科大讯飞股份有限公司
IFLYTEK CO.,LTD.

科大讯飞 MSC 新手指南（iOS）

目录

1. 概述

本文档是科大讯飞 iOS 语音 SDK 的使用指南，定义了语音听写、语音识别、语音合成、语义理解，语音唤醒，人脸识别等相关接口的使用说明。适用的读者为软件工程师和与技术相关的人员。阅读本文档，读者可以快速集成所需的服务。

1.1. iOS 语音 SDK 概述

如图 1-1-1 所示。

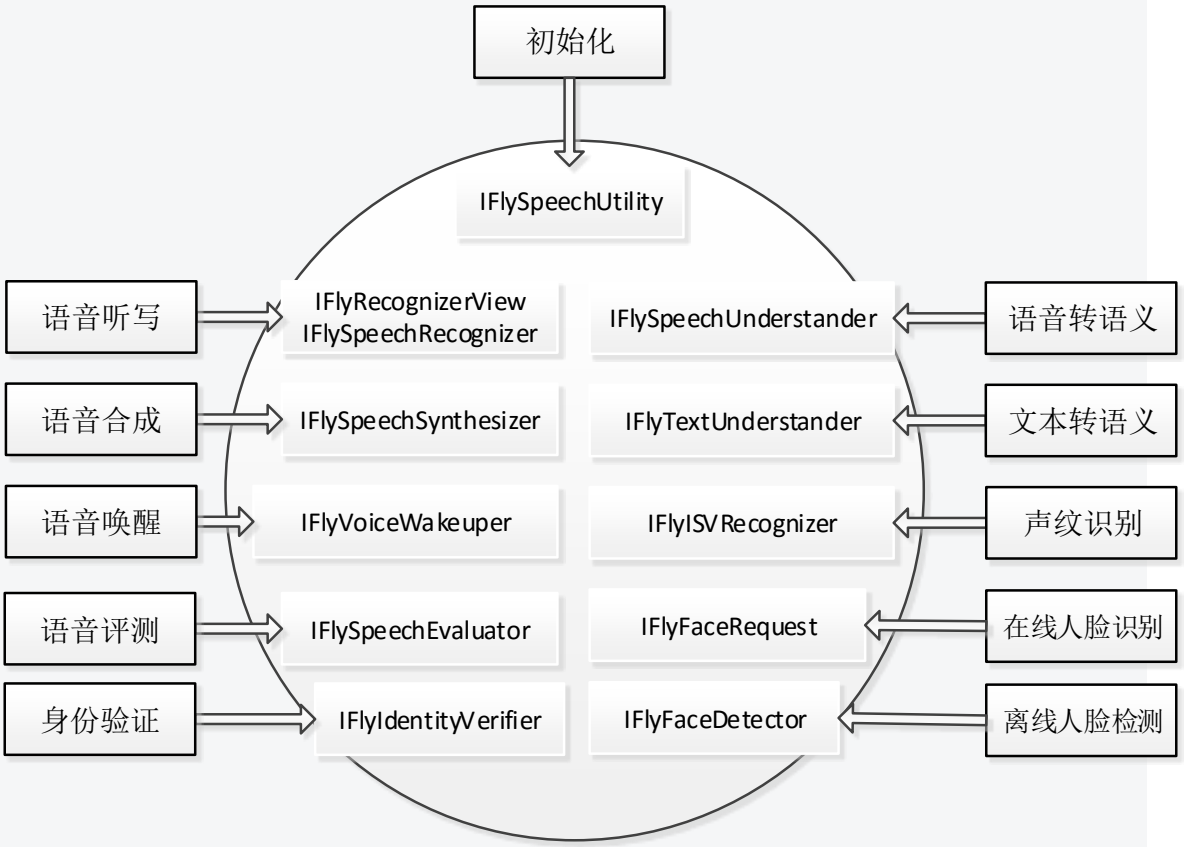


图 1-1-1 MSC 主要功能接口

1.2. 业务描述

为了更好地理解后续内容，这里对文档中出现的若干专有名词进行解释说明，更为详细的信息可以通过下面链接查看：
http://www.xfyun.cn/default/doccenter/doccenterInner?itemTitle=dHRz

名词	解释
语音合成	将一段文字转换成语音，可根据需要合成出不同音色、语速和语调的声音，让机器像人一样开口说话。
语音听写	将一段语音转换成文本，把语音中包含文字信息提取出来，并可以优先识别用户手机特有的联系人和个性化数据。
语法识别	判断用户所说的内容是否与预定义的语法相符合，主要用于识别用户是否下达某项指令，使用语法识别前，需要先定义语法。
语义理解	在语音听写基础上，分析理解用户的说话意图，返回结构化的指令信息。 开发者可在语义开放平台定义专属的问答格式。
语音评测	通过智能语音技术自动对发音水平进行评价，给出用户综合得分和发音信息。
声纹识别	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%。
语音唤醒	即设备（手机、玩具、家电等）在休眠（或锁屏）状态下也能检测到用户的声音，并根据声音提示进行相应操作，开启全语音交互，同时支持唤醒+识别、唤醒+语义的 OneShot 方案。 注：iOS 等智能移动平台下，启动唤醒需要关注录音权限。

表 1-1-1 名词解释

2. 快速集成流程

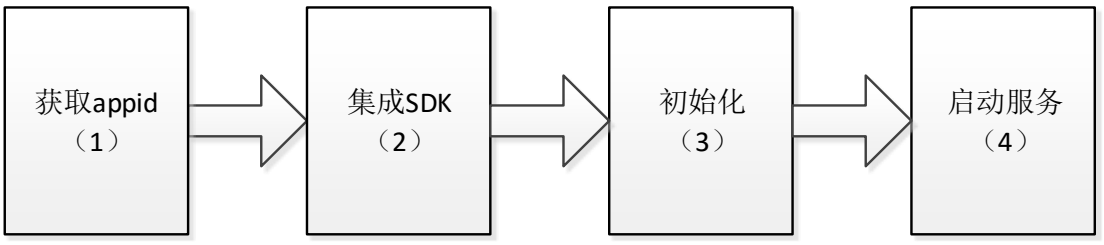


图 2-1 第三方应用快速集成流程

2.1. 第一步：获取 Appid

Appid 是第三方应用集成语音云开放平台 SDK 的身份标识，每款应用必须保持唯一，Appid 在开放平台申请应用时可以获得。下载 Demo 后一般可以在 Demo 中搜索 APPID_VALUE 得到。

注意：下载时应用的 appid 与 SDK 绑定。每个下载的 SDK 对应下载此 SDK 时使用应用的 appid。更换 appid 时需要同时更换 SDK，否则会报 10407 错误。

2.2. 第二步：集成 SDK

使用讯飞提供的语音等服务，需要集成讯飞 iOS MSC framework 以及 framework 依赖的系统库，并根据服务需要做适当的工程配置。

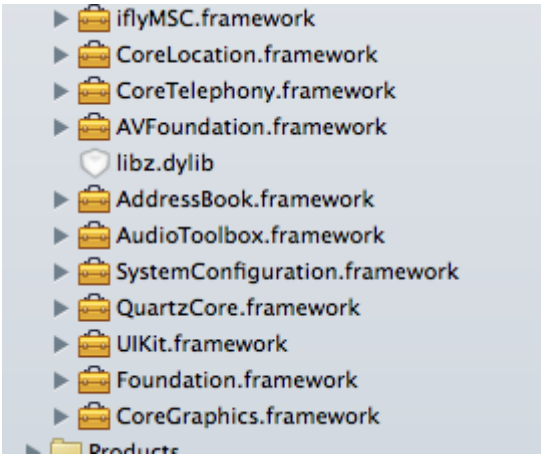
2.2.1. 集成讯飞库

讯飞 iOS framework 名称为 iflyMSC.framework，存放在下载 ZIP 包中的 lib 目录下。在工程中集成时，在工程 Build Phrases 页面的 link Binary With Libraries 项目中添加时选择 Add Other…项找到 iflyMSC.framework 的位置，选择添加。

