YFPlayerKit 帮助文档

文档修改记录

修改日期	版本/状态	作者	修改章节	修改描述
2015/12/23	2.0	谢家勋	略	附录【1】
2016/1/7	2.1	谢家勋	略	附录【2】
2016/1/16	3.0	谢家勋	略	附录【3】
2016/3/24	4.0	谢家勋	略	附录【4】
2016/4/16	5.0	谢家勋	略	附录【5】
2016/4/21	5.1	谢家勋	略	附录【6】
2016/5/5	5.2	谢家勋	略	附录【7】
2016/6/8	5.3	谢家勋	略	附录【8】
2016/6/17	5.3.1	谢家勋	略	附录【9】
2016/7/1	5.3.2	谢家勋	略	附录【10】
2016/7/4	5.3.3	谢家勋	略	附录【11】

目录

SDK 适用范围

-该 sdk 提供 Android 移动平台的播放功能。
-支持文件格式和协议：
flv, mp4, ts, rtmp, http, hls 等协议的直播和点播

-支持以下功能：
1、 软解播放
2、 硬解播放
3、 智能识别当前视频是否支持硬解
4、 支持硬解失败自动切换软解
5、 支持播放过程中缓冲超时自动重连
6、 动态调整音视频同步
7、 可获取播放视频的相关信息
8、支持纯音频和纯视频播放

SDK 鉴权

该 SDK 在使用前需进行验证，用户需到云帆官方网站注册，申请开通流媒体引擎服务。并获取 Access Key ID 和 Token。
其中 AccessKey 会被封装入用户拿到的 sdk，Token 则由用户填写。
注册完成后，用户可以自行填写域名白名单。只有该名单内的域名视频，才可以通过 SDK 的播放器播放。
用户需要在播放视频之前进行一次验证。验证方法如下：
Authentication.Authenticate(TOKEN);
一般而言只需要在打开 App 的时候进行一次鉴权即可。
然后就可以愉快地使用播放器了！

API for YfPlayerKit

YfPlayerKit 是一个用于播放视频的类，由 Android 的 VideoView 重写而来，大部分使用方法与 VideoView 类似。

常量：

常量类型	常量名	含义
缩放参数	VIDEO_LAYOUT_FIT_PARENT	适配父控件
	VIDEO_LAYOUT_FILL_PARENT	填充父控件
	VIDEO_LAYOUT_WRAP_CONTENT	适应子控件
	VIDEO_LAYOUT_MATCH_PARENT	适应父控件
	VIDEO_LAYOUT_16_9_FIT_PARENT	16:9 适配父控件
	VIDEO_LAYOUT_4_3_FIT_PARENT	4:3 适配父控件
解码类型	MODE_HARD	硬解模式
	MODE_SOFT	软解模式
onInfo()回调参数	INFO_CODE_UNKNOWN	未知信息
	INFO_CODE_VIDEO_RENDERING_START	视频开始渲染
	INFO_CODE_VIDEO_TRACK_LAGGING	视频延迟
	INFO_CODE_BUFFERING_START	缓冲开始

	INFO_CODE_BUFFERING_END	缓冲结束
	INFO_CODE_NETWORK_BANDWIDTH	网络带宽
	INFO_CODE_BAD_INTERLEAVING	交错异常
	INFO_CODE_NOT_SEEKABLE	无法 seek
	INFO_CODE_METADATA_UPDATE	Metadata 更新
	INFO_CODE_VIDEO_ROTATION_CHANGED	视频方向旋转
	INFO_CODE_AUDIO_RENDERING_START	音频开始播放
onError()回调参数	ERROR_CODE_UNKNOWN	未知错误
	ERROR_CODE_SERVER_DIED	服务器停止
	ERROR_CODE_NOT_VALID_FOR_PROGRESSIVE_PLAYBACK	播放不连续
	ERROR_CODE_IO	IO 异常
	ERROR_CODE_MALFORMED	数据异常
	ERROR_CODE_UNSUPPORTED	格式不支持
	ERROR_CODE_TIMED_OUT	连接超时

