华为 K 歌耳返&低时延通路开发指导书（V0.6）

1 背景和价值

当前 K 歌应用为提升用户 K 歌体验，都会提供“返听”功能，但是限于 android 各厂家的平台能力不同，往往耳返延时较大，不能满足用户要求。华为 K 歌耳返解决方案，可以提供小于 30ms 的低延时底回环耳返功能和随享混音效果能力，但是与 K 歌应用自有的耳返机制冲突，导致用户在选用 K 歌应用的耳返功能后，无法使用系统的返听低延时效果。

华为 K 歌耳返&低时延方案，可以实现以下目标：

(1) 降低华为手机K歌耳返时延至30ms以下；

(2) 或降低K歌应用自带耳返方案K歌耳返时延至55ms左右(经验值，仅做参考)；

(3) 使用华为提供的混音耳返音效效果；

注：华为 K 歌耳返方案提供的效果仅仅是耳机返听时可听的效果，提供给应用的数据还是用户录制的原音，并不经过华为 K 歌方案音效处理。

2 华为 K 歌方案介绍

华为 K 歌耳返方案提供双流的底回环通路，录音数据到达 HAL 层之后，一部分数据送给上层 APP，另外一部分数据送给 HIFI 模块，HIFI 模块混音特效加持后直接送给用户，而 APP 只要实现伴奏的播放即可。同时，在新平台对底层又进行深度优化，底层回环耳返延迟(见下图绿色箭头)可达到 30ms 以下。

