

「音频问题解析」打断无声疑难问题分享

一、背景

在现代社会，我们几乎无法离开我们的智能手机，因为它们已成为我们生活中不可或缺的一部分。许多人使用手机来听音乐或通过语音通话进行交流。然而，当第三方应用程序或电话打入时，由于concurrent策略，这些活动常常会被打断，特别是在音频采集方面。这种情况可能会导致用户感到沮丧和失望，因为他们不能像他们想要的那样完全享受他们的娱乐或与朋友和家人进行有效的通信。因此，解决与第三方应用程序和电话打断音频采集相关的问题变得越来越重要。（感谢ChatGPT的背景介绍模板\doge）

本篇博文，重点分享实际场景中遇到三种打断无声场景，每一个问题都会从背景入手，从初步分析过渡到深入分析，并在最后提供解决方案。如果时间充裕，大家可以在看完每一个问题的背景后停下来思考一小会，再往下阅读，以获得更深刻的印象。

二、系统语音助手打断无声

2.1 问题背景

当下的移动设备，少不了语音助手这个角色。正常情况下，语音助手如果不被主动触发，是不会打断App中正在进行的音频采集。然而，用户曾反馈一个无声问题，问题描述如下：使用红米note 8手机，本端进房并开启麦克风，但音浪波动表明采集无声；然而，如果将手机息屏再打开，采集恢复正常。到这里，不妨先停下来思考一分钟，这可能是什么原因？

2.2 初步分析

第一时间查看了关键埋点，可以确认的信息是：1. 采集设备启动成功；2. 从AudioRecordingCallback信息中看出有疑似设备抢占的行为。于是联系用户确认是否有后台App正在同时使用采集设备，然而用户清空了所有后台App重试，发现问题仍然存在。

刚好手头有一台红米note 8手机，进行本地测试发现可100%复现，这是非常乐观的现象。便在复现的同时，做了两件事：1. 再细化操作步骤，得到每一条AudioRecordingCallback信息；2. 利用dumpsys查看抢占采集的进程。得到如下数据：

	A	B	C	D
1	行为	是否有声	时间	message
2	进房开麦	×	16:23:00:005	{[audio_session_id:745,client_audio_source:1,audio_source:1,is_client_silenced:0,is_by_self:-1]} {audio_session_id:737,client_audio_source:1,audio_source:1,is_client_silenced:0,is_by_self:-1]}
3	息屏	√	16:23:12:561	{[audio_session_id:745,client_audio_source:1,audio_source:1,is_client_silenced:0,is_by_self:-1]}
4		×	16:23:21:147	{[audio_session_id:745,client_audio_source:1,audio_source:1,is_client_silenced:1,is_by_self:-1]}
5	亮屏不解锁	×	16:23:21.174	{[audio_session_id:753,client_audio_source:1,audio_source:1,is_client_silenced:0,is_by_self:-1]} {audio_session_id:745,client_audio_source:1,audio_source:1,is_client_silenced:1,is_by_self:-1]}
6	解锁	√	16:23:42:525	{[audio_session_id:753,client_audio_source:1,audio_source:1,is_client_silenced:0,is_by_self:-1]} {audio_session_id:745,client_audio_source:1,audio_source:1,is_client_silenced:0,is_by_self:-1]}
7	退房	×	16:23:51:998	\

```
1 // 进房【无声】
2 rec update riid:759 uid:10180 session:745 src:MIC pack:用户App
3
4 // 息屏【有声】
5 rec stop riid:751 uid:10162 session:737 src:MIC pack:com.miui.voiceassist
6
7 // 亮屏不解锁【无声】
8 rec update riid:759 uid:10180 session:745 src:MIC pack:用户App
9 rec start riid:767 uid:10162 session:753 src:MIC pack:com.miui.voiceassist
10
11 // 解锁【有声】
12 rec update riid:759 uid:10180 session:745 src:MIC pack:用户App
13
14 // 退房
15 rec stop riid:759 uid:10180 session:745 src:MIC pack:用户App
16 rec stop riid:767 uid:10162 session:753 src:MIC pack:com.miui.voiceassist
```

结合两个数据来看，问题原因就比较明显了。进房后看着只有用户App在使用采集，但实际上因为小爱同学语音助手早就启动，用户App抢占失败导致采集无声；而熄屏后，由于小爱同学语音助手停止，因此用户App的采集恢复正常；亮屏不解锁的情况下，由于小爱同学语音助手再次启动，音频

