安徽科大讯飞信息科技股份有限公司
ANHUI USTC iFLYTEK CO.,LTD

MSC 开发指南

重要声明

版权声明

商标声明

安徽科大讯飞信息科技股份有限公司的产品是安徽科大讯飞信息科技股份有限公司专有。在提及其他公司及其产品时将使用各自公司所拥有的商标，这种使用的目的仅限于引用。本文档可能涉及安徽科大讯飞信息科技股份有限公司的专利（或正在申请的专利）、商标、版权或其他知识产权，除非得到安徽科大讯飞信息科技股份有限公司的明确书面许可协议，本文档不授予使用这些专利（或正在申请的专利）、商标、版权或其他知识产权的任何许可协议。

不作保证声明

安徽科大讯飞信息科技股份有限公司不对此文档中的任何内容作任何明示或暗示的陈述或保证，而且不对特定目的的适销性及适用性或者任何间接、特殊或连带的损失承担任何责任。本手册内容若有变动，恕不另行通知。本手册例子中所用的公司、人名和数据若非特别声明，均属虚构。未得到安徽科大讯飞信息科技股份有限公司明确的书面许可，不得为任何目的、以任何形式或手段（电子的或机械的）复制或传播手册的任何部分。

保密声明

本文档（包括任何附件）包含的信息是保密信息。接收人了解其获得的本文档是保密的，除用于规定的目的外不得用于任何目的，也不得将本文档泄露给任何第三方。

本软件产品受最终用户许可协议（EULA）中所述条款和条件的约束，该协议位于产品文档和/或软件产品的联机文档中，使用本产品，表明您已阅读并接受了 EULA 的条款。

目 录

前言

语音作为人们获取和交流信息最便捷、最有效的方式，近年来，正在日益影响和改变着人们的生活。随着移动互联网时代的到来及迅猛发展，移动终端的交互方式日渐突显出诸多局限（如键盘太小影响文字输入，屏幕太小影响阅读体验，以及无法处理诸如开车和步行等特定场景下的交互）而智能语音交互技术使得移动终端像人一样 “能听会说”，从而可以大大提高移动终端的交互体验和效率，弥补传统交互方式的不足。

近年来国内外 IT巨头纷纷投入巨资加大对智能语音技术市场的争夺，如谷歌、微软、IBM、苹果等均相继推出了自己的语音产品。科大讯飞作为亚太地区最大的语音上市公司，在智能语音技术领域有着长期的研究积累，不仅多语种语音核心技术处于国际领先地位，而且拥有中国语音主流市场 80％的份额和众多开发合作伙伴，已成为业界公认的语音产业国家队，以讯飞为核心的中文语音产业链已初具规模。

讯飞语音云是基于公司已有的ISP（讯飞语音应用平台）和 IMS（讯飞 MRCP服务器）产品，利用云计算技术处理海量数据的独特优势，为符合移动互联网用户使用而开发的语音应用开放平台，提供语音合成、语音听写、语音识别等智能语音交互服务。它能够让更多的企业和开发爱好者克服语音应用创新的门槛，以最小的代价、最快的速度完成应用开发，从而促进细分移动互联网产业集群的跨越式发展，形成语音产业的集群优势和规模效应。其主要功能有：

1）实现基于 HTTP 协议的语音应用服务器，集成公司最新的语音引擎，提供语音合成、语音听写、语音识别等服务；

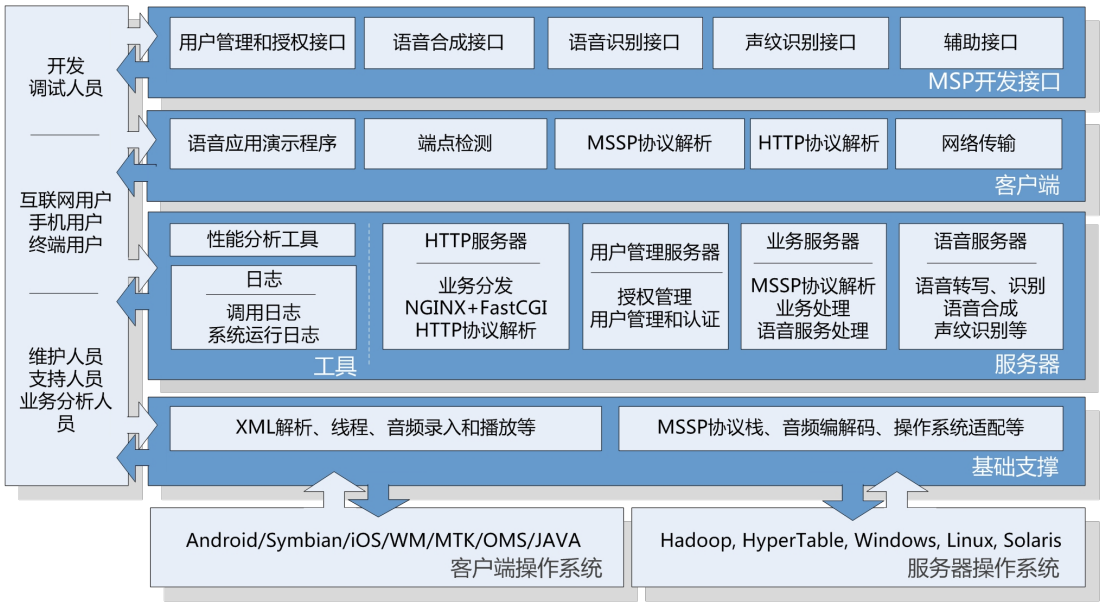
2）提供基于移动平台和桌面平台的语音应用客户端，内部集成音频处理和音频编解码模块，提供关于语音合成、语音听写、语音识别等完善的 API 和 Demo 示例。

本文档为您提供如下帮助：快速搭建开发环境，通过简单、方便、易用的开发接口以及完善的 API 和 Demo 示例，您可以将语音功能嵌入到自己客户端应用程序中构建多种语音应用。在应用程序的使用中，您不需要维护语音引擎，即可享有互联网上最好的语音服务，体验语音技术的魅力。

1. 概述

1.1. 系统架构

1.1.1. 软件架构



语音云平台软件架构图

上图蓝色区域为MSP系统的实现范围，浅色区域是与MSP密切相关的组件或第三方角色。

MSP系统主要包括语音应用接口（Speech Programming Interface，SPI）、客户端（Mobile Speech Client，MSC）、服务器（Mobile Speech Server，MSS）和基础支撑（MSP Infrastructure）四个层次，这四个逻辑层从用户到服务器操作系统底层，共同构成了完整的MSP系统架构。

- SPI

应用接口是MSP系统提供的开发接口，集成开发人员应关注这些接口的定义、功能和使用方法。

- MSC

MSC负责实现这些接口，同时封装了网络通讯、音频编解码（Audio Codec）、语音检测（VAD）、协议解析（MSSP）等功能，同时为了便于开发和使用，系统在这一层提供了一系列高效、易用的工具。