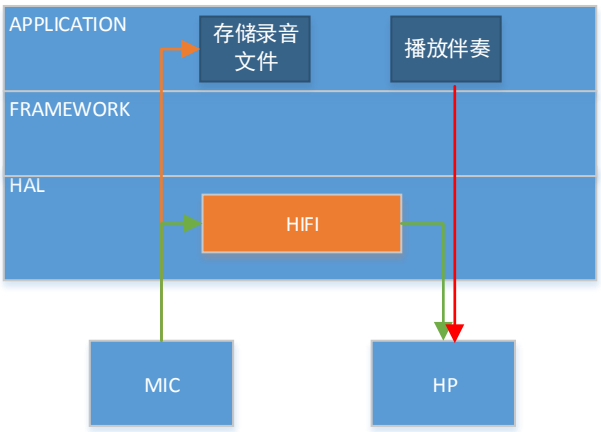


图 1 华为 K 歌耳方案框图

同时该方案也提供了一个界面提示用户进行相关的耳返和混音的设置，如下图所示，在 APP 上会提供一个悬浮按钮，点开后会有相应的设置界面。

图 2 操作界面

(1) 华为

通过 AudioManager:: getProperty，返回系统支持 K 歌耳返能力情况；

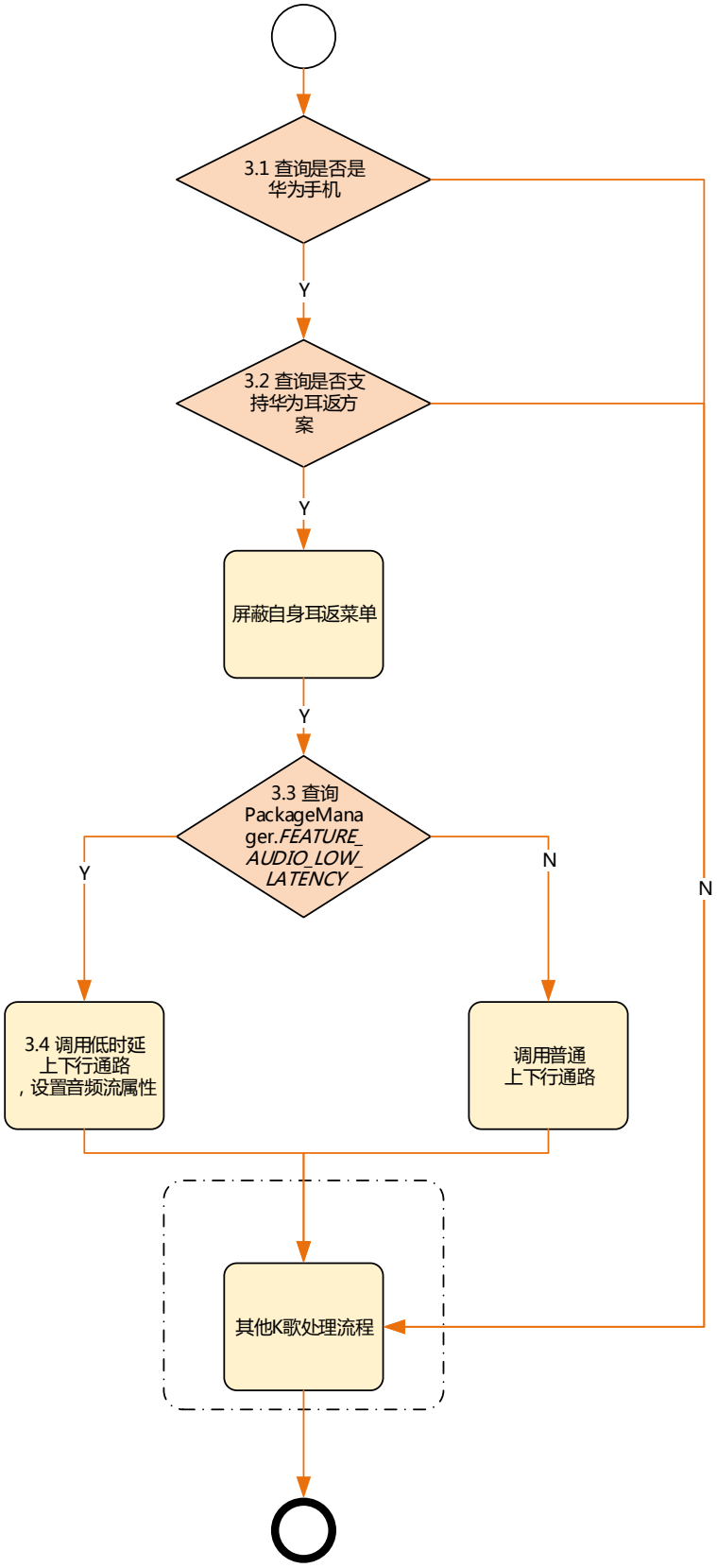
通过底层芯片级优化，提供低时延音频数据通路和混音耳返音效；

(2) APP 应用开发者

针对华为手机，通过获取系统支持 K 歌耳返能力情况，来使能和关闭业务自己的返听和音效功能；

同时通过系统低延时能力判断，是否设置低时延通路作为录音播放数据通路；

3 华为耳返方案适配流程



3.1 识别是否是华为手机

读取原生的 prop 项 ro.product.manufacturer ，用于判断是哪个厂家的手机，如果是 HUAWEI 就是华为的手机。

3.2 识别手机是否支持华为耳返方案

(1) 接口使用方法

使用 AudioManager 提供的 getProperty 函数获取 K 歌耳返支持情况。

示例：

```
AudioManager mAudioManager = (AudioManager) getSystemService(AUDIO_SERVICE);
if (null != mAudioManager)
String enable =
mAudioManager.getProperty("android.media.property.SUPPORT_HWKARAOKE_EFFECT");
    if ("true".equals(enable)) {
        // 手机支持耳返
    }
}
```

(2) 输入输出

输入参数：String “android.media.property.SUPPORT_HWKARAOKE_EFFECT”

返回结果：String “true” 或 “false” 或 null

返回结果解释： true 代表该手机可以开启华为 K 歌耳返，且应用在白名单内。

false 代表该手机不能开启华为 K 歌耳返，或应用不在白名单内。

null 代表该手机系统不支持查询该 Key。

3.3 识别是否有低时延通路能力

确定方法：

确定是否有 android.hardware.audio.low_latency 这个 feature，示例代码如下:

```
import android.content.pm.PackageManager;
...
PackageManager pm = getContext().getPackageManager();
boolean claimsFeature =
pm.hasSystemFeature(PackageManager.FEATURE_AUDIO_LOW_LATENCY);
```

如果返回为 True，则产品支持低时延通路；反之，不支持低时延通路.

