科大讯飞股份有限公司

iFLYTEK CO.,LTD

科大讯飞 MSC 新手指南

SDK Version: 4.5.1038.1048

Updated: 2014.05.15

1. 概述

本文档是开发科大讯飞 Android 语音程序的用户指南，定义了语音听写、语音识别、语音合成以及语义理解相关接口的使用说明和体系结构，如图 1 所示。

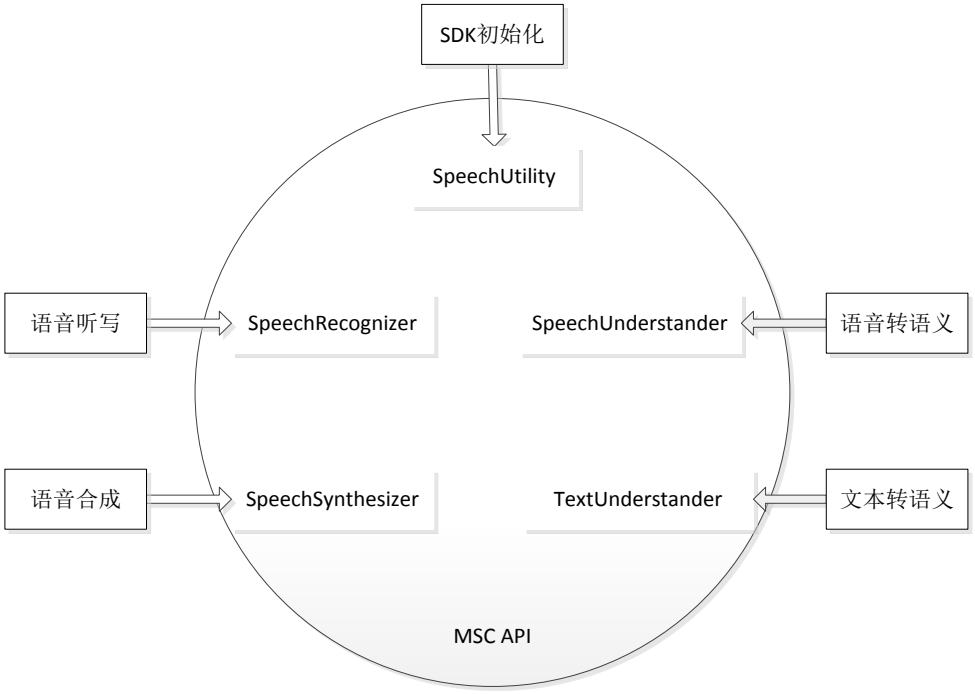


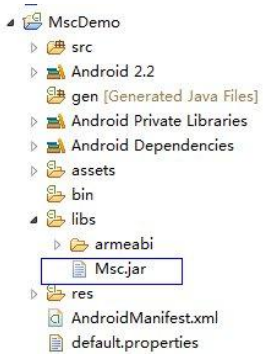
图 1 MSC 功能结构图

2. 集成说明

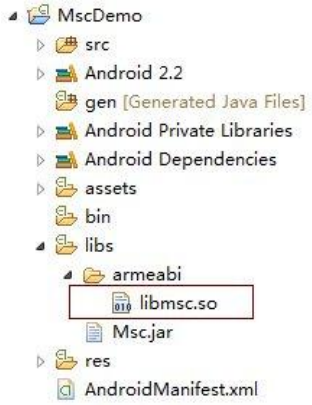
Step 1 导入 SDK

[1] 在 Eclipse 中建立你的 Android 工程。

[2] 将开发工具包中 libs 目录下的 Msc.jar 复制到新建工程的 libs 目录中（如下图所示）。



[3] 将开发工具包中 armeabi（armeabi-v7a,mips,x86）目录下的 libmsc.so 复制到新建工程的 armeabi 目录中（如下图所示）。



[4]在你需要使用 MSC 服务的文件中导入相应的类。

例如添加听写功能：import com.iflytek.cloud.SpeechRecognizer;

