科大讯飞股份有限公司

IFLYTEK CO.,LTD.

科大讯飞 MSC 集成指南

目 录

目 录 1
1. 概述 1
2. 预备工作 3
Step 1 导入 SDK 3
Step 2 添加用户权限 3
Step 3 初始化 4
3. 语音输入 UI 6
4. 语音听写 7
4.1. 上传联系人 8
4.2. 上传用户词表 9
5. 命令词识别（语法识别） 10
5.1. 在线命令词识别 10
5.1.1. 应用级命令词识别 10
5.1.2. 终端级命令词识别 12
5.2. 离线命令词识别 13
6. 语音合成 14
7. 语义理解 15
7.1. 语音语义理解 15
7.2. 文本语义理解 15
8. 本地功能集成（语记） 16
8.1. 本地识别 16
8.2. 本地合成 17
8.3. 获取语记参数 17
9. 语音评测 18
10. 唤醒 20
11. 人脸检测 21
11.1. 人脸检测 21
11.2. 人脸聚焦 21
12. 身份验证平台（MFV） 22
12.1. 基本介绍 22
12.2. 功能使用 23
12.2.1. 特征注册 24
12.2.2. 特征验证 27
12.2.3. 特征鉴别 28
12.2.4. 模型操作 29
12.2.5. 组管理 31
12.3. 参数设置 34
13. 附录 36
13.1. 识别结果说明 36
13.2. 合成发音人列表 37
13.3. 错误码列表 38
13.4. 人脸检测结果说明 39

13.5. 身份验证结果说明 39
13.5.1. 人脸注册字段 39
13.5.2. 声纹注册字段 40
13.5.3. 人脸、声纹和融合验证字段 40
13.5.4. 人脸、声纹鉴别字段 41
13.5.5. 查询/删除模型字段 41
13.5.6. 组管理字段 42
常见问题 43

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：

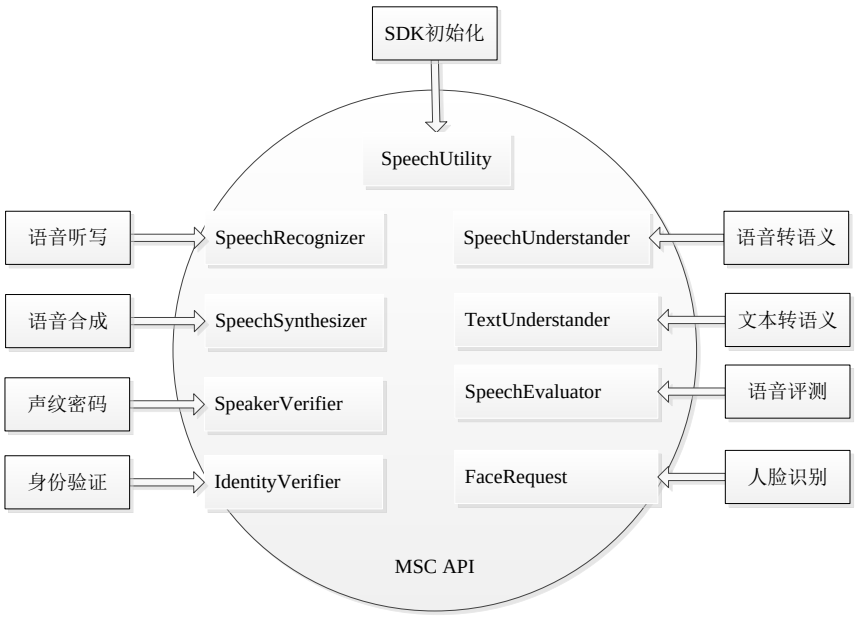


图 1 MSC 主要功能接口

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本，把语音中包含文字信息提取出来，并可以优先识别用户手机特有的联系人和个性化数据。
语法识别	判断用户所说的内容是否与预定义的语法相符合，主要用于识别用户是否下达某项指令，使用语法识别前，需要先定义语法。
语义理解	在语音听写基础上，分析理解用户的说话意图，返回结构化的指令信息。 开发者可在语义开放平台定义专属的问答格式。
语音评测	通过智能语音技术自动对发音水平进行评价，给出用户综合得分和发音信息。
声纹密码	据语音波形反映说话人生理和行为特征的语音参数，自动识别说话人身份，声纹识别所提供的安全性可与其他生物识别技术（指纹、掌形和虹膜）相媲美。
人脸识别	基于人的脸部特征信息进行身份识别的一种生物识别技术，可以自动在图像中检测和跟踪人脸，进而对检测到的人脸进行检测和验证。系统同时支持人脸关键点检出、视频流人脸检测等功能，识别率高达 99%。
身份验证	在本方案中，开发者可根据应用场景灵活的选择身份验证方式，如单人脸验证、单声纹验证以及

	人脸+声纹的融合验证方式。这样既解决了单生物特征识别暴露的局限性，也提供了更精准、更安全的识别和检测方案。身份验证方案还会持续增加更多的常用特征，达到更广泛的市场应用前景

