**. Occurs when error has happened in **Open()** function.
- **onPlaybackError()**. Occurs when error has happened in one of Paused/Started/Stopped states.
- **onPlaybackCompleted()**. Occurs in Started state only when end-of-stream has reached.

In case **onOpenError()** the closing procedure is processed automatically, i.e. MediaPlayer goes to **Closed** state.

In case **onPlaybackError() / CP_ERROR_DISCONNECTED** the closing procedure is not processed automatically, pipeline state is not changed, but in order to do further playback the application must close pipeline Close() before Open() pipeline again.

seekTo() is processed by either **setStreamPosition()** or **setLivePosition()** in Started or Paused states. In case if result of setStreamPosition() or setLivePosition() is equal to 0, the notification **PLP_SEEK_COMPLETED** will be.

In case **onPlaybackCompleted()/PLP_EOS** happened, the state of pipeline is not changed.

3.4 Functions description

Following functions are members of MediaPlayer class. These functions should be used to playback network content and media files.

Open

Connect to network server or open media file, create pipeline and playback media data.

Definition

```
(void) Open: (MediaPlayerConfig*)config
        callback: (id<MediaPlayerCallback>)callback;
```

All configuration parameters are described in the table below:

Name	Description	Values	Default	Type
setColorBackground	Set/Get Background color		Color.BLACK	Int
setEnableAspectRatio setAspectRatioMode	Set/Get Video output modes	0 - stretch, 1 - fit to screen with aspect ratio 2 - crop by height 21 - crop by width, 3 - 100% size 4 - zoom mode 5 - move mode	1	Int
setAspectRatioZoomModePercent	Zoom value if video output mode is “Zoom mode”	25-300	100	Int
setAspectRatioMoveModeX setAspectRatioMoveModeY	Set position to top and left for video output if video output mode is “Move mode”	-500 - 500	-1,-1 – Center of the screen	Int
setStartOffest	Set start offset for HLS stream, real position is last segment – offset	Depends on the stream, in milliseconds	0x8000000000000000L	long
setEnableAudio	Mute, unmute audio speaker	0-1	1	Int
setSslKey	Set rtmp_token for RTMP TL			String
setExtStream	Set stream number for HLS stream with various channels	Depends on the stream	0	Int
setSelectedAudio	Select audio track on start if there is more than one track in file or stream	Depends on the stream or file	0 – first track	Int
setSelectedSubtitle	Set subtitle track on stream opening	Depends on the stream or file, -1 - disabled	0 – first track	Int

setConnectionUrl	Set stream URL or file path			String
setDecodingType	Set video decoder type	0 - software, 1 - hardware	1	Int
setDecoderLatency	Control minimal latency on s/w decoder, This setting is for s/w decoder	1 - Low latency, frames are not buffered in decoder, 0 - frames are buffered in video decoder	0	Int
setRendererType	Obsolete parameter			
setSynchroEnable	Enable audio & video synchronization	0 - Disable synchronization 1 - Enable synchronization	1	Int
setSynchroNeedDropVideoFrames	Drop video frames if they are late	0 - drop frames 1 - render frames	0	Int
setDropOnFastPlayback	Obsolete parameter			
setEnableColorVideo	Obsolete parameter			
setNumberOfCPUCores	Set number of CPU for video decoding	Depends on the system ≤ 0 – autodetection	1	Int
setConnectionNetworkProtocol	Select preferred transport protocol for RTSP	0 - udp, 1 - tcp, 2 - http, 3 - https, -1 - AUTO	-1	Int
setConnectionDetectionTime	Time on start for detection video and audio formats (in milliseconds)	100-10000	5000	Int
setConnectionBufferingType	Set buffering type 0 – by time 1 – by size	0 or 1	0	Int
setConnectionBufferingTime	Buffer size on start in milliseconds	0-25000	1000	Int
setConnectionBufferingSize	Buffer size on start in bytes	0 - Max buffer size	0	Int

