科大讯飞股份有限公司

IFLYTEK CO.,LTD.

科大讯飞 MSC 集成指南

目录

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、语音评测等类和函数的基本使用。关于各类的函数和参数更详细的说明，请参考**《MSC Reference Manual.html》**；在集成过程有疑问，可登录语音云开发者论坛，查找答案或与其他开发者交流：http://bbs.xfyun.cn 。下载 SDK 请前往 讯飞开放平台。

安卓 MSC SDK 的功能从调用开始到结果返回，大多使用接口（Interface）回调（Callback）的方式返回结果和状态。更多关于接口和回调的介绍，可以参考 Java 语言的相关语法说明文档。

注意：

- 此文章的代码，仅为用于示例函数调用和参数设置的代码片段，很可能有参数被引用，却未曾声明等情况，请开发者不必过于考究其中的细节。更详细的示例，请参考 SDK 包中的 samples 目录下的示例工程。示例代码中返回的结果内容，可参考附录。
- 为了减少 SDK 包在应用中占用的大小，官网在下载单个功能的 SDK 包时，libmsc.so 可能并不包含其他功能，如下载人脸的 SDK 包时，可能不包含离线唤醒或离线合成等功能（在应用使用不包含的功能时，会出现崩溃或报 20021 的错误）。应该下载对应功能的 SDK 包使用，或下载组合的 SDK 包。

MSC SDK 的主要功能接口如下图所示：

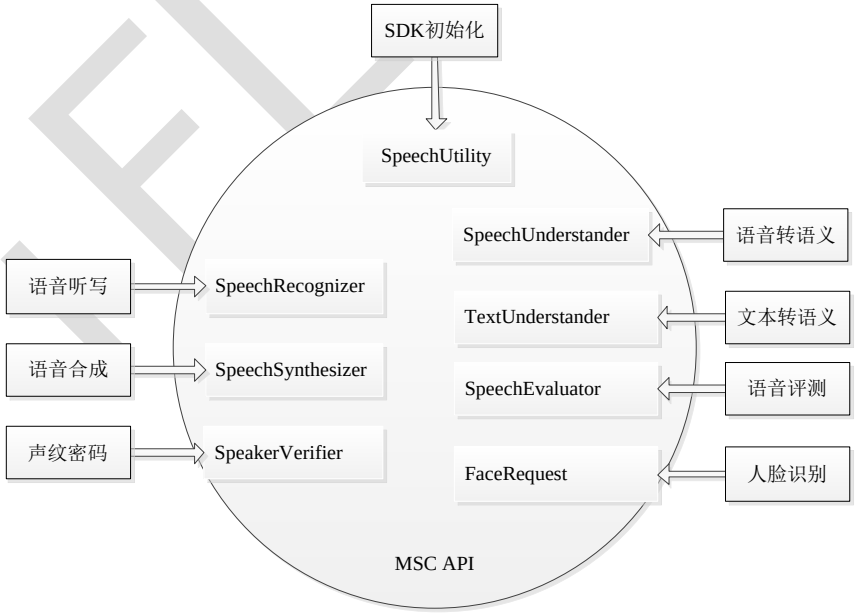


图 1 MSC 功能结构图

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：

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字内容，能识别常见的词汇、语句、语气并自动断句。
语法识别	判断所说的内容是否与预定义的语法相符合，主要用于判断用户是否下达某项命令。
语义理解	分析用户语音或文字的意图，给出相应的回答，如输入“今天合肥的天气”，云端即返回今天合肥的天气信息。
唤醒	通过说出特定的唤醒词（如“芝麻开门”）来唤醒处于休眠状态下的终端设备。
唤醒+识别	在唤醒的同时对用户所说的内容进行语法识别。
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	根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	应用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

2.1 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：

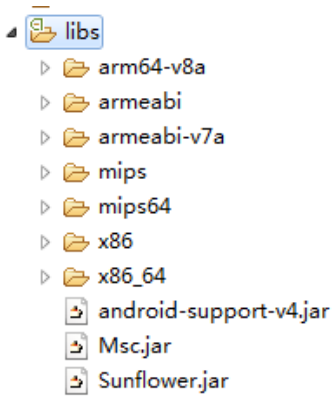


图 2 导入 SDK

注意：

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。

2.2 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，请在 proguard.cfg 中添加如下代码：

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

-keepattributes Signature
```

2.3 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 之间添加任何空字符或者转义符
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：

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

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”。

2.4 引擎类型

MSC SDK 有几种引擎类型（ENGINE_TYPE），此文章中我们主要关注并介绍以下两种：

- 在线引擎（TYPE_CLOUD），又称为云端模式，需要使用网络，速度稍慢，并产生一定流量，但有更好的识别和合成的效果，如更高的识别匹配度，更多的发音人等。
- 离线引擎（TYPE_LOCAL），又称为本地模式，不需要使用网络，且识别和合成的速度更快，但同时要求购买并使用对应的离线资源（下载使用对应离线功能的 SDK 包），或安装语记（语记的更多介绍，见后面章节）。

需要说明的是，在线引擎下，结果返回速度基本决定于用户网络的带宽限制。如在合成或识别下，默认的音频格式为 16000 HZ 采样，16 bit 精度，单声道，raw pcm，Little-endian；在未压缩的情况下，为 16000 * 16 = 256000 bit/s（比特每秒）。此时，如果要听写上传，或下载合成到的音频，时长为 t 秒，用户网络带宽为 x mpbs（兆比特每秒），则需要时长为 256000 * t / (x(2^10)(2^10))，假设 t 为 10，x 为 1，则 10 秒的音频，在 1M 的带宽下的传输时间约为 2.44s。

特别是在识别时，主要网络数据交互在音频的上传过程，此时网络上行带宽决定了音频上传的快慢，影响语音云服务器收到音频的快慢，继而决定了结果返回的快慢。

针对网络带宽的影响结果问题，MSC SDK 在识别音频上传和合成音频下载时，都做了相应的优化：

- 会话模式(ssm)，默认开启。只要 SDK 获取到一部份音频数据，就会开始上传到语音云服务器，而不是等到整段识别音频数据都获取到再上传。在实时的录制音频并进行识别时，此优化效果尤其明显：音频的录制需要时间，而 SDK 会利用这些时间，每录制到一小段音频，就开始上传到语音云服务器，且在有部份小分句（如有停顿的地方）的结果时，就会把结果返回给客户端。待音频录制完时，音频也已即将完成上传，此时，结果返回就更快，几乎能达到说完即得到结果的情况。更特别的，VAD（关于 VAD 的更多说明，参考《MSC Reference Manual.html》）生效的情况下，在应用层还未告知 SDK 已完成音频录制时，结果可能已返回。

音频压缩（aue），并默认开启。发送端对上下行的音频进行压缩（在客户端由 MSC SDK 自动压缩），压缩比约为 10:1，并在接收端解压还原（在客户端由 MSC SDK 自动解压），大大减少带宽占用，并减少网络交互的时间占用。

2.5 引擎模式

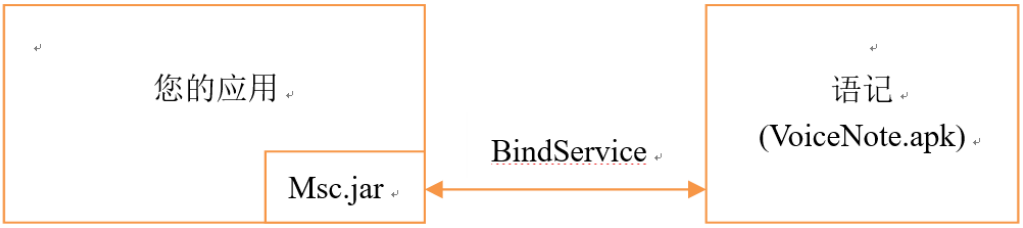
2.5.1 模式介绍

在引擎类型设置为离线引擎（TYPE_LOCAL）时，MSC SDK 提供两种方式（ENGINE_MODE）来使用离线功能：MSC，语记。

MSC 模式（MODE_MSC），使用 MSC SDK 本身提供的离线模块，需要在下载 SDK 时，选择离线功能，以购买对应的离线功能资源；

语记模式（MODE_PLUS），使用讯飞语记应用提供的离线模式，离线功能资源由语记应用自带，不需要另外购买，但需要安装语记应用——应用在集成时，可以通过代码检查设备中是否已安装语记，并自动下载安装语记，检查已安装语记的资源等。可以通过以下链接下载语记，或扫描下面的二维码下载，了解语记：

应用、MSC SDK 与语记间的关系大致如下图：



在引擎模式中，还有自动模式（MODE_AUTO），由 SDK 自动决定使用 MSC 模式，还是语记模式。在自动模式下，引擎类型为在线时，SDK 自动选择 MSC 模式；引擎类型为离线时，SDK 自动选择语记模式。

需要注意的是，为了方便应用的集成，MSC SDK 针对下仅有在线功能的 SDK，和含离线功能（唤醒、合成、识别）的 SDK，提供的 SDK 包会不一样：仅有在线功能的 SDK，默认的引擎模式为自动模式，方便应用使用语记提供的离线服务；含有离线功能的 SDK ，默认引擎模式为 MSC 模式，方便应用使用 MSC SDK 自带提供的离线服务。同时，为了减少 SDK 包的大小，在线 SDK 可能没有离线 SDK 的部份类。如果要混用 SDK 包，应用应明确指定使用的引擎模式。

在使用离线功能时，语记模式和 MSC 模式的区别在于，语记模式下，资源不必另外购买即可使用，但同时，资源覆盖程度也受语记的限制；MSC 模式下，如果有更多的需求，可以联系我们进行商务合作定制开发，提供更丰富的资源。

2.5.2 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-请参见下面示例及 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": "common"
      }
    ]
  }
}
```

