Android 多媒体
开发指南

版本号: 1.2
发布日期: 2025.01.10

版本历史

版本号	日期	制/修订人	内容描述
1.0	2024.10.30	AWA2127	初始版本文档
1.1	2024.11.11	AWA1729	增加对 CedarC 版本的描述
1.2	2025.01.10	AWA1729	移除版权要求的视频编码格式

目　录

插　图

1 前言

1.1 编写目的

为了让多媒体开发人员熟悉 Android 产品的多媒体框架，实现多媒体功能定制和简单调试。

1.2 适用范围

本模块说明适用于全志科技 Android 系统产品, 其他 Android 版本系统也可参考。

1.3 相关人员

- 多媒体开发工程师
- 技术支持工程师

1.4 相关术语

表 1-1: 术语介绍

术语	解释说明
CedarX	全志多媒体中间件，通过 AwPlayer 对接到 Android 系统中
CedarC	全志多媒体视频编解码驱动以及 openMAX IL 层实现。
Stream	CedarX 对多媒体协议类型访问的统一接口，支持的媒体协议类型包括：本地文件、文件描述符、RTSP、HTTP 等。
Parser	CedarX 对封装格式的解析的统一接口，支持的媒体封装类型包括：ASF、TS、AVI 等。
Demuxer	CedarX 对媒体的 Stream 和 Parser 解析的统一接口。
Decoder	音频，视频，字幕的解码器。
Render	音频，视频，字幕渲染。
Codec2	谷歌推出的新的对接 MediaCodec 的中间件，指在取代 ACodec 与 OpenMAX

