阿里巴巴集团

SDK 参考手册

安卓媒体播放器 SDK 参考手册

文档修订历史

版本	作者	工作描述	修订历史	修改日期
1.0	石惊	安卓媒体播放器SDK文档		2015-08-03
2.0	新野	安卓媒体播放器SDK文档		2016-05-24
2.1	新野	播放器在首帧、卡顿、seek、延迟、surface设置等方面优化，增加了相应的接口；		2016-07-11

参考文献

无

目录

文档说明

本文档面向所有使用该SDK的开发人员、测试人员以及对此感兴趣的用户，要求读者具有一定的Android开发能力。

1. 简介

安卓媒体播放器SDK是在安卓平台上使用的软件开发工具包(Soft Development Kit)，为Android开发者提供简单易用的接口，帮助开发者实现Android平台上的媒体播放应用开发。

安卓播放器SDK对目前主流的视频格式都提供了良好的支持，支持本地和网络媒体的播放，弥补了系统播放器在媒体格式上的不足。

1.1 功能说明

1. 方便快捷、低门槛实现媒体播放功能。用户无须关心内部实现细节，只需要自定义界面既可以实现专业级的播放应用。
2. 流媒体格式支持HLS、RTMP、HTTP FLV。
3. 文件媒体格式支持: mp4。
4. 编码格式支持H264+AAC。

1.2 安装包说明

播放器SDK的完整下载包中包含demo、doc、lib等：

1. demo：主要存放了调用SDK的源码示例工程，可以帮助用户了解如何使用该SDK。

2. lib：播放器SDK开发包,本SDK包括jar文件和so文件 。

3. doc：存放SDK相关接口文档。

   注：本Demo和SDK都是基于maven构建，使用时需要准备好自己的android maven 开发环境。

1.3 播放器性能

1. 目前播放器SDK对H.264、AAC格式视频采用的是硬解播放。

2. SDK的大小：

   libs/alivc-player-2.1.0.jar (0.08M)
   libs/armeabi/libffmpeg.so (3.4M)
   libs/armeabi/libtbMPlayer.so (0.5M)
   libs/arm64-v8a/libffmpeg.so (4.0M)
   libs/arm64-v8a/libtbMPLayer.so (0.8M)

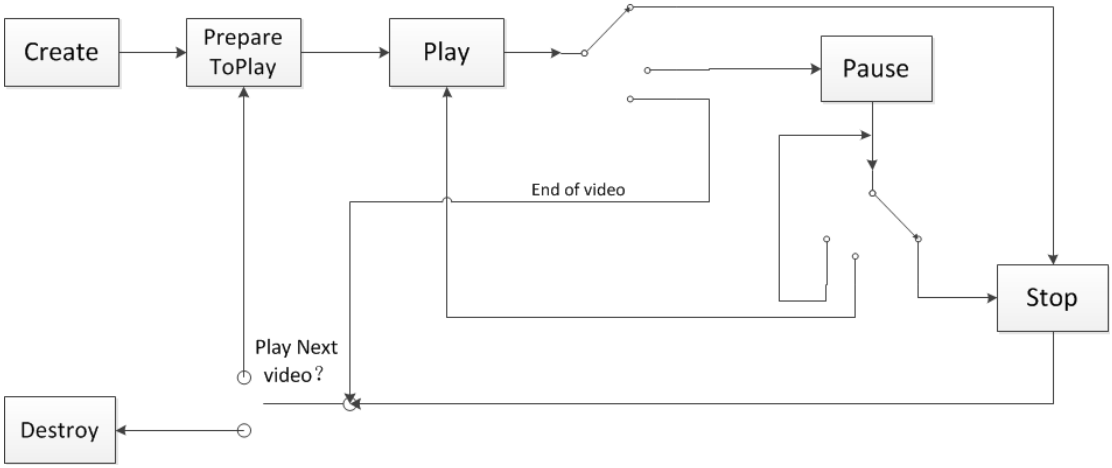
1.4 注意事项

1. 播放器SDK目前只支持单实例播放。不能够同时开2个播放器实例，同时只能存在一个实例，需要另开实例的时候，需要关闭之前存在的实例。

2. 如果使用硬件解码，操作系统版本需要在Android4.0以上。

2. 系统框架

2.1 系统框架图



3. 使用说明

3.1 开发环境配置

1. 需要配置好maven 的Android 开发环境。
2. 在阿里云官网上注册云帐号，并开通视频点播或视频直播服务。
   视频点播服务开通
   视频直播服务开通
3. 通过访问控制服务创建播放器专用子帐号及其AccessKey：
   a. 登陆访问控制服务控制台
   b. 在用户管理中新建用户：