2. 预备工作

Step 1 导入 SDK

将在官网下载的 Android SDK 压缩包中 libs 目录下所有子文件拷贝至 Android 工程的 libs 目录下。如下图所示：

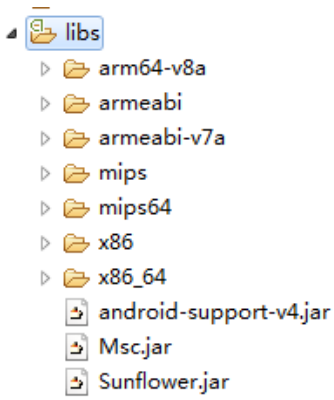


图 2 导入 SDK

注：

1、Android SDK 提供了各个平台 libmsc.so 文件，开发者可以根据工程需求选取适当平台库文件进行集成。

2、如果您需要将应用 push 到设备使用，请将设备 cpu 对应指令集的 libmsc.so push 到/system/lib 中。

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"/>
<!--外存储写入权限，构建语法需要用到此权限 -->
<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"/>
<!--手机定位信息，用来为语义等功能提供定位，提供更精准的服务-->
<!--定位信息是敏感信息，可通过Setting.setLocationEnable(false)关闭定位请求 -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

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，请务必在 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。 例如：libmsc_xxx_1072.so(xxx 为您的公司名,1072 为科大讯飞 sdk 版本号) 默认值:msc。 注：如您是预装软件，为了避免动态库冲突建议修改名称。	否

注意：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：

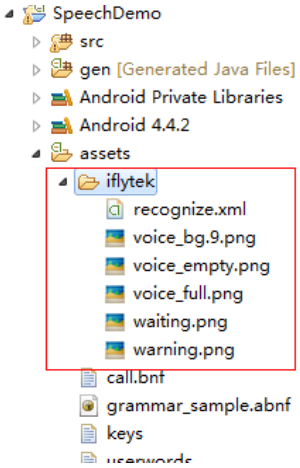


图 3 添加动画资源

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：本地听写时传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Demo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) {
         showTip(error.getPlainDescription(true)); //获取错误码描述
         }
        //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 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。

语法格式详见：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。

图 4 在线命令词识别入口

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 5 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，不启用终端级语法
// 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) {}
        // 事件回调
        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() {
    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 8.1 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《iFlytek MSC Reference Manual》SpeechSynthesizer 类
//设置发音人（更多在线发音人，用户可参见 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
};
```

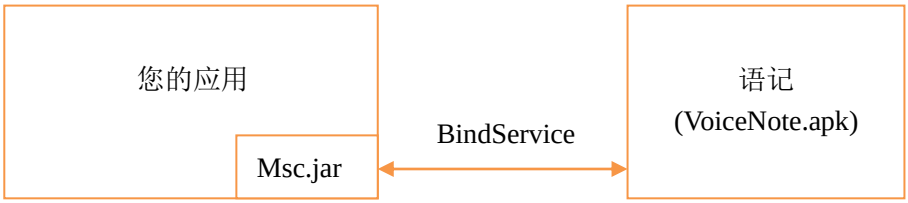
7.2.文本语义理解

用户通过输入文本获取语义结果，得到的专属语义结果和上述语音方式相同。

```
//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（语记）

本地识别、合成以及唤醒功能需要通过《语记》来实现。《语记》是基于讯飞语音云平台开发的应用，用户安装《语记》后，应用可以通过服务绑定来使用《语记》的本地功能，如下图所示：



在使用本地功能之前，先检查讯飞《语记》的安装情况：

```
//检查《语记》是否安装
//如未安装，获取《语记》下载地址进行下载。安装完成后即可使用服务。
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);
}
```

注：《语记》原名语音+，对外提供的本地服务是完全免费的。开发者若有意愿通过 sdk 在应用内集成本地功能，详情请见官网信息：http://www.xfyun.cn/

8.1.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《语记》，通过《语记》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.构建语法（本地识别引擎目前仅支持 BNF 语法），同在线语法识别 请参照 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");
ret = mAsr.startListening(mRecognizerListener);
```

8.2.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《语记》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);
```

8.3.获取语记参数

用户可以通过《语记》中的资源下载（包括：识别资源、发音人资源）来提升《语记》离线能力，开发者可以通过以下接口获取当前《语记》包含的离线资源列表，此接口从《语记》1.032(99)版本开始支持。（通过 getServiceVersion()获取版本号） 注：后续版本将支持获取《语记》当前设置的发音人字段

```
//1.设置所需查询的资源类型
/**
*1.PLUS_LOCAL_ALL: 本地所有资源
 2.PLUS_LOCAL_ASR: 本地识别资源
 3.PLUS_LOCAL_TTS: 本地合成资源
*/
String type = SpeechConstant.PLUS_LOCAL_ASR;
//2.获取当前《语记》包含资源列表
String resource = SpeechUtility.getUtility().getParameter(type);
//3.解析 json 请参见下表格式及 Demo 中解析方法
```

```
{"ret":0,"result":{"version":11,
"tts":[{"sex":"woman","language":"zh_cn","accent":"mandarin","nickname":"邻家姐姐
        ","age":"22","name":"xiaojing"},
        {"sex":"woman","language":"zh_cn","accent":"mandarin","nickname":"王老师
      ","age":"24","name":"xiaoyan"}],
"asr":[{"domain":"asr","samplerate":"16000","language":"zh_cn","accent":"mandarin","name":"commo
n"}]}}
```

