VXG Mobile Video Player SDK for Android Programmer's Guide

VXG Inc.,

Jul 21, 2017

Content

1. Overview ..... - 2 -
2. How to Use ..... - 3 -
2.1 Android version ..... - 3 -
2.2 Folders and files ..... - 3 -
2.3 Development tools ..... - 3 -
2.4 Integration with an application ..... - 3 -
2.4.1 Integration using a resource file in 2 steps: ..... - 3 -
2.4.2 Integration dynamically (without modifying resources) ..... - 5 -
2.4.3 Integration with Activity ..... - 7 -
3. Media Player ..... - 8 -
3.1 API Reference ..... - 8 -
3.2 Notifications ..... - 8 -
3.3 State diagram ..... - 10 -
3.4 Functions description ..... - 12 -
4. Thumbnailer ..... - 45 -
4.1 Functions description ..... - 45 -

1. Overview

Network Media 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.
Audio volume boost and volume detector – special API to increase audio volume above system ability and to detect max/min volume to avoid any audio clipping on raising volume.
Pre-buffering data in paused mode – accumulation of media data in Paused mode to avoid clipping on further playback (audio mode only).

2. How to Use

2.1 Android version

The SDK works with Android version 4.0 or newer. (Lower versions 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)*

- MediaPlayerSDKTest.apk
- MediaPlayerSDKTest.LowLatency.apk
- MediaPlayerSDKTest_view2x2.apk

libs *(Library files to be linked to the application)*

- mediaplayersdk.jar
- librtspplr-xx.so
- librtstm-xx.so
- libSDL2-xx.so
- libyuv_shared-xx.so

where xx is one of supported platforms: ARM general, x86, ARV V7, ARM V7a.

src *(Sample project to test Media Player SDK)*

doc *(Documentation including this document)*

2.3 Development tools

Build environment is Eclipse and Android Studio. Please import the project to Eclipse or Android Studio for building the sample application.

2.4 Integration with an application

2.4.1 Integration using a resource file in 2 steps:

Step1: Add to layout xml for your activity as below:

```
<FrameLayout
    android:id="@+id/playerViewLayout"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" >
```

```
        <veg.mediaplayer.sdk.MediaPlayer
            android:id="@+id/playerView"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_gravity="center" />

    </FrameLayout>
```

Step 2: Change main activity
(MainActivity.java)

```
public class MainActivity extends Activity implements
MediaPlayer.MediaPlayerCallback
{
...
    // callback handler
    #override
    public int Status(int arg) {return 0;}

   @Override
   public void onCreate(Bundle savedInstanceState)
   {
    ...
        // Create Player instance
        player = (MediaPlayer)findViewById(R.id.playerView);
// Get player instance
    ...
        // Connect or start playback
        player.Open(ConnectionUrl or File name,
      decoderType,
      rendererType,
      synchroEnable,
      synchroNeedDropVideoFrames,
                  rendererEnableColorVideo,
                  rendererEnableAspectRatio,
     DataReceiveTimeout,
```

```
     decoderNumberOfCpuCores,
      this);
     ...
    }

   @Override
   protected void onDestroy()
   {
          // Destroy and close player
          if (player != null)
          {
                  // Close connection to server
              player.close ();
                  // Desroy player
             player.onDestroy();
}
         super.onDestroy();
    }
   ...
}
```

2.4.2 Integration dynamically (without modifying resources)

Step 1: Change main activity

```
public class MainActivity extends Activity implements
MediaPlayer.MediaPlayerCallback
{
...
     // callback handler
     #override
     public int Status(int arg) {return 0;}

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
```

```
    ...
        // Create instance of Player
        boolean is_use_window = true;
        // boolean is_use_window = false; if audio only is to play in service
        player = new MediaPlayer(this, is_use_window);
//  Set size and position for layout
        FrameLayout.LayoutParams params = new FrameLayout.LayoutParams(250,250,
Gravity.CENTER);
        player.setLayoutParams(params);

//
        // Add Player Instance to layout
        FrameLayout lp = (FrameLayout)findViewById(R.id.playerView);
        lp.addView(player);
    ...
// connect and start playback
        player.Open(   ConnectionUrl,
                       decoderType,
rendererType,
synchroEnable,
synchroNeedDropVideoFrames,
                       rendererEnableColorVideo,
rendererEnableAspectRatio,
DataReceiveTimeout,
decoderNumberOfCpuCores,
this);
    ...
   }

  @Override
  protected void onDestroy()
  {
          // Close  network connection to server
             player.close ();
           // Desroy player
           player.onDestroy();
```

```
        super.onDestroy();
    }
```

2.4.3 Integration with Activity

The SDK is based on SurfaceView and can be integrated with any Activity using the code below:

```
<FrameLayout

        android:id="@+id/playerViewLayout"

        android:layout_width="fill_parent"

        android:layout_height="wrap_content" >

        <veg.mediaplayer.sdk.MediaPlayer

            android:id="@+id/playerView"

            android:layout_width="fill_parent"

            android:layout_height="fill_parent"

            android:layout_gravity="center" />

    </FrameLayout>
```