公共方法

public void setVideoPath(String path)

设置视频路径，可以是网络路径或者本地视频路径。设置路径后，如若一切正常，会直接开始播放。

public void setHardwareDecoder(Boolean hardware)

切换解码方式。当不调用该方法时默认使用硬解。如有需要需手动记录播放时间。

参数

hardware：true 代表使用硬解；false 代表使用软件。

参考代码：

```
int mSeekWhenPrepared = mVideoView.getCurrentPosition();//记录当前播放进度
mVideoView.setHardwareDecoder(false);//配置解码方式
mVideoView.seekTo(mSeekWhenPrepared);//继续上次播放进度
```

void setOnPreparedListener (OnPreparedListener listener)

void setOnCompletionListener (OnCompletionListener listener)

void setOnBufferingUpdateListener (OnBufferingUpdateListener listener)

void setOnSeekCompleteListener (OnSeekCompleteListener listener)

void setOnVideoSizeChangedListener (OnVideoSizeChangedListener listener)

void setOnErrorListener (OnErrorListener listener)

void setOnInfoListener (OnInfoListener listener)

设置播放器状态的回调方法。

public void setMediaController(MediaController controller)

设置 MediaController，同原生 Android API。

public void initVideoView()

重新注册回调等相关信息，详见注意事项 3。

public void setVideoLayout(int layout)

设置视频缩放模式。

参数

layout：如上述常量。

public void start()

开始播放。

public void pause()

暂停播放。

public void reset()

重置播放器。//该方法已抛弃。2016/4/15

public int getDuration()

获取视频长度。

public int getCurrentPosition()

获取当前播放时长，当不处于播放状态时，会返回 0。

public void seekTo(int pos)

跳至某时间点播放。

public boolean isPlaying()

判断是否正在播放中。

public int getAudioSessionId ()

恒定返回 0。

public void setVolume(float leftVolume, float rightVolume)

设置音量。

public int getBufferPercentage()

获取缓存进度。

public boolean canPause()

public boolean canSeekBackward()

public boolean canSeekForward()

以上三个方法暂时无效。恒定返回 true。

public MediaInfo getMediaInfo()

返回包含当前视频相关信息的 MediaInfo。

public String getDataSource()

返回当前视频地址，异常情况返回""。

public SurfaceHolder getSurfaceHolder()

返回 surfaceholder。

public void suspend ()

挂起视频。

public void resume ()

恢复播放。

public void setSurfaceCallBack(SurfaceHolder.Callback callBack) ()

设置 SurfaceHolder 回调。

public void setDelayTimeMs(int defaultDelayTime,int maxDelayTime)

设置最大延迟时间，该方法设置会使 setBufferSize(int)失效。

播放器会动态调整延迟时间在[defaultDelayTime,maxDelayTime]区间

defaultDelayTime 默认延迟时间，最小 500ms

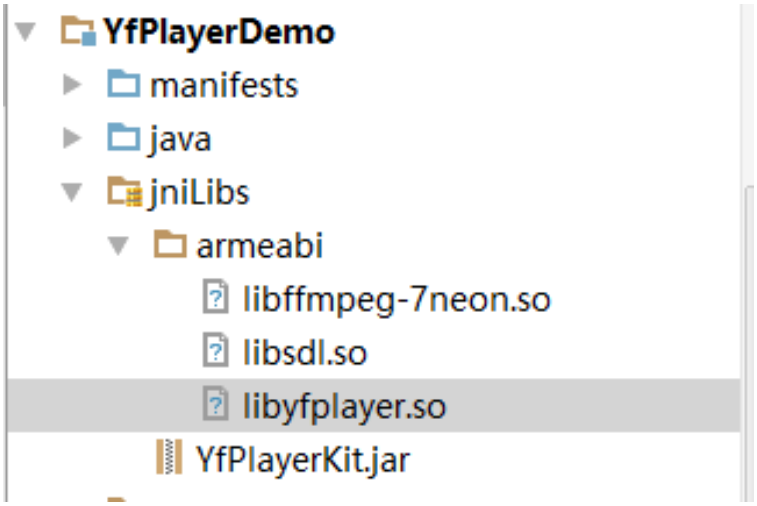
maxDelayTime 最大延迟时间，播放器会根据缓冲状况自动调整缓存大小，而该调整区间将始终不大于该值。

public void setHTTPTimeOutUs(int timeOutUs)

