Linux CedarX
开发指南

版本号: 1.2
发布日期: 2025.2.22

版本历史

版本号	日期	制/修订人	内容描述
1.0	2024.3.14	AWA1729	初始版本文档
1.1	2024.7.2	AWA1729	3.8.1 章节中新增关于 openwrt 文件系统的多媒体配置文件说明
1.2	2025.2.22	AWA2093	增加多媒体功能验证章节中 h13x 平台新增功能的规格功能说明

目　录

插　图

1 前言

1.1 编写目的

本文档系统介绍多媒体中间件 CedarX，方便播放中间件的开发、调试和后续维护。

1.2 适用平台范围

1. Android/Linux 各版本操作系统平台；
2. 公司沿用 CedarX 播放中间件的各个芯片平台；

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
CedarC	全志多媒体视频编解码驱动
Stream	CedarX 对多媒体协议类型访问的统一接口，提供本地文件和流媒体文件的访问功能，支持的媒体协议类型包括：本地文件、文件描述符、RTSP、HTTP 等。
Container	文件格式容器，例如常见的有 TS、MOV、AVI、MKV、FLV 等封装格式。
Parser	CedarX 对封装格式的解析的统一接口，于解析文件封装格式，从 AVI 等媒体文件中读取音频、视频、字幕等码流数据和其他相关数据的程序模块，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。
Muxer	封装器，负责把编码后的数据封装成一定容器格式的文件。
Decoder	音频，视频，字幕的解码器。

术语	解释说明
Render	音频，视频，字幕渲染。
PTS	Presentation Timestamp，显示时间戳。

2 CedarX 介绍

2.1 功能介绍

CedarX 是一套向应用程序提供媒体播放功能的框架，称之为多媒体播放中间件。CedarX 整合了流媒体库、Container Parser、Video Decoder、Audio Decoder、Subtitle Decoder 等各种多媒体功能库，向应用程序提供简单的程序接口，通过我司视频硬件解码加速以及其他相应的软件技术，使应用程序在实现多媒体播放功能的同时达到高性能、低功耗的要求。
CedarX 提供以下功能：
1. 各种音视频媒体文件的播放
2. 各种流媒体播放
3. 视频预览、媒体信息扫描
4. 蓝光文件夹播放
5. 3D 播放功能
6. 多屏互动播放功能
7. 速率播放、变速播放
8. 去隔行降噪的视频后处理功能

目前 Linux 方案仅支持前 3 项基本功能，后续待进一步扩展。

2.2 源码路径介绍

Android 系统上，CedarX 的代码路径为 [android_sdk]/frameworks/av/media/libcedarx

Linux 系统上，CedarX 的代码路径为 [tina_sdk]/platform/allwinner/multimedia/libcedarx

代码路径可能会有调整，请以最新代码为准

2.3 源码结构介绍

源码结构如图 “cedarx 代码目录结构” 所示，CedarX 的代码存放在 cedarx 目录下。

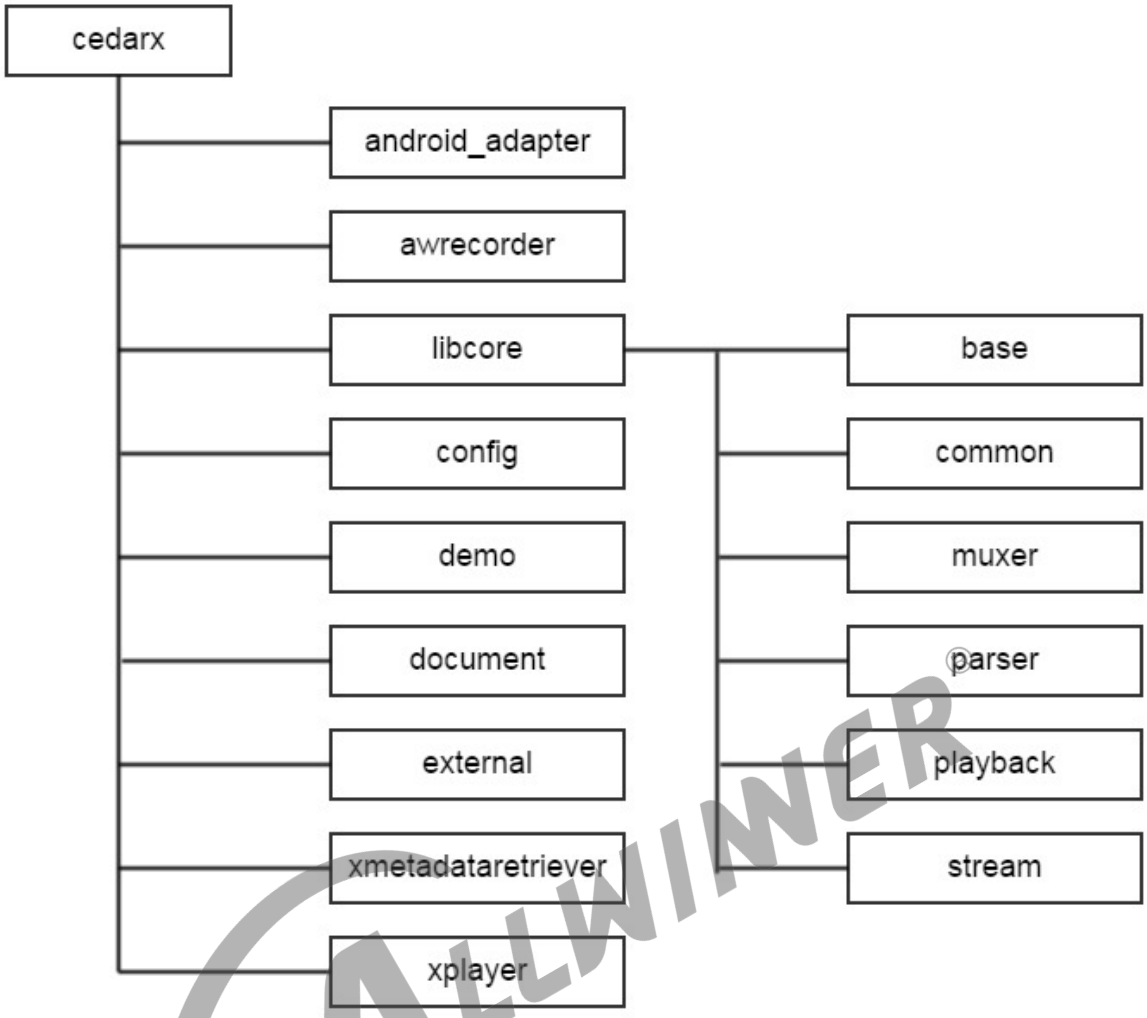

图 2-1: cedarx 代码目录结构

- android_adapter 是 Android 适配层，所有和 Android 平台相关的实现放在此目录。里面主要包括 awplayer、metadataretriever 和 output。
  - awplayer 是对 Android mediaplayer 接口的实现，实现播放器功能；
  - metadataretriever 是对 Android MediaMetadataRetriever 接口的实现，实现媒体扫描功能；
  - output 是和平台相关的输出控制操作，包括视频输出控制、音频输出控制、字幕输出控制和 deinterlace（去隔行）功能的实现。
- awrecorder 用于实现编码录像功能，包括视频编码、音频编码和音视频封装。目前主要用于 Linux 系统。
- libcore 是整个中间件的核心部分，包括 base、common、muxer、parser、playback 和 stream。
  - base 目录实现一些基础功能库，例如消息队列、链表、内存池等;
  - common 目录主要是一些通用功能的实现，例如配置文件解析库等；
  - muxer 目录实现音视频文件封装功能，目前支持 aac，mp3，ts，mov 封装;

  - parser 目录实现各封装文件的解析功能，包括纯音频格式和视频文件封装；
  - playerback 目录实现播放控制、音视频同步的逻辑，包括音频、视频、字幕解码和音频、视频、字幕输出渲染功能；
  - stream 目录提供本地文件和流媒体文件的访问功能，其中流媒体包括多种流媒体协议实现。

- config 目录是 cedarx 的配置文件目录。
- demo 目录下是各主要模块的 demo 程序实现，包括 player、recorder、decoder 等，方便开发使用。
- external 目录下存放闭源的动态链接库，主要包括视频、音频、字幕解码库。
- xmetadataretriever 目录实现媒体扫描逻辑，供 linux 平台直接使用，以及 Android 平台通过 metadataretriever 对其进行调用。
- xplayer 目录实现音视频播放器模块的基本框架，实现播放功能，提供上层应用播放控制的 API，方便上层实现播放器功能。