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"/>
```

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

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

Step 3 功能添加

[1] 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://open.voicecloud.cn
SpeechUtility.createUtility(context, SpeechConstant.APPID +"=12345678");
```

[2] 语音听写

主要指将连续语音快速识别为文字的过程，能识别通用常见的语句、词汇，不限制说法。

1.语音听写

```
//1.创建SpeechRecognizer对象，第二个参数：本地听写时传InitListener
SpeechRecognizer mIat= SpeechRecognizer.createRecognizer(context, null);
//2.设置听写参数，详见《科大讯飞MSC API手册(Android)》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);
    //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) {
        error.getPlainDescription(true) //获取错误码描述}
    //开始录音
    public void onBeginOfSpeech() {}
    //音量值0~30
    public void onVolumeChanged(int volume){}
    //结束录音
    public void onEndOfSpeech() {}
    //扩展用接口
    public void onEvent(int eventType,int arg1,int arg2,String msg) {}
};
```

2.上传联系人

上传联系人可以提高联系人名称识别率，也可以提高语义的效果，每个用户终端设备对应一个联系人列表，联系人格式详见《科大讯飞 MSC API 手册(Android)》**ContactManager** 类。

```
//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！");
        }
    }
};
```

3.上传用户词表

上传用户词表可以提高词表内词汇的识别率，也可以提高语义的效果，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《科大讯飞 MSC API 手册(Android)》**UserWords** 类。

```
//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！");
            }
      }
};

//下载用户词表
DataDownloader dataDownloader = new DataDownloader(context);
dataDownloader.setParameter(SpeechConstant.SUBJECT, "spp");
dataDownloader.setParameter(SpeechConstant.DATA_TYPE, "userword");
dataDownloader.downloadData(downloadlistener);
//用户词表下载监听器。
private SpeechListener downloadlistener = new SpeechListener(){
      //用户词表结果回调
      public void onData(byte[] data) {}
      //用户词表完成回调
      public void onCompleted(SpeechError error) {}
      //用户词表事件回调
```

[3] 语法识别

主要指基于命令词的识别，识别指定关键词组合的词汇，或者固定说法的短句。语法识别分云端识别和本地识别，云端和本地分别采用 ABNF 和 BNF 语法格式。

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

```
//云端语法识别：如需本地识别请参照3.本地功能集成(讯飞语音+)
//1.创建SpeechRecognizer对象
SpeechRecognizer mAsr = SpeechRecognizer.createRecognizer(context, null);
// ABNF语法示例，可以说”北京到上海”
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; ";
//2.构建语法文件
mAsr.setParameter(SpeechConstant.TEXT_ENCODING, "utf-8");
ret = mAsr.buildGrammar("abnf", mCloudGrammar ,  grammarListener);
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.开始识别,设置引擎类型为云端
mAsr.setParameter(SpeechConstant.ENGINE_TYPE, "cloud");
//设置grammarId
mAsr.setParameter(SpeechConstant.CLOUD_GRAMMAR, grammarId);
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 grammarListener = 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());
            }
        }};
```

[4] 语音合成

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。

```
//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《科大讯飞MSC API手册(Android)》SpeechSynthesizer 类
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
//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);

//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() {}
};
```

[5] 语义示例

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，见SpeechDemo
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){} //音量值0~30
	public void onEndOfSpeech() {}//结束录音
	public void onEvent(int eventType, int arg1, int arg2, String msg) {}//扩展用接口
};
```

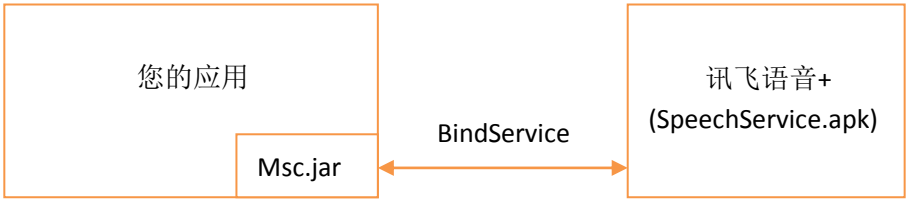
2. 文本语义理解

用户通过输入文本获取语义结果，专属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 = new TextUnderstander(context, mInitListener);
//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) {}
};
```