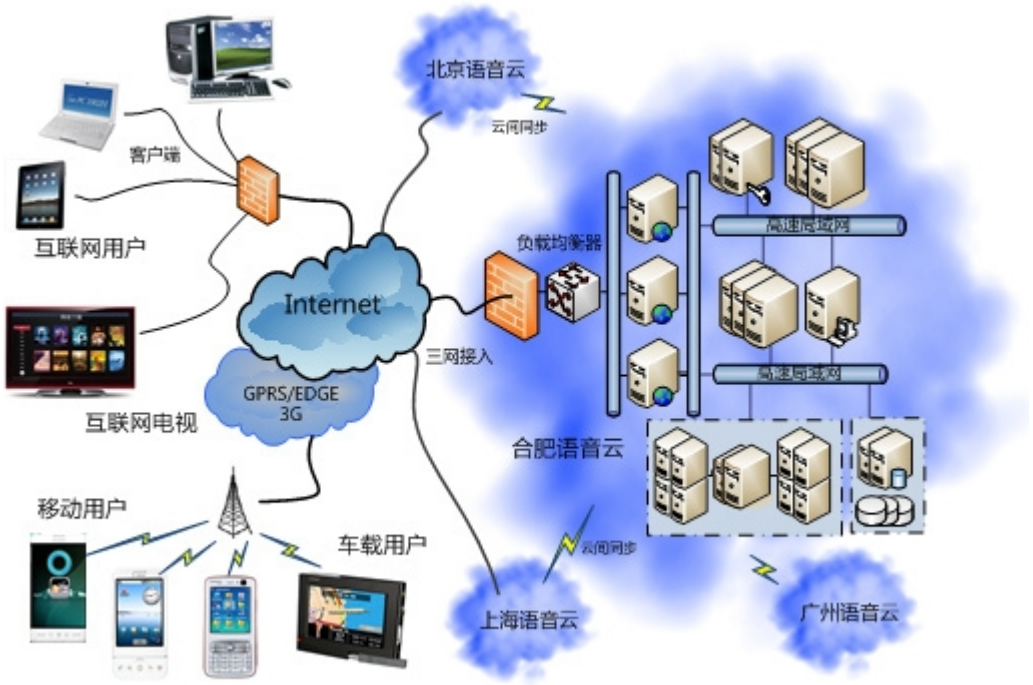
- MSS

MSS提供语音服务的服务端实现，使用服务端的识别引擎提供语音功能，同时提供管理和维护功能。

- **MSP Infrastructure**

基础支撑层是 MSP 的服务基础，负责提供适合云计算架构的负载均衡、并行计算、数据存储等功能。

1.1.2. 硬件架构



语音云平台硬件架构图

从图中可以看到，完整的MSP平台架构在Internet上，分为服务器端、移动客户端和Internet客户端三个部分。

服务器端为MSP平台的核心部分，提供HTTP应用、用户管理、语音服务等服务，位于局域网内，对外统一接入Internet，为客户端提供唯一的访问点。其中：HTTP服务器负责将客户端发送的服务请求发送至业务服务器，然后由业务服务器按照具体的服务类型进行处理，调用ISP语音应用平台获取具体的语音服务，而后把处理结果返回给HTTP服务器，再回复客户端。

互联网用户直接通过MSP服务器提供的Internet访问点使用语音服务，在集成了MSP平台提供的开发接口后即可在网络畅通的情况下在应用程序中调用语音服务。

移动用户使用智能手机通过移动运营商提供的 2G（GPRS/EDGE/CDMA）或 3G 网络接入 Internet，然后连接到 MSP 服务器获得服务。

1.2. 名词和缩略语

□ **TTS (Text to Speech)**

语音合成(Text To Speech，TTS)是一种能够将任意文字实时转换为连续的自然语音的技术，能够在任何时间、任何地点，向任何人提供语音信息服务的高效便捷手段，非常符合信息时代海量数据、动态更新和个性化查询的需求。

□ **IAT (iFly Auto Transform)**

语音听写(iFly Auto Transform，IAT)是一种使计算机能够识别人通过麦克风或者电话输入的词语或语句的技术，简单的说就是将语音中的具体内容转换成文字，更适合于日常用语的识别。

□ **ASR (Automatic Speech Recognition)**

语音识别(Automatic Speech Recognition，ASR)是在语音听写基础上的一种识别技术，着重于对某一领域或特定文法的识别，且识别结果和上传的内容紧密相关，如果用户想自定义识别的范围，在使用前，需上传相关的命令词列表或语法文件。

□ **NLP (Natural Language Processing)**

自然语言理解(Natural Language Processing，NLP)是人工智能（AI）的一个子领域，也是人工智能中最为困难的问题之一，它包含对语义的判断，使计算机不仅能够识别出语音内容，还能了解用户的意图，帮助用户准确地搜索出想要的结果，最终目的是让计算机做到像人一样的思考。

□ **ISP (iFLY Speech Platform)**

讯飞语音应用平台(iFLY Speech Platform，ISP)，是针对电信级应用场合开发的一个升级扩容方便、能提供高性能、高质量的负载均衡、方便部署、易于维护而且可以进行实时监控和维护的语音应用平台。

□ **IMS (iFLY MRCP Server)**

讯飞MRCP服务器(iFLY MRCP Server，IMS)，支持国际标准协议MRCP 1.0/2.0的语音服务平台，该平台基于ISP架构，提供对国际标准的支持。

□ **MSP (Mobile Speech Platform)**

讯飞语音云开放平台(Mobile Speech Platform，MSP)，或称为IMSP(iFLY Mobile Speech Platform)，是讯飞面向移动互联网领域开发的语音服务平台，本项目是该产品的第三个版本。

□ **MSSP (Mobile Speech Service Protocol)**

移动语音服务协议(Mobile Speech Service Protocol，MSSP)，是基于 HTTP1.1 协议扩展的语音应用协议。

1.3. 文档说明

本文档定义了科大讯飞的语音合成、语音听写和语音识别的使用说明、体系结构、API 接口，所有接口必需在联网状态下才能正常使用。

其适用的读者为使用语音 SDK 进行开发的产品设计师、软件工程师，通过阅读本文档，读者可以掌握如何集成和使用语音合成、语音听写和语音识别服务。

2. 使用说明

2.1. 开发说明

1. 使用语音服务，需要经过我们的授权，请到“http://open.voicecloud.cn”注册成为语音云开发者，并为所开发的软件申请 appid；
2. 如果开发者申请了多个 appid，不同应用需要使用与之相对应的 SDK libs，因为 SDK 中已包含该应用相关信息，导入错误的开发包将难以保证正常的语音服务；
3. 已申请 appid 的开发者，将可以免费使用语音识别、语音合成服务，但是对于文档中描述的语义、声纹等业务不具备使用权限，需要和我们签订商务协议并获得授权方可使用；
4. 应用程序开发语音功能，需要引入 SDK 中包含的 msc.jar 和 libmsc.so 动态库，so 动态库包含 arm、arm-v7a、mips、x86 四种架构，对于普通 Android 手机开发者，如果不需要支持特殊机型，只需引入 arm 架构 so 动态库即可，以减少应用程序安装包大小；
5. SDK 中 jar 库包含多个版本，在接口上并无差异，主要在界面上使用不同风格，以满足开发者差异化需求；
6. 具体业务，请参考开发示例章节。

2.2. 支持平台

1. 支持 Android 1.5 及以上版本系统；
2. 支持 arm、arm-v7a、mips、x86 架构处理器。

3. 环境搭建

Android 开发支持的操作系统为：Windows、Mac OS X 10.4.8、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：

1） 安装 JDK，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；

3） 安装 Android SDK，编者采用的版本是 android2.2，读者可以去从 Android 官方网站 http://developer.android.com/sdk/下载所需的版本；

4） 安装 ADT 插件，启动 eclipse 后，点击 eclipse 的 Help->soft update->find and install->search for new features to install->new remote site->name:https://dl-ssl.google.com/android/eclipse/，完成安装后重启 eclipse；

5） 安装 Android DDMS 和 Android Development Tools；

至此您已经完成了 Android 开发环境的搭建，更多 Android 了解请参考 SDK 的开发指导文档。

4. 语音识别

4. 1. 识别控件（RecognizerDialog）

4.1.1. 创建对象

函数原型

public RecognizerDialog(Context context, String params);

参数说明

参数名	参数解释
context	上下文环境
params	初始化参数列表，每项中间以半角逗号分隔， 如：”appid=1234567,timeout=15000” 可设置参数列表： 1. appid：应用程序ID （必选） 2. usr：语音云注册用户名，默认为null，使用匿名方式（可选） 3. pwd：用户密码 （可选） 4. timeout：网络超时时间，单位：ms，默认为20000，范围0-30000（可选） 5. server_url：默认连接语音云公网入口http://dev.voicecloud.cn/index.htm，只有特定业务才需要设置为固定ip或域名，普通开发者不需要设置（可选） 6. besturl_search：默认为1，如果server_url设置为固定ip地址，需要将此参数设置为0，表示不寻找最佳服务器。如果server_url为域名，可以将此参数设置为1（可选） 7. cancel_align_left：设置取消按钮在对话框中的位置，当设置为false时取消按钮在右边，当设置为true时取消按钮在左边。默认4.0系统版本以下为false，4.0及以上版本为true。