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）、词语和句子朗读三种题型，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");
// 设置评测题型
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	评测题型，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

请登录 http://www.xfyun.cn/index.php/services/awaken?type=awaken 下载体验吧！

11.人脸检测

人脸检测可以检测出照片中的人脸并返回关键点坐标。相关概念的说明如下：

表 5 人脸识别概念说明

名称	说明
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）。

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 6 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。

11.1. 人脸检测

```
// 设置业务类型为检测
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);
```

11.2. 人脸聚焦

```
// 设置业务类型为聚焦
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);
```

12.身份验证平台（MFV）

多生物特征融合验证（Multi-biometrics Fusion Verification，简称 MFV）平台是科大讯飞推出的新一代个人身份验证平台。功能上支持 **1：1 单一验证**，**1：1 融合验证**，**1：N 鉴别**。生物特征上支持**人脸**、**声纹**，并将在未来支持指纹、虹膜等其他特征信息。

12.1.基本介绍

MFV 目前提供的功能组合如表 7 所示。

表 7 MFV 功能组合说明

功能 / 生物特征	1：1 单一验证	1：1 融合验证	1：N 鉴别
人脸	**人脸验证**	**人脸声纹融合验证**	**人脸鉴别**
声纹	**声纹验证**		**声纹鉴别**

MFV 相关的概念如表 8 所示。

表 8 MFV 相关概念说明

分类	概念	英文标识	说明
id 相关	用户 id	auth_id	用户身份的唯一标识。
	组 id	group_id	组的唯一标识。 组被用来限定 1：N 鉴别的用户范围。
功能相关	特征注册	enroll	上传用户特征数据，在云端生成特征模型。 其中，人脸图像数据的大小应控制在 200K 以下。
	特征验证	verify	上传用户特征数据，云端将其与已注册的特征模型进行比对，返回结果（相似度、是否通过验证等）。
	融合验证	mixed verify	上传用户多项特征数据，云端将其与已注册的多项特征模型进行比对，返回结果（综合相似度、是否通过验证等）。
	特征鉴别	identify	上传用户特征数据，并指定鉴别组 id，云端将上传数据与组内用户对应的已注册的特征模型进行比对，返回结果（相似度排行、用户名称）。

会话相关	业务场景	scenes	会话的场景。 包括：人脸（ifr），声纹（ivp），人脸声纹融合（ifr|ivp），组管理（ipt）。
	业务类型	sst	会话的业务类型。 在不同的会话场景（scenes）下有不同的业务类型，详情见表 9。
	子业务类型	ssub	子业务类型。 包括：人脸（ifr），声纹（ivp），组管理（ipt）。

表 9 MFV 业务场景与业务类型组合

业务场景 / 业务类型	人脸（ifr）	声纹（ivp）	人脸声纹融合（ifr|ivp）	组管理（ipt）
注册	√	√		
验证	√	√	√	
鉴别	√	√		

表 10 MFV 子业务操作组合

子业务类型 / 操作类型	人脸（ifr）	声纹（ivp）	组管理（ipt）
创建			√
加入			√
查询		√	√
删除	√	√	√
密码下载		√	

12.2.功能使用

1：1 验证：

1) 首次使用需先进行特征注册。
2) 开始验证。设定各项参数，接着开启会话，上传待验证的生物特征数据，最后获取验证结果并解析。

1：N 鉴别：

1) 首次使用需先进行特征注册。
2) 创建组。
3) 将相关的用户 id 加入到组中。
4) 开始鉴别。指定鉴别组 id 及各项参数，接着开启会话，上传待鉴别的生物特征数据，最后获取鉴别结果并解析。

鉴别功能默认限制每个 Appid 提供 10 个鉴别组及每个鉴别 100 个成员。

注：

1、 未提交应用审核的 Appid，仅提供 10 个鉴别组进行开发调试。通过审核后将开放 5000 个上限的鉴别组，以保证应用正常使用；

2、 鉴别组默认仅支持 100 个成员，若开发者有更多成员需求，请发送邮件联系我们；

联系方式：发送邮件说明原因至 msp_support@iflytek.com

12.2.1. 特征注册

人脸注册流程：

1) 设置参数：业务场景“人脸（ifr）”，业务类型“注册（enroll）”，用户 id（auth_id）。
2) 设置监听，开启会话。
3) 指定子业务类型“人脸（ifr）”、人脸数据内容及数据长度，然后上传数据。
4) 待会话监听器返回结果，此次注册操作结束。

