一：MediaPlaybackService

1: 在onCreate()中将之前16进制串队列恢复成id数组(reloadQueue),(在这个里面一直根据歌曲数目的膨胀来扩容mPlayList的大小(ensurePlayListCapacity))，从本地文件中得到currentPosition, mPlayListLen，进而得到当前的cursor

2:在checkDrmWhenOpenTrack中进行drm验证，根据上述的cursor是否为空根据drm验证的结果再打开一次cursor(getCursorForId(mPlayList[mPlayPos]))赋给当前的成员变量

3:在openCurrentAndNext方法中打开分两种情况

①：打开成功

调用open(path)方法,在open方法中通过multiPlayer.setDataSourceAsync，来让mediaplayer来进入Prepared状态

②：打开失败：分两种情况

a:失败次数如果没有超过上限值

如果是上一首，则播放上一首，如果是下一首，则播放下一首，不管是怎么播放，要根据当前的模式来选择播放。当在随机播放的时候，将当前准备播放的歌曲在queue中的索引位置记录到histroy中，以便prev的时候的时候按照原来路径找回去

b:失败次数超过上限值

4：成员变量mIsPlayerReady表示的是只有在prepareAsync()调用完成之后才会置为true

5: 在onCreate()中注册电源管理

6: 一般设置当前MediaPlayer的NextPlayer的时候注意几个条件

①：播放模式是顺序播放

②：下一首的时间不能太长也不能太短

二：TrackBrowserActivity

当第一次打开的时候调用open(long [] list, int position)，如果第一次打开mPlayListLen为0，mPlayList则为空，此时会调用addToPlayList

三：MediaPlayer：

1:当初始化一个MediaPlayer的时候处于idle状态，当调用setDataSource()进入Initialized状态，调用prepare()方法用来解码数据，进入Prepared状态，意味着你可以调用start()方法来播放文件了，注意，当调用stop()方法的时候，不能再调用start()方法了，除非你在prepare()一次。开辟一个MediaPlayer是个很消耗资源的过程，当不在使用的时候一定要记得release()

2:即使是在service中运行，也同样需要重新开辟一个线程来做播放操作，用prepareAsync()代替prepare()，在设置它的回调接口MediaPlayer.OnPreparedListener

3:当播放媒体的时候时候出现异常MediaPlayer处于Error状态，所以必须reset它，进入idle状态才能再用它

4:对于音乐这种应用尽量将承载它们的service设置成前台服务，并提供相应的通知栏来告知用户这种Service是存在的

四：MediaPlayer状态注意点：

1:当新构造一个MediaPlayer对象的时候，如果没有进行reset()操作，当调用getCurrentPosition(), getDuration(), getVideoHeight(), getVideoWidth(), setAudioStreamType(int), setLooping(boolean), setVolume(float, float), pause(), start(), stop(), seekTo(int), prepare() or prepareAsync()这些方法的时候回调方法OnErrorListener.onError()是不会调用的，也不会进入Error状态，也只有在调用了reset()操作的时候才会正常的进入错误状态（没有setDataSource）

2：用new构造的MediaPlayer处于idle状态，create出来的MediaPlayer处于prepared状态

3:当运行程序的时候由于各种原因造成错误进入Error状态，可以在回调方法中可以通过调用reset()方法让对象恢复成idle状态

4:当将播放拖动至某个点的时候，这个过程会花费一段时间，所以通过注册回调setOnSeekCompleteListener(OnSeekCompleteListener)。在准备阶段也可以进行seekTo操作，再通过start来从这个点进行播放操作，进入start状态

五：本地存取歌曲的id，通过反转16进制来表示c4;d4;e4;89;05;15;f4;84;94;a4;b4;，然后加载的时候在转换成id数组[76, 77, 78, 152, 80, 81, 79, 72, 73, 74, 75, 0, 0, 0]( mPlayList), mPlayPos记录当前播放的位置mPlayListLen记录有效歌曲的长度

六：MediaPlaybackService：

1: reloadQueue中得到的cursor并没有急着赋给全部变量mCursor，而是先经过drm校验，通过openCurrentAndNext重新获取一次当前歌曲的cursor并赋给全局变量，再open相应路径的文件，这个会最终调用到setDataSourceImpl里面,一旦mCurrentMediaPlayer初始化成功，则setNextDataSource(null)

2:删除文件

①：删除最后一首

删除一个文件的时候时候内存中mPlayList的数据没有发生变化，而是mPlayLength的大小发生变化，mPlayPos会变成0

②：删除中间的文件

删除之后，mPlayList会将当前位置后面的id会往前覆盖 mPlayPos不会发生，因为覆盖过的id刚好是被删id的后面一首歌，再重新stop和openCurrentAndNext一次，然后发出通知notifyChange(META\_CHANGED),告诉注册过接受的activity重新根据当前播放的歌曲来绘制界面

③：两种操作都会与本地文件同步，根据mPlayList和有效mPlayLength来同步

结论：播放文件的时候一直是用的内存中的mPlayList ,mPlayPos,mPlayLength,删除文件会改变mPlayPos和mPlayLength操作，mPlayList中如果删除最后的文件数据还保留在里面，从中间删起会将后面的数据往前面覆盖，内存中mPlayList会出现暂时的数据存放，本地中放的是最新的播放列表，内存中根据mPlayPos和mPlayLength来进行一系列歌曲的播放操作

3:暂停(播放)文件

①：发出播放状态改变通知，并savequeue(false),储存当前这个首歌在列表中的位置和这首歌播放的进度

②：在播放界面的前一个界面进入播放界面，会根据这个界面的cursor来更新本地文件的播放列表，本地播放列表会随着不同界面进入播放界面而发生变化