APlayerAndroid 接口说明

目录

1 接口说明.....3
1.1 APlayerAndroid().....3
1.2 void destroy().....3
1.3 void useSystemPlayer(boolean use).....3
1.4 boolean isSystemPlayer().....3
1.5 int setView(SurfaceView surfaceview).....3
1.6 int setView(Surface surface).....4
1.7 int View createVRView(Context content).....4
1.8 int open(String).....4
1.9 int close().....5
1.10 int play().....5
1.11 int pause().....6
1.12 String getVersion().....6
1.13 int getState().....6
1.14 int getDuration().....7
1.15 int setConfig(int configID,String value).....7
1.16 String getConfig(int configID).....7
1.17 int setPosition (int msec).....12
1.18 int getPosition().....12
1.19 int getVideoWidth().....12
1.20 int getVideoHeight().....12
1.21 int getBufferProgress().....13
1.22 int MediaInfo parseThumbnail(String mediaPath, long timeMs, int width, int height).....13
1.23 boolean startRecord(String outMediaPath).....13
1.24 boolean isRecord().....13
1.25 void endRecord().....14
2 事件注册.....14
2.1 文件打开成功事件.....14
2.2 播放状态改变事件.....14
2.3 打开调用完毕事件.....15
2.4 播放完成事件.....15
2.5 缓冲进度改变事件.....16
2.6 媒体 Seek 完成事件.....16
2.7 Surface 销毁事件：.....16
2.8 字幕更新事件.....17
3 常见问题.....17
4 更新记录.....17
4.1 V1.1.0.29 更新.....17

4.2 V1.1.0.30 更新 18
4.2 V1.2.0.100 更新 18

1 接口说明

1.1 APlayerAndroid()

APlayerAndroid 构造函数；

1.2 void destroy()

彻底释放播放器资源，该函数一般最后调用。

和 Close()区别：Destroy()会释放内部资源，Close()不会释放仍会持有资源，Close()后再次打开播放器更快(*1.9 int close()*)。

1.3 void useSystemPlayer(boolean use)

设置是否使用系统播放器，默认不使用系统播放器（使用 APlayer 实现的播放器）。

注意：请保证仅在 Open(String)前调用该函数，不建议使用系统播放器，系统播放器和平台相关，且支持的格式较少，除 MP4 格式外不建议考虑系统播放器，如果当前媒体用系统播放器打开或者播放过程中失败，都会转到非系统播放器播放。

1.4 boolean isSystemPlayer()

如当前使用的是系统播放器，则返回 true。即使当前设置是系统播放器播放，但是如果系统播放器打开或者播放失败，则返回 false;

1.5 int setView(SurfaceView surfaceview)

设置渲染表面，就是显示视频图像的组件，如果这个函数没有调用，将无法显示图像。对于软解，可在任何时候设置渲染表面，但是对于硬解和使用系统播放，必须在 Open 之前设置渲染表面。软解的时候，画面会按照一定的纵横比（视频默认或者用户设置的纵横比）显示在 surfaceview 中，但是硬解和系统播放时画面会铺满整个 surfaceview 中，因此如果你想要一定的画面纵横比只能通过改变 surfaceview 的 Size 来达到，如果你需要视频默认的纵横比，只能在打开视频成功后根据视频的宽高，重新设置 surfaceview 的大小。即使对于软解，我也希望你设置 surfaceview 的宽高比为你希望的宽高比，因为字幕的显示是根据 surfaceview 的位置来显示的。

1.6 int setView(Surface surface)

同 int SetView(SurfaceView surfaceview)；

1.7 int View createVRView(Context content)

创建 VR View，创建 View 后需要手动设置大小。

```
//Create view
View VRView = CreateVRView(content);
//Set VRView Size and postion
…
```

1.8 int open(String)

打开一个视频文件，参数是视频文件路径，可以是本地文件，也可以是支持的网络协议媒体流。该函数立即返回，调用成功返回 0（但并不表示视频文件打开成功，只是函数调用成功（可通过注册 *2.1 文件打开成功事件*或 *2.3 打开调用完毕事件*进行处理）。

