科大讯飞股份有限公司

IFLYTEK CO.,LTD.

科大讯飞 MSC V5+集成指南

目 录

1. 概述

本文档是集成科大讯飞 MSC（Mobile Speech Client，移动语音终端）Android 版 SDK 的用户指南，介绍了语音听写、语音识别、语音合成、语义理解、语音评测等接口的使用。关于各类的函数更详细的说明，请考参考《MSC Reference Manual.html》；在集成过程有疑问，可登陆语音云开发者论坛，查找答案或与其他开发者交流：http://bbs.xfyun.cn/ 。

MSC SDK 的主要功能接口如下图所示：

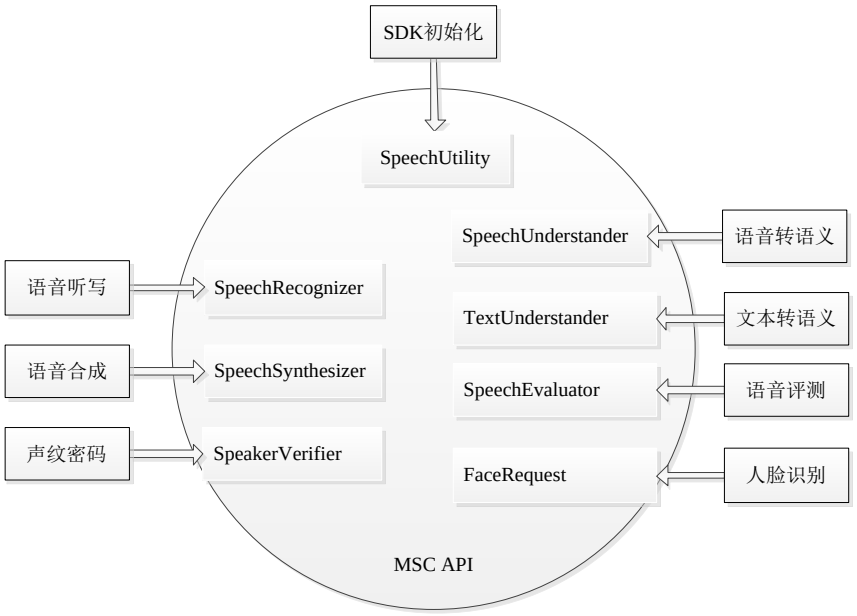

图 1 MSC 功能结构图

“MSCv5+”拥有“云+端”语音能力，包含多种语音引擎，支持云端、本地和混合三种模式。混合模式主要应对网络较差的环境，引擎设置为混合模式时，语音服务优先使用云端引擎，当网络状况较差，语音结果处理超时时，自动转换为本地引擎进行处理。

为了更好地理解后续内容，这里先对文档中出现的若干专有名词进行解释说明：

表 1 名词解释

名词	解释
语音合成	将一段文字转换为成语音，可根据需要合成出不同音色、语速和语调的声音，让机器像人一样开口说话。
语音听写	将一段语音转换成文字内容，能识别常见的词汇、语句、语气并自动断句。
语法识别	判断所说的内容是否与预定义的语法相符合，主要用于判断用户是否下达某项命令。
语义理解	分析用户语音或文字的意图，给出相应的回答，如输入“今天合肥的天气”，云端即返回今天合肥的天气信息。
唤醒	通过说出特定的唤醒词（如“芝麻开门”）来唤醒处于休眠状态下的终端设备。
唤醒+识别	在唤醒的同时对用户所说的内容进行语法识别。

2. 预备工作

Step 1 导入 SDK

将开发工具包中 libs 目录下的 Msc.jar 和 armeabi 复制到 Android 工程的 libs 目录（如果工程无 libs 目录，请自行创建）中，如下图所示：

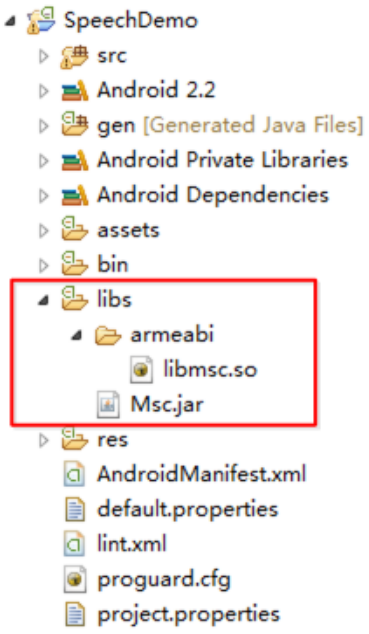

图 2 导入 SDK

如果您的项目有 libs/armeabi-v7a 这个目录，请务必把 libmsc.so 复制一份到这个目录。

Step 2 添加用户权限

在工程 AndroidManifest.xml 文件中添加如下权限

```
<!--连接网络权限，用于执行云端语音能力 -->
<uses-permission android:name="android.permission.INTERNET"/>
<!--获取手机录音机使用权限，听写、识别、语义理解需要用到此权限 -->
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<!--读取网络信息状态 -->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<!--获取当前 wifi 状态 -->
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<!--允许程序改变网络连接状态 -->
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
<!--读取手机信息权限 -->
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<!--读取联系人权限，上传联系人需要用到此权限 -->
```

```
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<!—外存储写权限，使用本地功能时需要用到此权限 -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<!--外存储读权限，构建语法需要用到此权限 -->
<uses-permission android:name="android.permission. READ_EXTERNAL_STORAGE"/>
<!--配置权限，用来记录应用配置信息 -->
<uses-permission android:name="android.permission. WRITE_SETTINGS"/>
```

如需使用人脸识别，还要添加：

```
<!--摄相头权限，拍照需要用到 -->
<uses-permission android:name="android.permission.CAMERA" />
```

注：如需在打包或者生成APK的时候进行混淆，请在proguard.cfg中添加如下代码：

```
-keep class com.iflytek.**{*;}
-keepattributes Signature
```

Step 3 初始化

初始化即创建语音配置对象，只有初始化后才可以使用 MSC 的各项服务。建议将初始化放在程序入口处（如 Application、Activity 的 onCreate 方法），初始化代码如下：

```
// 将“12345678”替换成您申请的 APPID，申请地址：http://www.xfyun.cn
// 请勿在“=”与 appid 之间添加任务空字符或者转义符
SpeechUtility.createUtility(context, SpeechConstant.APPID +"=12345678");
```

createUtility 方法的第二个参数为传入的初始化参数列表，可配置的参数如下：

表 2 初始化参数说明

参数	说明	必填
appid	8 位 16 进制数字字符串，应用的唯一标识，与下载的 SDK 一一对应。	是
usr	开发者在云平台上注册的账号。	否
pwd	账号对应的密码，与账号同时存在。	否
engine_mode	引擎模式，可选值为： msc：只使用 MSC 的能力； plus：只使用语记能力； auto：云端使用 MSC，本地使用语记； 默认取值为 auto。注：使用 MSC 本地功能的请设置为 msc。	否
force_login	在 createUtility 时会对进程名称进行检查，如果名称与应用包名不一致则不进行 login 操作，返回 null，用以规避在子进程反复进行调用的问题。此参数设置是否强制 login。 默认值：false（进行检查，不强制 login）。	否
lib_name	在 createUtility 时会加载动态库，此时可以传入动态库名称。如您是预装软件，为了避免动态库冲突建议修改名称。 例如：libmsc_xxx_1072.so（xxx 为您的公司名，1072 为科大讯飞 sdk 版	否

	本号） 默认值:msc。	

参数需要以键值对的形式存储在字符串中传入 createUtility 方法，以逗号隔开，如“appid=12345678,usr=iflytekcloud,pwd=123456”。

3. 语音输入 UI

为了便于快速开发，SDK 提供了一套默认的语音输入 UI。如需使用，请务必先将 SDK 资源包 assets 路径下的资源文件拷贝至 Android 工程 asstes 目录下，如图 3 添加动画资源所示：

