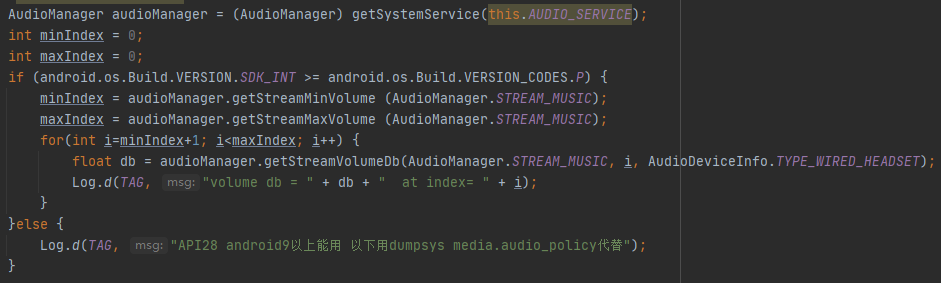
安卓音量曲线源码分析

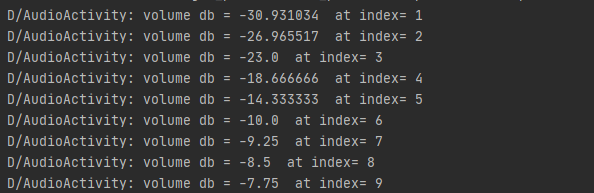
对于一个音频定制化的安卓产品，安卓系统需适配音频采集与播放的各个音频器件，为了达到一个合适的音频效果，需要调节音频采集和播放的增益，音频采集增益一般由音频codec驱动提供，音频播放增益一般主要由安卓系统的音频曲线控制。本文基于RK3568安卓11平台，探究安卓如何获取音量曲线。

1. **应用层面调用**

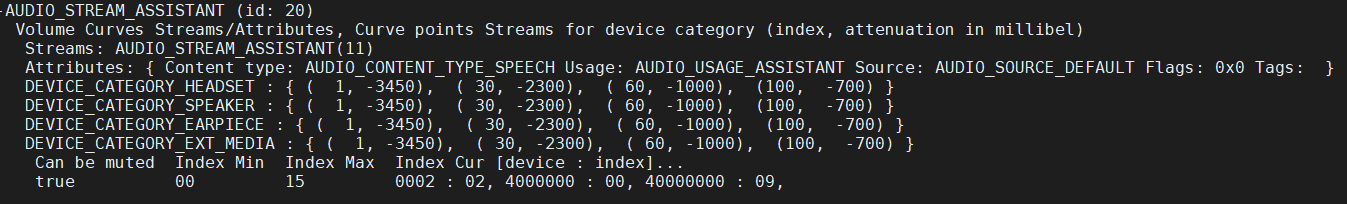
Android API大于等于28的设备提供获取音量db值函数接口，代码如图所示。



执行的结果如图所示。



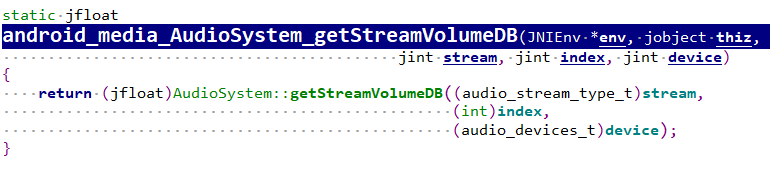
如果Android API 小于28则用dumpsys media.audio\_policy代替，部分结果如图所示。



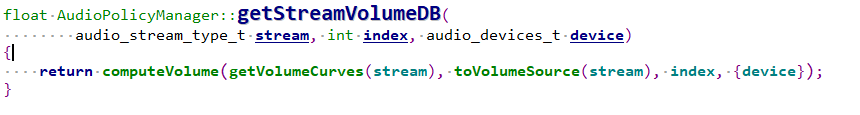
其中在音量6、音量3时，从安卓接口得到的值为-23db，-10db；从串口命令等到的值为-2300 -1000。两者从数值上相差100倍。

1. **获取音量曲线**

跟踪安卓API接口函数public float getStreamVolumeDb(@PublicStreamTypes int streamType, int index,@AudioDeviceInfo.AudioDeviceTypeOut int deviceType)可发现此函数调用public static native float getStreamVolumeDB(int stream, int index, int device);而上述函数是一个JNI函数接口。JNI是Java Native Interface的缩写，通过使用 [Java](https://baike.baidu.com/item/Java/85979)本地接口书写程序，可以确保代码在不同的平台上方便移植。故在android\_media\_AudioSystem.cpp文件中有对应JNI接口。