3 CedarX 架构设计

3.1 CedarX 架构

CedarX 的整体架构如图所示

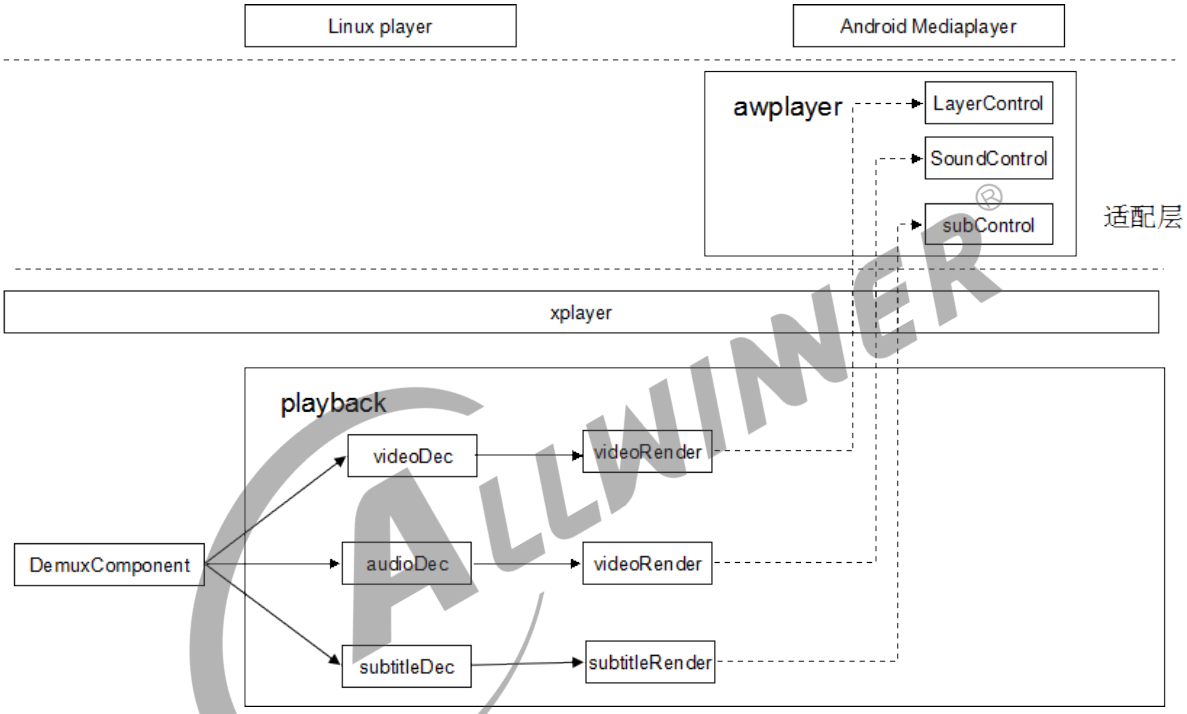

图 3-1: CedarX 架构

1. 对于 Linux 方案，播放器可直接使用 xplayer 层接口获取多媒体播放的能力

2. Android 方案则通过由 xplayer 进一步封装而来的 awplayer 适配层接口进行对接。

3. xplayer 模块对接 Player、Demux、VideoDecode、AudioDecode、SubtitleDecode、VideoRender、AudioRender、SubtitleRender 八个模块。不同的模块负责完成不同的功能:

   - xplayer 模块是实现了音视频播放器的基本框架，提供播放功能，为上层应用提供播放控制的 API，方便上层实现播放器功能，向下屏蔽复杂的运行逻辑。
   - Player 模块控制功能模块协调工作，共同完成多媒体播放任务，称为播控模块。
   - Demux、Video Decode、Audio Decode、Subtitle Decode、Video Render、Audio Render、Subtitle Render 七个模块分别负责某种特定的功能，且具有特定的运行状态，称为功能模块。

3.2 xplayer

xplayer 模块类似于播控中间件的作用，为上层应用提供播放控制的 API，方便上层实现播放器功能，上层可根据需求对接 Android 和 Linux 平台。

3.2.1 xplayer 整体框架

xplayer 模块设计初衷是提供一套跨平台、便于扩展的多媒体播放中间件。因此需要满足以下两点：
- 对上层屏蔽底层播控、流控以及解码等实现细节，提供一套标准接口共应用层调用。
- 平台差异化需求（如音视频输出、字幕输出等）抽象成统一接口由上层实现，底层不需要关心上层是如何实现的。允许不同平台实现不一样。