人脸注册示例代码：

```
// 身份验证对象
private IdentityVerifier mIdVerifier = IdentityVerifier.createVerifier(private, null);

// 设置参数
// 清空设置
mIdVerifier.setParameter(SpeechConstant.PARAMS, null);
// 设置业务场景：人脸（ifr）
mIdVerifier.setParameter(SpeechConstant.MFV_SCENES, "ifr");
// 设置业务类型：注册（enroll）
mIdVerifier.setParameter(SpeechConstant.MFV_SST, "enroll");
// 设置用户id
mIdVerifier.setParameter(SpeechConstant.AUTH_ID, mAuthId);
// 设置监听器，开始会话
mIdVerifier.startWorking(mEnrollListener);

// 上传数据，mImageData为bitmap格式。
// writeData可以连续多次调用，调用的时候要指定子业务类型、子业务参数（没有可为空）、
// 数据内容、数据偏移量以及数据长度。
StringBuffer ifrParams = new StringBuffer();
mIdVerifier.writeData("ifr", ifrParams.toString(), mImageData, 0, mImageData.length);
// 写入完成后，需要调用stopWrite停止写入，在停止的时候同样需要指定子业务类型。只有
// stopWrite之后，才会触发mEnrollListener中的回调接口，返回结果或者错误信息。
mIdVerifier.stopWrite("ifr");
```

```
// 注册回调接口
private IdentityListener mEnrollListener = new IdentityListener() {
    // 结果回调，result中有json结果，字段说明请见附录 13.5，解析方法请参考MFVDemo源码
    public void onResult(IdentityResult result, boolean islast) {}
    // 扩展接口
    public void onEvent(int eventType, int arg1, int arg2, Bundle obj) {}
    // 出错回调
    public void onError(SpeechError error) {}
};
```

声纹注册：

1) 从云端下载声纹注册所用的数字密码文本（详情见 12.2.4.子业务操作）。
2) 设置参数：业务场景“声纹（ivp）”，业务类型“注册（enroll）”，用户 id（auth_id）。
3) 设置监听，开启会话。
4) 指定子业务类型“声纹（ifr）”，设定声纹注册相关参数“训练次数（rgn）、密码内容（ptxt）、密码类型（pwdt）”，并指定声纹数据内容及长度，然后上传数据。
5) 待会话监听器返回结果，此次注册操作结束。

声纹注册示例代码：

```
// 身份验证对象
private IdentityVerifier mIdVerifier = IdentityVerifier.createVerifier(private, null);

// 下载声纹注册密码
// 清空设置
mIdVerifier.setParameter(SpeechConstant.PARAMS, null);
// 设置业务场景：声纹（ivp）
mIdVerifier.setParameter(SpeechConstant.MFV_SCENES, "ivp");
// 执行密码下载操作，密码类型：数字密码（pwdt=3）
mIdVerifier.execute("ivp", "download", "pwdt=3", mDownloadPwdListener);
// 下载密码回调接口
private IdentityListener mDownloadPwdListener = new IdentityListener() {
    // 结果回调，result中有json结果，字段名称为"num_pwd"，具体解析请参考MFVDemo源码
    public void onResult(IdentityResult result, boolean islast) {}
    // 扩展接口
    public void onEvent(int eventType, int arg1, int arg2, Bundle obj) {}
    // 出错回调
    public void onError(SpeechError error) {}
};
```

```
// 设置参数
// 清空设置
mIdVerifier.setParameter(SpeechConstant.PARAMS, null);
// 设置业务场景：声纹（ivp）
mIdVerifier.setParameter(SpeechConstant.MFV_SCENES, "ivp");
// 设置业务类型：注册（enroll）
mIdVerifier.setParameter(SpeechConstant.MFV_SST, "enroll");
// 设置用户id
mIdVerifier.setParameter(SpeechConstant.AUTH_ID, mAuthId);
// 设置监听器，开始会话
mIdVerifier.startWorking(mEnrollListener);
// 准备声纹注册相关参数，训练次数、密码内容和密码类型
StringBuffer params = new StringBuffer();
params.append("rgn=5")
            .append(",ptxt=" + mPwdText)
            .append(",pwdt=3");

// 写入音频数据，data为音频数据，需要开发者自行从外部录音机PcmRecorder获取，获取方法详
// 见demo源码。writeData可以连续多次调用，调用的时候要指定子业务类型、子业务参数、数据
// 内容及长度
mIdVerifier.writeData("ivp", params.toString(), data, 0, length);
// 写入完成后，需要调用stopWrite停止写入，在停止的时候同样需要指定子业务类型。只有
// stopWrite之后，才会触发mEnrollListener中的回调接口，返回结果或者错误信息
mIdVerifier.stopWrite("ivp");

// 注册回调接口
private IdentityListener mEnrollListener = new IdentityListener() {
    // 结果回调，result中有json结果，字段说明请见附录 13.5，解析方法请参考MFVDemo源码
    public void onResult(IdentityResult result, boolean islast) {}
    // 扩展接口，用于声纹音量通知和VAD尾端点，需要根据eventType判断是音量通知还是VAD
    // 尾端点事件
    public void onEvent(int eventType, int arg1, int arg2, Bundle obj) {}
    // 出错回调
    public void onError(SpeechError error) {}
};
```