说明

1．请通过 http://open.voicecloud.cn 网站申请开发包，appid 已写入动态库，开发者不需要传入，如果传入将会引起参数异常错误。
2．如果反复调用此接口，只有第一次传入的params参数有效。

4.1.2. 设置识别参数

函数原型

public void setEngine (String engine, String params, String grammar) ;

参数说明

参数名	参数解释
engine	识别引擎选择，目前支持以下五种 ”sms”：普通文本转写 “poi”：地名搜索 ”vsearch”：热词搜索 ”video”：视频音乐搜索 ”asr”：语法识别
params	参数列表，每项中间以半角逗号分隔，无附加参数传null，如：” vad_bos =4000, vad_eos =2000” 可设置参数： 1．search_area：指定 engine 为 poi 搜索时，可通过此参数设定搜索区域，如：” search_area =安徽省合肥市”； 2．asr_ptt：默认为 1，当设置为 0 时，将返回无标点符号文本； 3．vad_bos：静音超时时间，即用户多长时间不说话则当做超时处理，单位：ms，engine 指定 sms 识别默认值为 5000，其他情况默认值为 4000，范围 0-10000； 4．vad_eos：后端点静音检测时间，即用户停止说话多长时间内即认为不再输入，自动停止录音，单位：ms，sms 识别默认值为 1800，其他默认值为 700，范围 0-10000； 5．plain_result：返回结果是否在内部进行 json 解析，默认值为 0，即进行解析，返回外部的内容为解析后文本。对于语义等业务，由于服务端返回内容为 xml 或其他格式，需要应用程序自行处理，这时候需要设置 plain_result 为 1，结果回调中返回的 RecognizerResult.text 字段将为未解析原始结果，由外部进行处理； 6．asr_sch：是否需要进行语义处理，默认为 0，即不进行语义识别，对于需要使用语义的应用，需要将 asr_sch 设为 1，并且设置 plain_result 参数为 1，由外部对结果进行解析；

	7. grammar_type：语法文件识别类型，如果进行语法文件识别，需要设置此参数（可选值包括 abnf、url-list、grxml，请参考有关标准）。同时设置 engine 为 null，并将语法字符串作为 grammar 参数传入，请参考语法文件识别示例； 8. request_audio_focus：是否获取音频焦点，如果设置为 true，Android 2.3 及以上版本识别会话过程中暂停后台音乐播放，识别结束继续播放。如果设置为 false，识别会话过程中不影响后台音乐播放。默认值为 true。
grammar	不进行语法识别时，grammar 参数设为”null”； 进行语法识别时，engine 参数设为”asr”； 语法识别支持两种方式，第一种方式是通过上传命令词或上传 abnf 语法文件获得的 ID 进行识别，第二种方式是直接携带语法内容进行识别。 1. 第一种方式，grammar 参数设为语法 ID，语法 ID 通过上传接口上传命令词或语法内容获得； 2. 第二种方式，grammar 参数设为语法内容字符串，同时在 params 参数中设置” grammar_type =***”进行标识；

说明

1. 调用此接口后启动录音进行识别服务，无效的参数会抛出错误信息。
2. 语法 ID 识别示例：

RecognizerDialog#setEngine(”asr”,null,” 5a398ba6be0c6cc……”);

3. abnf 语法字符串识别示例：

RecognizerDialog#setEngine(” asr”,” grammar_type =abnf”,” #ABNF……”);

4. sms 转写示例：

RecognizerDialog#setEngine(“sms”,null,null);

4.1.3. 设置录音采样率

函数原型

public void setSampleRate(RATE rate);

参数说明

参数名	参数解释
rate	录音采样率，支持 rate8k、rate11k、rate16k、rate22k 四种，默认为 rate16k

说明

1. 调用此接口后在下次识别时生效。
2. Android 手机一般只支持 8K 和 16K 两种采样率，为了获得更好的识别效果，推荐使用 16K。

4.1.4. 获取上传流量

函数原型

public int getUpflowBytes(boolean isTotal);

参数说明

参数名	参数解释
isTotal	true 表示获取应用程序启动到当前的上传流量，false 表示获取最后一次语音识别的上传流量

返回值：

与服务器交互所产生的上传流量，单位：字节（byte）。

说明

本次交互所产生的流量可以在回调接口的 onEnd 函数中调用，总流量在任意时刻调用均有效。

4.1.5. 获取下载流量

函数原型

public int getDownflowBytes(boolean isTotal);

参数说明

参数名	参数解释
isTotal	true 表示获取应用程序启动到当前的下载流量，false 表示获取最后一次语音识别的下载流量

返回值：

与服务器交互所产生的下载流量，单位：字节（byte）。

说明

无

4.1.6. 设置回调接口

函数原型

public void setListener(RecognizerDialogListener listener);

参数说明

参数名	参数解释
listener	回调接口，通知外部获取识别结果

返回值

无

说明

1．开发者需要实现此接口以获得语音识别的结果，请参考 RecognizerDialogListener 接口。

2．如果用户手动点击界面取消了当前的识别，将不会再有消息通过此接口调用。

4.1.7. 错误界面显示

函数原型

public void showErrorView(boolean showView,boolean showErrorCode);

参数说明

参数名	参数解释
showView	如果设置为 false，发生错误时，RecognizerDialog 将自动调用 dismiss 方法消失，在回调 onEnd 接口中可以获得错误信息。如果设置为 true，将停留在错误界面，默认为 true
showErrorCode	1．设置为 false，错误页面将不显示错误码，只显示错误信息，设置为 true，错误码和错误信息均显示，默认为 true； 2．此参数在设置错误页面显示的情况下有效。

4.1.8. 更多按钮显示

函数原型

public static void showMoreButton(boolean more);

参数说明

参数名	参数解释
more	1．设置为 true，在发生网络超时，将显示更多按钮，用户可以使用更多页面中的重新说话、重新获取结果，录音回放功能，如果设置为 false，将不显示更多按钮，只有重新说话可用，默认为 true，建议使用默认设置； 2．此参数在设置错误页面显示的情况下有效。

4.1.9. 销毁对象

函数原型

public boolean destory();

参数说明

无

返回值

返回 true 表示销毁成功，如果当前有会话正在进行，无法立刻释放，则返回 false。

说明

1．调用销毁接口后，正在进行的会话取消，识别实例将被释放，需要重新初始化识别控件才能进行下次识别。

2．程序运行过程中不建议频繁调用，以防止需要重新创建识别实例。程序退出时如果此接口不被调用，也不会引起程序异常，识别实例会通过垃圾回收机制进行释放。

4. 2. 识别控件回调接口（RecognizerDialogListener）

4.2.1. 结果回调

函数原型

public void onResults(ArrayList<RecognizerResult> results,boolean isLast);

参数说明

参数名	参数解释
results	识别结果，请参考 RecognizerResult 定义
isLast	true 表示最后一次结果，false 表示结果未取完

说明

控件采用边录音边上传数据的方式，可能会多次返回结果，建议用户在此接口中只保存结果内容，在 onEnd 回调中进行下一步的结果处理。

4.2.2. 结束回调

函数原型

public void onEnd(SpeechError error);

参数说明

参数名	参数解释
error	请求成功 error 等于 null，否则返回错误码

说明

本次识别过程结束，如果识别成功，对话框自动消失，调用者可以在此函数中进行下一步的处理。如果出现错误，界面不消失，显示相应错误文字，开发者不需要对错误情况进行处理，用户会根据界面提示进行下一步操作。

4. 3. 识别结果（RecognizerResult）

4.3.1. 成员说明

成员名	参数解释
String text	文本结果
int confidence	结果置信度
ArrayList<HashMap<String,String>> semanteme	语义结果，由本次识别所选择服务定义

说明

1. 命令词和语法识别时，text 表示识别结果，confidence 表示置信度，取值范围 0-100。
2. 命令词识别时，如果需要和上传的原词表进行比对，请使用 semanteme 中 key 为 contact 的字段。例：[{"sc":"50","gm":"0","w":"双鹤药业","mn":[{"contact":"双鹤药业"}]},{"sc":"50","gm":"0","w":"西藏药业","mn":[{"contact":"西藏药业"}]}，其中"contact"对应的内容为原词表内容，"w"对应的内容为识别结果(text 字段)。
3. 定制化业务的搜索结果存放在 semanteme 中，由具体的业务类型进行约定。
4. 转写结果时，text 表示转写文本，confidence 置信度默认为 100。