2 多媒体模块框架和支持列表

2.1 Android 多媒体框架

Android 原生 MediaPlayer 多媒体框架，以及全志科技的移植框架图，下图中 CedarX 是全志多媒体中间件。

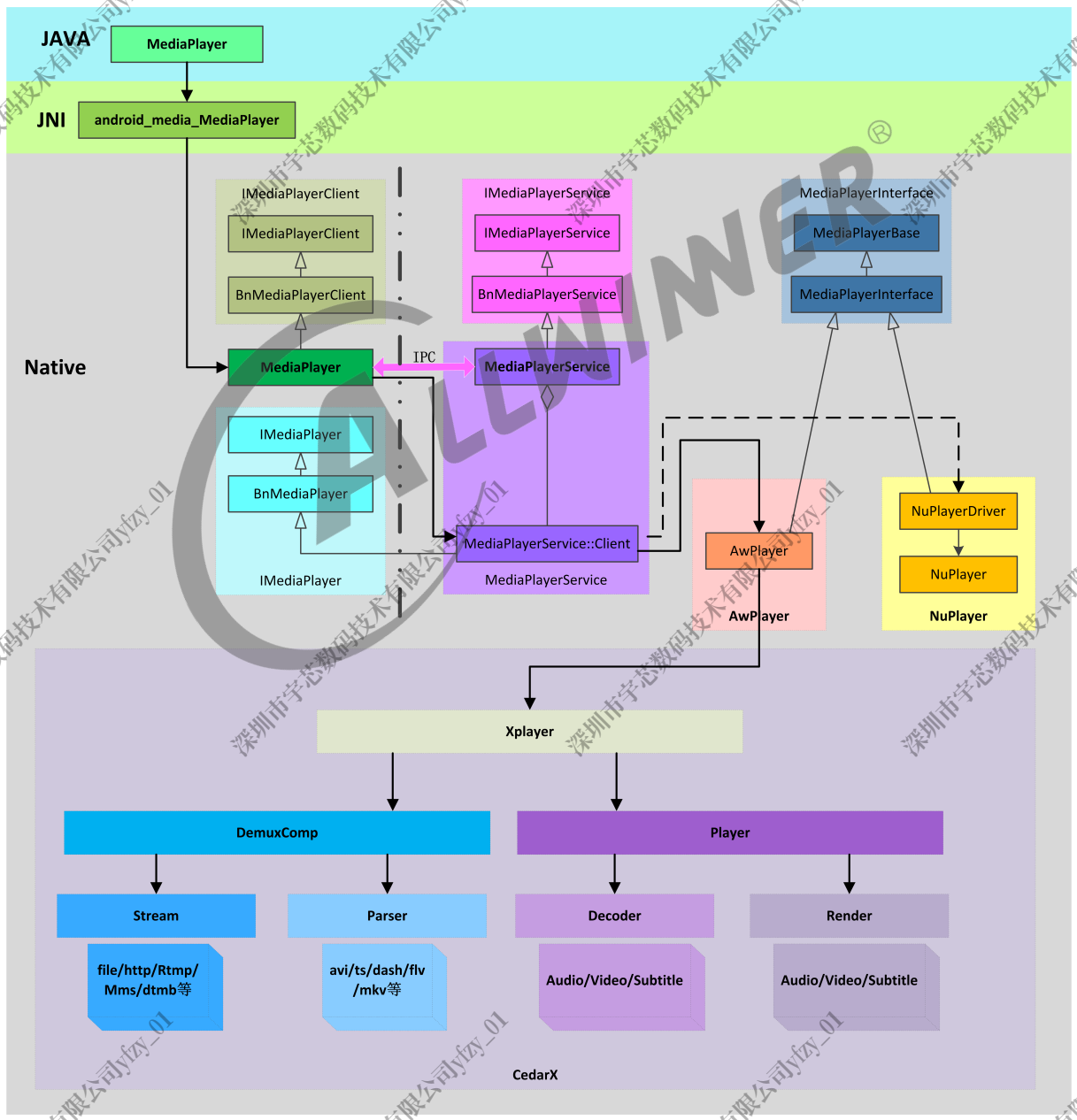

图 2-1: Android CedarX flows

Android 原生 MediaCodec 多媒体框架，以及全志科技的移植框架图，我司 Android Codec2 框架如下图所示。

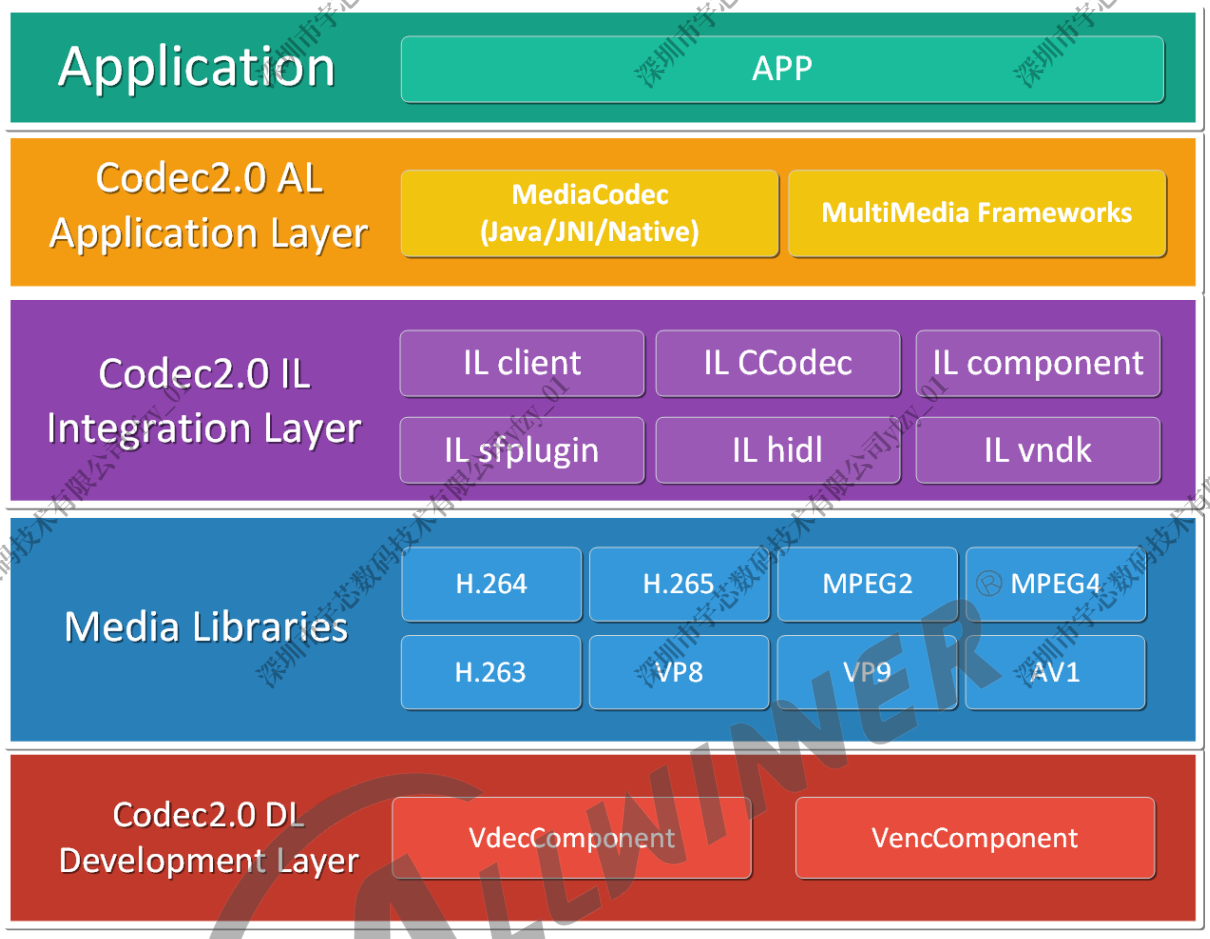

图 2-2: Codec2 架构图

进一步整合，有如下的简化框架图

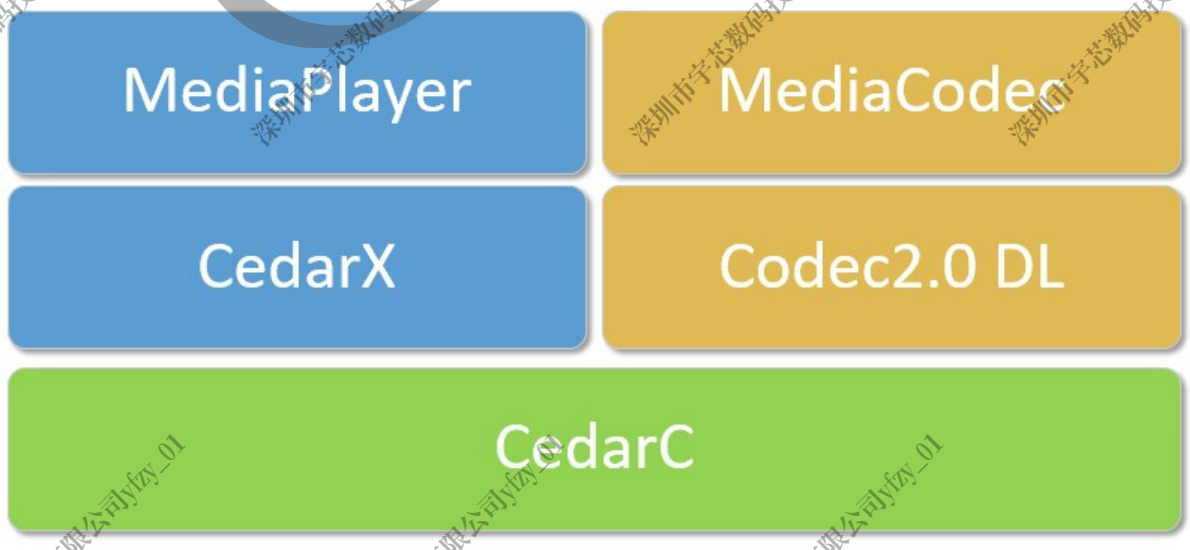

图 2-3: 多媒体简化框架图

2.2 代码结构

1. Android 多媒体模块，java 层和 jni 层代码目录:

```
android/frameworks/base/media
├── aidl
├── java
├── jni
├── lib
├── mca
├── native
├── packages
└── tests
```