注意：如果是替换当前正在播放的文件，需要先调用 Close()，并等待播放完成事件(*2.4 播放完成事件*)，播放完成事件发生后，与当前媒体文件（流）的相关操作失效，需要再次调用 Open（）。

strUrl

[输入参数] 一个表征媒体文件地址的字符串，该字符串可以是本地、局域网共享或网络文件，例如:

1、普通视频：
/test.rmvb
\\192.168.8.188\share\test.rmvb
http://218.221.12.181/test.rmvb
ftp://218.221.12.181/test.rmvb
rtmp://218.221.12.181/test.flv
rtsp://218.221.12.181/test.sdp

2、分段拼接视频：
APlayer 也支持无缝播放分段视频地址组成的 m3u8 地址，例如：
http://218.221.12.181/test.m3u8?some_param=value

这 m3u8 中的分段视频地址可以是标准的 ts, 也可以是非标准的 flv、mp4，只要这些分段编码格式一致即可。

调用者还可以把一些分段视频自行组织成一个本地 m3u8 文件给 APlayer 去无缝拼接播放，m3u8 的内容格式如下：

```
#EXTM3U
#EXTINF:20,
http://218.221.12.181/1.flv
#EXTINF:20,
http://218.221.12.181/2.flv
#EXTINF:20,
http://218.221.12.181/3.flv
#EXT-X-ENDLIST
```

上述 m3u8 中 #EXTINF:20 中的 20 代表下面的视频 http://218.221.12.181/1.flv 是 20 秒时长。

APlayer 也支持不设置时长，即无 "#EXTINF:??" 这些行，这时 APlayer 会获取第一个分段地址的时长乘以分段地址的个数来估算总时长，用以支持拖动播放进度，不过没设置时长那么准确。

3、压缩包视频：

APlayer 还支持播放 ZIP 格式压缩包中的文件，例如:

H:\1.zip?/abc/test.mp3

其中 "?" 前的部分是 zip 文件的路径, "?" 后面的部分是压缩文件在 zip 包中的相对路径。

返回值：

调用成功返回 0

1.9 int close()

关闭播放器,成功返回 0。

1.10 int play()

开始播放视频。视频打开成功后不会立刻播放，而是转为暂停状态，如果需要播放视频，需要调用该函数，因此该函数一般是在 Open 成功事件中调用，或者从暂停状态恢复播放状态时调用。该函数立即返回，返回值为-1 表示失败，返回值为 0 表示成功。

1.11 int pause()

暂停视频播放。暂停音视频解码和渲染，数据读取过程继续直至填满内部缓冲区，成功返回 0。

1.12 String getVersion()

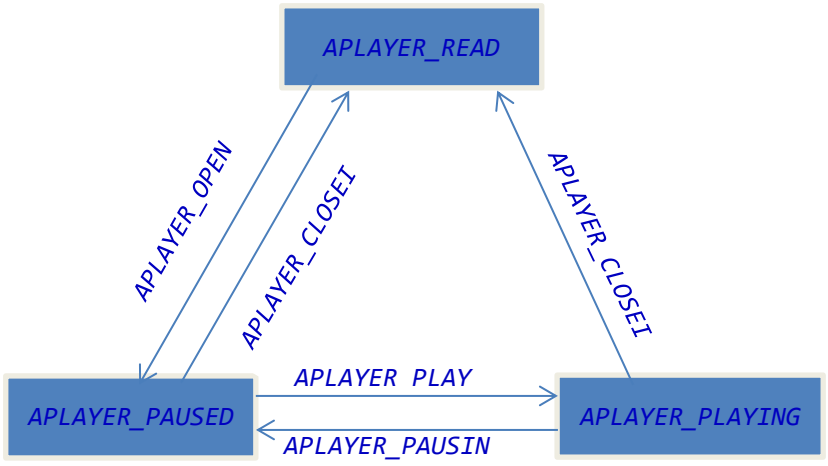
返回当前 jar 包版本字符串，形如："1.2.0.100"