5. 语音合成

5.1. 语音合成控件（SynthesizerDialog）

5.1.1. 创建对象

函数原型

public SynthesizerDialog(Context context, String params);

参数说明

参数名	参数解释
context	上下文环境
params	请参考RecognizerDialog构造函数

说明

无

5.1.2. 设置合成文本

函数原型

public void setText(String text, String params);

参数说明

参数名	参数解释
text	需要合成的文本
params	初始化参数列表，每项中间以英文逗号分隔，无附加参数传null，如：”tts_buffer_time=5000” 可设置参数： 1. tts_buffer_time：缓冲多少毫秒音频之后开始播放，默认为第一句话缓冲完成开始播放，单位：ms； 2. request_audio_focus：是否获取音频焦点，如果设置为true，Android 2.3及以上版本识别会话过程中暂停后台音乐播放，识别结束继续播放。如果设置为false，识别会话过程中不影响后台音乐播放。默认值为true； 3. stream_type：系统声音类型，默认值为AudioManager.STREAM_MUSIC。可以根据需要设置为其它AudioManager的audio stream类型。

说明

1．调用此函数后，需要将文本传送到服务端进行合成，会有一定缓冲时间，缓冲时间视手机网络状况而定。

5.1.3. 设置回调接口

函数原型

public void setListener (SynthesizerDialogListener listener) ;

参数说明

参数名	参数解释
listener	合成回调的通知接口

说明

如无特殊需要可以不用实现此接口。

5.1.4. 设置合成音频采样率

函数原型

public void setSampleRate(RATE rate);

参数说明

参数名	参数解释
rate	支持 rate8k、rate16k 两种，默认为 rate16k

说明

为了获得更好的音质效果，建议使用 16k 采样率。

5.1.5. 设置发音人

函数原型

public void setVoiceName(String vcn);

参数说明

参数名	参数解释
vcn	发音人名称，可以设置为"xiaoyan"、"xiaoyu"等，默认为"xiaoyan" 更多发音人请参考附录一个性发音人列表。

说明

1. 不同的发音人代表了不同的音色，如男声、女声、童声、地方话等。
2. 部分发音人只支持中文或英文中一种，如果需要进行中英文混读，请选择语言为中英文的发音人。
3. 如果需要设置发音人为青年女声玛丽，可以按照下面的方式进行设置

SynthesizerDialog.setVoiceName(“vimary”);

5.1.6. 设置合成语速

函数原型

public void setSpeed(int speed);

参数说明

参数名	参数解释
speed	范围值：0-100，默认为50

说明

无

5.1.7. 设置合成音量

函数原型

public void setVolume(int volume);

参数说明

参数名	参数解释
volume	范围值：0-100，默认为50

说明

无

5.1.8. 设置背景音乐

函数原型

public void setBackgroundSound(String id);

参数说明

参数名	参数解释
id	背景音乐id，默认为null，表示无背景音乐

说明

目前支持 id 为”1”的背景音乐，后续会在服务端不断添加，请关注语音云开发网站。

5.1.9. 获取播放状态

函数原型

public PLAY_STATE getState();

返回值

BUFFERING：缓冲音频状态

PLAYING：正在播放

PAUSED：暂停状态

STOPED：播放完成状态

说明

无

5.1.10. 暂停播放

函数原型

public void pause();

参数说明

无

说明

合成处于 BUFFERING 和 PLAYING 状态下有效。

5.1.11. 恢复播放

函数原型

public void resume();

参数说明

无

说明

合成处于 PAUSED 状态下有效。

5.1.12. 获取上传流量

函数原型

public int getUpflowBytes(boolean isTotal);

参数说明

参数名	参数解释
isTotal	true 表示获取应用程序启动到当前的合成上传流量，false 表示获取最后一次合成的上传流量

返回值：

与服务器交互所产生的上传流量。

说明

无

5.1.13. 获取下载流量

函数原型

public int getDownflowBytes(boolean isTotal);

参数说明

参数名	参数解释
isTotal	true 表示获取应用程序启动到当前的合成下载流量，false 表示获取最后一次合成的下载流量

返回值：

与服务器交互所产生的下载流量。

说明

无

5.1.14. 销毁对象

函数原型

public boolean destory();

参数说明

无

返回值

返回 true 表示销毁成功，如果当前有会话正在进行，无法立刻释放，则返回 false。

说明

1．调用销毁接口后，正在进行的会话取消，需要重新初始化合成控件才能进行下次播放。

2．调用此接口，将会同时销毁 SynthesizerPlayer 对象，合成实例释放。

3．程序运行过程中不建议频繁调用，以防止需要重新创建合成实例。程序退出时如果此接口不被调用，也不会引起程序异常，合成实例会通过垃圾回收机制进行释放。

5. 2. 语音合成控件回调接口（SynthesizerDialogListener）

5.2.1. 结束回调

函数原型

public void onEnd(SpeechError error);

参数说明

参数名	参数解释
error	播放完成或发生错误时调用此接口，error 为 null 表示播放完成

说明

播放完成后界面会显示重新播放按钮，不自动消失，如果发生错误会提示用户重新合成，开发者无需在 onEnd 回调中进行处理。

5. 3. 语音合成播放器（SynthesizerPlayer）

通过此接口可以在后台进行语音合成播放，而不用通过 SynthesizerDialog 显示界面。

5.3.1. 创建对象

函数原型

public static SynthesizerPlayer createSynthesizerPlayer(Context context ,String params);

参数说明

参数名	参数解释
context	上下文环境
params	请参考RecognizerDialog构造函数

说明

无

5.3.2. 播放文本

函数原型

public void playText(String text, String params, SynthesizerPlayerListener listener);

参数说明

参数名	参数解释
text	需要合成的文本
params	1. 请参考 SynthesizerDialog 的 setText 接口 2. tts_interrupt_error：合成被异常打断的情况下，是否通知外部错误码。默认值为 false，不回调 onEnd 接口返回错误码。
listener	合成回调接口

说明

如果上一次合成会话没有完成又启动了新会话，上一次会话将被强制中断。如需回调上一次会话中断的错误码，可以将 tts_interrupt_error 参数设置为 true。

5.3.3. 取消合成

函数原型

public void cancel();

参数说明

无

说明

取消本次音频合成和播放。

5.3.4. 设置接口

说明

设置接口请参考 SynthesizerDialog。

5.3.5. 播放控制接口

说明

播放控制接口请参考 SynthesizerDialog。

5.3.6. 销毁对象

函数原型

public boolean destory();

参数说明

无

返回值

返回 true 表示销毁成功，如果当前有会话正在进行，无法立刻释放，则返回 false。

说明

1．调用销毁接口后，正在进行的会话取消，合成实例将被释放，需要重新初始化才能进行下次合成。

2．程序运行过程中不建议频繁调用，以防止需要重新创建合成实例。程序退出时如果此接口不被调用，也不会引起程序异常，识别实例会通过垃圾回收机制进行释放。

3．使用支持 int 参数的 destory 接口，int 参数表示有会话进行情况下的阻塞时间，如果无正在进行的会话，将会立刻返回，不会阻塞。

5. 4. 语音合成播放器回调接口（SynthesizerPlayerListener）

5.4.1. 开始播放回调

函数原型

void onPlayBegin();

参数说明

无

说明

1. 音频缓冲完成，开始播放。
2. 通过合成开始时的 tts_buffer_time 参数可以设置缓冲时间。

5.4.2. 缓冲进度回调

函数原型

void onBufferPercent(int percent,int beginPos,int endPos);

参数说明

参数名	参数解释
percent	音频缓冲进度，范围值：0-100
beginPos	缓冲音频在文本中开始位置
endPos	缓冲音频在文本中结束位置

说明

无

5.4.3. 播放进度回调

函数原型

void onPlayPercent(int percent,int beginPos,int endPos);

