**CoreSS技术架构说明文档**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **变更（+/-）说明** | **作者** | **版本号** | **日期** | **批准** |
| 1 | 初稿 | 刘竹星 | V1.0.0 | 2016-11-30 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

文档简要功能及适用范围

1. 文档的简要功能

简要介绍了CoreSS的软件架构、CoreSS的功能及使用方法介绍和CoreSS依赖的模块。

1. 文档的适用范围

软件的开发人员，CoreSS的维护人员。

# CoreSS软件架构

## CoreSS软件架构图及简要介绍

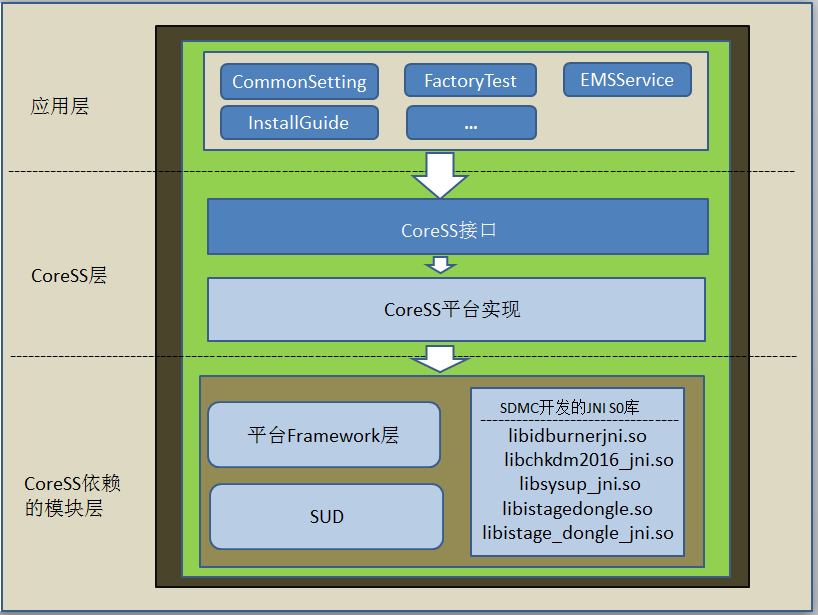


图1-1

CoreS Subsystem 以下简称CoreSS是我们公司非常重要的系统级组件，其存在的重要目的就是为了消除平台差异，减少重复代码及降低软件的维护成本。那么CoreSS是如何做到消除平台差异的呢？

首先我们都知道上层应用是通过与Framework层交互从而实现各种功能。由于上层应用的代码对Framework层代码存在很高耦合性，而不同平台SDK的Framework层代码一般存在差异的，导致了同一个应用在不同平台的代码不同，这样造成了一个功能完全一样的APK存在多份不同代码的情况。那么问题来了，我们要如何优化这种情况呢？这时候我们就需要一个“中间人”的角色，而CoreSS这个组件就是充当了上层应用与Framework层交互的中间人角色，这个中间人的主要职能就是将上层需要用到的，但是Framework层的存在差异的功能封装在一起，然后去满足上层应用的对这些差异功能的需求。由于有CoreSS的存在，对上层应用而言，在也不需要去关心Framework是否存在差异，因为上层应用可以通过CoreSS这个中间人同样可以满足自己提出需求。这样上层应用对Framework的耦合性就大大的降低了。而CoreSS提供的接口都是统一的，这个由差异到统一的过程，自然平台差异就消除了。

然后我们来谈谈CoreSS的构成，CoreSS层如图1-1所示，它分成了两个部分，分别是C oreSSAPI与CoreSS各个平台的实现。

CoreSSAPI的主要职能是将Framework 存在差异而上层应用需要用到的功能封装成统一的接口供上层应用调用。而CoreSSAPI调用的是各平台的不同的Framework实现，这里有一点需要特别注意的是，CoreSS代码并不是只有唯一一份。而是每一个平台都对应一份代码，这份代码是由CoreSSAPI 加对应平台CoreSS实现构成的。CoreSSAPI是所有平台共用一份。如果有新平台出现，只需要去把CoressAPI中所有的接口实现既可。

代码结构如图1-2所示：

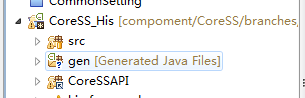


图1-2

为方便理解上面说的内容，将上层应用通过CoreSS调用到Framework层的完整流程，以S905为例，花了一张流程图如图1-3所示：

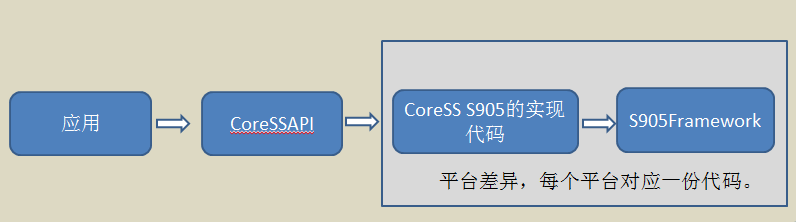


图1-3

除了调用Framework外，CoreSS还通过JNI直接调用了一些我们公司开发的SO库实现跟硬件相关的功能。主有是libidburnerjni.so libchkdm2016\_jni.so 和libsysup\_jni.so 。

