媒体云播放器

Android SDK 用户手册

V1.4

发布日期： 2013年8月8日

百度开发者中心

目录

第1章 概述

百度媒体云播放器 Android SDK（以下简称“播放器 SDK”）是百度官方推出的 Android 平台使用的软件开发工具包（SDK），为 Android 开发者提供简单、快捷的接口，帮助开发者实现 Android 平台上的媒体播放及多屏互动应用开发。

播放器 SDK 内嵌百度自主研发的 T5 播放内核，对目前主流的本地和网络媒体都提供了良好的功能支持，弥补了系统播放器在媒体支持格式上的不足，并在兼容性、稳定性和响应速度上有明显的提高。

播放器 SDK 提供了多种层面的调用方式，开发者可根据自己的需求定制化开发播放界面，也可使用默认播放界面实现快速开发。

同时播放器 SDK 提供了跨越电视、PC、平板或智能手机的多媒体互动播放的能力，目前支持 Android 端的 DLNA(Digital Living Network Alliance)控制器，可控制市面上兼容 DLNANA 协议的电视、机顶盒产品和大量智能设备，支持音频、视频、图片的跨设备播放，能实现基本的遥控器功能，提供丰富便捷的互动体验。

播放器 Android SDK 提供了两种集成模式：

- 共享播放引擎方案

  该方案下，各播放应用使用同一播放引擎，应用包本身容量可以大大减小。该方案 SDK 版本号为*.*，如 1.4。

- Jar 包及 so 动态库集成方案

  该方案下，各播放应用使用独享的播放引擎，免去了安装 apk 播放引擎的过程。该方案 SDK 版本号为*.*s，如 1.4s。

同版本号下，两种方案 SDK 提供的功能是一致。

本手册为 Jar 包及 so 动态库集成方案的用户手册，共享播放引擎方案请到百度开发中心网站（developer.baidu.com）的媒体云服务下载相关开发包及用户手册。

播放器 SDK 的完整下载包中包含 demo、doc、lib 和用户指南四个部分，目录结构如下所示：

- demo：主要存放 3 个 Android 示例工程，可以帮助用户了解如何使用该 SDK。其中 sample1 是同时调用 BVideoView 和 BMediaController 相关的示例；sample2 则是只调用 BVideoView 的相关示例。sample3 是通过 intent 的方式调用示例。DLNAControllerDemo 则是实现多屏互动的示例。
- doc： 主要存放播放器 SDK 相关接口参考文档，可离线查看 index.html 获取 API 的具体说明。
- lib： 主要存放播放器 SDK jar 包，即：Baidu_Cyberplayer_SDK_Android.jar。
- SDK 用户手册

第2章 阅读对象

本文档面向所有使用该 SDK 的开发人员、测试人员、合作伙伴以及对此感兴趣的其他用户，要求读者具有一定的 Android 编程经验。

第3章 播放器 SDK 功能说明

播放器 SDK 以开发者为中心，以高效创建媒体播放应用为目标，具有以下的特色功能：

- 低门槛、高灵活度实现播放功能

  利用 SDK 提供的 API 接口或者快捷的 Intent 调用方式，轻松创建专业级播放应用；UI 界面可自由定制。

- 轻松实现多屏互动

  利用 SDK 提供的 API 接口，轻松实现基于 DLNA 的多屏互动应用，实现图片、视频在不同手机、平板、电视间分享和控制。。

- 流媒体格式完美支持

  跨 Android 版本(2.2 以上)完美支持 HTTP Streaming 及 HTTP Live Streaming(HLS，M3U8）流媒体协议

- 智能硬件兼容与智能升级

  开发者不需要关注硬件平台的差异，播放器 SDK 内部的播放引擎会智能适配各种硬件设备，自动从云端获取最适合底层硬件特性的内核实现。同时，开发者再也不用为 SDK 后续的功能升级所困扰，内核支持智能升级。