参数说明

参数名	参数解释
percent	播放进度，范围值：0-100
beginPos	当前播放音频在文本中开始位置
endPos	当前播放音频在文本中结束位置

说明

播放过程中会不断调用此接口通知播放进度。

5.4.4. 暂停回调

函数原型

void onPlayPaused();

参数说明

无

说明

播放过程中如果无缓冲音频，会调用此接口通知播放暂停。

5.4.5. 重新播放回调

函数原型

void onPlayResumed();

参数说明

无

说明

无缓冲音频的暂停过程中，如果接收到音频，重新恢复播放时会调用此接口。

5.4.6. 结束回调

函数原型

void onEnd(SpeechError error);

参数说明

参数名	参数解释
error	播放完成返回 null，否则返回错误码

说明

无

6. 数据上传控件

6.1. 数据上传控件（UploadDialog）

6.1.1. 创建对象

函数原型

public UploadDialog(Context context);

参数说明

参数名	参数解释
context	上下文环境

说明

1. 数据上传控件用于上传自定义的命令词表，上传成功可以获得语法文件 ID 进行命令词识别。
2. 上传数据之前，需要调用 SpeechUser 接口先进行登录操作，请参考 SpeechUser 接口。

6.1.2. 设置上传数据

函数原型

public void setContent(String name, byte[] data, String params);

参数说明

参数名	参数解释
name	上传的数据名称，如果联系人数据，可以设置为”keys”，开发者可以自己进行设置
data	上传数据的二进制流
params	参数列表

说明

1. 命令词上传示例：

支持以”,”分割的命令词组，传入数据需要以 utf-8 格式进行编码，数据名称可以设为”keys”，参数为”subject =asr,data_type =keylist”。

UploadDialog#setContent("keys",keys.getBytes("utf-8"), "subject =asr,data_type =keylist");

2. abnf 语法上传示例：

支持标准 abnf 语法，编码格式由 abnf 文件头标识确定，如：#ABNF 1.0 gb2312;则需要将 abnf 语法字符串以 gb2312 方式进行编码，数据名称可以设为”abnf”，参数为”subject =asr,data_type =abnf”。

UploadDialog#setContent("abnf",keys.getBytes("gb2312"), "subject =asr,data_type =abnf");

6.1.3. 设置回调接口

函数原型

public void setListener(SpeechListener listener);

参数说明

参数名	参数解释
listener	请参考 SpeechListener

说明

无

6.1.4. 销毁对象

函数原型

public boolean destory();

参数说明

无

返回值

返回 true 表示销毁成功，如果当前会话正在进行，无法立刻释放，则返回 false。

说明

调用销毁接口，正在进行的会话取消，上传控件将被释放，再次上传需要重新初始化。

7. 用户登录接口

用户登录接口涉及到用户登录、数据上传、下载等操作。

7. 1. 用户登录（SpeechUser）

7.1.1. 获取用户对象

函数原型：

public static SpeechUser getUser();

说明：

1．获取用户对象，用于用户登录、注销。

2．进行数据上传、声纹等相关业务前，需要先调用此接口执行登录操作。

7.1.2. 登录

函数原型：

public boolean login(Context context, String usr, String pwd, String param, SpeechListener listener);

参数说明：

参数名	参数解释
context	上下文环境
usr	用户名，通过在语音云注册用户获得，传null使用匿名方式进行登录
pwd	密码
param	请参考RecognizerDialog或SpeechRecognizer初始化接口
listener	登录回调接口

说明：

1. 进行数据上传、声纹等相关业务前，需要先调用此接口执行登录操作。
2. username 和 password 为 http://open.voicecloud.cn 网站上申请的用户名和密码。
3. 如果 username 和 password 传 null，则表示使用匿名方式进行登录，匿名用户 ID 由语音云生成，并在手机端进行保存，下次登录仍然有效，是设备访问语音云的唯一性标识。
4. 对于终端应用，建议使用匿名用户方式进行登陆，以保证终端唯一性。如果使用同样的用户名和密码，服务端会当成一个用户来处理，不建议使用这种方式。

7.1.3. 获取登录状态

函数原型：

public Login_State getLoginState ();

返回值：

登录状态，返回的 Login_State 为枚举类型，Logined 表示已登录，Unlogin 表示未登录。

7.1.4. 注销

函数原型：

public boolean logout();

返回值：

注销是否成功，如果注销成功或用户未登录，返回 true，注销失败返回 false。

7. 2. 数据上传（DataUploader）

DataUploader 对象实现的功能和 UploadDialog 一样，所不同的是不会弹出上传提示框。

7.2.1. 创建 DataUploader 对象

函数原型：

public DataUploader();

说明：

构造 DataUploader 对象，用于上传数据到语音云。

7.2.2. 上传数据

函数原型

public void uploadData(Context context,SpeechListener listener,String name,String params, byte[] uploadData);

参数说明

参数名	参数解释
context	上下文环境
listener	接口监听器
name	上传数据名称，可以由开发者自定义
params	数据携带参数值，用以说明数据类型，由服务端具体业务进行约定
data	需要上传的数据

说明

1. 上传联系人示例：

DataUploader#uploadData(context,listener,”contacts”,”subject=uup,data_type=contact”,data)

data:联系人名称之间以换行符进行分隔，拼接成字符串后以 utf-8 方式进行编码。

2. 命令词上传示例：

支持以”,”分割的命令词组，传入数据需要以 utf-8 格式进行编码，数据名称可以设为”keys”，参数为”subject =asr,data_type =keylist”

DataUploader#uploadData(context, listener, " keys ", "subject=asr,data_type=keylist", keys.getBytes("utf-8"));

3. abnf 语法上传示例：

支持标准 abnf 语法，编码格式由 abnf 文件头标识确定，如：#ABNF 1.0 gb2312;则需要将 abnf 语法字符串以 gb2312 方式进行编码，数据名称可以设为”abnf”，参数为”subject =asr,data_type =abnf”。

DataUploader # uploadData (context, listener, "subject=asr,data_type=abnf","abnf",keys.getBytes("gb2312"));

7. 3. 数据下载（DataDownloader）

7.3.1. 创建 DataDownloader 对象

函数原型：

public DataDownloader ();

说明：

构造 DataDownloader 对象，用于下载数据。

7.3.2. 下载数据

函数原型

Public void downloadData(Context context, SpeechListener listener, String param);

参数说明

参数名	参数解释
context	上下文环境
listener	接口监听器
param	请求数据描述　　（必填）

说明

下载数据需要的参数由具体业务进行约定，暂无说明。

7. 4. 语义搜索（SpeechSearcher）

7.4.1. 创建 SpeechSearcher 对象

函数原型

public SpeechSearcher ();

说明

构造 SpeechSearcher 对象，用于语义搜索。语义搜索为直接传入文本形式接口。

7.4.2. 语义搜索

函数原型

public void searchText(Context context, SpeechListener listener, String param, String text);

参数说明

参数名	参数解释
context	上下文环境
listener	接口监听器
param	搜索文本所携带描述　　（可以为空）
text	需要进行语义搜索的文本

说明

1. 文本搜索返回语义格式结果，请参考《MSC 语义解析》，注：根据客户需求我们会提供不同的版本。
2. 语义搜索需要使用纯文本内容,过长或者包含特殊符号的字符串,服务端将会返回错误。
3. 搜索结果为 utf-8 编码的二进制数据。

7.5. 通用回调接口（SpeechListener）

7.5.1. 数据回调

函数原型

public void onData(byte[] buffer);

参数说明

参数名	参数解释
buffer	服务端返回的二进制数据

说明

1. 服务端返回的数据，由具体业务进行不同处理。
2. 上传命令词只会返回一次结果，但是下载数据时可能会多次调用此接口。

7.5.2. 结束回调

函数原型

public void onEnd(SpeechError error);

参数说明

参数名	参数解释
error	error 不为 null 表示发生错误，null 表示会话成功

说明

会话成功，上传界面消失，开发者可以进行下一步的处理；如果发生错误，控件界面会提示用户重新上传或设置网络，不需要开发者进行处理。

7.5.3. 消息回调

函数原型

public void onEvent(int eventType,Bundle params);