注意勾选为该用户自动生成AccessKey 选项：

创建子帐号成功，注意保存好该帐号的AccessKey：

c. 为子帐号分配调用播放器权限：

点击**授权**链接：

在 **可选授权策略名称** 中搜索mts，将 **AliyunMTSPlayerAuth**，授予此子帐号：

3.2 开发步骤

首先在安卓应用程序中，需要声明以下权限：

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

使用用媒体播放器SDK的调用顺序为：

1. 创建AliVcMediaPlayer播放接口。
2. 注册事件通知函数。
3. 设置缺省解码方式：如果缺省为硬解，会尝试使用硬解，如果失败使用软解；如果缺省为软解，那么会一直使用软解。
4. 如果从历史起点播放，那么调用seek方法；
5. 调用prepareAndPlay准备开始播放。

3.3 Demo 示例

在 SDK 中提供了 Demo，此 Demo 是用播放器 SDK 开发了一个完整的视频播放器，用户可以参考 Demo 进行播放器的开发。

下面给出了部分重要的 Demo 中调用 SDK 的代码。

一、应用启动的时候，给播放器类执行初始化工作

```
AliVcMediaPlayer.init(getApplicationContext(), businessId, new AccessKeyCallback() {
                    public AccessKey getAccessToken() {
                        return new AccessKey(accessKeyId, accessKeySecret);
                    }
                });
```

二、创建播放器，准备视频播放：

```
//1. 创建播放器
mPlayer = new AliVcMediaPlayer(context,surfaceView);

//2. 注册事件通知
mPlayer.setPreparedListener(new VideoPreparedListener());
mPlayer.setErrorListener(new VideoErrorListener());
mPlayer.setInfoListener(new VideoInfolistener());
mPlayer.setSeekCompleteListener(new VideoSeekCompletelistener());
mPlayer.setCompletedListener(new VideoCompletelistener());
mPlayer.setVideoSizeChangeListener(new VideoSizeChangelistener());
mPlayer.setBufferingUpdateListener(new VideoBufferUpdatelistener());

//3. 设置缺省编码类型：0表示硬解；1表示软解；
mPlayer.setDefaultDecoder(0);

//4. 如果从历史点开始播放
mPlayer.seekTo(position);

//5. 准备开始播放
mPlayer.prepareAndPlay(msURI.toString());
```

三、准备完成事件通知中：

```
private class VideoPrepareListener implements
AliVcMediaPlayer.MediaPlayerPreparedListener{
    @Override
    public void onPrepared() {
       //更新视频总进度
    }
}
```

四、错误事件通知中：

```
private class VideoErrorListener implements
AliVcMediaPlayer.MediaPlayerErrorListener {
```

```
    public void onError(int what, int extra) {
        switch(what)
        {
        case MediaPlayer.ALIVC_ERR_ILLEGALSTATUS:
            // 非法状态!
            break;
        case MediaPlayer.ALIVC_ERR_NO_NETWORK:
            //report_error("视频资源或网络不可用！", true);
            break;
        case MediaPlayer.ALIVC_ERR_INVALID_INPUTFILE:
            //视频资源或网络不可用!
            break;
        case MediaPlayer.ALIVC_ERR_NO_SUPPORT_CODEC:
            //无支持的解码器!
            break;
         case MediaPlayer.ALIVC_ERR_FUNCTION_DENY:
            //无此操作权限!
            break;
         case MediaPlayer.ALIVC_ERR_UNKNOWN:
            //未知错误!
            break;
         case MediaPlayer.ALIVC_ERR_NOTAUTH:
            //未鉴权!
            break;
         case MediaPlayer.ALIVC_ERR_READD:
            //资源访问失败!
            break;
        default:
            //播放器错误!
            break;
        }
    }
}
```

五、播放信息事件通知中：

```
private class VideoInfolistener implements
AliVcMediaPlayer.MediaPlayerInfoListener {
    public int onInfo(int what, int extra){
        switch (what)
        {
        case MediaPlayer.MEDIA_INFO_UNKNOW:
            // 未知
             break;
        case MediaPlayer.MEDIA_INFO_BUFFERING_START:
           // 开始缓冲
             break;
        case MediaPlayer.MEDIA_INFO_BUFFERING_END:
            // 结束缓冲
           break;
        case MediaPlayer.MEDIA_INFO_VIDEO_RENDERING_START:
             // 首帧显示时间
           break;
         }
        return 0;
    }
}
```