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

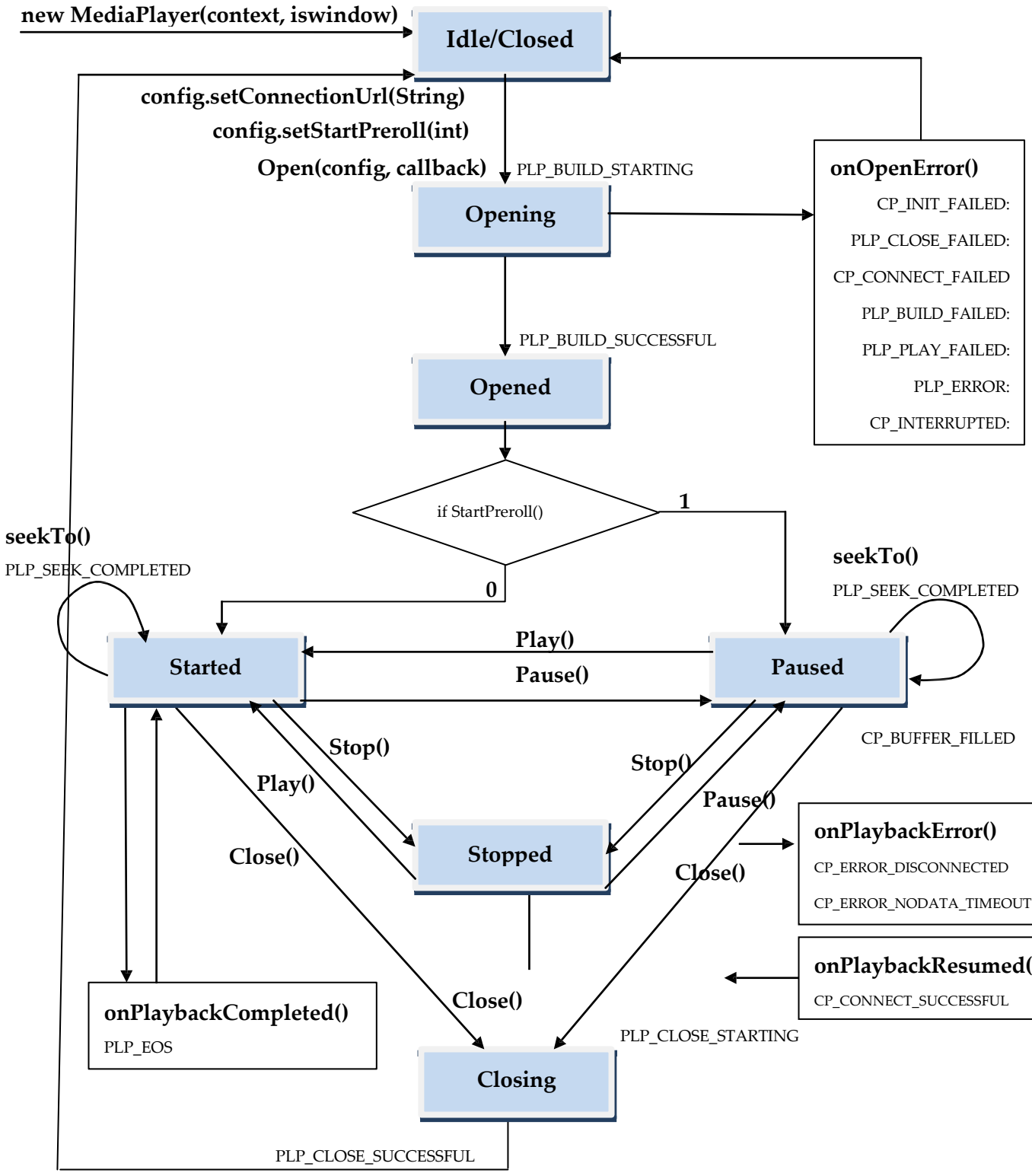
Providers notifies about results, errors and notifications using "MediaPlayerCallback" callback. All messages are synchronous and provider waits until the application handles a message.

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 can not 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
117	CP_DISCONNECT_SUCCESSFUL	NOTIFICATION	CP notifies that network source is disconnected successfully.
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
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
503	ARP_VOLUME_DETECTED	NOTIFICATION	ARP notifies that volume detector has finished and app can get min and max estimated audio volume values.

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 / CP_ERROR_NODATA_TIMEOUT** notification has received the closing procedure is not processed automatically, pipeline state is not changed, further playback goes on **automatically** when network connection has been restored (**CP_CONNECT_SUCCESSFUL** received).

onPlaybackResumed() / CP_CONNECT_SUCCESSFUL occurs after successful restoring of network connection, playback continues **automatically** pipeline state is not changed.

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.

CP_BUFFER_FILLED notification received in Paused state indication that pre-buffering has finished. getStreamPrebufferTime() function returns the time position of pre-buffered data. **seekTo()** up to the pre-buffered position doesn't require network connection.

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
public void Open(
final String ConnectionUrl,
final int DataReceiveTimeout,
final MediaPlayerCallback callback)

public void Open(final String ConnectionUrl,
final int ConnectionNetworkProtocol,
final int ConnectionDetectionTime,
final int ConnectionBufferingTime,
final int DecoderType,
final int RendererType,
final int SynchroEnable,
final int SynchroNeedDropVideoFrames,
final int EnableColorVideo,
final int EnableAspectRatio,
final int DataReceiveTimeout,
final int NumberOfCPUCores,
final MediaPlayerCallback callback)

public void Open(final MediaPlayerConfig config, final MediaPlayerCallback callback)

Parameters:

ConnectionUrl	URL to network resource (RTSP, HTTP, RTMP, HLS, UDP and so on) or full path for media file
ConnectionNetworkProtocol	Network protocol or RTP/RTSP tunneling (0 – RTP by UDP, 1 – RTP by TCP, 2 – RTSP over http, 3 – RTSP over https, -1 - AUTO)

