声智Azero平台用户入门指导文档Android版本

变更记录

日期	作者	版本号	更新说明
2019/1/3	邢维/王欣	v1.0.8	初稿

概述

开发者集成SoundAI Azero Device for Android到其终端设备中，快速实现智能语音终端应用开发。

Azero Device For Android已为开发者实现了智能语音交互的大部分通用功能，包括：

1. 语音唤醒
2. 语音识别
3. 语义理解

4. 语音播报信息内容（天气、交通信息、百科等）
5. 音视频播放
6. 音视频通话
7. 提醒功能等

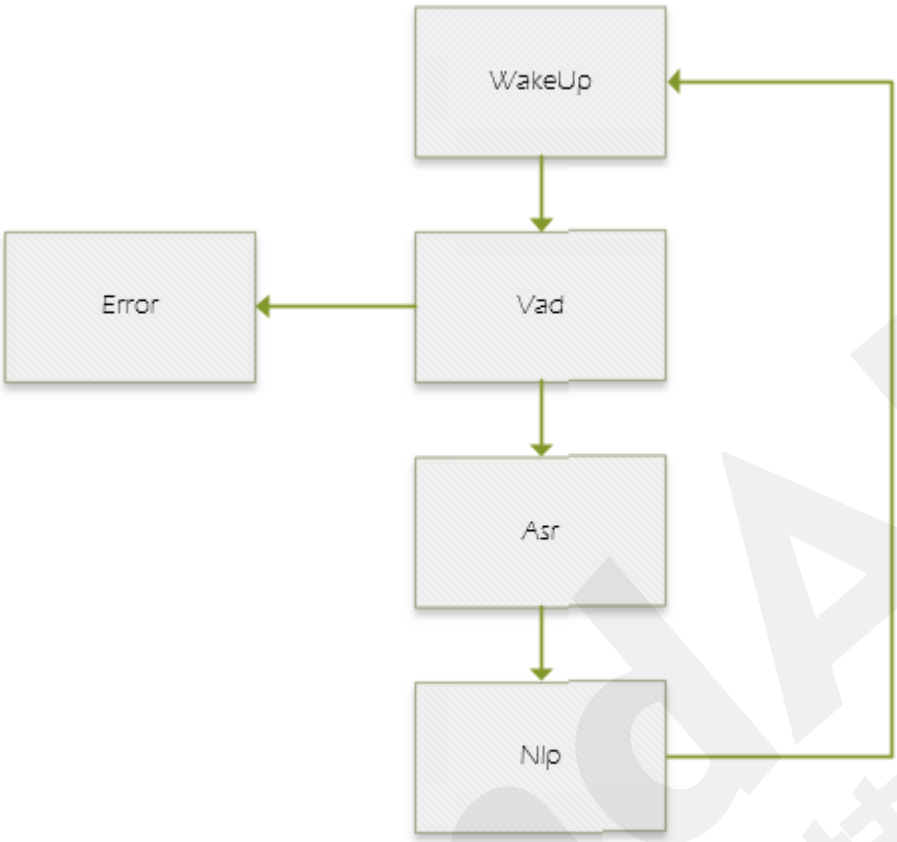
名词解释

名词缩写	名词解释	名词介绍
VAD (Voice Activity Detection)	语音活动检测，语音端点检测	从声音信号流里识别和消除是是一种长时间的静音期。在检查到语音时，SDK会收到 `VAD_BEGIN` 事件，在语音结束时，SDK会收到 `VAD_END` 事件，对应这两种事件未检测到而触发的超时事件为 `VAD_BEGIN_TIMEOUT` 和 `VAD_END_TIMEOUT`
ASR (Automatic Speech Recognition)	自动语音识别技术	是一种将人的语音转换为文本的技术
NLP (Natural Language Processing)	自然语言处理	将计算机数据转换成人类能够理解的自然语言，并通过语义理解系统反馈相应的内容。
VOIP (Voice over Internet Protocol)	网际通话协议	是一种语音通话技术，通过把语音信号经过数字化处理、压缩编码打包、通过网络传输（IP）、然后解压、把数字信号还原成声音，让通话对方听到
ONE SHOT	唤醒词和识别内容连贯识别的技术	可以将识别内容和唤醒词连贯表述，不必等待机器回应。例如： **[非ONE_SHOT模式]**“小爱同学” “哎，我在”（机器回答） “我想听一首歌”； **[ONE_SHOT模式]**“小爱同学，我想听一首歌”