libidburnerjni.so 主要用户烧录MAC SN的硬件的设备信息

libchkdm2016\_jni.so 负责检查是否是我们公司的硬件

libsysup\_jni.so 负责升级相关的业务逻辑

## CoreSS功能的实现架构



图1-4

由于CoreSS不同功能的模块间都是相互独立的。这里我以Audio这个类为例，结合图1-4简单的讲下CoreSS一个模块是如何实现的。

首先我们要区分AudioSetting AudioSettingImp Audio 这三个类的区别及关系。

AudioSetting 为抽象类，定义了上层应用需要用到的方法。

AudioSettingImp 为AudioSetting的子类，AudioSettingImp根据Framework层与Audio功能相关的接口对getAudios（） setAudio（）

及getCureentAudio（）进行实现。

Audio为CoreSSAPI的接口类。在Audio的构造函数里面，实例化了一个AudioSettingImp的对象。在getAudios方法中直接返回AudioSettingImp对象的getAudios方法的结果。

# CoreSS的功能及使用介绍

## 基本功能介绍：

要想了解全部的接口及功能请参考CoreSS接口帮助文档。

文档路径：svn://10.10.121.5/project/technology\_docs/00.CBB/09.CoreSS相关文档/API接口帮助文档/CoreSS接口帮助文档\_V1.0.2.docx

这里主要介绍一个重要的类SystemShell。

SystemShell中有两个比较常用的方法exec（String cmd）及execNewProcess(String cmd)

这两个方法都是相当于直接在串口中执行命令cmd 但是有一个很重要的区别是，exec（）这个方法会将执行的结果打印出来还有就是这个方法只有一个process且命令是按顺序执行的，也就是说执行的命令阻塞了，那么下一条命令将不会执行。

而execNewProcess(String cmd)是每次都新开一个process去执行命令，这样的每条命令都可以确保执行，但是无法控制命令执行的先后顺序。

在开发时，如何选择这两个方法呢？很简单，看执行的命令是否要按照先后顺序去执行及是否需要输出命令执行后的Log日志。如果需要则调用exec（）方法，不需要则两个方法都行。

Note：S905 S912 S905X S905D 平台需要配置SELinux权限同时确保sud的服务已经运行。如果不配置或者sud的服务未运行则 SystemShell的功能无法正常使用。

## 应用使用CoreSS的介绍：

应用开发人员的使用CoreSS方法请参考文档：

svn://10.10.121.5/project/technology\_docs/00.CBB/09.CoreSS相关文档/应用使用coressapi.jar的注意事项及帮助文档\_V1.0.0.docx

这里主要讲解下，应用最常见的一个错误，

**java java.lang.NoClassDefFoundError。**

1.首先确保应用的Androidmanifest.xml 配置了

<uses-library android:name="com.sdmc.coress" />

android:sharedUserId="android.uid.system"

2.然后确保STB的/system/etc/permissions目录中包含有com.sdmc.coress.xml 这个文件。

**3.尝试替换syste/framework目录下面的com.sdmc.coress.jar为服务器上最新的coreSS版本。鉴于CoreSS已经做了向下兼容，所以直接取最新的CoreSS来用即可。**

路径：svn://10.10.121.5/project/software/middleware/CoreSS

在确保上面三个配置的地方都没有问题，请找CoreSS维护人员解决。

# CoreSS依赖的模块及代码路径

## sud相关代码

sud的代码路径：svn://10.10.121.5/project/software/middleware/s905\_root/xbin/sdmc\_root\_v1.3.0（已确认）

 svn://10.10.121.5/project/software/middleware/s905\_root/xbin/sud\_v1.3.0（已确认）

## 底层so库相关的代码

库：

libidburnerjni.so

libchkdm2016\_jni.so

libsysup\_jni.so

libistagedongle.so

libistage\_dongle\_jni.so

libidburnerjni.so的代码路径：

svn://10.10.61.22/android/src/app/common/FactoryTester/trunk/FactoryTest\_v2.0/jni （暂时不清楚验证方法）

libchkdm2016\_jni.so的代码路径：

（代码未找到）

libsysup\_jni.so 的代码路径：

10.10.121.100:/home/svn/amlogic\_jb\_mr1\_mirror/jellybean/platform/external/sdmc-libs.git中的libsysup\_jni文件夹中

其中S905 S912 S905X的代码在：l-amlogic分支中的libsysup\_jni文件夹中。

已经验证通过。验证方法是：将sdmc\_libs目录中libsysup\_jni的代码与S905 S912 S905X SDK 中\vendor\sdmc\sdmc-libs的libsysup\_jni进行比较。比较结果是完全一致。

libistagedongle.so

libistage\_dongle\_jni.so

（代码未找到）

以上5个JNI库对应的C库代码在sdmc\_lib目录中，不同平台对应的分支不同。

主分支代码路径：10.10.121.100:/home/svn/amlogic\_jb\_mr1\_mirror/jellybean/platform/external/sdmc-libs.git

代码已经确认。

## 按键注入的代码：

代码路径：

svn://10.10.61.22/android/amlogic\_app/CoreSS/ext

已确认