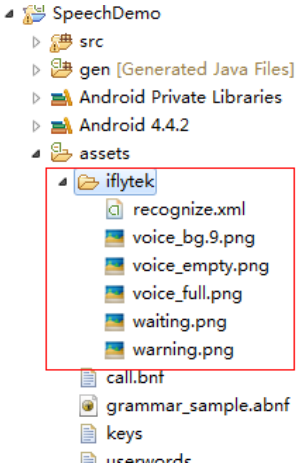

图 3 添加动画资源

语音输入 UI 控件 RecognizerDialog 可以用于语音听写、语法识别和语义理解，使用方法大致如下：

```
//1.创建RecognizerDialog对象
RecognizerDialog  mDialog = new RecognizerDialog(this, mInitListener);
//2.设置accent、language等参数
mDialog.setParameter(SpeechConstant.LANGUAGE, "zh_cn");
mDialog.setParameter(SpeechConstant.ACCENT, "mandarin");
//若要将UI控件用于语义理解，必须添加以下参数设置，设置之后onResult回调返回将是语义理解
//结果
// mDialog.setParameter("asr_sch", "1");
// mDialog.setParameter("nlp_version", "2.0");

//3.设置回调接口
mDialog.setListener(mRecognizerDialogListener);
//4.显示dialog，接收语音输入
mDialog.show();
```

4. 语音听写

听写主要指将连续语音快速识别为文字的过程，科大讯飞语音听写能识别通用常见的语句、词汇，而且不限制说法。语音听写的调用方法如下：

```
//1.创建SpeechRecognizer对象，第二个参数：本地识别时传InitListener
SpeechRecognizer mIat= SpeechRecognizer.createRecognizer(context, null);
//2.设置听写参数，详见《MSC Reference Manual》SpeechConstant类
mIat.setParameter(SpeechConstant.DOMAIN, "iat");
mIat.setParameter(SpeechConstant.LANGUAGE, "zh_cn");
mIat.setParameter(SpeechConstant.ACCENT, "mandarin ");
//3.开始听写
mIat.startListening(mRecoListener);
//听写监听器
private RecognizerListener mRecoListener = new RecognizerListener(){
    //听写结果回调接口(返回Json格式结果，用户可参见 附录13.1)；
    //一般情况下会通过onResults接口多次返回结果，完整的识别内容是多次结果的累加；
    //关于解析Json的代码可参见MscDemo中JsonParser类；
    //isLast等于true时会话结束。
    public void onResult(RecognizerResult results, boolean isLast) {
        Log.d("Result:",results.getResultString ());
    }
    //会话发生错误回调接口
    public void onError(SpeechError error) {
        //打印错误码描述
        Log.d(TAG, "error:" + error.getPlainDescription(true)); }
    //开始录音
    public void onBeginOfSpeech() {}
    //volume音量值0~30，data音频数据
    public void onVolumeChanged(int volume, byte[] data){}
    //结束录音
    public void onEndOfSpeech() {}
    //扩展用接口
    public void onEvent(int eventType,int arg1,int arg2, Bundle obj) {}
};
```

另外，您可以使用 SDK 提供的语音输入 UI 控件来提升交互体验（详见第 3 节），也可以通过上传联系人和用户词表增强听写效果。

4.1.上传联系人

上传联系人可以提高联系人名称云端识别率，也可以提高语义理解的效果，每个用户终端设备对应一个联系人列表，联系人格式详见开发包中 doc 目录下《MSC Reference Manual》中 ContactManager 类的介绍。

```
//获取 ContactManager 实例化对象
ContactManager mgr = ContactManager.createManager(context, mContactListener);
//异步查询联系人接口，通过 onContactQueryFinish 接口回调
mgr.asyncQueryAllContactsName();
//获取联系人监听器。
private ContactListener mContactListener = new ContactListener() {
    @Override
    public void onContactQueryFinish(String contactInfos, boolean changeFlag) {
        //指定引擎类型
        mIat.setParameter(SpeechConstant.ENGINE_TYPE, SpeechConstant.TYPE_CLOUD);
        mIat.setParameter(SpeechConstant.TEXT_ENCODING,"utf-8");
        ret = mIat.updateLexicon("contact", contactInfos, lexiconListener);
        if(ret != ErrorCode.SUCCESS)
            Log.d(TAG,"上传联系人失败：" + ret);
    }
};
//上传联系人监听器。
private LexiconListener lexiconListener = new LexiconListener() {
    @Override
    public void onLexiconUpdated(String lexiconId, SpeechError error) {
        if(error != null){
            Log.d(TAG,error.toString());
        }else{
            Log.d(TAG,"上传成功！");
        }
    }
};
```

4.2.上传用户词表

上传用户词表可以提高词表内词汇的云端识别率，也可以提高语义理解的效果，每个用户终端设备对应一个词表，用户词表的格式及构造方法详见开发包 doc 目录下《MSC Reference Manual》中 UserWords 类的介绍。

```
//上传用户词表，userwords 为用户词表文件。
String contents = "您所定义的用户词表内容";
mIat.setParameter(SpeechConstant.TEXT_ENCODING,"utf-8");
//指定引擎类型
mIat.setParameter(SpeechConstant.ENGINE_TYPE, SpeechConstant.TYPE_CLOUD);
ret = mIat.updateLexicon("userword", contents, lexiconListener);
if(ret != ErrorCode.SUCCESS){
	Log.d(TAG,"上传用户词表失败：" + ret);
}
//上传用户词表监听器。
private LexiconListener lexiconListener = new LexiconListener() {
	@Override
	public void onLexiconUpdated(String lexiconId, SpeechError error) {
		if(error != null){
			Log.d(TAG,error.toString());
		}else{
			Log.d(TAG,"上传成功！");
		}
	}
};
```

5. 命令词识别（语法识别）

在计算机科学和语言学中，语法分析是根据某种给定的形式文法对由单词序列构成的输入文本进行分析并确定其语法结构的一种过程。

科大讯飞命令词识别意即基于指定的语法结构，识别特定的命令词、关键词以及短语组合。根据联网状态不同，分为在线命令词识别和离线命令词识别。在线语法文件采用 ABNF 语法格式，本地识别语法文件采用 BNF 语法格式。

语法详解见：http://bbs.xfyun.cn/forum.php?mod=viewthread&tid=7595

5.1.在线命令词识别

在线命令词识别的语法文件根据作用范围不同，又分为应用级在线语法文件和终端级在线语法文件。

应用级在线语法文件，即语法文件绑定 Appid。应用级在线语法文件需在讯飞开放平台页面进行设置，一旦设置成功，不同设备上的同一应用将默认启用此在线语法。具备统一管理语法，语法更新无需更新客户端 App 的优点。

终端级在线语法文件，则是语法文件绑定某一终端，通过 App 先本地构建语法文件，再上传该语法文件获取相应的 ID 即 Grammar ID，然后在使用识别功能前指定 Grammar ID 以启用该语法文件。

在线命令词识别默认启用应用级在线语法文件。如果又指定了终端级语法文件的 Grammar ID，那么两种类型的语法文件同时生效，无优先级顺序，最终识别结果按照结果置信度降序返回。

5.1.1 应用级命令词识别

使用浏览器访问网址 http://www.xfyun.cn。在打开的页面中，点击“产品服务”、“在线命令词识别”。如图 4 所示。

在随后打开的页面中，点击“使用服务”，选择应用，点击“确定”，即可打开应用级在线语法文件上传页面，如图 5 所示。上传所需的语法文件，待页面提示“语法文件已生效”，则应用级在线语法文件启用成功。