- 无缝支持百度个人云存储（PCS）视频转码接口

  在播放百度个人云存储中的视频时，可根据设备计算能力及带宽的不同，调用百度云转码服务，智能转码实现移动端的流畅播放。

- 本地全媒体格式支持

  支持目前所有主流的媒体格式（mp4，avi，wmv，flv，mkv，mov，rmvb 等）。

- 智能统计

  开发者可以通过 web 管理界面，及时有效的查看播放器 SDK 给出的各种统计及分析结果。

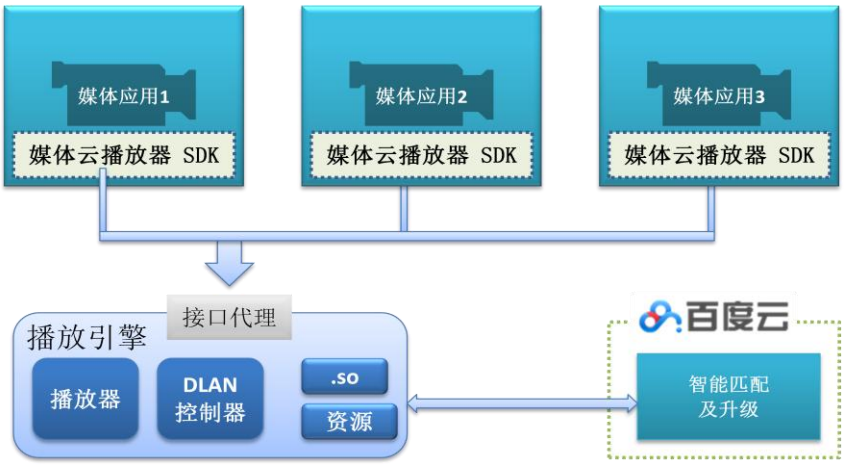
第4章 播放器 SDK 使用说明

4.1 应用中集成 SDK

开发者需要将播放器 SDK 集成到应用中方可使用，媒体云服务提供的是一种轻量级的集成方式，即多个应用可以共享一套播放引擎。

轻量级集成方式采用应用内轻量级 SDK 和应用间播放引擎共享的架构，实现多个应用使用同一套引擎，应用与内核彼此独立；不仅减少了应用程序大小，也免去了开发者自己去适配和升级引擎的工作。

设计框架结构如下图所示：



图表 1 播放器 SDK 框架设计

应用集成播放器 SDK 后，需要检测并安装播放引擎；应用在初次启动时会自动检测并安装引擎，并自动完成硬件版本检测及下载过程。播放器 SDK 后续升级会保持 API 接口向下兼容。

4.2 使用 SDK 实现媒体播放功能

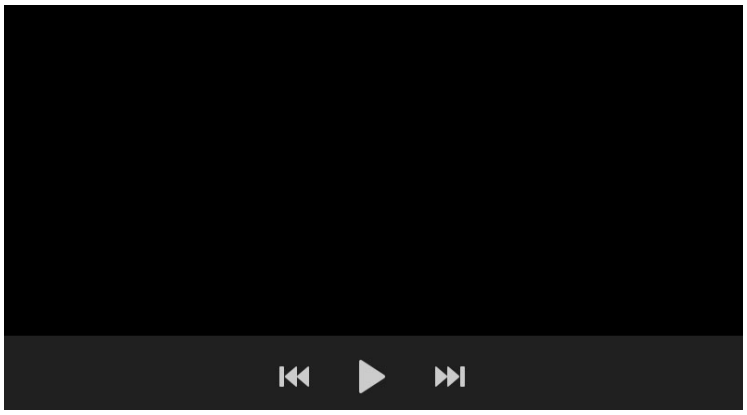
使用 SDK，可实现以下两种播放界面的：

- 自定义播放界面
- 使用内置播放界面

4.2.1 自定义播放界面

为满足开发者构建复杂、个性媒体播放应用，开发定制化播放器的需求，播放器 SDK 为开发者提供 BVideoView 和 BMediaController 两个类接口，其开发方式与基于 Android 原生播放器开发相同，具体请参考 API 调用中“自定义播放界面”的相关说明。