3 语音识别

语音识别（**SpeechRecognizer**），包括听写、语法识别功能。语音识别技术(Auto Speech Recognize， 简称 ASR)即把人的自然语言音频数据转换成文本数据。除了听写、语法识别外，还有语义理解 SpeechUnderstander（见后面章节）。关于文本数据转语音的功能，请参考语音合成类 SpeechSynthesizer （见后面章节）。

语法识别，是基于语法规则，将与语法一致的自然语言音频转换为文本输出的技术。语法识别的结果值域只在语法文件所列出的规则里，故有很好的匹配率，另外，语法识别结果携带了结果的置信度，应用可以根据置信分数，决定这个结果是否有效。语法识别多用于要更准确结果且有限说法的语音控制，如空调的语音控制等。在使用语法识别时，应用需要先构建一个语法文件上传给服务器，并在会话时，传入语法 ID，以使用该语法。

听写，是基于自然语言处理，将自然语言音频转换为文本输出的技术。语音听写技术与语法识别技术的不同在于，语音听写不需要基于某个具体的语法文件，其识别范围是整个语种内的词条。在听写时，应用还可以上传个性化的词表，如联系人列表等，提高列表中词语的匹配率（见后面章节）。

3.1 构建语法

MSC SDK 识别语法主要为两种：ABNF 和 BNF 格式。前者用于在线语法识别，后者用于离线语法识别。本文只对构建和使用语法的 SDK 调用过程进行介绍，关于语法的规则说明，请参考语法开发指南；关于上传应用级（对同一 APPID 所有设备均生效）语法文件，参考讯飞开放平台，以及《MSC Reference Manual.html》关于 SpeechRecognizer 类的说明。

3.1.1 构建 ABNF 语法

如前文所述，ABNF 语法为在线识别使用的语法，故在构建时，应指定引擎类型为在线引擎，构建时的语法类型为 ABNF，如：

```
// 设置引擎类型
mAsr.setParameter( SpeechConstant.ENGINE_TYPE, SpeechConstant.TYPE_CLOUD );

/* 其中 "abnf" 指定语法类型为 ABNF， grammarContent 为语法内容，grammarListener 为构建结果监听器*/
ret = mAsr.buildGrammar( "abnf", grammarContent, grammarListener );
```

构建语法状态通过监听器 grammarListener 获取，当构建成功时，将在回调中返回一个 grammarID，将在使用语法识别时用到。

3.1.2 构建 BNF 语法

BNF 语法为离线识别使用的语法，在构建时，除了指定引擎为本地引擎，语法类型为 BNF 外，还必须指定离线资源的路径（MSC 模式下，需下载使用对应的离线识别 SDK），语法构建的路径——本地语法构建结果文件保存的路径，或者指定为语记模式：

```
// 设置引擎类型
mAsr.setParamter( SpeechConstant.ENGINE_TYPE, SpeechConstant.TYPE_LOCAL );
// 设置引擎模式
mAsr.setParamter( SpeechConstant.ENGINE_MODE, engineMode );
if( SpeechConstant.MODE_MSC.equals(engineMode) ){
    // 设置语法结果文件保存路径，以在本地识别时使用
    mAsr.setParameter( ResourceUtil.GRM_BUILD_PATH, grmPath );
    //设置识别资源路径
    mAsr.setParameter( ResourceUtil.ASR_RES_PATH, asrResPath );
}
/* 其中 "bnf" 指定语法类型为 BNF， grammarContent 为语法内容，grammarListener 为构建结果监听器*/
ret = mAsr.buildGrammar( "bnf", grammarContent, grammarListener );
```

构建语法状态通过监听器 grammarListener 获取，当构建成功时，语法文件将保存到由 grmPath 指定的目录中，将在语法识别时用到（MSC 模式下）。

3.2 语法识别

在使用在线语法识别时，如果要使用已经通过官网上传（参考上文）的语法文件，则不需要再设置语法 ID 参数；而使用离线语法识别时，需要设置本地语法名字（在语法文件中定义），如果使用语记模式，则不需要设置语法文件路径和资源文件路径。

```
//设置引擎类型
mAsr.setParameter( SpeechConstant.ENGINE_TYPE, engineType );

if( SpeechConstant.TYPE_LOCAL.equals(engineType) ){
    //离线引擎，指定使用语记，还是 MSC 模式
    mAsr.setParameter( SpeechConstant.ENGINE_MODE, engineMode );
    if( SpeechConstant.MODE_MSC.equals(engineMode) ){
        // MSC 模式时，设置本地识别资源，需下载使用对应的离线识别 SDK
        mAsr.setParameter( ResourceUtil.ASR_RES_PATH, asrResPath );
```

```
        // MSC 模式时，设置语法构建路径(构建语法时指定的路径值)
        mAsr.setParameter( ResourceUtil.GRM_BUILD_PATH, grmPath );
    }//end of if msc mode

    // 设置本地语法名字，在语法文件中定义了此名字值
    mAsr.setParameter( SpeechConstant.LOCAL_GRAMMAR, grammarName );
}else{
    //在线引擎使用 MSC 模式即可
    mAsr.setParameter( SpeechConstant.ENGINE_MODE, SpeechConstant.MODE_MSC
 );

    //使用网站上传的语法文件时，只明确指定 SUBJECT，不用指定语法 ID；使用在应用上
传的则相反。
    if( usingWebsideGrammar ){
        mAsr.setParameter( SpeechConstant.CLOUD_GRAMMAR, null );
        mAsr.setParameter( SpeechConstant.SUBJECT, "asr" );
    }else{
        mAsr.setParameter( SpeechConstant.CLOUD_GRAMMAR, cloudGrammarID );
    }//end of if-else using grammar in webside or not
}//end of if-else local or not

ret = mAsr.startListening( mRecognizerListener );
```

此外，本地引擎在语法文件构建后，还可以通过更新词典，更新指定规则中的词语，如联系人等，见后面更新词典章节内容。此外，MSC SDK 还提供了录音交互的对话框控件，参考后面识别对话框的内容。

3.3 语音听写

目前 MSC 模式的离线听写暂未开放购买，如大量需求，可通过文章最后的联系方式，与我们联系进行商务合作。但语记提供了离线听写的功能。

```
//设置语法 ID 和 SUBJECT 为空，以免因之前有语法调用而设置了此参数；或直接清空所有
参数，具体可参考 DEMO 的示例。
mAsr.setParameter( SpeechConstant.CLOUD_GRAMMAR, null );
mAsr.setParameter( SpeechConstant.SUBJECT, null );

mAsr.setParameter( SpeechConstant.ENGINE_TYPE, engineType );
if( SpeechConstant.TYPE_LOCAL.equals(engineType) ){
    //使用默认的语记模式
    mAsr.setParameter( SpeechConstant.ENGINE_MODE, SpeechConstant.MODE_PLUS
 );
}else{
    //使用默认的 MSC 模式
```

```
    mAsr.setParameter( SpeechConstant.ENGINE_MODE, null );
}

mAsr.startListening(mRecogListener);
```

应用可以通过上传词典（又称个性化用户热词），提高听写的匹配率，参考后面更新词典的章节。MSC SDK 还提供了录音交互的对话框控件，参考后面识别对话框的内容。

3.4 更新词典

更新词典，又被称为个性化热词上传，包括更新本地语法文件的词典，以及更新在线听写的词典。词典的内容格式及更详细调用说明，参考《MSC Reference Manual.html》中 SpeechRecognizer 类的 updateLexicon 函数介绍。

3.4.1 在线听写词典

无论在哪一种语言中，不同的单词或字（word），或多或少，都会有相似的发音（pronounce）。尤其在汉语中，这种现象更普遍，如当一个人说 /zhang/ /s[h]an/ 时，对应的词语的组成，可能是 {张，章，彰，...} {三，姗，珊，...}，这些文字的组合，在汉语的习惯中出现频率最高的，当然是“张三”了。