图 4 应用级在线语法文件上传页面

若 App 只使用应用级在线语法进行命令词识别，则示例代码如下：

```
//在线命令词识别，启用终端级语法
//1.创建SpeechRecognizer对象
SpeechRecognizer mAsr = SpeechRecognizer.createRecognizer(context, null);
// 2.设置参数
mAsr.setParameter(SpeechConstant.ENGINE_TYPE, "cloud");
mAsr.setParameter(SpeechConstant.SUBJECT, "asr");
// 3.开始识别
int ret = mAsr.startListening(mRecognizerListener);
if (ret != ErrorCode.SUCCESS) {
        Log.d(TAG,"识别失败,错误码: " + ret);
}
// 识别监听器
private RecognizerListener mRecognizerListener = new RecognizerListener() {
        // 音量变化
        public void onVolumeChanged(int volume, byte[] data) {}
        // 返回结果
        public void onResult(final RecognizerResult result, boolean isLast) {}
        // 开始说话
        public void onBeginOfSpeech() {}
        // 结束说话
        public void onEndOfSpeech() {}
        // 错误回调
        public void onError(SpeechError error) {}
        // 件回调
        public void onEvent(int eventType, int arg1, int arg2, Bundle obj) {}
};
```

5.1.2 终端级命令词识别

终端级在线命令词识别需要先在终端上构建语法文件，上传语法文件之后获得相应的 Grammer ID，以后每次使用识别功能前，设置该 Grammar ID 参数即可。其示例代码如下：

```
// 在线命令词识别，启用终端级语法
// ABNF语法示例
String mCloudGrammar = "#ABNF 1.0 UTF-8;
                        languagezh-CN;
                        mode voice;
                        root $main;
                        $main = $place1 到$place2 ;
                        $place1 = 北京 | 武汉 | 南京 | 天津 | 天京 | 东京;
                        $place2 = 上海 | 合肥; ";
// 1.创建SpeechRecognizer对象
SpeechRecognizer mAsr = SpeechRecognizer.createRecognizer(context, null);
// 2.构建语法文件
mAsr.setParameter(SpeechConstant.TEXT_ENCODING, "utf-8");
int ret = mAsr.buildGrammar("abnf",  mCloudGrammar,  mGrammarListener);
if (ret != ErrorCode.SUCCESS){
    Log.d(TAG,"语法构建失败,错误码： " + ret);
}else{
    Log.d(TAG,"语法构建成功");
}
// 3.设置参数
mAsr.setParameter(SpeechConstant.ENGINE_TYPE, "cloud");
mAsr.setParameter(SpeechConstant.CLOUD_GRAMMAR, grammarId);
// 4.开始识别,
ret = mAsr.startListening(mRecognizerListener);
if (ret != ErrorCode.SUCCESS) {
    Log.d(TAG,"识别失败,错误码: " + ret);
}
//构建语法监听器
private GrammarListener mGrammarListener = new GrammarListener() {
    @Override
    public void onBuildFinish(String grammarId, SpeechError error) {
        if(error == null){
            if(!TextUtils.isEmpty(grammarId)){
                //构建语法成功，请保存grammarId用于识别
            }else{
                Log.d(TAG,"语法构建失败,错误码： " + error.getErrorCode());
            }
        }
    }
};
```

5.2.离线命令词识别

离线命令词识别，请参考 8.4 本地识别章节的相关内容。

6. 语音合成

与语音听写相反，合成是将文字信息转化为可听的声音信息，让机器像人一样开口说话。合成的调用方法如下：

```
//1.创建 SpeechSynthesizer 对象，第二个参数：本地合成时传 InitListener
SpeechSynthesizer mTts= SpeechSynthesizer.createSynthesizer(context, null);
//2.合成参数设置，详见《MSC Reference Manual》SpeechSynthesizer 类
//设置发音人（更多在线发音人，开发者可参见 附录13.2 ）
mTts.setParameter(SpeechConstant.VOICE_NAME, "xiaoyan");//设置发音人
mTts.setParameter(SpeechConstant.SPEED, "50");//设置语速
mTts.setParameter(SpeechConstant.VOLUME, "80");//设置音量，范围 0~100
mTts.setParameter(SpeechConstant.ENGINE_TYPE, SpeechConstant.TYPE_CLOUD); //设置云端
//设置合成音频保存位置（可自定义保存位置），保存在“./sdcard/iflytek.pcm”
//保存在 SD 卡需要在 AndroidManifest.xml 添加写 SD 卡权限
//仅支持保存为 pcm 和 wav 格式，如果不需要保存合成音频，注释该行代码
mTts.setParameter(SpeechConstant.TTS_AUDIO_PATH, "./sdcard/iflytek.pcm");
//3.开始合成
mTts.startSpeaking("科大讯飞，让世界聆听我们的声音", mSynListener);

//合成监听器。
private SynthesizerListener mSynListener = new SynthesizerListener(){
    //会话结束回调接口，没有错误时，error为null
    public void onCompleted(SpeechError error) {}
    //缓冲进度回调
    //percent为缓冲进度0~100，beginPos为缓冲音频在文本中开始位置，endPos表示缓冲音频在文本中结束位置，info为附加信息。
    public void onBufferProgress(int percent, int beginPos, int endPos, String info) {}
    //开始播放
    public void onSpeakBegin() {}
    //暂停播放
    public void onSpeakPaused() {}
    //播放进度回调
    //percent为播放进度0~100,beginPos为播放音频在文本中开始位置，endPos表示播放音频在文本中结束位置.
    public void onSpeakProgress(int percent, int beginPos, int endPos) {}
    //恢复播放回调接口
    public void onSpeakResumed() {}
    //会话事件回调接口
    public void onEvent(int arg0, int arg1, int arg2, Bundle arg3) {}
};
```

7. 语义理解

7.1.语音语义理解

您可以通过后台配置出一套您专属的语义结果，详见 http://osp.voicecloud.cn/

```
//1.创建文本语义理解对象
SpeechUnderstander understander = SpeechUnderstander.createUnderstander(context, null);
//2.设置参数，语义场景配置请登录 http://osp.voicecloud.cn/
understander.setParameter(SpeechConstant.LANGUAGE, "zh_cn");
//3.开始语义理解
understander.startUnderstanding(mUnderstanderListener);
// XmlParser为结果解析类，请参照Demo
private SpeechUnderstanderListener mUnderstanderListener = new SpeechUnderstanderListener(){
    public void onResult(UnderstanderResult result) {
            String text = result.getResultString();
    }
    public void onError(SpeechError error) {}//会话发生错误回调接口
    public void onBeginOfSpeech() {}//开始录音
    public void onVolumeChanged(int volume, byte[] data){} //volume音量值0~30，data音频数据
    public void onEndOfSpeech() {}//结束录音
    public void onEvent(int eventType, int arg1, int arg2, Bundle obj) {}//扩展用接口
};
```

7.2.文本语义理解

用户通过输入文本获取语义结果，文本语义结果和上述语音的方式相同。

```
//创建文本语义理解对象
TextUnderstander mTextUnderstander = TextUnderstander.createTextUnderstander(this, null);
//开始语义理解
mTextUnderstander.understandText("科大讯飞", searchListener);
//初始化监听器
TextUnderstanderListener searchListener = new TextUnderstanderListener(){
    //语义结果回调
    public void onResult(UnderstanderResult result){}
    //语义错误回调
    public void onError(SpeechError error) {}
};
```

8. 本地功能集成

在导入和加载本地资源文件后，通过设置就可以使用 V5+的本地功能。

8.1.导入资源文件

MSC V5+支持本地听写、识别和合成等语音功能，使用时需要导入本地资源文件（.jet 后缀），资源可以存放在以下三个位置：Assets、Resources、SD 卡。下图以 Assets 方式为例：