其中，BVideoView 为媒体播放显示提供支持；BMediaController 则为媒体播放控制提供支持，其中包括“播放\暂停”键及其它两个自定义键，可实现“上一部”及“下一部”功能。（具体界面功能如下图所示）



图表 2BVideoView 和 BMediaController 界面功能

4.2.2 使用内置播放界面

播放器 SDK 的播放引擎也同时提供了完整的媒体播放和操作界面。

开发者如果不需要修改播放界面，可使用 Intent 方式调用播放引擎，具体调用方式请参考 API 调用中的“使用内置播放界面”相关说明。

内置播放器界面的主要功能如下图所示：

图表 3 内置播放界面功能展示

媒体播放器界面支持四类功能：播放控制、功能控制、信息展示及手势控制。

1. **播放控制**
   - **“1. 播放/暂停按钮”**：播放和暂停状态切换
   - **“2. 上一部/下一部”**：播放播放列表中当前文件的上一文件或者下一文件
   - **“10.播放进度条”**：拖动以更改播放进度
2. **功能控制**
   - **“8.返回”**：点击后返回调用的程序；
   - **“4.音量按钮和音量条”**：点击后静音，拖动音量条可改变音量

- **“6.显示比例调节”**：原始模式对应视频原始大小、等比维持视频比例缩放到全屏，屏幕两侧可能出现黑边，铺满则不维持比例直接缩放到全屏

3. 信息展示

- **“9.状态展示”**：更改音量及屏幕亮度，拖动调节时显示具体状态。
- **“3.播放位置/总时长”**：显示当前播放时间和影片时长；
- **“7.标题栏”**：单行跑马灯显示播放内容的名字；

4. 手势控制

- **音量控制**：屏幕右半区，单指竖直向上滑动为增大音量，向下为降低音量；
- **屏幕亮度控制**：屏幕左半区，单指竖直向上滑动为增加亮度，向下为减小亮度；
- **30s 内快进/快退**：屏幕单指横向向左滑动为快退，向右滑动为快进。

4.3 使用 SDK 实现多屏互动

SDK 除了支持视频播放外，还提供了基于 DLNA 控制器的多屏互动支持，来实现手机，平板电脑，PC，以及智能电视（机顶盒）之间的内容的共享与互动。整个库包含一套完整的 SDK，内含丰富的 API，涵盖 DLNA 激活与停止，设备的发现，资源文件的设定，播放控制，状态查询，事件订阅及出错通知，帮助开发者轻松创建功能强大的媒体应用，为用户提供优质家庭影音及娱乐新体验。

主要包含以下特点：

- 支持图片，音频，视频等多种格式的媒体文件
- 良好的兼容性，支持主流电视（Sony）或机顶盒（小米盒子，Letv，快播大屏幕）
- 功能强大，支持拖动，音量控制等功能，完全可以代替遥控器
- 延时小，反应迅速；体积小，运行时资源损耗小
- 使用方便，只需导入 jar 包即可，无额外库依赖
- 同步调用，简化编程；异步通知，实时可靠

SDK 目前仅开放了 DLNA 的控制器功能，使用 API 可以发现并控制网络中的其他 DLNA 设备，如果希望将智能设备支持 DLNA 协议并能够被其他的控制器控制，请关注后续的 SDK 更新。

第5章 开发前准备

5.1 运行环境

- Android 2.2 及以上的所有系统
- 支持的硬件 CPU 目前覆盖：

  ARM v5，ARM v6，ARM v7，ARM v7 Neon 及 Intel X86

5.2 参数申请及权限开通

开发者需要使用百度账号登录百度开发者中心注册成为百度开发者并创建应用，方可获取应用ID、对应的 API Key（即：ak）及 Secret Key（即：sk）等信息。具体信息，请参考百度开发者中心上的“创建应用”的相关介绍。