设置 HTTP 最大延迟时间（播放过程中），仅针对某些 TCP 策略异常的设备，无特殊需要请勿设置。

YfPlayerKit 使用方法

使用 YFPlayerKit 的方法与谷歌官方 VideoView 类似，先导入 jar 包及 SO 库（如下图）：



1、在 layout 中使用 YfVideoView

```
<com.yunfan.player.widget. YfPlayerKit
    android:id="@+id/surface_view"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

2、根据需要设置 YfVideoView 相关参数及回调方法。

```
    mVideoView=(YfPlayerKit)findViewById(R.id.surface_view);
```

3、设置视频路径（或链接）。

```
    mVideoView.setVideoPath(path);
    mVideoView.start();
```

具体使用可以参考提供的 Demo。

API for YfCloudPlayer

内部类

public static class Factory

用来生成播放器实例的类，调用其方法构造 Player 实例：

public static YfCloudPlayer createPlayer(Context mContext, int decodeMode)

参数：

mContext：上下文

decodeMode：可设置为 YfCloudPlayer 类中的以下两个常量：

public static final int MODE_AUTO; 硬解模式

public static final int MODE_SOFT;软解模式

公共方法

MediaInfo getMediaInfo();

获取媒体信息

void setDataSource(String path);

设置播放路径

void prepareAsync();

开始异步准备，等待准备完成回调

void start();

开始播放，请务必保证在 onPrepared()回调之后使用

void pause();

暂停播放

void stop();

停止播放

int getVideoWidth();

获取视频宽度，在 onVideoSizeChanged()回调之后使用

int getVideoHeight();

获取视频高度，在 onVideoSizeChanged ()回调之后使用

int getVideoSarNum ();

获取采样宽高比分子，在 onVideoSizeChanged ()回调之后使用

int getVideoSarDen ();

获取视频宽度比分母，在 onVideoSizeChanged ()回调之后使用

void seekTo(int pos);

定位到指定播放位置，在 onPrepared()回调之后使用

int getCurrentPosition();

获取当前的播放位置，当且仅当视频未准备好时返回 0

long getDuration();

获取视频时长，直播流也可能获得不等于 0 的长度，以<1000 为条件区分直播流和非直播流；单位毫秒

boolean isPlaying();

获取视频是否在播放状态

String getDataSource ()

返回当前视频地址。

void setVolume(float leftVolume, float rightVolume);

设置音量

void setDisplay(SurfaceHolder sh);

设置播放 surface。

void setBufferSize(int bufferSize)

设置缓存大小。

void reset();

重置播放器到初始状态。

void release();

释放播放器资源，此时该播放器将不能调用任意接口。

boolean canContinueRender();

判断当前播放器能否直接开始渲染视频。

void stopRender();

仅仅停止渲染而不重置播放器状态。

void continueRender(SurfaceHolder holder);

继续渲染通过 stopRender()停止渲染的视频。

int getAudioSessionId();

获取 AudioSessionId。

public void setDelayTimeMs(int defaultDelayTime,int maxDelayTime)

设置最大延迟时间，该方法设置会使 setBufferSize(int)失效。

播放器会动态调整延迟时间在[defaultDelayTime,maxDelayTime]区间

defaultDelayTime 默认延迟时间，最小 500ms

maxDelayTime 最大延迟时间，播放器会根据缓冲状况自动调整缓存大小，而该调整区间将始终不大于该值。

public void setHTTPTimeOutUs(int timeOutUs)

设置 HTTP 最大延迟时间（播放过程中），仅针对某些 TCP 策略异常的设备，无特殊需要请勿设置。

public void setBuffingOnPrepared()

设置当视频 prepared 之后即开始下载数据，主要用于视频的预加载。该方法只在 setDataSource()之前调用有效，且会在 start()后恢复默认设置（默认设置为在 start 之后调用有效）

设置回调

OnPreparedListener

调用 prepareAsync()后，播放器进入准备状态，准备完成可以播放后会回调该方法。

OnCompletionListener

当当前视频播放完毕后会回调该方法；直播流不会回调该方法。

OnSeekCompleteListener

当 seek 完成后会回调该方法。

OnBufferingUpdateListener

当视频开始缓冲、缓冲中、缓冲结束会回调该方法。

OnVideoSizeChangedListener

当视频大小发生改变时会回调该方法。

OnErrorListener

当出现视频必须停止播放时的错误时会回调该方法。错误类型见末节【回调信息】。

OnInfoListener

视频播放过程中提示用户相关信息时会回调该方法。回调内容见末节【回调信息】。

使用方法

使用 YfCloudplayer 的基本方法与谷歌 MediaPlayer 类似。基本流程如下：

1、设置各种回调方法。

2、初始化播放器，参考代码如下：