setDataReceiveTimeout	Set max timeout Interrupt source if data is not received in defined time		60000	Int
setConnectionTimeout	Interrupt source if connection is not passed in defined timeout		60000	Int
setStartPreroll	Enable Pause on start	0 - start immediately 1 - start - play 1 frame - pause	0	Int
setStartCookies	Set cookie in HTTP request			String
setFadeOnStart	Fade audio on stream start	0 - audio comes straight off 1 - audio is faded ~200ms	1	Int
setFadeOnSeek	Fade audio on change position	0 - audio comes straight off 1 - audio is faded ~200ms	1	Int
setFFRate	Fade audio on change rate	0 - audio comes straight off 1 - audio is faded ~200ms	1	Int
setVolumeDetectMaxSamples	Number of samples to detect middle volume		0	Int
setVolumeBoost	Set volume boost	0 - off min:-30dB, max:+30dB	0	Int
setFadeOnChangeFFSpeed	Fade audio on change speed	0 - audio comes straight off 1 - audio is faded ~200ms	1	Int
setRecordPath	Set path for recorded files			String
setRecordFlags	Set flag setting for recording	PP_RECORD_NO_START(0x00000000) PP_RECORD_AUTO_	0	Int

		START(0x00000001) PP_RECORD_SPLIT_BY_TIME(0x00000002) PP_RECORD_SPLIT_BY_SIZE(0x00000004), PP_RECORD_DISABLE_VIDEO(0x00000008) PP_RECORD_DISABLE_AUDIO(0x00000010) PP_RECORD_PTS_CORRECTION(0x00000020); PP_RECORD_PTS_CORRECTION(0x00000020)		
setRecordSplitTime	Split stream on chunks by time if flags are PP_RECORD_SPLIT_BY_TIME, in seconds	0-100	0	Int
setRecordSplitSize	Split stream on chunks by size if flags are PP_RECORD_SPLIT_BY_ SIZE, in seconds		0	Int
setRecordPrefix	Prefix is added to name of recorded files			String
setRecordTrimPosStart	Start position for trim from file, in milliseconds			Int
setRecordTrimPosEnd	Stop position for trim file, in milliseconds			Int
setEnableABR	Set adaptive bitrate control, experimental version			Int

VideoRotate	Set video rotation	90,180,270	0	Int

Return Value

There is no return value.

Remarks

Connect to network resource or open local media file, create pipeline, allocate resource and start video playback.

Examples

```
MediaPlayerConfig* config = [[MediaPlayerConfig alloc] init];
config.connectionUrl = URL;
config.decodingType = hardware_decoder; // HW
config.numberOfCPUCores = 2;
[player Open:config callback:self];
```

Play

Resume play if player is in Pause state.

Definition

- (void) Play: (int)drawNumberOfFramesAndPause;

Parameters

drawNumberOfFramesAndPause – how many frames will be played before pause. Now supported only drawNumberOfFramesAndPause:1 - draw one frame and pause.

Return Value

There is no return value.

Remarks

Resume play if player is in Pause state. This function can be used with playback from files only.

Examples

[Player Play:0];

Pause

Change playback state from Play to Pause.

Definition

(void) Pause

Parameters

There are no parameters for this call

Return Value

There is no return value.

Remarks

Pause playback if player is in Play state. This function can be used with playback from file only.

Examples

[Player Pause];

Stop

Change playback state from Play to Stop.

Definition

(void) Stop

Parameters

There are no parameters for this call

Return Value

There is no return value

Remarks

Stop playback if player is in Play state.

Examples

[Player Stop];

getState

Return player state.

Definition

- (MediaPlayerState) getState;

Parameters

There are no parameters for this call

Return Value

Following states are provided:
0 – Opening

1 – Opened
2 – Started
3 – Paused
4 – Stopped
5 – Closing
6 – Closed

Remarks

Provide the current state of player.