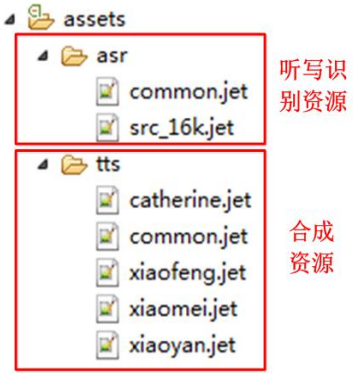

图 5 资源列表

8.2.本地资源加载

本地资源可以在不同阶段加载，且资源只需加载一次。以听写为例：

1) createUtility 初始化时

```
StringBuffer param = new StringBuffer();
//加载识别本地资源，resPath为本地识别资源路径
param.append(”,”+ResourceUtil.ASR_RES_PATH+”=”+resPath);
param.append(”,”+ResourceUtil.ENGINE_START+”=”+SpeechConstant.ENG_ASR);
param.append(”;”+ SpeechConstant.APPID+"=12345678");
SpeechUtility.createUtility(context, param);
```

2) setParameter 方式

```
StringBuffer param =new StringBuffer();
//加载识别本地资源，resPath为本地识别资源路径
param.append(ResourceUtil.ASR_RES_PATH+”=”+resPath);
param.append(”,”+ResourceUtil.ENGINE_START+”=”+SpeechConstant.ENG_ASR);
SpeechUtility.getUtility().setParameter(ResourceUtil.ENGINE_START,param);
```

3) 会话开始时

```
//加载识别本地资源，mIat为听写对象，resPath为本地识别资源路径
mIat.setParameter(ResourceUtil.ASR_RES_PATH, resPath);
//设置引擎类型为本地
mIat.setParameter(SpeechConstant.ENGINE_TYPE, SpeechConstant.TYPE_LOCAL);
mIat.startListening(recognizerListener);
```

其中上述代码中 resPath 指本地资源路径，因在 Android 中资源存放路径不同，其导入方式有三种：

a) Assets

```
String resPath = ResourceUtil.generateResourcePath(mContext, RESOURCE_TYPE.assets,
"asr/common.jet")+ ";" + ResourceUtil.generateResourcePath(mContext, RESOURCE_TYPE.assets,
"asr/src_16k.jet");
```

b) Resources

本地资源文件以 Resources 方式导入时，需存放到新建工程的 res/raw 目录下。

```
String resPath = ResourceUtil.generateResourcePath(mContext, RESOURCE_TYPE.res,
String.valueOf(R.raw.asr_common)) + ";" + ResourceUtil.generateResourcePath(mContext,
RESOURCE_TYPE.res, String.valueOf(R.raw.asr_src_16k));
```

c) SD 卡

```
String resPath = ResourceUtil.generateResourcePath(mContext,
RESOURCE_TYPE.path,"/sdcard/msc/res/asr/common.jet")+ ";" +
ResourceUtil.generateResourcePath(mContext,RESOURCE_TYPE.path,"/sdcard/msc/res/asr/src_16k.j
et");
```

8.3.本地引擎销毁

本地引擎销毁方式有两种，一种是注销方式，所有的本地引擎都会被销毁；另一种是指定资源销毁方式，只会销毁指定的引擎。代码示例如下：

1) 注销方式

```
SpeechUtility.getUtility().destroy();
```

2) 指定引擎销毁方式

```
SpeechUtility.getUtility().setParameter("engine_destroy", engType);
```

其中 engType 的取值有“asr”（识别引擎，即 SpeechConstant.ENG_ASR）、“tts”（合成引擎，即 SpeechConstant.ENG_TTS）和“ivw”（唤醒引擎，即 SpeechConstant.ENG_IVW）。

8.4.本地识别

利用本地引擎进行识别，需要将引擎类型设为本地，调用方法与在线识别差不多。

```
//1.创建 SpeechRecognizer 对象，需传入初始化监听器
SpeechRecognizer mAsr = SpeechRecognizer.createRecognizer(context, null);
//2.构建语法（本地识别引擎目前仅支持 BNF 语法），方法同在线语法识别，详见 Demo
//3.开始识别,设置引擎类型为本地
mAsr.setParameter(SpeechConstant.ENGINE_TYPE, SpeechConstant.TYPE_LOCAL);
//设置本地识别使用语法 id(此 id 在语法文件中定义)、门限值
mAsr.setParameter(SpeechConstant.LOCAL_GRAMMAR, "call");
mAsr.setParameter(SpeechConstant.ASR_THRESHOLD, "30");
//设置语法构建生成路径，buildpath是生成路径，可以设置在sd卡内
mAsr.setParameter(ResourceUtil.GRM_BUILD_PATH, buildpath);
mAsr.setParameter(ResourceUtil.ASR_RES_PATH, resPath);
ret = mAsr.startListening(mRecognizerListener);
```

8.5.本地合成

利用本地引擎合成语音，需要将引擎类型设为本地。

```
//1.创建 SpeechSynthesizer 对象
SpeechSynthesizer mTts= SpeechSynthesizer.createSynthesizer(context, null);
//2.合成参数设置
//设置引擎类型为本地
mTts.setParameter(SpeechConstant.ENGINE_TYPE, SpeechConstant.TYPE_LOCAL);
//设置本地发音人
mTts.setParameter(SpeechConstant.VOICE_NAME, "xiaoyan");
加载本地合成资源，resPath为本地合成资源路径
mTts.setParameter(ResourceUtil.TTS_RES_PATH, resPath);
//设置合成音频保存位置（可自定义保存位置），保存在“./sdcard/iflytek.pcm”
//保存在 SD 卡需要在 AndroidManifest.xml 添加写 SD 卡权限
//如果不需要保存合成音频，注释该行代码
mTts.setParameter(SpeechConstant.TTS_AUDIO_PATH, "./sdcard/iflytek.pcm");
//3.开始合成
mTts.startSpeaking("科大讯飞，让世界聆听我们的声音", mSynListener);
```

8.6.混合模式

混合能力是 MSC 的功能亮点，能够在较差的网络环境下，选择最优的方式进行语音交互。使用混合模式时，只需将引擎设置为 mixed。

混合模式包含两种，一种是**实时模式**，即同时向云端和本地发送音频，当云端返回结果超时时，返回本地结果；另一种是**延时模式**，先向云端发送音频，当云端返回结果超时后自动转为向本地发送音频的模式；其对应的参数设置分别为 mixed_type：realtime（实时）、delay（延时）。其中，混合模式超时的时间设置参数为 mixed_timeout，单位为 ms。具体调用代码如下：

```
//1.按照 Asrdemo 构建语法、设置本地语法路径资源等。
//2.设置引擎为混合模式：
mAsr.setParameter(SpeechConstant.ENGINE_TYPE, "mix");
//3.设置在线语义参数：
//是否进行语义识别
mAsr.setParameter("asr_sch", "1");
//通过此参数，设置开放语义协议版本号
mAsr.setParameter(SpeechConstant.NLP_VERSION, "2.0");
//返回文本结果类型
mAsr.setParameter(SpeechConstant.RESULT_TYPE, "json");
//混合模式的类型
//1)realtime 实时返回本地结果，优先使用本地识别引擎
//2)delay 延时等待云端结果，即等待云端超时返回本地结果，优先使用云端语义
mAsr.setParameter("mixed_type", "realtime");
//混合超时，仅在在 delay 类型下生效
//延时等待云端语义，若云端语义超时返回本地识别结果
mAsr.setParameter("mixed_timeout", "2500");
//4.启动识别
mAsr.startListening(mListener);
```