而在听写返回结果时，会结合上下文，把日常生活中，出现频率最高的词汇返回给客户端。这时，如果我们实际想要的结果并不是出现频率最高的词汇，如上文中我们实际要的是“张姗”——这样的情况在手机联系人信息中经常会出现，此时听写结果就不是我们想要的。这种情况下，我们可以通过上传个性化热词的方式，把在同样发音情况下，自己希望最优先匹配的词语告知语音云服务器。

个性化热词通过应用调用 SDK 函数上传时，影响的范围是，当前 APPID 应用的当前设备——即同一应用，不同设备里上传的热词互不干扰；同一设备，不同 APPID 的应用上传的热词互不干扰。另外，更新的热词仅对对应的语言区域(LANGUAGE)和方言(ACCENT)（在部份特殊场景，此两个参数可能被 "ent" 参数代替）生效，如果听写时指定了不同的 LANGUAGE，ACCENT，或 "ent"，则更新热词时，也应当指定对应的参数值。

如果需要一次上传热词，更新给当前 APPID 的所有使用设备，则可通过开发者平台网站上传应用级热词：讯飞开放平台。

词典上传后，也会在语义理解（见后面章节）中生效。

```
mAsr.setParameter( SpeechConstant.ENGINE_TYPE, SpeechConstant.TYPE_CLOUD );


// lexiconName 为词典名字，lexiconContents 为词典内容，lexiconListener 为回调监听器
ret = mAsr.updateLexicon( lexiconName, lexiconContents, lexiconListener );
```

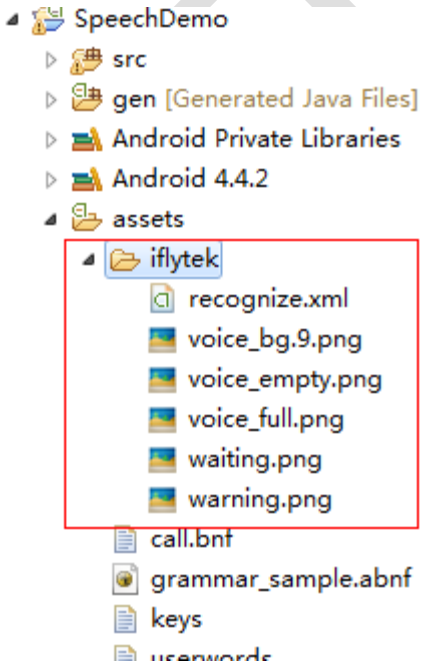
3.4.2 离线语法词典

离线语法词典，旨在更新已构建的语法文件中某个规则里的内容，因此在更新时，需要指定识别资源路径，语法文件路径，语法列表（语法文件的语法名字）。

```
mAsr.setParameter( SpeechConstant.ENGINE_TYPE, SpeechConstant.TYPE_LOCAL );

// 指定资源路径
mAsr.setParameter( ResourceUtil.ASR_RES_PATH, asrResPath );

// 指定语法路径
mAsr.setParameter( ResourceUtil.GRM_BUILD_PATH, grmPath );

// 指定语法名字
mAsr.setParameter( SpeechConstant.GRAMMAR_LIST, grammarName );

// lexiconName 为词典名字，lexiconContents 为词典内容，lexiconListener 为回调监听器
ret = mAsr.updateLexicon( lexiconName, lexiconContents, lexiconListener );
```

3.5 识别对话框

为了便于快速开发，SDK 提供了识别时用户交互（User Interface）的对话框控件类 RecognizerDialog，又被称为语音输入 UI。使用时，需先将 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 添加资源所示：



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);

//若要将 RecognizerDialog 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的结果
// mDialog.setParameter("asr_sch", "1");
// mDialog.setParameter("nlp_version", "3.0");

//3.设置回调接口
mDialog.setListener( mRecognizerDialogListener );

//4.显示 dialog，接收语音输入
mDialog.show();
```

在显示对话框后，录音自动开始，RecognizerDialog 中包含了根据当前状态显示不同图片的处理，如声音的大小，错误的提示；同时，点击对话框内任意地方，可结束录音，点击对话框外，则取消会话；出现错误后，再点击对话框内，可启动下一次会话。应用根据回调状态，进行结果和错误的处理。

4 语音合成

与语音听写相反，语音合成（**SpeechSynthesizer**）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。

语音合成中，主要参数包括合成的

- 语言（LANGUAGE，中文、英文等）
- 方言（ACCENT，中文的普通话，粤语等）
- 发音人特征（性别，年龄，语气）
- 语速（SPEED）
- 音量（VOLUME）
- 语调（PITCH）
- 音频采样率（SAMPLE_RATE）

在 MSC SDK 参数中的前三者（语言、方言和特征）基本由发音人决定——即不同的发音人，支持不一样的语言、方言和特征，参考附录中的发音人列表。

如果使用的是离线合成，MSC 模式下还必须设置合成资源的路径，需下载使用对应的离线合成 SDK。

```
mTts.setParameter( SpeechConstant.ENGINE_TYPE, engineType );
mTts.setParameter( SpeechConstant.ENGINE_MODE, engineMode );

if( SpeechConstant.TYPE_LOCAL.equals(engineType)
    &&SpeechConstant.MODE_MSC.equals(engineMode) ){
    // 需下载使用对应的离线合成 SDK
    mTts.setParameter( ResourceUtil.TTS_RES_PATH, ttsResPath );
}

mTts.setParameter( SpeechConstant.VOICE_NAME, voiceName );

final String strTextToSpeech = "科大讯飞，让世界聆听我们的声音";
mTts.startSpeaking( strTextToSpeech, mSynListener );
```

5 语义理解

语义理解，分为文本语义和语音语义，主要是把自然语言内容，转换为有一定结构的文本数据，使应用能够抓取其中的重点数据，理解用户的使用意图，进行下一步的处理。

文本语义，即将自然语言的文本，转换（进行理解）为一定结构的文本数据。如“今天合肥的天气怎么样”这句话，在假设内容有限时，实际可以在应用里直接分解出用户的意图——查询合肥的天气。然而应用很难从众多的文本中，理解每一句话里用户的意图——当覆盖面变广时，在硬件计算速度较低的 PC 和移动设备里，仅字符串匹配计算所用的开销时间也是无法接受的，而通过语义理解的服务器可以做到。

语音语义， 是先把音频数据转为听写结果数据——自然语言的文本，再由服务器自动进行文本语义理解，相当于在文本语义前，先进行听写。

当前语义理解仅有在线模式。

SDK 申请时，默认未开通语义理解功能（使用时报 10402 错误），可通过 AIUI 开放平台开通，并选择需要的语义场景。

5.1 使用 SpeechUnderstander

从 1115 以前的版本（含 1115）更新到 1116 以上版本的 SDK 的应用，已开通的语义，若在新版本 SDK 使用时，结果中有错误码，则需要通过 AIUI 开放平台重新配置方可使用新的语义。

已开通 AIUI 功能的 APPID，请通过 AIUIAgent 使用语义理解。

5.1.1 文本语义理解示例

文本语义，使用 TextUnderstander 类的函数，传入文本后，通过 TextUnderstanderListener 获取服务器返回的结构化文本结果，或错误信息。

```
mTextUnderstander.understandText( "科大讯飞", searchListener );
```

5.1.2 语音语义理解示例

语音语义，通过 SpeechUnderstander 类的函数，开启会话后，通过麦克风或 writeAudio 函数，录入音频数据，并通过 SpeechUnderstanderListener 获取服务器返回的结构化文本结果，或错误信息。

```
understander.startUnderstanding( mUnderstanderListener );
```

返回的语义结果，参考语义结果说明文档。

5.2 使用 AIUIAgent

AIUIAgent 仅对已开通 AIUI 功能的 APPID 可下载使用，未开通的应用，可通过 SpeechUnderstander 使用语义理解。

创建 AIUIAgent 实例并开启服务，AIUIAgent 需要使用 VAD 资源，请把 res/vad 目录下的资源文件，拷到由 AIUI 参数中 vad 的 res_path 参数（参考附录说明）指定的工程 assets/res/vad 目录下。

```
mAIUIAgent = AIUIAgent.createAgent( this, aiuiParams, mAIUIListener );
AIUIMessage startMsg = new AIUIMessage(AIUIConstant.CMD_START
    , 0
    , 0
    , null
    , null );
mAIUIAgent.sendMessage( startMsg );
```

aiuiParams ，AIUIConsatnt.CMD_START 分别为 AIUI 参数和消息值，参考附录的相应说明。服务开启后，服务的状态和结果消息将通过 AIUIListener 返回，具体事件说明请参考附录。关于 AIUIAgent 类的更多说明，参考**《AIUI Service Kit Manual.html》**。

5.2.1 文本语义理解示例

通过 sendMessage 发送要进行语义理解的文本数据，并通过 AIUIListener 的回调，获取结果或错误信息。