SDK 认证时必须传入 ak 及 sk（只需前 16 位）参数。

第6章 使用 SDK 开发媒体播放应用

6.1 添加播放器 SDK 到 App 工程

请参考以下步骤，将播放器 SDK 添加到 App 工程中：

1. 创建一个 Android 工程；
2. 将 Baidu_CyberPlayer_SDK_Android.jar 添加到 App 工程的 libs 目录下；
3. 打开工程“Properties” > “Java Build Path”>“Add Jars”；
4. 浏览“Baidu_CyberPlayer_SDK_Android.jar”完成添加。

6.2 权限声明

在您的 Android App 的 AndroidManifest.xml 中声明如下权限：

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

6.3 调用 API

下面介绍如何调用播放器 SDK 中已封装的 API 完成各项操作：

6.3.1 初始化 BCyberPlayerFactory

首先初始化 BcyberPlayerFactory：

```
    BCyberPlayerFactory.init(Context context);
```

6.3.2 安装 CyberPlayerEngine

使用 BVideoView 和 BMediaController 之前，必须确保 CyberPlayerEngine 是否安装。

可通过如下方式判断并安装 CyberPlayerEngine：

```
    BEngineManager mgr = BCyberPlayerFactory.createEngineManager();
    //判断 CyberPlayerEngine 是否已安装
    Boolean isInstalled = mgr.EngineInstalled();
    //如果未安装，则执行安装
         If(!isInstalled){
            mgr.installAsync(new OnEngineListener(){
                @Override
                public boolean onPrepare() {
                      return true;
                }

                @Override
                public int onDownload(int total, int current) {
                      // TODO Auto-generated method stub
                      return 0;
                }

                @Override
                public int onPreInstall() {
                      return 0;
                }

                @Override
                public void onInstalled(int result) {
                      // TODO Auto-generated method stub

                }

         });
}
```

说明：

由于播放器 SDK 采用了 SDK 和 Engine 分离的设计策略，而所有播放功能均依赖于 Engine；开发者需留意以下回调函数，根据自身需要设计安装流程，以确保安装成功。

1. **boolean onPrepare()**

   说明：

   - 返回 true，会继续后面的安装过程
   - 返回 false，则会直接回调 onInstalled(ret)结束安装。

2. **int onDownload(int total, int current)**

   说明：

   - 会报告下载总大小及当前下载进度；
   - 返回 DOWNLOAD_CONTINUE，表示继续；
   - 返回 DOWNLOAD_STOP，则会结束下载，但不会删除已下载文件，重新安装会继续下载；
   - 返回 DOWNLOAD_CANCEL，则会停止下载并删除已经下载文件，重新安装会

重新下载；

- 最终返回 onInstalled(ret)，则安装过程结束，可通过返回值 ret 判断安装结果。

3. int onPreInstall()

说明：

- 下载已经完成，准备进行安装；
- 返回 DOWNLOAD_CONTINUE，表示继续；
- 返回 DOWNLOAD_STOP，则会停止安装，但不会删除已下载文件，重新安装会直接使用已下载文件；
- 返回 DOWNLOAD_CANCEL 会停止安装过程并删除已下载文件，重新安装会重新下载；
- 最终返回 onInstalled(ret)，则安装过程结束，可通过返回值 ret 判断安装结果

4. void onInstalled(int result)

说明：

- 成功安装完成后会删除已下载的.apk 文件；
- 关于 installAsync 过程中的回调及处理的详细内容，请参考《API 参考文档》。
- 在确保 CyberplayerEngine 成功安装后，可直接以 Intent 方式调用媒体播放功能。