9. 语音评测

提供汉语、英语两种语言的评测，支持单字（汉语专有）、词语和句子三种题型，通过简单地接口调用就可以集成到您的应用中。语音评测的使用主要有三个步骤：

1) 创建对象和设置参数

```
// 创建评测对象
SpeechEvaluator mSpeechEvaluator = SpeechEvaluator.createEvaluator(
    IseDemoActivity.this, null);
// 设置评测语种
mSpeechEvaluator.setParameter(SpeechConstant.LANGUAGE, "en_us");
// 设置评测种类
mSpeechEvaluator.setParameter(SpeechConstant.ISE_CATEGORY, "read_word");
// 设置试题编码类型
mSpeechEvaluator.setParameter(SpeechConstant.TEXT_ENCODING, "utf-8");
// 设置前、后端点超时
mSpeechEvaluator.setParameter(SpeechConstant.VAD_BOS, vad_bos);
mSpeechEvaluator.setParameter(SpeechConstant.VAD_EOS, vad_eos);
// 设置录音超时，设置成-1 则无超时限制
mSpeechEvaluator.setParameter(SpeechConstant.KEY_SPEECH_TIMEOUT, "-1");
// 设置结果等级，不同等级对应不同的详细程度
mSpeechEvaluator.setParameter(SpeechConstant.RESULT_LEVEL, "complete");
```

可通过 setParameter 设置的评测相关参数说明如下：

表 3 评测相关参数说明

参数	说明	是否必需
language	评测语种，可选值：en_us（英语）、zh_cn（汉语）。	是
category	评测类型，可选值：read_syllable（单字，汉语专有）、read_word（词语）、read_sentence（句子）。	是
text_encoding	上传的试题编码格式，可选值：gb2312、utf-8。当进行汉语评测时，必须设置成 utf-8，建议所有试题都使用 utf-8 编码。	是
vad_bos	前端点超时，默认 5000ms。	否
vad_eos	后端点超时，默认 1800ms。	否
speech_timeout	录音超时，当录音达到时限将自动触发 vad 停止录音，默认-1（无超时）。	否
result_level	评测结果等级，可选值：plain、complete，默认为 complete	否

2) 上传评测试题和录音

```
// 首先创建一个评测监听接口
private EvaluatorListener mEvaluatorListener = new EvaluatorListener() {
    // 结果回调，评测过程中可能会多次调用该方法，isLast为true则为最后结果
    public void onResult(EvaluatorResult result, boolean isLast) {}
    // 出错回调
    public void onError(SpeechError error) {}
    // 开始说话回调
    public void onBeginOfSpeech() {}
    // 说话结束回调
    public void onEndOfSpeech() {}
    //volume音量值0~30，data音频数据
    public void onVolumeChanged(int volume, byte[] data){}
    // 扩展接口，暂时没有回调
    public void onEvent(int eventType, int arg1, int arg2, Bundle obj) {}
};

// 然后设置评测试题、传入监听器，开始评测录音。evaText为试题内容，试题格式详见《语音
// 评测参数、结果说明文档》，第二个参数为扩展参数，请设置为null
mSpeechEvaluator.startEvaluating(evaText, null, mEvaluatorListener);
```

调用 startEvaluating 即开始评测录音，读完试题内容后可以调用 stopEvaluating 停止录音，也可以在一段时间后由 SDK 自动检测 vad 并停止录音。当评测出错时，SDK 会回调 onError 方法抛出 SpeechError 错误，通过 SpeechError 的 getErrorCode()方法可获得错误码，常见的错误码详见附录 13.3 和下表：

表 4 评测错误码

错误码	错误值	含义
MSP_ERROR_ASE_EXCEP_SILENCE	11401	无语音或音量太小
MSP_ERROR_ASE_EXCEP_SNRATIO	11402	信噪比低或有效语音过短
MSP_ERROR_ASE_EXCEP_PAPERDATA	11403	非试卷数据
MSP_ERROR_ASE_EXCEP_PAPERCONTENTS	11404	试卷内容有误
MSP_ERROR_ASE_EXCEP_NOTMONO	11405	录音格式有误
MSP_ERROR_ASE_EXCEP_OTHERS	11406	其他评测数据异常，包括错读、漏读、恶意录入、试卷内容等错误
MSP_ERROR_ASE_EXCEP_PAPERFMT	11407	试卷格式有误
MSP_ERROR_ASE_EXCEP_ULISTWORD	11408	存在未登录词，即引擎中没有该词语的信息

3) 解析评测结果

SDK 通过 onResult 回调抛出 xml 格式的评测结果，结果格式及字段含义详见《语音评测参数、结果说明文档》文档，具体的解析过程可参考 demo 工程 com.iflytek.ise.result 包中的源代码。

10.唤醒功能集成

10.1.导入资源文件

使用唤醒功能需要将开发包中\res\ivw\路径下的唤醒资源文件引入，引入方式有三种 1：Assets、Resources、SD 卡；资源文件以.jet 为后缀（下图以 Assets 方式为例）。

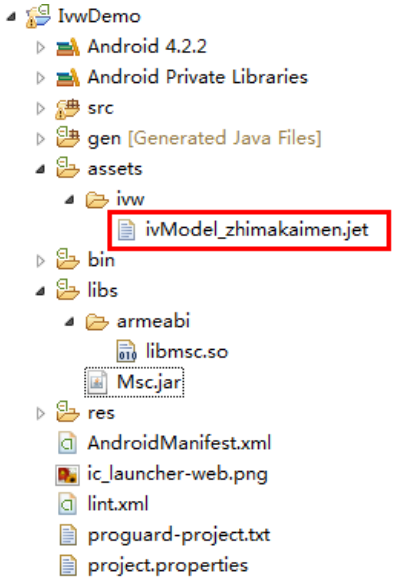

图 6 唤醒资源文件

10.2.初始化

创建用户语音配置对象后才可以使用语音服务，建议在程序入口处调用。

```
// 将“12345678”替换成您申请的 APPID，申请地址：http://www.xfyun.cn
SpeechUtility.createUtility(context, SpeechConstant.APPID +"=12345678");
```

10.3.代码集成

```
//1. 创建唤醒对象
VoiceWakeuper mIvw = VoiceWakeuper.createWakeuper(context, null);
//2.获取唤醒词路径，resPath为唤醒资源路径
String resPath = ResourceUtil.generateResourcePath(WakeDemo.this, RESOURCE_TYPE.assets,
"ivw/" + getString(R.string.app_id) + ".jet");
```

```
//3.设置唤醒参数，详见《iFlytek MSC Reference Manual》SpeechConstant类
//设置唤醒词加载路径
mIvw.setParameter(SpeechConstant.IVW_RES_PATH, resPath);
//唤醒门限值，根据资源携带的唤醒词个数按照“id:门限;id:门限”的格式传入
mIvw.setParameter(SpeechConstant.IVW_THRESHOLD,"0:"+curThresh);
//设置当前业务类型为唤醒
mIvw.setParameter(SpeechConstant.IVW_SST,"wakeup");
//设置唤醒一直保持，直到调用stopListening，传入0则完成一次唤醒后，会话立即结束（默认0）
mIvw.setParameter(SpeechConstant.KEEP_ALIVE,"1");
//4.开始唤醒
mIvw.startListening(mWakeuperListener);
//唤醒监听器
private WakeuperListener mWakeuperListener = new WakeuperListener() {
        public void onResult(WakeuperResult result) {
            //唤醒结果回调接口(返回Json格式结果，开发者可参见 附件13.4)
            try {
                String text = result.getResultString();
            } catch (JSONException e) {
                e.printStackTrace();
        }}
        public void onError(SpeechError error) {}
        public void onBeginOfSpeech() {}
        public void onEvent(int eventType, int arg1, int arg2, Bundle obj) {
            if (SpeechEvent.EVENT_IVW_RESULT == eventType) {
            //当使用唤醒+识别功能时获取识别结果
            //arg1:是否最后一个结果，1:是，0:否。
            RecognizerResult reslut =
        ((RecognizerResult)obj.get(SpeechEvent.KEY_EVENT_IVW_RESULT));
        }}
};
```

