科大讯飞股份有限公司
iFLYTEK CO.,LTD

科大讯飞 MSC 新手指南

SDK Version: 1019

Updated: 2015.11.26

1. 概述

本文档是开发科大讯飞 Java 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。所有类和函数的详细使用说明，请参考《MSC Reference Manual》。

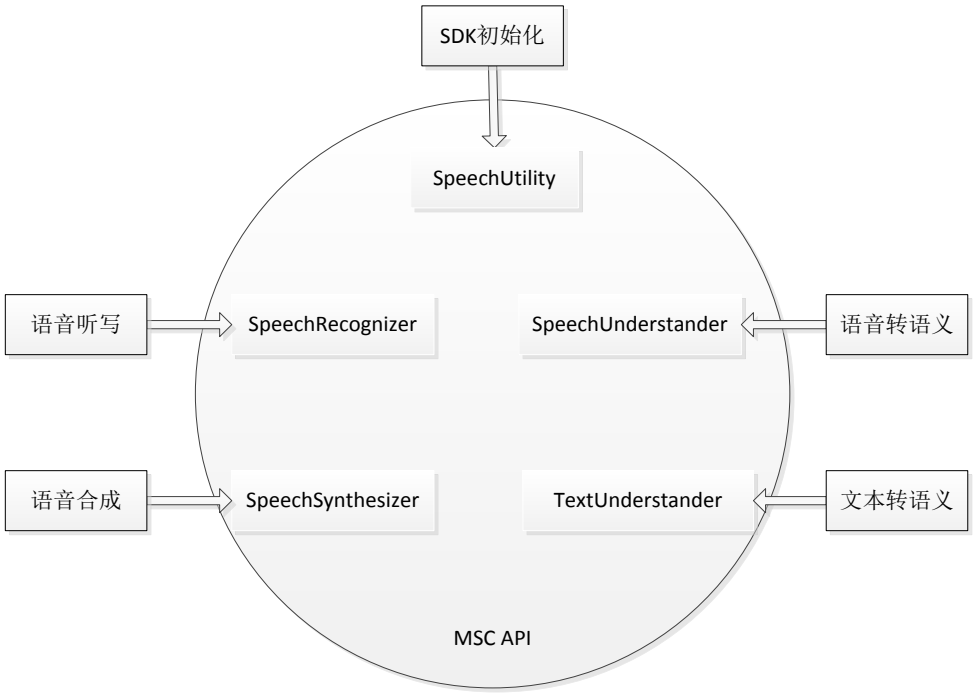


图 1 MSC 功能结构图

2. 集成说明

支持 Java 平台开发的操作系统为 x86 或 x64 的：Windows、Linux。

由于 Windows 系统为开发者广泛使用的，这里将在 Windows 上的安装环境的搭建步骤简要介绍如下。

Step 1 搭建开发环境

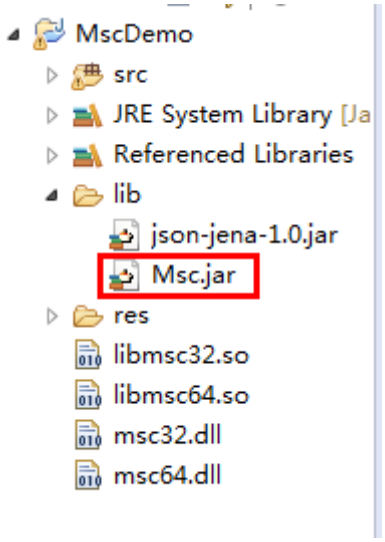
1) 配置 JDK 环境，编者采用的版本是 jdk1.6.0_20，读者可以从 Sun 官网 http://java.sun.com/javase/downloads/index.jsp 下载所需的版本；

2) 安装 Eclipse Java IDE，编者采用的版本是 Ecilpse3.4，读者可以从官网 http://www.eclipse.org/downloads/packages/release/ganymede/sr2 下载所需的版本