12.2.2. 特征验证

根据场景不同，特征验证又分为**人脸验证**、**声纹验证**以及**人脸声纹融合验证**。验证的过程和注册的过程很相似，以下是融合验证的示例代码。

```
// 身份验证对象
private IdentityVerifier mIdVerifier = IdentityVerifier.createVerifier(private, null);

// 设置参数
mIdVerifier.setParameter(SpeechConstant.PARAMS, null);
// 设置业务场景：人脸声纹融合（ifr|ivp）
mIdVerifier.setParameter(SpeechConstant.MFV_SCENES, "ifr|ivp");
// 设置业务类型：验证（verify）
mIdVerifier.setParameter(SpeechConstant.MFV_SST, "verify");
// 设置验证模式：混合生物特征数据验证（mix），只在融合场景下生效
mIdVerifier.setParameter(SpeechConstant.MFV_VCM, "mix");
// 设置用户id
mIdVerifier.setParameter(SpeechConstant.AUTH_ID, mAuthId);
// 设置监听器，开始会话
mIdVerifier.startWorking(mVerifyListener);

// 写入人脸数据，imageData为bitmap格式
StringBuffer ifrParams = new StringBuffer();
mIdVerifier.writeData("ifr", ifrParams.toString(), imageData, 0, imageData.length);
mIdVerifier.stopWrite("ifr");

// 准备声纹验证相关参数，密码内容、密码类型
StringBuffer ivpParams = new StringBuffer();
ivpParams.append("ptxt=" + mPwdText + ",pwdt=3");
// 写入音频数据，data为音频数据
mIdVerifier.writeData("ivp", ivpParams.toString(), data, 0, length);
mIdVerifier.stopWrite("ivp");

// 声纹验证监听器定义如下
private IdentityListener mVerifyListener = new IdentityListener() {
    // 结果回调，result中有json结果，字段说明请见附录 13.5，解析方法请参考MFVDemo源码
    public void onResult(IdentityResult result, boolean islast) {}
    // 扩展接口
    public void onEvent(int eventType, int arg1, int arg2, Bundle obj) {}
    // 出错回调
    public void onError(SpeechError error) {}
};
```

12.2.3. 特征鉴别

根据场景不同，特征鉴别分为**人脸鉴别**和**声纹鉴别**。两种鉴别流程相同，以人脸鉴别为例。

```
// 身份验证对象
private IdentityVerifier mIdVerifier = IdentityVerifier.createVerifier(private, null);

// 设置参数
// 清空设置
mIdVerifier.setParameter(SpeechConstant.PARAMS, null);
// 设置业务场景：人脸（ifr）
mIdVerifier.setParameter(SpeechConstant.MFV_SCENES, "ifr");
// 设置业务类型：鉴别（identify）
mIdVerifier.setParameter(SpeechConstant.MFV_SST, "identify");
// 设置监听器，开始会话
mIdVerifier.startWorking(mVerifyListener);

// 准备参数，指定组id，"topc=3"为显示相似度最高的前3个结果
StringBuffer ifrParams = new StringBuffer();
ifrParams.append("group_id=" + mGroupId)
            .append(",topc=3");
// 写入人脸照片数据，imageData为bitmap格式
mIdVerifier.writeData("ifr", ifrParams.toString(), imageData, 0, imageData.length);
mIdVerifier.stopWrite("ifr");

// 鉴别监听器定义如下
private IdentityListener mSearchListener = new IdentityListener() {
    // 结果回调，result中有json结果，字段说明请见附录 13.5，解析方法请参考MFVDemo源码
    public void onResult(IdentityResult result, boolean islast) {}
    // 扩展接口
    public void onEvent(int eventType, int arg1, int arg2, Bundle obj) {}
    // 出错回调
    public void onError(SpeechError error) {}
};
```

