云知声智能语音交互平台

Android SDK V3.0 开发指南

北京云知声信息技术有限公司

Beijing Unisound Information Technology Co., Ltd.

重要声明

版权声明

商标声明

北京云知声信息技术有限公司的产品是北京云知声信息技术有限公司专有。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。本文档可能涉及北京云知声信息技术有限公司的专利（或正在申请的专利）、商标、版权或其他知识产权，除非得到北京云知声信息技术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

北京云知声信息技术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。未得到北京云知声信息技术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

1. 概述

云知声智能语音交互平台旨在使第三方应用便利的集成和使用语音理解(语音云和语义云)服务。

本 SDK 能够帮助开发者迅速开发基于语音识别、语义理解及语音合成需求的客户端软件。SDK 内置了基础的技术服务，并不包含内容服务，为开发者提供创建一个全新自定义应用的入口。

本文档默认读者已经掌握 Android 应用程序开发的相关知识。

1.1. 目的

本文档对云知声 Android SDK 接口定义进行说明。

文档读者为使用云知声 Android SDK 进行开发的产品设计师、软件工程师。

1.2. 范围

本文档定义云知声 Android SDK 的使用说明、体系结构、API 接口。

不包含核心引擎的性能定义，也不包含其它配套或附赠产品的使用说明。

2. 使用说明

2.1. 开发说明

本文属于入门级文档，旨在帮助开发者快速学习云知声 Android SDK 的使用，并应用到实际的开发工作中，开发者仅需关注文档中所提供的接口方法而不用了解具体实现。

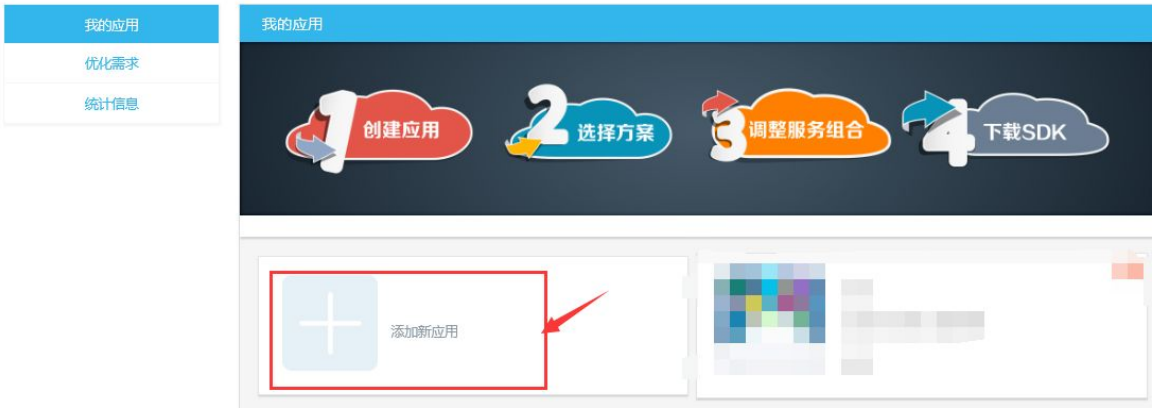
2.2. 开发前准备

对于个人开发者使用云知声语音服务，请到“http://dev.hivoice.cn”注册成为云知声开发者，并创建应用，在“我的应用”中获取 AppKey 和 AppSecret。使用应用授权码可以帮助开发者监控语音识别服务的使用情况。

1） 开发者在开放平台中点击“登录/注册”。

开发者通过注册的方式完成账号，注册账号可通过邮箱或者手机号码注册，或第三方账号登陆。

2） 开发者完整注册，成功登录后，点击“开发中心”，点击“添加新应用”。



3） 进入“创建新应用”页面，开发者可以根据自己的需求选择方案从而创建一个应用。

方案是应用的模板，内置了与特定领域相关的语音技术服务（语音识别、语音合成等）以及相应的内容服务（天气、股票、音乐、视频等），如果不确定使用领域，则可以选择“通用方案”进行自定义的配置使用。