2. Android 多媒体模块，Native 层代码目录:

```
android/frameworks/av/media
├── audioaidlconversion
├── audioserver
├── codec2
├── common_time
├── img_utils
├── janitors
├── libaaudio
├── libaudioclient
├── libaudiofoundation
├── libaudiohal
├── libaudioprocessing
├── libaudiousecasevalidation
├── libcpustats
├── libdatasource
├── libeffects
├── liberror
├── libheadtracking
├── libheif
├── libmedia
├── libmediahelper
├── libmediametrics
├── libmediaplayerservice
├── libnbaio
├── libnblog
├── libshmem
├── libstagefright
├── mediaserver
├── module
├── mtp
├── ndk
├── tests
└── utils
```

3. CedarX 多媒体中间件目录:

```
libcedarx
├── android_adapter
│   ├── awplayer
│   ├── base
│   ├── iptv
│   ├── metadataretriever
│   └── output
├── awrecorder
├── config
├── demo
├── external
├── libcore
│   ├── base
│   ├── common
│   ├── muxer
│   ├── parser
│   ├── playback
│   └── stream
├── xmetadataretriever
└── xplayer
```

4. CedarC 多媒体编解码库目录按版本区分，分为 v1.3.0 和 v2.0.0，不同 IC 使用的版本有差异:

```
libcedarc
├── v1.3.0
│   ├── base
│   ├── conf
│   ├── config
│   ├── demo
│   ├── docs
│   ├── include
│   ├── library
│   ├── memory
│   ├── openmax
│   ├── vdecoder
│   └── vencoder
└── v2.0.0
    ├── base
    ├── conf
    ├── config
    ├── demo
    ├── docs
    ├── include
    ├── library
    ├── memory
    ├── openmax
    ├── vdecoder
    └── vencoder
```

5. Codec2 DL 实现的目录:

```
hardware/aw/libcodec2
├── aidl
├── components
│   ├── decoders
│   └── encoders
├── config
```

```
├── plugins
│   └── widevine_plugins
├── services
└── vndk
```

2.2.1 CedarX 与 CedarC 代码路径

从 Android15（V）开始，CedarX 和 CedarC 的代码路径做了修改，如下是 15 和以前 Android 版本的代码路径区别

```
Android15(V)以及以后的版本
  hardware/aw/media/
   ├── libcedarc
   │   ├── v1.3.0
   │   └── v2.0.0
   └── libcedarx

Android14(U)
  frameworks/av/media
   ├── libcedarc
   ├── libcedarc_v2.0.0
   └── libcedarx

Android13(T)以及更早以前的版本
  frameworks/av/media
   ├── libcedarc
   └── libcedarx
```

2.2.2 CedarC 版本说明

根据 IC 的不同，使用的 CedarC 版本有差异。因此 Android15 和 Android14 上对 CedarC 分成了 2 个目录，分别存放 2 个版本的 CedarC 源码和闭源库。需要特别说明的是，Android14 上 libcedarc 就是指 v1.3.0 版本的源码，libcedarc_v2.0.0 就是指 v2.0.0 版本的源码。

```
Android15(V)
  hardware/aw/media/libcedarc
   ├── v1.3.0
   └── v2.0.0

Android14(U)的版本
  frameworks/av/media
   ├── libcedarc
   └── libcedarc_v2.0.0
```

表 2-1: IC 与 CedarC 版本对应表

IC	CedarC 版本
A523/A133/T527	v1.3.0
A733/A537/A333/T736/TV323	v2.0.0

2.2.3 GRF 与多媒体

关于 GRF 的详细描述，可以参考《Android GRF 编译指南》，这里不做重点介绍。需要关注的是 GRF 下，多媒体代码如何修改和生效。

2.2.3.1 多媒体代码所属分区

libcedarx：属于 system 分区
libcedarc：system 和 vendor 分区均包含
libcodec2：属于 vendor 分区

2.2.3.2 GRF 方案下多媒体开发方式

对于非 GRF 方案（Android13 及其 Android13 以下的版本），system 分区和 vendor 分区都是来自同一个 Android SDK，多媒体代码按正常方式提交即可。但对于 GRF 方案，也就是双 SDK 的开发需要特别注意，vendor 来自于 [Frozen SDK]，framework 来自 [新版本 Android SDK]。以 Android14+Android15 为例，[Frozen SDK] 就是 Android14，[新版本 SDK] 就是 Android15，这时候多媒体的开发方式如下