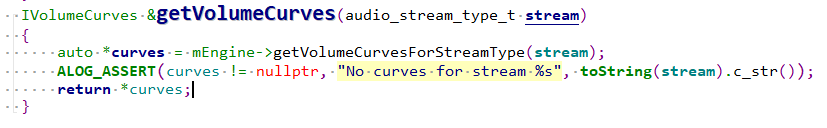
函数签名与包名路径均与API接口一致。以此函数为桥梁，可建立Java与Cpp之间的调用关系，将具体实现过程由Cpp来完成。最后会进入AudioPolicyManager这个类中的getStreamVolumeDB()这个函数，如图所示。

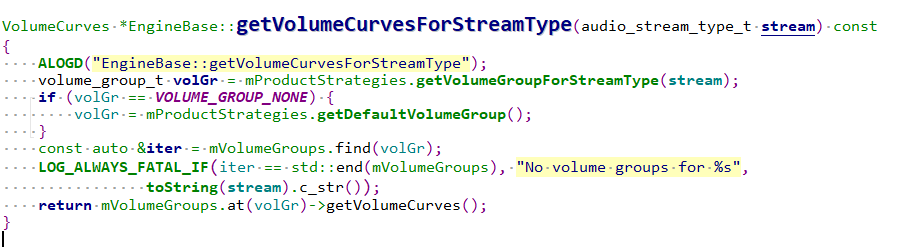


其中IVolumeCurves &getVolumeCurves(audio\_stream\_type\_t stream)主要的功能为获取安卓的音量曲线，float AudioPolicyManager::computeVolume(IVolumeCurves &curves,

VolumeSource volumeSource, int index, const DeviceTypeSet& deviceTypes)；主要的功能为计算音量曲线的DB值。

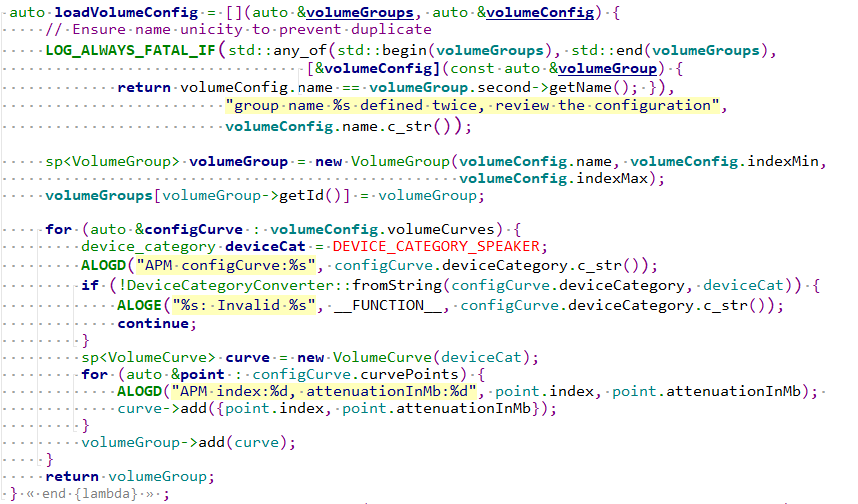
getVolumeCurves函数中，mEngine是AudioPolicyManager 这个类中的一个成员变量，是一个智能指针，函数类型为std::unique\_ptr<EngineInterface, std::function<void (EngineInterface\*)>>,EngineBase是EnginInterface的父类，根据函数多态原则，可使用基类的指针调用父类的getVolumeCurvesForStreamType(audio\_stream\_type\_t stream)函数，从而能够获取音量曲线。





从上述函数中可看出首先获取stream的声音类型，本例是以HEADSET类型来说明（测试程序输入的类型为STREAM\_MUSIC中的TYPE\_WIRED\_HEADSET输出设备），在mVolumeGroups对象中查找HEADSET这个类型，而mVolumeGroups包含 std::map<volume\_group\_t, sp<VolumeGroup> >这个键值对，即可通过volume\_group\_t这个对象找到VolumeGroup这个指针，而通过这个指针可找到成员函数中的音量曲线。

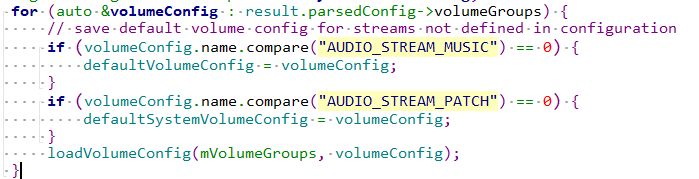
mVolumeGroups通过执行loadAudioPolicyEngineConfig()这个函数中的loadVolumeConfig lamda表达式获取，如下图所示。