```
YfMediaPlayer = YfCloudPlayer.Factory.createPlayer(this, MediaInfo.MODE_HARD);
YfMediaPlayer.setDataSource(path);
YfMediaPlayer.setDisplay(mSurface);
YfMediaPlayer.prepareAsync();
YfMediaPlayer.setOnCompletionListener(mCompletionListener);
YfMediaPlayer.setOnPreparedListener(mPreparedListener);
YfMediaPlayer.setOnVideoSizeChangedListener(mSizeChangedListener);//设置需要的回调
```

3、在 OnPrepared()回调中调用播放接口：start()。

4、设置 SurfaceHolder 回调。

视频播放在不同 Surface 间无缝切换

对于一些应用场景需要在不同 Surface 间切换视频的用户，YfCloudPlayer 提供了对应的接口满足需求。下面就页面 A 切换到页面 B，在各自的窗口播放同一个视频的场景，介绍使用方法如下：

1、维护同一个 YfCloudPlayer。仅切换 surface 需保证是同一个 YfCloudPlayer 实例，本 demo 采用最简单的静态变量来实现，具体实现方式请根据 App 场景选择。

```
public static YfCloudPlayer YfMediaPlayer;
```

2、确保该 YfCloudPlayer 不会在 surfaceDestroyed()回调中(或任意继续播放前的场景中)被 release()/reset()。

```
@Override
public void surfaceDestroyed(SurfaceHolder holder) {
    Log.d(TAG, "surfaceDestroyed");
    if (!keepPlayerAlive)
        YfMediaPlayer.release();
}
```

3、在退出页面 A 时调用 stopRender()方法，只停止渲染而不释放。

```
if (//让下个界面能继续播放该视频) {

    keepPlayerAlive = true;//保证后续状态不会重置 YfMediaPlayer

    YfMediaPlayer.stopRender();//只停止渲染

}
```

4、在页面 B 调用 continueRender(SurfaceHolder holder)方法，并将新 Surface 的 holder 作为参数传入。可以通过 canContinueRender()判断是否可以继续渲染。

```
if (YfMediaPlayer != null && YfMediaPlayer.canContinueRender()) {
    YfMediaPlayer.continueRender(holder);
} else {

    openVideo();//正常的播放新视频

}
```

预加载视频（无缝播放下一个视频）

场景：希望在播放完当前视频（点播）后，尽可能快地播放下一个视频（点播/直播）。
具体使用方法参考 Demo 里的 SmoothPlayDemo.java
思路简介：

1、维护两个 YfCloudPlayer，循环复用。

```
for (int i = 0; i < yfPlayer.length; i++) {
     yfPlayer[i] = YfCloudPlayer.Factory.createPlayer(this,
YfCloudPlayer.MODE_HARD);
     yfPlayer[i].setBufferSize(8 * 1024 * 1024);
}
```

2、正常播放第一个视频。

3、设置计时器，每隔 2s 判断一次剩余时长，当剩余时间在 10s 内时，预加载第二个视频。（该策略根据 App 场景自行制定）