Examples

```
MediaPlayerState state = [player getState];
if (state == MediaPlayerStarted)
```

getStreamDuration

Return duration of media file in milliseconds. This function works only in case of file playback.

Definition

```
- (int64_t) getStreamDuration;
```

Parameters

There are no parameters for this call.

Return Value

Upon successful completion, getStreamDuration() returns file duration in milliseconds. Otherwise -1 is returned. All errors are provided in callback status.

Remarks

Provides duration of the file played.

Examples

[player getStreamDuration];

getStreamPosition

Get position in played media file. This function works only in case of file playback.

Definition

(int64_t) getStreamPosition;

Parameters

There are no parameters for this call.

Return Value

Upon successful completion, getStreamPosition() returns current position of played file (in milliseconds).

Remarks

Provides the played file position.

Examples

[player getStreamPosition];

setStreamPosition

Set position of played media file. This function works only in case of file playback.

Definition

- (void) setStreamPosition: (int64_t)lTime;

Parameters

lTime – new position in file (in milliseconds)

Return Value

There is no return value

Remarks

Provides the position of played file.

Examples

[player setLiveStreamPosition:slider.value * 1000];

getLiveStreamPosition

Function provides current, first, and last positions for live stream. This function works only in case of live stream playback (HLS).

Definition

```
- (BOOL) getLiveStreamPosition: (int64_t*)first
        current: (int64_t*)current
        last: (int64_t*)last
        duration: (int64_t*)duration
        stream_type: (int*)stream_type;
```

Parameters

first	- dts of first segment in m3u8 list
last	- dts of last segment in m3u8 list
current	- dts of last downloaded packet in HLS stream
duration	- stream duration
stream_type	- stream type finished file or live stream Time base is milliseconds.

Return Value

True - on successful completion, Otherwise – error result.

Remarks

Provide the current, first, last positions in played stream.

Examples

```
int64_t _first = 0;
int64_t _current = 0;
int64_t _last = 0;
int64_t _duration = 0;
int     _stream_type = 0;

[player getLiveStreamPosition:&_first
        current:&_current
        last:&_last
        duration:&_duration
        stream_type:&_stream_type];
```

setLiveStreamPosition

Change position of played live stream. This function works only in case of live stream.

Definition

(void) setLiveStreamPosition: (int64_t)lTime;

Parameters

lTime - new position in live stream (milliseconds)

Return Value

There is no return value.

Remarks

Change the position of life stream played.

Examples

[player setStreamPosition:1000000];

setFFRate

Change speed of playback for local file and network stream.

Definition

- (void) setFFRate: (int)rate;

Parameters

rate - rate value.

Correct values:

Rate	Value
x0.1	100 – Min Value
x0.2	200
x0.5	500
x0.9	900
x1	1000
x2	2000
x3	3000
x4	4000
…	…
x16	16000 – Max Value

Return Value

No value is returned by function setFFRate.

Remarks

Change speed of playback for local file and network stream.

Important note: Some data is skipped if rate is less or more than normal playback rate.

Examples

[player setFFRate:2000]; // Set playback rate to x2

getVideoShot

Capture video picture from video stream. Video format is RGBA32.

Definition

```
- (int) getVideoShot: (void*)buffer
        buffer_size: (int32_t*)buffer_size
        width: (int32_t*)width
        height: (int32_t*)height
        bytes_per_row: (int32_t*)bytes_per_row;
```

Parameters

buffer - allocated buffer for shot
buffer_size - in: allocated before buffer size,
out: real image size
width, height - in: desired scale size. -1 for original.
out: real image sizes
note: work only for software decoding
bytes_per_row - image bytes per row

Return Value

0 - ok, (-1) - error, (-2) - need more buffer space for image.

Remarks

Provide the video shot of last render frame in format ARGB_8888.

