视频边缘智能服务

应用开发指南

应用开发指南

Android应用端接入

工程配置

工程配置

您可以使用视频边缘智能服务提供的应用端Android SDK，实现视频播放的功能。

Android SDK Demo

Android SDK提供Demo，供您参考使用。单击下载Android SDK Demo。下载本Demo将默认您同意本软件许可协议。

配置

在Android工程根目录下的build.gradle 基础配置文件中，加入阿里云仓库地址，进行仓库配置。

```
allprojects {
repositories {
jcenter()
google()
// 阿里云仓库地址
maven {
url "http://maven.aliyun.com/nexus/content/repositories/releases/"
}
maven {
url "http://maven.aliyun.com/nexus/content/repositories/snapshots"
}
}
}
```

在模块的 build.gradle 中，添加SDK的依赖，引入 SDK：linkvisual-media

```
implementation('com.aliyun.iotx:linkvisual-media:1.1.0')
```

混淆配置(proguard-rules.pro)

```
# keep linkvisual
-keep class com.aliyun.iotx.linkvisual.media.** { *; }

# keep netty
-keepattributes Signature,InnerClasses
-keepclasseswithmembers class io.netty.** {
*;
}
-dontwarn io.netty.**
-dontwarn sun.**
```

视频播放器

视频播放器

播放器按功能分为三种:

RTMP直播播放器

- 用于播放摄像头直播源，具有时延低的特点.

RTMP点播播放器

- 用于设备录像回放的播放，可调整播放进度.

HLS播放器

- 用于云端录像回放的播放，支持AES-128加密的ts文件的播放.

FFmpeg使用版本为n4.0.1.
音频编码支持AAC_LC和G711a.
视频编码支持H264.
Android SDK封装ExoPlayer作为HLS的播放器，版本为2.8.3.

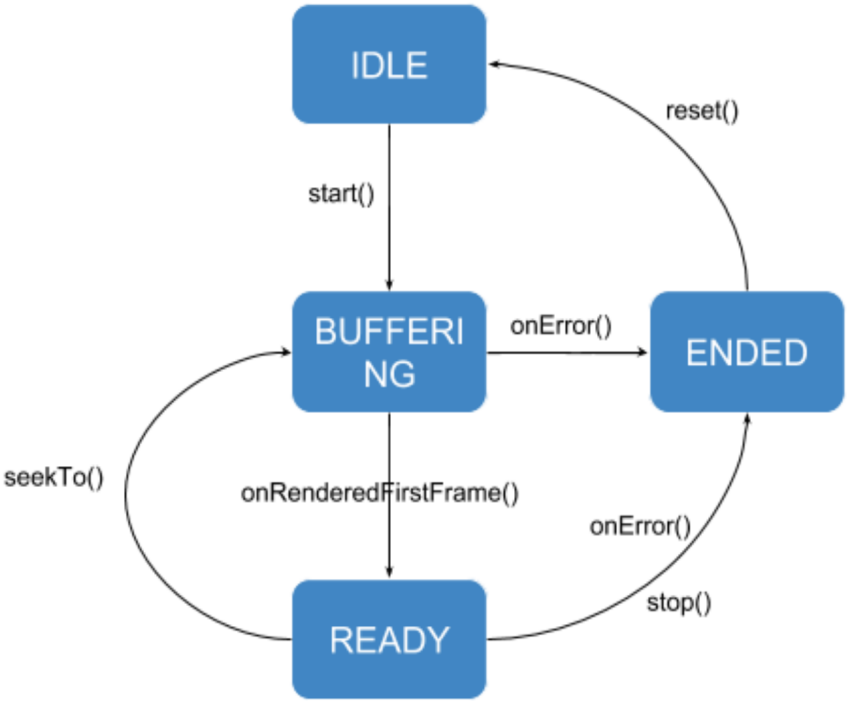
功能列表

功能	直播播放器	设备录像播放器	HLS播放器

视频播放	√	√	√
音频播放	√	√	√
暂停/恢复	X	√	√
跳至指定位置播放	X	√	√
总时长	X	√	√
当前播放进度	X	√	√
播放器状态变更通知	√	√	√
设置播放音量	√	√	√
变速播放	X	X	√
循环播放	X	X	√
画面缩放模式设置	√	√	√
播放器截图	√	√	X
边播边录	√	√	X

播放器状态

通过设置播放器状态监听器, 可接收到状态变更事件, 用于相关UI元素的变更.