ConnectionDetectionTime	Probing time to detect video and audio formats of network stream (in milliseconds)
ConnectionBufferingTime	Buffering on playback start to avoid network jitter (in milliseconds)
DecoderType	Select s/w or h/w video decoder
RendererType	Select SDL or openGL render
SynchroEnable	Enable A/V synchronization, 1 - synchronization is on, 0 - is off
SynchroNeedDropVideoFrames	Drop video framer if frame is late, 1 - is on, 0 - is off
EnableColorVideo	Enable grayscaled video
EnableAspectRatio	Set video output mode (0 - stretch, 1 - fit to screen with aspect ratio, 2 - crop, 3 - 100% size, 4 - zoom mode, 5 - move mode)
DataReceiveTimeout	Reconnect timeout, SDK does reconnect if there is no data received during some time (milliseconds).
MediaPlayerCallback	Notification callback, event is provided over this callback
NumberOfCPUCores	Number of CPU cores to decode video, ≤ 0 – autodetect and set the number according device capability, positive number sets number according application needs

Return Value

Upon successful completion **Open**() returns 0. Otherwise -1 is returned. All errors are provided in callback status.

Remarks

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

Examples

Example #1

```
player.Open(
 "http://example", // correct URL or full path for media file
        2,      // RTSP over http tunneling
        500,    // 500 ms on probing
        500,    // 500 ms buffer on start
        0,      //  Decoder type :  0- S/W 1, - H/W
        1,      // Renderer Type : 0 - SDL, 1 - pure OpenGL
        1,      // A/V synchronization:  1- Enable , 0 - Disable
```

```
        0,      // Drop Video frame if it is late : 1- Enable , 0 - Disable
        1,      // Color / Grayscale video output  : 0 - grayscale, 1 - color
        1,      // Aspect ratio / Full size : 1 – aspect rate
        30000, // Reconnection  timeout  (milliseconds),
        0,      // Number Of Cpu Cores for decoding (1-6), 0 - autodetect
        This);
```

Example #2

```
// Create config
 MediaPlayerConfig conf = new MediaPlayerConfig();
conf.setConnectionUrl(http://example);  // correct URL or full path to media file
conf.setConnectionNetworkProtocol(2);  // RTSP over http tunneling
conf.setConnectionDetectionTime(500);  // Probing time – 500 ms
conf.setConnectionBufferingTime(500);  //  Buffering on start – 500 ms
conf.setDecodingType(1);  // H/W decoder
 conf.setRendererType(1);  // pure OpenGL
 conf.setSynchroEnable(1);  // Audio and Video synchronization is ON

conf.setSynchroNeedDropVideoFrames(0); // Do not drop video if pts is later
 conf.setEnableColorVideo(1); // Set color video
 conf.setEnableAspectRatio(1); // Set  aspect ratio
 conf.setDataReceiveTimeout(30000); // Set timeout of connection , Disconnect event is
sent after(in milliseconds)
 conf.setNumberOfCPUCores(0); // Number Of Cpu Cores for decoding (1-6), 0-
autodetect
conf.setStartPreroll(1); //Start player in Paused mode

//Recording options
Int record_flags =
        PlayerRecordFlags.PP_RECORD_AUTO_START | //auto start open
        PlayerRecordFlags.PP_RECORD_DISABLE_AUDIO | //video only
        PlayerRecordFlags.PP_RECORD_SPLIT_BY_TIME | //split by time
        PlayerRecordFlags.PP_RECORD_SPLIT_BY_SIZE; //split by size
conf.setRecordFlags(record_flags);
conf.setRecordSplitTime(30); //split by 30 sec
conf.setRecordSplitSize(20); //split by 20 megabytes
```

```
conf.setRecordPath("/sdcard/DCIM");
conf.setRecordPrefix("my_rec");
```

player.Open(conf, This);

All configuration parameters are described in the table below:

Name	Description	Values	Default value	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	Depends on the stream or file	0 – first track	Int

	track in file or stream			
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 2 – hardware decoder and openGL render	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 syncronization	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		60000	Int

	data is not received in defined time			
setConnectionTimeout	Interrupt source if connection is not passed in defined timeout		60000	Int
setStartPreroll	Enable Pause on start	0 - start immediately 1 - start->present frame frame->pause 2 - start->do not present first frame - > pause 3 – connect -> 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_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_FAST_START(0x00000040), PP_RECORD_FRAG_KEYFRAME(0x00000080);	0	Int

setRecordSplitTime	Split stream on chunks by time if flags are PP_RECORD_SP LIT_BY_TIME, in seconds	0-100	0	Int
setRecordSplitSize	Split stream on chunks by size if flags are PP_RECORD_SP LIT_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
setExtraDataOnStart	Add extra data in begin of stream		0	Int

OpenAsPreview

Connect to network server or open media file, create pipeline and playback media data in Preview mode. Preview mode differs from normal: s/w decoding only key frames, real time render, no audio stream (only video).

Definition

public void OpenAsPreview(
final String ConnectionUrl,
final int DataReceiveTimeout,
final MediaPlayerCallback callback)

Parameters

ConnectionUrl	URL to network resource (RTSP, HTTP, RTMP, UPD) or full path for media file
DataReceiveTimeout	Reconnect timeout, SDK does reconnect if there is no received data during some time (milliseconds)
MediaPlayerCallback	Notification callback, event is provided over this callback

Return Value

Upon successful completion **OpenAsPreview**() returns 0. Otherwise -1 is

returned. All errors are provided in callback status.