1.13 int getState()

得到播放器目前的状态。播放器有7个如下状态。

```
public class PlayerState
{
    public static final int APLAYER_READ      = 0;
    public static final int APLAYER_OPENING   = 1;
    public static final int APLAYER_PAUSING   = 2;
    public static final int APLAYER_PAUSED    = 3;
    public static final int APLAYER_PLAYING   = 4;
    public static final int APLAYER_PLAY      = 5;
    public static final int APLAYER_CLOSEING   = 6;
}
```

7个状态中*APLAYER_READ*，*APLAYER_PAUSED* *APLAYER_PLAYING*是稳态 *APLAYER_OPENING*、*APLAYER_PAUSING*、*APLAYER_PLAY*、 *APLAYER_CLOSEING*是瞬时状态，各个状态之间可以互相转换，状态转换图如下



有多种情况会使播放器进入 PS_READY 状态，进入原因可通过 GetConfig 方法查询 CONFIGID.*PLAYRESULT* 信息（见 *1.16 String getConfig(int configID)*）。

1.14 int getDuration()

得到视频的 duration,单位为毫秒。视频打开成功后调用该函数获得视频的 duration,但视频文件可能不包含该字段信息，这时得到的 duration 为-1。如果使用非系统播放器，内部可以通过读取一定数量的包之后来估算视频的 duration,播放一段时间可获取到 duration 估计值（等待时间越久相对越精确）。

1.15 int setConfig(int configID,String value)

详见 *1.16 String getConfig(int configID)*。

1.16 String getConfig(int configID)

设置和获取 APlayer 的参数，APlayer 中将所有非基本的功能集中在该函数中，以减少接口数量。configID 是属性 ID，所有的属性 ID 都定义在类 CONFIGID 中，value 和 GetConfig 的返回值是属性值，对于不同的属性，value 是字符串形式，但不同参数格式不同。属性配置如下：

```
public class CONFIGID
{
    public static final int PLAYRESULT              = 7;
    public static final int AUTO_PLAY               = 8;
    public static final int READPOSITION            = 31;
    public static final int UPDATEWINDOW            = 40;

    public static final int ORIENTATION             = 41;
    public static final int ASPECT_RATIO_NATIVE     = 203;
    public static final int ASPECT_RATIO_CUSTOM     = 204;

    public static final int HW_DECODER_USE          = 209;
    public static final int HW_DECODER_ENABLE       = 230;
    public static final int HW_DECODER_DETEC        = 231;

    public static final int AUDIO_TRACK_LIST        = 402;
    public static final int AUDIO_TRACK_CURRENT     = 403;

    public static final int SUBTITLE_USABLE         = 501;
```

```
        public static final int SUBTITLE_EXT_NAME       = 502;
        public static final int SUBTITLE_FILE_NAME      = 503;
        public static final int SUBTITLE_SHOW           = 504;
        public static final int SUBTITLE_LANGLIST       = 505;
        public static final int SUBTITLE_CURLANG        = 506;

        public static final int HTTP_COOKIE             = 1105;
        public static final int HTTP_REFERER            = 1106;
        public static final int HTTP_CUSTOM_HEADERS     = 1107;
        public static final int HTTP_USER_AGENT         = 1108;

        public static final int NET_BUFFER_ENTER        = 1001;
        public static final int NET_BUFFER_LEAVE        = 1002;
        public static final int NET_BUFFER_READ         = 1003;
        public static final int NET_BUFFER_READ_TIME    = 1005;
        public static final int NET_SEEKBUFFER_WAITTIME = 1004;

        public static final int VR_ENABLE               = 2401;
        public static final int VR_ROTATE               = 2411;
        public static final int VR_FOVY                 = 2412;
        public static final int VR_MODEL                = 2413;
        public static final int VR_ENABLE_INNER_TOUCH_ROTATE  =
2414;
    }
```