Example

int rc = [players[0] getVideoShot:shot_buffer buffer_size:&shot_buffer_size width:&desired_width height:&desired_height bytes_per_row:&bytes_per_row];

getRenderPosition

Function provides last position in played media file or stream.

Definition

- (int64_t) getRenderPosition;

Parameters

There are no parameters for this call.

Return Value

Upon successful completion, getStreamPosition() returns PTS of last video frame or audio sample (milliseconds).

Remarks

Provide the PTS of last played video frame or audio sample.

Examples

long position = [player getRenderPosition];

Close

Disconnect from server and destroy pipeline.

Definition

- (void) Close

Parameters

There are no parameters for this call

Return Value

There is no return value

Remarks

Disconnect from network server, destroy pipeline, free all resources that were allocated on Open() call.

Examples

[Player Close];

toggleMute

Set mute and unmute on audio render.

Definition

- (void) toggleMute: (BOOL)mute;

Parameters

mute - true – audio is off , false – audio is on.

Return Value

There is no return value

Remarks

Mute and unmute audio render.

Examples

```
for (int i = 0; i < actualPlayerCount; i++)
{
        [players[i] toggleMute:true];
};
```

GetDataDelayOnSource

Return delay in milliseconds if input stream comes with some delay in case network bottleneck.

Definition

```
- (int) getDataDelayOnSource;
```

Parameters

There are no parameters for this call

Return Value

Upon successful completion, it returns the difference between when package is expected and when package comes

Example

int position = [player getDataDelayOnSource];

getDataBitrateOnSource

Return bitrate of network input stream comes on device.

Definition

- (int) getDataBitrateOnSource;

Parameters

There are no parameters for this call

Return Value

Return stream bitrate in kbps.

Example

int position = [player getDataBitrateOnSource];

getStatFPS

Return frame rate of downloaded stream so application can estimate if network bandwidth is enough for defined stream.

Definition

- (int) getStatFPS;

Parameters

There are no parameters for this call

Return Value

Upon successful completion, GetStatFPS returns fps of network stream. It is frame rate of stream that is downloaded from network, otherwise -1 is returned. All errors are provided in callback status.

Remarks

Provide the frame rate of captured stream (download speed) to estimate if network speed is enough to playback stream in real time.

Example

int fps = [player getStatFPS];

recordSetup

Set the record setting.

Definition

- (void) recordSetup: (NSString*)path
    flags: (MediaPlayerRecordFlags)flags
    splitTime: (int32_t)splitTime
    splitSize: (int32_t)splitSize
    prefix: (NSString*)prefix;

Parameters

(NSString*)path - path for recorded files
flags: (MediaPlayerRecordFlags)flags – recorded flags

PP_RECORD_NO_START(0x00000000) - Record is off
PP_RECORD_AUTO_START(0x00000001) - Launch record on start streaming
PP_RECORD_SPLIT_BY_TIME(0x00000002) - Split stream on chunks by time
PP_RECORD_SPLIT_BY_SIZE(0x00000004) - Split stream on chunks by size
PP_RECORD_DISABLE_VIDEO(0x00000008) – Video is not recorded
PP_RECORD_DISABLE_AUDIO(0x00000010) – Audio is not recorded

PP_RECORD_PTS_CORRECTION(0x00000020) – Correct PTS before recording if there is discontinue

splitTime: (int32_t)splitTime - Size of chunks in milliseconds if flags are PP_RECORD_SPLIT_BY_TIME (in seconds)

splitSize: (int32_t)splitSize - Split stream on chunks by size if flags are PP_RECORD_SPLIT_BY_SIZE (in megabytes)

prefix: (NSString*)prefix – Prefix is added to name of recorded files

Return Value

There is no return value.

Example

```
[players recordSetup: tmpfile flags: RECORD_AUTO_START splitTime:0 splitSize:0 prefix:@"TestRecord"];
```

recordStart

Start recording.

Definition

```
- (void) recordStart;
```

Parameters

There are no parameters for this call

Return Value

none.

recordStop

Stop recording.