```
String params = "data_type=text";
byte[] textData = "今天合肥的天气".getBytes();
AIUIMessage msg = new AIUIMessage(AIUIConstant.CMD_WRITE
        , 0
        , 0
        , params
        , textData);
mAIUIAgent.sendMessage(msg);
```

5.2.2 语音语义理解示例

通过 sendMessage 发送消息，设置 AIUI 服务为已唤醒状态，再发送开始录音消息，通过麦克风录入音频，并通过 AIUIListener 的回调，获取结果或错误信息。

```
// 先发送唤醒消息，改变 AIUI 内部状态，只有唤醒状态才能接收语音输入
if( AIUIConstant.STATE_WORKING != this.mAIUIState ){
  AIUIMessage wakeupMsg = new AIUIMessage(AIUIConstant.CMD_WAKEUP
    , 0
    , 0
```

```
    , ""
    , null);
  mAIUIAgent.sendMessage(wakeupMsg);
}
// 打开 AIUI 内部录音机，开始录音
String params = "sample_rate=16000,data_type=audio";
AIUIMessage writeMsg = new AIUIMessage( AIUIConstant.CMD_START_RECORD
    , 0
    , 0
    , params
    , null );
mAIUIAgent.sendMessage(writeMsg);
```

返回的语义结果，参考语义结果说明文档。

5.2.3 更新词典

AIUIAgent 提供了上传词典的功能。如在线听写词典章节所述，更新词典只在对应的 LANGUAGE，ACCENT 或 “ent”下生效，而 AIUIAgent 在进行语音语义理解时使用的默认参数值，可能与听写的不一样，因此，直接通过 AIUIAgent 更新词典，可以保证其在使用 AIUIAgent 进行语音语义会话时生效。

```
AIUIMessage msg = new AIUIMessage(AIUIConstant.CMD_UPLOAD_LEXICON
    , 0
    , 0
    , params
    , null);
mAIUIAgent.sendMessage(msg);
```

其中，params 为含词典内容 JSON 数据，参考附录对 CMD_UPLOAD_LEXICON 消息的说明。上传成功与否的状态，通过 AIUIListener 的事件获取。

6 语音评测

语音评测（**SpeechEvaluator**）通过智能语音技术自动对发音水平进行评价、发音错误、缺陷进行定位和问题分析。目前评音评测提供汉语、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。

评测的参数主要有

- 语言（LANGUAGE）
- 题型（ISE_CATEGORY）
- 结果等级（RESULT_LEVEL）
- 试题内容

```
// 设置评测语种
mSpeechEvaluator.setParameter(SpeechConstant.LANGUAGE, language);

// 设置评测题型
mSpeechEvaluator.setParameter(SpeechConstant.ISE_CATEGORY, category);

// 设置结果等级，不同等级对应不同的详细程度
mSpeechEvaluator.setParameter(SpeechConstant.RESULT_LEVEL, resultLevel);

// evaText 为试题内容
mSpeechEvaluator.startEvaluating(evaText, null, mEvaluatorListener);
```

可通过 setParameter 设置的评测相关参数说明如下：

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	评测类型，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

与评测相关的错误码如下：

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

评测试题和结果格式及字段含义详见《Speech Evaluation API Documents》文档。

7 语音唤醒

语音唤醒（**VoiceWakeuper**）通过辨别输入的音频中特定的词语（如“讯飞语点”），返回被命中（唤醒）结果，应用通过回调的结果，进行下一步的处理，如点亮屏幕，或与用户进行语音交互等。唤醒资源中含有一个或多个资源，只要命中其中一个，即可唤醒。需下载使用对应的语音唤醒 SDK。

唤醒的参数主要有：

- 唤醒资源路径（IVW_RES_PATH）
- 唤醒类型（IVW_SST）
- 唤醒门限（IVW_THRESHOLD）
- 是否持续唤醒（KEEP_ALIVE）

```
// 唤醒资源路径，需下载使用对应的语音唤醒 SDK
mIvw.setParameter( SpeechConstant.IVW_RES_PATH, ivwResPath );

// 唤醒类型
mIvw.setParameter( SpeechConstant.IVW_SST, ivwSst );

// 唤醒门限
mIvw.setParameter( SpeechConstant.IVW_THRESHOLD, threshold );

// 持续唤醒
mIvw.setParameter( SpeechConstant.KEEP_ALIVE, keepAlive );

ret = mIvw.startListening( listener );
```

唤醒状态（结果和错误）通过 listener 的回调获取。

7.1 唤醒识别

唤醒类型中，有一种类型叫“唤醒识别”（oneshot），是在说唤醒词后，马上说识别命令，SDK 则在唤醒的同时，对命令进行识别，如“讯飞语点，打电话给张三”，其中，“讯飞语点”是唤醒词，“打电话给张三”是命令（语法识别中的某条规则，关于语法识别可以参考对应的章节）。从以上特点可以知道，在唤醒识别时，还需要传入在线语法 ID，或本地语法路径。

```
// 设置业务类型为唤醒识别
mIvw.setParameter( SpeechConstant.IVW_SST,"oneshot" );

//设置识别引擎，只影响唤醒后的识别（唤醒本身只有离线类型）
```

```
mIvw.setParameter( SpeechConstant.ENGINE_TYPE, asrEngineType );

if( SpeechConstant.TYPE_CLOUD.equals(asrEngineType) ){
    //设置在线识别的语法 ID
    mIvw.setParameter( SpeechConstant.CLOUD_GRAMMAR, grammarID );
}else{
    // 设置本地识别资源
    mIvw.setParameter( ResourceUtil.ASR_RES_PATH, asrResPath );

    // 设置语法构建路径
    mIvw.setParameter( ResourceUtil.GRM_BUILD_PATH, grmPath );
}

ret = mIvw.startListening( listener );
```

唤醒识别时，唤醒的状态获取不变，而识别的结果则通过回调中的事件获取。

7.2 闭环优化

闭环优化是针对开发者的唤醒资源由云端优化系统不断优化的功能。通过开发者 APP 使用场景，本地唤醒 SDK 自动挑选音频数据上传至云端，进行训练生成优化唤醒资源。开发者 APP 使用场景中，优化唤醒资源在相比原有资源在提升唤醒率及抑制误唤醒方面有良好的表现。持续优化包含两种网络模式：

- 模式 0：关闭优化功能，禁止向服务端发送本地挑选数据及启动唤醒时进行查询下载；
- 模式 1：开启优化功能，允许向服务端发送本地挑选数据。需要开发者自行进行优化资源的查询下载，及对资源的使用进行管理；

```
// 设置开启优化功能
mIvw.setParameter( SpeechConstant.IVW_NET_MODE, "1" );
```

可以每隔一段时间后查询是否有资源更新，如每周，每月，当有资源更新时，下载新的资源以替换当前使用的资源.

```
// 查询资源
mIvw.queryResource( ivwResPath, requestListener );

private RequestListener requestListener = new RequestListener() {
    ... // 其他回调函数

    @Override
    public void onBufferReceived(byte[] buffer) {
        try {
            String resultInfo = new String(buffer, "utf-8");
```

```
            JSONTokener tokener = new JSONTokener(resultInfo);
            JSONObject object = new JSONObject(tokener);

            int ret = object.getInt("ret");
            if(ret == 0) {
                String uri = object.getString("dlurl");
                String md5 = object.getString("md5");

                //下载资源，下载状态通过 wakeListener 获取
                mIvw.downloadResource( downloadUri, filePath, downloadMd5, wa
keListener );
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
};
```

7.3 AIMIC 唤醒

针对智能硬件，MSC SDK 提供了可处理多声道音频，获取音频中有效了几路音频，同时进行唤醒的模式，我们称之为 AIMIC 唤醒。在这种模式下，需要使用额外的音频处理库 libaimic.so，结合智能硬件的加密模块，对多声道的原始音频处理，并在唤醒结果中，包含当前唤醒的角度等信息。

目前 AIMIC 唤醒为定制方案，可通过文章最后的联系方式与我们商务同事联系获取 SDK。

在 AIMIC 唤醒中，包含以下与普通唤醒不同的参数：

- 唤醒路数（IVW_CHANNEL_NUM）
- 音频源（AUIOD_SOURCE）
- ALSA 录音卡号（IVW_ALSA_CARD）
- ALSA 录音采样率(IVW_ALSA_RATE)

```
// 如果设置 AUDIO_SOURCE 为 -3,则要求项目中包含 alsa 的 SDK
// 同时需要设置与麦克风硬件一致的 录音卡号和采样率 ，默认
// 卡号=2，采样率=16000。更多说明，参考《MSC Reference Manual.html》相关参数。
//mIvw.setParameter( SpeechConstant.AUDIO_SOURCE, "-3" );
//mIvw.setParameter( SpeechConstant.IVW_ALSA_CARD, X ); //X 为硬件实际可用的卡
号
```