PLAYRESULT:

支持函数：GetConfig

返回播放结束的结果。有三种原因导致播放结束。

返回“0”—— 视频播放完成。

返回“1”—— 用户主动结束播放。

其他字符串（非 0 非 1） —— 播放出错结束，返回错误代码。

AUTO_PLAY:

支持函数：GetConfig/SetConfig

文件打开成功后是否自动播放，默认不播放

“0”—— 打开成功后自动播放。

“1”—— 打开成功后不自动播放（需要调用 Play（）播放）。

READPOSITION:

支持函数：GetConfig

返回当前视频缓冲位置，单位是毫秒，注意是字符串形式。该缓冲位置只是播放内核的缓冲位置，表示播放内核已经读取到了该位置的音视频数据包，并不表示网络播放中已经读到本地的视频数据位置。

UPDATEWINDOW:

支持函数：SetConfig

只用于 SetConfig 刷新窗口。

ASPECT_RATIO_NATIVE:

支持函数：GetConfig

获取视频的自然宽高比，格式："4;3"。

ASPECT_RATIO_CUSTOM:

支持函数：GetConfig/SetConfig

视频的自定义宽高比，格式："4;3"。

例：SetConfig(CONFIGID.ASPECT_RATIO_CUSTOM,"4;3");

纵横比设置只对软解有用，对于硬解和使用系统播放器无效，此时应该通过改变 surfaceview 的大小来改变纵横比

HW_DECODER_USE:

支持函数：GetConfig/SetConfig

获取/设置硬件解码状态，默认关闭。

"0" —— 关闭。

"1" —— 开启。

需要在 Open 之前设置该参数，Open 之后设置对当前播放的媒体无效。如果媒体不支持硬解 设置该参数为"1",也只能是软解。注意硬件解码非 VR 模式下，需要用户自己调整画面比例。

HW_DECODER_ENABLE:

支持函数：GetConfig

获取系统是否支持当前媒体硬件解码

"0" —— 不支持。

"1" —— 支持。

HW_DECODER_DETEC:

支持函数：GetConfig/SetConfig

是否开启硬件解码侦测，侦测是否支持当前媒体硬件解码，默认开启侦测，如不需要硬件解码，可在 Open()前设置关闭。

"0" —— 不开启。

"1" —— 开启。

如果确定不需要硬解，可以设置为"0"，毕竟侦测媒体是否支持硬解需要执行一些代码

AUDIO_TRACK_LIST:

支持函数：GetConfig

获取音轨列表。音轨列表，格式: "语言,音轨 1;语言,音轨 2"。

AUDIO_TRACK_CURRENT:

支持函数：GetConfig/SetConfig
获取或设置当前音轨索引，索引值从”0”开始。

SUBTITLE_USABLE：

支持函数：GetConfig
获取字幕加载功能是否可用。
“0” ——　不可用字幕加载
“1” ——　可用字幕加载

SUBTITLE_EXT_NAME：（待实现）

支持函数：GetConfig
支持的字幕格式列表，例如："srt;ssa;ass;idx"。

SUBTITLE_FILE_NAME：

支持函数：GetConfig/SetConfig
获取/设置外挂字幕的文件名，例如："/subtitle.srt"。

SUBTITLE_SHOW：

支持函数：GetConfig/SetConfig
获取/设置字幕状态（显示/隐藏）
”0” ——　隐藏。
”1” ——　显示。
默认字幕可用，由 APlayer 内部实现显示，显示位置对齐 surfaceview 的下方，如果使用软解，画面没有铺满整个 surfaceview， 字幕的显示位置可能离画面很远，可以通过改变 surfaceview 的大小来使字幕对齐画面的下方。
如果想更灵活的控制字幕的位置，大小，字体，颜色等 用户可以通过接受字幕消息事件来自己控制字幕的显示（详见 *2.8 字幕更新事件*）。