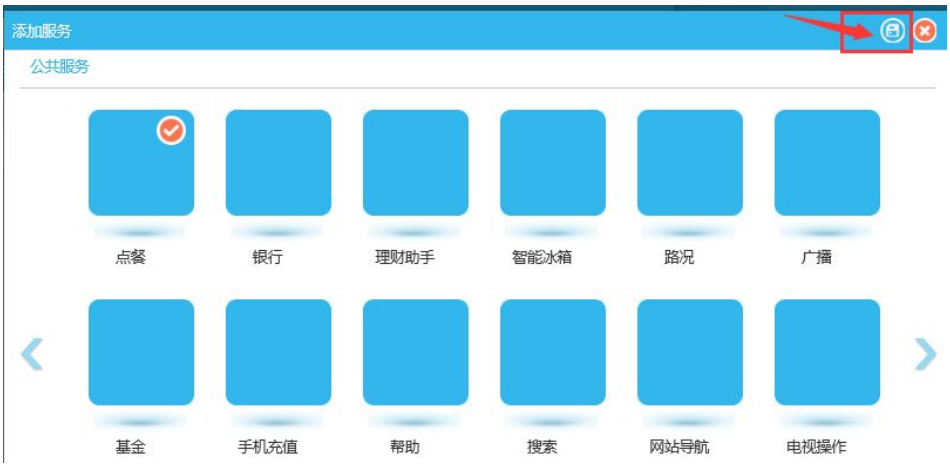
4） 创建应用成功后，或者在“我的应用”页面点击某个应用后，进入应用的详细信息页面

开发者可以查看应用对应的 appKey & appSecret，修改方案的应用描述，其他信息不可修改。

5） 开发者选择了相关的方案，方案中内置了相关的技术服务和内容服务。

如果方案中内置的内容服务无法满足开发者现有的需求，则可以点击“内容服务”增加。

开发者可以选择自己需要的内容服务，保存后即可添加到现有应用中。



2.3. 支持的平台

系统：支持 Android 2.1及以上系统。通过minSdkVersion参数检测。

机型：手机、平板及其它智能终端设备。

架构：支持 arm、x86 及 mips 平台。

外设：设备上有麦克风。

网络：有网络连接。

3. 环境搭建

本文档默认开发者已掌握 JDK、Eclipse、Android SDK、ADT 等相关 Android 开发工具的搭建，如要查看 Android 开发环境搭建的相关流程，请参考 Android 开发环境搭建。

1) 在 Eclipse 中建立 Android 工程。

2) 将开发工具包中的 libs 目录拷贝到工程根目录下，确保 libuscasr.so 文件存在。

说明：so 动态库包含 armeabi、armeabi-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 armeabi 架构 so 动态库即可，以减少应用程序安装包大小。

3) 在 Eclipse 中右键单击工程，选择 Project->Properties->Java Build Path->Libraries->Add JARS 或 ADD External JARS 导入工程 libs 目录下的 usc.jar 文件。

4) 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.READ_PHONE_STATE"/>
```

各权限的使用说明

名称	用途
android.permission.RECORD_AUDIO	允许应用使用麦克风
android.permission.INTERNET	允许应用联网,发送语音数据至服务器,获得识别结果
android.permission.ACCESS_NETWORK_STATE	获取当前网络状态，优化录音参数及网络参数
android.permission.ACCESS_WIFI_STATE	获取当前 wifi 状态， 优化录音参数及网络参数
android.permission.READ_PHONE_STATE	获取用户手机的 IMEI，用来唯一标示用户

4. 语音识别

4.1. 基本调用流程

```
//1. 创建对象
SpeechUnderstander mUnderstander = new SpeechUnderstander(this, Config.appKey,
Config.secret);
//2. 设置参数
mUnderstander.setOption(SpeechConstants.ASR_SERVICE_MODE,SpeechConstants.ASR_SER
VICE_MODE_NET);
//3. 设置回调监听
mUnderstander.setListener(new SpeechUnderstanderListener() {
        public void onResult(int type, String jsonResult) {}
        public void onEvent(int type, int timeMs) {}
        public void onError(int type, String errorMSG) {}
});
//4. 识别引擎初始化
mUnderstander.init(null);
//5. 开始识别
mUnderstander.start();
```

4.2. 在线识别

4.2.1. 调用流程