4.接口说明

SDK中提供了两个类AliVcMediaPlayer和AliVcMediaPlayerFactory，其中AliVcMediaPlayer是播放器SDK使用类，AliVcMediaPlayerFactory用来创建播放器AliVcMediaPlayer。同时提供了多个事件通知接口，用来监听播放器的各种状态。

类名	功能
AliVcMediaPlayerFactory	**创建媒体播放器接口**
MediaPlayer	**媒体播放器功能**接口**类**

AliVcMediaPlayer	**媒体播放器功能实现类**
MediaPlayerPrepareListener	**视频播放准备完成监听接口**
MediaPlayerCompletedListener	**视频播放完成监听接口**
MediaPlayerInfoListener	**视频播放信息监听接口**
MediaPlayerSeekCompleteListener	**视频跳转完成监听接口**
MediaPlayerBufferingUpdateListener	**视频缓冲监听接口**
MediaPlayerVideoSizeChangeListener	**视频大小改变监听接口**
MediaPlayerErrorListener	**视频播放错误监听接口**

4.1 AliVcMediaPlayerFactory

类名：AliVcMediaPlayerFactory

功能：创建媒体播放器接口类 MediaPlayer

成员：

成员	功能
createPlayer	**创建媒体播放器**MediaPlayer

详细说明：

```
MediaPlayer createPlayer(Context context,int decoder_type,
String path)
```

createPlayer用来创建播放器，返回MediaPlayer类。

参数:

- path：播放器文件路径，本地或者网络地址。

返回值：返回空为错误，正确则为有效的MediaPlayer值。

4.2 MediaPlayer

类名：MediaPlayer

功能：媒体播放器接口类 MediaPlayer，提供播放控制

成员：

成员	功能
init	初始化播放器
prepareToPlay	准备视频播放
play	开始播放视频
pause	暂停视频播放
stop	停止视频播放
reset	释放播放器
seekTo	跳转到指定位置
isPlaying	是否正在播放
setVolumn	调节音量
getVideoWidth	获取视频宽度
getVideoHeight	获取视频高度
getDuration	获取视频长度
getCurrentPosition	获取当前视频播放位置
getUserPriority	获取播放器权限
setPreparedListener	注册视频准备完成通知
setCompletedListener	注册播放完成通知
setInfoListener	注册播放信息通知
setErrorListener	注册播放错误通知
setSeekCompleteListener	注册跳转完成通知
setBufferingUpdateListener	注册缓冲更新通知
setVideoSizeChangeListener	注册视频大小改变通知
getErrorCode	获取错误码
setSurfaceChanged	设置 surface 发生改变
enalbeNativeLog	打开底层日志，在开发阶段使用
disableNativeLog	关闭底层日志，在 release 阶段使用
setVideoSurface	设置视频显示的 surface
releaseVideoSurface	释放视频显示的 surface
setTimeout	设置 IO 超时时间，单位毫秒
setMaxBufferDuration	设置最大的缓冲时长，直播中有效
setMediaType	设置视频源类型
getPropertyDouble	获取性能参数

getPropertyLong	**获取长整型性能参数**
getCurrNativeLog	**获取 Natvie 的日志**
getAllDebugInfo	**获取全部 debug 信息**
destroy	**回收播放器**

下面详细介绍一下各个成员函数的具体使用：

1. 初始化播放器

```
/** init video player */
public static void init(Context context, String businessId, AccessKeyCallback callback);
```

参数：

context: Android 上下文；

callback：AccessKey 的回调函数；

businesssId：业务 ID

2. 准备并开始视频播放

```
/** Start to prepare and play the video. */
public void prepareAndPlay(String url);
```