Remarks

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

Examples

```
player.OpenAsPreview(
        "http://example", // correct URL or full path for media file
        30000, // Connection timeout (milliseconds),
        This);
```

Play

Resume play if player is in Pause state.

Definition

public void Play()

Parameters

There are no parameters for this call

Return Value

Upon successful completion, **Play()** returns 0. Otherwise -1 is returned. All errors are provided in callback status.

Remarks

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

Examples

```
player.Play ();
```

Pause

Change playback state from Play to Pause.

Definition

public void Pause()

Parameters

There are no parameters for this call

Return Value

Upon successful completion, **Pause()** returns 0. Otherwise -1 is returned. All errors

are provided in callback status.

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

Examples

```
player.Pause ();
```

PauseFlush

Change playback state from Play to Pause on all modules expect for network source module.

Definition
public void PauseFlush()

Parameters
There are no parameters for this call

Return Value
Upon successful completion, **PauseFlush()** returns 0. Otherwise -1 is returned. All errors

are provided in callback status.

Remarks
Pause playback and all modules expect for network module if player was in Play state. Nertowork module works in this mode and puts data in video and audio buffers. Video frames and audio samples are skipped if buffers are full.

Examples

```
player.PauseFlush ();
```

getState

Return player state.

Definition
public PlayerState 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

```
if (player.getState() == PlayerState.Closing) ;
```

getStreamDuration

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

Definition
public long getStreamDuration()

Parameters
There are no parameters for this call.

Return Value
Upon successful completion, getStreamDurarion**()** returns file duration in seconds. Otherwise -1 is returned. All errors are provided in callback status.

Remarks
Provides duration of the file played.

Examples
int duration = getStreamDuration() ;

getStreamPosition

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

Definition
public long 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
int position = getStreamPosition() ;

setStreamPosition

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

Definition
public void setStreamPosition (final long lTime)

Parameters
lTime - new position in file (in milliseconds)

Return Value
Integer value is returned. 0 - on successful completion, PLP_SEEK_COMPLETED notification will send on success. Otherwise – error result.

Remarks
Provides the position of played file.

Examples
long position;
setStreamPosition(position) ;

getLiveStreamPosition

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

Definition
Position getLiveStreamPosition()

Parameters
There are no parameters for this call.

Return Value

Upon successful completion, getLiveStreamPosition returns Position object.

```
public class Position
{
        private long first = 0;
        private long current = 0;
        private long last = 0;
        private long duration = 0;
}
```

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.
Time base is milliseconds.

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

Examples

```
Position pos    =       player.getLiveStreamPosition();
if (pos == null)
{
        long duration = pos.getDuration();
        long first    = pos.getFirst();
        long current  = pos.getCurrent();
        long last     = pos.getLast();
 }
```

getStreamInfo

Function returns the Meta data of media file or network stream.

Definition

public String getStreamInfo()

Parameters

There are no parameters for this call.

Return Value

Type – String, Format - xml.

Remarks

Provide the information about media file or network stream.

```
<?xml version="1.0"?>
<StreamInfo name="MediaPlayerSDK" version="1.0">
        <Metadata>
                <SHOUTCastMetadata>
                string1
                string2
                </SHOUTCastMetadata>
                <ContextMetadata>
                string
                </ContextMetadata>

                <ContextChapterMetadata id="0">
                string
                </ContextChapterMetadata>
```

```
        ...
        <ContextChapterMetadata id="N">
        string
        </ContextChapterMetadata>

        <ContextProgramMetadata id="0">
        string
        </ContextProgramMetadata>
        ...
        <ContextProgramMetadata id="N">
        string
        </ContextProgramMetadata>

        <StreamMetadata id="0" type="AV_MEDIA_TYPE">
        string
        </StreamMetadata>
        ...
        <StreamMetadata id="N" type="AV_MEDIA_TYPE">
        string
        </StreamMetadata>
</Metadata>

<Stream name="stream name" duration="value in ms" >
        <VideoStreams>
                <VideoStream id="0" >
                        <title value="name" />
                        <format         value="AV_VIDEO_FORMAT_TYPE"    />
                        <duration value="in ms" />
                        <width value="" />
                        <height         value="" />
                        <fps value="" />
                </VideoStream>
                ...
                <VideoStream id="N" >
                        <title value="name" />
                        <format         value="AV_VIDEO_FORMAT_TYPE"    />
```

```
                <duration value="in ms" />
                <width value="" />
                <height value="" />
                <fps value="" />
        </VideoStream>
</VideoStreams>

<AudioStreams>
        <AudioStream id="0" >
                <title value="name" />
                <format value="AV_AUDIO_FORMAT_TYPE" />
                <duration value="in ms" />
                <samplerate value="in Hz" />
                <channels value="number of channels" />
        </AudioStream>
        ...
        <AudioStream id="N" >
                <title value="name" />
                <format value="AV_AUDIO_FORMAT_TYPE" />
                <duration value="in ms" />
                <samplerate value="in Hz" />
                <channels value="number of channels" />
        </AudioStream>
</AudioStreams>

<SubtitleStreams>
        <SubtitleStream id="0" >
                <title value="name" />
                <format value="AV_SUBTITLE_FORMAT_TYPE" />
        </SubtitleStream>
        ...
        <SubtitleStream id="N" >
                <title value="name" />
                <format value="AV_SUBTITLE_FORMAT_TYPE" />
        </SubtitleStream>
```

```
                </SubtitleStreams>
        </Stream>
</StreamInfo>
```

All possible string values for tag: SHOUTCastMetadata
More info:http://www.indexcom.com/streaming/player/SHOUTcast.html