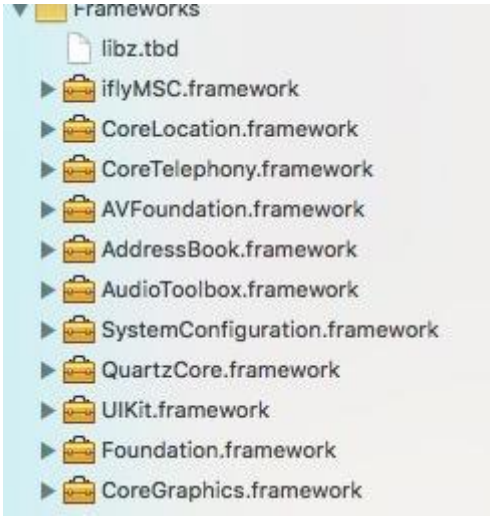
注意：iflyMSC.framework 非系统库文件，添加 iflyMSC.framework 时，请注意工程 BuildSetting 中的 framwork path 的设置，如果出现找不到 framework 中头文件的编译警告，可以将 path 清空，在 xcode 中删除 framework，然后重新添加。

2.2.2. 集成系统库

iflyMSC.framework 提供的服务依赖多项系统库，在使用时需要同时添加这些系统库。在工程中集成时，在工程 Build Phrases 页面的 link Binary With Libraries 项目中添加。



（Xcode 6 及以前版本）



（Xcode 7 及以前版本）

Xcode6 及以前版本	Xcode7 及以后版本
libz.dylib	libz.tbd
AVFoundation.framework	AVFoundation.framework
SystemConfiguration.framework	SystemConfiguration.framework
Foundation.framework	Foundation.framework
CoreTelephoney.framework	CoreTelephoney.framework

AudioToolbox.framework	AudioToolbox.framework
UIKit.framework	UIKit.framework
CoreLocation.framework	CoreLocation.framework
AddressBook.framework	AddressBook.framework
QuartzCore.framework	QuartzCore.framework
CoreGraphics.framework	CoreGraphics.framework

注意：如果使用了离线识别，还需要增加 libc++.dylib（Xcode7 下为 libc++.tbd）。

2.2.3. 工程配置

2.2.3.1. ATS 设置

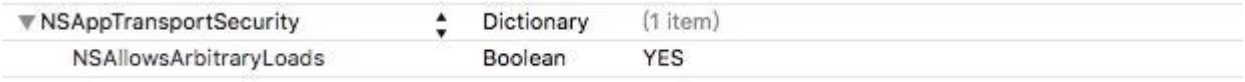
在 iOS 9 下直接进行 HTTP 请求时会收到如下错误提示：

App Transport Security has blocked a cleartext HTTP (http://) resource load since it is insecure.Temporary exceptions can be configured via your app‘s Info.plist file.

不能直接使用 HTTP 进行请求，需要在 Info.plist 新增一段用于控制 ATS 的配置：

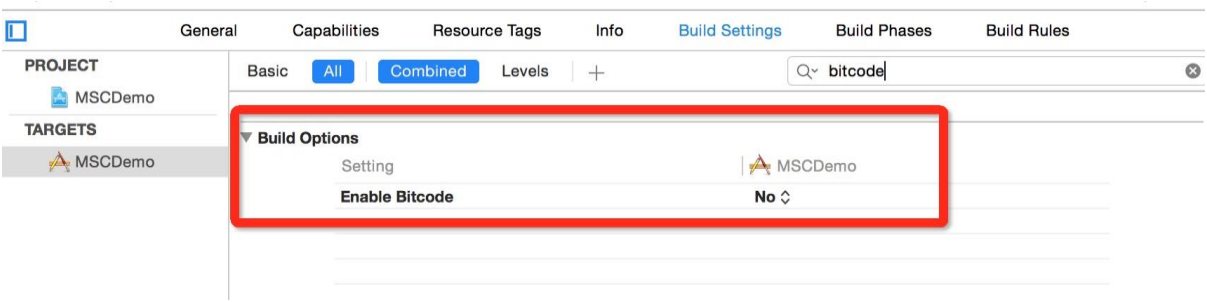
```
<key>NSAppTransportSecurity</key>
<dict>
<key>NSAllowsArbitraryLoads</key>
<true/>
</dict>
```

即：



2.2.3.2. BitCode 设置

在 Xcode 7 默认开启了 Bitcode，Bitcode 需要工程依赖的类库同时支持。而语记 SDK 暂时还不支持 Bitcode，所以可以先临时关闭。后续支持 Bitcode 请关注讯飞开放平台版本更新，QQ 群中也会提醒。在 Targets - Build Settings 中搜索 Bitcode 即可，找到相应选项，设置为 NO。



2.3. 第三步：初始化

```
Appid是应用的身份信息，具有唯一性，初始化时必须要传入Appid。
NSString *initString = [[NSString alloc] initWithFormat:@"appid=%@",@"YourAppid"];

[IFlySpeechUtility createUtility:initString];
```

注意：初始化是一个异步过程，可放在 App 启动时执行初始化，具体代码可以参照 Demo 的 MSCAppDelegate.m。未初始化时使用服务，一般会返回错误码 10111.

2.4. 第四步：启动服务

所有的服务皆遵循如下的流程，如图。

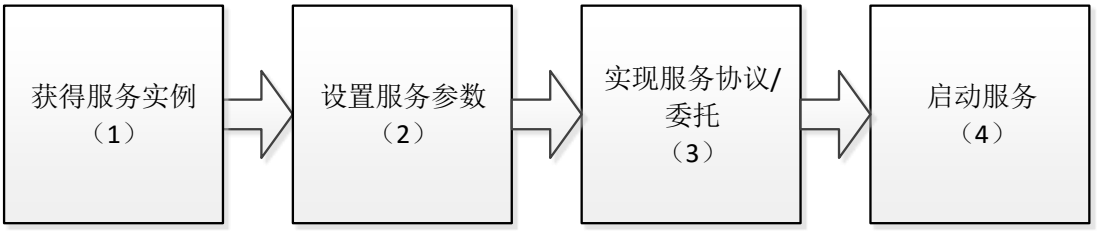


图 2-4-1 服务使用流程

注意：1、服务为单例模式，使用服务时从sharedInstance方法获取。2、由于服务为单例模式，不同的使用场景时注意重新设置服务参数。3、某些服务协议或委托可能相同，使用时可以使用不同委托对象实现委托和协议。4、启动服务前请确保[IFlySpeechUtility createUtility:initString];方法已被调用。

3. 语音合成

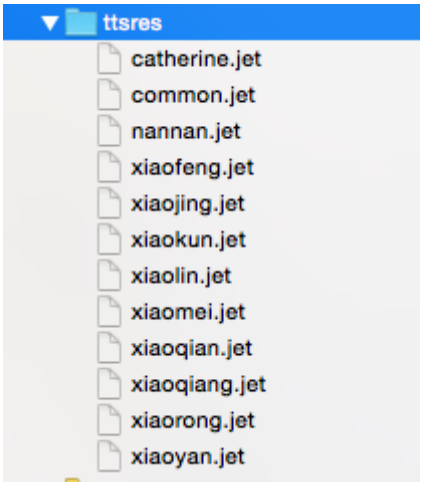
合成服务支持在线和离线两种工作方式，默认为在线。如果使用离线服务，有 2 种方式，一种是使用语记（原语音+）提供的免费服务，一种是付费购买后使用离线 SDK 在应用内部集成。相关细节请关注 http://www.xfyun.cn/。