参数说明

参数名	参数解释
eventType	消息类型
params	消息数据对象

返回值

无

说明

扩展用接口，由具体业务进行约定。

8. 个性化接口

个性化接口包含上传联系人和上传用户词表两部分。通过上传联系人，可以体验快速准确的语音识别效果；通过上传用户词表，可以获得专属用户用语习惯的独特语音体验。使用了个性化接口后，用户在进行识别时，系统会优先从已上传的联系人和词表中选择词汇进行匹配，显著提高识别准确率，从而提升用户的使用体验。接口包含在 com.iflytek.util 包中。

8.1. 联系人（Contact）

8.1.1. 创建 ContactManager 对象

函数原型

public static ContactManager createManager(Context context, ContactListener contactListener)

参数说明

参数名	参数解释
context	上下文环境
contactListener	参考 8.1.2 ContactListener 回调监听器

返回值

返回创建后的 ContactManager 对象；若以前创建了，返回以前创建的对象。

说明

1. 调用 8.1.4 同步查询联系人接口则 contactListener 可以为空
2. 调用 8.1.5 异步查询联系人接口需要实现 ContactListener（8.1.2 ContactListener 回调监听器）用于获取联系人名称列表。

8.1.2. ContactListener 回调监听器

接口原型

```
public interface ContactListener {
        void onContactQueryFinish(String contactInfos, boolean changeFlag);
 }
```

回调方法说明

onContactQueryFinish 联系人查询结束回调接口，包括联系人列表信息和是否发生修改。

参数说明

参数名	参数解释
contactInfos	联系人姓名列表
changeFlag	联系人是否变化标志

说明

1. 若联系人列表为空，返回 null；若联系人列表不为空，返回值以一个字符串的形式输

出，字符串格式为：”联系人 1\n 联系人 2\n 联系人 3\n 联系人 4”

2. contactFlag 为 true，联系人发生变更；为 false，联系人未发生变更。
3. 本回调接口仅在使用异步查询联系人接口时被回调。

8.1.3. 获取 ContactManager 对象

函数原型

public static ContactManager getManager()

返回值

若之前已经创建 ContactManager 对象，返回创建的 ContactManager 对象；若之前未创建或创建失败，则返回 null。

说明

获取 ContactManager 对象，用于获取联系人数据。

8.1.4. 同步获取所有联系人

函数原型

public String QueryAllContactsName()

返回值

若联系人列表为空，返回 null；若联系人列表不为空，返回所有联系人

说明

1. 查询的联系人来自手机和 SIM 卡。
2. 若联系人列表不为空，返回值为联系人集合的字符串，中间以\n 拼接。
3. 本接口是阻塞接口，调用此接口会有一定延迟。

8.1.5. 异步获取所有联系人

函数原型

public void asyncQueryAllContactsName()

返回值

无

说明

1. 参考 8.1.4 同步查询联系人名称
2. 本接口启用线程查询所有联系人名称，通过 ContactListener 返回所有联系人名称。

8.2. 用户词表（UserWords）

8.2.1. 词表构建

函数原型

public UserWords(String datas)

参数说明

参数名	参数解释
datas	Json 格式的用户词表字符串，null 表示创建一个空的用户词表。

说明

1. 词表包含若干词组，词组包含若干词条，数据格式如下：
   {"userword":[{"name":"词组 A","words":["词条 1","词条 2"]},{"name":"词组 B","words":["词条 1","词条 2"]}]}。
2. 词组和词条若有重复，则仅保存两条重复数据中的一条。
3. 词组和词条内容不可包含*￥#%@等特殊字符。

8.2.2. 获取词组名称列表

函数原型

public ArrayList<String> getKeys()

返回值

若词表中包含词组，返回词组名称列表；若不含词组，返回 null。

说明

需要初始化 UserWords 之后或者调用 DataDownloader#downloadData 接口后方可获得所需词组名称列表。

8.2.3. 判断词组是否存在

函数原型

public boolean hasKey(String key)

参数说明

参数名	参数解释
key	是否包含名称为 key 的词组

返回值

返回 true，词组存在；返回 false，词组不存在

说明

本函数是检测名称为 key 的词组是否存在。

8.2.4. 添加单个词条

函数原型

public boolean putWord(String key,String value)

public Boolean putWord(String value)

参数说明

参数名	参数解释
key	需要插入的词组名称，若不传入该参数表示向默认词表中添加词条。

value	需要添加的词条内容

返回值

返回 true，添加成功；返回 false 添加失败。

说明

1. 本函数是将值为 value 的词条加入名为 key 的词组。
2. 需要调用 DataUploader#uploadData 上传用户词表之后方可生效。
3. 若待插入的词组中已存在该词条，则操作不生效。待插入的词组不存在，则新建名称为 key 的词组并将 value 添加到该词组中。
4. key 或者 value 为 null 不执行添加操作，value 不可包含*￥#%@等特殊字符。

8.2.5. 添加多个词条

函数原型

public boolean putWords(String key, ArrayList<String> words)

public Boolean putWords(ArrayList<String> words)

参数说明

参数名	参数解释
key	需要插入的词组名称，若不传入该参数表示向默认词组中插入词条列表。
words	需要插入的词条列表

返回值

返回 true，添加成功；返回 false 添加失败。

说明

1. 本函数是将多个词条加入名为 key 的词组。
2. 参考 8.2.3 添加单个词条。
3. 词条列表是采用追加方式插入到词组中，重复的词条只保存一个。

8.2.6. 获取词组内容

函数原型

public ArrayList<String> getWords(String key)

public ArrayList<String> getWords()

参数说明

参数名	参数解释
key	返回名称为 key 的词组下所有词条，若不传入该参数表示获取默认词组中的词条列表。

返回值

若词组存在，返回词组中所有词条；若词组不存在，返回 null。

说明

1．需要初始化 UserWords 之后或者调用 DataDownloader#downloadData 接口后方可获得所需词条列表。

2．数据格式参考 8.2.1 用户词表构建。

8.2.7. 获取词表内容

函数原型

public String toString()

返回值

词表为空，返回{"userword":[]}；否则，返回词表中所有内容

说明

1. 需要初始化 UserWords 之后或者调用 DataDownloader#downloadData 接口后方可获得所需词条列表。
2. 获取词表的所有内容，数据格式参考 8.2.1 用户词表构建。

9. 调试设置接口

获取版本号接口包含在 com.iflytek.Version 类中，通过 getVersion 接口实现。其他设置接口包含在 com.iflytek.Setting 类中，主要用以记录调试日志，请参考如下的接口。

9.1. 获取版本号

函数原型

public static String getVersion();

返回值

版本号，示例：2.0.1013.1034

9.2. 测试日志

函数原型

public static void saveLogFile(LOG_LEVEL level,String path);

参数说明

参数名	参数解释
level	日志打印级别，如果发生一般的网络问题，可以将级别设为 detail，级别说明如下： all：所有日志 detail：高，异常分析需要的级别 normal：中，打印基本日志信息 low：低，只打印主要日志信息 none：不打印

path	日志文件保存路径，设置时请确保路径可读写，如：/sdcard/msc.log，如果设为 null，默认保存路径为/sdcard/msc/msc.log。

说明

软件发布时尽量不要保存日志，以免生成大量日志文件，如果发生网络连接等问题，可以在开发阶段将生成的日志文件发送到 msp_support@iflytek.com 获得支持。

9.3. Logcat 日志

函数原型

public static void showLogcat(boolean showlog);

参数说明

参数名	参数解释
showlog	设置为 false，将不显示 logcat 的打印信息，设置为 true，将显示 logcat 的打印信息，默认为 true，建议开启，以便出现问题时技术支持人员可以及时定位。

9.4. 网络状态检查

函数原型

public static void checkNetwork(boolean checknetwork);

参数说明

参数名	参数解释
checknetwork	设置为 true，进行识别和合成会话前，将进行网络检查，直接尝试连接服务器，false 将不进行检查，默认为 true。

说明

1．提供此设置接口的主要原因是在部分电视平台 Android 系统中，获取网络状态的结果与实际情况不一致造成。

2．如无特殊需要，建议使用默认值，这样在无网络的情况下，可以直接通过对话框设置按钮打开系统网络设置页面。

9.5. 设置语言

函数原型

public static void setLanguage(Locale lan);