12.2.4. 模型操作

声纹模型目前支持的操作有**查询（query）、删除（delete）、密码下载（download）**三种。人脸模型目前支持**删除（delete）**操作。

声纹密码下载（download），已经在声纹模型注册的过程中展示，此处不再赘述。

人脸模型删除示例代码：

```
// 身份验证对象
private IdentityVerifier mIdVerifier = IdentityVerifier.createVerifier(private, null);

// 设置参数
// 清空设置
mIdVerifier.setParameter(SpeechConstant.PARAMS, null);
// 设置业务场景：人脸（ifr）
mIdVerifier.setParameter(SpeechConstant.MFV_SCENES, "ifr");
// 设置用户id
mIdVerifier.setParameter(SpeechConstant.AUTH_ID, mAuthId);

// 调用execute方法执行操作。第一个参数为子业务类型，取值为“ifr”。
// 第二个参数为操作类型，取值为“delete”。
StringBuffer ifrParams = new StringBuffer();
mIdVerifier.execute("ifr", "delete", ifrParams.toString(), mModelListener);

// 模型操作监听器定义如下
private IdentityListener mModelListener = new IdentityListener() {
    // 结果回调，result中有json结果，字段说明请见附录 13.5，解析方法详细MFVDemo源码
    public void onResult(IdentityResult result, boolean islast) {}
    // 扩展接口
    public void onEvent(int eventType, int arg1, int arg2, Bundle obj) {}
    // 出错回调
    public void onError(SpeechError error) {}
};
```

声纹模型查询示例代码：

```
// 身份验证对象
private IdentityVerifier mIdVerifier = IdentityVerifier.createVerifier(private, null);

// 设置参数
// 清空设置
mIdVerifier.setParameter(SpeechConstant.PARAMS, null);
// 设置业务场景：声纹（ivp）
mIdVerifier.setParameter(SpeechConstant.MFV_SCENES, "ivp");
// 设置用户id
mIdVerifier.setParameter(SpeechConstant.AUTH_ID, mAuthId);

// 调用execute方法执行操作。第一个参数为子业务类型，取值为“ivp”。
// 第二个参数支持“query”（查询）、“delete”（删除）和“download”（密码下载）三种。
StringBuffer ivpParams = new StringBuffer();
mIdVerifier.execute("ivp", "query", ivpParams.toString(), mModelListener);

// 模型操作监听器定义如下
private IdentityListener mModelListener = new IdentityListener() {
    // 结果回调，result中有json结果，字段说明请见附录 13.5，解析方法详细MFVDemo源码
    public void onResult(IdentityResult result, boolean islast) {}
    // 扩展接口
    public void onEvent(int eventType, int arg1, int arg2, Bundle obj) {}
    // 出错回调
    public void onError(SpeechError error) {}
};
```

12.2.5. 组管理

组管理目前支持的操作有：**创建组、删除组、查询组用户、用户加入组以及用户退出组。**

创建组示例代码：

```
// 1、创建身份验证对象
private IdentityVerifier mIdVerifier = IdentityVerifier.createVerifier(private, null);
// 2、设置参数
// 清空设置
mIdVerifier.setParameter(SpeechConstant.PARAMS, null);
// 设置业务场景：组管理（ipt）
mIdVerifier.setParameter(SpeechConstant.MFV_SCENES, "ipt");
// 设置用户id
mIdVerifier.setParameter(SpeechConstant.AUTH_ID, mAuthId);

//3、准备创建组参数
StringBuffer params = new StringBuffer();
params.append("scope=group");
params.append(",group_name=" + mGroupName);
///4、调用execute方法执行操作
mIdVerifier.execute("ipt", "add", params.toString(), mCreateListener);

// 创建组监听器
private IdentityListener mCreateListener = new IdentityListener() {
    // 结果回调，result中有json结果，字段说明请见附录 13.5，解析方法详细MFVDemo源码
    public void onResult(IdentityResult result, boolean islast) {}
    // 扩展接口
    public void onEvent(int eventType, int arg1, int arg2, Bundle obj) {}
    // 出错回调
    public void onError(SpeechError error) {}
};
```

删除组示例代码：

```
// 1、创建身份验证对象（参考创建组）
// 2、设置参数（参考创建组）
// 3、准备删除组参数
StringBuffer params = new StringBuffer();
params.append("scope=group");
params.append(",group_id=" + group_id);
// 4、调用execute方法执行操作
mIdVerifier.execute("ipt", "delete", params.toString(),mDeleteListener);

// 删除组监听器
private IdentityListener mDeleteListener = new IdentityListener() {
    // 结果回调，result中有json结果，字段说明请见附录 13.5，解析方法详细MFVDemo源码
    public void onResult(IdentityResult result, boolean islast) {}
    // 扩展接口
    public void onEvent(int eventType, int arg1, int arg2, Bundle obj) {}
    // 出错回调
    public void onError(SpeechError error) {}
};
```

查询组用户示例代码：

```
// 1、创建身份验证对象（参考创建组）
// 2、设置参数（参考创建组）
// 3、准备查询组用户参数
StringBuffer params = new StringBuffer();
params.append("scope=group");
params.append(",group_id=" + group_id);
// 4、调用execute方法执行操作
mIdVerifier.execute("ipt", "query", params.toString(),mQueryListener);

// 查询组用户监听器
private IdentityListener mQueryListener = new IdentityListener() {
    // 结果回调，result中有json结果，字段说明请见附录 13.5，解析方法详细MFVDemo源码
    public void onResult(IdentityResult result, boolean islast) {}
    // 扩展接口
    public void onEvent(int eventType, int arg1, int arg2, Bundle obj) {}
    // 出错回调
    public void onError(SpeechError error) {}
};
```