3.1. 在线合成

本示例对应 Demo 的 TTSViewController 文件，为在线合成的代码示例。

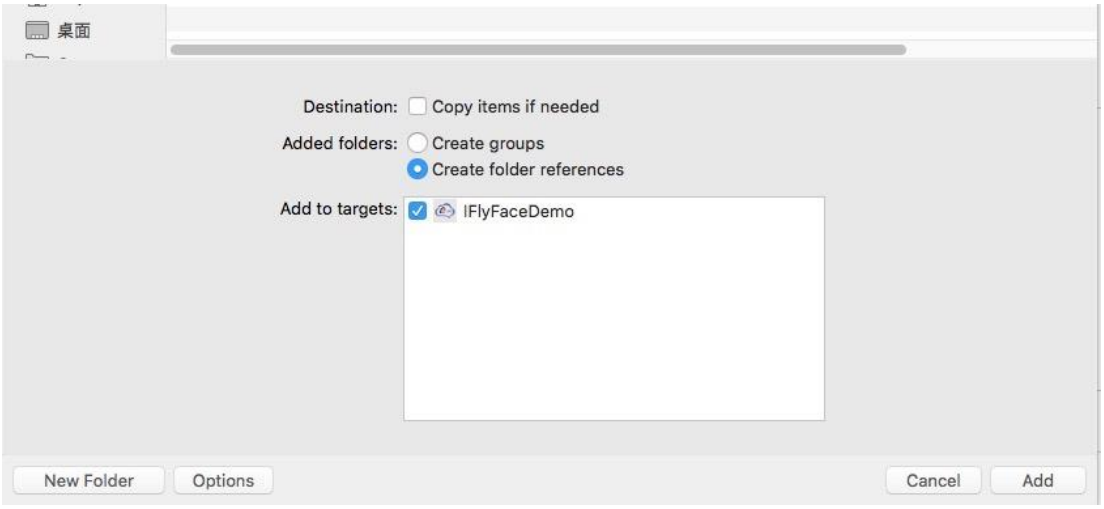
```
//包含头文件
#import "iflyMSC/IFlySpeechConstant.h"
#import "iflyMSC/IFlySpeechSynthesizer.h"
#import "iflyMSC/IFlySpeechSynthesizerDelegate.h"
//需要实现IFlySpeechSynthesizerDelegate合成会话的服务代理
@interface TTSViewController : UIViewController<IFlySpeechSynthesizerDelegate>
{
   IFlySpeechSynthesizer           * _iFlySpeechSynthesizer;
}
//1.创建合成对象
_iFlySpeechSynthesizer = [IFlySpeechSynthesizer sharedInstance]; _iFlySpeechSynthesizer.delegate =
self;
//2.设置合成参数
//设置在线工作方式
 [_iFlySpeechSynthesizer setParameter:[IFlySpeechConstant TYPE_CLOUD]
forKey:[IFlySpeechConstant ENGINE_TYPE]];
//音量，取值范围 0~100
[_iFlySpeechSynthesizer setParameter:@"50" forKey: [IFlySpeechConstant VOLUME]];
//发音人，默认为”xiaoyan”，可以设置的参数列表可参考“合成发音人列表”
[_iFlySpeechSynthesizer setParameter:@" xiaoyan " forKey: [IFlySpeechConstant VOICE_NAME]];
//保存合成文件名，如不再需要，设置设置为nil或者为空表示取消，默认目录位于
library/cache下
[_iFlySpeechSynthesizer setParameter:@" tts.pcm" forKey: [IFlySpeechConstant TTS_AUDIO_PATH]];
//3.启动合成会话
[_iFlySpeechSynthesizer startSpeaking: @"你好，我是科大讯飞的小燕"];
//4.IFlySpeechSynthesizerDelegate 实现代理
//结束代理
- (void) onCompleted:(IFlySpeechError *) error{}
//合成开始
- (void) onSpeakBegin{}
//合成缓冲进度
- (void) onBufferProgress:(int) progress message:(NSString *)msg{}
//合成播放进度
- (void) onSpeakProgress:(int) progress{}
```

3.2. 离线合成



ttsres 目录是离线合成的引擎资源集合，common.jet 是基础资源，其他文件是发音人各自对应的资源。在实际使用时，common.jet 和发音人资源需要同时设置。发音人可以根据需要自行选择。

注意：添加离线资源时请使用引用方式，以使得程序编译时将资源编译到 app 中。如下图所示。



集成后在 Xcode 对应工程的 Build Phases 下的 copy bundle resources 项目中可以看到已集成的资源。

引擎大小：

	引擎大小
编译前静态库	18.2 MB（静态库大小）
编译后	5 MB

资源大小：

	资源大小
基础资源	4.3 MB
小燕	4.1 MB
小峰	1.3 MB
小梅	1.7 MB
凯瑟琳	3.0 MB

空间大小：

ipa 文件大小 = 引擎编译后（5MB） + 基础资源（4.3MB） + 所选择的发音人资源（如：小燕 4.1MB）。
下面代码是集成本地资源时需要添加的部分，其他代码与在线一致。

```
//生成合成引擎的资源文件路径，以发音人小燕为例，请确保资源文件的存在
NSString *resPath = [[NSBundle mainBundle] resourcePath];
NSString *newResPath = [[NSString alloc]
initWithFormat:@"%@/tts64res/common.jet;%@/tts64res/xiaoyan.jet",resPath,resPath];
//设置TTS的启动参数
[[IFlySpeechUtility getUtility] setParameter:@"tts" forKey:[IFlyResourceUtil ENGINE_START]];
//生成TTS的实例
_iFlySpeechSynthesizer = [IFlySpeechSynthesizer sharedInstance];
_iFlySpeechSynthesizer.delegate = self;
//设置本地引擎类型
 [_iFlySpeechSynthesizer setParameter:[IFlySpeechConstant TYPE_LOCAL]
forKey:[IFlySpeechConstant ENGINE_TYPE]];
//设置发音人为小燕
[_iFlySpeechSynthesizer setParameter:@"xiaoyan" forKey:[IFlySpeechConstant VOICE_NAME]];
//设置TTS合成的引擎资源文件路径
[_iFlySpeechSynthesizer setParameter:newReth forKey:@"tts_res_path"];
```

4. 语音听写

IFlySpeechRecognizer 是不带界面的语音听写控件，IFlyRecognizerView 是带界面的控件，此处仅介绍不带界面的语音听写控件。

使用示例如下所示：

```
//需要实现IFlyRecognizerViewDelegate识别会话服务代理
@interface IATViewController : UIViewController<IFlySpeechRecognizerDelegate>
{
```

```
    //不带界面的识别对象
     IFlySpeechRecognizer *iFlySpeechRecognizer;
}
```

//1.创建语音听写对象

```
_iFlySpeechRecognizer = [IFlySpeechRecognizer sharedInstance];
//设置听写模式
 [iFlySpeechRecognizer setParameter:@"iat" forKey:[IFlySpeechConstant IFLY_DOMAIN]];
```

//2.设置听写参数

```
[iFlySpeechRecognizer setParameter: @"iat" forKey: [IFlySpeechConstant IFLY_DOMAIN]];
//asr_audio_path是录音文件名，设置value为nil或者为空取消保存，默认保存目录在Library/cache下。
[iFlySpeechRecognizer setParameter:@"asrview.pcm" forKey:[IFlySpeechConstant
ASR_AUDIO_PATH]];
```

//3.启动识别服务

```
[_iFlySpeechRecognizer start];
```

//4. IFlySpeechRecognizerDelegate识别代理

```
/*识别结果返回代理
@param ： results识别结果
@ param ： isLast 表示是否最后一次结果
*/
- (void) onResults:(NSArray *) results isLast:(BOOL)isLast
/*识别会话结束返回代理
@ param   error 错误码,error.errorCode=0表示正常结束，非0表示发生错误。
*/
- (void)onError: (IFlySpeechError *) error{}
/**
 停止录音回调
 ****/
- (void) onEndOfSpeech
{}
/**
 开始识别回调
 ****/
- (void) onBeginOfSpeech
{}
/**
 音量回调函数
 volume 0－30
```

```
 ****/
- (void) onVolumeChanged: (int)volume
{}
```

5. 语义理解

使用语义理解前需要首先确保对应的appid已经开通语义功能，可参照：http://www.xfyun.cn/services/osp，语音识别结果请参照“语义开放平台API规范文档”所示。使用示例如下所示：

```
//需要实现IFlySpeechRecognizerDelegate识别会话服务代理
@interface UnderstandViewController : UIViewController<IFlySpeechRecognizerDelegate>
{
    iFlySpeechUnderstander *_iFlySpeechUnderstander;}
//1.创建语音对象
_iFlySpeechUnderstander = [IFlySpeechUnderstander sharedInstance];
_iFlySpeechUnderstander.delegate = self;
//2.启动语义理解服务
[_iFlySpeechUnderstander startListening]; //启动识别服务
//3.语义理解实现 delegate
/*语义识别结果返回代理
@ param resultArray  识别结果
@ param isLast  表示是否最后一次结果
*/
- (void) onResults:(NSArray *) results isLast:(BOOL) isLast{}
/*
@  会话结束回调，error.errorCode=0表示正常结束，非0表示发生错误。
*/
- (void) onError:(IFlySpeechError*) error{}
/*音量回调*/
- (void) onVolumeChanged: (int)volume;
/*录音开始回调*/
- (void) onBeginOfSpeech;
/*录音结束回调*/
- (void) onEndOfSpeech;
```