```
//mIvw.setParameter( SpeechConstant.IVW_ALSA_RATE, Y ); //Y 为硬件实际可用的采样率


// 设置唤醒路数
mIvw.setParameter(SpeechConstant.IVW_CHANNEL_NUM, "3");
```

AIMIC 唤醒成功后，除在 WakeuperListener 的 onResult 函数返回结果后，还通过 onEvent 返回单声道 16K 可用于语音识别音频。

```
public void onEvent(int eventType, int arg1, int arg2, Bundle obj) {
   if( SpeechEvent.EVENT_RECORD_DATA==eventType ){
    //获取用于识别的音频
   byte[] data = obj.getByteArray(SpeechEvent.KEY_EVENT_RECORD_DATA);
  }
}
```

8 声纹密码

声纹与指纹一样，也是一种独一无二的生理特征，可以用来鉴别人的身份。MSC SDK 声纹密码（**SpeakerVerifer**）的使用包括注册（训练）、验证和模型操作。类似于一个网站的用户登录一样，用户必须先注册，才能登录（验证），在用户忘记密码时，可以提供重设密码的操作（模型操作）。

声纹使用过程的主要参数有：

- 用户 ID（AUTH_ID）
- 业务类型（ISV_SST）
- 密码类型（ISV_PWDT）
- 密码（ISV_PWD）
- 模型操作命令（ISV_CMD）
- 注册次数（ISV_RGN）

不同 APPID 的用户 ID 相互独立，即不同的 APPID 可以用相同的用户 ID——他们注册的模型也相互独立。如果需要两个不同的 APPID 的用户 ID 共用，可通过文章末尾的技术支持联系方式与我们联系。

8.1 声纹注册

目前 MSC SDK 支持两种类型的声纹密码：数字密码和文本密码。文本密码的效果在优化中，建议使用数字密码。密码类型的取值说明如下表所示：

取值	说明
1	文本密码。用户通过读出指定的文本内容来进行声纹注册和验证，现阶段支持的文本有“芝麻开门”。
3	数字密码。从云端拉取若干组特定的数字串（默认有 5 组，每组 8 位数字），用户依次读出这 5 组数字进行注册，在验证过程中会生成一串特定的数字，用户通过读出这串数字进行验证。

注册时使用的密码通过调用 getPasswordList 方法获取：

```
mVerify.setParameter( SpeechConstant.ISV_PWDT, pwdType );

if( pwdType.equals("3") ){
    //在数字数码时，服务器将根据设置的要训练的次数，返回对应有多少组的数字
    mVerify.setParameter( SpeechConstant.ISV_RGN, rgn );
}

mVerify.getPasswordList( mPwdListener );
```

通过 mPwdListener 回调中获取的密码，进行声纹注册。

```
// 设置业务类型为训练
mVerify.setParameter(SpeechConstant.ISV_SST, "train");
mVerify.setParameter( SpeechConstant.ISV_RGN, rgn );
mVerify.setParameter(SpeechConstant.ISV_PWDT, pwdType);
mVerify.setParameter(SpeechConstant.ISV_PWD, pwdText);
mVerify.setParameter(SpeechConstant.AUTH_ID, auth_id);
mVerify.startListening(mRegisterListener);
```

应用通过 mRegisterListener 的 onResult 方法来处理注册结果。在结果 result 中携带了一个返回码（0 表示成功，-1 为失败）和错误码，用来判别注册是否成功以及出错原因，错误码的含义如下：

错误码	错误值	说明
MSS_ERROR_IVP_GENERAL	11600	正常，请继续传音频
MSS_ERROR_IVP_EXTRA_RGN_SOPPORT	11601	rgn 超过最大支持次数 9
MSS_ERROR_IVP_TRUNCATED	11602	音频波形幅度太大，超出系统范围，发生截幅
MSS_ERROR_IVP_MUCH_NOISE	11603	太多噪音
MSS_ERROR_IVP_TOO_LOW	11604	声音太小
MSS_ERROR_IVP_ZERO_AUDIO	11605	没检测到音频
MSS_ERROR_IVP_UTTER_TOO_SHORT	11606	音频太短
MSS_ERROR_IVP_TEXT_NOT_MATCH	11607	音频内容与给定文本不一致
MSS_ERROR_IVP_NO_ENOUGH_AUDIO	11608	音频长达不到自由说的要求

结果中包含的字段以及各字段的含义见附录。

8.2 声纹验证

声纹验证过程与声纹注册类似，不同之处仅在于 ISV_SST 需要设置为“verify”，且不用设置 ISV_RGN 参数，并且，在数字密码类型时，密码通过 generatePassword 生成——实际上，验证密码的数字组合只要满足要求即可。其他参数的设置、验证结果的处理过程完全可参考上一节。

另外，为了达到较好的效果，请在声纹注册与验证过程中尽量与麦克风保持同样的距离（建议的最佳距离是 15 厘米左右）。如果距离差距较大的话，可能会对验证通过率产生较大影响。

8.3 模型操作

声纹注册成功后，在语音云端上会生成一个对应的模型来存储声纹信息，声纹模型的操作即对模型进行查询和删除。

```
// 首先设置声纹密码类型
mVerify.setParameter(SpeechConstant.ISV_PWDT, pwdType);

if( pwdType.equals("3") ){
    // 对于文本密码，必须设置声纹注册时用的密码文本
    mVerify.setParameter( SpeechConstant.ISV_PWD, pwdText );
}

// 调用 sendRequest 方法查询或者删除模型，cmd 的取值为“que”或“del”，表示查询或者删除
mVerify.sendRequest(cmd, auth_id, listener);
```

8.4 安全性问题

从上文的 3 个小节可以看到，对应用来说，操作声纹都是很简单直接的。因此，有些安全性也是应用应该考虑的：

- 更新模型：

应用在给用户更新声纹模型等操作时，为了安全性，应该考虑必要的验证，如，必须先通过声纹密码或应用登录密码验证，才能进行下一步更新声纹模型（重新注册）——类似修改密码时，必须先进行安全性验证一样。

- APPID：

从前面的特征可以看到，如果 APPID 与 libmsc.so 泄漏后，使用这个 APPID 和 libmsc.so 就可以绕过应用层的安全验证，操作该 APPID 的所有用户 ID 的模型。所以，应用的 APPID 须保证不泄漏给他人——如不要以明文方式在代码中以字符串保存；在论坛发帖求助时，不要在所有浏览者都可见的正文中带 APPID 内容等。

- 用户 ID：

SDK 对通过验证 APPID 的应用，只要指定用户 ID 就可以进行更新模型操作（先删除，再注册，或注册时指定替换）。所以，应用应当考虑用户可能会恶意指定他人的用户 ID，进行注册或验证的情况。应用可以在展示给用户的账号和用来注册 MSC SDK 声纹的用户 ID 间，做一层映射，**使用户无法直接看到或猜测到**实际用于 MSC SDK 声纹时的用户 ID 值；同时，应用在用户操作声纹密码的页面中，应尽可能的把账号设置为只读——避免恶意修改他人账号声纹密码的情况。

- 验证密码：

因为人的声音是可以录制下来的，当验证的密码固定不变时，用户的声音就比较大可能被别人录制下来，然后再次用来验证——在密码相同时。所以，用来验证的密码时，每次都不一样时，安全性可以更大的提搞——随机数字密码，比固定文本密码更安全。同时，也应该提醒用户，在注意声音的泄漏或被盗用——如留意不要随意输入银行卡密码一样。

通过上传一段音频，鉴别是哪个用户 ID 的 1:N 鉴别功能，参考身份验证章节。

9 人脸识别

人脸和声纹一样有着独特的特征，可以用来鉴别人的身份。 MSC SDK 中人脸识别（**FaceRequest**）功能，主要分为下面几类型：

- 鉴别身份（注册、验证和模型管理）
- 人脸检测
- 人脸聚焦

名称	说明
reg/注册	上传包含一张人脸的图片到云端，引擎对其进行特征抽取，生成一个与之对应的模型，返回模型 id(gid)。
verify/验证	注册成功后，上传包含一张人脸的图片到云端，引擎将其与所注册的人脸模型进行比对，验证是否为同一个人，返回验证结果。
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）。
auth_id/用户 id	由应用传入，用于标识用户身份，长度为 6-18 个字符（由英文字母、数字、下划线组成，不能以数字开头），不支持中文字符。 注：注册和验证都必须指定 auth_id。

为了获得较高的准确率，请确保输入的图片满足以下要求：

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。

在人脸识别过程的主要参数有：

- 人脸识别业务类型（WFR_SST）
- 用户 ID（AUTH_ID）

其中人脸检测和聚焦是对任意人脸图片检测，与用户是否已注册无关，故不需要设置用户 ID。

识别结果通过 RequestListener 回调返回，其中包含了是否成功等信息，详细的 JSON 格式请参照附录。

9.1 人脸注册