xplayer 整体框架图如下图所示：

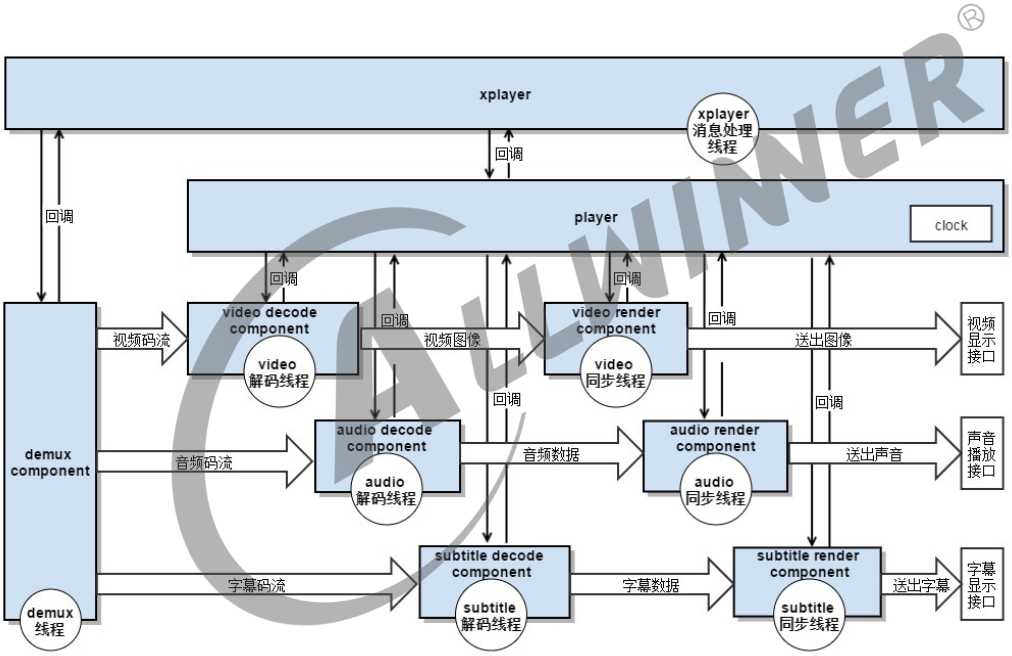

图 3-2: xplayer 整体框架

3.2.2 xplayer 状态机设计

xplayer 模块整体框架、流程以及接口设计借鉴于 android mediaplayer，因此其状态机也与 mediaplayer 类似，如下图所示。

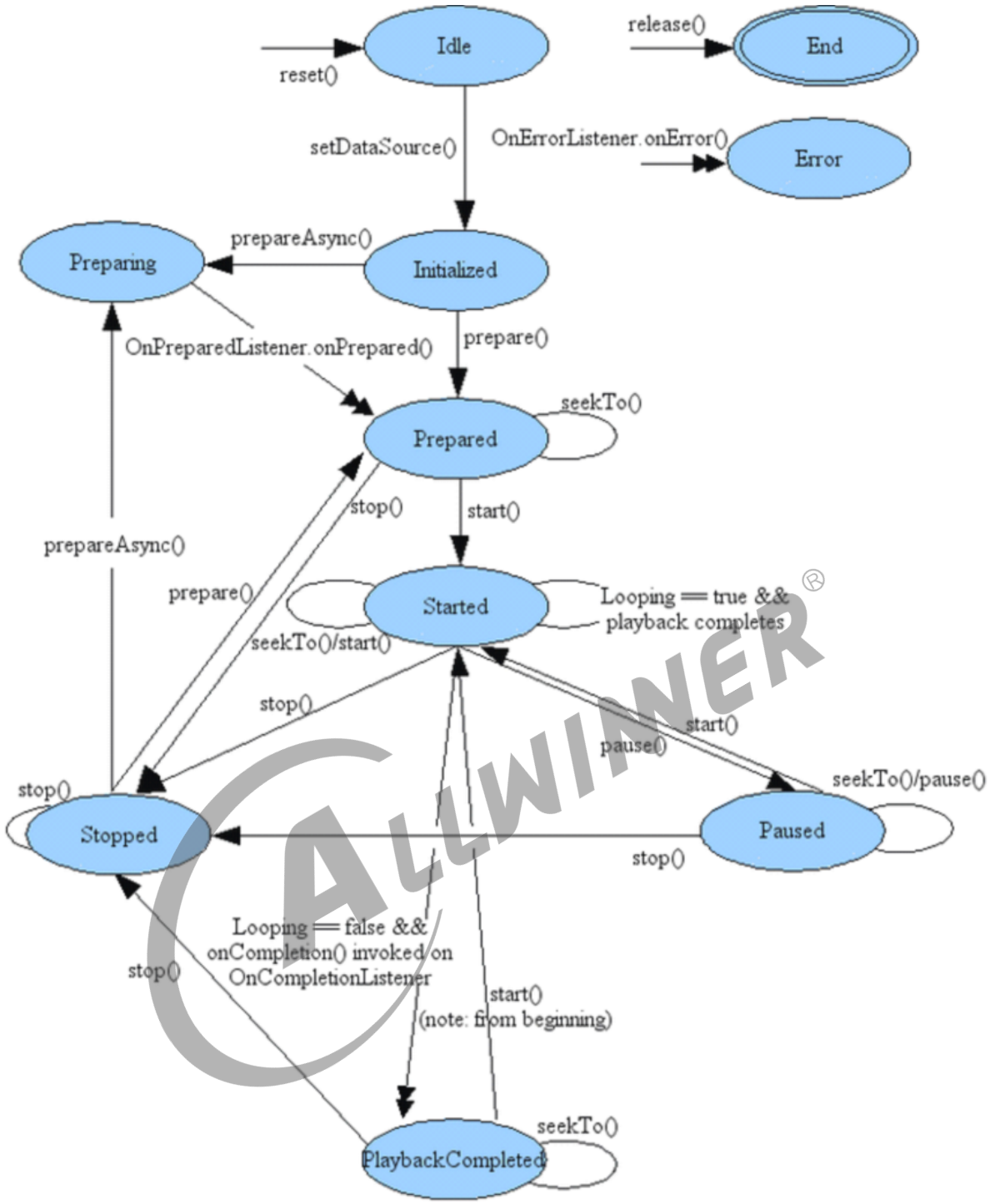

图 3-3: xplayer 状态转换图

xplayer 的状态主要有 Idle、Initilized、Preparing、Prepared、started、stopped、paused、completed、error。

- Idle：创建 xplayer 或 reset 时，处于 idle 状态，此时播放器所需资源还未准备好，不能进入播放。
- Initilized：从 idle 状态调用 setDataSource 设置播放源后，进入 Initilized 状态。此时播放器已经获取到需要播放的音视频数据内容。

- Preparing：调用异步 prepareAsync 操作后会进入该状态。此时播放器解析音视频信息用于创建解码器。如果是播放网络视频源，该操作会比较耗时，因此网络播放一般是调用 prepareAsync 函数而不是 prepare。
- prepared：initialized 状态时调用 prepare 操作或 prepareAsync 完成后，播放器会进入 prepared 状态。此时播放器的准备工作已经完成可以开始播放。
- started：开始播放
- paused：暂停播放
- stopped：停止播放
- completed：播放完成
- error：播放过程中出错，出错时 xplayer 会通过回调方式通知上层。

3.3 功能模块介绍

在 CedarX 中，功能模块（Component）是一个具有内建线程和特定运行状态、对外提供程序接口、负责完成某类功能的程序单元。CedarX 包含 7 个功能模块，如下所示：
Video Decode：负责视频流的解码
Audio Decode：负责音频流的解码
Subtitle Decode：负责字幕流的解码
Video Render：负责视频画面的同步输出
Audio Render: 负责音频数据的同步输出
Subtitle Render: 负责字幕的同步输出
Demux: 负责读取媒体文件，并分发码流到音频、视频和字幕解码器
具体支持的封装格式和解码格式，请查看对应项目提供的用户手册或者 datasheet。

3.4 播控模块

Player（播控模块）控制各个功能模块协调工作，对外提供简单的播放器控制接口。在播放过程中，Player 需要控制计时器，使音视频和字幕能够同步输出。

3.5 输出接口

输出接口是多媒体中间件向系统的显卡、声卡输出媒体内容的途径。不同的 OS 系统通常具有不同的显示系统、声音系统，因此我们需要定义播放器的渲染接口，使 CedarX 能够在 Linux、Android 等系统中方便地移植。
输出接口包括视频显示接口（LayerControl）、声音播放接口（SoundControl）和字幕显示接口（subtitleControl）。这三部分接口分别由 Video Render 模块、Audio Render 模块、Subtitle Render 模块对接。

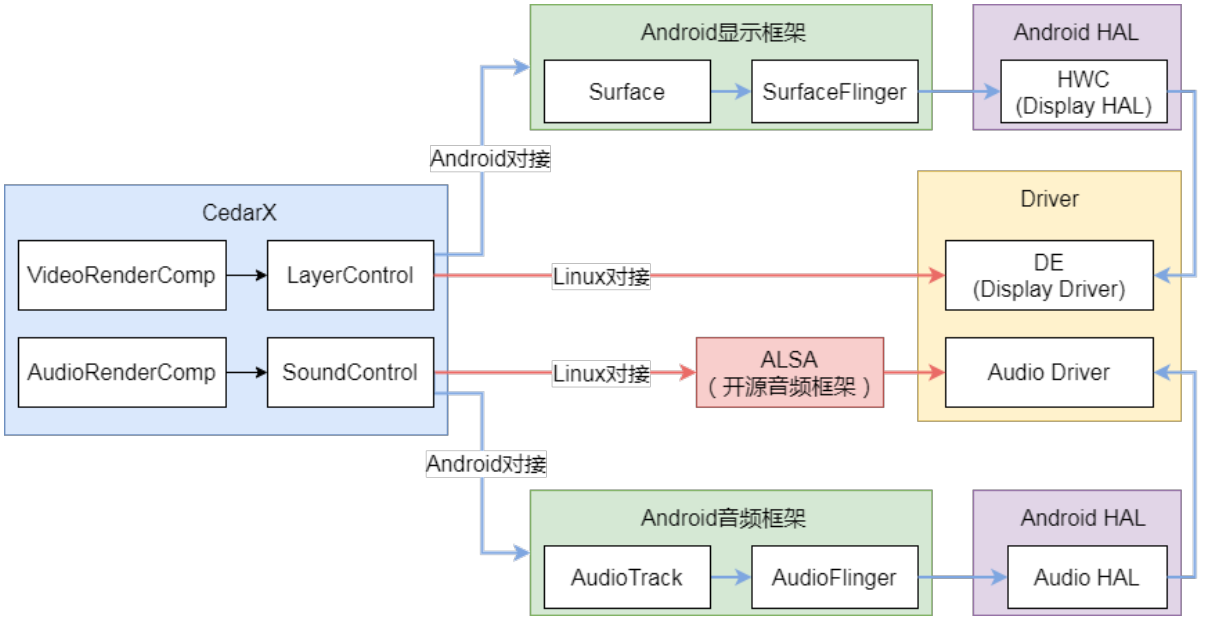

图 3-4: 输出接口系统通路架构图

- 视频显示接口：Video Render 模块向显示框架输出图像的途径。
- 声音播放接口：Audio Render 模块向音频框架输出音频的途径。
- 字幕显示接口：Android、Linux 中不存在专门的字幕显示系统，但在不同的应用中，字幕绘制的实现时有不同。有些应用程序要求字幕由应用程序来绘制，有些则需要播放器底层实现字幕的绘制，需要依据不同的使用场景来决定。

3.6 媒体扫描

MediaMetadataRetriever 类提供了统一的接口用于从一个输入媒体文件中取得帧和元数据。CedarX 的 metadataretriever 目录是对 Android 系统的 MediaMetadataRetriever 接口的实现。而 xmetadataretriever 目录是真正实现媒体扫描逻辑，供 linux 平台直接使用，metadataretriever 是调用 xmetadataretriever 的接口。如下图所示。