会话

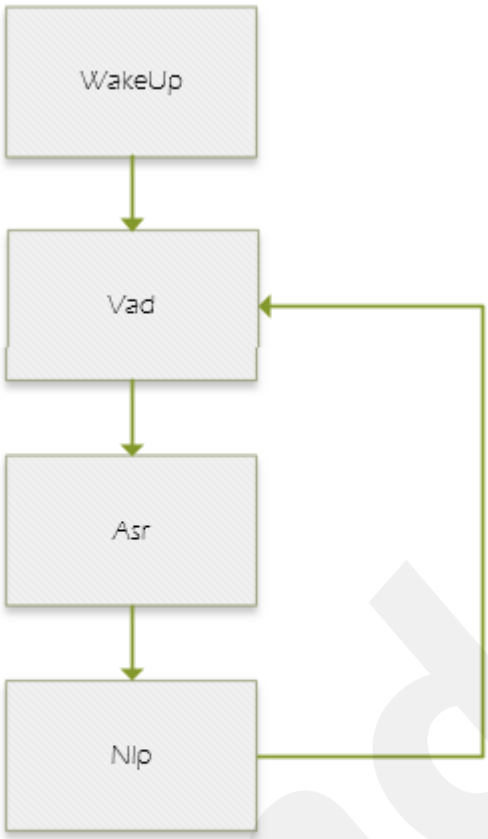
从唤醒 -> VAD状态判断 -> ASR识别结果 -> NLP语义理解结果，这一过程成为一轮会话

- 单轮回话



单轮会话

- 多轮会话



多轮会话

开始集成

集成环境

Azero Device for Android目前支持armeabi，armeabi-v7a，arm64-v8a平台，其他平台暂不支持。

minSdkVersion需大于19。

Gradle版本4.+，Android Studio版本3.+。版本可能会影响编译，建议使用推荐版本。

配置Gradle

依赖配置

```
dependencies {
    //图片处理
    implementation 'com.github.bumptech.glide:glide:3.7.0'
    //json解析库
    implementation 'com.google.code.gson:gson:2.8.0'
```

```
    //波纹动画
    implementation 'com.skyfishjy.ripplebackground:library:1.0.1'
    //工具类
    implementation 'com.blankj:utilcode:1.19.3'
    //二维码
    implementation 'com.google.zxing:core:3.2.1'
    implementation 'com.journeyapps:zxing-android-embedded:3.3.0'
    //播放器
    implementation "com.google.android.exoplayer:exoplayer-core:2.8.3"
    implementation "com.google.android.exoplayer:exoplayer-hls:2.8.3"
    //高斯模糊
    implementation 'jp.wasabeef:glide-transformations:2.0.1'
    //kotlin
    implementation 'org.jetbrains.kotlin:kotlin-stdlib-jdk7:1.3.10'
    //okhttp
    implementation "com.squareup.okhttp3:okhttp:3.6.0"

    implementation 'io.reactivex.rxjava2:rxjava:2.2.3'
    implementation 'io.reactivex.rxjava2:rxandroid:2.1.0'
    implementation name: 'library-communication', ext: 'aar'
    implementation name: 'library-push', ext: 'aar'
    implementation name: 'library-nlp', ext: 'aar'
    implementation name: 'library-saisdk', ext: 'aar'
    implementation name: 'library-tts', ext: 'aar'
    implementation name: 'soundbus', ext: 'aar'
}
```

ndk abi版本配置

如果需要指定当前支持的平台版本，可以在build.gradle中配置：

```
defaultConfig {
        ndk {
        // 配置相应需要的平台即可
        abiFilters "armeabi-v7a", "arm64-v8a"
    }
}
```

配置AndroidManifest

根据下面描述配置应用工程中的AndroidManifest.xml。

配置步骤为：

- 复制属性为Required的部分
- 替换Your package name为您的应用包名
- 配置您的SAI_PUSH_KEY
- 注册你的Broadcast Receiver用于接收消息

AndroidManifest的样例代码如下或参照Demo工程中的清单文件。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="Your package name">

    <!--Required:  接收PushReceiver必须权限-->
    <permission
        android:name="Your package name.permission.SAIPUSH"
        android:protectionLevel="signature" />