Definition

- (void) recordStop;

Parameters

There are no parameters for this call

Return Value

none.

recordGetFileName

Retrieve the name of file has been recording.

Definition

- (NSString*) recordGetFileName: (int32_t)param;

Parameters

0 – get last stopped file; 1 – get last started file

Return Value

Name and full path of last recorded file.

Remarks

Notifications CP_RECORD_STARTED and CP_RECORD_STOPPED are received when recording activities took place. In order to ensure what is latest file name has been recorded we'd better

call RecordGetFileName(0) at CP_RECORD_STOPPED event, and RecordGetFileName(1) at CP_RECORD_STARTED event happen.

recordGetStat

Return statistics about recorded chunks.

Definition

(int64_t) recordGetStat: (MediaPlayerRecordStat)param;

Parameters

Param can be one of following value:

Flag name	Flag value	Description of returned value
PP_RECORD_STAT_LASTERROR	0	last error
PP_RECORD_STAT_DURATION	1	duration of last chunk in milliseconds
PP_RECORD_STAT_SIZE	2	size of last recorded chunk in bytes
PP_RECORD_STAT_DURATION_TOTAL	3	Total duration in milliseconds
PP_RECORD_STAT_SIZE_TOTAL	4	Total size of recorded data (in bytes)
PP_RECORD_STAT_PREBUFFER	5	Number of packets in pre recorded buffer (in milliseconds)
PP_RECORD_STAT_PKT_COUNT	6	Count of recorded packets
PP_RECORD_STAT_TRIM_POS_START	7	First PTS in trimmed period (milliseconds)
PP_RECORD_STAT_TRIM_POS_END	8	Last PTS in trimmed period (milliseconds)
PP_RECORD_STAT_STATE	9	State of recording: 0: stopped 1: paused 2: run

Example

int64_t total_duration = [player recordGetStat: 3];

UpdateView

Set video output mode for current player instance.

Definition

- (int) updateView;

Parameters

There are no parameters for this call

Return Value

There is no return value

Remarks

UpdateView() function uses settings that are set in player config structure.

Video output mode of output picture
player.getConfig().setAspectRatioMode(VideoOutputMode);

Where VideoOutputMode is :

0 – stretch
1 – fit to screen with aspect ratio
2 – crop video
3 - 100% size of picture
4 - zoom mode
Zoom multiplier of output picture (in percent, 25-400%) is set in player config :
player.getConfig().setAspectRatioZoomModePercent(ZoomMultiplier);
5 - move mode

X and Y position is set in player config:
X position of output picture (in percent, 0-100%)
player.getConfig().setAspectRatioMoveModeX(X);
Y position of output picture (in percent, 0-100%)
player.getConfig().setAspectRatioMoveModeY(Y);

Example

```
// Present video: picture size is 100% in center of screen
MediaPlayerConfig* config = [player getConfig];
config.aspectRatioMode = 0;
config.aspectRatioMoveModeX = 50;
config.aspectRatioMoveModeY = 50;
config.aspectRatioZoomModePercent = 100;
config.aspectRatioMode = 5;
[player updateView];
```

SubtitleGetCount

Retrieves a count of subtitle tracks.

Definition

```
- (int) subtitleGetCount;
```

Parameters

There are no parameters for this call

Return Value

Returns a count of available subtitle tracks.

Remarks

SubtitleGetCount() retrieves a number of subtitle tracks. It can be used when player is in opened state only (PlayerState.Opened) or after notification PlayerNotifyCodes. PLP_BUILD_SUCCESSFUL.

SubtitleSelect

Select a subtitle track to play.

Definition

- (int) subtitleSelect: (int)stream_i;

Parameters

track_i – the number of selected track, the value must be in the range of value has been returned by SubtitleGetCount().

Remarks

SubtitleSelect(track_i) can be used also before opening of the player. track_i value is saved automatically into MediaPlayerConfig.setSelectedSubtitle(). This track will actually used after calling Open.