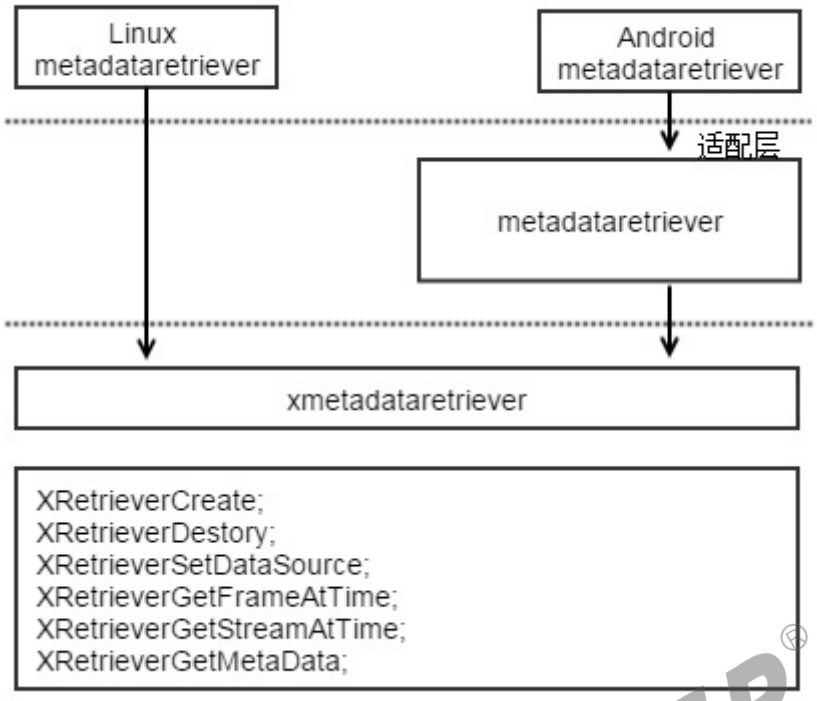

图 3-5: CedarX 媒体扫描框架

3.7 录制模块

awrecorder（录制模块）主要用于编码录像功能，里面分别有视频编码（VideoEncodeComponent）、音频编码（AudioEncodeComponent）和音视频封装（Muxer）三个部分。
视频编码器根据各个 ic 的硬件编码器的支持一般可以支持 h264 和 mjpeg 编码，音频编码支持 aac、mp3 和 pcm，封装器支持的格式有 aac、mov 和 ts。
目前录制模块主要用于对接 Linux 系统。

3.8 配置文件介绍

3.8.1 配置文件概述

CedarX 有 2 类配置文件，一类是编译配置文件，一类是运行配置文件。
- 编译配置文件：通过在指定的配置文件中添加对应的功能配置，实现在编译时使能产品的对应特性。
- 运行配置文件：通过在指定的配置文件中修改对应参数的值，让 CedarX 在运行时读取该配置文件并实现程序行为的动态调整。

不同系统上，关于 CedarX 的配置文件的路径存在差异，按系统来区分，可以分为 Android 系统和

Linux 系统：

- Android 系统
  - 编译配置文件路径：[Android_SDK]/frameworks/av/media/libcedarx/config/cedarx_config.go
  - 运行配置文件路径：[Android_SDK]/frameworks/av/media/libcedarx/conf/。
    - 文件的命名规则按项目平台划分，为 “[TARGET_PRODUCT 的第一个字段]_cedarx.conf”，例如 A 项目的 TARGET_PRODUCT=test_p1，因此其配置文件应命名为：test_cedarx.conf
    - test_cedarx.conf 文件最终在打包固件时会重命名为 cedarx.conf 并拷贝到 system/etc 目录。
- Linux 系统
  - 编译配置文件分为三部分
    - 通用配置：[tina_sdk]/platform/allwinner/multimedia/libcedarx/Makefile.inc
    - buildroot 文件系统差异化配置：[tina_sdk]/buildroot/config/buildroot/allwinner/multimedia/libcedarx/libcedarx.mk
    - openwrt 文件系统差异化配置：[tina_sdk]/openwrt/package/allwinner/multimedia/tina_multimedia/machinfo/[LICHEE_IC]/config.mk
  - 运行配置文件路径
    - buildroot 文件系统：[tina_sdk]/platform/allwinner/multimedia/libcedarx/conf/[CEDARX_TARGET_IC]_cedarx.conf
    - openwrt 文件系统：[tina_sdk]/openwrt/package/allwinner/multimedia/tina_multimedia/machinfo/[LICHEE_IC]/libcedarx/[LICHEE_IC]_linux_cedarx.conf
    - 文件的命名规则按项目平台划分，为 “[CEDARX_TARGET_IC]_cedarx.conf” 或者 “[LICHEE_IC]_linux_cedarx.conf”。例如 A 芯片的 [CEDARX_TARGET_IC] 为 test，因此其配置文件应命名为：test_cedarx.conf
    - test_cedarx.conf 文件最终在打包固件时会重命名为 cedarx.conf 并拷贝到/etc 目录。

不同的产品对应不同的配置文件，新的平台移植 CedarX 时，注意需要添加和修改对应的配置文件。

3.8.2 配置项说明

3.8.2.1 编译配置项

已有的编译配置项如下表所示：

表 3-1: 编译配置项介绍

编译配置项	含义
CONF_NEW_BDMV_STREAM	支持新的蓝光播放方式
CONF_NEW_DISPLAY	新显示框架
CONF_PRODUCT_STB	盒子产品
CONF_PTS_TOSF	传 PTS 到 SurfaceFlinger 用于做显示同步
CONF_H264_4K_P2P	支持 H264 4K p2p 透传
CONF_H265_4K_P2P	支持 H265 4K p2p 透传
CONF_KERNEL_VER	Linux 内核版本
CONF_GPU_MALI	MALI GPU
CONF_KERN_BITWIDE	内核位宽，32/64
CONF_VE_PHY_OFFSET	VE 物理偏移地址
CONF_3D_ENABLE	支持 3D 播放
CONF_DI_300_SUPPORT	支持 Deinterlace300 去隔行模块
CONF_DI_SINGEL_FILE	Deinterlace300 去隔行模块支持场数据输入
CONF_DI_V1XX_MD_60HZ 支持	Deinterlace 110 60HZ 去隔行模式
CONF_USE_IOMMU	支持 IOMMU

3.8.2.2 运行配置

目前已有的运行配置项如下四部分，可分为缓冲参数运行配置、解码外部 buffer 运行配置、其他参数运行配置以及音视频解码器插件配置。

3.8.2.2.1 缓冲参数运行配置项

缓冲参数运行配置项	含义
start_play_cache_video_frame_num	开始播放前需缓冲的视频帧数
start_play_cache_size	开始播放前需缓冲的媒体码流数据大小
cache_buffer_size	缓冲线程缓冲数据 buffer 的大小，点播
cache_buffer_size_live	缓冲线程缓冲数据 buffer 的大小，直播
start_play_cache_time	缓冲一次可以播放的时长
max_start_play_chache_size	可设置的最大的开始播放前需缓冲的媒体码流数据大小
max_cache_buffer_size	可设置的最大的缓冲线程缓冲数据 buffer 的大小
av_sync_duration	视频 pts 大于音频 pts 超过此阈值时，丢弃音频数据

该组配置参数主要是针对在线视频播放的场景而设置的，目前需要调节配置参数的场景主要有以下两个：

- 加快起播速度
  - 通过减小 start_play_cache_video_frame_num、start_play_cache_size 和 start_play_cache_time 等参数的值可以降低起播时需要缓冲的数据量，从而减小起播时的耗时。其中，涉及起播的场景主要有首次播放、seek 操作以及直播切台。
- 优化网络视频播放的内存占用
  - 通过减小 cache_buffer_size、cache_buffer_size_live 和 max_cache_buffer_size 等参数来调整在线播放所缓存数据的最大内存占用。需要说明的是，这个内存并不是播放开始时直接一次性申请，而是在播放过程中随着缓存数据量的增大而逐步申请并累加。