```
//1. 创建语音识别对象，appKey和secret通过 http://dev.hivoice.cn/ 网站申请
SpeechUnderstander mUnderstander = new SpeechUnderstander(this, Config.appKey,
Config.secret);
//2. 设置参数，设置识别模式为在线识别。更多识别参数的设置可以参考SDK30_Developer_API
下相关类的javadoc
mUnderstander.setOption(SpeechConstants.ASR_SERVICE_MODE,SpeechConstants.ASR_SER
VICE_MODE_NET);
//3. 语音识别对象回调监听
mUnderstander.setListener(new SpeechUnderstanderListener() {
      //结果回调
      public void onResult(int type, String jsonResult) {
            switch (type) {
            case SpeechConstants.ASR_RESULT_NET:
            // 在线识别结果，通常onResult接口多次返回结果，识别结果以json格式返回，更多
            json返回信息参考附录3
                  break;
```

```
            }
        }
        //语音识别事件回调，支持的回调类型见4.2.3 事件回调
        public void onEvent(int type, int timeMs) {}
        //发生错误时回调，详见附录2错误列表说明
        public void onError(int type, String errorMSG) {}
});

//4. 识别引擎初始化
mUnderstander.init(null);

//5. 开始语音识别
mUnderstander.start();
```

4.2.2. 参数设置

1. 参数的设置如下所示：

```
//创建语音识别对象
SpeechUnderstander mSpeechUnderstander = new SpeechUnderstander(this,
Config.appKey,Config.secret);

//设置识别参数
mSpeechUnderstander.setOption(SpeechConstants.ASR_SERVICE_MODE,SpeechConstants.A
SR_SERVICE_MODE_NET);
```

2. 参数在com.unisound.client.SpeechConstants类中，下面所示key及value省去 com.unisound.client.SpeechConstants，主要参数如下，更多参数设定，请参考 SDK30_Developer_API下相关类的javadoc。

Key	Value 类型	注释	备注
ASR_SERVICE_MODE	int	设置识别模式	在线模式： ASR_SERVICE_MODE_NET
ASR_SAMPLING_RATE	int	设置采样率	自动采样率： ASR_SAMPLING_RATE_BANDWIDTH_AUTO
			采样率 8000: ASR_SAMPLING_RATE_8K
			采样率 16000: ASR_SAMPLING_RATE_16K
ASR_DOMAIN	String	设置识别领域	例如: general,movietv,song,poi,medical,food,customized,fridge,eshopping,home,law,kar
ASR_VOICE_FIELD	boolean	设置远近讲	默认为 VOICE_FIELD_NEAR
ASR_LANGUAGE	String	设置识别语言	默认为 LANGUAGE_MANDARIN 会默认设置为公有云服务地址

4.2.3. 事件回调

```
public void onEvent(int type, int timeMs) {
        switch (type) {
        case SpeechConstants.ASR_EVENT_ENGINE_INIT_DONE:
        // 初始化完成
                break;
        case SpeechConstants.ASR_EVENT_RECORDING_PREPARED:
        // 录音准备就绪
                break;
        case SpeechConstants.ASR_EVENT_RECORDING_START:
        //录音设备打开
                break;
        case SpeechConstants.ASR_EVENT_SPEECH_DETECTED:
        //用户开始说话
                break;
        case SpeechConstants.ASR_EVENT_RECORDING_STOP:
        // 停止录音
                break;
        case SpeechConstants.ASR_EVENT_VAD_TIMEOUT:
        // 收到用户停止说话事件
                break;
        case SpeechConstants.ASR_EVENT_NET_END:
        // 在线识别结束
                break;
        case SpeechConstants.ASR_EVENT_RECOGNITION_END:
        // 识别结束
                break;
        case SpeechConstants.ASR_EVENT_VOLUMECHANGE:
        // 说话音量实时返回
                int volume = (Integer)mUnderstander.
                                getOption(SpeechConstants.GENERAL_UPDATE_VOLUME);
                break;
        default:
                break;
        }
}
```

4.3. 上传用户数据

4.3.1. 调用流程