- **IDLE**: 播放器没有任何内容播放时的状态.
- **BUFFERING**: 播放器正在缓冲, 当前的位置还不可以播放.状态变更事件如: 开始播放的缓冲、seek后重

新缓冲.
- **READY**: 播放器已经有内容在播放.状态变更事件如: 首帧数据已经渲染.
- **ENDED**: 播放器已结束播放.状态变更事件如: 播放中发生错误、主动停止播放、内容播放完毕.

错误列表

错误枚举	描述
PlayerException.SOURCE_ERROR	数据源错误
PlayerException.RENDER_ERROR	渲染错误
PlayerException.UNEXPECTED_ERROR	不符合预期的错误

RTMP播放器

RTMP播放器

提供两种RTMP播放器，LivePlayer用于rtmp直播源的播放，VodPlayer用于rtmp点播源的播放.

RTMP直播播放器(LivePlayer)

直播播放器使用说明:

```
// 创建播放器实例
LivePlayer player = new LivePlayer();

// 设置surfaceview, 必须为GLSurfaceView
// 注意:GLSurfaceView必须在Activity的onResume和onPause回调方法中调用GLSurfaceView的onResume和onPause方法
player.setSurfaceView(GLSurfaceView);
// 设置必要的状态监听
player.setOnPlayerStateChangedListener(new OnPlayerStateChangedListener() {
@Override
public void onPlayerStateChange(int playerState) {
Log.d(TAG, "play state= " + playerState);
switch (playerState) {
case Player.STATE_BUFFERING:
break;
case Player.STATE_IDLE:
break;
case Player.STATE_READY:
break;
case Player.STATE_ENDED:
```

```
break;
default:
break;
}
}
});
// 设置错误监听
player.setOnErrorListener(new OnErrorListener() {
@Override
public void onError(PlayerException exception) {
makeToast("errorcode: " + exception.getCode() + "\n" + exception.getMessage());
}
});

// 设置rtmp地址
player.setDataSource("rtmp://live.hkstv.hk.lxdns.com/live/hks2");
// 设置数据源就绪监听器
player.setOnPreparedListener(new OnPreparedListener() {
@Override
public void onPrepared() {
// 数据源就绪后开始播放
player.start();
}
});
player.prepare();

...

// 停止播放
player.stop();

...

// 释放播放器资源
player.release();
```

RTMP点播播放器(VodPlayer)

点播播放器使用说明:

```
// 创建播放器实例
VodPlayer player = new VodPlayer();

// 设置surfaceview, 必须为GLSurfaceView
// 注意:GLSurfaceView必须在Activity的onResume和onPause回调方法中调用GLSurfaceView的onResume和onPause方法
player.setSurfaceView(GLSurfaceView);
// 设置必要的状态监听
player.setOnPlayerStateChangedListener(new OnPlayerStateChangedListener() {
@Override
public void onPlayerStateChange(int playerState) {
Log.d(TAG, "play state= " + playerState);
switch (playerState) {
case Player.STATE_BUFFERING:
break;
```

```
case Player.STATE_IDLE:
break;
case Player.STATE_READY:
break;
case Player.STATE_ENDED:
break;
default:
break;
}
}
});
// 设置错误监听
player.setOnErrorListener(new OnErrorListener() {
@Override
public void onError(PlayerException exception) {
makeToast("errorcode: " + exception.getCode() + "\n" + exception.getMessage());
}
});

// 设置支持点播的rtmp地址
player.setDataSource("rtmp://xxxx");
// 设置数据源就绪监听器
player.setOnPreparedListener(new OnPreparedListener() {
@Override
public void onPrepared() {
// 数据源就绪后开始播放
player.start();
}
});
player.prepare();

...

// 暂停播放
player.pause();

...

// 恢复播放
player.resume();

...

// 停止播放
player.stop();

...

// 释放播放器资源
player.release();
```

接口说明:

LivePlayer

- 构造方法

```
/**
* 支持{@link #setDataSource(String)}
* {@link #setDataSource(String, boolean, byte[], byte[])}的源
*/
LivePlayer();
```

- 设置非加密播放源

```
/**
* 设置播放源
* @param url RTMP地址
*/
void setDataSource(String url) throws IllegalArgumentException;
```