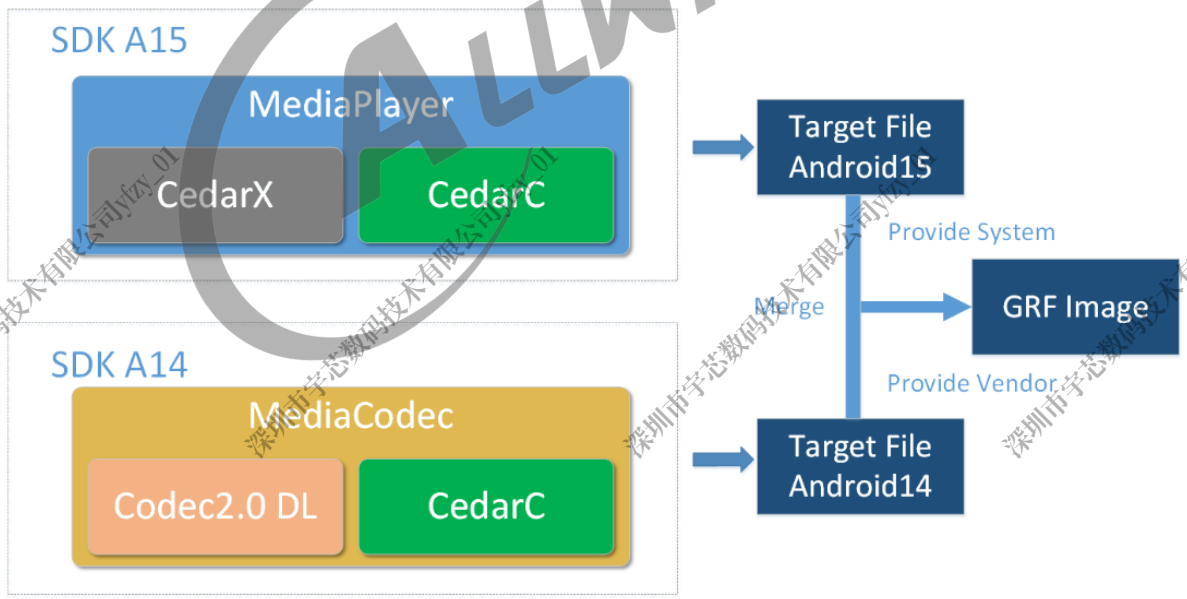

图 2-4: GRF 下多媒体开发方式

- 修改 MediaPlayer 的多媒体代码，需要修改 Android 15 SDK 下的 CedarX，或者根据 IC 平台的不同，修改对应版本的 CedarC
  - [Android 15 SDK]/hardware/aw/media/libcedarx
  - [Android 15 SDK]/hardware/aw/media/libcedarc

- 修改 MediaCodec 的 Codec2.0 DL 的多媒体代码，需要修改 Android 14 SDK 下的 libcodec2，或者根据 IC 平台的不同，修改对应版本的 CedarC
  - [Android 14 SDK]/hardware/aw/libcodec2
  - [Android 14 SDK]/frameworks/av/media/libcedarc_v2.0.0
  - [Android 14 SDK]/frameworks/av/media/libcedarc
- 多媒体代码同步方式
  - CedarC：推荐对 Android 15 和 Android 14 都做更新
  - CedarX：Android15 的修改，Android14 可以选择是否同步
  - libcodec2：Android14 的修改，Android15 可以选择是否同步

3 多媒体支持说明

3.1 视频容器格式支持说明

默认支持如下封装格式：asf, avi, flv, f4v, mkv, mov, mp4/m4v, vob, mpg, pmp, ts/tp, m2ts, mts, webm, 3GP。

3.2 视频解码格式支持说明

一般默认支持如下视频格式：H.265 MP/L5.2、H.264 Baseline/HP/MP Level5.1、MPEG1/MPEG2/MPEG4、H.263 Baseline 等。

3.3 音频容器格式支持说明

默认支持如下音频封装格式: aac, aiff, amr, ape, atrac, caf, dsd, flac, g729, mp3, ogg, wav 等。

3.4 音频解码格式支持说明

默认支持如下音频解码格式: AMR, MP1/MP2/MP3, OGG, WAV, AAC, APE, FLAC, DSD, G729, ALAC 等。

3.5 流媒体协议支持说明

默认支持如下流媒体协议：http、https、rtmp、hls。

3.6 多屏互动支持说明

A 系列芯片支持 Miracast 多屏互动协议，包括发送端和接收端。T/H/TV 系列芯片仅支持接收端。

Miracast 是由 Wi-Fi 联盟于 2012 年所制定，以 Wi-Fi 直连为基础的无线显示标准。支持此标准的设备可通过无线方式分享视频画面，例如手机可通过 Miracast 将影片或照片直接在电视或其他装置播放而无需受到连接线缆长度的影响。

3.7 支持第三方使用 MediaCodec 的播放器

默认支持使用 Android 原生标准接口 MediaCodec 的视频播放 apk，比如 Kodi、腾讯视频、爱奇艺等。

3.8 支持多音轨切换

默认支持多音轨切换。

3.9 支持的其他播放特性

- H.265 4K@60fps@10bit
- H.264 4K@30fps@8bit

详情请查看多媒体支持列表

4 模块配置使用说明

4.1 MediaCodec 支持的硬件编解码格式

使用系统命令进行确认，命令如下：

```
dumpsys media.player
```

通过上面命令，可以查看当前系统下 MediaCodec 支持的所有编解码组件信息，通过过滤关键字“c2.allwinner”，可以看到我司支持的硬件编解码组件信息

4.2 最后一帧显示黑屏配置说明

最后一帧显示黑屏是指在切台场景，切换中屏幕保持黑屏。修改对应方案的 CedarX 配置文件：hardware/aw/media/libcedarx/config/cedarx_config.go

```
var xxx_cflags = []string {
  "-DCONF_SEND_BLACK_FRAME_TO_GPU",
    ...
```