10.4.唤醒+识别

```
//使用唤醒+识别功能需要设置如下参数
//唤醒门限值，根据资源携带的唤醒词个数按照“id:门限;id:门限”的格式传入
mIvw.setParameter(SpeechConstant.IVW_THRESHOLD,"0:"+curThresh);
//设置/当前业务类型为唤醒+识别
mIvw.setParameter(SpeechConstant.IVW_SST,"oneshot");
//设置识别引擎
mIvw.setParameter(SpeechConstant.ENGINE_TYPE, "cloud");
//设置云端识别使用的语法id
mIvw.setParameter(SpeechConstant.CLOUD_GRAMMAR, grammarID);
```

10.5.闭环优化功能

闭环优化是针对开发者的唤醒资源由云端优化系统不断优化的功能。通过开发者 APP 使用场景，本地唤醒挑选音频数据上传至云端，进行训练生成优化唤醒资源。开发者 APP 使用场景中，优化唤醒资源在相比原有资源在提升唤醒率及抑制误唤醒方面有良好的表现。持续优化包含三种网络模式：

1） **模式 0**：关闭优化功能，禁止向服务端发送本地挑选数据及启动唤醒时进行查询下载；

2） **模式 1**：开启优化功能，允许向服务端发送本地挑选数据，但是禁止在启动唤醒时进行查询下载。该模式需要开发者自行进行优化资源的查询下载，并且管理对资源的使用进行管理；

3） **模式 2**：开启优化功能，允许向服务端发送本地挑选数据及启动唤醒时进行查询下载。模式 2 下 Msc 会通过启动唤醒时进行查询下载，并且为开发者的应用管理唤醒优化资源。在不需要开发者介入的情况下，完成持续优化功能；

```
//1.创建唤醒对象
//2.设置唤醒参数，具体请参考 代码集成
//3.设置唤醒持续优化网络模式为模式2
mIvw.setParameter(SpeechConstant.IVW_NET_MODE, "2");
//4.开始唤醒
mIvw.startListening(mWakeuperListener);
```

在**模式 1** 的情况下，Msc 提供相应主动请求及下载唤醒资源的接口。开发者可以通过查询接口获取优化资源的下载链接，下载并替换本地资源，提升唤醒效果。具体代码调用如下：

```
//1.创建唤醒对象，获取唤醒资源路径
//2.启动唤醒资源查询
mIvw.queryResource(resPath,requestListener);
//资源查询回调
private RequestListener requestListener = new RequestListener() {
        public void onEvent(int eventType, Bundle params) {}
        public void onCompleted(SpeechError error) {}
        public void onBufferReceived(byte[] buffer) {
            //查询结果回调接口(返回Json格式结果，开发者可参见 附件13.4)
}};
//3.根据查询结果中下载链接及Md5等信息，详情请参考示例代码
//4.启动唤醒资源下载
mIvw.downloadResource(downloadUri, filePath, downloadMd5, wakeListener);
//文件下载回调
private FileDownloadListener wakeListener = new FileDownloadListener() {
        public void onCompleted(String filePath, SpeechError error) {}
        public void onStart() {}
        public void onProgress(int percent) {}
};
```

注：闭环优化功能从 Msc SDK 1091 开始支持，持续优化功能会产生一定的网络流量，建议 wifi 环境下开启。

11.声纹密码

与指纹一样，声纹也是一种独一无二的生理特征，可以用来鉴别用户的身份。声纹密码的使用包括注册、验证和模型操作。

11.1. 声纹注册

现阶段语音云平台支持三种类型的声纹密码，即数字密码及文本密码，在注册之前要选择声纹的类型。注：文本密码的效果在优化中，建议使用数字密码。

```
// 首先创建SpeakerVerifier对象
mVerify = SpeakerVerifier.createVerifier(this, null);
// 通过setParameter设置密码类型，pwdType的取值为1、3，分别表示文本密码和数字密码
mVerify.setParameter(SpeechConstant.ISV_PWDT, "" + pwdType);
```

pwdType 的取值说明如下表所示：

表 5 pwdType 取值说明

取值	说明
1	文本密码。用户通过读出指定的文本内容来进行声纹注册和验证，现阶段支持的文本有“芝麻开门”。
3	数字密码。从云端拉取若干组特定的数字串（默认有 5 组，每组 8 位数字），用户依次读出这 5 组数字进行注册，在验证过程中会生成一串特定的数字，用户通过读出这串数字进行验证。

除自由说外，其他两种密码需调用接口从云端获取：

```
// 通过调用getPasswordList方法来获取密码。mPwdListener是一个回调接口，当获取到密码后，SDK会调用其中的onBufferReceived方法对云端返回的JSON格式（具体格式见附录13.5）的密码进行处理，处理方法详见声纹Demo示例
mVerify.getPasswordList(SpeechListener mPwdListener);
SpeechListener mPwdListenter = new SpeechListener() {
        public void onEvent(int eventType, Bundle params) {}
        public void onBufferReceived(byte[] buffer) {}
        public void onCompleted(SpeechError error) {}
};
```

获取到密码后，接下来进行声纹注册，即要求用户朗读若干次指定的内容，这一过程也称为声纹模型的训练。

```
// 设置业务类型为训练
mVerify.setParameter(SpeechConstant.ISV_SST, "train");
// 设置密码类型
mVerify.setParameter(SpeechConstant.ISV_PWDT, "" + pwdType);
// 对于文本密码和数字密码，必须设置密码的文本内容，pwdText的取值为“芝麻开门”或者是从云端拉取的数字密码(每8位用“-”隔开，如“62389704-45937680-32758406-29530846-58206497”)。
mVerify.setParameter(SpeechConstant.ISV_PWD, pwdText);
// 设置声纹模型对应的AUTH_ID，它是用户的唯一标识，为空时表示这是一个匿名用户
mVerify.setParameter(SpeechConstant.AUTH_ID, auth_id);
// 开始注册，当得到注册结果时，SDK会将其封装成VerifierResult对象，回调VerifierListener对象listener的onResult方法进行处理，处理方法详见Demo示例
mVerify.startListening(mRegisterListener);
VerifierListener mRegisterListener =new VerifierListener() {
        public void onVolumeChanged(int volume, byte [] data) {}
        public void onResult(VerifierResult result) {
        public void onEvent(int eventType, int arg1, int arg2, Bundle obj) {}
        public void onError(SpeechError error) {}
        public void onEndOfSpeech() {}
        public void onBeginOfSpeech() {}
};
```

注意，当 auth_id 为空时（匿名用户），将使用设备的设备 ID 来标识注册的声纹模型。由于设备 ID 不能跨设备，而且不同的设备所获取到的设备 ID 也有可能相同，推荐的作法是在注册模型的时为 app 的每个用户都指定一个唯一的 auth_id。auth_id 的格式为：6-18 个字符，为字母、数字和下划线的组合且必须以字母开头，不支持中文字符，不能包含空格。

开发者通过声明一个 VerifierListener 对象并重写 onResult 方法来处理注册结果。在结果 result 中携带了一个返回码（0 表示成功，-1 为失败）和错误码，用来判别注册是否成功以及出错原因，错误码的含义如下：

表 7 声纹错误码

错误码	错误值	说明
MSS_ERROR_IVP_GENERAL	11600	正常，请继续传音频
MSS_ERROR_IVP_EXTRA_RGN_SOPPORT	11601	rgn 超过最大支持次数 9
MSS_ERROR_IVP_TRUNCATED	11602	音频波形幅度太大，超出系统范围，发生截幅
MSS_ERROR_IVP_MUCH_NOISE	11603	太多噪音
MSS_ERROR_IVP_TOO_LOW	11604	声音太小
MSS_ERROR_IVP_ZERO_AUDIO	11605	没检测到音频
MSS_ERROR_IVP_UTTER_TOO_SHORT	11606	音频太短
MSS_ERROR_IVP_TEXT_NOT_MATCH	11607	音频内容与给定文本不一致
MSS_ERROR_IVP_NO_ENOUGH_AUDIO	11608	音频长达不到自由说的要求