```
// 设置业务类型为注册
face.setParameter( SpeechConstant.WFR_SST, "reg" );
// 设置 auth_id
face.setParameter( SpeechConstant. AUTH_ID, mAuthId );
// imgData 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 );
```

9.2 人脸验证

```
// 设置业务类型为验证
face.setParameter( SpeechConstant.WFR_SST, "verify" );
// 设置 auth_id
face.setParameter( SpeechConstant. AUTH_ID, mAuthId );
// imgData 为图片的二进制数据
face.sendRequest( imgData, mRequestListener );
```

9.3 模型管理

目前 FaceRequest 暂未提供单独管理人脸模型的方法和参数，应用要给一个 AUTH_ID 重设模型时，可以参考 AUTH_ID 的参数说明：在注册时，提定参数“property”的 值为"del"，以覆盖已有的模型。

9.4 安全性问题

在人脸识别中，最重要的问题，可能是图片的可复制重复使用的问题。根据前文可知道，MSC SDK 在人脸识别时，并不是直接使用摄像头等硬件，只要应用传入二进制的图片数据即可。这时，在验证时，即使用一张他人照片，MSC SDK 也是无法分辨的；同时，应用即使是使用摄像头来拍摄实时照片，也有可能被用户使用照片来欺骗。所以，在这样的情况下，人为的管理则显得尤其重要：如要求用户验证时，有管理员在场保证其不使用他人照片（在公司打卡、门禁的场景）等。

此外，模型更新，APPID 和 用户 ID 要注意的问题，可参考声纹密码章节中的安全性问题说明，此处不再重复。

把人脸识别结合声纹密码，可以提供更高的安全性，参考融合验证章节。

9.5 人脸检测

```
// 设置业务类型为检测
face.setParameter(SpeechConstant.WFR_SST, "detect");
```

```
// imgData 为图片的二进制数据
face.sendRequest(imgData, mRequestListener);
```

9.6 人脸聚焦

```
//设置业务类型为聚焦
face.setParameter(SpeechConstant.WFR_SST, "align");
// imgData 为图片的二进制数据
face.sendRequest(imgData, mRequestListener);
```

针对人脸检测和人脸聚焦，MSC SDK 还提供了离线检测，请参考离线人脸检测章节内容。通过上传一张图片，鉴别是哪个用户 ID 的 1:N 鉴别功能，参考身份验证章节。

10 身份验证

身份验证（**IdentityVerifier**），又称为多生物特征融合验证（Multi-biometrics Fusion Verification，简称 MFV），支持人脸和声纹的 1:1 单一验证，1:1 融合验证，1:N 鉴别。需下载使用对应的身份验证 SDK。

MFV 目前提供的功能组合：

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）。

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	√	√		

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		√	

在 1:1 的首次验证前，需要先指定用户 ID 进行注册。

组鉴别前，需要先创建组，然后把用户 ID 加入到组中；在组鉴别时，指定组 ID，上传数据（音频或图片）后，将返回组内最匹配的几个用户 ID 及相似度。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，每个组 100 个成员。

注册或验证声纹或人脸时，须指定以下参数：

- 业务类型（MFV_SST）
- 业务场景（MFV_SCENES）
- 用户 ID（AUTH_ID）
- 验证模式（MFV_VCM）

10.1 注册

在声纹注册时，需要先获取密码：

```
// 设置业务场景：声纹（ivp）
mIdVerifier.setParameter(SpeechConstant.MFV_SCENES, "ivp" );

// 设置声纹密码组数
mIdVerifier.setParameter( "rgn", rgn );

// 执行密码下载操作，密码类型：数字密码（pwdt=3）
mIdVerifier.execute("ivp", "download", "pwdt=3", listener );
```

密码结果通过 listener 中的回调获取。

进行注册：

```
// 设置业务场景：人脸（ifr）或声纹（ivp）
mIdVerifier.setParameter( SpeechConstant.MFV_SCENES, scence );

// 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, authID );

// 设置监听器，开始会话
mIdVerifier.startWorking( listener );

// 在注册声纹时，需要多次写入音频数据
while( !isDataFinished ){
    if( scence.equals("ivp") ){
        // 在注册声纹时，需要在 params 中，指定声纹注册的参数，包括注册次数（rg
n），密码类型（pwdt），以及密码（ptxt）
        params = "pwdt=3,ptxt="+pwdTxt+",rgn="+rgn;
    }
    mIdVerifier.writeData( scence, params, data, offset, 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 listener 中的回调接口，返回结果或者错误信息。
mIdVerifier.stopWrite( scence );
```

注册人脸时，只要上传一张图片即可；而声纹注册时，则可能需要多次写入音频数据。注册结果状态通过 listener 的回调获取。

10.2 验证

验证时，可通过 MFV_VCM 指定验证模式为单一（人脸或声纹），还是混合（人脸+声纹）。在混合模式下，需要分别写入图片和音频数据。

```
// 设置业务场景
mIdVerifier.setParameter( SpeechConstant.MFV_SCENES, scene );

// 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( listener );

// 指定组 id，最相似结果数
String params = "group_id="+groudID

while( !isDataFinished ){
   // 写入数据
    mIdVerifier.writeData( scence, params, data, offset, length );
}

mIdVerifier.stopWrite( scence );
```

10.3 鉴别

鉴别与验证的过程相似，不过鉴别需要设置组 ID，以指定要鉴别的组。

```
// 设置业务场景
mIdVerifier.setParameter( SpeechConstant.MFV_SCENES, scene );

// 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( listener );

// 指定组 id，最相似结果数
String params = "group_id="+groudID

while( !isDataFinished ){
   // 写入数据
    mIdVerifier.writeData( scence, params, data, offset, length );
}

mIdVerifier.stopWrite( scence );
```

鉴别结果通过 listener 的回调返回，应用可通过返结果中，排在最前面，且相似度超过一定值的，认为数据属于此用户 ID。

10.4 模型和组管理

声纹模型目前支持的操作有查询（query）、删除（delete）、密码下载（download）3 种。人脸模型目前支持删除（delete）操作。组管理包括创建，查询，加入和删除。

```
// 设置业务场景，ifd，ivp，ipt
mIdVerifier.setParameter( SpeechConstant.MFV_SCENES, scene );

// 设置用户 id
mIdVerifier.setParameter( SpeechConstant.AUTH_ID, authID );

// params 根据不同的操作而不同
if( isCreateGroup ){
    //创建组
    params="scope=group,group_name=" + groupName;
    cmd = "add";
}else if( isDeleteGroup ){
    //删除组
    params = "scope=group,group_id=" + groupID );
    cmd = "delete";
}else if( isAddGroupMember ){
    //加入组员
    params = "scope=person,group_id=" + groupID+",auth_id=" + authID;
    cmd = "add";
}else if( isDeleteGroupMember ){
    //删除组员
    params = "scope=person,group_id=" + groupID+",auth_id=" + authID;
    cmd = "delete";
}else if( isQueryGroupMember ){
```

```
    //查询组员
    params = "scope=group,group_id=" + groupID );
    cmd = "query";
}else{
    // 人脸或声纹的模型管理 query, delete, download
    cmd = ...;
}

// cmd 为 操作，模型管理包括 query, delete, download，组管理包括 add，query，de
lete
mIdVerifier.execute( scene, cmd , params, listener );
```

操作结果通过 listener 回调获取。

10.5 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，参数下表所示。

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。

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。

10.6 安全性问题

从前面模型和组管理的章节可以看到，在身份验证时，除在注意在声纹密码和人脸识别章节提到的安全性问题外（参考对应的章节内容），还需要**注意对组 ID 的保密**。在组管理中，通过指定组 ID，即可查询到所有组员的 AUTH_ID，当他人同时也获取到 libmsc.so（通过 APK 解压等）以及 APPID 时，通过 group_id 查到所有组员 auth_id，便可直接修改对应 auth_id 的模型。应用可以对展示给用户的组 ID 进行映射，例如可以让展示界面显示的，只是一个组的昵称等。

从组 ID 需要保密可以知道，为了安全性，组管理并未提供通过 APPID 查询其所有组 ID 的功能——因为这样获取到 libmsc.so 和 APPID 的人就会获取到所有组 ID，从而获取到所有组员的 AUTH_ID，继而修改其模型。所以，应用在创建组后，需要保存组 ID，以在组管理和鉴别时使用。

此外，模型更新，APPID 和 用户 ID 要注意的问题，可参考声纹密码章节中的安全性问题说明，此处不再重复。

11 离线人脸检测

离线人脸检测（**FaceDetector**）包含离线图片检测和离线视频流检测。离线人脸功能具有较强的实时性，效果比在线稍低，对于某些场景（如侧脸、偏头等），可能出现在线接口检测到人脸，而离线接口检测不到的情况。需下载使用对应的人脸识别 SDK。

11.1 图片检测

为了使图片检测获得较高的准确率，请保证检测的图片满足以下要求：

1. 请在使用时传入 PNG、JPG 和 BMP 格式的图片（离线人脸检测只支持 PNG、JPG 和 BMP 格式的图片）；
2. 离线图片检测最大支持 1500 万像素的图片，由于检测使用 Bitmap 类保存图片数据，应用需要根据图片大小做好压缩或者裁剪，防止出现内存溢出异常，具体请参考人脸示例；