此功能配置默认关闭。

4.3 开机视频、动画使用说明

4.3.1 开机视频使用说明

将视频命名为 bootanimation.mp4，存放位置有三个：

- /system/media/bootanimation.mp4
- /data/local/bootanim/bootanimation.mp4
- /product/media/bootanimation.mp4

需要设置系统属性 persist.sys.bootanim.video_enable 为 1，才能使能开机视频，默认不开启。
如需使能，可以在方案配置文件 [device/softwinner/[platform_name]/] 中设置。
使能属性之后，系统启动会优先从/system/media/检测视频文件，如果没有则从/data/local/bootanim/下获取，再没有才会检测/product/media/。如果三个路径都没有，则使用默认启动开机动画。

4.3.2 开机动画使用说明

1. 文件存放位置

开机画面都是以附件 bootanimation.zip 的形式存放在机子中。存放位置有两个：

- /system/media/bootanimation.zip
- /data/local/bootanimation.zip

读取顺序：机子开机画面或视频，优先读取/data/local/bootanimation.zip，如果发现这个 zip 包没有或者解压有问题或者包中的文件有异常或损坏，则使用/system/media/bootanimation.zip 作为开机画面或开机视频展现。

2. 开机画面的内部结构

图 4-1: bootanim01

文件夹中存放图片，part0 内容

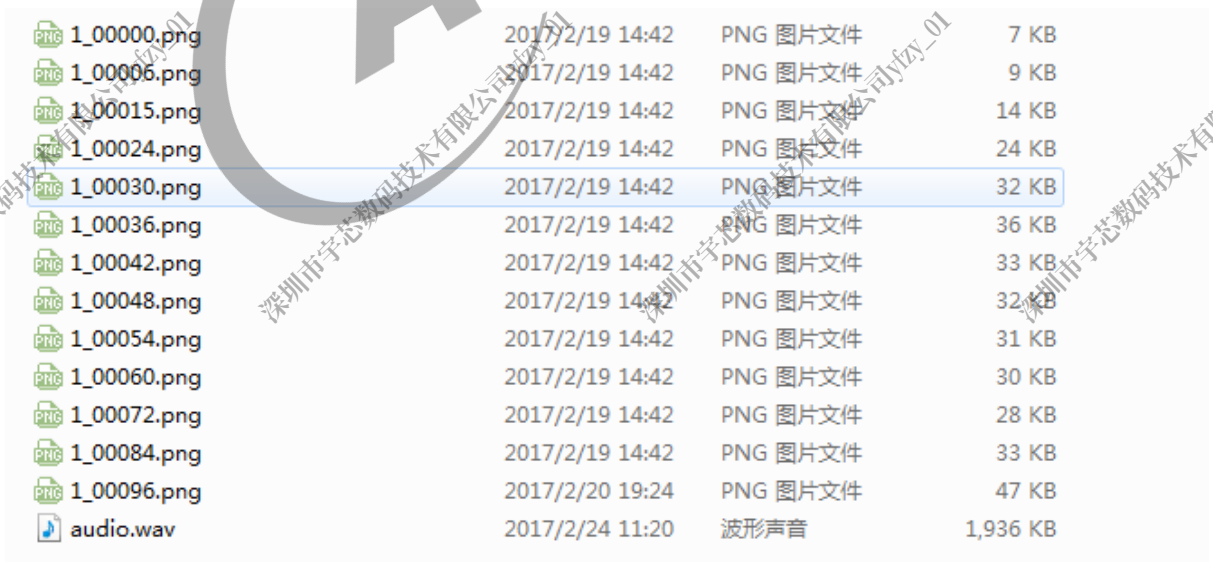

图 4-2: bootanim02

/desc.txt 文件及内容分析：

```
840 840 12
p 1 0 part0
p 0 0 part1
```

图 4-3: bootanim03

840 840 是指前面文件夹里 png 的分辨率，12 是指每秒播放帧数；p 是标识符，1 0 两个数字分别指循环次数和阶段间隔时间，0 0 就代表循环播放；part0 就是文件夹的名字，文件夹的名称和存放图片的目录名一致，设计结构：上图/desc.txt 文件设置为：part0 播放一次，间隔 5 秒，part1 循环播放。

4.3.3 客户开机动画、开机视频定制使用说明

设置或者更新开机动画视频时，客户需要自行实现开机动画视频文件下载逻辑（开机动画 bootanimation.zip 或开机视频 bootanimation.mp4），把文件存放到制定目录即可。如果已经有开机动画视频文件，覆盖即可。

4.4 开机音乐使用说明

4.4.1 开机音乐使用说明

开机音乐代码位于 frameworks/base/cmds/bootanimation 下，会将 bootanimation.zip 中打包的 wav 音频直接播放出来。bootanimation.zip 在 sdk 中的存放目录是 device/softwinner/[platform_name]/，在小机的存放目录是/system/media/。

4.4.2 配置 audio_conf

在 frameworks/base/cmds/bootanimation 中的 BootAnimation.cpp 中, 预留了 audioplay 的服务, 可以直接播放 wav 格式的数据, 前提是配置好 audio_conf.txt。配置文件 audio_conf.txt 同样打包在 bootanimation.zip 中，该文件会配置具体音频路由通路, 包含喇叭和耳机通路。audio_conf.txt 如下所示，其中 mixer 表示喇叭配置，headset mixer 表示耳机的配置。

```
card=0
device=0
period_size=2048
period_count=4
#Input_Event or Switch_State
Headset detection=Switch_State
switch_path=/sys/module/snd_soc_sunxi_component_jack/parameters/jack_state
single_volume_control=1
speaker_headphones_out=0
```

```
#mixer "Headphone Volume"=0
#mixer "LINEOUT volume"=26
mixer "LINEOUT Gain"=15
mixer "LINEOUT Output Select"=single
mixer "LINEOUTL Switch"=1
mixer "LINEOUTR Switch"=1
mixer "SPK Switch"=1
#headset mixer "Headphone Volume"=0
#headset mixer "LINEOUT volume"=26
headset mixer "HPOUT Gain"=4
headset mixer "SPK Switch"=0
headset mixer "HPOUT Switch"=1
```