采集抢占成功导致无声；**解锁**后，用户App重新抢占麦克风成功，音频采集恢复正常。

顺着小爱同学语音助手这条线索，在【设置】→【小爱同学】→【小爱语音】→ 关闭【语音唤醒】，便可以解决问题。这种现象当前仅在红米 note 8中发现，后来在[google/oboe](#)的[issue](#)中也有发现类似的报错，可以确认的是，这属于厂商机型问题。

2.3 深入分析

然而上述的解决方案需要每一个用户主动在设置中关闭语音唤醒，这并不是一种好的解决方案。因此，需要继续探究作为开发者可以为用户排忧解难的方案。

针对音频采集而言，有几种常用可替换的配置参数：1. 采样率；2. 通道数；3. `audioSource`；4. `audioLayer`(java or `opensles`)；5. 音频模式。由于理论分析认为通道数非影响因素，故简单测试后便将其排除在外。此外，用户使用的是媒体模式，故去除音频模式这一维度。需要进行一定排列组合的参数还剩下三种，测试结果如下：

测试一：固定`audioSource`(0)，结合`audioLayer`和采样率进行测试

	A	B	C	D	E
1	<code>audioSource</code>	通道数	<code>audioLayer</code>	采样率	是否有声
2	DEFAULT (0)	1	opensles	16000	√
3				22050	√
4				24000	不支持
5				32000	√
6				44100	√
7				48000	×
8				64000	√
9				88200	√
10				96000	√
11			java	16000	√
12				22050	√
13				24000	√
14				32000	√
15				44100	√
16				48000	√
17				64000	√
18				88200	√
19				96000	√

测试二：固定采样率(48k)，结合`audioLayer`和`audioSource`进行测试

	A	B	C
1	<code>audioLayer</code>	<code>audioSource</code>	是否有声
2	opensles	DEFAULT (0)	×
3		CAMCORDER (5)	×
4		VOICE_RECOGNITION (6)	×
5		VOICE_COMMUNICATION (7)	√
6		UNPROCESSED (9)	×
7	java	DEFAULT (0)	√
8		MIC (1)	√
9		CAMCORDER (5)	√
10		VOICE_RECOGNITION (6)	√
11		VOICE_COMMUNICATION (7)	√
12		UNPROCESSED (9)	√
13		VOICE_PERFORMANCE (10)	√

从测试结果看，可以得出以下结论：**1. 若使用`opensles`作为`audioLayer`，且采样率为48k时，只能使用`audioSource`为7才能抢占过小爱同学语音助手；2. 若使用`java`作为`audioLayer`，常用的48k的采样率可以搭配任意的`audioSource`，而媒体模式下常用的`audioSource` 0也可以任意搭配采样率；**

2.4 解决方案

那么对应的解决方案也主要有以下三方面。针对第二种方法，整个pipeline中会对采样率有一定的要求，故随机型更改不太合适；针对第三种方法，该`audioSource`是通话模式常用的属性。因此，最终选择对红米note 8机型下发`java`的`audioLayer`，**此方法最终非常有效地解决了用户问题并给用户提供了满意的答复，顺利通过其正式环境测试。**

- a. **改变`audioLayer`**：切换为`java`采集
- b. **改变采样率**：媒体模式采集采样率设为非48k
- c. **改变`audioSource`**：媒体模式`preset`设置为`SL_ANDROID_RECORDING_PRESET_VOICE_COMMUNICATION`

💡 此类问题可以查询logcat关键词如下：

`audio_hw_primary: start_input_stream: enter: stream(0xf4836500)usecase(23: audio-record)`
`audio_hw_primary: start_input_stream: use case assigned already in use, stream(0xf4836500)usecase(23: audio-record)`

三、来电打断无声 1

3.1 问题背景

当App正在前台采集音频数据时，用户手机来电打断采集是非常常见的现象，而这一场景在多人长时间会议中尤其普遍。这里就要引出一个问题：[使用华为P30 Pro手机，本端开麦克风在房间内正在与对端愉快地交流，突然本端来了个电话，过了30s电话挂断回到房间内，对端表示听不到本端声音。同样，不妨停下来想一会原因可能是什么？](#)

3.2 初步分析

根据经验，使用不同的audioLayer进行音频数据采集时，被打断后的表现可能是不同的。分析日志得知，该问题发生时，原本使用的是aaudio采集，但当来电打断后，采集流接连触发了自身的重启，但在request_start时失败。这一断流现象被device manager侦测到，触发了使用其他audioLayer重启的机制，遗憾的是，这一重启行为在通话过程中被执行，opensles和java两种audioLayer的采集均以启动失败告终，表现为采集无声现象。但如果在通话过后主动重启一次aaudio的采集，采集流是可以恢复的。