3.4 设置低延时通路音频流属性

具体实现低延时通路（opensl）示例代码参考如下（基于 Google NDK Demo：audio-echo）：

1. 在 audio-echo/app/src/main/cpp/audio_recorder.cpp 中，创建 AudioRecorder 对象时，将 inputConfig 对象的 presetValue 属性即 audio_source 修改为 **SL_ANDROID_RECORDING_PRESET_GENERIC** 来满足华为手机低时延通路对 audio_source 的特性要求， 配置**其他 source 不生效。**

```
AudioRecorder::AudioRecorder(SampleFormat *sampleFormat, SLEngineItf slEngine)
    : freeQueue_(nullptr),
      recQueue_(nullptr),
      devShadowQueue_(nullptr),
      callback_(nullptr) {
  SLresult result;
  ...
  // Configure the voice recognition preset which has no
  // signal processing for lower latency.
  SLAndroidConfigurationItf inputConfig;
  result = (*recObjectItf_)
               ->GetInterface(recObjectItf_, SL_IID_ANDROIDCONFIGURATION,
                              &inputConfig);
  if (SL_RESULT_SUCCESS == result) {
    //原 SLuint32 presetValue = SL_ANDROID_RECORDING_PRESET_VOICE_RECOGNITION;
    SLuint32 presetValue = SL_ANDROID_RECORDING_PRESET_GENERIC;
    (*inputConfig)
        ->SetConfiguration(inputConfig, SL_ANDROID_KEY_RECORDING_PRESET,
                           &presetValue, sizeof(SLuint32));
  }
  result = (*recObjectItf_)->Realize(recObjectItf_, SL_BOOLEAN_FALSE);
  SLASSERT(result);
  result =
      (*recObjectItf_)->GetInterface(recObjectItf_, SL_IID_RECORD, &recItf_);
  SLASSERT(result);
```

2. 在 audio-echo/app/src/main/cpp/audio_player.cpp 中，将 AudioPlayer 低延时播放的音量调节为 **SL_MILLIBEL_MIN**，AudioPlayer 正常进行播放，不关闭，保持 K 歌时有低延时的上行和下行通路。

音量关闭是为了在保证低延时播放通路在打开的情况下，自带耳返的声音与华为耳返声音不会同时存在，由于自带耳返时延稍大，会出现重音，所以需要将自带耳返音量调节为 0。

设置音量方法如下：

```
SLresult AudioPlayer::Start(void) {
  SLuint32 state;
  SLresult result = (*playItf_)->GetPlayState(playItf_, &state);
  if (result != SL_RESULT_SUCCESS) {
    return SL_BOOLEAN_FALSE;
  }
  if (state == SL_PLAYSTATE_PLAYING) {
    return SL_BOOLEAN_TRUE;
  }

  result = (*playItf_)->SetPlayState(playItf_, SL_PLAYSTATE_STOPPED);
  SLASSERT(result);

  result =
      (*playBufferQueueItf_)
          ->Enqueue(playBufferQueueItf_, silentBuf_.buf_, silentBuf_.size_);
  SLASSERT(result);
  devShadowQueue_->push(&silentBuf_);

  result = (*playItf_)->SetPlayState(playItf_, SL_PLAYSTATE_PLAYING);
  SLASSERT(result);
  SetVolumeClose();
  return SL_BOOLEAN_TRUE;
}
```

```
void AudioPlayer::SetVolumeClose(void) {
  SLresult result;
  SLVolumeItf  volItf;
  // get the volume interface
  result = (*playerObjectItf_)->GetInterface(playerObjectItf_, SL_IID_VOLUME,  (void*)&volItf);
  assert(SL_RESULT_SUCCESS == result);
  //Set the player volume
  result = (*volItf)->SetVolumeLevel(volItf, SL_MILLIBEL_MIN);
  assert(SL_RESULT_SUCCESS == result);
}
```

4 可适配机型

上述方案会在华为下半年 980 芯片平台产品上首先支持。