5 调试说明

5.1 如何修改多媒体中间件打印等级

针对多媒体音视频播放问题，客户可以修改 CedarX 配置文件打印等级，增加打印：hardware/aw/media/libcedarx/conf/[具体项目名称]_cedarx.conf

```
# log will output if level >= log_level
#VERBOSE = 2,
#DEBUG = 3,
#INFO = 4,
#WARNING = 5,
#ERROR = 6,
log_level = 3
```

5.2 如何简单定位 MediaPlayer 播放问题

遇到播放问题，可以先将 log_level 改为 VERBOSE，反馈给 FAE。

5.2.1 不能播放问题的定位

第一步：播放器有没有创建成功

查看 adb logcat，搜集 CedarX 打印

```
D/awplayer: <LogVersionInfo:34>:
D/awplayer: >>>>>>>>>>>>>>>>>>>>>>>>>>> CedarX <<<<<<<<<<<<<<<<<<<<<<<<<<
D awplayer: tag  : CedarX-2.8.0
D awplayer: branch: master
D awplayer: commit: 967535b8ff6a073cb4f38e85a4ae5fa6008014d8
D awplayer: date  : Mon, 15 May 2017 01:30:22 +0000 (09:30 +0800)
D awplayer: author:
D awplayer: ----------------------------------------------------------------------
```

查看 adb logcat，搜集 CedarC 打印

```
I cedarc  : >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Cedar Codec <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
I cedarc  : tag  : CedarC-v1.3.0
I cedarc  : branch: stable_v1.3.0_common
I cedarc  : commit: dedc20c57e6626caf25e0fa3042be4c8faf13e0f
I cedarc  : date  : Fri Nov 13 10:19:03 2020 +0800
I cedarc  : author: jenkins8080
I cedarc  : patch :
```

```
I cedarc : ------------------------------------------------------------------
```

或者

```
D cedarc : >>>>>>>>>>>>>>>>>>>>>>>>>>>> Cedar Codec <<<<<<<<<<<<<<<<<<<<<<<<<<<<
D cedarc : tag  : CedarC-v2.0.0
D cedarc : branch: stable_v2.0.0_common
D cedarc : commit: d25dcc3e6144f9ab143f34291b5b8705273cacb3
D cedarc : date  : Tue Jun 11 09:46:52 2024 +0800
D cedarc : author: jenkins8080
D cedarc : patch :
D cedarc : ------------------------------------------------------------------
```

这里的目的在于确定对应平台使用的 cedarc 版本。

第二步：parser 有没有解析成功

```
D/demuxComponent( 1515): <PrintMediaInfo:426>: *****PrintMediaInfo begin*****
D/demuxComponent( 1515): <PrintMediaInfo:440>: fileSize = 47174592, bSeekable = 1,
duration = 202996, audioNum = 1, videoNum = 1, subtitleNum = 0
D/demuxComponent( 1515): <PrintMediaInfo:458>: **Video[0]** eCodecFormat = 0x115,
nWidth = 720, nHeight = 480, nFrameRate = 24000,
nFrameDuration = 0, bIs3DStream = 0
D/demuxComponent( 1515): <PrintMediaInfo:476>: ***Audio[0]*** eCodecFormat = 0x3,
eSubCodecFormat = 0x0, nChannelNum = 2,
nBitsPerSample = 0, nSampleRate = 44100
D/demuxComponent( 1515): <PrintMediaInfo:492>: ****PrintMediaInfo end*****
```

没有如上类似的打印，说明 parser 没有解析成功。如果需要的话，搜集 CedarC 和 CedarX 打印，反馈给 FAE 定位问题。

5.2.2 播放异常定位

播放的处理流程如下：

```
          ——> AudioDecoder ——> AudioRender
Demuxer ——|
          ——> VideoDecoder ——> VideoRender
```

播放异常的分析流程与之相反：Render —> Decoder —> Demuxer

1、查看 Audio/Video Render 是否正常

hardware/aw/media/libcedarx/libcore/playback/player.c 文件

```
diff --git a/media/libcedarx/libcore/playback/player.c b/media/libcedarx/libcore/playback/player.c
index 0e0baa0..304ae85 100644
--- a/media/libcedarx/libcore/playback/player.c
+++ b/media/libcedarx/libcore/playback/player.c
@@ -3415,7 +3415,7 @@ static int CallbackProcess(void* pSelf, int eMessageId, void* param)
         nCurTime  = p->pAvTimer->GetTime(p->pAvTimer);
         nTimeDiff = nVideoPts - nCurTime;

-        logv("notify video pts = %" PRId64 " ms, curTime = %" PRId64 " ms, diff = %"
+        logd("notify video pts = %" PRId64 " ms, curTime = %" PRId64 " ms, diff = %"
```

```
            PRId64 " ms", nVideoPts/1000, nCurTime/1000, nTimeDiff/1000);

          if (p->onResetNotSync)
@@ -3614,7 +3614,7 @@ static int CallbackProcess(void* pSelf, int eMessageId, void* param)

          }

-         logv("notify audio pts %" PRId64 " ms, curTime %" PRId64 " ms, diff %" PRId64
+         logd("notify audio pts %" PRId64 " ms, curTime %" PRId64 " ms, diff %" PRId64
            " ms, cacheTime %" PRId64 " ms", nAudioPts/1000, nCurTime/1000,
            nTimeDiff/1000, nCachedTimeInSoundDevice/1000);
```