SUBTITLE_LANGLIST：

支持函数：GetConfig
获取当前加载的字幕的可用语言列表，用";"分割，例如："chinese;english"。
如果加载了外挂字幕，外挂字幕也在字幕列表中，位于内嵌字幕后面

SUBTITLE_CURLANG：

支持函数：GetConfig/SetConfig
获取/设置 当前选择的字幕语言索引，例 ”0”,没有选择是返回无效的负值。

HTTP_COOKIE：

支持函数：GetConfig/SetConfig
获取/设置 HTTP 请求头中带的 Cookie 字符串，默认为无。

HTTP_REFERER：

支持函数：GetConfig/SetConfig

获取/设置 HTTP 请求头中带的 Referer 字符串，默认为无。

HTTP_CUSTOM_HEADERS：

支持函数：GetConfig/SetConfig

获取/设置 HTTP 自定义头字段列表，每个头字段之间用回车换行符号 "\r\n"(即 0x0d，0x0a) 分割。

HTTP_USER_AGENT

支持函数：GetConfig/SetConfig

获取/设置 HTTP 请求头中带的 User Agent 字符串，默认为 APlayer 的默认值。

NET_BUFFER_ENTER：

支持函数：GetConfig/SetConfig

获取/设置 当网络没有读取到数据时，等待多久进入缓冲状态，单位毫秒,注意这里的时间是字符串。

备注：缓冲状态一般会暂停播放持续一段时间，以填满缓冲队列。

NET_BUFFER_LEAVE：

支持函数：GetConfig/SetConfig

获取/设置 在缓冲状态下，缓冲多少个帧退出缓冲，默认为“1000”。

NET_BUFFER_READ：

支持函数：GetConfig/SetConfig

获取/设置 最多预先读取多少个帧到内存中去。

NET_SEEKBUFFER_WAITTIME：

支持函数：GetConfig/SetConfig

获取/设置 拖动播放进度后，没有数据时多久进入缓冲（缓冲一直到队列满，较耗时），单位毫秒。

NET_BUFFER_READ_TIME：

支持函数：GetConfig/SetConfig

获取/设置 最多预先读取多长时间到内存中去，单位毫秒。

VR_ENABLE：

支持函数：GetConfig/SetConfig

获取/设置 VR 是否可用。

“1” —— 可用

“0” —— 不可用

VR_ROTATE：

支持函数：SetConfig

设置 VR 旋转角度（单位：角度）。格式：” X 角度;Y 角度”, 例如”45;60”,该函数立即旋转图片角度，无需相应的获取函数。

“1” ——　　可用
“0” ——　　不可用

VR_FOVY：

支持函数：GetConfig/SetConfig

获取/设置 VR 观察视角（单位：角度）。例如”90.0”，默认视角为 90°，视角过大容易产生畸变，过小不便于观察。

VR_MODEL：

支持函数：GetConfig/SetConfig

获取/设置 VR 模式。

“1” ——　　普通 VR
“2” ——　　上下 3D VR

注意，该函数调用，需要后置于 CreateVRView（）；

VR_ENABLE_INNER_TOUCH_ROTATE：

支持函数：GetConfig/SetConfig

获取/设置 VR 视图触摸时，场景是否同步旋转。

“1” ——　　旋转
“0” ——　　不旋转

1.17 int　　setPosition (int msec)

设置当前播放进度，单位为毫秒，成功返回 0。

1.18 int　　getPosition()

得到当前播放进度，单位为毫秒。

1.19 int　　getVideoWidth()

返回获取到的视频宽度，单位像素(pixel)，如不包含视频流，则返回 0。打开成功后才能正确得到视频的宽度

1.20 int　　getVideoHeight()

返回获取到的视频高度，单位像素(pixel)，如不包含视频流，则返回 0。打开成功后才能正确得到视频的高度

1.21 int getBufferProgress()

获取当前缓冲进度 [0, 100],不在缓冲和返回负值，缓冲进度也可以在缓冲消息中获取。

1.22 int MediaInfo parseThumbnail(String mediaPath, long timeMs, int width, int height)