同一台P30 Pro手机，如果在电话打断之前使用兼容性更好的java采集呢？通过日志分析，系统会通过回调函数AudioRecordingCallback告知App此时被静音，而当电话打断结束后，**采集流自行恢复了**。

3.3 深入分析

小米11测试

各手机厂商的各种型号手机系统经过个性化修改，相同场景的表现很容易出现差异。因此，取来一台小米11的手机，分别尝试在aaudio、opensles、java三种audioLayer场景下的行为。结果发现：

1. aaudio在来电打断时内部重启流成功，但此时表现为静音；在电话挂断后再次触发内部重启，同样成功，此时未被系统静音，因此不会出现采集无声的问题；
2. opensles与java两种情况下，来电均表现为系统静音，而挂断电话后音频采集自行恢复正常。

总结来说，**针对小米11这款机型，依赖采集自行恢复不会存在问题**。

华为P30测试

此外，还测试了华为P30 Pro的兄弟机型P30，行为表现与小米11相同。由此可见，同一厂商的不同机型，即使是同一系列，也是在机型适配的范畴内的。

3.4 解决方案

使用 [READ_PHONE_STATE](#) 权限，配合来电主动停止采集设备，电话挂断后主动启动采集设备的策略，可以有效地解决前台电话打断无声问题。

四、来电打断无声 2


4.1 问题背景

上一节谈到前台电话打断的问题，这一节分享一个更加隐晦的行为。[在已经申请了电话权限并有电话打断主动重启的逻辑条件下，发现系统为鸿蒙3.0的华为P50 Pro 用户如果在App退至后台时被电话打断，若没有及时在电话打断结束后回到前台，采集无声](#)。重要的是，这一现象在A网的avc视频会议demo中并不存在，且该其无前后台切换检测，因此是一个需要重点排查的问题。

4.2 初步分析

一般来说，在明确和其他App的差异之前，最好能对齐相关的参数以排除系统行为影响。为此，通过dumpsys确定了avc demo所使用的关键采集参数(采样率，通道数，audioLayer，audioSource，inputFlag)、关键播放参数(采样率，通道数，audioLayer，streamType，outputFlag)和音频模式。然后一通对齐后，发现问题依旧存在，那就需要更仔细地对比分析系统日志上的差异，以更准确地对齐关键调用的时机。

此外，前台重启完全没有问题，又根据怀疑点做了以下几种尝试：

- 
 1. **是否为mutepush启动**：远端带上耳机进房，仔细地听，会听到本端后台启动音频采集过程会有极其短暂的声音，怀疑音频采集本来启动成功，而后来又启动了mutepush导致，[查看线程后确认并不是](#)；
 2. **延长重启时间**：反编译avc demo查看电话重启延时为1s，而我们的重启时间为0.5s，怀疑电话打断后重启过快导致问题。[但测试发现无效](#)，且即使把延时增加到5s，后台启动问题仍在；但若在5s内回到前台，采集有声；

3. 尝试停采集在前台，启动采集在后台：测试发现无效；

4.3 深入分析

用三个关键词收缩下问题范围：鸿蒙3.0，电话打断，后台重启音频采集。而该采集无声的异常也比较有特色，即设备启动过程无任何报错，但最后的结果就是稳定的无声。结合dumpsys看，系统确实在App启动麦克风后做了静音处理，因此怀疑问题可能是什么操作没有符合系统要求导致。

```
1 rec update riid:5135 uid:10298 session:5081 src:VOICE_COMMUNICATION not silenced
2 rec stop riid:5135 uid:10298 session:5081 src:VOICE_COMMUNICATION not silenced
3 rec update riid:5143 uid:10298 session:5089 src:MIC not silenced pack:our app
4 rec stop riid:5143 uid:10298 session:5089 src:MIC not silenced pack:our app
5 rec start riid:5167 uid:10298 session:5121 src:MIC silenced pack:our app
6
7 rec start riid:5191 uid:10243 session:5145 src:MIC not silenced pack:avc demo
8 rec stop riid:5191 uid:10243 session:5145 src:MIC not silenced pack:avc demo
9 rec update riid:5199 uid:10243 session:5161 src:MIC not silenced pack:avc demo
```