参数

备注：准备并开始进行视频播放。

3. 暂停视频播放

```
public void pause();
```

4. 停止视频播放

```
public void stop();
```

5. 重置视频播放

```
public void reset();
```

备注：当 error 发生时需调用 reset 重置播放器状态

6. 视频跳转：跳转到指定位置前的第一个关键帧的位置

```
public void seekTo(int msc);
```

备注：msc 参数为毫秒，范围为 0-视频总长度。

7. 视频是否在播放

```
public boolean isPlaying();
```

备注：返回 true 代表正在播放，否则没有在播放

8. 设置音量

```
public void setVolume(int vol);
```

参数：

vol 为音量大小，范围为 0-100，100 为最大，0 为最小。

9. 获取视频宽度

```
public int getVideoWidth();
```

备注：返回视频宽度，0 为返回失败

10. 获取视频高度

```
public int getVideoHeight();
```

备注：返回视频高度，0 为返回失败

11. 获取视频总长度

```
public int getDuration();
```

备注：返回视频总长度，单位为毫秒

12. 获取视频当前位置

```
public int getCurrentPosition();
```

备注：返回视频当前播放位置，单位为毫秒

13. 获取错误码

```
public int getErrorCode();
```

备注：当播放器出错时，调用该函数获取播放器错误码。

14. 通知 surface 改变

```
public void setSurfaceChanged();
```

备注：在播放暂停或卡顿时，这个时候旋转手机屏幕，会发生渲染错位。为了解决这一问题，请在surfaceChanged发生时，调用此方法。如果播放界面关闭了自动旋转功能，可以不许调用此方法；

15. 打开底层日志

```
public void enableNativeLog();
```

备注：务必在开发阶段才调用此方法。打开底层日志，意味着底层的日志首先会通过adb logcat的形式输出，另外在应用层，还可以通过getCurrNatvieLog方法获取底层日志。

16. 关闭底层日志

```
public void disableNativeLog();
```

17. 获取 Native 日志

```
public List<VideoNativeLog> getCurrNatvieLog();
```

备注：仅仅在 enableNativeLog 有效。此方法返回底层日志列表，每条日志(VideoNativeLog) 包含如下几个字段：

Tag：日志的 tag，不唯一

Content：日志的内容

Time：日志的时间

Level：日志的级别（0 表示 ANDROID_LOG_UNKNOWN ； 1 表示 ANDROID_LOG_DEFAULT ； 2 表示 ANDROID_LOG_VERBOSE ；3 表示 ANDROID_LOG_DEBUG；4 表示 ANDROID_LOG_INFO；5 表示 ANDROID_LOG_WARN ； 6 表示 ANDROID_LOG_ERROR ； 7 表示 ANDROID_LOG_FATAL ； 8 表示 ANDROID_LOG_SILENT）

18. 设置视频 Surface

```
public void setVideoSurface(Surface surface);
```

参数：

surface: 视频播放 View 的 surface

备注：使用场景是之前的 surface 已经销毁，但是还要继续播放；或者想在一个新的 surface 上显示视频 。特别注意，在初始化播放器的时候，已经传入了 surface，所以在释放以前的 surface 之前，是不允许再次设置新的 surface 的。也就是说请先 releaseVideoSurface 再 setVideoSurface。

19. 释放视频 Surface

```
public void releaseVideoSurface();
```

备注：使用场景是当前的 surface 被销毁；或者想在一个新 surface 上显示视频，需要提前释放当前的 surface。如果使用播放器构造函数或者 setVideoSurface 设置了 surface，那么就可以通过 releaseVideoSurface 释放当前的 surface，但是一旦释放之后，就不能再次调用，否则就会出现黑屏。

20. 设置 IO 超时时间

```
public void setTimeout(int timeout);
```

参数：

timeout：超时时长，单位**毫秒**

备注：当播放器超过设定时间没有下载到任何数据，会发送 ALIVC_ERR_LOADING_TIMEOUT 错误事件。系统默认 timeout 时间为 15 秒。

21. 设置直播最大缓冲时长

```
public void setMaxBufferDuration(int duration);
```

参数：

duration: 缓冲时长，单位**毫秒**

备注：该函数仅对直播场景有效，主要用于缩短主播与观众之间的时间延迟。当缓冲区中的视频时长超过设置的 duration 时，播放器会自动丢弃部分音视频数据，以减少延迟。系统默认 rtmp、http flv 直播时 duration 为 8 秒，HLS 直播时 duration 为 40 秒。建议 rtmp、http flv 直播时，duration 的值超过 GOP 时长（即两个关键帧之间的时间长度）的 2 倍；HLS 直播时，duration 的值超过 m3u8 文件中所有 ts 分片的总时长。这样可以避免出现经常性的视频丢帧。

22. 设置播放类型

public void setMediaType(MediaType type);

参数：

type: 媒体类型，MediaType.Live 直播；MediaType.Vod 点播

备注：建议在可以清晰分辨视频类型的情况下尽量调用该函数。如果不调用，则播放器会自动根据视频的 duration 来判断媒体类型。Duration 为 0 且格式为 HLS、rtmp、http flv 的为直播类视频，其他为点播类视频。

