**MediaPlayerService实现**

frameworks/av/media/libmediaplayerservice/MediaPlayerService.cpp

frameworks/av/media/libmediaplayerservice/ MediaPlayerService.h

frameworks/av/include/media/IMediaPlayerService.h

frameworks/av/media/libmedia/IMediaPlayerService.cpp

frameworks/av/include/media/IMediaPlayer.h

frameworks/av/media/libmedia/ IMediaPlayer.cpp

class MediaPlayerService : public BnMediaPlayerService

class MediaPlayerService :: Client : public BnMediaPlayer

**1、MediaPlayerService**

**1.1、instantiate**

mediaserver进程启动时调用，实现是调用addService，作用：新建MediaPlayerService对象。看看MediaPlayerService的构造函数：

1、mBatteryAudio，BatteryStats，不知道干嘛的

2、MediaPlayerFactory::registerBuiltinFactories()

MediaPlayerFactory的静态方法，注册底层player类，比如WhaleyPlayerFactory、NuPlayerFactory等，registerBuiltinFactories确保只被执行一次。作用：把各底层player类的Factory实例保存在键值对容器sFactoryMap中。

**1.2、createMetadataRetriever**

新建一个new MetadataRetrieverClient(pid)类对象，此类可以获取多媒体数据的元数据信息：作者、MimeType等，MediaScaner会用到。

**1.3、create**

之前讲binder机制的时候讲过了，通过BpMediaPlayerService create走到这里，新建一个内部类MediaPlayerService::Client的实例与MediaPlayerClient通信。由于内部类无法访问外部类，所以会保存外部类MediaPlayerService的对象指针，另外保存BpMediaPlayerClient对象与MediaPlayerClient通信，新建的Client对象保存在容器MediaPlayerService mClients中。

**1.4、getCodecList、getOMX**

有时间再分析MediaCodec和OMX机制。

**1.5、removeClient**

顾名思义，清理之前创建的MediaPlayerService::Client mClient。

**2、Client**

**2.1、构造、析构函数**

构造函数：保存BnMediaPlayerService对象指针、BpMediaPlayerClient对象指针引用，进程号、线程号等。

析构函数：disconnect()，mClient.clear();mPlayer.clear();p->setNotifyCallback(0, 0); p->reset();disconnectNativeWindow();智能指针的引用清理和底层player的reset等。

IPCThreadState::self()->flushCommands()，清空mOut数据，有则传到binder驱动处理，处理完返回后清空。如果本来就是空的，会直接返回。

**2.2、setDataSource**

MediaPlayerFactory::getPlayerType，Factory中底层player打分，返回分数最高的player名字。遍历调用注册在sFactoryMap中的player的scoreFactory方法。

setDataSource\_pre，根据上一步得到的player类型，创建player。MediaPlayerFactory createPlayer还会调用setNotifyCallback(cookie, notifyFunc)，把MediaPlayerService::Client对象指针和它的notify函数（静态的）传到底层播放器实例保存。

setDataSource\_post，保存新建的底层播放器实例到mPlayer，并调用它的setDataSource，开始初始化底层播放器。

**2.3、Surface、Graphic**

disconnectNativeWindow，setVideoSurfaceTexture，新建Surface对象，保存在mConnectedWindow，native\_window\_api\_connect。mConnectedWindowBinder应该是一个BpBinder。有时间再分析Surface、Graphic相关知识。

**2.4、notify**

1、Client\* client = static\_cast<Client\*>(cookie)

从底层播放器返回client对象指针，还记得吧，前面新建底层播放器时有传入一个this指针。静态方法需要单独保存this指针，才能访问非静态数据成员和方法。

2、根据mNextClient和MEDIA\_PLAYBACK\_COMPLETE，自动播放下一个音视频文件。

3、c->notify(msg, ext1, ext2, obj)，调用BpMediaPlayerClient的notify，binder通信传到MediaPlayer实例。

**2.5、decode**

封装了player起播的整个流程，getPlayerType、createPlayer、setDataSource、prepareAsync、start。

MediaPlayerFactory的定义中有一个宏DISALLOW\_EVIL\_CONSTRUCTORS，名字很霸气，其实就是禁止了这个类的默认拷贝构造函数和赋值运算符重载。

**3、延伸**

关于底层player，之前有写过WhaleyPlayer（基于FFmpeg）的实现，这次准备分析NuPlayer。

MediaPlayerService类，好像也是一个封装接口类，只是一个搬砖的，实现app进程与底层player的交互。