用户加入组示例代码：

```
// 1、创建身份验证对象（参考创建组）
// 2、设置参数（参考创建组）
// 3、准备用户加入组参数，加入指定auth_id用户
StringBuffer params = new StringBuffer();
params.append("scope=person");
params.append(",auth_id=" + auth _id);
params.append(",group_id=" + group_id);
// 4、调用execute方法执行操作
mIdVerifier.execute("ipt", "add", params.toString(),mAddListener);

// 加入组监听器
private IdentityListener mAddListener = new IdentityListener() {
    // 结果回调，result中有json结果，字段说明请见附录 13.5，解析方法详细MFVDemo源码
    public void onResult(IdentityResult result, boolean islast) {}
    // 扩展接口
    public void onEvent(int eventType, int arg1, int arg2, Bundle obj) {}
    // 出错回调
    public void onError(SpeechError error) {}
};
```

用户退出组示例代码：

```
// 1、创建身份验证对象（参考创建组）
// 2、设置参数（参考创建组）
// 3、准备用户退出组参数，删除指定auth_id用户
StringBuffer params = new StringBuffer();
params.append("scope=person");
params.append(",auth_id=" + auth _id);
params.append(",group_id=" + group_id);
// 4、调用execute方法执行操作
mIdVerifier.execute("ipt", "delete", params.toString(),mDeleteListener);

// 删除用户监听器
private IdentityListener mDeleteListener = new IdentityListener() {
    // 结果回调，result中有json结果，字段说明请见附录 13.5，解析方法详细MFVDemo源码
    public void onResult(IdentityResult result, boolean islast) {}
    // 扩展接口
    public void onEvent(int eventType, int arg1, int arg2, Bundle obj) {}
    // 出错回调
    public void onError(SpeechError error) {}
};
```

12.3.参数设置

多生物特征融合验证平台的参数分为两种，分别为 MFV 主参数和子业务参数。

MFV 主参数通过 IdentityVerifier.setParameter(String key, String value)方法进行设置，参数如表 11 所示。

表 11 MFV 主参数

名称	说明	取值范围	默认值
auth_id	用户 id,用户身份的唯一标识	自拟，长度 6-18 位，仅包括英文、数字	无
group_id	通过组管理功能创建的鉴别组的唯一标识	长度 20 以内的字符串，由组管理功能创建得到，或由他人告知	无,在鉴别场景下必须指定
scenes	会话场景	ifr（人脸）,ivp（声纹）,ifr|ivp（人脸+声纹）,组管理（ipt）	无，必须指定
sst	会话的业务类型	enroll（注册）,verify（验证）,identify（鉴别）	无，必须指定
vcm	验证模式,仅在验证场景下使用	sin（单一特征）,mix（融合）,agi（灵活）	无,在验证场景下必须指定
afc	灵活验证保留结果时间	0-43200s	无，只在 vcm 设置成 agi 时生效
prot_type	联网协议	ssl	非 ssl 协议
sslcert	证书内容	证书内容	赛门铁克安全证书（仅在 prot_type=ssl 时生效）

注意：

1.scenes 和 vcm 必须组合使用：例如指定 vcm 为 sin 则 scenes 只可以选择 ifr 或者 ivp 单独业务，vcm 选择 mix 则 scenes 只可以选择 ifr|ivp。另：agi(灵活)模式可以设置 sences=ifr|ivp，在当次会话只发送一种业务数据，服务端保留验证结果，如在设置的时间间隔内再传入另外的业务数据即可做到混合验证。

2. prot_type、sslcert 参数也可在 SpeechUtility. createUtility 时传入，之后每次会话都会生效。

3.支持服务端回调通知业务。当用户进行了注册、验证操作后，讯飞服务端发送结果消息给开发者的业务服务器，如需此服务请联系：msp_support@iflytek.com

子业务参数以 String 格式的键-值对在调用 IdentityVerifier.writeData(String ssub, String params, byte[] data, int offset, int length)写入子业务特征数据或者调用 IdentityVerifier.execute(String ssub, String cmd, String params, IdentityListener listener)执行模型操作时传入，对应于 params 参数。详见 Demo。

表 12 MFV 子业务参数

子业务	名称	说明	取值	默认值
ivp（声纹）	rgn	注册次数。	2-9。	5(建议使用默认值，效果最好)
	ptxt	密码文本，指定声纹密码注册时使用的声纹密码内容。	由服务端下发，比如数字密码所需的数字串。	无，必须指定
	pwdt	密码类型，指定声纹密码注册时使用的声纹密码类型。	3（数字密码） 注：其他类型暂不支持。	无，必须指定
ifr（人脸）	暂无			
ipt（组管理）	auth_id	用户 id		设备 id，必须指定
	scope	操作对象	group（组） person（成员）	无，必须指定
	group_name	创建的组名称		无，必须指定
	group_id	加入的组 id		无，必须指定