All possible string values for tags:
ContextMetadata, ContextChapterMetadata, ContextProgramMetadata,
StreamMetadata:

album -- name of the set this work belongs to
album_artist -- main creator of the set/album, if different from artist.
e.g. "Various Artists" for compilation albums.
artist -- main creator of the work
comment -- any additional description of the file.
composer -- who composed the work, if different from artist.
copyright -- name of copyright holder.
creation_time-- date when the file was created, preferably in ISO 8601.
date -- date when the work was created, preferably in ISO 8601.
disc -- number of a subset, e.g. disc in a multi-disc collection.
encoder -- name/settings of the software/hardware that produced the file.
encoded_by -- person/group who created the file.
filename -- original name of the file.
genre -- <self-evident>.
language -- main language in which the work is performed, preferably in
ISO 639-2 format. Multiple languages can be specified by
separating them with commas.
performer -- artist who performed the work, if different from artist.
E.g for "Also sprach Zarathustra", artist would be "Richard
Strauss" and performer "London Philharmonic Orchestra".
publisher -- name of the label/publisher.
service_name -- name of the service in broadcasting (channel name).
service_provider -- name of the service provider in broadcasting.
title -- name of the work.
track -- number of this work in the set, can be in form current/total.

variant_bitrate -- the total bitrate of the bitrate variant that the current stream is part of

Examples
String info = player.getStreamInfo();

setLiveStreamPosition

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

Definition
public void setLiveStreamPosition(final long lTime)

Parameters
lTime - new position in live stream (milliseconds)

Return Value
Integer value is returned. 0 - on successful completion,
PLP_SEEK_COMPLETED notification will send on success. Otherwise – error result.

Remarks
Change the position of life stream played.

Examples
setStreamPosition(1000000);

getStreamPrebufferTime

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

Definition

public long getStreamPrebufferTime()

Parameters
There are no parameters for this call.

Return Value
Upon successful completion, getStreamPrebufferTime**()** returns pre-buffered position of played file/stream (in milliseconds).

Remarks
Provide the pre-buffered position that is played by player.

Examples

```
int position = getStreamPrebufferTime ();
```

setFFRate

Change speed of playback for local file and network stream.

Definition
public void setFFRate(final 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
setFFRate(2000); // Set playback rate to x2

getRenderPosition

Function provides last position in played media file. This function works only in case of file playback.

Definition
public long 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 = getRenderPosition();

Close

Disconnect from server and destroy pipeline.

Definition
public void Close()

Parameters
There are no parameters for this call

Return Value
Upon successful completion, **Close()** returns 0. Otherwise -1 is returned. All errors are provided in callback status.

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

Examples
player.Close ();

UpdateView
Set video output mode for current player instance.

Definition
public int UpdateView(final boolean isAspectRatioEnabled)
public int UpdateView()

Parameters
isAspectRatioEnabled – set aspect ratio that is set in network stream, 1 – set aspect ratio that is set in network stream, 0 – resize picture on full screen.

Return Value
Upon successful completion, isAspectRatioEnabled**()** returns 0, otherwise -1 is returned. All errors are provided in callback status.

Remarks
UpdateView(1) sets aspect ratio or full screen mode. This function can be used during playback. UpdateView() function uses settings that are set in player config structure.

Video output mode of output picture
player.getConfig().setAspectRatioMode(VideoOutputMode);
VideoOutputMode can be:
0 – stretch
1 – fit to screen with aspect ratio
2 – crop video
3 – 100% size of pciture
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 positions are 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);
// zoom and move modes are experimental functions, they may cause some issues.

Examples
Example #1
player.UpdateView (0);

Example #2
// Present video: picture size is 100% in the center of screen
player.getConfig().setAspectRatioMoveModeX(50); // 50% center of screen
player.getConfig().setAspectRatioMoveModeY(50); // 50% center of screen
player.getConfig().setAspectRatioZoomModePercent(100);//size is 100%
player.getConfig().setAspectRatioMode(5); // Zoom and move mode player.
UpdateView();

backgroundColor
Set background color of player.

Definition
public void backgroundColor(final int clr)

Parameters
clr – color in RGB format (ARGB is not supported).

Return Value
Upon successful completion, backgroundColor**()** returns 0, otherwise -1 is returned. All errors are provided in callback status.

Remarks
Set background color of player.

Examples
backgroundColor(Color.BLACK);

setVisibility
Set the enabled state of this view

Definition
public void setVisibility(int visibility)

Parameters
visibility – Controls the initial visibility of the view. Value of parameters are described in Android documentation.
http://developer.android.com/reference/android/view/View.html#attr_android:visibility

Return Value
No value is returned by function setStreamPosition.

Examples
player. setVisibility (1);

getVideoShot

Capture video picture from video stream.

Definition
public VideoShot getVideoShot(
final int desiredWidth,
final int desiredHeight
)

Parameters
desiredWidth - width of returned picture
desiredHeight - height of returned picture

Return Value
Upon successful completion, getVideoShot**()** returns VideoShot object.

```
public class VideoShot
{
        public int getWidth();
        public int getHeight();
        public ByteBuffer getData();
}
```

Remarks
Provide the video shot of last render frame in format ARGB_8888. This function works in Preview mode only.

Example

```
VideoShot vs = player.getVideoShot(width, height);
Bitmap bm = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888);
bm.copyPixelsFromBuffer(vs.getData());
```

GetStatFPS

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

Definition
public 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();

GetStatBitrate

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

Definition
public int GetStatBitrate ()

Parameters
There are no parameters for this call.