```
public void prepareAsNextPlayer() {
    //和一般的 prepare 不一样的在于，display 设置为 null，在 setDataSource 之前调用
setBuffingOnPrepared
    Log.d(TAG,"prepare as next player!"+inerPlayer);
    setListener();//设置回调
    inerPlayer.setDisplay(null);
    try {
        inerPlayer.setBuffingOnPrepared();
        inerPlayer.setDataSource(inerPath);
```

```
        } catch (IOException e) {
            e.printStackTrace();
        }
        inerPlayer.prepareAsync();//记得在收到 onprepared()回调的时候判断一下，不要立即 start()
    }
```

4、 在第一个视频收到 OnComplete 回调之后，立即调用 reset()，并通知预加载的第二个播放器设置 surface 并开始播放

```
    @Override
    public void onCompletion(YfCloudPlayer mp) {
        inerPlayer.reset();//释放资源，并重置该播放器
        next();//通知预加载的播放器开始播放
    }
```

5、 循环以上步骤轮播视频。

回调信息

包括 OnInfo()和 OnError()，回调值参考 YfPlayerKit 的常量列表。

MediaInfo 类

存储了播放中视频的相关信息，包括持续时间、比特率、视频长宽等，使用方法参考 Demo。

附录

【1】 2.0 版本文档修改记录：、
删除方法：canSeek()、canPause()、release()、reset();
增加方法：setAutoSwitchDecodeMode()、setHardwareDecoder()、getMediaInfo();
新增类 ：MediaInfo
替换类 ：YfVideoView->YfPlayerKit

【2】 2.1 版本文档修改记录：
增加缩放参数：VIDEO_LAYOUT_FULL_PARENT

【3】 3.0 版本文档修改记录：
增加新的代码层：IMediaPlayer
增加新章节:回调信息

【4】 4.0 版本文档修改记录:
增加 YfPlayerKit 下的方法：initVideoView(),reset(),getPlayingVideoPath()
增加 IMediaPlayer 下的方法：reset(),getPlayingVideoPath()
新增 API for YfPlayerKit 下的注意事项

【5】 5.0 版本文档修改记录:
YfPlayerKit:
删除方法:
reset()、initVideoView()、getPlayingVideoPath()、
setAutoSwitchDecodeMode(Boolean auto)、
setMediaPlayerListener(OnMediaPlayerListener listener)
新增方法:
getDataSource()、getSurfaceHolder()、suspend()、resume()、getAudioSession()、setVolume(float leftVolume, float rightVolume)
删除注意事项。
IMediaPlayer 改名为 YfCloudPlayer:
增加无缝切换的相关函数、增加获取视频宽高比的方法。
增加无缝切换的使用方法。
删除注意事项。

【6】 5.1 版本修改记录：
YfPlayerKit：
在设置播放路径后不再直接进入播放，而需要调用 start();在其他页面返回播放界面时也不会再自动播放，而需要手动调用 start()

【7】 5.2 版本修改记录：
YfPlayerKit：
增加方法 setMaxDelayTimeMs();
YfCloudPlayer:
增加方法 setMaxDelayTimeMs();

【8】 5.3 版本修改记录:
YfPlayerKit：
增加方法 setHTTPTimeOut();
YfCloudPlayer:

增加方法 setHTTPTimeOut ();

增加方法 setBuffingOnPrepared()

增加预加载视频的参考使用方法。

【9】 5.3.1 版本修改记录：

YfPlayerKit：

修正方法 setVideoLayout()的参数与实际 sdk 参数不符的问题。

【10】 5.3.2 版本修改记录:

setMaxDelayTimeMs(int delayTimeMs)方法更改为 setDelayTimeMs(int defaultDelayTime,int maxDelayTime)

如果设置该方法，播放器将动态更改缓存

【11】 5.3.3 版本修改记录：

添加 sdk 鉴权的方法。