```
//1. 创建语音识别对象，appKey和secret通过 http://dev.hivoice.cn/ 网站申请
SpeechUnderstander mUnderstander = new SpeechUnderstander(this, Config.appKey,
Config.secret);
//2. 语音识别对象回调监听
mUnderstander.setListener(new SpeechUnderstanderListener() {
      //结果回调
      public void onResult(int type, String jsonResult) {}
      //事件回调
      public void onEvent(int type, int timeMs) {
            if (type == SpeechConstants.ASR_EVENT_USERDATA_UPLOADED ) {
            //用户上传数据结束
            }
      }
      //发生错误时回调，详见附录2错误列表说明
      public void onError(int type, String errorMSG) {}
});
//3. 识别引擎初始化
mUnderstander.init(null);

//4. 构造需要上传的用户数据
Map<Integer, List<String>> _HashMap = new HashMap<Integer, List<String>>();
List<String> _NameList = new ArrayList<String>();
_NameList.add("刘德华");
_NameList.add("周杰伦");
_HashMap.put(SpeechConstants.UPLOAD_DATA_NAME, _NameList);

//5. 上传用户数据
mUnderstander.uploadUserData(_HashMap);
```

4.4. 本地识别

4.4.1. 调用流程

```
//1. 创建语音识别对象，appKey和secret通过 http://dev.hivoice.cn/ 网站申请
SpeechUnderstander mUnderstander = new SpeechUnderstander(this, Config.appKey,
Config.secret);
//2. 设置参数，设置识别模式为本地识别。更多识别参数的设置可以参考SDK30_Developer_API
下相关类的javadoc
mUnderstander.setOption(SpeechConstants.ASR_SERVICE_MODE,SpeechConstants.ASR_SER
VICE_MODE_LOCAL);
//3. 语音识别对象回调监听
mUnderstander.setListener(new SpeechUnderstanderListener() {
      //结果回调
      public void onResult(int type, String jsonResult) {
            switch (type) {
            case SpeechConstants. ASR_RESULT_LOCAL:
            // 离线识别结果，识别结果以json格式返回，更多json返回信息参考附录3
                  break;
            }
      }
      //语音识别事件回调，支持的回调类型见4.3.3事件回调
      public void onEvent(int type, int timeMs) {}
      //发生错误时回调，详见附录2错误列表说明
      public void onError(int type, String errorMSG) {}
});

//4. 识别引擎初始化
mUnderstander.init(null);

//5. 开始语音识别
mUnderstander.start();
```

4.4.2. 参数设置

1. 参数的设置如下所示：

```
//创建语音识别对象
SpeechUnderstander mSpeechUnderstander = new SpeechUnderstander(this,
Config.appKey,Config.secret);

//设置识别参数
mSpeechUnderstander.setOption(SpeechConstants.ASR_SERVICE_MODE,SpeechConstants.ASR_SERVICE_MODE_LOCAL);
```

2. 参数在com.unisound.client.SpeechConstants类中，下面所示key及value省去com.unisound.client.SpeechConstants，主要参数如下，更多参数设定，请参考SDK30_Developer_API下相关类的javadoc。

Key	Value 类型	注释	备注
ASR_SERVICE_MODE	int	设置识别模式	离线模式： ASR_SERVICE_MODE_LOCAL

4.4.3. 事件回调

```
public void onEvent(int type, int timeMs) {
        switch (type) {
        case SpeechConstants.ASR_EVENT_ENGINE_INIT_DONE:
        // 初始化完成
                break;
        case SpeechConstants.ASR_EVENT_RECORDING_PREPARED:
        // 录音准备就绪
                break;
        case SpeechConstants.ASR_EVENT_RECORDING_START:
        //录音设备打开
                break;
        case SpeechConstants.ASR_EVENT_SPEECH_DETECTED:
        //用户开始说话
                break;
        case SpeechConstants.ASR_EVENT_RECORDING_STOP:
        // 停止录音
                break;
```

```
        case SpeechConstants.ASR_EVENT_VAD_TIMEOUT:
        // 收到用户停止说话事件
            break;
        case SpeechConstants.ASR_EVENT_LOCAL_END:
        // 本地识别结束
            break;
        case SpeechConstants.ASR_EVENT_RECOGNITION_END:
        // 识别结束
            break;
        case SpeechConstants.ASR_EVENT_VOLUMECHANGE:
        // 说话音量实时返回
            int volume = (Integer)mUnderstander.
                        getOption(SpeechConstants.GENERAL_UPDATE_VOLUME);
            break;
        default:
            break;
        }
}
```

4.5. 本地唤醒

4.5.1. 基本调用流程