6.3.3 实现媒体播放功能

如 4.2 节所述，使用 SDK 实现媒体播放功能有两种方式：“自定义播放界面”及“使用内置播放界面”。本节即针对上述两种方式的具体实现进行详细说明。

6.3.3.1 自定义播放界面

1. 初始化 CyberPlayerEngine

CyberPlayerEngine 安装成功后，需要对其进行初始化，实现代码如下：

```
        BEngineManager mgr =
BCyberPlayerFactory.createEngineManager();
        mgr.initCyberPlayerEngine(ak,sk);
```

注意：

- 如果 BVideoView 及 BMediaController 在 layout 的 xml 文件中创建，mgr.initCyberPlayerEngine(ak, sk)必须在 setContentView()之前执行；为保证安全性，sk 仅传递前 16 位字节完成校验。
- 如果 BVideoView 及 BMediaController 通过 new 的方式动态创建，只需要确保 mgr.initCyberPlayerEngine(ak, sk)在 new 之前执行即可。

2. 创建 BVideoView 及 BMediaController

初始化 CyberPlayerEngine 后，请通过以下方式之一创建 BVideoView 及 BMediaController：

- 直接在布局文件中创建
- 在源码中动态创建，然后添加到页面中

方式 1：直接在布局文件中创建

示例如下：

```
…
        <RelativeLayout
       android:id="@+id/videoview_holder"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
android:layout_centerInParent="true"
   <com.baidu.cyberplayer.sdk.BvideoView
         Android:id=”@id/videoview”
         Android:layout_width=”fill_parent”
         Android:layout_height=”fill_parent”
   />
/>
…
   <RelativeLayout
       android:id="@+id/videoview_holder"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
android:layout_centerInParent="true"
   <com.baidu.cyberplayer.sdk.BMediaController
         Android:id=”@id/controllerbar”
         Android:layout_width=”fill_parent”
         Android:layout_height=”fill_parent”
   />
/>
…
```

方式 2： 在源码中动态创建，然后添加到页面中

示例如下：

```
   BVideoView mVV;
   BMediaController mVVCtl;
   …
         mViewHolder = (RelativeLayout)
findViewById(R.id.videoview_holder);
         mControllerHolder = (LinearLayout)
findViewById(R.id.controlbar_holder);

         mVV = new BVideoView(MediaControllerPlayingActivity.this);
         mVVCtl = new
BMediaController(MediaControllerPlayingActivity.this);
         LinearLayout.LayoutParams param = new
LinearLayout.LayoutParams(
                     LinearLayout.LayoutParams.FILL_PARENT,
                     LinearLayout.LayoutParams.FILL_PARENT);
         mViewHolder.addView(mVV, param);
         mControllerHolder.addView(mVVCtl, param);
```

3. 关联 BVideoView 与 BMediaController 实现播放

首先，可注册 listener 监听播放过程，以便应用程序根据监听结果进行相应处理。示例如下：

```
mVV.setOnPreparedListener();
mVV.setOnCompletionListener();
mVV.setOnErrorListener();
mVV.setOnInfoListener();
```

其次，关联 BVideoView 与 BMediaController，开始播放。示例如下：

```
mVV.setMediaController(mVVCtl);
mVV.setDecodeMode(BVideoView.DECODE_SW); //可选择软解模式或硬解模式
mVV.setVideoPath(path);
mVV.start();
```

其中 setDecodeMode()是设定软硬解模式。默认是软解模式，开发者可根据 Android 系统默认支持的格式情况来决定，如果是默认支持的格式（如 MP4、3GP）建议打开硬解开关以节省功耗、提高播放性能。

6.3.3.2 使用内置播放界面

如果 SDK 提供的内置播放界面和功能已可满足您的需求，可直接以 Intent 方式启用内置播放功能。在 Activity 中启动：

```
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.putExtra("isHW", boolean isHW); （可选）
Uri uri = Uri.parse(videoPath);
it.setClassName("com.baidu.cyberplayer.engine","com.baidu.cyberplayer.engine.PlayingActivity");
it.setData(uri);
startActivity(it);
```