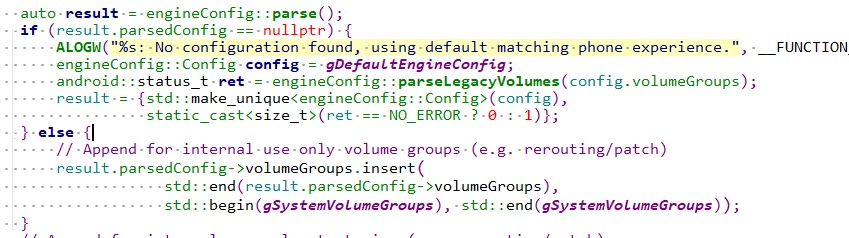


上述lamda表达式中，主要是将volumeConfig中的volumeCurves和curvePoints解析出来并放入mVolumeGroups中。而volumeConfig主要是从

result.parsedConfig->volumeGroups中获取，而result这个对象由auto result =

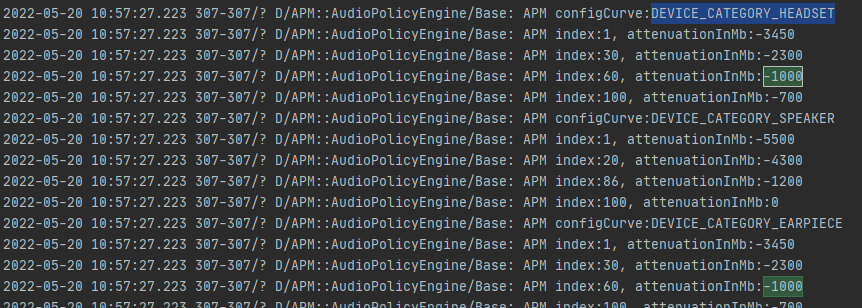
engineConfig::parse();获取。





系统开机时，音频服务会启动Engine这个类的对象，在这个类的构造函数中会执行loadAudioPolicyEngineConfig()这个函数，从而就能获取音量曲线。

部分结果如图所示

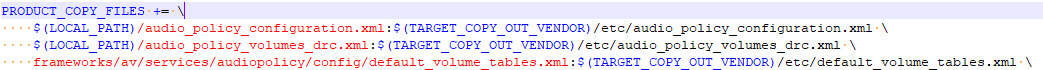


而engineConfig::parse()主要使用默认的路径，去解析本地的xml文件。

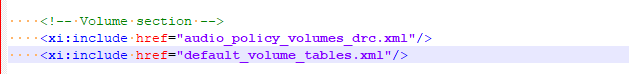
ParsingResult parse(const char\* path = DEFAULT\_PATH);

constexpr char DEFAULT\_PATH[] = "/vendor/etc/audio\_policy\_engine\_configuration.xml";

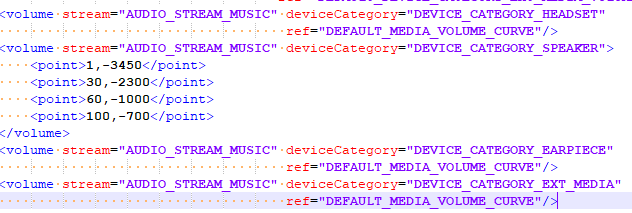
​ 在操作系统SDK\device\rockchip\common\device.mk脚本文件中，将当前目录下的audio\_policy\_configuration.xml,audio\_policy\_volumes\_drc.xml，default\_volume\_tables.xml拷贝到vendor/etc目录下。



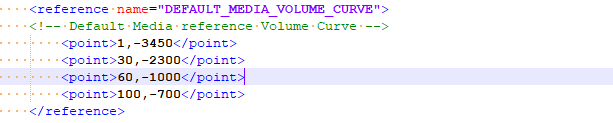
打开audio\_policy\_configuration.xml文件中可发现此配置文件会加载audio\_policy\_volumes\_drc.xml，default\_volume\_tables.xml这两个配置文件。



从audio\_policy\_volumes\_drc.xml可得，在STREAM\_MUSIC这个音量类型中，只有喇叭DEVICE\_CATEGORY\_SPEAKER的音量曲线在此修改，其余均使用默认值。



以本例HEADSET输出设备为例，使用DEFAULT\_MEDIA\_VOLUME\_CURVE这个音量曲线，文件路径在SDK\frameworks\av\services\audiopolicy\config\default\_volume\_tables.xml中，如图所示。



所得的结果与测试程序以及日志均一致。

1. **音量曲线与DB值转换**

计算DB值主要是由VolumeCurves 这个类中的float VolumeCurve::volIndexToDb(int indexInUi, int volIndexMin, int volIndexMax) const;方法完成，如图所示。



主要是包含边界值的处理以及计算音量DB值的方法，音量曲线各个节点（0-10）的计算方法为在音量曲线的节点范围内，两节点近似成一个三角形，中间的节点对应的db值计算为音量曲线值除以100，根据节点所形成的两个相似三角形等比例放大或缩小。以本例为例，如计算节点4的db值，即音量为40的db值，在30-60音量曲线范围内，对应音量曲线的刻度值为-2300，-1000。则40对应的db值为 -2300/100+（40-30）\*（-10-(-23)）/(60-30)=-18.6666与得到的结果相符。