```
// 创建一个 FaceDetector 单例对象，FaceDetector 集成了所有离线能力
FaceDetector faceDector = FaceDetector.createDetector(this, null);

// 离线人脸检测有两种方式一种是灰度图检测，一种是 ARGB 图检测。（注意：灰度图检测方法传入的必须为灰度图，SDK 提供 VerifierUtil 类可将 ARGB 图转换成灰度图）
//1、调用 detectARGB 方法实现 ARGB 图检测：
String result = faceDector.detectARGB(img);

//2、调用 VerifierUtil 工具将 ARGB 转换成灰度图，再调用 detectGray 方法实现：
Bitmap src = VerifierUtil.doARGB2Gray(img);

// 检测结果为 JSON 格式，具体格式见附录
String result = faceDector.detectGray(src);
```

11.2 视频流检测

FaceDetector 可以对摄像头视频流中的人脸进行实时检测。目前仅支持 640*480 的 nv21 格式视频数据。

视频流检测有四个关键步骤：

1. 创建离线视频流对象，请参考图片检测；
2. 摄像头参数设置；

```
Camera mCamera = Camera.open(mCameraId);

Parameters params = mCamera.getParameters();
```

```
// 设置视频流格式为 nv21，视频帧尺寸为 640*480
params.setPreviewFormat(ImageFormat.NV21);
params.setPreviewSize(640, 480);
mCamera.setParameters(params);
```

3. 获取设备朝向。设备朝向的定义如下图所示：

0度　　90度　　180度　　270度

```
// 在进行检测之前，需要创建并开启重力感应器对象，用于获取设备的朝向
Accelerometer mAcc = new Accelerometer(this);
mAcc.start();
// 获取设备朝向，返回值 0,1,2,3 分别表示 0,90,180 和 270 度
int direction = Accelerometer.getDirection();
// 判断当前使用的是否为前置摄像头
boolean frontCamera = (Camera.CameraInfo.CAMERA_FACING_FRONT == mCameraId);
// 由于前置摄像头预览显示的是镜像，需要将手机朝向换算成摄相头视角下的朝向
// 转换公式：a' = (360 - a)%360，a 为人眼视角下的朝向（单位：角度）
if (frontCamera) {
// SDK 中使用 0,1,2,3,4 分别表示 0,90,180,270 和 360 度
    direction = (4 - direction)%4;
}
```

4. 调用 FaceDetector 的 trackNV21 方法检测视频帧中的人脸：

```
// buffer 为一帧 640*480 的 nv21 视频（注意：SDK 暂时只支持 640*480 的 nv21 视频帧，传入其他格式（尺寸）的视频都不能正常工作），direction 为设备朝向
// 返回结果为 json 数据，参考附录
String result = faceDector.trackNV21(buffer, 640, 480, 0, direction);
```

结果说明请参考附录。

12 附录

附录包含了各功能的结果解析，合成的发音人列表，以及错误码等，供应用开发时参考。请展开各小节查看对应的内容。

12.1 识别结果

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"
                }
            ]
        }
    ]
}
```

12.2 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	属性	语言	参数值	新引擎参数	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	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		
晓琳	青年女声	中英文（台湾普通话）		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		
小倩	青年女声	汉语（东北话）		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		

12.3 AIUIAgent 参数和消息

12.3.1 AIUI 参数字段说明

参数类型		参数名称	
global	全局参数设置	scene	用户定制的场景参数，不同的场景可对应不同的云端处理流程。
		clean_dialog_history	清除语义对话历史方式，取值：auto（默认），user（用户自己控制）。设置为 auto 时，在首次唤醒时消除对话历史，设置为 user 则需要用户发送 CMD_CLEAN_DIALOG_HISTORY 命令清除历史。
interact	人机交互参数	interact_timeout	交互超时（单位：ms），即唤醒之后，如果在这段时间内无有效交互（无有效语义结果返回），则重新进入待唤醒状态，取值：[10000,180000)，默认为 1min。
		result_timeout	结果超时（单位：ms），即检测到语音后端点后一段时间内无结果返回则抛出超时错误，默认值：5000。
vad	音频端点检测参数	engine_type	vad 引擎类型，取值：meta（模型 vad），fixfront（能量 vad），默认值：meta。
		res_type	资源类型，取值：assets 资源（apk 工程的 assets 文件），res 资源（apk 工程的 res 文件），path 资源（sdcard 文件）。使用模型 vad 时必须设置。
		res_path	资源文件路径，使用模型 vad 时必须设置。
iat	语音识别参数	sample_rate	音频采样率（单位：Hz），取值：8000,16000，默认值：16000。
speech	语音业务相关参数	data_source	数据源，取值：user（由用户从外部写入），sdk（SDK 内部录音机采集）。

		interact_mode	交互模式，取值：oneshot（唤醒后一次交互后即休眠），continuous（默认，唤醒后可以持续交互）。

对于有默认值的参数，若参数中不存在对应的 key-value 设置，即表示取默认值。若某类型参数全部使用默认值，可以将该类型从配置文件中删除。

```
{
    /* 交互参数 */
    "interact":{
        "interact_timeout":"60000",
        "result_timeout":"5000"
    },

    /* 全局设置 */
    "global":{
        "scene":"main",
        "clean_dialog_history":"auto"
    },

    /* 业务相关参数 */
    // 本地 vad 参数
    "vad":{
        "vad_enable":"1",
        "engine_type":"meta",
        "res_type":"assets",
        "res_path":"vad/meta_vad_16k.jet"
    },

    // 识别（音频输入）参数
    "iat":{
        "sample_rate":"16000"
    },

    /* 业务流程相关参数 */
    // 语音业务流程控制
    "speech":{
        "data_source":"sdk"
    }
}
```

12.3.2 AIUIMessage 类型说明

msgType（消息类型）	取值	说明	返回
CMD_GET_STATE	1	获取服务状态。	有
CMD_WRITE	2	向 AIUI 服务写入数据。需要在 params 中指定数据类型、调用的业务等，例如：”data_type=audio,sample_rate=16000”，data 为待写入的二进制数据（如音频、图像、文本等）。	无
CMD_STOP_WRITE	3	停止写入数据。需要在 params 中指定要停止写入的数据类型，如“data_type=audio,sample_rate=16000”即停止写入音频。	无
CMD_RESET	4	重置 AIUI 服务的状态。服务会立即停止并重新启动，进入到待唤醒状态。	无
CMD_START	5	启动 AIUI 服务。当 AIUI 服务停止后，使用此命令启动服务。	无
CMD_STOP	6	停止 AIUI 服务。服务停止之后，将不响应外部输入。	无
CMD_WAKEUP	7	唤醒消息。用于手动唤醒 AIUI 服务，arg1 为唤醒后拾音的波束号，默认为 0。关于波束的定义参见《麦克风设计参考 V0.7》。	无
CMD_RESET_WAKEUP	8	重置唤醒状态。AIUI 服务重置为待唤醒状态。若当前为唤醒状态，发送该消息重置后会抛出 EVENT_SLEEP 事件。	无
CMD_SET_BEAM	9	设置麦克风阵列的拾音波束。用 arg1 携带拾音波束号。	无

CMD_SET_PARAMS	10	设置参数配置。用 params 携带参数设置 JSON 字符串，具体格式参照 aiui.cfg 文件。暂时只可以用来修改 scene 参数，实时生效。
		示例：
		{
		“global”:{
		“scene”:”nlp31”}
		}
		该示例将 scene 修改为 nlp31。修改其他参数方法也是如此，即发送一条 params 字段为 JSON 字符串（指明待修改的参数类型、名称和取值，格式参照 aiui.cfg）的 CMD_SET_PARAMS 消息。
			无

CMD_UPLOAD_LEXICON	11	上传用户词表。将用户词表按格式组成 JSON 字符串，放在 params 字段传入 SDK，具体格式： {“name”:”userword”, // 词表名称 “content”:”XXXX” // 词表内容 },其中 XXXX 也为一个 JSON 字符串，示例： {"userword":[{"name":" 我 的 常 用 词","words":["佳晨实业","蜀南庭苑","高兰路","复联二"]},{"name":"我的好友","words":["李馨琪","鹿晓雷","张集栋","周家莉","叶震珂","熊泽萌"]}]}	有
CMD_SEND_LOG	12	发送应用日志到云端，可以帮助分析应用问题。需要将 JSON 格式的字符串放在 params 字段中携带。	无