23. 获取 Double 型性能参数

public double getPropertyDouble(int key,double defaultValue);

参数：

defaultValue：缺省数据

key：关键字常量

MediaPlayer.PROP_DOUBLE_VIDEO_DECODE_FRAMES_PER_SECOND

视频解码帧率

MediaPlayer.PROP_DOUBLE_VIDEO_OUTPUT_FRAMES_PER_SECOND

视频渲染帧率

MediaPlayer.FFP_PROP_DOUBLE_OPEN_FORMAT_TIME

调用 avformat_open_input 的时刻，单位毫秒

MediaPlayer.FFP_PROP_DOUBLE_FIND_STREAM_TIME

调用 avformat_find_stream_info 的时刻，单位毫秒

MediaPlayer.FFP_PROP_DOUBLE_OPEN_STREAM_TIME

MediaPlayer.FFP_PROP_DOUBLE_1st_VFRAME_SHOW_TIME

首个视频帧渲染时刻，单位毫秒

MediaPlayer.FFP_PROP_DOUBLE_1st_AFRAME_SHOW_TIME

首个音频帧渲染时刻，单位毫秒

MediaPlyer.FFP_PROP_DOUBLE_1st_VPKT_GET_TIME

首个视频帧下载时刻，单位毫秒

MediaPLyer.FFP_PROP_DOUBLE_1st_APKT_GET_TIME

首个音频帧下载时刻，单位毫秒

MediaPlayer.FFP_PROP_DOUBLE_1st_VDECODE_TIME

首个视频帧解码时刻，单位毫秒

MediaPlayer.FFP_PROP_DOUBLE_1st_ADECODE_TIME

首个音频帧解码时刻，单位毫秒

MediaPLyaer.FFP_PROP_DOUBLE_DECODE_TYPE

视频解码器类型

MediaPlayer.FFP_PROP_DOUBLE_LIVE_DISCARD_DURATION

直播视频丢弃音视频帧时长，单位毫秒

MediaPlayer.FFP_PROP_DOUBLE_LIVE_DISCARD_CNT

直播视频丢弃音视频帧数量总和

MediaPlayer.FFP_PROP_DOUBLE_DISCARD_VFRAME_CNT

直播视频丢弃视频帧数量总和

MediaPlayer.FFP_PROP_DOUBLE_RTMP_OPEN_DURATION

RTMP 流打开时长，单位毫秒

MediaPlayer.FFP_PROP_DOUBLE_RTMP_OPEN_RTYCNT

RTMP 重连次数

MediaPlayer.FFP_PROP_DOUBLE_RTMP_NEGOTIATION_DURATION

RTMP 连接握手时长，单位毫秒

MediaPlayer.FFP_PROP_DOUBLE_HTTP_OPEN_DURATION

HTTP 流打开时长，单位毫秒

MediaPlayer.FFP_PROP_DOUBLE_HTTP_OPEN_RTYCNT

HTTP 重连次数

MediaPlayer.FFP_PROP_DOUBLE_HTTP_REDIRECT_CNT

HTTP 重定向次数

MediaPlayer.FFP_PROP_DOUBLE_TCP_CONNECT_TIME

TCP 连接时长，单位毫秒

MediaPlayer.FFP_PROP_DOUBLE_TCP_DNS_TIME

TCP 连接 DNS 时长，单位毫秒

24. 获取 Long 型性能参数

public long getPropertyLong(int key,long defaultValue);

参数：

defaultValut：缺省数据

key：关键字常量

MediaPlayer.FFP_PROP_INT64_VIDEO_CACHED_DURATION

视频缓冲时长，单位毫秒

MediaPlayer.FFP_PROP_INT64_AUDIO_CACHED_DURATION

音频缓冲时长，单位毫秒

MediaPlayer.FFP_PROP_INT64_VIDEO_CACHED_BYTES

视频缓冲大小，单位 byte

MediaPlayer.FFP_PROP_INT64_AUDIO_CACHED_BYTES

音频缓冲大小，单位毫秒

MediaPlayer.FFP_PROP_INT64_VIDEO_CACHED_PACKETS
视频缓冲帧数
MediaPlayer.FFP_PROP_INT64_AUDIO_CACHED_PACKETS
音频缓冲帧数
MediaPlayer.FFP_PROP_INT64_SELECTED_VIDEO_STREAM
视频流 index
MediaPlayer.FFP_PROP_INT64_SELECTED_AUDIO_STREAM
音频流 index