6. 个性化识别

个性化识别适用于语音听写，语义理解。支持上传联系人和上传词表功能，上传联系人可以增加通用录联系人的识别率，上传词表可以增加词表的识别率。

6.1. 上传联系人

上传联系人，可以提升联系人的识别率，并在语义理解的打电话、发短信业务中亦生效。使用示例如下所示：

```
//1.创建上传对象
_uploader = [[IFlyDataUploader alloc] init];
//2.创建联系人对象
IFlyContact *iFlyContact = [[IFlyContact alloc] init];
NSString *contactList = [iFlyContact contact]; //获取联系人列表
//3.设置上传参数[_uploader setParameter:@"uup" forKey:@"subject"];
[_uploader setParameter:@"contact" forKey:@"dtt"];
//4.启动上传
[_uploader uploadDataWithCompletionHandler:^(NSString *grammerID, IFlySpeechError *error){
}name:@"contact" data: contactList];
```

6.2. 上传用户词表

上传的用户词表在语音听写中优先识别。使用示例如下所示：

```
//1.创建上传对象
_uploader = [[IFlyDataUploader alloc] init];
//2.用户词表
#define  USERWORDS     @"{\"userword\":[{\"name\":\"iflytek\",\"words\":[\"德国盐猪手\",\"1912酒吧街\",\"清蒸鲈鱼\",\"挪威三文鱼\",\"黄埔军校\",\"横沙牌坊\",\"科大讯飞\"]}]}"
IFlyUserWords *iFlyUserWords = [[IFlyUserWords alloc] initWithJson:USERWORDS ];
//3.设置上传参数
```

```
[_uploader setParameter:@"iat" forKey:@"sub"];
[_uploader setParameter:@"userword" forKey:@"dtt"];
//4.启动上传（请注意 name 参数的不同）
 [_uploader uploadDataWithCompletionHandler:^(NSString * grammerID, IFlySpeechError *error){
}name: @"userwords" data:[iFlyUserWords toString]];
```

7. 语法识别

语法识别是基于语法文件的一种命令词识别技术，在线基于 abnf 语法文件，离线基于 bnf 语法文件。语法文件可以参照 demo 的工程所示。
使用示例如下所示

7.1. 在线识别

```
//1.创建识别对象
    //此方法为demo中封装，具体实现请参照demo。
    self.iFlySpeechRecognizer = [RecognizerFactory CreateRecognizer:self Domain:@"asr"];
//2.设置在线识别参数
    //开启候选结果
    [_iflySpeechRecognizer setParameter:@"1" forKey:@"asr_wbest"];
    //设置引擎类型，cloud 或者 local
    [_iflySpeechRecognizer setParameter:@ "cloud" forKey:[IFlySpeechConstant ENGINE_TYPE]];
    //设置字符编码为 utf-8
    [_iflySpeechRecognizer setParameter:@"utf-8" forKey:[IFlySpeechConstant TEXT_ENCODING]];
    //语法类型，本地是 bnf，在线识别是 abnf
    [_iflySpeechRecognizer setParameter:@ "abnf" forKey:[IFlyResourceUtil GRAMMARTYPE]];
    //启动 asr 识别引擎
    [[IFlySpeechUtility getUtility] setParameter:@"asr" forKey:[IFlyResourceUtil ENGINE_START]];
    //设置服务类型为 asr 识别
    [_iflySpeechRecognizer setParameter:@"asr" forKey:[IFlySpeechConstant IFLY_DOMAIN]];
    //设置语法构建路径，该路径为 sandbox 下的目录，请确保目录存在
    [_iflySpeechRecognizer setParameter:_grammBuildPath forKey:[IFlyResourceUtil
GRM_BUILD_PATH]];
//3. 编译语法文件，注意 grammerType 参数的区别
```

```
//读取本地 abnf 语法文件内容
grammerContent = [self readFile:_abnfFilePath];
//调用语法编译接口
[_iflySpeechRecognizer buildGrammarCompletionHandler:^(NSString * grammerID,
IFlySpeechError *error){
//4.设置 grammerID
[_iFlySpeechRecognizer setParameter:_cloudGrammerId forKey:[IFlySpeechConstant
LOCAL_GRAMMAR]];
}grammarType:@"abnf" grammarContent:grammarContent];
//5.启动识别
[_iFlySpeechRecognizer startListening];
//6.识别代理
```

本地和在线的识别返回代理是一致的，不同的是设置参数有所区别。

在切换在线和离线服务时还需要注意参数的重置,具体可以参照demo所示，一般demo是在viewWillAppear，viewWillDisappear进行处理。

7.2. 离线命令词识别

引擎大小：

	引擎大小
编译前静态库	24.5 MB
编译后	8 MB

资源大小：

	资源大小
离线命令词识别资源	5.0 MB

空间大小：

ipa 文件大小 = 引擎编译后（8MB） +资源（5MB）= 13MB。

```
//1.设置本地识别参数，其他参数与在线方式一致
    //设置引擎类型，cloud 或者 local
    [_iflySpeechRecognizer setParameter:@ "local" forKey:[IFlySpeechConstant ENGINE_TYPE]];
    //语法类型，本地是 bnf，在线识别是 abnf
    [_iflySpeechRecognizer setParameter:@ "bnf" forKey:[IFlyResourceUtil GRAMMARTYPE]];
    //启动 asr 识别引擎
    [[IFlySpeechUtility getUtility] setParameter:@"asr" forKey:[IFlyResourceUtil ENGINE_START]];
    //设置引擎资源文件路径，如 demo 中的 aitalkResource 中的 common.mp3
    [_iflySpeechRecognizer setParameter:_aitalkResourcePath forKey:[IFlyResourceUtil
ASR_RES_PATH]];
//2.编译语法文件（注意 grammarType 参数的区别）
     //读取本地 bnf 语法文件内容
    grammarContent = [self readFile:_bnfFilePath];
    //调用语法编译接口
    [_iflySpeechRecognizer buildGrammarCompletionHandler:^(NSString * grammerID,
IFlySpeechError *error){
    //3.设置 grammerID
    [_iFlySpeechRecognizer setParameter:_localgrammerId forKey:[IFlySpeechConstant
    LOCAL_GRAMMAR]];
}grammarType:@"bnf" grammarContent:grammarContent];
//4.启动识别
[_iFlySpeechRecognizer startListening];
//5.识别代理
    与在线完全一致
```

8. 声纹识别

声纹识别，主要是提供基于用户声纹特征的注册、验证服务，语音云平台支持 2 种类型的声纹密码类型，即文本密码和数字密码，在注册时需要指定声纹类型。

8.1. 声纹注册

```
//1.创建声纹对象
isvRec=[IFlyISVRecognizer sharedInstance];
isvRec.delegate=self;

//2.设置声纹工作参数
//设置密码类型，pwdt的取值为1、3，分别表示文本密码和数字密码
[isvRec setParameter:[NSString stringWithFormat:@"%d",pwdt] forKey:@"pwdt"];
```

pwdt 的取值说明如表 8-1 所示：

表 8-1 pwdt 取值与声纹服务类型

pwdt 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，现阶段支持的文本只有“芝麻开门”一种。
3	数字密码。从云端拉取一组特定的数字串（共分 5 组，每组 8 位数字），用户依次读出这 5 组数字进行注册，在验证过程中会生成一串特定的数字，用户通过朗读这串数字进行验证。

密码内容需调用接口从云端获取：

```
//通过调用getPasswordList方法来获取密码。获取密码的时候需指定声纹密码类型，pwdt为1表示固定文本密码，pwdt为3表示数字密码，自由说不需从云端获取密码。

// getPasswordList可以参照demo所示。
NSArray *tmpArray=[isvRec getPasswordList:ivppwdt];
```

获取到密码后，接下来进行声纹注册，即要求用户朗读若干次指定的内容，这一过程也称为声纹模型的训练。

```
// 设置业务类型为训练
[isvRec setParameter:@"train"    forKey:@"sst"];
// 设置密码类型
[isvRec setParameter:[NSString stringWithFormat:@"%d",pwdt] forKey:@"pwdt"];
// 对于文本密码和数字密码，必须设置密码的文本内容，ptxt的取值为“我的地盘我做主”、“移动改变生活”、“芝麻开门”或者是从云端拉取的数字密码(每8位用“-”隔开)。自由说略过此步
[isvRec setParameter:ptxt forKey:@"ptxt"];
// 设置声纹对应的auth_id，它用于标识声纹对应的用户
[isvRec setParameter:auth_id forKey:@"auth_id"];
// 设置有效录音时间
[isvRec setParameter:@"3000" forKey:@"vad_timeout"];
// 末端静音检测时间，用于检测到静音自动停止录音
[isvRec setParameter:@"700" forKey:@"vad_speech_tail"];


//3.启动训练服务
// 开始注册，当得到注册结果时，SDK会将其封装成NSDictionary对象，回调onResult方法进行处理，处理方法详见Demo示例
[isvRec startListening];

//4.声纹实现 delegate
@protocol IFlyISVDelegate
-(void) onResult:(NSDictionary *)dic;            //训练和验证正常结果返回
-(void) onError:(IFlySpeechError *) errorCode; //会话结束，正常结束错误码为0，非0表示有错误，错误码见下面描述
@optional
-(void) onRecognition;                                  //结果处理中
-(void) onVolumeChanged: (int)volume;           //录音音量发生改变
@end
```