分析打印，看看 Audio 和 video 的 pts 是否正常。

2、查看 Audio/Video Decoder 是否正常

修改 hardware/aw/media/libcedarx/libcore/playback/audioDecComponent.c 文件以及 hardware/aw/media/libcedarx/libcore/playback/videoDecComponent.c 文件。

```
diff --git a/media/libcedarx/libcore/playback/audioDecComponent.c b/media/libcedarx/libcore/playback/
    audioDecComponent.c
index da72249..ab00299 100755
--- a/media/libcedarx/libcore/playback/audioDecComponent.c
+++ b/media/libcedarx/libcore/playback/audioDecComponent.c
@@ -1312,7 +1312,7 @@ static void doDecode(AwMessage *msg, void *arg)
                    &p->pStreamInfoArr[p->nStreamSelected],
                    pOutputBuf,
                    &nPcmDataLen);
-   logv("AdecDecode, ret = %d",ret);
+   logd("AdecDecode, ret = %d",ret);
    if(ret == ERR_AUDIO_DEC_NONE)
    {
      if(p->pStreamInfoArr[p->nStreamSelected].nSampleRate != p->bsInfo.out_samplerate ||
diff --git a/media/libcedarx/libcore/playback/videoDecComponent.c b/media/libcedarx/libcore/playback/
    videoDecComponent.c
index 8b050ad..6bec2e4 100755
--- a/media/libcedarx/libcore/playback/videoDecComponent.c
+++ b/media/libcedarx/libcore/playback/videoDecComponent.c
@@ -820,7 +820,7 @@ static void doDecode(AwMessage *msg, void *arg)
                    p->bConfigDropDelayFrames,
                    nCurTime);

-   logv("DecodeVideoStream return = %d, p->bCrashFlag(%d)", ret, p->bCrashFlag);
+   logd("DecodeVideoStream return = %d, p->bCrashFlag(%d)", ret, p->bCrashFlag);

    if(ret == VDECODE_RESULT_NO_BITSTREAM)
    {
```

分析解码库的返回值意义：

返回值名称	对应数值	描述
VDECODE_RESULT_FRAME_DECODED	1	解码成功，输出了一帧图像
VDECODE_RESULT_CONTINUE	2	码流被解码，但没有图像输出，需继续解码
VDECODE_RESULT_KEYFRAME_DECODED	3	解码成功，输出了一帧关键帧图像
VDECODE_RESULT_NO_FRAME_BUFFER	4	当前无法获取到图像 Buffer
VDECODE_RESULT_NO_BITSTREAM	5	当前无法获取到码流数据

返回值名称	对应数值	描述
VDECODE_RESULT_RESOLUTION_CHANGE	6	视频分辨率发生变化
VDECODE_RESULT_UNSUPPORTED	-1	不能支持的格式或申请内存失败，无法继续解码

3、查看 Demuxer 给解码库 submit Audio/Video 数据是否正常

hardware/aw/media/libcedarx/libcore/playback/player.c 文件

```
diff --git a/media/libcedarx/libcore/playback/player.c b/media/libcedarx/libcore/playback/player.c
index 0e0baa0..7a2416c 100644
--- a/media/libcedarx/libcore/playback/player.c
+++ b/media/libcedarx/libcore/playback/player.c
@@ -1365,7 +1365,7 @@ int PlayerSubmitStreamData(Player*          pl,

    p = (PlayerContext*)pl;

-   logv("submit stream data, eMediaType = %d", eMediaType);
+   logd("submit stream data, eMediaType = %d", eMediaType);

    if(p->eStatus == PLAYER_STATUS_STOPPED)
```

根据这句打印，判断 Demuxer 后的音视频数据是否送给解码库，每笔数据的 PTS。其中 eMediaType 含义如下。

```
enum EMEDIATYPE
{
    MEDIA_TYPE_VIDEO = 0,
    MEDIA_TYPE_AUDIO,
    MEDIA_TYPE_SUBTITLE,
    MEDIA_TYPE_METADATA
};
```

经过以上三步分析，可以初步定位出播放异常问题属于 Render —> Decoder —> Demuxer （Audio/Video/Subtitle）哪个模块。

5.3 如何保存解码前的码流和解码后的图片

5.3.1 方法一修改代码

hardware/aw/media/libcedarc/vdecoder/vdecoder.c 文件

```
diff --git a/media/libcedarc/vdecoder/vdecoder.c b/media/libcedarc/vdecoder/vdecoder.c
index 47c1cdd..9346fdc 100755
--- a/media/libcedarc/vdecoder/vdecoder.c
+++ b/media/libcedarc/vdecoder/vdecoder.c
@@ -32,8 +32,8 @@
 #include <sys/ioctl.h>
 #include <fcntl.h>

-#define DEBUG_SAVE_BITSTREAM    (0)
```

```
-#define DEBUG_SAVE_PICTURE    (0)
+#define DEBUG_SAVE_BITSTREAM   (1) //* 保存解码前的码流
+#define DEBUG_SAVE_PICTURE    (1) //* 保存解码出来的图片

 /* show decoder speed */
 #define AW_VDECODER_SPEED_INFO (0)
```

5.3.2 方法二修改配置文件

修改小机端的/vendor/etc/cedarc.conf

```
########### paramter ############
[paramter]
# save bitstream in vdecoder.c
vdecoder_save_bitstream = 0
vdecoder_save_bitstream_path = /data/camera/bs.dat

# save picture from start_num to start_num + total_num
vdecoder_save_picture_en = 0
vdecoder_save_picture_start_num = 0
vdecoder_save_picture_total_num = 10
vdecoder_save_picture_path = /data/camera/pic.dat
```