验证结果 VerifierResult 类中有一个 vid 字段，用于标识成功注册的声注模型。结果中包含的所有字段，以及各字段的含义见 附录 13.5。

11.2. 声纹验证

声纹验证过程与声纹注册类似，不同之处仅在于 ISV_SST 需要设置为“verify”，且不用设置 ISV_RGN 参数，其他参数的设置、验证结果的处理过程完全可参考上一节。

另外，为了达到较好的效果，请在声纹注册与验证过程中尽量与麦克风保持同样的距离（建议的最佳距离是 15 厘米左右）。如果距离差距较大的话，可能会对验证通过率产生较大影响。

11.3. 模型操作

声纹注册成功后，在语音云端上会生成一个对应的模型来存储声纹信息，声纹模型的操作即对模型进行查询和删除。

```
// 首先设置声纹密码类型
mVerify.setParameter(SpeechConstant.ISV_PWDT, "" + pwdType);
// 对于文本和数字密码，必须设置声纹注册时用的密码文本，pwdText的取值为“芝麻开门”或者是从语音云平台请求的数字密码。
mVerify.setParameter(SpeechConstant.ISV_PWD, pwdText);
// 设置待操作的声纹模型的vid
mVerify.setParameter(SpeechConstant.ISV_VID, vid);
// 调用sendRequest方法查询或者删除模型，cmd的取值为“que”或“del”，表示查询或者删除，auth_id是声纹对应的用户标识，操作结果以异步方式回调SpeechListener类型对象listener的onBufferReceived方法进行处理，处理方法详见Demo示例
mVerify.sendRequest(cmd, auth_id, listener);
```

12.人脸识别

人脸识别不仅可以检测出照片中的人脸，还可以进行人脸注册和验证。相关概念的说明如下：

表 8 人脸识别概念说明

名称	说明
reg/注册	上传包含一张人脸的图片到云端，引擎对其进行特征抽取，生成一个与之对应的模型，返回模型 id(gid)。
verify/验证	注册成功后，上传包含一张人脸的图片到云端，引擎将其与所注册的人脸模型进行比对，验证是否为同一个人，返回验证结果。
detect/检测	上传一张图片，返回该图片中人脸的位置（支持多张人脸）。
align/聚焦	上传一张图片，返回该图片中人脸的关键点坐标（支持多张人脸）。
auth_id/用户 id	由应用传入，用于标识用户身份，长度为 6-18 个字符（由英文字母、数字、下划线组成，不能以数字开头），不支持中文字符。 注：注册和验证都必须指定 auth_id。

为了获得较高的准确率，请确保输入的图片满足以下要求：

表 9 上传图片规格要求

项目	要求
色彩、格式	彩色，PNG、JPG、BMP 格式的图片。
人脸大小、角度	大小应超过 100*100 像素，可以容忍一定程度的侧脸，为保证识别准确率，最好使用正脸图片。
光照	均匀光照，可容忍部分阴影。
遮挡物	脸部尽量无遮挡，眼镜等物品会一定程度上影响准确率。

12.1.人脸注册

```
// 使用FaceRequest(Context context)构造一个FaceRequest对象
FaceRequest face = new FaceRequest(this);
// 设置业务类型为注册
face.setParameter(SpeechConstant.WFR_SST, "reg");
// 设置auth_id
face.setParameter(SpeechConstant. AUTH_ID, mAuthId);
// 调用sendRequest(byte[] img, RequestListener listener)方法发送注册请求，img为图片的二进制数据，listener为处理注册结果的回调对象
face.sendRequest(imgData, mRequestListener);
```

回调对象 mRequestListener 的定义如下：

```
RequestListener mRequestListener = new RequestListener() {
    // 获得结果时返回，JSON格式。
    public void onBufferReceived(byte[] buffer) {}
    // 流程结束时返回，error不为空则表示发生错误。
    public void onCompleted(SpeechError error) {}
    // 保留接口，扩展用。
    public void onEvent(int eventType, Bundle params) {}
}
```

12.2.人脸验证

```
// 设置业务类型为验证
face.setParameter(SpeechConstant.WFR_SST, "verify");
// 设置auth_id
face.setParameter(SpeechConstant. AUTH_ID, mAuthId);
// 调用sendRequest(byte[] img, RequestListener listener)方法发送注册请求，img为图片的二进制数据，listener为处理注册结果的回调对象
face.sendRequest(imgData, mRequestListener);
```

注册/验证结果中包含了是否成功、gid 等信息，详细的 JSON 格式请参照 附录 13.6，具体解析过程详见 FaceDemo 示例。

12.3.人脸检测

```
// 设置业务类型为验证
face.setParameter(SpeechConstant.WFR_SST, "detect");
// 调用sendRequest(byte[] img, RequestListener listener)方法发送注册请求，img为图片的二进制数据，listener为处理注册结果的回调对象
face.sendRequest(imgData, mRequestListener);
```

12.4.人脸聚焦

```
// 设置业务类型为验证
face.setParameter(SpeechConstant.WFR_SST, "align");
// 调用sendRequest(byte[] img, RequestListener listener)方法发送注册请求，img为图片的二进制数据，listener为处理注册结果的回调对象
face.sendRequest(imgData, mRequestListener);
```

13.附录

13.1.识别结果说明

JSON 字段	英文全称	类型	说明
sn	sentence	number	第几句
ls	last sentence	boolean	是否最后一句
bg	begin	number	开始
ed	end	number	结束
ws	words	array	词
cw	chinese word	array	中文分词
w	word	string	单字
sc	score	number	分数

听写结果示例：

```
{"sn":1,"ls":true,"bg":0,"ed":0,"ws":[
{"bg":0,"cw":[{"w":"今天","sc":0}]},
{"bg":0,"cw":[{"w":"的","sc":0}]},
{"bg":0,"cw":[{"w":"天气","sc":0}]},
{"bg":0,"cw":[{"w":"怎么样","sc":0}]},
{"bg":0,"cw":[{"w":"。","sc":0}]}]}
```

多候选结果示例：

```
{"sn":1,"ls":false,"bg":0,"ed":0,"ws":[
{"bg":0,"cw":[{"w":"我想听","sc":0}]},
{"bg":0,"cw":[{"w":"拉德斯基进行曲","sc":0},{"w":"拉得斯进行曲","sc":0}]}]}
```

语法识别结果示例：

```
{"sn":1,"ls":true,"bg":0,"ed":0,"ws":[
        {"bg":0,"cw":[{"sc":"70","gm":"0","w":"北京到上海"},
                {"sc":"69","gm":"0","w":"天京到上海"},
                {"sc":"58","gm":"0","w":"东京到上海"}]}]}
```

13.2.合成发音人列表

1、 语言为中英文的发音人可以支持中英文的混合朗读。
2、 英文发音人只能朗读英文，中文无法朗读。
3、 汉语发音人只能朗读中文，遇到英文会以单个字母的方式进行朗读。
4、 使用**新引擎参数**会获得更好的合成效果。

发音人名称	属性	语言	参数名称	新引擎参数	备注
小燕	青年女声	中英文（普通话）	xiaoyan		默认
小宇	青年男声	中英文（普通话）	xiaoyu		
凯瑟琳	青年女声	英文	catherine		
亨利	青年男声	英文	henry		
玛丽	青年女声	英文	vimary		
小研	青年女声	中英文（普通话）	vixy		
小琪	青年女声	中英文（普通话）	vixq	xiaoqi	
小峰	青年男声	中英文（普通话）	vixf		
小梅	青年女声	中英文（粤语）	vixm	xiaomei	
小莉	青年女声	中英文(台湾普通话)	vixl	xiaolin	
小蓉	青年女声	汉语（四川话）	vixr	xiaorong	
小芸	青年女声	汉语（东北话）	vixyun	xiaoqian	
小坤	青年男声	汉语（河南话）	vixk	xiaokun	
小强	青年男声	汉语（湖南话）	vixqa	xiaoqiang	
小莹	青年女声	汉语（陕西话）	vixying		
小新	童年男声	汉语（普通话）	vixx	xiaoxin	
楠楠	童年女声	汉语（普通话）	vinn	nannan	
老孙	老年男声	汉语（普通话）	vils		
Mariane		法语	Mariane		
Allabent		俄语	Allabent		
Gabriela		西班牙语	Gabriela		
Abha		印地语	Abha		
XiaoYun		越南语	XiaoYun		