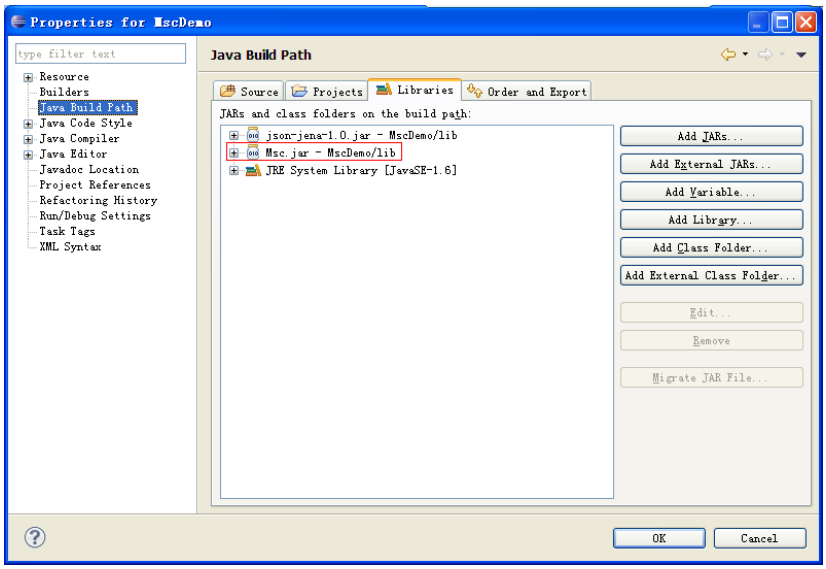
Step 2 导入 SDK

1) 在 Eclipse 中建立你的 Java 工程。

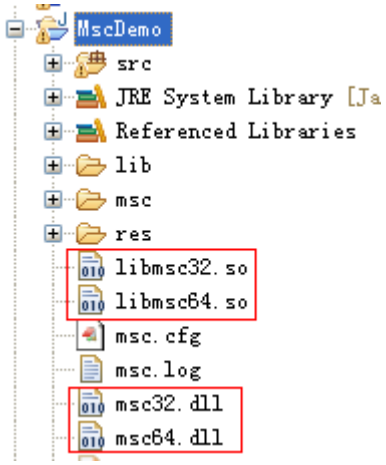
2) 将开发工具包中 libs 目录下的 Msc.jar 复制到新建工程的 libs 目录中（如下图所示）。



3) 在 Eclipse 中选中工程，通过工具栏 Project->Properties->Java Build Path->Libraries->Add JARS 或 ADD External JARS 引入 Msc.jar（如下图所示）。



4) 将 SDK.\lib 目录下库文件拷贝到工程根目录（如下图所示）。



5） 在你需要使用 MSC 服务的文件中导入相应的类。

例如：import com.iflytek.cloud.speech.SpeechRecognizer;

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。

关于初始化时指定库名，或报加载库失败的解决办法，请参考《MSC Reference Manual》中，关于 SpeechUtility 类，以及 SpeechConstant 类的说明。

```
// 将“XXXXXXXX”替换成您申请的 APPID
SpeechUtility.createUtility( SpeechConstant.APPID +"= XXXXXXXX ");
```

[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
SpeechRecognizer mIat= SpeechRecognizer.createRecognizer( );
//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);
    //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) {
            DebugLog.Log("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.音频流听写

```
//1.创建SpeechRecognizer对象
SpeechRecognizer mIat= SpeechRecognizer.createRecognizer( );
//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 ");
mIat.setParameter(SpeechConstant.AUDIO_SOURCE, "-1");
//3.开始听写
mIat.startListening(mRecoListener);
//voiceBuffer为音频数据流，splitBuffer为自定义分割接口，将其以4.8k字节分割成数组
ArrayList<byte[]> buffers = splitBuffer(voiceBuffer,voiceBuffer.length, 4800);
                for (int i = 0; i < buffers.size(); i++) {
                    // 每次写入msc数据4.8K,相当150ms录音数据
                    mIat.writeAudio(buffers.get(i), 0, buffers.get(i).length);
                    try {
                        Thread.sleep(150);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
                mIat.stopListening();
    }
//听写监听器
private RecognizerListener mRecoListener = new RecognizerListener(){
    public void onResult(RecognizerResult results, boolean isLast) {
                DebugLog.Log("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) {}
};
```