3.8.2.2.2 解码外部 buffer 运行配置

解码外部 buffer 运行配置	含义
pic_4list_num	GPU 显示占用显存个数
pic_4di_num	DeInterlace 占用显存个数
pic_4rotate_num	旋转模块占用显存个数
pic_4smooth_num	为使播放平滑，解码器需要的显存个数

由于视频播放场景下目前是采用解码和显示共用 buffer 的操作，所以解码器申请的 buffer 数量除了与码流本身的特性相关（参考帧个数等）之外，还需要保证图像的后处理和显示平滑性。所以，该组参数配置决定了解码器最终申请的输出图像 buffer 个数。目前需要调节配置参数的场景主要有以下几个：
- 在 Andrioid 平台上，由于显示端的机制要求在播放过程中必须占用一定数量的 frame buffer，因此当显示端占用的个数发生变化时，pic_4list_num 的值要同步进行修改，否则会造成播放过程中 buffer 无法正常轮转，出现画面卡住的问题。
- 在 Linux 平台上，为了保证视频在小内存方案上可以正常进行播放，可通过调节配置参数来达到优化内存的目的。由于 Linux 平台上没有 Android 平台针对显示端要求最大占用个数的限制，所以在保证播放效果的前提下可以通过适当的减少 pic_4list_num 和 pic_4smooth_num 值的方式来降低播放过程中的图像 buffer 内存占用。
- 当出现画面卡顿的现象时，可直接通过增大 pic_4list_num、pic_4smooth_num 的值来初步判定是否是因为 buffer 数量不够而导致的卡顿，然后再做具体的分析。

3.8.2.2.3 输出参数及其他参数运行配置

输出参数及其他参数运行配置	含义
deinterlace_fmt	Deinterlace 模块输入图像的格式
vd_output_fmt	视频解码器输出图像格式

输出参数及其他参数运行配置	含义
gpu_align_bitwidth	GPU 对齐位宽
black_pic_4_SP	切台是否显示黑屏
compensate_vsync	音视频同步补偿时间差
log_level	调试打印日志级别

3.8.2.2.4 音视频解码器插件配置

该部分配置与对应平台支持的音视频编码格式有关，配置文件的解码器插件会在播放开始前进行逐个加载，并在播放过程中通过编码格式来找到匹配解码器。如果有需要添加新的解码器，则在修改配置文件时需要保证使用的解码库名字与配置文件中的保持一致，否则会出现加载解码库失败的现象。

4 xplayer 关键 API 介绍

4.1 XPlayerCreate

- 原型：XPlayer* XPlayerCreate()
- 作用：创建一个播放器实例
- 参数：NA
- 返回：
  - 非空: 成功
  - NULL: 失败
- 调用说明：NA

4.2 XPlayerSetNotifyCallback

- 原型：int XPlayerSetNotifyCallback(XPlayer* p, XPlayerNotifyCallback notifier, void* pUserData)
- 作用：设置 XPlayer 的回调通知
- 参数：
  - p：通过 XPlayerCreate 创建的 XPlayer 指针
  - notifier：回调通知
  - pUserData：应用程序传下来的自定义数据
- 返回：
  - 0: 成功
  - 非 0: 失败
- 调用说明：XPlayer 将接收来自下层的回调通知，进行相应的操作

4.3 XPlayerInitCheck

- 原型：int XPlayerInitCheck(XPlayer* p)
- 作用：检测播放器是否创建成功
- 参数：

  - p：通过 XPlayerCreate 创建的 XPlayer 指针

- 返回：
  - 0：成功
  - -1：失败
- 调用说明：NA

4.4 XPlayerSetAudioSink

- 原型：void XPlayerSetAudioSink(XPlayer* p, const SoundCtrl* audioSink)
- 作用：设置音频接收设备的操作指针，用于接收音频
- 参数：
  - p：通过 XPlayerCreate 创建的 XPlayer 指针
  - audioSink：音频接收设备的操作指针
- 返回：NA
- 调用说明：在 XPlayerStart 之前调用此接口

4.5 XPlayerSetVideoSurfaceTexture

- 原型：int XPlayerSetVideoSurfaceTexture(XPlayer* p, const LayerCtrl* surfaceTexture)
- 作用：设置视频显示控制
- 参数：
  - p：通过 XPlayerCreate 创建的 XPlayer 指针
  - surfaceTexture：显示控制模块的操作指针
- 返回：
  - 0：成功
  - -1：失败
- 调用说明：NA

4.6 XPlayerSetDataSourceUrl

- 原型：int XPlayerSetDataSourceUrl(XPlayer* p, const char* pUrl, void* httpService, const CdxKeyedVectorT* pHeaders)
- 作用：设置 url 数据源
- 参数：

  - p：通过 XPlayerCreate 创建的 XPlayer 指针
  - pUrl：网址或本地文件路径
  - httpService：服务器信息
  - pHeaders：头文件信息
- 返回：
  - 0：成功
  - -1：失败
- 调用说明：NA

4.7 XPlayerPrepareAsync

- 原型：int XPlayerPrepareAsync(XPlayer* p)
- 作用：播放器异步准备
- 参数：
  - p：通过 XPlayerCreate 创建的 XPlayer 指针
- 返回：
  - 0：成功
  - -1：失败
- 调用说明：NA

4.8 XPlayerStart

- 原型：int XPlayerStart(XPlayer* p)
- 作用：播放器开始播放
- 参数：
  - p：通过 XPlayerCreate 创建的 XPlayer 指针
- 返回：
  - 0：成功
  - -1：失败
- 调用说明：NA

4.9 XPlayerPause

- 原型：int XPlayerPause(XPlayer* p)
- 作用：播放器暂停
- 参数：
  - p：通过 XPlayerCreate 创建的 XPlayer 指针
- 返回：
  - 0：成功
  - -1：失败
- 调用说明：在 XPlayer 处于播放状态时可调用此接口

4.10 XPlayerReset

- 原型：int XPlayerReset(XPlayer* p)
- 作用：重置播放器，将相关变量复位，并销毁各功能模块
- 参数：
  - p：通过 XPlayerCreate 创建的 XPlayer 指针
- 返回：
  - 0：成功
  - -1：失败
- 调用说明：NA

4.11 XPlayerGetDuration

- 原型：int XPlayerGetDuration(XPlayer* p, int* msec)
- 作用：获取节目时长
- 参数：
  - p：通过 XPlayerCreate 创建的 XPlayer 指针
  - msec：保存节目时长
- 返回：
  - 0：成功
  - -1：失败
- 调用说明：在 XPlayer 处于 PREPARED、STARTED、PAUSED、STOPPED 或 COMPLETE 状态下才可调用此接口，否则操作无效

4.12 XPlayerSeekTo

- 原型：int XPlayerSeekTo(XPlayer* p, int nSeekTimeMs, SeekModeType nSeekModeType)
- 作用：跳转到给定的时间点
- 参数：
  - p：通过 XPlayerCreate 创建的 XPlayer 指针
  - nSeekTimeMs：跳转的时间点
  - nSeekModeType：以什么样的方式进行跳转
- 返回：
  - 0：成功
  - -1：失败
- 调用说明：如果跳转前播放处于暂停状态，则跳转后将保持在暂停状态

4.13 XPlayerSetSpeed

- 原型：int XPlayerSetSpeed(XPlayer* p, int nSpeed)
- 作用：设置播放器的快进或快退
- 参数：
  - p：通过 XPlayerCreate 创建的 XPlayer 指针
  - nSpeed：设置快进或快退的速度
- 返回：
  - 0：成功
  - -1：失败
- 调用说明：当 nSpeed 大于 0 时为快进，可选倍速为 1、2、4，分别对应 1 倍速、2 倍速和 4 倍速；小于 0 时为快退，可选倍速为-2、-4、-8，分别对应 0.5 倍速、0.25 倍速和 0.125 倍速

4.14 XPlayerGetCurrentPosition

- 原型：int XPlayerGetCurrentPosition(XPlayer* p, int* msec)
- 作用：获取当前的播放时间点（即播放位置）
- 参数：
  - p：通过 XPlayerCreate 创建的 XPlayer 指针
  - msec：保存当前的播放时间
- 返回：

  - 0：成功
  - -1：失败

- 调用说明：在播放器处于 PREPARED、STARTED、PAUSED、STOPPED 或 COMPLETE 状态下才可调用此接口，否则操作无效，在 complete 状态下，可能会调用 prepare 方法并更改媒体信息，获取的播放时间以秒为单位