- 设置加密播放源

```
/**
* 设置加密播放源(请确保源是做过AES加密)
* @param url rtmp源地址
* @param isEncrypted 是否是加密源
* @param decryptIv 解密向量，16 byte array
* @param decryptKey 解密密钥, 16 byte array
* @throws IllegalArgumentException
*/
void setDataSource(String url, boolean isEncrypted, byte[] decryptIv, byte[] decryptKey) throws
IllegalArgumentException;
```

- 校验和准备数据

```
/**
* 校验和准备数据
*/
void prepare();
```

- 开始播放视频

```
/**
* 开始播放视频
*/
void start();
```

- 停止播放

```
/**
* 停止播放
```

```
*/
void stop();
```

- 重置播放器

```
    /**
* 重置播放器
*/
void reset();
```

- 释放播放器资源

```
    /**
* 释放播放器资源
*/
void release();
```

- 截图

```
    /**
* 当前视频画面截图
*
* @return 如果当前无画面则返回 null
*/
Bitmap snapShot();
```

- 开始录屏

```
    /**
* 开始录制当前播放内容，生成MPEG-4格式转存到指定的文件中
* 文件名后缀必须为.mp4
* 须在{@link PlayerState#STATE_READY}时调用有效
*
* @param contentFile
* @return 操作成功与否
*/
boolean startRecordingContent(File contentFile) throws IOException;
```

- 结束录屏

```
    /**
* 停止记录播放内容
* @return 操作成功与否
*/
boolean stopRecordingContent();
```

- 设置播放器音量

```
    /**
* 设置播放器音量
* @param audioVolume 范围 0-1, 0为静音
*/
void setVolume(float audioVolume);
```

- 设置音频流通道类型

```
    /**
* 设置音频流通道类型,see {@link android.media.AudioManager}
* 如果音频正在播放，则会因为重新创建AudioTrack导致有短暂停顿.
* @param streamType
*/
void setAudioStreamType(int streamType);
```

- 设置画面缩放模式

```
    /**
* 设置视频画面缩放模式, 默认为{@link
android.media.MediaCodec#VIDEO_SCALING_MODE_SCALE_TO_FIT_WITH_CROPPING}
* @param videoScalingMode 参考:
* {@link android.media.MediaCodec#VIDEO_SCALING_MODE_SCALE_TO_FIT}
* {@link android.media.MediaCodec#VIDEO_SCALING_MODE_SCALE_TO_FIT_WITH_CROPPING}
*/
void setVideoScalingMode(int videoScalingMode);
```

- 设置surfaceview

```
    /**
* 设置SurfaceView, 必须为GLSurfaceView
* @param surfaceview
*/
void setSurfaceView(SurfaceView surfaceview);
```

- 清除surfaceview

```
    /**
* 清除surfaceview
*/
void clearSurfaceView();
```

- 设置数据源准备就绪事件监听器

```
    /**
* 设置数据源准备就绪事件监听器
*
```

```
* @param listener
*/
void setOnPreparedListener(OnPreparedListener listener);
```

- 设置播放器错误事件监听器

```
    /**
* 设置播放器错误事件监听器，错误类型参考:
* {@link PlayerException.SOURCE_ERROR}
* {@link PlayerException.RENDER_ERROR}
* {@link PlayerException.UNEXPECTED_ERROR}
* @param listener
*/
void setOnErrorListener(OnErrorListener listener);
```

- 设置播放状态变更事件监听器

```
    /**
* 设置播放状态变更事件监听器
* @param listener
*/
void setOnPlayerStateChangedListener(OnPlayerStateChangedListener listener);
```

- 设置首帧被渲染事件监听器

```
    /**
* 设置首帧被渲染事件监听器
* @param listener
*/
void setOnRenderedFirstFrameListener(OnRenderedFirstFrameListener listener);
```

- 获取音量

```
    /**
* 获取音量
* @return 范围0-1
*/
float getVolume();
```

- 获取播放器状态

```
    /**
* 获取播放状态
* @return 状态枚举，参考:
* {@link PlayerState#STATE_IDLE} 播放器初始状态
* {@link PlayerState#STATE_BUFFERING} 缓冲中状态
* {@link PlayerState#STATE_READY} 缓冲结束开始播放状态
* {@link PlayerState#STATE_ENDED} 播放完成状态
```

```
*/
int getPlayState();
```

