利用AEC（回声消除算法）实现TTS Barge-In（提示音打断）和Music Barge-In（音乐打断）功能

1. Music Barge-in 功能概述
1.1 什么是 Barge-in 通常，大部分语音识别的应用有个比较大的缺陷，就是如果当时设备在播放 的时候，由于此时设备 Speaker 同时在发出声音，声音会不可避免的传入到 麦克风里面，此时麦克风拾取的 audio 数据混杂了用户说出的命令和
Speaker 的回声，识别引擎已经不能很好的从中分辨需要识别的命令，导致 此时识别的效果大打折扣，用户的体验下降厉害。从而很多的语音识别场景 很不完整，比如新闻/邮件的朗读，音乐的播放，当用户通过语音指令要求 系统去朗读邮件/新闻或者是播放音乐的时候，却不能通过语音去停止，必 须通过手工去停止。

云知声的语音打断技术，也称为 barge-in，能有效的解决这个问题，在播放 的同时能够用语音进行打断，从而获得更好的用户体验。
1.2 Barge-in 关键技术描述

在 Barge-in 里使用的关键技术 AEC(Acoustic Echo canceling)叫回声消除技术， 这项技术最早起源于电话通话和 VOIP 的发展，基本原理和步骤是
1. 原始声音被采样，做为回声消除参考
2. 麦克风拾取语音输入.
3. 针对直接路径和反射路径建模
4. 语音输入和原始声音,相关分析(起始的延时和比对窗口 )
5. 自适应滤波器降噪处理.

云知声的 AEC 算法， 在能迅速的进行回声 消除处理的同时尽量保证原始信号不失真能被识别引擎有效的处理。
1.3 TTS Bargein 和 Music Barge-in 的区别

根据播放场景的不同， 云知声的 Barge-in 有两种方式，一种是 TTS Barge-in, 另一种是 Music Barge-in。

TTS Barge-in 是指应用在播放语音合成内容的时候能够打断，这个时候因为 应用自身在播放 TTS，应用完全知道 Speaker 正在播放 Audio 的数据，根据
TTS 播放的 Audio 做为回声消除的参考，做相关性算法计算将麦克风数据和 参考数据进行对齐，如果能够对齐成功后面就可以使用 AEC 模块去进行回声 消除处理。所有的算法处理可以由应用层全部完成。

Music Barge-in 则困难得多，因为 app 不知道原始的 audio 数据，没有参考 数据，也就无法进行回声消除。
1.4 Music Barge-in 实现的机会与挑战

如果上层应用能有可能获取 Speaker 播放的内容，问题就可以迎刃而解了。 事实上因为 Speaker 和麦克风都在同一个设备上， 上层应用还是有机会能获得送 给 Speaker 的 audio 数据的， 为了达到良好的效果，我们还需要 audio 能够 保证 2 点。

1. App 在接收到麦克风 audio 数据的同时能连续不断的获取 Speaker 的 数据。
2. 麦克风采样的数据和从底层获取的 Speaker 的 audio 数据要保证同步， 两者的时延必须严格的控制在 2ms 以内。 要达到上面的 2 个目标， 云知声需要和客户紧密的配合进行的深度合作。

2. 云知声软件Music barge-in 的实现概述

前面说过，对 Music BargeIn 功能来说，我们主要原理是 AEC模块提供了回 声消除算法去消除麦克风里面的回声以达到比较好的识别效果。 我们具体的实现过程是这样的。

1. 应用程序打开录音设备进行录音的同时应该有能力同步获取 Speaker 的音频数据，这是做为回声消除算法工作的基础。
2. 为了算法的有效工作， Speaker 的数据和 Microphone 的数据应该对齐， 组织在一起做为一路数据的多个 channel 交给AEC
3. 算法本身对 Speak 和 microphone 数据要求，两路音频数据的时差要 控制在 2 毫秒以内。 为了保证以上的要求，我们可以临时借用 Android 现成的立体声录音的功能， 通常情况下，因为系统只有一个麦克风，所以当我们调用立体声录音的时候， 左右声道的数据是完全一致没有任何差别的，在 Music Barge-in 的时候我们 会启动双声道的 Microphone 录音,录音开始的同时，采集一个声道 audio 包 给 Microphone 的时候同时从把给 Speaker 的包放入 Microphone 的另外一个 声道。然后依次一直进行下去，一直到录音的停止。 由于立体声录音的特性， 可以保证应用在录音的同时有能力同步的获取 Speaker 的数据。

3. Music barge-in 技术实现的说明

我们推荐采用修改 Linux driver 层或者通过修改IIS引线去获取 Speaker 的 audio。 因为只有在 Linux Driver 层或者IIS最接近 Codec 硬件，而且通常情况下，录音和播 放是同样的 Codec 芯片所以会采用相同的采样率 (例如 48K，根据不同的
Codec 方案而定)， 所以我们期望在同一个采样周期里面，送给 Speaker 的audio 包和从 Microphone 取得的 Audio 包应该是对齐的，不会有时差。 这 是取得良好性能的关键，之前纯上层的 TTS Barge In 方案有延时的原因就是 要通过软件算法去尝试对齐，这是一个很耗时的运算。 整个 barge-in 的流程如下
1. 上层应用打开录音设备，要求系统进行双声道立体声录音。
2. Linux driver或者IIS芯片 在收到上层的调用之后，进行录音。
3. Linux driver 或者IIS芯片每次在收到麦克风来的 Audio 数据，放入左声道。
4. 同时把将要送给 Speaker 的数据，填入到右声道。
5. 上层应用收到双声道的数据，分离成两路数据。
6. 两路数据分别进行重新采样转换成 16K 的数据流。
7. 两路转换之后的音频数据流交给 AEC 做回声消除。
8. 回声消除之后的音频流送给引擎去进行识别。

4. 双方配合开发建议 我们客户功能开发过程有如下的建议。
1. 云知声介绍 Music Barge-in 实现原理
2. 客户内部评估能否从 Linux driver 层录音的同时获取 Speaker 数据。或者修改IIS取得speaker数据
3. 确定可行之后，可以完成一个 Linux 的小程序，在录音的同时将录音和Speaker 的数据同时分开保存成不同的 pcm 格式的文件给云知声评估
4. 云知声内部对两个文件进行评估，确定时延是否符合 2ms 的要求。
5. 云知声确认满足时延要求之后，客户开展 driver 层的修改实施工作。
6. 云知声同步进行测试程序的开发。
7. 实施完成，双方进行联调测试，看看打断效果。
8. 功能的进一步修改和完善。

5. 其他补充建议
•客户必须确保自动的切换，也就是音乐播放结束之后，确保右声道立刻 填充空数据。
•为了避免对 **Android** 系统本身功能产生影响。建议对 **Android** 系统做一 定的扩充，增加新的 **AudioSource** 如下：
android.media.MediaRecorder.AudioSource.VOICE\_RECOG\_UNISOUND\_MUSIC\_BARGE\_IN = 11