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《MSC Reference Manual》**UserWords** 类。

```
private void uploadUserWords() {
    SpeechRecognizer recognizer = SpeechRecognizer.getRecognizer();
    UserWords userwords = new UserWords(USER_WORDS);
    recognizer.setParameter( SpeechConstant.DATA_TYPE, "userword" );
    recognizer.updateLexicon("userwords",
            userwords.toString(),
            lexiconListener);
}
/**
 * 词表上传监听器
 */
LexiconListener lexiconListener = new LexiconListener() {
    @Override
    public void onLexiconUpdated(String lexiconId, SpeechError error) {
        if (error == null)
            DebugLog.Log("**********上传成功**********");
        else
            DebugLog.Log("**********" + error.getErrorCode()
                    + "**********");
        onLoop();
    }

};
```

[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。语法识别采用 ABNF 语法格式，参考《MSC Reference Manual》中对 SpeechConstant 类的 CLOUD_GRAMMAR 成员的说明。

```
//1.创建SpeechRecognizer对象
SpeechRecognizer mAsr = SpeechRecognizer.createRecognizer( );
// 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){
      DebugLog.Log("语法构建失败,错误码：" + ret);
}else{
      DebugLog.Log("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) {
      DebugLog.Log("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{
                        DebugLog.Log("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.播放合成

```
//1.创建 SpeechSynthesizer 对象
SpeechSynthesizer mTts= SpeechSynthesizer.createSynthesizer( );
//2.合成参数设置，详见《MSC Reference Manual》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“./tts_test.pcm”
//如果不需要保存合成音频，注释该行代码
mTts.setParameter(SpeechConstant.TTS_AUDIO_PATH, "./tts_test.pcm");
//3.开始合成
mTts.startSpeaking("语音合成测试程序", 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() {}
};
```

2.无声合成

```
//1.创建 SpeechSynthesizer 对象
SpeechSynthesizer mTts= SpeechSynthesizer.createSynthesizer( );
//2.合成参数设置，详见《MSC Reference Manual》SpeechSynthesizer 类
mTts.setParameter(SpeechConstant.VOICE_NAME, "xiaoyan");//设置发音人
mTts.setParameter(SpeechConstant.SPEED, "50");//设置语速，范围 0~100
mTts.setParameter(SpeechConstant.PITCH, "50");//设置语调，范围 0~100
mTts.setParameter(SpeechConstant.VOLUME, "50");//设置音量，范围 0~100
//3.开始合成
//设置合成音频保存位置（可自定义保存位置），默认保存在“./tts_test.pcm”
mTts.synthesizeToUri("语音合成测试程序", "./tts_test.pcm",synthesizeToUriListener);
//合成监听器
SynthesizeToUriListener synthesizeToUriListener = new SynthesizeToUriListener() {
        //progress为合成进度0~100
        public void onBufferProgress(int progress) {}
        //会话合成完成回调接口
        //uri为合成保存地址，error为错误信息，为null时表示合成会话成功
        public void onSynthesizeCompleted(String uri, SpeechError error) {}
    };
```

[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《MSC Reference Manual》关于SpeechUnderstander 类的介绍。

```
//1.创建文本语义理解对象
SpeechUnderstander understander = SpeechUnderstander.createUnderstander( );
//2.设置参数
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( );
//开始语义理解
mTextUnderstander.understandText("今天的天气", 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. 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以下过程：

