**BnMediaPlayerService BpMediaPlayerService**

Code路径：frameworks/av/media/libmediaplayerservice/MediaPlayerService.cpp

frameworks/av/include/media/IMediaPlayerService.h

frameworks/av/media/libmedia/IMediaPlayerService.cpp

**class MediaPlayerService : public BnMediaPlayerService**

MediaPlayerService继承BnMediaPlayerService，通过binder机制与代理端BpMediaPlayerService通信，下面看看这两个类的结构。

**1、BnMediaPlayerService**

**1.1、instantiate**

addService，新建一个MediaPlayerService（BnMediaPlayerService是它的基类）实例并注册到ServiceManager中，binder通信的功能由基类BnMediaPlayerService完成。

**1.2、class BnMediaPlayerService: public BnInterface<IMediaPlayerService>**

在讲ServiceManager时接触过BpInterface，BnInterface与它对应，也是一个模板函数。先从基类分析。

**1.2.1、IMediaPlayerService**

1、class IMediaPlayerService: public IInterface

DECLARE\_META\_INTERFACE，定义String16 descriptor、getInterfaceDescriptor()、asInterface()

IMPLEMENT\_META\_INTERFACE中实现String16 descriptor、getInterfaceDescriptor()、asInterface()

2、class IInterface : public virtual RefBase

frameworks/native/include/binder/IInterface.h

frameworks/native/libs/binder/Binder.cpp

frameworks/native/include/binder/Binder.h

这是一个纯虚类，功能是把IInterface对象转换成IBinder对象，转换方法是asBinder，asBinder里面调用纯虚方法onAsBinder()，BnInterface的onAsBinder返回this指针，所以asBinder最终返回IInterface的派生类对象。IMediaPlayerService也是一个纯虚类，不能被实例化。由于asBinder的返回值是IBinder指针，那么IMediaPlayerService的可以被实例化的派生类必须继承或包含IBinder。android构建了BnInterface与BpInterface模板类，同时继承或封装IInterface和BBinder/BpBinder，完成IBinder与IInterface的相互转换，同时拥有这两个类的功能。

sp<IBinder> IInterface::asBinder() { return this ? onAsBinder() : NULL; }

**1.2.2、class BBinder : public IBinder**

1、class IBinder : public virtual RefBase

IBinder是一个纯虚类，看看它实现了哪些功能：

queryLocalInterface、localBinder、remoteBinder，都是直接返回NULL，说明要由其派生类来实现。

2、BBinder

BBinder不是一个纯虚类，可以被实例化，看看它有什么功能：

isBinderAlive，返回true，因为是服务本身，所以直接返回true，能走到这里就说明服务还活着。

pingBinder，返回no error，逻辑同上。

Transact，只判断case PING\_TRANSACTION，其它case走default: err = onTransact(code, data, reply, flags)，特别注意，这里最终走到BBinder的派生类BnMediaPlayerService中的onTransact。

linkToDeath、unlinkToDeath，Bn不需要知道自己是否还活着，直接返回无效操作。

localBinder，返回this指针

onTransact，这个方法被BnMediaPlayerService覆盖，基类默认实现：INTERFACE\_TRANSACTION、DUMP\_TRANSACTION、SYSPROPS\_TRANSACTION

**1.2.3、BnInterface**

是一个模板类：template<typename INTERFACE> class BnInterface : public INTERFACE, public BBinder，继承IMediaPlayerService和BBinder

1. queryLocalInterface

返回自己，它本身是一个本地Bn类型IMediaPlayerService对象，所以返回自己。

1. getInterfaceDescriptor

返回IMediaPlayerService getInterfaceDescriptor，牵涉到两个宏DECLARE\_META\_INTERFACE和IMPLEMENT\_META\_INTERFACE，在IMediaPlayerService中定义，在IMediaPlayerService.cpp中实现，描述符是"android.media.IMediaPlayerService"。

1. onAsBinder

返回自己，这是上面介绍过的IInterface类的纯虚方法，因为派生类继承了BBinder，所以又可以作为IBinder对象。

**1.2.4、BnMediaPlayerService onTransact**

覆盖BBinder中的onTransact。参考《addService-通过MediaPlayerService分析binder机制》，了解到，驱动中的binder实体binder node保存了service的进程信息以及MediaPlayerService对象指针。当BpMediaPlayerService发出请求时，驱动唤醒MediaPlayerService进程，并把MediaPlayerService对象指针传回用户空间，在MediaPlayerService进程中最终走到onTransact方法，根据Bp侧的cmd走不同的case。

通过Bp端的create等方法怎样一步一步走到onTransact？参考《BpMediaPlayerService::create》。

**2、BpMediaPlayerService**

**2.1、getMediaPlayerService中新建BpMediaPlayerService对象**

1、首先用defaultServiceManager获取BpServiceManager，详细过程参考《Bp-通过ServiceManager分析binder机制》。

2、通过binder = sm->getService(String16("media.player"))获取BpBinder，描述符handle与MediaPlayerService在驱动中的binder引用一致。

3、binder->linkToDeath(sDeathNotifier)，参考《getService-通过MediaPlayerService分析binder机制》，当前进程中注册MediaPlayerService的死亡通知对象。

4、sMediaPlayerService = interface\_cast<IMediaPlayerService>(binder)，这里的binder是一个BpBinder，所以interface\_cast后得到一个BpMediaPlayerService。

interface\_cast模板，之前有分析过，走IMediaPlayerService ::asInterface，根据它的参数类型：BpBinde或BBinder，返回不同的类对象，BBinder则返回this指针，BnInterface对象，可以向下转型为MediaPlayerService；BpBinder则new一个BpMediaPlayerService（BpBinder）并返回。

**2.2、class BpMediaPlayerService: public BpInterface<IMediaPlayerService>**

先从基类分析。

**2.2.1、IMediaPlayerService**

上面分析过了，class IMediaPlayerService: public IInterface，有一个asBinder方法。

**2.2.2、BpRefBase**

class BpRefBase : public virtual RefBase

这个类比较简单，功能：保存一个IBinder\* const mRemote指针。通过方法remote()返回私有数据成员mRemote。刚好是上面BpInterface onAsBinder的返回值。

**2.2.3、class BpInterface : public INTERFACE, public BpRefBase**

BpInterface构造函数，赋值一个IBinder指针给BpRefBase(remote)，onAsBinder直接返回remote()。

BpMediaPlayerService的基类已经分析完了，BpMediaPlayerService同时有IInterface和IBinder的功能，和Bn的区别：类型转换到IBinder指针的方法是使用方法remote()，取出构建时保存的IBinder\* const mRemote指针；而Bn则是直接返回this指针。

**2.3、BpMediaPlayerService**

看看这个类本身新增的方法以及实现，全部是通过binder通信，从Bn侧获取结果，真正的实现是BnMediaPlayerService，名副其实的代理。代理功能如下：

createMetadataRetriever，cmd是CREATE\_METADATA\_RETRIEVER。

create，cmd是CREATE，这一步很重要，单独开一篇分析，参考《BpMediaPlayerServicecreate》。

createMediaRecorder，cmd是CREATE\_MEDIA\_RECORDER

decode，cmd是DECODE\_URL和DECODE\_FD，本地和在线播放。

getOMX，cmd是GET\_OMX，获取OMX

getCodecList，cmd是GET\_CODEC\_LIST