```
//1. 创建语音识别对象，appKey和secret通过 http://dev.hivoice.cn/ 网站申请
SpeechUnderstander mUnderstander = new SpeechUnderstander(this, Config.appKey,
Config.secret);
//2. 设置参数，设置识别模式为本地识别
mUnderstander.setOption(SpeechConstants.ASR_SERVICE_MODE,SpeechConstants.ASR_SER
VICE_MODE_LOCAL);
//2. 语音识别对象回调监听
mUnderstander.setListener(new SpeechUnderstanderListener() {
    //结果回调
    public void onResult(int type, String jsonResult) {}
    //语音识别事件回调，支持的回调类型见4.4.2事件回调
    public void onEvent(int type, int timeMs) {}
    //发生错误时回调，详见附录2错误列表说明
    public void onError(int type, String errorMSG) {}
});

//4. 识别引擎初始化
mUnderstander.init(null);

//5. 开始唤醒
mUnderstander.start("wakeup");
```

4.5.2. 事件回调

```
public void onEvent(int type, int timeMs) {
        switch (type) {
        case SpeechConstants.ASR_EVENT_ENGINE_INIT_DONE:
        // 初始化完成
                break;
        case SpeechConstants.ASR_EVENT_RECORDING_PREPARED:
        // 录音准备就绪
                break;
        case SpeechConstants.ASR_EVENT_RECORDING_START:
        // 录音设备打开
                break;
        case SpeechConstants.ASR_EVENT_SPEECH_DETECTED:
        //用户开始说话
                break;
        case SpeechConstants.ASR_EVENT_RECORDING_STOP:
        // 停止录音
                break;
        case SpeechConstants.WAKEUP_EVENT_RECOGNITION_SUCCESS:
        // 唤醒成功
                break;
        default:
                break;
        }
}
```

5. 语音合成

5.1. 调用流程

```
//1. 创建语音合成对象, appKey和secret通过 http://dev.hivoice.cn/ 网站申请
SpeechSynthesizer mTTSPlayer = new SpeechSynthesizer(this, Config.appKey, Config.secret);
//2. 设置语音合成参数,设置语音合成模式为本地合成。更多识别参数的设置可以参考
SDK30_Developer_API下相关类的javadoc
mTTSPlayer.setOption(SpeechConstants.TTS_SERVICE_MODE,
SpeechConstants.TTS_SERVICE_MODE_NET);
//3. 设置回调监听
mTTSPlayer.setTTSListener(new SpeechSynthesizerListener() {
        // 语音合成事件回调，支持的回调类型见5.3事件回调
        public void onEvent(int type) {}
        // 语音合成错误回调,错误输出格式及错误列表见附录。
        public void onError(int type, String errorMSG) {}
});
//4. 初始化语音合成引擎
mTTSPlayer.init(null);;
//5. 开始语音合成并播报
mTTSPlayer.playText ("北京云知声信息技术有限公司");
```

5.2. 参数设置

1. 参数的设置如下所示：

```
//创建语音合成对象
SpeechSynthesizer mTTSPlayer = new SpeechSynthesizer(this, Config.appKey, Config.secret);

//设置参数
mSpeechUnderstander.setOption(SpeechConstants. TTS_SERVICE_MODE,SpeechConstants.
TTS_SERVICE_MODE_NET);
```

2. 参数在com.unisound.client.SpeechConstants类中，下面所示key及value省去com.unisound.client.SpeechConstants，主要参数如下，更多参数设定，请参考SDK30_Developer_API下相关类的javadoc。

Key	Value 类型	注释	备注
TTS_SERVICE_MODE	int	设置识别模式	在线模式：TTS_SERVICE_MODE_NET 离线模式：TTS_SERVICE_MODE_LOCAL
TTS_KEY_VOICE_SPEED	int	设置合成语速	范围 0 ~ 100
TTS_KEY_VOICE_PITCH	int	设置合成音高	范围 0 ~ 100
TTS_KEY_VOICE_VOLUME	int	设置合成音量	范围 0 ~ 100
TTS_KEY_SAMPLE_RATE	String	设置合成码率	例如：”16000”
TTS_KEY_STREAM_TYPE	int	设置合成采样率	例如：AudioManager.STREAM_MUSIC
TTS_KEY_PLAY_START_BUFFER_TIME	int	设置播放开始缓冲时间	例如：0 ~ 500 单位
TTS_KEY_FRONT_SILENCE	int	设置语音开始段的静音时长	0 ~ 1000 单位 ms
TTS_KEY_BACK_SILENCE	int	设置语音结尾段的静音时长	0 ~ 1000 单位 ms

5.3. 事件回调