结果格式说明详见附录 13.5。

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
小艾	青年女声	中文（普通话）	aisxa		支持情感设置，需付费使用
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. 人脸检测结果说明

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_bottom":{"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"}

13.5. 身份验证结果说明

13.5.1. 人脸注册字段

JSON 字段	类型	说明
sst	String	业务类型，注册业务为 enroll
ssub	String	子业务类型，人脸业务为 ifr
ret	int	返回值，0 为请求成功，其他为请求失败
suc	int	本次注册已成功的训练次数
rgn	int	本次注册需要的训练次数
fid	String	人脸模型 id (当前无需关注)

人脸注册结果示例：

{"ret":0,"suc":1,"rgn":1,"sst":"enroll","ssub":"ifr","fid":"90f821fa7381ee297a80ed9570dea635"}

13.5.2. 声纹注册字段

JSON 字段	类型	说明
sst	String	业务类型，注册业务为 enroll
ssub	String	子业务类型，声纹业务为 ivp
ret	int	返回值，0 为请求成功，其他为请求失败
rgn	int	本次注册需要的训练次数
suc	int	本次注册已成功的训练次数
vid	String	声纹模型 id (当前无需关注)

声纹注册结果示例：

{"vid":"16eb6a9f24c96405647347f8458e4cea","suc":5,"rgn":5,"sst":"enroll","ssub":"ivp","ret":0}

13.5.3. 人脸、声纹和融合验证字段

JSON 字段	类型	说明
sst	String	业务类型，验证业务为 verify
ssub	String	子业务类型，取值： ivp：声纹验证； ifr：人脸验证； ivp| ifr：融合验证。
ret	int	返回值，0 为请求成功，其他为请求失败
decision	String	accepted：验证成功，rejected：验证失败
fusion _score	double	相似度得分，仅验证业务返回
face _score	double	人脸验证得分，仅验证业务返回
voice _score	double	声纹验证得分，仅验证业务返回

验证结果示例：

{"ret":0,"face_score":99.732,"voice_score":86.874,"ssub":"ivp|ifr","decision":"accepted","fusion_score":99.823,"sst":"verify"}

13.5.4. 人脸、声纹鉴别字段

JSON 字段	类型	说明
sst	String	业务类型，鉴别业务为 identify
ssub	String	子业务类型，取值： ivp：声纹； ifr：人脸。
ret	int	返回值，0 为请求成功，其他为请求失败
group_id	String	本次鉴别的成员组 id
group_name	String	本次鉴别的成员组 id 对应的组名称
topc	int	本次鉴别返回的结果数
model_id	String	模型 id
decision	String	accepted：匹配成功，rejected：匹配失败
score	double	匹配相似度
user_name	String	该模型对应用户名

鉴别结果示例：

{"ret":0,"group_id":"xxxxxx","group_name":"xxxxxx","ifv_result":{"candidates":[{"model_id":"xxxxxxxx","decision":"accepted","score":88.888888,"user":"user_name"}]},"sst":"identify","ssub":"ivp","topc":1}

13.5.5. 查询/删除模型字段

JSON 字段	类型	说明
ssub	String	业务类型，取值： ivp：声纹业务； ifr：人脸业务（暂无查询业务）
ret	int	返回值，0 为请求成功，其他为请求失败
sst	String	子业务类型，取值： query：查询模型； delete：删除模型；

查询结果示例：

{"ssub":"ivp","sst":"query","ret":0}

删除结果示例：

{"ssub":"ivp","sst":"delete","ret":0}

13.5.6. 组管理字段

JSON 字段	类型	说明
ssub	String	ipt：组管理
ret	int	返回值，0 为请求成功，其他为请求失败
group_name	String	组名称
group_id	String	组 id
person	array	组内成员集
user	String	用户名

创建组结果示例：

{"ssub":"ipt","group_name":" xxxxxxxx ","sst":"add","ret":0,"group_id":"xxxxxxxx"}

删除组结果示例：

{"ssub":"ipt","group_name":" xxxxxxxx ","sst":"delete","ret":0,"group_id":" xxxxxxxx "}

查询组中人员结果示例：

{"ssub":"ipt","person":[{"user":" xxxxxxxx "}],"group_name":" xxxxxxxx ","sst":"query","ret":0,"group_id":" xxxxxxxx "}

用户加入组结果示例：

{"ssub":"ipt","group_name":" xxxxxxxx ","ret":0,"sst":"add","user":" xxxxxxxx ","group_id":" xxxxxxxx "}

用户退出组结果示例：

{"ssub":"ipt","group_name":" xxxxxxxx ","ret":0,"sst":"delete","user":" xxxxxxxx ","group_id":" xxxxxxxx "}

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、本地听写语音唤醒功能了。

(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://bbs.xfyun.cn/

联系方式：

邮箱：msp_support@iflytek.com