说明：

其中 intent.putExtra("isHW", boolean isHW); 为设定是否开启硬解（默认"关闭"）。开发者可根据 Android 系统默认支持的格式情况来决定，如果是默认支持的格式（如 MP4、3GP）建议打开硬解开关以节省功耗、提高播放性能。

第7章 使用 SDK 开发多屏互动应用

SDK 开放了 DLNA 的控制器功能，实现多屏互动。

7.1 添加控制端 SDK 到 App 工程

参见 6.1 即可，权限声明参考 6.2 即可

7.2 调用 API

使用 DLNA 功能, API 的调用分为以下几步：

- 传入 AK，SK 进行初始化和权限验证

- 启动 DLNA 服务
- 获取 Render（电视等显示设备）列表
- 指定某个 Render 与之关联
- 指定待播放资源的 URL
- 播放控制阶段，开始，暂停，拖动，调解音量；
- 停止 DLNA 服务

其他可选步骤：

- 播放前获取 Render 能力，支持哪些格式和协议
- 播放过程中，主动获取 Render 状态信息
- 订阅 Render 状态，避免主动轮询
- 出错监听

注意：DLNAServiceProvider 是对外服务的唯一接口，在执行完初始化操作后，必须先调用 addActionCallback 注册回调， API 调用根据是否为耗时操作分为同步调用和异步调用；同步调用立即返回结果；异步调用立即返回，通过已注册对应的回调获取执行结果，由于回调不是在 UI 线程中执行，请勿在回调中执行 UI 相关操作；如果操作失败，还可以通过对应回调查看错误码和错误消息详情。API 文档中标明了具体 API 是同步还是异步操作。

7.2.1 获取 DLNA 服务实例

首先通过 DLNAServiceProvider 的 getInstance 方法获取服务实例：

```
DLNAServiceProvider::getInstance(Context context);
```

7.2.2 初始化 SDK

传入 AK，SK 调用以下接口进行初始化

说明：AK,SK 的获取参见 5.2

```
void initialize(String accessKey, String secretKey);
```

7.2.3 注册 ActionCallback

通过注册 ActionCallback，得到异步操作的结果

说明：该步骤是必须的，而且必须在所有 DLNA 操作之前进行执行

```
boolean addActionCallback(DLNAActionListener);
```

7.2.4 启动 DLNA 服务

调用 DLNAServiceProvider 的以下 API 即可启动底层服务，执行结果通过 DLNAActionlistener 中的 onEnableDLNA 回调返回，APP 运行过程中只需要启动一次该服务即可。

说明：异步调用,除非系统底层网络服务不可用，否则 onEnableDLNA 都将返回 true

```
void enableDLNA ();
```

7.2.5 获取 Render 列表

调用 DLNAServiceProvider 的以下 API 即可得到 Render 列表，如果当前网络没有 Render，则

返回 null：

```
List<String> getRenderList();
```

7.2.6 关联到某个 Render

调用 DLNAServiceProvider 的以下 API 即可关联到指定的 Render，接下来就可以对 Render 进行控制和设定了，通过对应回调获取关联结果，随后对统一 render 进行的操作都无需再次关联：

```
void selectRenderDevice(String devName);
```

7.2.7 设置待播放资源的 URL

调用 DLNAServiceProvider 的以下 API 即可设置媒体资源的 URL，DLNAActionlistener 中的 onSetMediaURI 将会返回设置结果：

```
void setMediaURI(String mediaURI);
```

7.2.8 DLNAServiceProvider 调用 API 进行播放控制

调用 DLNAServiceProvider 的以下 API 进行播放控制，由于交互对象是网络中的其他终端设备，控制和交互过程存在网络延时和设备响应延时，即 API 调用返回后，被控制设备可能会出现延后一段时间作出响应的情况。