25. 获取实时性能参数

```
public Map<String, String> getAllDebugInfo();
```

备注：返回的性能参数包含：

"dec-fps"：视频解码 fps
"out-fps"：视频渲染 fps
"select-v"：视频流 index
"select_a"：音频流 index
"v-dec"：视频解码器名称
"a-dec"：音频解码器名称
"vcache-dur"：视频缓冲时长，单位秒
"acache-dur"：音频缓冲时长，单位秒
"vcache-bytes"：视频缓冲大小，单位 byte
"acache-bytes"：音频缓冲大小，单位 byte
"vcache-pkts"：视频缓冲帧数
"acache-pkts"：音频缓冲帧数

26. 回收播放器

```
public void destroy();
```

备注：当整个播放器退出时调用，回收播放器。

4.3 AliVcMediaPlayer

MediaPlayer 的实现类。

4.4 MediaPlayerPrepareListener

当调用prepareAsync后,视频准备完成后会发送准备完成事件，用户需要注册该事件，以便获取到该事件通知，在准备完成后调用start接口进行视频播放。

```
public interface MediaPlayerPrepareListener {
    void onPrepared();
}
```

4.5 MediaPlayerCompletedListener

当视频播放完成后，会发出该事件通知消息，用户需要注册该事件，在播放完成后完成相关清理工作。

```
public interface MediaPlayerCompletedListener {
    void onCompleted();
}
```

4.6 MediaPlayerInfoListener

当视频开始播放，用户需要知道视频的相关信息，可以注册该事件。

```
public interface MediaPlayerInfoListener{
    void onInfo(int what, int extra);
}
```

参数：

1. what：获取到的播放信息或警告的类型

   播放的相关信息有：

   - MEDIA_INFO_UNKNOW：未知信息
   - MEDIA_INFO_BUFFERING_START：当开始缓冲时，收到该信

息

- MEDIA_INFO_BUFFERING_END：缓冲结束时收到该信息

2. extra：对播放信息的额外表述。

4.7 MediaPlayerErrorListener

当视频播放出现错误后,会发出该事件通知消息，用户需要注册该事件通知，以便在出现错误后给出相关错误提示。

```
public interface MediaPlayerErrorListener {
    void onError(int what, int extra);
}
```

参数：

1. what：错误信息的类型

错误信息有：

- ALIVC_ERR_UNKNOW：未知错误
- ALIVC_ERR_LOADING_TIMEOUT：缓冲超时
- ALIVC_ERR_NO_INPUTFILE：未设置视频源
- ALIVC_ERR_NO_VIEW：无效的surface
- ALIVC_ERR_INVALID_INVALID_INPUTFILE: 无效的视频源
- ALIVC_ERR_NO_SUPPORT_CODEC：无支持的解码器
- ALIVC_ERR_FUNCTION_DENIED：操作无权限
- ALIVC_ERR_NO_NETWORK: 网络不可用
- ALIVC_ERR_ILLEGALSTATUS：非法状态
- ALIVC_ERR_NOTAUTH：未鉴权

- ALIVC_ERR_READD：视频源访问失败

2. extra：错误信息的额外描述

- ALIVC_ERR_EXRA_DEFAULT：缺省值
- ALIVC_ERR_EXTRA_PREPARE_FAILED：prepare失败
- ALIVC_ERR_EXTRA_OPEN_FAILED: open stream 失败

4.8 MediaPlayerSeekCompleteListener

当视频进行seek跳转后，会发出该事件通知消息，用户注册该事件通知后，能收到跳转完成通知。

```
public interface MediaPlayerSeekCompleteListener {
    void onSeekCompleted();
}
```

4.9 MediaPlayerBufferingUpdateListener

当网络下载速度较慢来不及播放时，会发送下载缓冲进度通知。

```
public interface MediaPlayerBufferingUpdateListener {
    void onBufferingUpdateListener(int percent);
}
```

参数 percent，代表的意思是目前视频缓冲的进度，范围为 0-100，100 代表缓冲完成，0 代表缓冲开始。

4.10 MediaPlayerVideoSizeChangeListener

当视频播放时视频大小改变后，会发出该事件通知。

```
public interface MediaPlayerVideoSizeChangeListener {
```

```
    void onVideoSizeChange(int width, int height);
}
```

参数：

1. width：视频改变之后的宽度
2. height：视频改变之后的高度

5. 注意事项

无

6. 版权声明