SubtitleGetSelected

Retrieves a selected subtitle track.

Definition

- (int) subtitleGetSelected;

Parameters

There are no parameters for this call

Return Value

Number of selected subtitle track.

Remarks

SubtitleGetSelected() returns actually selected subtitle tack. In case player is not Opened state, SubtitleGetSelected() returns number that was set in config.

SubtitleSourceAdd

Add an external subtitle source

Definition

- (int) subtitleSourceAdd: (NSString*)path;

Parameters

String path2 – path to subtitle source

Return Value

Upon successful completion, returns 0. Otherwise –ERROR is returned.

Remarks

Function SubtitleSourceAdd adds a path to subtitle source. Application can set up multiple external subtitle sources. After adding subtitle source, the player will increase the count of subtitle tracks (SubtitleGetSelected) and select required track by SubtitleSelect call. SubtitleSourceAdd can be called in both before and after Open() call. Also these paths to external sources can be added through MediaPlayerConfig. subtitlePaths string list.

SubtitleSourceRemove

Removes an external subtitle source has been added by SubtitleSourceAdd function.

Definition

- (int) subtitleSourceRemove: (NSString*)path;

Parameters

String path2 – path to subtitle source

Return Value

Upon successful completion, returns 0. Otherwise –ERROR is returned.

getAvailableDirectionsForAspectRatioMoveMode

Get available directions for move mode.

Definition

- (int) getAvailableDirectionsForAspectRatioMoveMode;

Parameters

none.

Return Value

possible move 0 - nothing, 1 - to left, 2 - to right, 4 - to top, 8 - to bottom.

getViewSizesAndVideoAspects

Get view sizes and video aspects like video width/height, aspect calculations etc.

Definition

- (void)getViewSizesAndVideoAspects:(int*)view_orientation
view_width:(int*)view_width

view_height:(int*)view_height
video_width:(int*)video_width
video_height:(int*)video_height
aspect_left:(int*)aspect_left
aspect_top:(int*)aspect_top
aspect_width:(int*)aspect_width
aspect_height:(int*)aspect_height
aspect_zoom:(int*)aspect_zoom;

Parameters

view_orientation - current view orientation
view_width - current view width
view_height - current view height
video_width - video width
video_height - video height
aspect_left - left video position after current aspect ratio mode calculations
aspect_top - top video position after current aspect ratio mode calculations
aspect_width - video width after current aspect ratio mode calculations
aspect_height - video height after current aspect ratio mode calculations
aspect_zoom - current zoom coefficient

Return Value

None.

getInternalBuffersState

Get internal buffers states.

Definition

- (void)getInternalBuffersState:(int*)source_videodecoder_filled
source_videodecoder_size:(int*)source_videodecoder_size
videodecoder_videorenderer_filled:(int*)videodecoder_videorenderer_filled
videodecoder_videorenderer_size:(int*)videodecoder_videorenderer_size
source_audiodecoder_filled:(int*)source_audiodecoder_filled

source_audiodecoder_size:(int*)source_audiodecoder_size
audiodecoder_audiorenderer_filled:(int*)audiodecoder_audiorenderer_filled
audiodecoder_audiorenderer_size:(int*)audiodecoder_audiorenderer_size;

Parameters

source_videodecoder_filled - bytes filled in buffer between source and video decoder
source_videodecoder_size - buffer size between source and video decoder
videodecoder_videorenderer_filled - buffer size between video decoder and video renderer
videodecoder_videorenderer_size - buffer size between video decoder and video renderer
source_audiodecoder_filled - buffer size between source and audio decoder
source_audiodecoder_size - buffer size between source and audio decoder
audiodecoder_audiorenderer_filled - buffer size between audio decoder and audio renderer
audiodecoder_audiorenderer_size - buffer size between audio decoder and audio renderer.

Return Value

none.