4.15 XPlayerDestroy

- 原型：void XPlayerDestroy(XPlayer* p)
- 作用：销毁播放器
- 参数：
  - p：通过 XPlayerCreate 创建的 XPlayer 指针
- 返回：NA
- 调用说明：NA

5 多媒体功能验证

5.1 Android 系统的多媒体功能验证

Android 平台对应不同产品线，我司有对应的 APK 可供验证 CedarX 的多媒体功能，例如平板可使用 Videos，而盒子则可以使用 TvdVideo 应用来播放视频文件。以及其他第三方视频应用均支持验证。Videos 和 TvdVideo 两者都是走 MediaPlayer 通路，架构图如下所示。

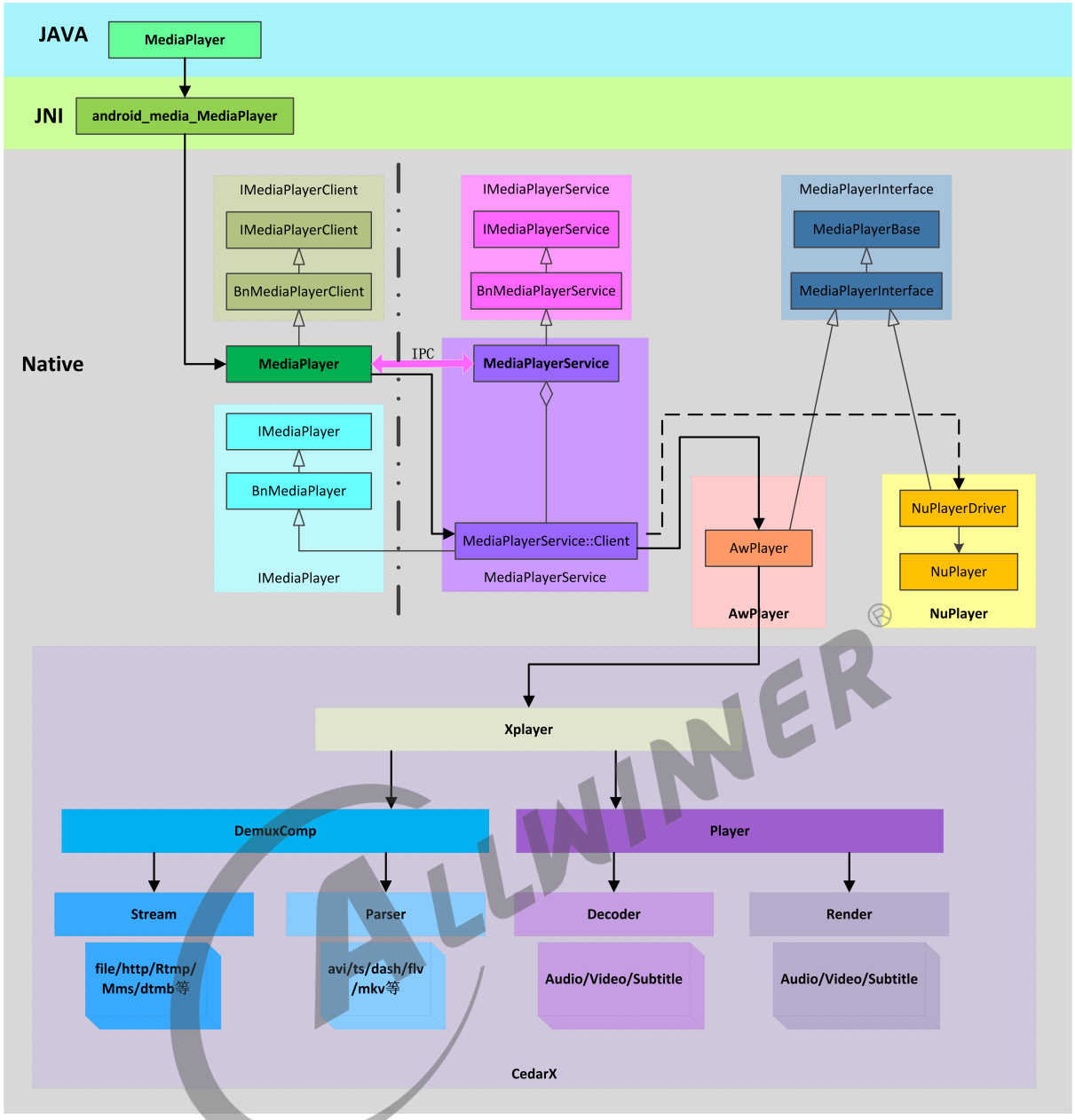

图 5-1: MediaPlayer 对接 CedarX 框架图

测试方法：

1. 把多媒体文件推入到产品的目录中，例如/sdcard/Movies 目录
2. 打开 Videos 等视频应用
3. 点击多媒体文件进行播放

5.2 Linux 系统的多媒体功能验证

在 Linux 系统上，可使用 xplayerdemo 进行多媒体功能的验证，该测试用例相当于一个播放器，可实现视频的文件路径设置、播放设置（开始、暂停、快进、停止）等功能。

该测试用例可以在 Linux 下直接编译使用，相关编译方法可以参考《Tina Linux 多媒体解码开发指南》。
验证步骤：

1. 基于 Linux 系统编译成功后，生成名为 xplayerdemo 或者 tplayerdemo(2 个播放器功能一样，只是 tplayerdemo 是基于 tplayer 对接，多封装了一层，具体使用的是哪个 demo，取决于当前开发平台) 的可执行文件，获取后将其推入小机/usr/bin/目录下，并更改权限为 777，具体命令如下：

```
adb push xplayerdemo /usr/bin/xplayerdemo
adb shell chmod 777 /usr/bin/xplayerdemo
或
adb push tplayerdemo /usr/bin/tplayerdemo
adb shell chmod 777 /usr/bin/tplayerdemo
```

2. 将播放的视频推入小机当中（可以是小机中的任何可访问目录，这里选择 mnt 目录）进入播放器，设置播放源、播放及退出播放器

```
adb push test.mp4 /mnt/test.mp4
adb shell
su
./usr/bin/xplayerdemo 或 ./usr/bin/tplayerdemo
set url:/mnt/test.mp4
play
quit
```

参数	解释说明	命令范例
help	输出该 demo 的使用方法和参数等帮助	# help
quit	结束 xplayerdemo	# quit
set url	设置文件路径，可以是本地文件的 url 也可以是网络文件	# set url:/mnt/test.mp4
play	播放的命令，表示开始播放	# play
pause	暂停播放	# pause
stop	停止播放	# stop
set speed	设置播放快进或快退速度	# set speed:2
seek to	跳播，跳播到某个时间点，seek to 时间点（秒为单位），注意时间不要超过文件总时长	# seek to:20
show media info	输出文件的媒体信息，包括时长，分辨率等	# show media info
show duration	输出文件时长	# show duration
show position	输出当前播放位置	# show position

5.2.1 xplayerdemo 流程说明

1. demo 主函数获取配置参数；
2. 播放器创建：利用 XPlayerCreate 函数创建一个播放器实例，通过 XPlayerInitCheck 判断播放器创建是否成功；
3. 消息回调：XPlayer 需要回调一些信息给上层应用，利用 XPlayerSetNotifyCallback 函数设置一个回调函数；
4. 适配层：分别通过 TinaSoundDeviceInit 和 LayerCreate_DE 函数，获取音频播放设备管理模块和显示控制模块的指针，用于输出音频和视频；
5. 准备工作：XPlayer 在通过 XPlayerCreate 函数创建了一个播放器实例, 使 XPlayer 处于 Idle 状态，此时播放器所需资源还未准备好，不能进入播放。在 Idle 状态下 XPlayerSetDataSourceUrl 函数设置播放源，获取需要播放的音视频数据内容，并使 XPlayer 进入 Initilized 状态，在 Initilized 状态下调用 XPlayerPrepareAsync 函数进行准备工作，获取媒体信息，用于初始化视频解码器、音频解码器、字幕解码器等模块，初始化完成后 XPlayer 将进入 Prepared 状态，播放器准备工作已经完成，可以开始播放；
6. 开始播放：通过 XPlayerStart 函数启动播放器进行播放，播放器向解码器传送码流数据，解码器将解码完的数据送入渲染模块进行播放和显示。在 start 状态下可以调用 XPlayerPause 函数进行暂停播放，调用 XPlayerReset 函数关闭播放器，将相关变量复位，并销毁各个模块。在 PREPARED、STARTED 和 PAUSED 状态下可通过 XPlayerSeekTo 函数跳转到给定的时间点；
7. 播放时间点获取：在 XPlayer 处于 PREPARED、STARTED、PAUSED、STOPPED 或 COMPLETE 状态时，可通过 XPlayerGetDuration 函数和 XPlayerGetCurrentPosition 函数获取节目时长和当前播放的时间点。在进行跳播时，设置的跳转时间 nSeekTimeMs 不可超过节目时长，否则操作无效；
8. 内存资源释放：在播放完后，调用 XPlayerDestroy 函数，销毁播放器，进行内存资源释放。