5.3.2.1 使能保存解码前码流的功能

1. vdecoder_save_bitstream 默认为 0，表示关闭保存解码前码流的功能；配置为 1 就可以使能保存解码前的码流的功能，保存路径为/data/camera/
2. 在配置 vdecoder_save_bitstream 为 1 之后，需要关闭系统的 SELinux，命令为 “setenforce 0”，并在/data 目录下手动创建 camera 目录，并赋予 “777” 的权限
3. 之后退出正在播放视频的程序，或者重启 Mediaplayer 或者 MediaCodec 即可让修改的配置生效

5.3.2.2 使能保存解码出来图片的功能

1. vdecoder_save_picture_en 默认为 0，表示关闭保存解码出来图片的功能；配置为 1 就可以使能保存解码出来图片的功能，保存路径为/data/camera/
2. vdecoder_save_picture_start_num 和 vdecoder_save_picture_total_num 分别表示要开始保存解码出来图片的起始位置，以及保存图片的数量
3. 在配置 vdecoder_save_picture_en 为 1 之后，需要关闭系统的 SELinux，命令为 “setenforce 0”，并在/data 目录下手动创建 camera 目录，并赋予 “777” 的权限
4. 之后退出正在播放视频的程序，或者重启 Mediaplayer 或者 MediaCodec 即可让修改的配置生效
5. 解码出来的图片，根据格式使用 YUV 或 NV 工具查看。

5.4 如何处理解码器内部问题

5.4.1 方法一修改代码

打开保存码流文件 special.awsp 的调试宏，把保存的数据反馈给 FAE。注意：如果保存失败，请查看当前机子是否有"/data/camera/"目录，权限是否正确。

```
diff --git a/media/libcedarc/vdecoder/vdecoder.c b/media/libcedarc/vdecoder/vdecoder.c
index 904d78d..6a09462 100644
--- a/media/libcedarc/vdecoder/vdecoder.c
+++ b/media/libcedarc/vdecoder/vdecoder.c
@@ -48,7 +48,7 @@
 #define DEBUG_SAVE_FRAME_TIME   (0)
 #define DEBUG_MAX_FRAME_IN_LIST 16

-#define DEBUG_MAKE_SPECIAL_STREAM (0)
+#define DEBUG_MAKE_SPECIAL_STREAM (1)
 #define SPECIAL_STREAM_FILE "/data/camera/special.awsp"
 #if DEBUG_SAVE_BITSTREAM
 const char* fpStreamPath = "/data/camera/bitstream.dat";
```

5.4.2 方法二修改配置文件

修改小机端的/vendor/etc/cedarc.conf

```
# save special bitstream in vdecoder.c
vdecoder_save_special_bitstream = 0
vdecoder_save_special_bitstream_path = /data/camera/spec.awsp
```

1. vdecoder_save_special_bitstream 默认为 0，表示关闭保存 awsp 数据的功能；配置为 1 就可以使能保存 awsp 数据的功能，保存路径为/data/camera/
2. 在配置 vdecoder_save_picture_en 为 1 之后，需要关闭系统的 SELinux，命令为 “setenforce 0”，并在/data 目录下手动创建 camera 目录，并赋予 “777” 的权限
3. 之后退出正在播放视频的程序，或者重启 Mediaplayer 或者 MediaCodec 即可让修改的配置生效

5.5 如何简单定位 MediaCodec 播放问题

1. 确认 MediaCodec 是否使用我司硬件编解码组件，通过过滤打印 “CCodec : allocate” 打印来简单确认 MediaCodec 当前使用的组件名称，如果带有 “c2.allwinner” 关键字说明使用的是我司的组件。
2. 可以通过下面的命令来设置系统属性，动态打开 Codec2 的调试打印，来分析当前硬件组件的运行情况。

```
setprop persist.vendor.codec2.debug [参数]
```

参数可选范围	对应数值	描述
DEBUG_NONE	0	默认状态，不输出调试信息
DEBUG_DUMP_INPUT	1	保存送给解码器的码流数据
DEBUG_DUMP_OUTPUT	2	保存解码出来的图像数据
DEBUG_DUMP_PERFORMANCE	4	查看当前送码流以及解码出图像的性能状态
DEBUG_DUMP_BUFFER_STATUS	8	查看当前组件的 Buffer 管理状态
DEBUG_DUMP_DECODE_STATUS	16	查看当前组件的解码状态

参数可以自由组合。例如想同时查看当前 Codec2 硬件组件的 Buffer 管理状态以及组件的解码状态，就可以把 persist.vendor.codec2.debug 设置为 24

3. 通过 prefetto 来抓取视频播放卡顿，或者卡死时系统的运行状态，命令如下

```
perfetto -o /data/misc/perfetto-traces/trace -t 10s sched gfx view ss video
```

之后把 trace 文件用 prefetto 的官方网站 https://ui.perfetto.dev/打开进行查看。

6 FAQ

6.1 为什么有些音视频不能播放

1、判断封装格式、流媒体协议是否支持。

2、判断音频/视频的 Codec 格式是否支持，有些音频/视频格式属于版权视频，需要查看支持列表看看是否支持。如果需要支持，请咨询 FAE。

3、判断视频 Codec 规格是否支持对应 size 格式播放，比如 Video 格式 H265 4K@30fps。

4、如果平台规格支持，又不能播放，可以尝试使用 5.2 章节或者 5.5 章节的方法定位问题，反馈给 FAE。

著作权声明

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

商标声明





（不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。