Return Value
Upon successful completion, **GetStatBitrate()** returns bitrate of network stream. It is the bitrate of a stream that is downloaded from network, otherwise -1 is returned. All errors are provided in callback status

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

Example
Int bitrate = player.GetStatBitrate();

GetDroppedFrame

Return number dropped frames on render.

Definition
public int GetDroppedFrame ()

Parameters
There are no parameters for this call.

Return Value
Upon successful completion, **GetDroppedFrame()** returns number of dropped frame in render provider.

Example
Int bitrate = player.GetDroppedFrame();

getInternalBuffersState

Return fullness of inner buffers in pipeline so application. Detail information are provided for every buffer in video and audio pipeline.

Definition
public BuffersState getInternalBuffersState ()

Parameters
There are no parameters for this call

Return Value
Upon successful completion, **getInternalBuffersState()** returns buffers fullness and size with detail information about every buffer, otherwise null is returned. All errors are provided in callback status

Remarks

int getBufferSizeBetweenSourceAndVideoDecoder() – Size of buffer between Source and Video decoder

int getBufferFilledSizeBetweenSourceAndVideoDecoder() – Fullness of buffer between Source and Video decoder

int getBufferSizeBetweenSourceAndAudioDecoder() – Size of buffer between Source and Audio decoder

int getBufferFilledSizeBetweenSourceAndAudioDecoder() – Fullness of buffer between Source and Audio decoder

int getBufferSizeBetweenVideoDecoderAndVideoRenderer() – Size of buffer between Video decoder and Render

int getBufferFilledSizeBetweenVideoDecoderAndVideoRenderer() – Fullness of buffer between Video decoder and Render decoder

int getBufferSizeBetweenAudioDecoderAndAudioRenderer() – Size of buffer between Audio decoder and Render

int getBufferFilledSizeBetweenAudioDecoderAndAudioRenderer() – Fullness of buffer between Audio decoder and Render decoder

int getBufferFramesSourceAndVideoDecoder() – number of video frames between network source and video decoder

int getBufferFramesBetweenVideoDecoderAndVideoRenderer() – number of video frames between video decoder and video render

int getBufferVideoLatency() – difference between latest received video frame on netowrk source and latest rendered video frame on video render (in milliseconds)

int getBufferAudioLatency() - difference between latest received audio sample on network source and latest rendered audio sample on audio render (in milliseconds)

Example :

```
BuffersState buf_state;
buf_state = player.getInternalBuffersState();
if (buf_state.getBufferSizeBetweenAudioDecoderAndAudioRenderer() != 0 &&
buf_state.getBufferSizeBetweenAudioDecoderAndAudioRenderer()!=0)
Log.e("TEST", "buf_level: Source: " +
buf_state.getBufferFilledSizeBetweenSourceAndAudioDecoder()*100/buf_state.getBuff
erSizeBetweenSourceAndAudioDecoder() + " Render: " +
buf_state.getBufferFilledSizeBetweenAudioDecoderAndAudioRenderer()*100/buf_stat
e.getBufferSizeBetweenAudioDecoderAndAudioRenderer());
```

GetStatPercFree

Return fullness of inner buffers in pipeline so application can estimate if device can playback data in real time or with latency (Video data only).

Definition

public int GetStatPercFree()

Parameters

There are no parameters for this call

Return Value
Upon successful completion, **GetStatPerFree()** returns level of capacity for inner buffers, otherwise -1 is returned. All errors are provided in callback status

Remarks
Return fullness of inner buffers in pipeline so application can estimate if device can playback data in real time or latency.

Example
Int buf_level = player.GetStatPerFree ();

GetDataDelayOnSource

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

Definition
public int GetDataDelaySource ()

Parameters
There are no parameters for this call

Return Value
Upon successful completion, **GetDataDelayOnSource()** returns the difference between when package is expected and when package comes

Remarks

Example
int delay = player.GetDataDelayOnSource ();

getDataBitrateOnSource

Return bitrate of network input stream comes on device.

Definition

public int getDataBitrateOnSource ()

Parameters

There are no parameters for this call

Return Value

Upon successful completion, **getDataBitrateOnSource ()** returns Return stream bitrate in kbps.

Remarks

Example

```
int delay = player.getDataBitrateOnSource();
```

IsHardwareDecoding

Return what decoder (s/w or h/w) is used by player.

Definition

public boolean IsHardwareDecoding ()

Parameters

There are no parameters for this call.

Return Value

Upon successful completion, IsHardwareDecoding returns true if h/w decoder is used and false for s/w decoder

Remarks

Shows if h/w or s/w video decoder is used in player.

Example

Boolean hw_decoder = IsHardwareDecoding ();

recordSetup

Set the recording settings.

Definition

public void RecordSetup(String record_path, int record_flags, int record_split_time, int record_split_size, String record_prefix)

Parameters

record_path - path for recorded files

record_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

PP_RECORD_FAST_START(0x00000040) - Run a second pass moving the index (moov atom) to the beginning of the file. This operation can take a while, and will not work in various situations such as fragmented output, thus it is not enabled by default.

PP_RECORD_FRAG_KEYFRAME(0x00000080) – Start a new fragment at each video keyframe

record_split_time - Size of chunks in milliseconds if flags are PP_RECORD_SPLIT_BY_TIME (in seconds)

record_split_size - Split stream on chunks by size if flags are PP_RECORD_SP LIT_BY_ SIZE (in megabytes)

record_prefix – Prefix is added to name of recorded files

Return Value

There is no return value.