推荐在注册声纹模型时每个用户都指定一个唯一的 auth_id。auth_id 的格式为：6-18 个字符，为字母、数字和下划线的组合且必须以字母开头，不支持中文字符，不能包含空格。

开发者通过重写 onResult 方法来处理注册和验证结果。在结果 result 中携带错误码，用来判别注册是否成功以及出错原因，部分错误码的含义如表 8-2 所示：

表 8-2 onResult 返回的部分错误码说明

错误码	说明
10106	缺少某个必要参数

10107	某个必要参数存在但无效
10110	引擎授权不足或者说授权客户端用户数达到上限
10114	操作超时
10116	数据库中模型不存在
10212	数据库中模型已经存在
10400	数据库中一般性错误，此时此 ssb 已经登录超过 3 次且均失败
10407	APPID 非法
10606	音频太短

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@“sst”参数需要设置为@“verify”，其他参数的设置、验证结果的处理过程可参考上一节。
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）。若距离较远，可能会对验证通过率产生较大影响。

8.3. 声纹模型操作

声纹注册成功后，在云端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。

```
// 开发者调用sendRequest方法查询或者删除模型，该函数的定义如下：
-(BOOL) sendRequest:
      (NSString*)cmd
      authid:(NSString *)auth_id
      pwdt:(int)pwdt
      ptxt:(NSString *)ptxt
      vid:(NSString *)vid
      err:(int *)err;
//cmd: @”query”表示查询，@”del”表示删除
//auth_id表示用户名;
//pwdt表示声纹类型;
//ptxt表示查询或者删除的密码文本;
//vid是用户注册成功后服务器返回的32位标识，查询和删除时，vid可以设置位nil;
//err是查询的错误码。 通常查询或者删除成功，该函数会返回YES,否则返回NO;
```

9. 语音评测

提供汉语、英语两种语言的评测，支持单字（汉语专有）、词语和句子朗读三种题型，通过简单地接口调用就可以集成到您的应用中。语音评测的使用主要有四个步骤：

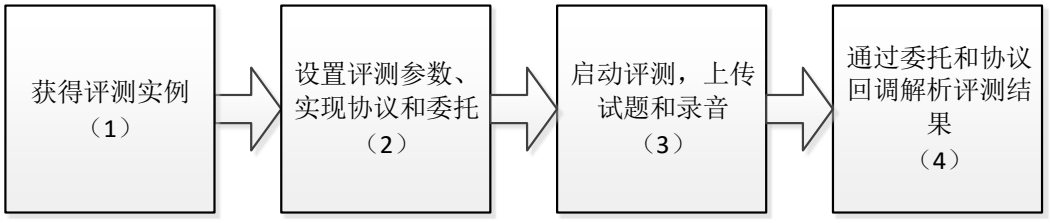


图 9-1 评测服务集成流程

```
//1.创建评测对象
_iFlySpeechEvaluator = [IFlySpeechEvaluator sharedInstance];
_iFlySpeechEvaluator.delegate = self;
//2.设置训练参数
// 清空参数
[_iFlySpeechEvaluator setParameter:@"" forKey:[IFlySpeechConstant PARAMS]];
// 设置评测采样率
[self.iFlySpeechEvaluator setParameter:@"16000" forKey:[IFlySpeechConstant SAMPLE_RATE]];
// 设置评测题目编码，如果是utf-8格式，请添加bom头，添加方式可参考demo。
[self.iFlySpeechEvaluator setParameter:@"utf-8" forKey:[IFlySpeechConstant TEXT_ENCODING]];
// 设置评测题目结果格式，目前仅支持xml
[self.iFlySpeechEvaluator setParameter:@"xml" forKey:[IFlySpeechConstant ISE_RESULT_TYPE]];
// 设置评测前端点超时
[self.iFlySpeechEvaluator setParameter:self.iseParams.bos forKey:[IFlySpeechConstant VAD_BOS]];
// 设置评测后端点超时
[self.iFlySpeechEvaluator setParameter:self.iseParams.eos forKey:[IFlySpeechConstant VAD_EOS]];
// 设置评测前端点设置评测题型
[self.iFlySpeechEvaluator setParameter:self.iseParams.category forKey:[IFlySpeechConstant
ISE_CATEGORY]];
// 设置评测语言
[self.iFlySpeechEvaluator setParameter:self.iseParams.language forKey:[IFlySpeechConstant
LANGUAGE]];
// 设置评测结果级别
[self.iFlySpeechEvaluator setParameter:self.iseParams.rstLevel forKey:[IFlySpeechConstant
```

```
ISE_RESULT_LEVEL]];
// 设置评测超时
[self.iFlySpeechEvaluator setParameter:self.iseParams.timeout forKey:[IFlySpeechConstant
SPEECH_TIMEOUT]];
```

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	评测题型，可选值：read_syllable（单字，汉语专有）、read_word（词语）、read_sentence（句子）	是
text_encoding	上传的试题编码格式，可选值：gb2312、utf-8。当进行汉语评测时，必须设置成 utf-8，建议所有试题都使用 utf-8 编码	是
vad_bos	前端点超时，默认 5000ms	否
vad_eos	后端点超时，默认 1800ms	否
speech_timeout	录音超时，当录音达到时限将自动触发 vad 停止录音，默认-1（无超时）	否
result_level	评测结果等级，可选值：plain（仅英文）、complete，默认为 complete	否

表 3-1 评测相关参数说明

```
//3.语音评测实现 Delegate
/*!
 *  音量和数据回调
 *
 *  @param volume  音量
 *  @param buffer  音频数据
 */
- (void)onVolumeChanged:(int)volume buffer:(NSData *)buffer{
}

/*!
 *  开始录音回调
 *  当调用了`startListening`函数之后，如果没有发生错误则会回调此函数。如果发生错误则回调onError:函数
 */
```

```
- (void)onBeginOfSpeech{
}

/*!
 *  停止录音回调
 *  当调用了`stopListening`函数或者引擎内部自动检测到断点，如果没有发生错误则回调此函数。
 *  如果发生错误则回调onError:函数
 */
- (void)onEndOfSpeech{
}

/*!
 *  正在取消
 */
- (void)onCancel{
}

/*!
 *  评测错误回调
 *  在进行语音评测过程中的任何时刻都有可能回调此函数，你可以根据errorCode进行相应的处理.
 *  当errorCode没有错误时，表示此次会话正常结束，否则，表示此次会话有错误发生。特别的当调用
 *  `cancel`函数时，引擎不会自动结束，需要等到回调此函数，才表示此次会话结束。在没有回调此函
 *  数之前如果重新调用了`startListenging`函数则会报错误。
 *
 *  @param errorCode  错误描述类
 */
- (void)onError:(IFlySpeechError *)errorCode{
}

/*!
 *  评测结果回调
 *  在评测过程中可能会多次回调此函数，你最好不要在此回调函数中进行界面的更改等操作，只需要将回调的结果保存起来。
 *
 *  @param results -[out] 评测结果。
 *  @param isLast  -[out] 是否最后一条结果
 */
- (void)onResults:(NSData *)results isLast:(BOOL)isLast{
}
```