- 获取播放器当前流的连接类型

```
    /**
* 获取播放器当前流的连接类型
* 播放器状态为{@link PlayerState#STATE_READY}时调用有效
*
* @return {@link StreamConnectType}
*/
StreamConnectType getStreamConnectType();
```

- 获取播放器当前的帧率/码率等信息

```
    /**
* 获取播放器当前的帧率/码率等信息
* 播放器状态为{@link PlayerState#STATE_READY}时调用有效
*
* @return 包含帧率/码率等信息的json string
*/
PlayInfo getCurrentPlayInfo();
```

VodPlayer

- 构造方法

```
    /**
* 对于一般rtmp数据源，使用该构造方法初始化
* 支持{@link #setDataSource(String)}
* {@link #setDataSource(String, boolean, byte[], byte[])}的源
*/
VodPlayer();
```

- 设置非加密播放源

```
    /**
* 设置播放源
* @param url RTMP地址
*/
void setDataSource(String url) throws IllegalArgumentException;
```

- 设置加密播放源

```
     /**
* 设置加密播放源(请确保源是做过AES加密)
* @param url rtmp源地址
* @param isEncrypted 是否是加密源
```

```
* @param decryptIv 解密向量，16 byte array
* @param decryptKey 解密密钥, 16 byte array
* @throws IllegalArgumentException
*/
void setDataSource(String url, boolean isEncrypted, byte[] decryptIv, byte[] decryptKey) throws
IllegalArgumentException;
```

- 校验和准备数据

```
    /**
* 校验和准备数据
*/
void prepare();
```

- 开始或恢复播放视频

```
    /**
* 开始播放或恢复播放视频
*/
void start();
```

- 暂停播放

```
    /**
* 暂停播放, 调用start()恢复播放
*/
void pause();
```

- seek到指定位置

```
    /**
* seek到指定位置
* @param position 毫秒
*/
void seekTo(long positionInMs);
```

- 停止播放

```
    /**
* 停止播放
*/
void stop();
```

- 重置播放器

```
    /**
```

```
* 重置播放器
*/
void reset();
```

- 释放播放器资源

```
    /**
* 释放播放器资源
*/
void release();
```

- 截图

```
    /**
* 当前视频画面截图
*
* @return 如果当前无画面则返回 null
*/
Bitmap snapShot();
```

- 开始录屏

```
    /**
* 开始录制当前播放内容，生成MPEG-4格式转存到指定的文件中
* 文件名后缀必须为.mp4
* 须在{@link PlayerState#STATE_READY}时调用有效
*
* @param contentFile
* @return 操作成功与否
*/
boolean startRecordingContent(File contentFile) throws IOException;
```

- 结束录屏

```
    /**
* 停止记录播放内容
* @return 操作成功与否
*/
boolean stopRecordingContent();
```

- 设置播放器音量

```
    /**
* 设置播放器音量
* @param audioVolume 范围 0-1, 0为静音
*/
void setVolume(float audioVolume);
```

- 设置音频流通道类型

```
    /**
* 设置音频流通道类型,see {@link android.media.AudioManager}
* 如果音频正在播放，则会因为重新创建AudioTrack导致有短暂停顿.
* @param streamType
*/
void setAudioStreamType(int streamType);
```

- 设置画面缩放模式

```
    /**
* 设置视频画面缩放模式, 默认为{@link
android.media.MediaCodec#VIDEO_SCALING_MODE_SCALE_TO_FIT_WITH_CROPPING}
* @param videoScalingMode 参考:
* {@link android.media.MediaCodec#VIDEO_SCALING_MODE_SCALE_TO_FIT}
* {@link android.media.MediaCodec#VIDEO_SCALING_MODE_SCALE_TO_FIT_WITH_CROPPING}
*/
void setVideoScalingMode(int videoScalingMode);
```

- 设置surfaceview

```
    /**
* 设置SurfaceView, 必须为GLSurfaceView
* @param surfaceview
*/
void setSurfaceView(SurfaceView surfaceview);
```

- 清除surfaceview

```
    /**
* 清除surfaceview
*/
void clearSurfaceView();
```

- 设置数据源准备就绪事件监听器

```
    /**
* 设置数据源准备就绪事件监听器
*
* @param listener
*/
void setOnPreparedListener(OnPreparedListener listener);
```