获取媒体信息。
mediaPath: 媒体路径
timeMs: 指定时间点的截图（单位毫秒）
width: 生成的截图宽度
height: 生成的截图高度
返回值：成功返回 MediaInfo,失败返回 NULL

```
public static class MediaInfo
{
    public int width;
    public int height;
    public long duration_ms;
    public byte[] bitMap;

    public static Bitmap byteArray2BitMap(byte[] bitMap){};
}
```

1.23 boolean startRecord(String outMediaPath)

开始视频录制截取，请确保当前视频没有处于录制状态，录制的视频默认格式为原视频格式，和输入的文件名无关。
注意：视频录制中不能手动改变视频播放进度。
返回值：成功返回 true，失败返回 false。

1.24 boolean isRecord()

获取当前媒体，是否处于录取状态中。
当视频处于录制中，返回 true。

1.25 void endRecord()

停止录制。

2 事件注册

可注册关注的事件，以响应特定事件，一个事件只能注册一个监听器，后注册的监听器会覆盖先注册的监听器。

注意：事件函数中，不要执行耗时操作。

2.1 文件打开成功事件

```
void setOnOpenSuccessListener (OnOpenSuccessListener listener);

public interface OnOpenSuccessListener{
   void onOpenSuccess();
}
```

在文件打开成功后，会产生该事件。

如果没有设置自动播放，在该事件接口函数中调用 Play（）就可以开始播放了。要获得视频的宽高，时长的信息 也需要在视频打开成功之后。

该事件与 OnOpenCompleteListener 事件在回调函数的参数为 TRUE 时一模一样。

2.2 播放状态改变事件

```
void setOnPlayStateChangeListener (OnPlayStateChangeListener listener);

public interface OnPlayStateChangeListener{
   void onPlayStateChange(int nCurrentState,int nPreState);
}
```

在播放器状态改变时，会产生该事件。

nCurrentState：当前状态。

nPreState：改变前的状态。

如（1.12）对状态的描述可知，状态的改变可以知道很多的事情，但其中很多事情都有了独立的事件，因此一般情况下没有必要使用该事件。

2.3 打开调用完毕事件

```
void setOnOpenCompleteListener (OnOpenCompleteListener listener);

public interface OnOpenCompleteListener{
   void onOpenComplete(boolean isOpenSuccess);
}
```

在文件打开操作执行完毕后，会产生该事件。

isOpenSuccess：文件是否打开成功

当参数 isOpenSuccess 为 TRUE 时　该事件与 OnOpenSuccessListener 事件一样。

当参数 isOpenSuccess 为 FLASE 时 该事件与 OnPlayCompleteListener 事件在参数为 PLAYRE_RESULT_OPENRROR(“0x80000001”)时一样。

从上面可以看出，这个事件完全可以被别的事件替代，因此需要注意可能接受相同的消息而导致一些动作重复执行而发生错误

2.4 播放完成事件

```
void setOnPlayCompleteListener(OnPlayCompleteListener listener);

public interface OnPlayCompleteListener{
   void onPlayComplete(String playRet);
}
```

播放完成，该事件将会被调用。

playRet：播放完成返回的结果码。

为“0”—— 视频播放完成。

为“1”—— 用户主动结束播放。

其他字符串（非 0 非 1） —— 播放出错结束，返回错误代码。

```
public class PlayCompleteRet{
    public static final String PLAYRE_RESULT_COMPLETE          = "0x0";//播放结束
    public static final String PLAYRE_RESULT_CLOSE             = "0x1";//手动关闭
    public static final String PLAYRE_RESULT_OPENRROR          = "0x80000001";
    public static final String PLAYRE_RESULT_SEEKERROR         = "0x80000002";
    public static final String PLAYRE_RESULT_READEFRAMERROR    = "0x80000003";
    public static final String PLAYRE_RESULT_CREATEGRAPHERROR  = "0x80000004";
    public static final String PLAYRE_RESULT_DECODEERROR       = "0x80000005";
    public static final String PLAYRE_RESULT_HARDDECODERROR    = "0x80000006";
}
```