参数说明

参数名	参数解释
lan	设置界面显示语言，支持简体中文-Locale.CHINA，英文-Locale.US，繁体中文 Locale.TRADITIONAL_CHINESE，设置无效值使用默认语言-简体中文

10.使用示例

10.1. Project 设置

1. 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。
2. 将 SDK.\lib 目录下 libs 文件夹拷贝到工程根目录，确保.\libs**\libmsc.so 文件存在。
   so 动态库包含 arm、arm-v7a、mips、x86 四种架构，对于普通 Android 手机开发者，如果不需要支持特殊机型，只需引入 arm 架构 so 动态库即可，以减少应用程序安装包大小。
3. 在工程 AndroidManifest.xml 文件中添加如下权限

```
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
```

4. 工程混淆设置：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.**{*;}
```

5. 为显示高质量的 UI 效果，需要在 AndroidMainfest.xml 文件中，加入如下的声明

```
<supports-screens android:anyDensity="true"/>
```

10.2. 语音转写示例

```
// 创建识别对话框,需传入正确appid
RecognizerDialog isrDialog = new RecognizerDialog(this, "appid=12345678");
isrDialog.setEngine(”sms”,null,null);
isrDialog.setListener(recognizeListener);
isrDialog.show();
String text = “”;
// 转写回调监听器.
RecognizerDialogListener recognizeListener = new RecognizerDialogListener(){
    // 识别结果回调接口
    public void onResults(ArrayList<RecognizerResult> results, boolean
    isLast){
    // 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.
        for(int i = 0; i < results.size(); i++)
    text += results.get(i).text;
}

    // 会话结束回调接口.
```

```
public void onEnd(SpeechError error) {
    // error为null表示会话成功，可在此处理text结果，error不为null，表示发生错误，对话框停留在错误页面
}
};
```

10.3. 语音合成示例

```
// 创建SynthesizerDialog对象，需传入正确appid
// 显示对话框方式
 SynthesizerDialog    synDialog = new SynthesizerDialog(this, "appid=12345678");
synDialog.setListener(synListener);
synDialog.setVoiceName("xiaoyu");
synDialog.setText("安徽科大讯飞信息科技股份有限公司", null);
synDialog.show();
SynthesizerDialogListener synListener = new SynthesizerDialogListener(){
// 会话结束回调接口
public void onEnd(SpeechError error) {
    // error为null，表示播放完成，error不为null，表示发生错误
}
};
// 不显示对话框方式
SynthesizerPlayer player = SynthesizerPlayer.createSynthesizerPlayer(this, "appid=12345678");
player.setVoiceName("vixm");
player.playText(" 安 徽 科 大 讯 飞 信 息 科 技 股 份 有 限 公 司 ","tts_buffer_time=2000",
synbgListener);
SynthesizerPlayerListener synbgListener = new SynthesizerPlayerListener (){
void onPlayBegin(){
    // 播放开始回调，表示已获得第一块缓冲区音频开始播放
    }
void onBufferPercent(int percent,int beginPos,int endPos){
    // 缓冲回调，通知当前缓冲进度
    }
void onPlayPaused(){
    // 暂停通知，表示缓冲数据播放完成，后续的音频数据正在获取
    }
void onPlayResumed(){
    // 暂停通知后重新获取到后续音频，表示重新开始播放
    }
void onPlayPercent(int percent,int beginPos,int endPos){
    // 播放回调，通知当前播放进度
    }
void onEnd(SpeechError error){
    }
```

```
};
```

10.4. 登录示例

```
// 用户只有在登录状态才能进行数据上传、下载、语义搜索、密码文本下载等操作
// 注册用户登录需要上传用户名和密码，非匿名方式
// 用户名和密码传null，表示使用匿名方式进行登录,每台设备具备唯一性，建议使用此种方式
SpeechUser.getUser().login(this, null, null, "appid=12345678", loginListener);
// 登录回调监听器
SpeechListener loginListener = new SpeechListener()
{
    public void onEnd(SpeechError error) {
    // 若error为null，表示登录成功；反之登录失败
    }
    public void onData(byte[] buffer) {
    // 登录操作服务端不会返回数据，忽略此接口
    }
    public void onEvent(int eventType, Bundle params) {
    // 扩展消息，忽略此接口
    }
};
```

10.5. 命令词上传示例

```
// 上传语法需要先进行登录操作，请参考登录示例
SpeechUser.getUser().login(this, null, null, "appid=12345678", null);
// 命令词示例，命令词中间以半角逗号进行分割
String keys = “张三,李四,Andy,Tom,张学友”;
// 创建上传对话框
UploadDialog uploadDialog = new UploadDialog(this);
uploadDialog.setListener(uploadListener);
// subject=asr,data_type=keylist表示上传命令词参数
uploadDialog.setContent("keys",keys.getBytes("UTF-8"), "subject=asr,data_type=keylist");
uploadDialog.show();
// 语法ID
String grammarID = null;
// 上传数据回调监听器
SpeechListener uploadListener = new SpeechListener()
{
    // 会话结束回调
    public void onEnd(SpeechError error) {
        // 若error不为null，对话框将显示错误提示
```

```
        }
        // 回调数据
        public void onData(byte[] buffer) {
        // 获得语法ID
            grammarID = new String(buffer);
        }
        public void onEvent(int eventType, Bundle params) {
        }
};
```

10.6. abnf 语法上传示例

```
// 上传语法需要先进行登录操作，请参考登录示例
SpeechUser.getUser().login(this, null, null, "appid=12345678", null);
// ABNF语法示例，可以说”北京到上海”
String grammar = “#ABNF 1.0 gb2312;
    language zh-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;”
// 创建上传对话框
UploadDialog uploadDialog = new UploadDialog(this);
uploadDialog.setListener(uploadListener);
// subject=asr,data_type=abnf表示上传命令词参数
uploadDialog.setContent("abnf", grammar.getBytes("gb2312"), "subject=asr,data_type=abnf");
uploadDialog.show();
// 语法ID
String grammarID = null;
// 上传数据回调监听器
SpeechListener uploadListener = new SpeechListener()
{
    // 会话结束回调
    public void onEnd(SpeechError error) {
        // 若error不为null，对话框将显示错误提示
    }
    // 回调数据
    public void onData(byte[] buffer) {
    // 获得语法ID
        grammarID = new String(buffer);
    }
    public void onEvent(int eventType, Bundle params) {
    }
```

```
};
```

10.7. 语法 ID 识别示例

```
// 语法ID通过上传命令词或abnf语法获得，请参考上传命令词、上传abnf语法示例
String grammarID = "adkjdnn……";
// 创建识别对话框
RecognizerDialog isrDialog = new RecognizerDialog(this, "appid=12345678");
isrDialog.setEngine("asr", null, grammarID);
isrDialog.setListener(recognizeListener);
isrDialog.show();
// 识别回调监听器
RecognizerDialogListener recognizeListener = new RecognizerDialogListener(){
    // 识别结果回调
    public void onResults(ArrayList<RecognizerResult> results,boolean
        isLast){
    // results表示识别结果列表，可能会返回多个识别结果，默认通过置信度进行排序
}
    //会话结束回调
public void onEnd(SpeechError error) {
    // 若error不为null，对话框将显示错误提示
}
};
```

10.8. 语法文件识别示例

```
// 创建对话框，appid已写入动态库，开发者不需要传入
RecognizerDialog isrDialog = new RecognizerDialog(this, "appid=12345678");
isrDialog.setListener(recognizeListener);
// ABNF语法示例，可以说”北京到上海”
String grammar = “#ABNF 1.0 gb2312;
    language zh-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;”
isrDialog.setEngine("asr", "grammar_type=abnf", grammar);
isrDialog.show();
//识别回调监听器
RecognizerDialogListener recognizeListener = new RecognizerDialogListener(){
    //识别结果回调
    public void onResults(ArrayList<RecognizerResult> results,boolean
```

```
            isLast){
        // results表示识别结果列表，可能会返回多个识别结果，默认通过置信度进行排序
        }
        //会话结束回调
    public void onEnd(SpeechError error) {
        // 若error不为null，对话框将显示错误提示
    }
    };
```

10.9. 语义识别示例