- 设置播放器错误事件监听器

```
    /**
```

```
* 设置播放器错误事件监听器，错误类型参考:
* {@link PlayerException.SOURCE_ERROR}
* {@link PlayerException.RENDER_ERROR}
* {@link PlayerException.UNEXPECTED_ERROR}
* @param listener
*/
void setOnErrorListener(OnErrorListener listener);
```

- 设置播放状态变更事件监听器

```
    /**
* 设置播放状态变更事件监听器
* @param listener
*/
void setOnPlayerStateChangedListener(OnPlayerStateChangedListener listener);
```

- 设置首帧被渲染事件监听器

```
    /**
* 设置首帧被渲染事件监听器
* @param listener
*/
void setOnRenderedFirstFrameListener(OnRenderedFirstFrameListener listener);
```

- 获取当前播放进度

```
    /**
* 获取当前播放进度
* 播放器状态为{@link PlayerState#STATE_READY}时调用有效
* @return 单位MS
*/
long getCurrentPosition();
```

- 获取视频总时长

```
    /**
* 获取视频总时长
* 播放器状态为{@link #STATE_READY}时调用有效
* @return 单位MS
*/
long getDuration();
```

- 获取音量

```
    /**
* 获取音量
* @return 范围0-1
*/
```

```
float getVolume();
```

- 获取播放器状态

```
    /**
* 获取播放状态
* @return 状态枚举，参考:
* {@link PlayerState#STATE_IDLE} 播放器初始状态
* {@link PlayerState#STATE_BUFFERING} 缓冲中状态
* {@link PlayerState#STATE_READY} 缓冲结束开始播放状态
* {@link PlayerState#STATE_ENDED} 播放完成状态
*/
int getPlayState();
```

- 获取播放器当前流的连接类型

```
    /**
* 获取播放器当前流的连接类型
* 播放器状态为{@link PlayerState#STATE_READY}时调用有效
*
* @return {@link StreamConnectType}
*/
StreamConnectType getStreamConnectType();
```

- 获取播放器当前的帧率/码率等信息

```
    /**
* 获取播放器当前的帧率/码率等信息
* 播放器状态为{@link PlayerState#STATE_READY}时调用有效
*
* @return 包含帧率/码率等信息的json string
*/
PlayInfo getCurrentPlayInfo();
```

HLS播放器

HLS播放器

HLS播放器使用:

```
ExoHlsPlayer player = new ExoHlsPlayer(getApplicationContext());
```

```
// 使用Exo的SimpleExoPlayerView来作为播放器的UI组件, 也可以自己实现UI组件
simpleExoPlayerView.setPlayer(player.getExoPlayer());
simpleExoPlayerView.requestFocus();
// 设置错误监听
player.setOnErrorListener(new OnErrorListener() {
@Override
public void onError(PlayerException exception) {
makeToast("errorcode: " + exception.getCode() + "\n" + exception.getMessage());
}
});
// 设置状态监听
player.setOnPlayerStateChangedListener(new OnPlayerStateChangedListener() {
@Override
public void onPlayerStateChange(int playerState) {
switch (playerState) {
case Player.STATE_BUFFERING:
break;
case Player.STATE_IDLE:
break;
case Player.STATE_READY:
break;
case Player.STATE_ENDED:
break;
default:
break;
}
}
});

// 设置m3u8地址
player.setDataSource("http://devimages.apple.com.edgekey.net/streaming/examples/bipbop_4x3/gear3/prog_index
.m3u8");
// 设置数据源就绪监听器
player.setOnPreparedListener(new OnPreparedListener() {
@Override
public void onPrepared() {
// 数据源就绪后开始播放
player.start();
}
});
player.prepare();

...

// 暂停播放
player.pause();

...

// 恢复播放
player.resume();

...

// 停止播放
```

```
player.stop();

...

// 释放播放器资源
player.release();
...
```

接口说明:

ExoHlsPlayer

- 构造方法

```
    ExoHlsPlayer(Context context);
```

- 设置m3u8播放地址

```
    /**
* 设置播放源
* @param url m3u8地址
*/
void setDataSource(String url);
```

- 设置播放地址为已接入飞燕的IPC设备指定录像名的云端录像文件地址.

```
    /**
* 设置播放地址为已接入飞燕的IPC设备指定录像名的云端录像文件地址.
*
* @param iotId 设备iotId
* @param fileName 录像文件名
*/
void setDataSourceByIPCRecordFileName(String iotId, String fileName);
```