被控制的终端所处的状态会影响 API 调用是否成功，譬如只有在播放状态下才能去进行 Seek 操作。所有的控制操作结果均通过 DLNAActionlistener 中的对应回调函数返回。

播放：

```
void play( );
```

暂停：

```
void pause();
```

停止播放：

```
void stop();
```

拖动：position：xx:xx:xx 格式

```
void seek(position);
```

设置音量：volume:音量百分比

```
void setVolume(int volume);
```

设置是否静音：

```
void setMute(Boolean isMute);
```

7.2.9 播放过程中获取被控设备状态

调用 DLNAServiceProvider 的以下 API 获取被控设备相关状态，如果操作失败，请结合错误码进行分析；通常由设备异常或者网络中断引起，请及时检查设备或网络状态。

获取音量：请求音量，获取音量后会返给注册的 DLNAActionlistener::onGetVolume;

```
void getVolume();
```

获取 Mute 状态：请求获取被控设备是否处于静音状态，获取后会将结果返给注册的 DLNAActionlistener::onGetMute;

```
void getMuteStat();
```

获取媒体文件播放时长：立即返回播放总时长

说明：可能返回 00:00:00，尝试重新获取；推荐注册 DeviceEventListener，duration 发生变化时底层负责通知上层最新的 duration 值，如果一直拿不到正常值，可以尝试主动获取

```
String getMediaDuration();
```

获取当前的播放进度：立即返回当前的播放位置

说明：可能返回 00:00:00，忽略并重新获取，推荐注册 DeviceEventListener，底层同时上层最新的 Position 值

```
String getMediaPosition() ;
```

获取 Render 的当前状态：立即返回 Render 的当前状态，推荐注册 DeviceEventListener，当远端 Render 设备状态变化时底层通知上层应用

```
String getRenderState()
```

7.2.10 停止 DLNA 服务

要结束 DLNA 的服务，请调用以下 API，便于系统进行资源回收及线程关闭，异步操作

```
void disableDLNA();
```

7.2.11 订阅被控设备状态变动通知

关联设备成功后，根据以下几步，可订阅被控设备的状态变动通知，避免主动轮询：

1. 实现 DeviceEventListener 接口
2. 添加自定义监听器

```
boolean addEventListener(DeviceEventListener listener);
```

说明：如果订阅底层事件更新，请保持数据来源的单一性，防止上层 App 和底层通知同时操作某个状态值导致的异常

7.2.12 获取被控设备的多媒体格式支持能力集

作为 Debug 工具使用，检查是否是超出设备播放能力造成的播放失败：

```
String getSupportedProtocols();
```

第8章 API 说明

有关 API 的详细说明，请查看 doc 目录下的《API 参考文档》。

第9章 播放器 SDK/Engine 升级

播放器 SDK 可通过百度开发者中心媒体云相关帮助文档处下载获取更新。

第10章 播放信息统计

媒体云服务为开发者提供了媒体应用播放的相关统计信息，开发者可以通过查看统计信息了解应用的使用现状。

进入媒体云服务管理控制台步骤如下：

- 登录进入百度开发者中心的“管理中心”；
- 点击使用媒体云服务的应用，进入应用基本信息页；
- 点击左侧边栏中的“云平台 > 媒体服务 > 媒体播放”，即可进入播放器 SDK 统计查看页面。

第11章 术语表

缩略语	英文全称	说明
SDK	Software Development Kit	软件开发工具包
DLNA	Digital Living Network Alliance	数字生活网络联盟，开放式互操作性指南

第12章 联系我们

如果以上信息无法帮助您解决在开发中遇到的具体问题，请通过以下方式联系我们：

邮箱: dev_support@baidu.com

百度工程师会在第一时间回复您。

第13章 文档变更历史

版本号	发布日期	描述
1.3	2013.6.18	新增使用 SDK 开发多屏互动应用内容
1.0	2013.5.10	百度媒体云播放器 Android SDK 用户手册 1.0 正式发布