对两者的logcat开展进一步的对比分析，尽量去对齐两者行为，做了以下的尝试：

1. **怀疑采集和播放后台启动顺序的影响**：由于播放启动是成功的，故怀疑后台启动采集是否需要先启动播放进行状态激活，但发现avc demo和我们的demo都是先启动播放，行为一致，**故不是该原因**；
2. **对齐setMode调用次数和时机**：分析logcat差异发现avc demo在电话挂断后重启中，多了一次setMode的调用，怀疑是否来电导致的mode改变在华为机型上需要在重启前再次主动还原，**但测试发现无效**；
3. **调用setMicrophoneMute(false)**：电话挂断后重启后，avc demo是有麦克风采集图标，而我们的没有，怀疑系统采集被mute了，为此在App上临时增加了一个setMicrophoneMute接口调用开关，电话打断重启后调用setMicrophoneMute(false)，**但测试发现无效**；
4. **换机型测试**：持续验证无果后，尝试了别的机型（mi 12, oppo findx5 pro），发现这两款机型表现和鸿蒙3.0一致，**这表明问题的范围可能是任何厂商的机型**；
5. **换系统测试**：怀疑是android 12适配问题，尝试使用android 10的oneplus 7T pro进行测试，**发现并不存在该问题**。

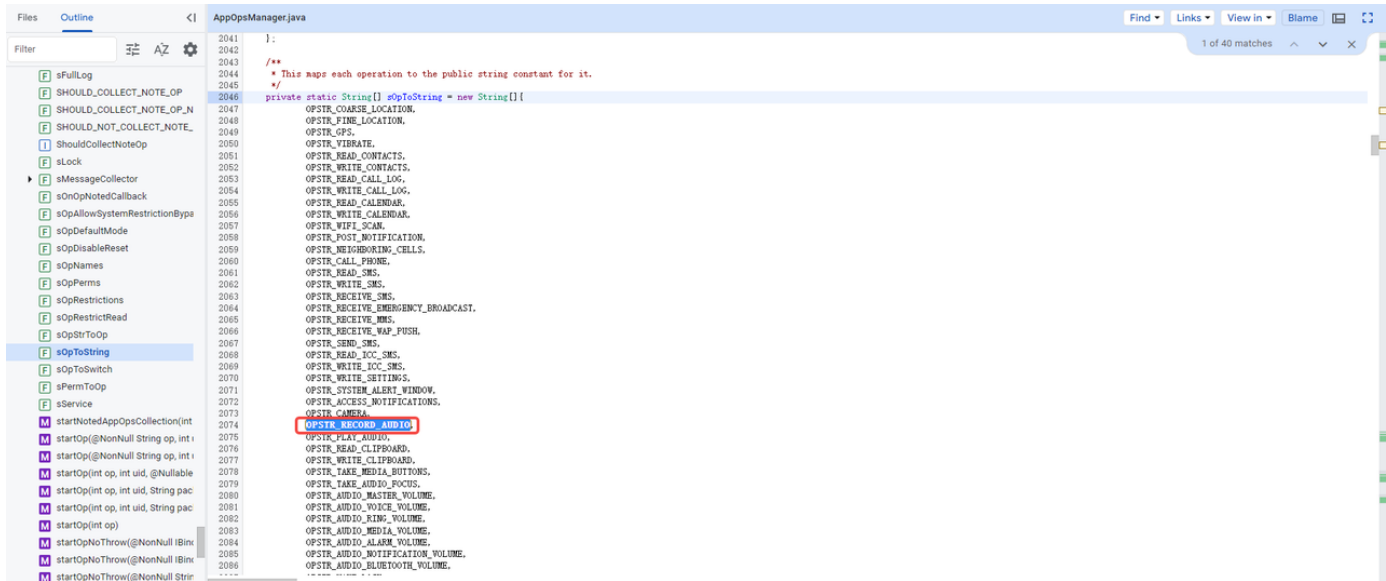
至此，由于发现了一个行为正常的案例，问题的解决有了一定转机。**故打算两手抓：1. 更进一步挖掘logcat中的差异点；2. 查询android 10以后与采集相关的适配项。**

- **关于第一点**：仔细对比发现logcat中有一个必现且非常明显的差异，就是采集启动时的系统setClientActive调用参数中：若app在前台state为2；若app在后台，byte的为0，对应的silenced为1，avc demo的为1，对应的silenced为0。查询android源码后，怀疑后台启动采集和foreground相关的行为控制有关。

```
1 【our App】
2 前台启动采集
3 I/APM_AudioPolicyManager: stopInput portId 2423
4 I/APM::AudioInputDescriptor: setClientActive(uid:10298, session:7281, state:2,
5 I/APM_AudioPolicyManager: startInput portId 2428
6 I/APM::AudioInputDescriptor: setClientActive(uid:10298, session:7305, state:2,
7 后台启动采集
8 I/APM_AudioPolicyManager: stopInput portId 2428
9 I/APM::AudioInputDescriptor: setClientActive(uid:10298, session:7305, state:1,
10 I/APM_AudioPolicyManager: startInput portId 2433
11 I/APM::AudioInputDescriptor: setClientActive(uid:10298, session:7329, state:0,
12
13 【avc demo】
14 前台启动采集
15 I/APM_AudioPolicyManager: stopInput portId 2446
16 I/APM::AudioInputDescriptor: setClientActive(uid:10243, session:7385, state:2,
17 I/APM_AudioPolicyManager: startInput portId 2451
18 I/APM::AudioInputDescriptor: setClientActive(uid:10243, session:7409, state:2,
19 后台启动采集
20 I/APM_AudioPolicyManager: stopInput portId 2438
21 I/APM::AudioInputDescriptor: setClientActive(uid:10243, session:7353, state:1,
22 I/APM_AudioPolicyManager: startInput portId 2442
23 I/APM::AudioInputDescriptor: setClientActive(uid:10243, session:7369, state:1,
```