3. 本地功能集成(讯飞语音+)

本地识别、合成以及唤醒功能需要通过“讯飞语音+”来实现。“讯飞语音+”是基于讯飞语音云平台开发的应用，用户安装语音+后即可使用本地功能。



(1).检查语音+是否安装

```
//检查语音+是否安装
//如未安装，获取语音+下载地址进行下载。
if(!SpeechUtility.getUtility().checkServiceInstalled ()){
    String url = SpeechUtility.getUtility().getComponentUrl();
    Uri uri = Uri.parse(url);
    Intent it = new Intent(Intent.ACTION_VIEW, uri);
    context.startActivity(it);
}
//下载完成后调用，重新连接语音+，详见Demo。
SpeechUtility.getUtility().checkServiceInstalled();
```

(2).本地识别

```
//1.创建 SpeechRecognizer 对象，需传入初始化监听器
SpeechRecognizer mAsr = SpeechRecognizer.createRecognizer(context, mInitListener);
//初始化监听器，只有在使用本地语音服务时需要监听（即安装讯飞语音+，通过语音+提供本地服务），初始化成功后才可进行本地操作。
private InitListener mInitListener = new InitListener() {
    public void onInit(int code) {
        if (code == ErrorCode.SUCCESS) {}
    }
};
//2.上传语法文件（本地识别引擎目前仅支持 BNF 语法），同在线语法识别
//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.MIXED_THRESHOLD, "30");
ret = mAsr.startListening(mRecognizerListener);
```

(3).本地合成

```
//1.创建 SpeechSynthesizer 对象
SpeechSynthesizer mTts= SpeechSynthesizer.createSynthesizer(context, mInitListener);
//初始化监听器,同听写初始化监听器，使用云端的情况下不需要监听即可使用，本地需要监听
private InitListener mInitListener = new InitListener() {...};
//2.合成参数设置
//设置引擎类型为本地
mTts.setParameter(SpeechConstant.ENGINE_TYPE, SpeechConstant.TYPE_LOCAL);
//可跳转到语音+发音人设置页面进行发音人下载
SpeechUtility.getUtility().openEngineSettings(SpeechConstant.ENG_TTS);
//3.开始合成
mTts.startSpeaking("科大讯飞，让世界聆听我们的声音", mSynListener);
```

(4).唤醒、声纹（即将开放，敬请期待）

4. 附录

(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	socre	number	分数

转写结果示例：

```
{"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"}]}]}
```

(2).个性发音人列表

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。

发音人名称	属性	语言	参数名称	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	
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	
小莉	青年女声	中英文（台湾普通话）	vixl	
小蓉	青年女声	汉语（四川话）	vixr	
小芸	青年女声	汉语（东北话）	vixyun	
小坤	青年男声	汉语（河南话）	vixk	
小强	青年男声	汉语（湖南话）	vixqa	
小莹	青年女声	汉语（陕西话）	vixying	
小新	童年男声	汉语（普通话）	vixx	
楠楠	童年女声	汉语（普通话）	vinn	
老孙	老年男声	汉语（普通话）	vils	
Mariane		法语	Mariane	
Guli		维语	Guli	
Allabent		俄语	Allabent	
Gabriela		西班牙语	Gabriela	
Abha		印地语	Abha	
XiaoYun		越南语	XiaoYun	

(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	意义
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

5. 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。

也可以在转写监听器的 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 已经支持本地合成和本地命令词识别功能了，语音唤醒和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://open.voicecloud.cn/index.php/default/doccenter/doccenterInner?itemTitle=ZmFx&anchor=Y29udGl0bGU2Mw==

联系方式：
邮箱：msp_support@iflytek.com　　QQ 群：91104836，153789256