```
    // 创建对话框，appid已写入动态库，开发者不需要传入
    RecognizerDialog isrDialog = new RecognizerDialog(this, "appid=12345678");
    isrDialog.setListener(recognizeListener);
    // 设置语义识别的参数，plain_result=1表示服务端返回的结果SDK内部不需要进行解析，由
    // 开发者根据服务端约定格式进行处理，asr_sch=1表示需要进行语义处理
    isrDialog.setEngine("sms", “asr_sch=1,plain_result=1”, null);
    isrDialog.show();
    private RecognizerDialogListener recognizeListener = new
    RecognizerDialogListener() {
        public void    onResults(ArrayList<RecognizerResult> results,boolean isLast) {
    // 开发者通过获取 RecognizerResult.text 字段进行自定义处理
        }
        public void onEnd(SpeechError error) {
        // 若error不为null，对话框将显示错误提示
        }
    };
    // 对于部分语义业务，需要上传联系人名称，以支持打电话给***的说法，可以按照下面方式
    // 传入，上传数据格式utf-8，联系人中间以\n换行符进行分割
    // 上传命令词需要先进行登录操作，请参考登录示例
    SpeechUser.getUser().login(this, null, null, "appid=12345678", null);
    UploadDialog uploadDialog = new UploadDialog(this);
    String contacts = "王小贰\n张小山\n李四\n科大讯飞\ngoogle\n汪山三\n章栋";
    byte[] datas = contacts.getBytes("utf-8");
    uploadDialog.setListener(uploadListener);
    // 上传联系人参数为"subject=uup, data_type=contact"
    uploadDialog.setContent("contacts",datas, "subject=uup,data_type=contact");
    uploadDialog.show();
    // 上传联系人接口回调
    SpeechListener    uploadListener = new SpeechListener()
    {
    @Override
```

```
public void onEnd(SpeechError error) {
    // error为null表示上传成功
}
@Override
public void onData(byte[] buffer) {
    // 上传联系人服务端不返回数据，忽略此消息
}
@Override
public void onEvent(int arg0, Bundle arg1) {
}
};
```

10.10. 后台上传命令词示例

```
// 用户登录
SpeechUser.getUser().login(this, null, null, "appid=12345678", null);
// 创建上传数据对象
DataUploader    uploader = new DataUploader();
uploader.setListener(uploadListener);
String keys = “张三,李四,Andy,Tom,张学友”;
// subject=asr,data_type=keylist表示上传命令词参数
uploader.uploadData(this,"contact","subject=asr,data_type=keylist"， keys.getBytes("UTF-8"));
// 语法ID
String grammarID = null;
SpeechListener uploadListener = new SpeechListener()
{
    //会话结束接口
    public void onEnd(SpeechError error) {
    // error 不为 null 表示上传失败
    }
    public void onData(byte[] buffer) {
    // 获得语法ID
    grammarID = new String(buffer);
    }
    public void onEvent(int eventType, Bundle params) {
    // 扩展接口，忽略此消息
    }
};
```

10.11. 语义搜索示例

```
// 用户登录
SpeechUser.getUser().login(this, null, null, "appid=12345678", null);
// 创建搜索对象
SpeechSearcher searcher = new SpeechSearcher();
searcher.searchText(this, searchListener, null, “科大讯飞”);
SpeechListener searchListener = new SpeechListener()
{
    // 会话结束接口
    public void onEnd(SpeechError error) {
    // error 不为 null 表示搜索失败
    }
    // 返回搜索结果
    public void onData(byte[] buffer) {
    // 搜索结果，utf-8 编码
    String text = new String(arg0,"utf-8")
    }
    public void onEvent(int eventType, Bundle params) {
    // 扩展接口，忽略此消息
    }
};
```

10.12. 用户词表上传和下载示例

```
//用户词表  示例内容，数据格式请参考<8.2.1词表构建>
UserWords userwords = new UserWords();
userwords.putWord("词条1");
userwords.putWord("词条2");
userwords.putWord("词条3");
//调用toString获取用户词表的json字符串，再转换为二进制数组.
byte[] datas = userwords.toString().getBytes("utf-8");
//用户词表上传的参数为subject=uup,data_type=userword
uploadDialog.setContent("userwords", datas,
"subject=uup,data_type=userword");
//词表数据上传
uploadDialog.show();
//词表数据下载
DataDownloader dataDownloader = new DataDownloader();
dataDownloader.downloadData(NLPDemo.this, downloadlistener,
"subject=spp,data_type=userword");
/**
```

```
 * 用户词表下载监听器
 */
SpeechListener downloadlistener = new SpeechListener()
{
    // 返回下载数据
    public void onData(byte[] arg0) {
    // 获得用户词表内容new String(arg0,"utf-8")
    }
    //会话结束接口
    public void onEnd(SpeechError error) {
        // error不为null表示上传失败
    }
    public void onEvent(int eventType, Bundle params) {
        // 扩展接口，忽略此消息
}
};
```

10.13. 联系人上传示例

```
ContactManager mgr = ContactManager.getManager();
if(mgr == null)
mgr = ContactManager.creatManager(NLPDemo.this, new ContactListener()
{
     /**
      * 联系人信息回调接口
      * @param contactInfos  联系人信息，数据格式为：张三/n李四/n王五
      * @param changeFlag    为true时表示联系人信息发生变更，为false表示未发生变更.
    */
public void onContactQueryFinish(String contactInfos, boolean changeFlag) {
    try {
        // 若非首次上传且联系人信息未发生变更，则不进行上传操作
            //若联系人信息未空，则不进行上传操作
        if(!changeFlag || TextUtils.isEmpty(contactInfos))
            return;
           // 联系人信息转成utf-8格式二进制串
        byte[] datas = contactInfos.getBytes("utf-8");
// 联系人上传的参数为subject=uup,data_type=contact
uploadDialog.setContent("contacts",datas,"subject=uup,data_type=contact");
        uploadDialog.show();
    } catch (UnsupportedEncodingException e) {
```

```
            e.printStackTrace();
        }
        }
});
// 异步查询联系人接口，通过onContactQueryFinish接口回调
mgr.asyncQueryAllContactsName();
```

附录一：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	

附录二：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	网络连接断开
ERROR_NETWORK_TIMEOUT	20002	网络连接超时
ERROR_NET_EXPECTION	20003	网络交互异常
ERROR_INSUFFICIENT_PERMISSIONS	20004	应用程序授权不足
ERROR_INVALID_RESULT	20005	返回结果为空
ERROR_SERVER_CONNECT	20006	无法连接到服务器
ERROR_INVALID_PARAM	20007	无效的参数
ERROR_CLIENT	20008	客户端应用程序错误
ERROR_AUDIO_RECORD	20009	麦克初始化错误
ERROR_NO_MATCH	20010	无匹配的识别结果
ERROR_SPEECH_TIMEOUT	20011	无有效的音频输入
ERROR_INVALID_ENCODING	20012	无效的编码
ERROR_EMPTY_UTTERANCE	20013	合成或上传数据为空
ERROR_FILE_ACCESS	20014	文件读写错误
ERROR_PLAY_MEDIA	20015	合成音频播放中发生错误
ERROR_MEMORY_WRANING	20016	存储空间不足
ERROR_TEXT_OVERFLOW	20017	参数个数超过限制
ERROR_LOGIN	20018	登录错误
ERROR_IN_USE	20019	多路会话并发错误
ERROR_INVALID_DATA	20020	客户端数据解析错误
ERROR_INVALID_GRAMMAR	20021	识别语法错误
ERROR_INVALID_LOCAL_RESOURCE	20022	服务端资源错误
ERROR_LOGIN_INVALID_USER	20023	用户 ID 无效
ERROR_LOGIN_INVALID_PWD	20024	用户密码无效
ERROR_PERMISSION_DENIED	20025	引擎授权错误
ERROR_BROWSER_NOT_INSTALLED	20026	浏览器未安装
ERROR_TTS_INTERRUPT	20027	合成被异常打断
UNKNOWN	30000	未知错误