进一步收缩logcat对比范围到系统startInput(...)调用与setClientActive(...)之间，找到了一行非常关键的差异信息。明确了系统因为缺失某项op，因此静音掉了采集。查询这个op 27，发现是麦克风权限，因此重点怀疑android版本更新过程中对麦克风权限有变动。

```
8558 8558 I InputMethod: startInput(): editor=android.view.inputmethod.EditorInfo@a4c31ac
8558 8558 D InlineSuggestionSessionController: notifyOnStartInput: com.huawei.android.launcher, 0
8558 8558 V InputMethodService: CALL: onStartInput
1719 4338 I AudioManager: setMode 0
804 804 V AudioPolicyInterfaceImpl: setPhoneState()
804 804 I APM AudioPolicyManager: setPhoneState() state 2->0
1719 1738 D hwPS_ProcessMonitor: onForegroundActivitiesChanged: 31669, 1001, false
804 16794 W ServiceManager: Permission failure: com.huawei.permission.SET_AUDIO_PARAMETERS from uid=10300 pid=8662
804 16794 V APM:AudioInputDescriptor: opening input for device type:0x80000004,@bottom profile 0xb4000075b8ca3a40 name primary
804 16794 V APM:AudioInputDescriptor: openInput returned input handle 6798 mId 3148 for device 80000004
804 16794 I AudioPolicyService: App op 27 missing, silencing record AttributionSourceState{pid: 8662, uid: 10300, packageName:
804 16794 I APM AudioPolicyManager: startInput portId 3148
804 16794 I APM AudioPolicyManager: startInput input:6798, session:9897, uid:10300
804 16794 I APM:AudioInputDescriptor: IOprofile maxActiveCount == 1 curActiveCount = 0
804 16794 I APM:AudioInputDescriptor: start, input handle 6798, profile name: primary in, curActiveCount: 1
804 16794 I APM:AudioInputDescriptor: setClientActive(uid:10300, session:9897, state:0, silenced:1 active:1)
804 16794 V APM:AudioInputDescriptor: updateClientRecordingConfiguration riid 9527 uid 10300 port 3149 session 9897 event 0
804 16794 V APM:AudioInputDescriptor: updateClientRecordingConfiguration riid 9527 uid 10300 port 3149 session 9897 event 2
804 16794 I AudioPolicyManagerCustomImpl: StartInput activeCount() = 1
1719 4669 I AudioService.RecordingActivityMonitor: HuaweiProcess, uid is 10300, silenced is true
1719 2227 I hwPS_AudioAdapter: onAudioConfigs packageName: com.huawei.htc.ctdemp uidAudio: 10300 silenced: true source: 1
```



- 关于第二点：以foreground麦克风权限为线索查询发现，android 11系统针对麦克风的权限限制就有了更新，即若想在后台使用麦克风和摄像头，除了之前的权限申请操作外，还需要添加service适配。这也恰好解释了为什么之前在android 10上测试没有问题。

将前台服务类型添加到长时间运行的 worker

如果您的应用以 Android 10（API 级别 29）或更高版本为目标平台，且包含需要位置信息访问权限的长时间运行的 worker，请指明该 worker 使用 location 的前台服务类型。此外，如果您的应用以 Android 11（API 级别 30）或更高版本为目标平台，且包含需要访问摄像头或麦克风的长时间运行的 worker，请分别声明 camera 或 microphone 前台服务类型。

4.4 解决方案

总结来说，需要适配的工作主要分为两个方面，适配后即可解决该问题。

- a. 在AndroidManifest.xml中加入service，确认foregroundServiceType包含microphone权限；
- b. 运行时在恰当时机启动service，其中要指明前台服务类型 FOREGROUND_SERVICE_TYPE_MICROPHONE