    <uses-permission android:name="Your package name.permission.SAIPUSH" />

    <application
        android:name="Your Application"
        android:allowBackup="false"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <!--Required:  授权App Key-->
        <meta-data
            android:name="SAI_PUSH_KEY"
            android:value="@string/saipush_appkey" />

        <!--Required:  建立推送服务-->
        <service
            android:name="com.soundai.saipush.service.SaiPushService"
            android:enabled="true"
            android:exported="false">
            <intent-filter>
                <action android:name="saipush.intent.service.REGISTER" />
            </intent-filter>
        </service>

        <!--接收推送的消息或响应-->
        <!--如果动态注册，需要配置相应的Action-->
        <receiver
            android:name="Your BroadcastReceiver"
            android:enabled="true"
            android:exported="false">
            <intent-filter>
                <action android:name="saipush.intent.receiver.AUTH" />
                <action android:name="saipush.intent.receiver.BIND" />
                <action android:name="saipush.intent.receiver.PUSH" />
                <action android:name="saipush.intent.receiver.NOTIFY" />
                <action android:name="saipush.intent.receiver.RECEIVE_MESSAGE" />
                <action android:name="saipush.intent.receiver.RECEIVE_RECORDS" />
                <action android:name="saipush.intent.receiver.SMART_DEVICES" />
                <action android:name="saipush.intent.receiver.EXCEPTION" />
            </intent-filter>
```

```
        </receiver>

    </application>

</manifest>
```

参数配置

第一次运行Azero Device for Android在硬件上可能会出现初始化失败的情况，为适配多种硬件设备的差异，需要用户手动配置相关硬件参数，配置方法如下：

```
SoundConfigHelper.getInstance()
        .setSampleRate(ConfigPattern.R16K)
        .setBit(ConfigPattern.BIT32)
        .setChannelMap(new int[]{0,1,2,3,4,5,6,7})
        .setHardware("hw:0,0")
        .setMicNum(6)
        .setSpeakNum(2);
```

> 用户配置相关硬件参数必须在SoundBus注册之前

参数含义介绍

参数	描述
SampleRate [采样频率]	音频信号单位时间的采样次数。设备一般为16K或48K
Bit [位深]	每个单位的音频信号的精度。设备一般为16bit或32bit
ChannelMap [通道顺序]	麦克风阵列采集音频信号的顺序，可以参照**硬件自述文档**中的通道号确认方法进行通道顺序确认
Hardware [音频设备节点]	Azero Device For Android中默认使用的是Android系统的tinyalsa音频驱动，需要填写正确的声卡设备节点才可以使用。设备节点可在/dev/snd/下查看
MicNum [麦克风数量]	麦克风阵列中的麦克风数量，即实际中有录音信号的通道个数
SpeakNum [喇叭数量]	硬件的喇叭数量，即实际中有回采信号的通道个数

示例代码

初始化

在Application中注册SoundBus，并在onTerminate中取消注册。

```
public class TestApplication extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
        // 配置录音相关参数
        SoundConfigHelper.getInstance().setSampleRa(ConfigPattern.R16K)
                .setBit(ConfigPattern.BIT32)
                .setChannelMap(new int[]{0,1,2,3,4,5,6,7})
                .setHardware("hw:0,0")
                .setMicNum(6)
                .setSpeakNum(2);
        SoundBus.getInstance().register(this);
    }

    @Override
    public void onTerminate() {
        super.onTerminate();
        SoundBus.getInstance().unRegister();
    }
}
```

Activity使用

需要语音服务的Activity继承BaseMvpActivity

```
public class HomeActivity extends BaseMvpActivity{
    ...
}
```

在需要的语音技能中加入自己的逻辑

选择需要的语音技能，在View界面绑定Presenter和IView，并实现View接口