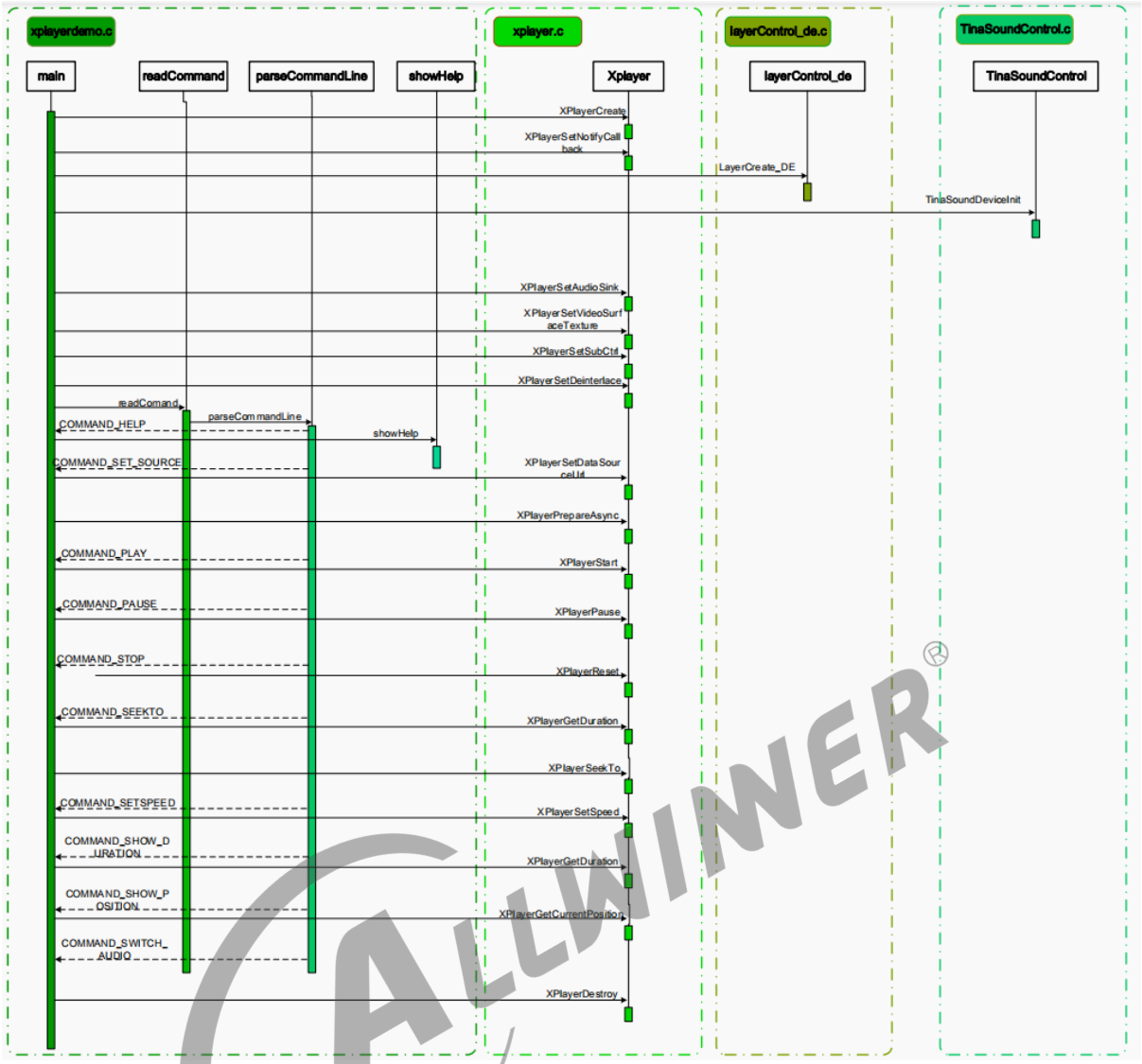

图 5-2: xplayerdemo 流程图

5.2.2 tplayerdemo 流程说明

tplayerdemo 和 xplayerdemo 这 2 个播放器功能一样，只是 tplayerdemo 是基于 tplayer 对接，相对比 xplayerdemo 多封装了一层 tplayer。

tplayerdemo 源码路径：sdk/openwrt/package/allwinner/multimedia/tina_multimedia_demo/tplayerdemo

编译方式：在上述源码路径执行 mm 命令，即可启动编译。

编译出来的 bin 文件路径：out/h135/p1/openwrt/staging_dir/target/root-h135-p1/usr/bin/tplayerdemo

⚠ 注意

注意：h13x 中，tplayerdemo 仅用于内部功能调试验证使用，不作为正式播放器使用。

5.2.3 lvgl_projector 多媒体 UI 界面播放说明

对于 h13x 平台支持了 UI 界面播放器进行多媒体播放，不需要使用 tplayerdemo 进行播放。

lvgl UI 界面播放器的调用逻辑和 tplayerdemo 一样，对接的也是 tplayer 这一层，而投屏通路则对接了 miracast 闭源库：

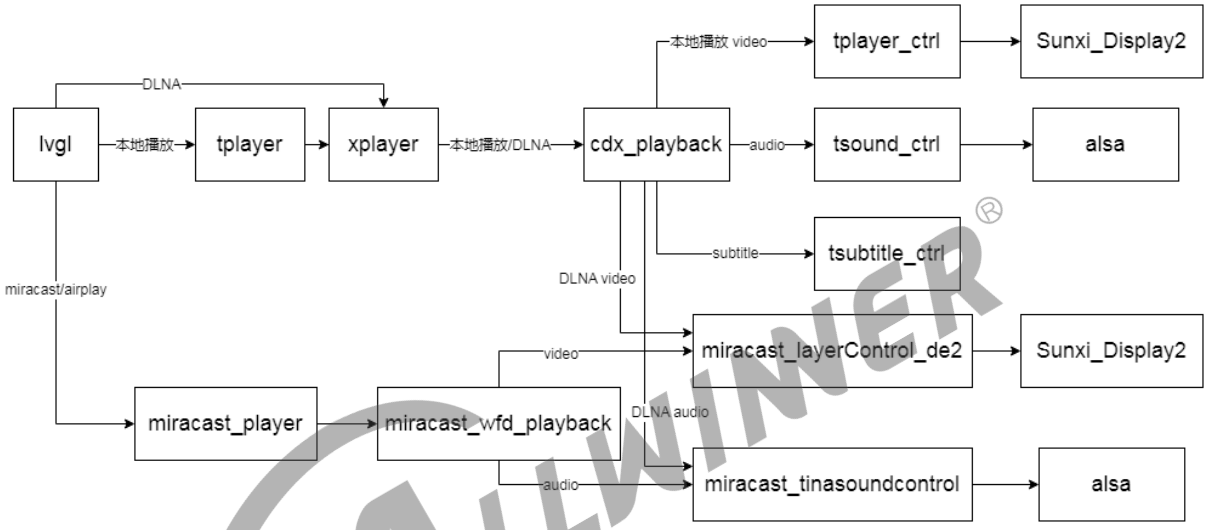

图 5-3: h13x 多媒体通路

5.3 LBC 压缩

对于 h13x 平台，支持 LBC 压缩，配置参数已默认设置好，直接使用即可。

配 置 路 径：[tina_sdk]/openwrt/package/allwinner/multimedia/tina_multimedia/machinfo/[LICHEE_IC]/libcedarx/[LICHEE_IC]_linux_cedarx.conf

```
# video lbc mode, is_lossy and rc_en
vd_lbc_mode = 1
vd_lbc_is_lossy = 1 #0:lossless; 1:lossy. lossy or lossless is little different
vd_lbc_rc_en = 1
vd_lbc_4k_en = 1 #0: disable 4k lbc; 1:enable 4k lbc
vd_lbc_smallsize_en = 0 #0: disable less than 1080p lbc; 1:enable less than 1080p lbc
```