Example

RecordSetup("Video_test",0x3, 60, 100, "test");

RecordStart

Start recording.

Definition

public void RecordStart();

Parameters

There are no parameters for this call

RecordStop

Stop recording.

Definition
public void RecordStop();

Parameters
There are no parameters for this call

RecordGetFileName

Retrieve the name of file has been recording.

Definition
public ***String*** RecordGetFileName(**int** param)

Parameters
int param; //0: get last stopped file. 1: get last started 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(int 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);

AudioGetCount

Retrieve a number of audio tracks.

Definition
public ***int*** AudioGetCount()

Parameters

none

Return Value

Returns a number of available audio tracks.

Remarks

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

AudioSelect

Select an audio track to play.

Definition

public ***int*** AudioGetCount(int track_i)

Parameters

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

Remarks

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

AudioGetSelected

Retrieve a selected audio track.

Definition

public ***int*** AudioGetSelected()

Parameters

none

Return Value

Returns selected audio track.

Remarks

AudioGetSelected() returns actually selected audio tack. In case player is not Opened state, AudioGetSelected() returns MediaPlayerConfig.getSelectedAudio() value.

startVolumeDetect

Starts audio volume detection process

Definition

public ***void*** startVolumeDetect(final int vd_max_samples)

Parameters

int vd_max_samples – a number of audio samples has to be processed

Return Value

void

Remarks

When startVolumeDetect() has finished ARP_VOLUME_DETECTED sent. Application can retrieve maximum and mean estimated audio volumes through getting the properties getPropInt(PlayerProperties.PP_PROPERTY_AUDIO_VOLUME_MAX) and getPropInt(PlayerProperties.PP_PROPERTY_AUDIO_VOLUME_MEAN).

setVolumeBoost

Gain audio volume

Definition

public *void* setVolumeBoost(final int volume_boost)

Parameters

int volume_boost – value of audio volume gain in range [-30, 30] dB

Return Value

void

Remarks

Application should first retrieve volume statistics through startVolumeDetect() and max_volume = getPropInt(PlayerProperties.PP_PROPERTY_AUDIO_VOLUME_MAX); mean_volume = setPropInt(PlayerProperties.PP_PROPERTY_AUDIO_VOLUME_MEAN). After all, set corresponded value **volume_boost** to avoid audio clipping in raising audio volume.

For instance app obtains mean_volume=-27, max_volume=-4.

It means that:

The mean square energy is approximately -27 dB.

The largest sample is at -4 dB, or more precisely between -4 dB and -5 dB.

In other words, raising the volume by +4 dB (**volume_boost=4**) does not cause any clipping.

SubtitleGetCount

Retrieves a number of subtitle tracks.

Definition

public ***int*** SubtitleGetCount()

Parameters

none

Return Value

Returns a number 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
public ***int*** SubtitleSelect(int track_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 player.Open().

SubtitleGetSelected

Retrieves a selected subtitle track.

Definition
public ***int*** SubtitleGetSelected()

Parameters
none

Return Value
Returns selected subtitle track.

Remarks

SubtitleGetSelected() returns actually selected subtitle track. In case player is not in Opened state, SubtitleGetSelected() returns MediaPlayerConfig.getSelectedSubtitle() value.

SubtitleSourceAdd

Add an external subtitle source

Definition

public ***int*** SubtitleSourceAdd(String path2)

Parameters

String path2 – path to subtitle source

Return Value

0 - OK. Otherwise error.

Remarks

SubtitleSourceAdd(String path2) 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

public ***int*** SubtitleSourceRemove(String path2)

Parameters

String path2 – path to subtitle source

Return Value

0 - OK. Otherwise error.

4. Thumbnailer

Thumbnailer is Class that provides the functionality to make thumbnails and stream information for local files and network streams. Smart searching is used to make thumbnails with maximum informativity.

4.1 Functions description

Following functions are member of Thumbnailer class. These functions should be used to get a thumbnail for file or network stream.

Open

Connect to network server or open local media file.

Definition
public Object Open(final String ConnectionUrl)
public Object Open(final ThumbnailerConfig config)

Parameters

ConnectionUrl	URL of network resource (RTSP, HLS, RTMP, MMS, UDP and so on) or full path of local media file.

Class ThumbnailerConfig provides additional setting to open Thunbnailer.

```
public ThumbnailerConfig( String connectionUrl,
                          int connectionNetworkProtocol,
                          int dataReceiveTimeout,
                          int numberOfCPUCores,
                          float bogoMIPS )
```

connectionUrl	URL of network resource (RTSP, HLS, RTMP, MMS, UDP and so on) or full path of local media file
connectionNetworkProtocol	Protocol for RTP or RTSP tunneling, 0 – RTP by UDP, 1 – RTP by TCP, 2 – RTSP and HTTP tunneling, -1 – AUTO mode
dataReceiveTimeout	reconnect timeout, SDK does reconnect if there is no received data during some time (milliseconds)

numberOfCPUCores	Number of CPU core to decode video, 0 – autodetect and set the number according device capability, positive number sets number according application needs

Return Value
Upon successful completion **Open**() returns 0. Otherwise - ERROR is returned.

Remarks
Connect to network resource or open local media file, create pipeline, allocate resource. This function should be called before get Frame.

Example #1
thumbnailer.Open("http://example.com");
Example #2
thumbnailer.Open(ThumbnailerConfig);

getFrame

Capture thumbnail frame.

Definition
public ThumbnailFrame getFrame()

Parameters
There are no parameters for this call.