```
public void onEvent(int type) {
    switch (type) {
            case SpeechConstants.TTS_EVENT_INIT:
                // 初始化成功回调
                break;
            case SpeechConstants.TTS_EVENT_SYNTHESIZER_START:
                // 开始合成回调
                break;
            case SpeechConstants.TTS_EVENT_SYNTHESIZER_END:
                // 合成结束回调
                break;
            case SpeechConstants.TTS_EVENT_BUFFER_BEGIN:
                // 开始缓存回调
                break;
            case SpeechConstants.TTS_EVENT_BUFFER_READY:
                // 缓存完毕回调
                break;
            case SpeechConstants.TTS_EVENT_PLAYING_START:
                // 开始播放回调
                break;
            case SpeechConstants.TTS_EVENT_PLAYING_END:
                // 播放完成回调
                break;
            case SpeechConstants.TTS_EVENT_PAUSE:
                // 暂停回调
                break;
            case SpeechConstants.TTS_EVENT_RESUME:
                // 恢复回调
                break;
            case SpeechConstants.TTS_EVENT_STOP:
                // 停止回调
                break;
            case SpeechConstants.TTS_EVENT_RELEASE:
                // 释放资源回调
                break;
            default:
                break;
            }
    }
```

6. 语义理解

6.1. 调用流程

```
//1. 创建语音语义理解对象，appKey和secret通过 http://dev.hivoice.cn/ 网站申请
SpeechUnderstander mUnderstander = new SpeechUnderstander(this, Config.appKey,
Config.secret);
//2. 设置语音语义理解参数，设置识别模式为在线识别。更多识别参数的设置可以参考
SDK30_Developer_API下相关类的javadoc
mUnderstander.setOption(SpeechConstants.ASR_SERVICE_MODE,SpeechConstants.ASR_SER
VICE_MODE_NET);
//3. 语音语义理解对象回调监听
mUnderstander.setListener(new SpeechUnderstanderListener() {
//结果回调
   public void onResult(int type, String jsonResult) {
        switch (type) {
        case SpeechConstants.ASR_RESULT_NET:
            // 语音语义理解结果在jsonResult中，json格式返回，更多json返回信息参考附录3
              break;
        }
      }
      //发生错误时回调 更多错误信息参考附录2
      public void onError(int type, String errorMSG) {}
});

//4. 识别引擎初始化
mUnderstander.init(null);

//5. 开始语音语义理解
mUnderstander.start();
```

6.2. 参数设置

1. 参数的设置如下所示：

例如：

```
//创建语音语义理解对象
SpeechUnderstander mUnderstander = new SpeechUnderstander(this, Config.appKey,
Config.secret);