13.3.错误码列表

1、10000~19999 的错误码参见 MSC 错误码链接。
2、其它错误码参见下表

错误码	错误值	含义
ERROR_NO_NETWORK	20001	无有效的网络连接
ERROR_NETWORK_TIMEOUT	20002	网络连接超时
ERROR_NET_EXPECTION	20003	网络连接发生异常
ERROR_INVALID_RESULT	20004	无有效的结果
ERROR_NO_MATCH	20005	无匹配结果
ERROR_AUDIO_RECORD	20006	录音失败
ERROR_NO_SPPECH	20007	未检测到语音
ERROR_SPEECH_TIMEOUT	20008	音频输入超时
ERROR_EMPTY_UTTERANCE	20009	无效的文本输入
ERROR_FILE_ACCESS	20010	文件读写失败
ERROR_PLAY_MEDIA	20011	音频播放失败
ERROR_INVALID_PARAM	20012	无效的参数
ERROR_TEXT_OVERFLOW	20013	文本溢出
ERROR_INVALID_DATA	20014	无效数据
ERROR_LOGIN	20015	用户未登陆
ERROR_PERMISSION_DENIED	20016	无效授权
ERROR_INTERRUPT	20017	被异常打断
ERROR_VERSION_LOWER	20018	版本过低
ERROR_COMPONENT_NOT_INSTALLED	21001	没有安装语音组件
ERROR_ENGINE_NOT_SUPPORTED	21002	引擎不支持
ERROR_ENGINE_INIT_FAIL	21003	初始化失败
ERROR_ENGINE_CALL_FAIL	21004	调用失败
ERROR_ENGINE_BUSY	21005	引擎繁忙
ERROR_LOCAL_NO_INIT	22001	本地引擎未初始化
ERROR_LOCAL_RESOURCE	22002	本地引擎无资源
ERROR_LOCAL_ENGINE	22003	本地引擎内部错误
ERROR_IVW_INTERRUPT	22004	本地唤醒引擎被异常打断
ERROR_UNKNOWN	20999	未知错误

13.4.唤醒业务结果

唤醒结果（WakeuperResult）字段说明：

成员名	参数解释
sst	本次业务标识：wakeup 表示语音唤醒；oneshot 表示唤醒+识别；
id	当前唤醒词的 id
score	当前唤醒得分
bos	当前唤醒音频的前端点
eos	当前唤醒音频的尾端点

唤醒资源查询结果字段说明：

成员名	参数解释
ret	本次请求是否成功：0 表示成功；非 0 表示失败，ret 为错误码；
resize	最新资源大小
dlurl	最新资源下载链接
md5	最新资源 md5 值，便于下载完成校验
sid	本地请求会话 id，便于问题查询

13.5.声纹业务

文本密码 JSON 示例

```
{"txt_pwd":["我的地盘我做主","移动改变生活","芝麻开门"]}
```

数字密码 JSON 示例

```
{"num_pwd":["03285469","09734658","53894276","57392804","68294073"]}
```

声纹业务结果（VerifierResult）成员说明

成员	说明
sst	业务类型，取值为 train 或 verify
ret	返回值，0 为成功，-1 为失败
vid	注册成功的声纹模型 id
score	当前声纹相似度

suc	本次注册已成功的训练次数
rgn	本次注册需要的训练次数
trs	注册完成描述信息
err	注册/验证返回的错误码
dcs	描述信息

13.6.人脸识别结果说明

JSON 字段	类型	说明
sst	String	业务类型，取值为“reg”或“verify”
ret	int	返回值，0 为成功，-1 为失败
rst	String	注册/验证成功
gid	String	注册成功的人脸模型 id
score	double	人脸验证的得分（验证时返回）
sid	String	本次交互会话的 id
uid	String	返回的用户 id

注册结果示例：

{"ret":"0","uid":"","rst":"success","gid":"wfr278b0092@hf9a6907805f269a2800","sid":"wfr278b0092@hf9a6907805f269a2800","sst":"reg"}

验证结果示例：

{"ret":"0","uid":"","sid":"wfr27830092@hf9a6907805fb19a2800","sst":"verify","score":"100.787","rst":"success","gid":"wfr278b0092@hf9a69"}

检测结果示例：

{"ret":"0","uid":"a12456952","rst":"success","face":[{"position":{"bottom":931,"right":766,"left":220,"top":385},"attribute":{"pose":{"pitch":1}},"tag":"","confidence":"
8.400"}],"sid":"wfr278f0004@hf9a6907bcc8c19a2800","sst":"detect"}

聚焦结果示例：

{"ret":"0","uid":"a1316826037","rst":"success","result":[{"landmark":{"right_eye_right_corner":{"y":"98.574","x":"127.327"},"left_eye_left_corner":{"y":"101.199","x":"40.101"},"right_eye_center":{"y":"98.090","x":"113.149"},"left_eyebrow_middle":{"y":"83.169","x":"46.642"},"right_eyebrow_left_corner":{"y":"85.135","x":"96.663"},"mouth_right_corner":{"y":"164.645","x":"109.419"},"mouth_left_corner":{"y":"166.419","x":"60.044"},"left_eyebrow_left_corner":{"y":"89.283","x":"28.029"},"right_eyebrow_middle":{"y":"80.991","x":"117.417"},"left_eye_center":{"y":"99.803","x":"53.267"},"nose_left":{"y":"137.397","x":"66.491"},"mouth_lower_lip_bo

ttom":{"y":"170.229","x":"86.013"},"nose_right":{"y":"136.968","x":"101.627"},"left_eyebrow_right_corner":{"y":"86.090","x":"68.351"},"right_eye_left_corner":{"y":"99.898","x":"100.736"},"nose_bottom":{"y":"144.465","x":"84.032"},"nose_top":{"y":"132.959","x":"83.074"},"mouth_middle":{"y":"164.466","x":"85.325"},"left_eye_right_corner":{"y":"101.043","x":"67.275"},"mouth_upper_lip_top":{"y":"159.418","x":"84.841"},"right_eyebrow_right_corner":{"y":"84.916","x":"136.423"}}}],"sid":"wfr278500ec@ch47fc07eb395d476f00","sst":"align"}

常见问题

(1). 集成语音识别功能时，程序启动后没反应？

答：请检查是否忘记使用 SpeechUtility 初始化。

也可以在监听器的 onError 函数中打印错误信息，根据信息提示，查找错误源。

```
public void onError(SpeechError error) {
    Log.d(error.toString());
}
```

(2). SDK 是否支持本地语音能力？

答：Android 平台 SDK 已经支持本地合成、本地命令词识别、本地听写以及语音唤醒功能了，声纹功能也即将上线。

(3). Appid 的使用规范？

答：申请的 Appid 和对应下载的 SDK 具有一致性，请确保在使用过程中规范传入。一个 Appid 对应一个平台下的一个应用，如在多个平台开发同款应用，还需申请对应平台的 Appid。

更多问题，请见：

http://www.xyun.cn/index.php/default/doccenter/doccenterInner?itemTitle=ZmFx&anchor=Y29udGl0bGU2Mw==

联系方式：

邮箱：msp_support@iflytek.com　　QQ 群：242973048