CMD_SYNC	13	同步操作。arg1 字段为待同步的数据类型，取值：SYNC_DATA_STATUS（状态同步，params 字段为状态 JSON 字符串），SYNC_DATA_INDIVIDUAL（个性化数据同步，如联系人），SYNC_DATA_ACCOUNT（第三方账号关系同步），SYNC_DATA_ATHENA_INDIVIDUAL（雅典娜个性化数据同步）。	有
CMD_START_SAVE	14	开始保存数据。params 字段为待保存的数据属性，如“data_type=raw_audio”则保存阵列原始音频。	无
CMD_BUILD_GRAMMAR	16	构建本地语法。用 params 字段携带 bnf 语法内容，bnf 语法规则请参见《BNF 语法开发指南》。	有
CMD_UPDATE_LOCAL_LEXICON	17	更新本地词表。当构建好本地语法之后，可以动态更新某个槽（声明为 slot，实质上为词表）的内容，以动态变更支持的说法。将需要更新的槽名称和内容组成 JSON 字符串，用 params 字段携带。示例： {“name”:”<contact>”, // 槽名称 “content”:”张三\n 李四\n” //词表内容 } 注：只有在刚刚成功构建语法（即发送了 CMD_BUILD_GRAMMAR 消息并返回成功事件）之后才能更新本地词表。	有

注：“有返回”的含义是在向 AIUI 发送一条 CMD 消息后，AIUI 会抛出一个对应的 EVENT 事件返回 CMD 消息的处理结果。

12.3.3 AIUIEvent 类型说明

eventType（事件类型）	取值	说明
EVENT_RESULT	1	结果事件。data 字段携带结果数据，info 字段为描述数据的 JSON 字符串。
EVENT_ERROR	2	出错事件。arg1 字段为错误码，info 字段为错误描述信息。
EVENT_STATE	3	服务状态事件。当向 AIUI 发送 CMD_GET_STATE 命令时抛出该事件，arg1 字段取值为 STATE_IDLE（空闲状态）、STATE_READY（就绪状态，待唤醒）、STATE_WORKING（工作状态，已唤醒）状态之一。
EVENT_WAKEUP	4	唤醒事件。info 字段为唤醒结果 JSON 字符串。
EVENT_SLEEP	5	休眠事件。当出现交互超时，服务会进入休眠状态（待唤醒），或者发送了 CMD_RESET_WAKEUP 时，抛出该事件。arg1 字段取值：TYPE_AUTO（自动休眠，即交互超时）、TYPE_COMPEL（外部强制休眠，即发送 CMD_RESET_WAKEUP）。
EVENT_VAD	6	VAD 事件。当检测到输入音频的前端点后，会抛出该事件，用 arg1 标识前后端点或者音量信息：0（前端点）、1（音量）、2（后端点）。当 arg1 取值为 1 时，arg2 为音量大小。
EVENT_CMD_RETURN	8	某条 CMD 命令对应的返回事件。对于除 CMD_GET_STATE 外的有返回的命令，都会返回该事件，用 arg1 标识对应的 CMD 命令，arg2 为返回值，0 表示成功，info 字段为描述信息。

12.4 声纹结果

文本密码 JSON 示例

```
{"txt_pwd":["我的地盘我做主","移动改变生活","芝麻开门"]}
```

数字密码 JSON 示例

```
{"num_pwd":["03285469","09734658","53894276","57392804","68294073"]}
```

声纹业务结果（VerifierResult）成员说明

成员	说明
sst	业务类型，取值为 train 或 verify
ret	返回值，0 为成功，-1 为失败
vid	注册成功的声纹模型 id
score	当前声纹相似度
suc	本次注册已成功的训练次数
rgn	本次注册需要的训练次数
trs	注册完成描述信息
err	注册/验证返回的错误码
dcs	描述信息

12.5 唤醒结果

唤醒结果（WakeuperResult）字段说明:

成员名	参数解释
sst	本次业务标识：wakeup 表示语音唤醒；oneshot 表示唤醒+识别；
id	当前唤醒词的 id
score	当前唤醒得分
bos	当前唤醒音频的前端点
eos	当前唤醒音频的尾端点
angle	当前唤醒角度（仅 AIMIC 唤醒出现）
beam	当前唤醒波束（仅 AIMIC 唤醒出现）
frame_count	当前唤醒音频帧数（仅 AIMIC 唤醒出现）
channel	当前唤醒的线程号（仅 AIMIC 唤醒出现）

唤醒资源查询结果字段说明：

成员名	参数解释
ret	本次请求是否成功：0 表示成功；非 0 表示失败，ret 为错误码；
resize	最新资源大小
dlurl	最新资源下载链接
md5	最新资源 md5 值，便于下载完成校验
sid	本地请求会话 id，便于问题查询

12.6 人脸识别结果

JSON 字段	类型	说明
sst	String	业务类型，取值为“reg”或“verify”
ret	int	返回值，0 为成功，-1 为失败
rst	String	注册/验证成功
gid	String	注册成功的人脸模型 id
score	double	人脸验证的得分（验证时返回）
sid	String	本次交互会话的 id
uid	String	返回的用户 id

注册结果示例：

```
{
    "ret":"0",
    "uid":"",
    "rst":"success",
    "gid":"wfr278b0092@hf9a6907805f269a2800",
    "sid":"wfr278b0092@hf9a6907805f269a2800",
    "sst":"reg"
}
```

验证结果示例：

```
{
    "ret":"0",
    "uid":"946529856",
    "score":99.999992,
    "gid":"a722f3e50e6d77df6b00a548557df8cb",
    "verf":true,
    "sst":"verify",
    "rst":"success",
    "sid":"wfr03004b55@ch3d550a541d0d476f00"
}
```

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"
}
```

12.7 身份验证结果

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"
}
```

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
}
```

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"
}
```

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

```
{
    "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
}
```

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
}
```

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 "
}
```

12.8 离线人脸检测结果

离线人脸仅提供人脸框检测及视频流检测功能，故返回结果为图片或者视频每帧人脸坐标信息。

JSON 字段	类型	说明
ret	int	返回值 0 为成功，其他值为错误吗
face	JsonArray	检测到人脸信息
postion	JsonObject	人脸框位置信息
landmark	JsonObject	视频流人脸关键点位置信息（可选）

人脸检测结果示例：

```
{
    "ret": 0,
    "face": [
        {
            "position": {
                "bottom": 244,
                "right": 244,
                "left": 94,
                "top": 94
            }
        }
    ]
}
```

视频流检测结果示例：

```
{
    "ret": 0,
    "face": [
        {
            "position": {
                "bottom": 341,
                "right": 364,
                "left": 97,
                "top": 74
            },
            "landmark": {
                "right_eye_right_corner": {
                    "y": 287,
```

```
            "x": 291
        },
        "left_eye_left_corner": {
            "y": 112,
            "x": 282
        },
        "right_eye_center": {
            "y": 258,
            "x": 293
        },
        "left_eyebrow_middle": {
            "y": 125,
            "x": 327
        },
        "right_eyebrow_left_corner": {
            "y": 221,
            "x": 324
        },
        "mouth_right_corner": {
            "y": 241,
            "x": 154
        },
        "mouth_left_corner": {
            "y": 149,
            "x": 150
        },
        "left_eyebrow_left_corner": {
            "y": 92,
            "x": 319
        },
        "right_eyebrow_middle": {
            "y": 265,
            "x": 335
        },
        "left_eye_center": {
            "y": 136,
            "x": 286
        },
        "nose_left": {
            "y": 158,
            "x": 217
        },
        "mouth_lower_lip_bottom": {
            "y": 194,
```

```
                    "x": 133
                },
                "nose_right": {
                    "y": 228,
                    "x": 219
                },
                "left_eyebrow_right_corner": {
                    "y": 163,
                    "x": 323
                },
                "right_eye_left_corner": {
                    "y": 233,
                    "x": 288
                },
                "nose_bottom": {
                    "y": 190,
                    "x": 208
                },
                "nose_top": {
                    "y": 185,
                    "x": 235
                },
                "mouth_middle": {
                    "y": 192,
                    "x": 156
                },
                "left_eye_right_corner": {
                    "y": 161,
                    "x": 285
                },
                "mouth_upper_lip_top": {
                    "y": 191,
                    "x": 176
                },
                "right_eyebrow_right_corner": {
                    "y": 305,
                    "x": 325
                }
            }
        }
    ]
}
```

12.9 错误码

10000~19999 的错误码参见 MSC 错误码链接。

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 常见问题

◆ 集成语音识别功能时，程序启动后没反应

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。

◆ SDK 是否支持本地语音能力

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，声纹功能也即将上线。

◆ Appid 的使用规范

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。

◆ 错误码 21001, 21002, 20021，创建单例返回 null

参考以下帖子：http://bbs.xfyun.cn/forum.php?mod=viewthread&tid=9688

◆ 更多问题解答，请见

技术支持论坛 http://bbs.xfyun.cn

◆ 其它联系方式

商务合作邮箱：msp_support@iflytek.com