//设置参数
mUnderstander.setOption(SpeechConstants.NLU_ENABLE, true);
```

2. 参数在com.unisound.client.SpeechConstants类中，下面所示key及value省去com.unisound.client.SpeechConstants，主要参数如下，更多参数设定，请参考SDK30_Developer_API下相关类的javadoc。

Key	Value 类型	注释	备注
NLU_ENABLE	boolean	设置同步请求语义结果	true:同步请求语义 false:不请求语义
NLU_SCENARIO	String	设置语义理解场景	例如： music,incar,video,ecommerce,videoDefault
NLU_SERVER_ADDR	String	设置语义解析服务器	例如：192.168.0.13:8080
GENERAL_CITY	String	设置解析城市信息	例如：北京
GENERAL_GPS	String	设置 gps 信息	例如： 40.0027465820312,116.328216552734

附录 1：Android 开发环境搭建

本 SDK 支持的操作系统为：Windows、Mac OS X 10.4.8、Linux。

这里将介绍 Windows 上安装环境的搭建：

1） 安装 JDK，开发者可以从 Java 官网

http://www.oracle.com/technetwork/java/javase/downloads/index.html 下载所需的版本；

2） 安装 Eclipse Java IDE，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，可以从 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。

附录 2：错误代码说明

错误代码	代码解释
-30001	VAD 超时错误
-30002	说话时间超出限制
-30003	数据压缩错误
-30004	不合法参数错误
-61001	启动录音失败
-61002	录音异常
-62001	识别异常
-63001	上传个性化数据服务器拒绝
-63002	上传个性化数据网络连接失败
-63003	上传个性化数据不能为空
-63007	上传个性化数据内容太多
-63009	上传个性化数据过于频繁
-63010	上传个性化数据编码失败
-63011	上传场景数据:服务器拒绝
-63012	上传场景数据:网络连接失败
-63013	上传场景数据:不能为空
-63017	上传场景数据:内容太多
-63019	上传场景数据:编码失败

-63020	上传场景数据:上传过于频繁
-71001	语义服务器错误
-71002	语义请求空错误
-63004	离线识别插入词表名称错误
-63005	离线识别插入词表内容错误
-63006	离线识别插入词表内容错误
-63591	离线引擎内部错误
-63592	离线引擎通讯错误
-63593	离线引擎压缩格式错误
-63595	离线引擎内存分配错误
-63596	离线引擎服务未运行
-63597	离线引擎超时错误
-63598	离线引擎超时错误
-63599	离线引擎未授权
-63600	离线引擎转义错误
-63999	离线引擎通用错误
-63502	离线引擎未初始化
-63503	离线编译模块未初始化
-63504	离线编译模块错误
-63505	离线编译 INSERTVOCAB_EXT 错误
-63531	SDK 初始化错误

-63532	SDK Start 错误
-63540	SDK 唤醒错误
-64001	初始化错误
-90001	合成模型生成失败
-90002	合成模型加载失败
-90003	合成文本为空错误
-90004	离线合成 setText 错误
-91100	在线合成初始化错误
-91101	播放线程打开 audioSource 错误
-91102	播放异常错误
-91103	离线合成引擎未初始化
-91002	在线请求初始化错误（可能未联网）
-91003	请求超时
-91004	http 协议错误
-91005	音频解码错误
-91007	未联网 或 网络未连接成功
-91008	请求过程中网络错误
-91009	请求过程中网络错误
-91131	合成 文本文件为空 /文本文件过长
-91132	合成 文本文件为空 /文本文件过长
-91134	获取信息错误

-91138	handle 设置错误
-91154	加密校验错误
-91158	未设置 secret
-911XX	参数设置错误
-91725	Appkey 验证错误
-91735	服务器超时
-917XX	http 请求错误
-918XX	语义请求错误
-91900	在线 Login 失败

附录 3 识别结果 json 字段说明

字段名称	说明
net_asr	在线识别结果
net_nlu	在线语义结果
local_asr	离线识别结果

net_asr 字段说明：

字段名称	说明
result_type	结果类型，partial 为部分结果，full 为全部结果,change 为可变结果
last_result	是否为最后一次返回，true 为最后一次结果
recognition_result	识别结果
engine_mode	当前引擎模式，mix 为混合模式，local 为离线模式，net 为在线模式

local_asr 字段说明：

字段名称	说明
result_type	结果类型，partial 为部分结果，full 为全部结果
score	识别分数
recognition_result	识别结果
engine_mode	当前引擎模式，mix 为混合模式，local 为离线模式，net 为在线模式

示例：


```
{
"net_asr":[
{
"result_type":"full",
"last_result":true,
"recognition_result":"12345。",
"engine_mode":"mix"
}
],
"net_nlu":[
{
"history":"cn.yunzhisheng.chat",
"service":"cn.yunzhisheng.chat",
"responseId":"aed855a985dc45989a1124b3e9405e93",
"text":"12345。",
"code":"ANSWER",
"general":{
"type":"T",
"text":"上山打老虎"
},
"rc":0
}
],
"local_asr":[
{
"result_type":"full",
"score":-6.22,
"recognition_result":" 你 打 3 4 5 0 ",
"engine_mode":"mix"
}
]
}
```

FAQ

1. 如何联系我们?

   如果应用开发过程中遇到问题，可随时与我们联系，联系方式如下：

   联系电话：(+8610-) 62369899-664　　传真：(+8610-) 82601009

   请发邮件至：support@unisound.com