Return Value
Upon successful completion, getFrame returns ThumbnailFrame object.

```
public class ThumbnailFrame
{
        public int getWidth();
        public int getHeight();
        public ByteBuffer getData();
}
```

Remarks

Provide the thumbnail for local file or stream in format ARGB_8888.

Example

```
ThumbnailFrame frame = thumbnailer.getFrame();
shot.getData().rewind();
Bitmap bmp = Bitmap.createBitmap(shot.getWidth(), shot.getHeight(),
Bitmap.Config.ARGB_8888);
bmp.copyPixelsFromBuffer(shot.getData());
```

getInfo

Function returns the information about media file or network stream.

Definition

```
public String getInfo()
```

Parameters

There are no parameters for this call.

Return Value

Type – String, Format - xml.

Remarks

Provide the information about media file or network stream.

String is xml format like below:

```
<?xml version=1.0?>
<StreamInfo name="AVFileFormat" version="1.0">
<name          value="test.mp4" />
<duration      value="100" />
        <VideoStreams>
                <VideoStream id=0 >
                        <format        value="h264" />
                        <duration      value="100"    />
                        <width         value="1920" />
                        <height        value="1080" />
                        <fps           value="30"   />
```

```
            </VideoStream>
        </VideoStreams>
        <AudioStreams>
            "<AudioStream id=1 >
                <format      value="aac"     />
                <duration    value="100"     />
                <samplerate  value="48000" />
                <channels    value="2"       />
            </AudioStream>
        </AudioStreams>
</StreamInfo>
```

Examples

String info = thumbnailer.getInfo();

GetState

Return thumbnailer state: .

Definition

public ThumbnailerState getState()

Parameters

There are no parameters for this call

Return Value

Following states are provided:

0 - Opening,

1 - Opened,

2 - Closing,

3 - Closed;

Remarks

Provide the current state of Thumbnailer.

Example

if (thumbnailer.getState() == ThumbnailerState.Opened);

Close

Disconnect from server or close file and destroy all resources.

Definition
public void Close()

Parameters
There are no parameters for this call

Return Value
Upon successful completion, **Close()** returns 0. Otherwise – ERROR is returned.

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

Example
thumbnailer.Close ();

toggleMute

Control mute and unmute on audio render.

Definition
public void toggleMute(final boolean 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

players.toggleMute(true);

setOnDataListener

Set data receiver callback to get video and audio data.

Definition

public void setOnDataListener(final MediaPlayerCallbackData callback)

public void setOnDataListener(final MediaPlayerCallbackData callback, int callbackMask)

public void setOnDataListener(final MediaPlayerCallbackData callback, int callbackMask, PlayerCallbackDataConfig callbackDataConfig)

Parameters

callback – SDK will call if there are any video frame or audio sample according defined mask.

callbackMask – masks that set what data will be provided to application level.

All mask are enumed in class PlayerCallbackDataMask:

There are following masks :

PP_CALLBACK_DATA_ALL(0xFFFFFFFF) – Provide media data from all sources

PP_CALLBACK_DATA_OFF(0x00000000) – do not provide data

PP_CALLBACK_DATA_SOURCE_VIDEO_DATA(0x00000001) – Provide video frames that come from network source (encoded frame in various formats : H.264, MPEG2 and other

PP_CALLBACK_DATA_SOURCE_AUDIO_DATA(0x00000002) - Provide audio samples that come from network source (encoded frame in various format : AAC, G711 and other

PP_CALLBACK_DATA_RENDERER_VIDEO_DATA(0x00000008) – Provide video frames that come on video render in various formats : YUV, NV12 and other .Format depends from input stream , device , decoder and other

PP_CALLBACK_DATA_RENDERER_AUDIO_DATA(0x00000010) - Provide audio samples that come on audio render (raw audio sample , format : PCM , 16 bit , little ending)

PlayerCallbackDataConfig - config file to set cropping and other setting .

Return Value

Application sets callback to get video and audio data . Application can get encoded or decoded data.

Examples

```
int callbackDataMask =
PlayerCallbackDataMask.forType(PlayerCallbackDataMask.PP_CALLBACK_DATA_RENDERER_VIDEO_DATA);
PlayerCallbackDataConfig dataConfig = player.new PlayerCallbackDataConfig();
// Enable crop here
```

```
if (btnVideoCrop.isChecked())
        {
           int video_width = 1920;
           int video_height = 1080;

           int crop_width = 640;
           int crop_height = 480;

           int crop_left = (video_width / 2) - (crop_width / 2);
           int crop_top = (video_height / 2) - (crop_height / 2);
           int crop_right = crop_left + crop_width;
           int crop_bottom = crop_top + crop_height;

            dataConfig.setVideoRendererFrameCropRect(new Rect(crop_left, crop_top,
    crop_right, crop_bottom));
        }
player.setOnDataListener(mthis, callbackDataMask, dataConfig);
```

getPlayerCallbackDataConfig

Return Data config that is used to set additional option to capture video.

Definition
public PlayerCallbackDataConfig getPlayerCallbackDataConfig()

Parameters
No input parameters

Return Value
PlayerCallbackDataConfig set config for callback data.

```
public class PlayerCallbackDataConfig
  {
    public Rect getVideoRendererFrameCropRect() { return videoRendererFrameCropRect;
}
public void setVideoRendererFrameCropRect(final Rect videoRendererFrameCropRect)    }
```

Remarks

getPlayerCallbackDataConfig return current config.

Examples

```
PlayerCallbackDataConfig  config = players.getPlayerCallbackDataConfig ();
```