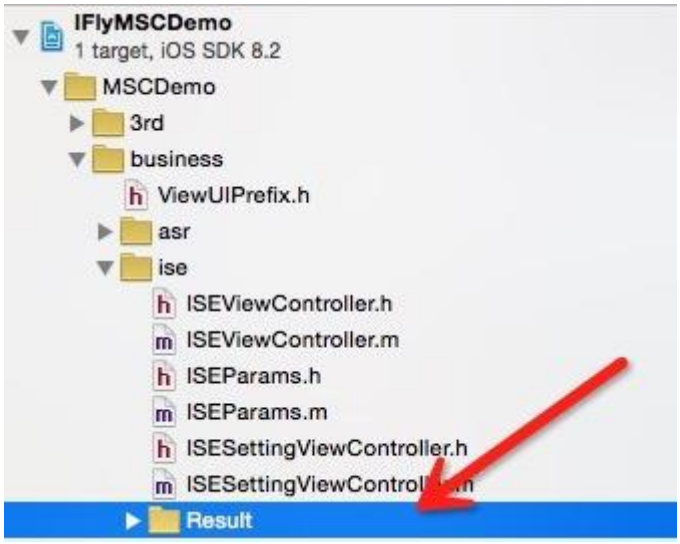
调用 startListening 即开始评测录音，读完试题内容后可以调用 stopListening 停止录音，也可以在一段时间后由 SDK 自动检测 vad 并停止录音。当评测出错时，SDK 会回调 onError 方法抛出 IFlySpeechError
错误，通过 IFlySpeechError 的 getErrorCode()方法可获得错误码，常见的错误码详见附录 14.3 和下表：

表 2 评测错误码

错误码	错误值	含义
MSP_ERROR_ASE_EXCEP_SILENCE	11401	无语音或音量太小
MSP_ERROR_ASE_EXCEP_SNRATIO	11402	信噪比低或有效语音过短
MSP_ERROR_ASE_EXCEP_PAPERDATA	11403	非试卷数据
MSP_ERROR_ASE_EXCEP_PAPERCONTENTS	11404	试卷内容有误
MSP_ERROR_ASE_EXCEP_NOTMONO	11405	录音格式有误
MSP_ERROR_ASE_EXCEP_OTHERS	11406	其他评测数据异常，包括错读、漏读、恶意录入、试卷内容等错误
MSP_ERROR_ASE_EXCEP_PAPERFMT	11407	试卷格式有误
MSP_ERROR_ASE_EXCEP_ULISTWORD	11408	存在未登录词，即引擎中没有该词语的信息

解析评测结果

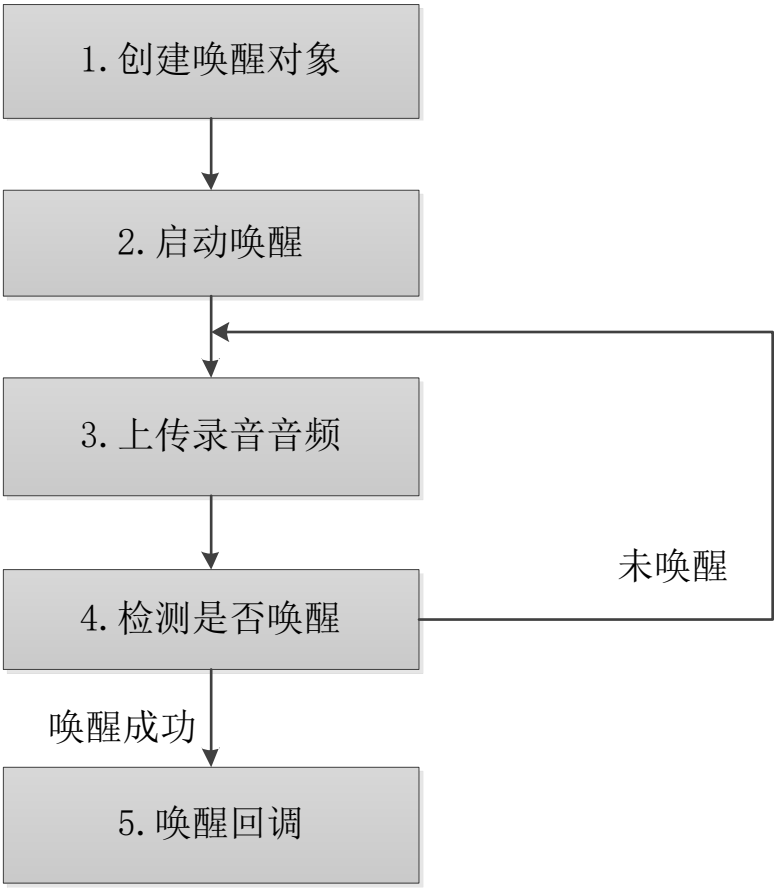
SDK 通过 onResult 回调抛出 xml 格式的评测结果，结果格式及字段含义详见《语音评测参数、结果说明文档》文档，具体的解析过程可参考 demo 工程 IFlyMscDemo 中 ISE 目录下 Result 目录中的源代码。



10. 语音唤醒

10.1. 语音唤醒

语音唤醒集成流程如下。



使用示例如下所示：(请注意，启动唤醒需要关注系统录音权限)

```
#import "iflyMSC/IFlyVoiceWakeuperDelegate.h"
#import "iflyMSC/IFlyVoiceWakeuper.h"

//1.创建唤醒对象

_iflyVoiceWakeuper = [IFlyVoiceWakeuper sharedInstance];
_iflyVoiceWakeuper.delegate = self;

//2.设置唤醒参数

//生成唤醒资源路径，唤醒资源需要定制，与唤醒词一一对应。
NSString *wordPath = [[NSBundle mainBundle] pathForResource:@"ivwres/wakeresource"
ofType:@"irf"];
NSString *ivwResourcePath = [[NSString alloc] initWithFormat:@"fo|%@",wordPath];
```

```
//设置唤醒资源
 [[IFlySpeechUtility getUtility] setParameter:[NSString stringWithFormat:
@"engine_start=ivw,ivw_res_path=%@",ivwResourcePath] forKey:[IFlyResourceUtil
ENGINE_START]];
//设置唤醒门限值。
//0：表示第一个唤醒词，-20 表示唤醒词对应的门限值；
//1：表示第二个唤醒词，-20 表示唤醒词对应的门限值
//唤醒门限值需要根据下载的 SDK 中的说明来设置
[_iflyVoiceWakeuper setParameter:@"0:-20;1:-20;" forKey:@"ivw_threshold"];
```

请注意：此实例包含两个唤醒词，如果只有一个唤醒词，则只需要设置第一个则可，后面的“1:-20”需要删除。

```
//设置唤醒的服务类型，暂时仅支持 wakeup
[_iflyVoiceWakeuper setParameter:@"wakeup" forKey:@"ivw_sst"];
//设置唤醒的工作模式，keep_alive 表示一次唤醒成功后是否继续录音等待唤醒。1：表示继续；0：表示唤醒终止
[_iflyVoiceWakeuper setParameter:@"1" forKey:@"keep_alive"];
```

//3.启动唤醒

```
int bRet = [self.iflyVoiceWakeuper startListening];
```

//4.唤醒实现 delegate

```
//录音开始
-(void) onBeginOfSpeech{}
//录音结束
-(void) onEndOfSpeech{}
//会话错误
-(void) onError:(IFlySpeechError *)error{}
//音量变化回调
- (void) onVolumeChanged: (int)volume{}
//唤醒结果回调
-(void) onResult:(NSMutableDictionary *)resultArray{}
```

10.2. 语音唤醒 Oneshot

oneshot 是唤醒的一种扩展方式，支持唤醒+识别，唤醒+听写，唤醒+语义的组合解决方案。

此处介绍唤醒+识别模式，可以解决唤醒词数量有限，命令词无法常驻运行的限制。在游戏场景下，可采用声控的方式操控游戏。定义唤醒词为阿里巴巴，定义命令词序列：“打开城门”，“发起攻击”，这样就形成若干命令序列。用户可以说：

- 阿里巴巴打开城门；
- 阿里巴巴发起攻击。

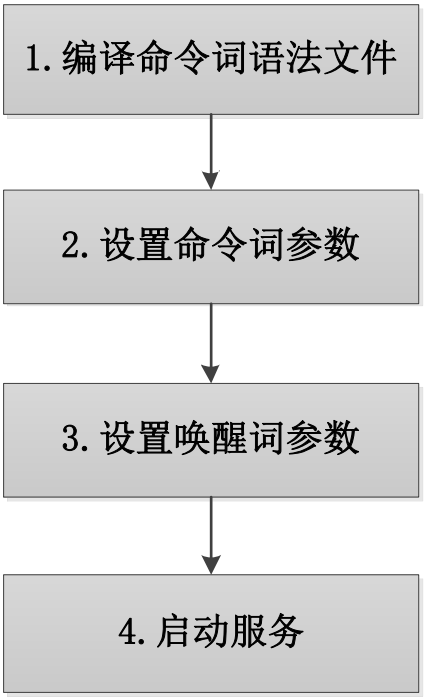
除了游戏外，用户可以很容易的将唤醒+识别的应用扩展到其他领域，比如家电，车载等比较适合语音的场所。