vd_lbc_mode：是否开启 LBC 压缩，1：开启，0：关闭

vd_lbc_is_lossy：1：有损压缩，0：无损压缩

vd_lbc_rc_en：1：紧凑型排列，0：非紧凑型排列

vd_lbc_4k_en：1：1080P 以上使用 LBC 压缩，0：1080P 以上不使用 LBC 压缩

vd_lbc_smallsize_en：1：1080P 及以下使用 LBC 压缩，0：1080P 及以下不使用 LBC 压缩

5.4 DI 处理策略

对于 h13x 平台，默认开启运动模式双帧 (MD_60HZ) 的去隔行处理模式。

处理的逻辑：每次 ioctl 进行驱动调用处理时，输入两帧，输出两帧（均为非 lbc 压缩的普通 buffer），DI 处理后输出的这两帧，分别做音视频同步，再进行送显。

di_process_fb_arg参数设置

参数	参数值
di_mode	DI_INTP_MODE_MOTION
out_frame_mode	DI_DIT_OUT_2FRAME

数据流基本模型

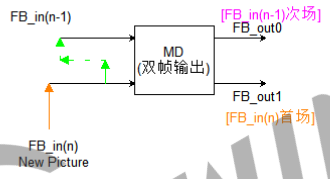

图 5-4: DI MD_60HZ 模型

5.5 多屏互动

5.5.1 Miracast

Miracast 基于 P2P 连接，不要求手机和投屏设备在同个 WiFi 网络。

多媒体处理通路：miracast 闭源库的 player.c –> miracast 闭源库的 wfd_playback –> miracast 闭源库的 layerControl_de2.c 进行显示

5.5.2 DLNA

DLNA 连接要求手机和投屏设备在同个 WiFi 网络，h13x 平台中使用的是全志的 DLNA 闭源库。

多媒体处理通路：cedarx 的 player.c –> cedarx 的 cdx_playback –> miracast 闭源库的 layerControl_de2.c 进行显示。

5.5.3 Airplay

Airplay 连接要求手机和投屏设备在同个 WiFi 网络

多媒体处理通路：miracast 闭源库的 player.c –> miracast 闭源库的 wfd_playback –> miracast 闭源库的 layerControl_de2.c 进行显示

音频通路用的是三方协议厂送进来的裸 pcm 数据，音频解码器再对裸 pcm 加包再解包，再对解包后的 pcm 数据分笔送入 audio 进行输出。

完整通路参考上图 “h13x 多媒体通路”。

具体处理通路如下：

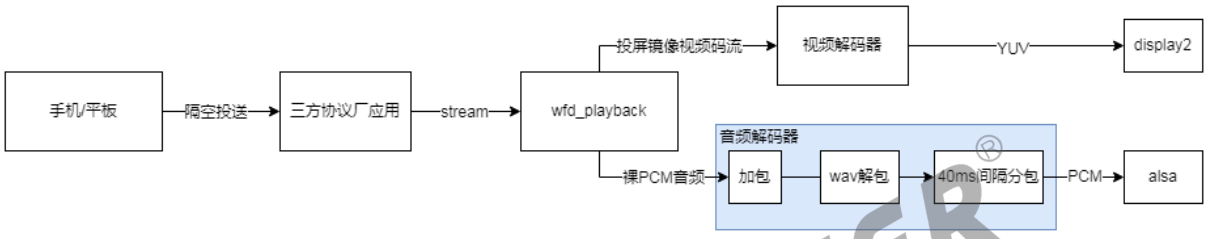

图 5-5: airplay 处理通路

5.6 视频播放内存管理

对于 h13x 平台，底层使用 2 种模式支持视频播放。

5.6.1 单路模式

h135：

宽 * 高 <= 1500000 时，开启单路非 LBC 压缩模式。解码显示内存从 awion 模块申请，原图输出。

awion 模块路径：[tina_sdk]/platform/allwinner/common/libawion/src/ion_alloc_5_15.c

h136：

默认分辨率逻辑均跑单路非 LBC 压缩模式。

5.6.2 双路模式

h135:

宽 * 高 > 1500000 时，开启双路模式，并且打开 lbc 功能，第一路从 cedarc 内部 ionAlloc 模块申请 lbc buffer，第二路从 awion 模块申请，使用第二路进行送显。

ionAlloc 模块路径：[tina_sdk]/platform/allwinner/multimedia/libcedarc/memory/ionMemory/ionAlloc.c

awion 模块路径：[tina_sdk]/platform/allwinner/common/libawion/src/ion_alloc_5_15.c

5.7 字幕策略

对于 h13x 平台，字幕数据在多媒体底层进行解码，将解码出来的解码 buffer、显示位置、颜色等信息以结构体的形式回调给 lvgl 上层，lvgl 接收到之后结构体指针后，自行绘制字幕并显示出来。应用层接收到的字幕类型有两种：文本类型和 bitmap 图形字幕，一笔字幕 buffer 数据只有其中一种字幕类型进行绘制显示。

5.7.1 内置字幕

内置字幕数据包含在视频码流内，lvgl 层不需要调用额外参数和接口，默认即可解码，收到底层回调的字幕 buffer 后，应用层再将其绘制显示。

5.7.2 外置字幕

由于外置字幕是一个单独的文件，Linux 方案需要在应用端主动调用 TPlayerSetExternalSubUrl 接口，将外置字幕文件的指针路径传给底层字幕解码器进行解码，并将解码后得到的包含字幕信息的结构体回调给应用端进行显示。

5.8 投屏旋转策略

仅在投屏场景支持旋转功能。使用解码图像后处理模块实现。开启旋转功能后，系统需要使用额外的预留内存。

6 FAQ

6.1 如何确认当前多媒体播放通路走的是硬件编解码还是软件编解码

使用系统命令 “mtop” 查看 VE 模块是否有带宽的使用情况，若有则表示正在使用硬件编解码；若没有则表示走软件编解码

```
a523-pro:/ # mtop -m
set devfreq governor node 0444!
set min_freq to max_freq value!

iter: -1
delay: 1.000000
unit: MB
output:


Max:862
total     caltot    cpu       gpu       npu       de        ve_r      ve_rw     g2d       isp       csi
862.00    858.00    205.00    213.00    0.00      169.00    183.00    88.00     0.00      0.00      0.00
827.00    823.00    168.00    214.00    0.00      170.00    183.00    88.00     0.00      0.00      0.00
807.00    803.00    170.00    214.00    0.00      167.00    169.00    83.00     0.00      0.00      0.00
```

图 6-1: 硬件解码中

6.2 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/CedarC 的运行配置文件 cedarx.conf/cedarc.conf 打印等级（log_level），增加打印

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

cedarc.conf（如有）
# printf the log of vdecoder.c
vdecoder_printf_log = 0
vdecoder_fbm_printf_log = 0
vdecoder_sbm_printf_log = 0
```

```
# printf the log of libcedarc
cdc_log_level = 3  #设置cedarc模块打印日志为DEBUG等级

ve_printf_register = 0
```

6.3 不能播放问题的定位

1. 确认播放器有没有创建成功

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

2. 确认 CedarC 是否有创建

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

3. 确认 parser 有没有解析成功

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

4. 没有如上类似的打印，说明 CedarX/CedarC/parser 没有解析成功，需搜集相关打印后，反馈给 FAE 定位问题

6.4 视频花屏问题

花屏问题涉及通路复杂，如下提供简单的分析思路进行快速排查

1. 排除视频本身存在花屏画面：先在 PC 上用播放软件（推荐 Mplayer、VLC 播放器）播放片源，

是否也存在花屏。

2. Android 系统上简单定位是 DE 或 GPU 图层问题：打开 “系统设置” —>“开发者选项”，将“停用 HW 叠加层” 打开，那默认视频显示都用 GPU 合成，若此时花屏或绿条纹现象有改观，可以进一步排查 HW 是否存在一些异常。

3. 是否存在异常打印的日志，收集相关日志反馈给 FAE

著作权声明

商标声明

免责声明