当 playRet 为 PLAYRE_RESULT_OPENRROR 时 表示打开失败与 OnOpenCompleteListener

事件在参数为 FLASE 时表示相同的事件。

当 playRet 为 PLAYRE_RESULT_HARDDECODERROR 时表示因为硬件解码出错而导致播放结束。因为硬解出错可能无法恢复，因此建议在进程的此次生命周期之内不再使用硬解。播放事件发生后，当前媒体文件（流）的相关操作失效，需要再次调用 Open（）。

2.5 缓冲进度改变事件

```
void setOnBufferListener(OnBufferListener listener);

public interface OnBufferListener{
   void onBuffer(int progress);
}
```

缓冲数据，当缓冲进度改变时，会产生该事件。

Progress：当前新的缓冲进度。

在非缓冲状态下，如果首次接受到该消息，则播放器进入缓冲状态（不论缓冲进度是多少），当缓冲进度为 100 时 缓冲结束 退出缓冲状态。

注意播放器进入缓冲后，播放器的状态并没有发生改变.

2.6 媒体 Seek 完成事件

```
void setOnSeekCompleteListener(OnSeekCompleteListener listener);

public interface OnSeekCompleteListener{
   void onSeekComplete();
}
```

当 Seek 结束时，会产生该事件。

如果用户想在 Seek 结束后给用户提示，可以定制该事件。

2.7 Surface 销毁事件：

```
void setOnSurfaceDestroyListener(OnSurfaceDestroyListener
listener);

public interface OnSurfaceDestroyListener{
   void OnSurfaceDestroy();
}
```

当媒 Surface 销毁时，会产生该事件。

当使用硬解或者系统播放器时，Surface 销毁后需要停止播放器，不然可能导致崩溃或者黑屏。注意 Surface 销毁时 Activity 也可能销毁了，你可能在 Activity 销毁事件里面已经做

了处理，然后又在该事件中又重复做了处理，然后导致冲突。

2.8 字幕更新事件

```
void setOnShowSubtitleListener(OnShowSubtitleListener listener);

public interface OnShowSubtitleListener{
   void OnShowSubtitle(String subtitle);
}
```

当字幕需要更新时，会产生该事件。
Subtitle 内容不为空 —— 显示新的字幕
Subtitle 内容为空 —— 清除显示的字幕。

可以注册该事件，来自定义实现字幕显示，用户无需关心字幕显示和消失的时间，消息会在正确的时间出现，但自定义显示时，应关闭 APlayer 的内部显示 SetConfig(APlayerAndroid .SUBTITLE_SHOW, “0”);。

3 常见问题

1） 部分机型，在软硬解切换后，屏幕变黑或图像不动，一般是 SurfaceView 无效导致，将 SurfaceView 重绘即可。

```
//stop player
//reopen player
displayView.setVisibility(View.GONE);
...
displayView.setVisibility(View.VISIBLE);
```

4 更新记录

4.1 V1.1.0.29 更新

1) PlayCompleteRet 中，自动播放结束事件、手动关闭完成事件 进行更新。
自动播放结束事件 PLAYRE_RESULT_COMPLETE 返回值由”0” 更新为 ”0x0”。
手动关闭完成事件 PLAYRE_RESULT_CLOSE 返回值由”1” 更新为 ”0x1”。

4.2 V1.1.0.30 更新

1) setConfig() VR_MODEL 暂不支持 左右 3D，同步文档。

4.2 V1.2.0.100 更新

1) 支持 H265 硬件解码。
2) 提高 H264 硬件解码兼容性。
3) 支持视频截取功能（MP4、FLV、AVI、3GP、MKV、MOV、TS）。
4) 支持视频缩略图获取，
5) 修复了一些崩溃问题。