1) 详细 demo

唤醒+识别的代码比较复杂，限于篇幅不在本文档描述，详细过程可以参考 Demo 中的 OneshotViewController 示例代码。

Demo 的示例，用户可以说 讯飞语音/讯飞语点 + 张三/李四/张海洋 的组合。

2) 使用唤醒+识别步骤



3) 唤醒+识别回调

```
//录音开始
- (void) onBeginOfSpeech
//录音结束
- (void) onEndOfSpeech
//音量回调
- (void) onVolumeChanged: (int)volume
//服务结束回调
- (void) onError:(IFlySpeechError *) error
//唤醒结果回调
- (void) onResult:(NSMutableDictionary *)resultArray
//服务结果回调
      eventType= IFly_EVENT_IVW_RESULT，表示识别结果回调
      isLast，表示是否最后结果
eventData，存放识别结果
- (void) onEvent:(int)eventType isLast:(BOOL)isLast arg1:(int)arg1
data:(NSMutableDictionary *)eventData
```

11. 人脸识别

人脸识别不仅可以检测出照片中的人脸，还可以进行人脸注册和验证。相关概念的说明如下：

表 11-1 人脸识别概念说明

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：

表 11-2 上传图片规格要求

项目	要求
色彩、格式	彩色，PNG、JPG、BMP 格式的图片。
人脸大小、角度	大小应超过 100*100 像素，可以容忍一定程度的侧脸，为保证识别准确率，最好使用正脸图片。

光照	均匀光照，可容忍部分阴影。
遮挡物	脸部尽量无遮挡，眼镜等物品会一定程度上影响准确率。

11.1. 人脸注册

```
// 获得人脸识别请求对象
self.iFlySpFaceRequest=[IFlyFaceRequest sharedInstance];
[self.iFlySpFaceRequest setDelegate:self];
// 设置参数
[self.iFlySpFaceRequest setParameter:[IFlySpeechConstant FACE_REG] forKey:[IFlySpeechConstant
FACE_SST]];
[self.iFlySpFaceRequest setParameter:USER_APPID forKey:[IFlySpeechConstant APPID]];
// 以下参数可选，当设置参数auth_id时，验证时可通过auth_id验证
[self.iFlySpFaceRequest setParameter:USER_APPID forKey:@"auth_id"];
// 以下参数可选，当设置参数property为del时原模型删除后重新注册
[self.iFlySpFaceRequest setParameter:@"del" forKey:@"property"];
// 压缩图片大小
NSData* imgData=[_imgToUse.image compressedData];
// 发送请求
[self.iFlySpFaceRequest sendRequest:imgData];

// 响应回调
/**
 * 消息回调
 * @param eventType 消息类型
 * @param params 消息数据对象
 */
- (void) onEvent:(int) eventType WithBundle:(NSString*) params{
      NSLog(@"onEvent | params:%@",params);
}

/**
 * 数据回调，可能调用多次，也可能一次不调用
 * @param buffer 服务端返回的二进制数据
 */
- (void) onData:(NSData* )data{
}
```

```
/**
 * 结束回调，没有错误时，error 为 null
 * @param error 错误类型
 */
- (void) onCompleted:(IFlySpeechError*) error{
}
```

以上示例仅给出大致流程，详情请参考 Demo。

11.2. 人脸验证

```
// 获得人脸识别请求对象
self.iFlySpFaceRequest=[IFlyFaceRequest sharedInstance];
[self.iFlySpFaceRequest setDelegate:self];
// 设置参数
[self.iFlySpFaceRequest setParameter:[IFlySpeechConstant FACE_VERIFY] forKey:[IFlySpeechConstant F
ACE_SST]];
[self.iFlySpFaceRequest setParameter:USER_APPID forKey:[IFlySpeechConstant APPID]];
//auth_id 用于第三方跟用户关联，由第三方传入和管理
[self.iFlySpFaceRequest setParameter:USER_APPID forKey:@"auth_id"];
//若第三方未传入auth_id 则需要传入gid 来验证，详情参考Demo示例
NSUserDefaults* userDefaults=[NSUserDefaults standardUserDefaults];
NSString* gid=[userDefaults objectForKey:KCIFlyFaceResultGID];
[self.iFlySpFaceRequest setParameter:gid forKey:[IFlySpeechConstant FACE_GID]];
[self.iFlySpFaceRequest setParameter:@"2000" forKey:@"wait_time"];
// 压缩图片大小
 NSData* imgData=[_imgToUse.image compressedData];
// 发送请求
[self.iFlySpFaceRequest sendRequest:imgData];
```

注册/验证结果中包含了是否成功、gid 等信息，详细的 JSON 格式请参照附录 13.5 在线人脸识别结果说明 具体解析过程详见 Demo 工程。

11.3. 人脸检测

```
// 获得人脸识别请求对象
self.iFlySpFaceRequest=[IFlyFaceRequest sharedInstance];
[self.iFlySpFaceRequest setDelegate:self];
// 设置参数
[self.iFlySpFaceRequest setParameter:[IFlySpeechConstant FACE_DETECT] forKey:[IFlySpeechConstant FACE_SST]];
```

```
[self.iFlySpFaceRequest setParameter:USER_APPID forKey:[IFlySpeechConstant APPID]];
// 压缩图片大小
NSData* imgData=[_imgToUse.image compressedData];
//发送请求
[self.iFlySpFaceRequest sendRequest:imgData];
```

11.4. 人脸聚焦

```
// 获得人脸识别请求对象
self.iFlySpFaceRequest=[IFlyFaceRequest sharedInstance];
[self.iFlySpFaceRequest setDelegate:self];
// 设置参数
[self.iFlySpFaceRequest setParameter:[IFlySpeechConstant FACE_ALIGN] forKey:[IFlySpeechConstant FACE_SST]];
[self.iFlySpFaceRequest setParameter:USER_APPID forKey:[IFlySpeechConstant APPID]];
// 压缩图片大小
NSData* imgData=[_imgToUse.image compressedData];
//发送请求
[self.iFlySpFaceRequest sendRequest:imgData];
```

12. 离线人脸检测

离线人脸检测包含离线图片检测和离线视频流检测两个功能。离线人脸功能具有较强的实时性，效果却不如在线能力好， 对于某些图片（如侧脸照、偏头照等），有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。

注意：离线检测目前只支持 iPhone 4s、iPad3 以上设备（不包括 iPhone 4s、iPad3）。

12.1. 图片检测

```
// 获得离线人脸检测对象
self.faceDetector=[IFlyFaceDetector sharedInstance];
// 检测
NSString* strResult=[self.faceDetector detectARGB:[_imgToUse image]];
//解析结果，详情请参考 demo
[self praseDetectResult:strResult];
```

12.2. 视频流检测

视频流检测中难点在视频帧的捕获和帧图片的位置转换，具体方法可以参考 Demo，使用时注意及时清理不再使用的数据以免内存卷积导致占用过大。

```
// 获得离线人脸检测对象
self.faceDetector=[IFlyFaceDetector sharedInstance];
// 设置参数
[self.faceDetector setParameter:strEnable forKey:@"detect"];
|[self.faceDetector setParameter:strEnable forKey:@"align"];
// 检测
NSString* strResult=[self.faceDetector trackFrame:faceImg.data withWidth:faceImg.width height:faceImg.h
eight direction:(int)faceImg.direction];
// 解析部分请参考 demo
```

结果格式请参考附录 13.6 离线人脸检测结果格式说明

13. 附录

13.1. 常见问题整理

参见论坛帖子：iOS MSC SDK 常见问题整理。

13.2. 语音识别结果说明

json 字段	英文全称	类型	说明
sn	sentence	int	第几句
ls	last sentence	boolean	是否最后一句
bg	begin	int	开始
ed	end	int	结束
ws	words	array	词
cw	chinese word	array	中文分词
w	word	string	单字
sc	socre	int	分数

语音听写结果示例：

```
{"sn":1,"ls":true,"bg":0,"ed":0,"ws":[{"bg":0,"cw":[{"w":"  今
天  ","sc":0}]},{"bg":0,"cw":[{"w":"  的","sc":0}]},{"bg":0,"cw":[{"w":"  天
气  ","sc":0}]},{"bg":0,"cw":[{"w":"  怎么样  ","sc":0}]},{"bg":0,"cw":[{"w":"  。
","sc":0}]}]}
```