[1] 初始化

创建用户语音配置对象后才可以使用声纹服务，建议在程序入口处调用。

```
// 将“XXXXXXXX”替换成您申请的 APPID，参考《MSC Reference Manual》关于 SpeechConstant 类的 APPID 成员的介绍
SpeechUtility.createUtility( SpeechConstant.APPID +"= XXXXXXXX ");
```

[2] 声纹注册

现阶段仅支持数字密码，其他类型这在开发中。在注册之前要选择声纹的类型。

```
// 首先获取SpeakerVerifier对象
mVerify = SpeakerVerifier.createVerifier(this, null);
// 通过setParameter设置密码类型
mVerify.setParameter(SpeechConstant.ISV_PWDT, "" + pwdType);
```

pwdType 的取值说明如下表所示：

pwdType 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“我的地盘我做主”、“移动改变生活”和“芝麻开门”三种。
2	自由说密码。用户通过读一段文字来进行声纹注册和验证，注册时要求声音长度为 15 秒左右，验证时要求 10 秒左右，内容不限。
3	数字密码。从云端拉取一组特定的数字串（共分 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onData方法对云端返回的JSON格式（具体格式见附录(4)）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 onData(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“我的地盘我做主”、“移动改变生活”、“芝麻开门”或者是从云端拉取的数字密码(每8位用“-”隔开)。自由说略过此步
mVerify.setParameter(SpeechConstant.ISV_PWD, pwdText);
// 对于自由说，必须设置采样频率为8000，并设置ISV_RGN为1。其他密码可略过此步
mVerify.setParameter(SpeechConstant.SAMPLE_RATE, "8000");
mVerify.setParameter(SpeechConstant.ISV_RGN, "1");
// 设置声纹对应的auth_id，它用于标识声纹对应的用户
mVerify.setParameter(SpeechConstant.ISV_AUTHID, 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) {}
        public void onResult(VerifierResult result) {
        public void onEvent(int arg0, int arg1, int arg2, Bundle arg3) {}
        public void onError(SpeechError error) {}
        public void onEndOfSpeech() {}
        public void onBeginOfSpeech() {}
};
```

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。

应用通过声明一个 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：

错误码	说明
MSS_ERROR_IVP_GENERAL	正常，请继续传音频
MSS_ERROR_IVP_EXTRA_RGN_SOPPORT	rgn 超过最大支持次数 9
MSS_ERROR_IVP_TRUNCATED	音频波形幅度太大，超出系统范围，发生截幅
MSS_ERROR_IVP_MUCH_NOISE	太多噪音

MSS_ERROR_IVP_TOO_LOW	声音太小
MSS_ERROR_IVP_ZERO_AUDIO	没检测到音频
MSS_ERROR_IVP_UTTER_TOO_SHORT	音频太短
MSS_ERROR_IVP_TEXT_NOT_MATCH	音频内容与给定文本不一致

另外，结果中还有一个 vid 字段，用于标识成功注册的声纹模型。结果中包含的字段，以及各字段的含义见附录（4）。

[3] 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。

[4] 声纹模型操作

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“我的地盘我做主”、“移动改变生活”、“芝麻开门”或者是从云平台拉取的数字密码。自由说略过此步
mVerify.setParameter(SpeechConstant.ISV_PWD, pwdText);
// 特别地，自由说一定要设置采样频率为8000，其他密码则不需要
mVerify.setParameter(SpeechConstant.SAMPLE_RATE, “8000”);
// 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onData方法进行处理，处理方法详见Demo示例
mVerify.sendRequest(cmd, auth_id, listener);
```

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

请参考《MSC Reference Manual》对 ErrorCode 及 SpeechError 类的说明。

(4).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

成员名	参数解释
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

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。

也可以在听写监听器的 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：Java 平台暂时不支持本地能力，后续会考虑开放，敬请期待。
(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.xfyun.cn/index.php/default/doccenter/doccenterInner?itemTitle=ZmFx&anchor=Y29udGl0bGU2Mw==

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