```
@CreatePresenter(presenter = {VoicePresenter.class, IntentPresenter.class})
public class HomeActivity extends BaseMvpActivity implements BaseVoiceView,
BaseIntentView {
    @PresenterVariable
    private VoicePresenter voicePresenter;
    @PresenterVariable
    private IntentPresenter intentPresenter;
```

语音技能说明

技能	presenter	view	说明
Voice	VoicePresenter	BaseVoiceView	提供基础的语音会话状态(唤醒、识别结果、识别状态)和TTS播报状态(ttsStart、ttsStop)
Media	MediaPresenter	BaseMediaView	提供媒体类消息(继续播放、停止播放等)
Intent	IntentPresenter	BaseIntentView	提供意图类消息(界面跳转，返回)
Common	CommonPresenter	BaseCommonView	提供通用类消息(asrError、未定义消息等)
Chat	ChatPresenter	BaseChatView	提供对话类消息(天气、百科、闲聊等)

自定义语音订阅

如果你不喜欢Mvp的模式，你可以通过以下方式实现对消息的订阅

如订阅Chat类消息：

```
//定义Observer
private ChatObserver chatObserver = new ChatObserver() {
        @Override
        public void onWeather(Directive directive, List<WeatherBean> weatherBeanList) {
            if (mView.handleDirective(directive)) {
                mView.showWeather(weatherBeanList);
            }
        }

        @Override
        public void onChat(Directive directive) {
            if (mView.handleDirective(directive)) {
                mView.showChat(directive);
            }
        }
    };

//注册Observer
SoundBus.getInstance().addObserver(mView.getClass().getName(), chatObserver);

//解绑Observer
SoundBus.getInstance().removeObserver(mView.getClass().getName(), chatObserver);
```

注意事项

库链接错误

SDK中的C/C++ lib库在Android N以上可能会出现下述crash:

```
java.lang.UnsatisfiedLinkError: dlopen failed: library "libcutils.so"
("/system/lib/libcutils.so") needed or dlopened by
"/system/lib/libnativeloader.so" is not accessible for the namespace
"classloader-namespace"
  at java.lang.Runtime.loadLibrary0(Runtime.java:977)
  at java.lang.System.loadLibrary(System.java:1602)
```

这是由于库中使用到了部分非NDK公开库，**可以通过降低compileSdkVersion = 22，并在工程中的jniLibs加入LinkError失败的so。** so可以在系统的/system/lib（64位在/system/lib64）下导出。

需要打包的so包括：

1. libtinyalsa.so
2. libcutils.so
3. libc++.so

也可以通过修改系统源码，**在/system/etc/public.libraries.txt中添加libtinyalsa.so,** 使其成为NDK公开库。

权限

alsa声卡权限

当SDK返回的错误码为102时，可能是由于声卡权限缺失导致的，需要执行 `chmod 666 /dev/snd`，使SDK可以访问alsa声卡。该权限如果缺失，每次SDK启动都需要赋予权限，可以通过修改系统源码，在系统启动时直接设置权限。

SELinux

如果在赋予声卡权限后，仍然返回102错误码，可能是系统的SELinux问题，导致SDK仍然无法正常访问声卡。

Logcat日志如下：

```
   com.soundai.smartbox I/oundai.smartbox: type=1400 audit(0.0:14): avc: denied { read
write } for name="pcmC0D0c" dev="tmpfs" ino=1117 scontext=u:r:untrusted_app:s0
tcontext=u:object_r:audio_device:s0 tclass=chr_file permissive=0
       type=1400 audit(0.0:15): avc: denied { open } for path="/dev/snd/pcmC0D0c"
dev="tmpfs" ino=1117 scontext=u:r:untrusted_app:s0 tcontext=u:object_r:audio_device:s0
tclass=chr_file permissive=0
       type=1400 audit(0.0:16): avc: denied { ioctl } for path="/dev/snd/pcmC0D0c"
dev="tmpfs" ino=1117 ioctlcmd=4101 scontext=u:r:untrusted_app:s0
tcontext=u:object_r:audio_device:s0 tclass=chr_file permissive=0
```

查看SELinux：`getenforce` **[Permissive(关闭状态)，Enforcing(开启状态)]**

错误码

初始化错误

错误码	释义	建议
100	配置文件路径错误	检查assets文件是否包含config目录
101	资源文件错误	检查config目录下的文件是否和SDK包中一致
102	麦克风阵列初始化失败	1.检查配置文件中HW设备是否设置正确; 2.检查HW设备节点是否被其他程序占用; 3.检查各项参数(通道数，波特率，位深)是否存在配置问题 上述问题(1~3)可通过系统自带的tinycap使用相同参数运行，运行成功即可排除; 4.检查Selinux防火墙是否关闭(setenforce 0); 5.检查HW设备是否有system级别的可读写权限(chomd +rw /dev/snd/*)