- 校验和准备数据

```
    /**
* 校验和准备数据
*/
void prepare();
```

- 开始或恢复播放视频

```
    /**
* 开始播放或恢复播放视频
*/
void start();
```

- 暂停播放

```
    /**
* 暂停播放, 调用start()恢复播放
*/
void pause();
```

- seek到指定位置

```
    /**
* seek到指定位置
* @param position 毫秒
*/
void seekTo(long positionInMs);
```

- 停止播放

```
    /**
* 停止播放
*/
void stop();
```

- 重置播放器

```
    /**
* 重置播放器
*/
void reset();
```

- 释放播放器资源

```
    /**
* 释放播放器资源
*/
void release();
```

- 设置是否循环播放

```
    /**
* 设置是否循环播放
* @param circlePlay true 为循环播放
*/
void setCirclePlay(boolean circlePlay);
```

- 设置回放速率

```
    /**
* 设置回放的播放速率
* @param speed 速率因子，需大于0
*/
void setPlaybackSpeed(float speed);
```

- 设置播放器音量

```
    /**
* 设置播放器音量
* @param audioVolume 范围 0-1, 0为静音
*/
void setVolume(float audioVolume);
```

- 设置音频流通道类型

```
    /**
* 设置音频流通道类型,see {@link android.media.AudioManager}
* 如果音频正在播放，则会因为重新创建AudioTrack导致有短暂停顿.
* @param streamType
*/
void setAudioStreamType(int streamType);
```

- 设置画面缩放模式

```
    /**
* 设置视频画面缩放模式, 默认为{@link
android.media.MediaCodec#VIDEO_SCALING_MODE_SCALE_TO_FIT_WITH_CROPPING}
* @param videoScalingMode 参考:
* {@link android.media.MediaCodec#VIDEO_SCALING_MODE_SCALE_TO_FIT}
* {@link android.media.MediaCodec#VIDEO_SCALING_MODE_SCALE_TO_FIT_WITH_CROPPING}
*/
void setVideoScalingMode(int videoScalingMode);
```

- 设置surfaceview

```
    /**
* 设置SurfaceView
* @param surfaceview
*/
void setSurfaceView(SurfaceView surfaceview);
```

- 清除surfaceview

```
    /**
* 清除surfaceview
*/
void clearSurfaceView();
```

- 设置数据源准备就绪事件监听器

```
    /**
* 设置数据源准备就绪事件监听器
*
* @param listener
*/
void setOnPreparedListener(OnPreparedListener listener);
```

- 设置播放器错误事件监听器

```
    /**
* 设置播放器错误事件监听器，错误类型参考:
* {@link PlayerException.SOURCE_ERROR}
* {@link PlayerException.RENDER_ERROR}
* {@link PlayerException.UNEXPECTED_ERROR}
* @param listener
*/
void setOnErrorListener(OnErrorListener listener);
```

- 设置播放状态变更事件监听器

```
    /**
* 设置播放状态变更事件监听器
* @param listener
*/
void setOnPlayerStateChangedListener(OnPlayerStateChangedListener listener);
```

- 设置首帧被渲染事件监听器

```
    /**
* 设置首帧被渲染事件监听器
* @param listener
*/
void setOnRenderedFirstFrameListener(OnRenderedFirstFrameListener listener);
```

- 获取当前播放进度

```
    /**
* 获取当前播放进度
* 播放器状态为{@link PlayerState#STATE_READY}时调用有效
* @return 单位MS
*/
long getCurrentPosition();
```

- 获取视频总时长

```
    /**
* 获取视频总时长
* 播放器状态为{@link #STATE_READY}时调用有效
* @return 单位MS
*/
long getDuration();
```

- 获取音量

```
    /**
* 获取音量
* @return 范围0-1
*/
float getVolume();
```

- 获取播放器状态

```
    /**
* 获取播放状态
* @return 状态枚举，参考:
* {@link PlayerState#STATE_IDLE} 播放器初始状态
* {@link PlayerState#STATE_BUFFERING} 缓冲中状态
* {@link PlayerState#STATE_READY} 缓冲结束开始播放状态
* {@link PlayerState#STATE_ENDED} 播放完成状态
*/
int getPlayState();
```