多候选结果示例：

```
{"sn":1,"ls":false,"bg":0,"ed":0,"ws":[
{"bg":0,"cw":[{"w":"我想听","sc":0}]},
{"bg":0,"cw":[{"w":"拉德斯基进行曲","sc":0},{"w":"拉得斯进行曲","sc":0}]}]}
```

语法识别结果示例：

```
{"sn":1,"ls":true,"bg":0,"ed":0,"ws":[
        {"bg":0,"cw":[{"sc":"70","gm":"0","w":"北京到上海"},
                      {"sc":"69","gm":"0","w":"天京到上海"},
                      {"sc":"58","gm":"0","w":"东京到上海"}
                     ]
        }
        ]}
```

13.3. 声纹结果说明

文本密码 json 格式串

```
{"txt_pwd":["芝麻开门"]}
```


数字密码 json 格式串

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

声纹业务结果（NSDictionary 类型结果）成员说明

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

13.4. 合成发音人列表

语言：

1、 语言为中英文的发音人可以支持中英文的混合朗读。
2、 英文发音人只能朗读英文，中文无法朗读。
3、 汉语发音人只能朗读中文，遇到英文会以单个字母的方式进行朗读。
4、 使用**新引擎参数**会获得更好的合成效果。

发音人名称	属性	语言	参数名称	新引擎参数	备注
小燕	青年女声	中英文（普通话）	xiaoyan		默认
小宇	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	vixl	xiaolin	

		话）			
小蓉	青年女声	汉语（四川话）	vixr	xiaorong	
小芸	青年女声	汉语（东北话）	vixyun	xiaoqian	
小坤	青年男声	汉语（河南话）	vixk	xiaokun	
小强	青年男声	汉语（湖南话）	vixqa	xiaoqiang	
小莹	青年女声	汉语（陕西话）	vixying		
小新	童年男声	汉语（普通话）	vixx	xiaoxin	
楠楠	童年女声	汉语（普通话）	vinn	nannan	
老孙	老年男声	汉语（普通话）	vils		
Mariane		法语	Mariane		
Guli		维语	Guli		
Allabent		俄语	Allabent		
Gabriela		西班牙语	Gabriela		
Abha		印地语	Abha		
XiaoYun		越南语	XiaoYun		

13.5. 在线人脸识别结果说明

json 字段	类型	说明
sst	string	指定本路会话是属于何种性质
gid	string	图像模型 id
rst	bool	结果
sid	string	会话 id
ret	int	错误码
uid	string	用户 id
score	float	打分
pose	dictionary	面部朝向
confidence	float	置信度
position	dictionary	面部的矩形区域
landmark	dictionary	关键点数组

注册时响应结果返回 json 格式串：

```
{
    "sst": "reg",
    "gid": "4a6c124ed6b78436ee5aac4563f13eb5",
    "rst": "success",
    "sid": "wfr2717002e@hf4aed073fdccb4af700",
    "ret": "0",
    "uid": "123456"
```

```
}
```

验证时响应结果返回 json 格式串：

```
{
    "gid": "wfr278722e9@ch51370817c1f0477300",
    "ret": "0",
    "rst": "success",
    "score": "83.470",
    "sid": "wfr279e22e8@ch51370817c1fb477400",
    "sst": "verify",
    "uid": "a1644276827",
    "verf": true
}
```

人脸检测时响应结果返回 json 格式串：

```
{
    "face": [
        {
            "attribute": {
                "pose": {
                    "pitch": 1
                }
            },
            "confidence": "10.412",
            "position": {
                "bottom": 447,
                "left": 140,
                "right": 419,
                "top": 168
            },
            "tag": ""
        }
    ],
    "ret": "0",
    "rst": "success",
    "sid": "wfr278422e9@ch47fc0817c22e477000",
    "sst": "detect",
    "uid": "a1644276827"
}
```

关键点检测时响应结果返回 json 格式串：

```
{
```

```
"result": [
    {
        "landmark": {
            "left_eye_center": {
                "x": "209.739",
                "y": "229.428"
            },
            "left_eye_left_corner": {
                "x": "177.219",
                "y": "230.914"
            },
            "left_eye_right_corner": {
                "x": "235.839",
                "y": "236.793"
            },
            "left_eyebrow_left_corner": {
                "x": "155.253",
                "y": "187.392"
            },
            "left_eyebrow_middle": {
                "x": "199.240",
                "y": "182.701"
            },
            "left_eyebrow_right_corner": {
                "x": "246.582",
                "y": "192.358"
            },
            "mouth_left_corner": {
                "x": "204.203",
                "y": "386.777"
            },
            "mouth_lower_lip_bottom": {
                "x": "262.768",
                "y": "416.832"
            },
            "mouth_middle": {
                "x": "263.705",
                "y": "390.507"
            },
            "mouth_right_corner": {
                "x": "317.841",
                "y": "390.864"
            },
            "mouth_upper_lip_top": {
```

```
                "x": "264.736",
                "y": "367.996"
            },
            "nose_bottom": {
                "x": "267.811",
                "y": "339.358"
            },
            "nose_left": {
                "x": "225.449",
                "y": "319.586"
            },
            "nose_right": {
                "x": "308.086",
                "y": "323.936"
            },
            "nose_top": {
                "x": "271.755",
                "y": "310.934"
            },
            "right_eye_center": {
                "x": "335.608",
                "y": "234.335"
            },
            "right_eye_left_corner": {
                "x": "306.995",
                "y": "238.703"
            },
            "right_eye_right_corner": {
                "x": "364.231",
                "y": "240.307"
            },
            "right_eyebrow_left_corner": {
                "x": "300.652",
                "y": "194.243"
            },
            "right_eyebrow_middle": {
                "x": "347.711",
                "y": "188.787"
            },
            "right_eyebrow_right_corner": {
                "x": "391.572",
                "y": "197.455"
            }
        }
```

```
            }
        ],
        "ret": "0",
        "rst": "success",
        "sid": "wfr278522e9@ch47fc0817c255477600",
        "sst": "align",
        "uid": "a1644276827"
}
```


13.6. 离线人脸检测结果格式说明

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， 其他值为错误码
face	dictionary	检测到的人脸信息
position	dictionary	人脸框位置信息
landmark	dictionary	视频流人脸关键点位置信息

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
                    }
                }
            }
        ]
}
```

13.7. 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
SPEECH_ERROR_NO_NETWORK	20001	无有效的网络连接
SPEECH_ERROR_NETWORK_TIMEOUT	20002	网络连接超时
SPEECH_ERROR_NET_EXPECTION	20003	网络异常
SPEECH_ERROR_INVALID_RESULT	20004	无有效的结果
SPEECH_ERROR_NO_MATCH	20005	无匹配结果
SPEECH_ERROR_AUDIO_RECORD	20006	录音失败
SPEECH_ERROR_NO_SPPECH	20007	未检测到语音
SPEECH_ERROR_SPEECH_TIMEOUT	20008	音频输入超时
SPEECH_ERROR_EMPTY_UTTERANCE	20009	无效的文本输入
SPEECH_ERROR_FILE_ACCESS	20010	文件读写失败
SPEECH_ERROR_PLAY_MEDIA	20011	音频播放失败
SPEECH_ERROR_INVALID_PARAM	20012	无效的参数
SPEECH_ERROR_TEXT_OVERFLOW	20013	文本溢出
SPEECH_ERROR_INVALID_DATA	20014	无效数据
SPEECH_ERROR_LOGIN	20015	用户未登陆
SPEECH_ERROR_PERMISSION_DENIED	20016	无效授权
SPEECH_ERROR_INTERRUPT	20017	被异常打断
SPEECH_ERROR_VERSION_LOWER	20018	版本过低
ERROR_IN_USE	20019	未知错误
SPEECH_ERROR_NO_NETWORK	20001	无有效的网络连接
SPEECH_ERROR_NETWORK_TIMEOUT	20002	网络连接超时
SPEECH_ERROR_NET_EXPECTION	20003	网络异常
SPEECH_ERROR_INVALID_RESULT	20004	无有效的结果
SPEECH_ERROR_NO_MATCH	20005	无匹配结果
SPEECH_ERROR_AUDIO_RECORD	20006	录音失败
SPEECH_ERROR_NO_SPPECH	20007	未检测到语音
SPEECH_ERROR_SPEECH_TIMEOUT	20008	音频输入超时
SPEECH_ERROR_EMPTY_UTTERANCE	20009	无效的文本输入

13.8. 集成帮助文档

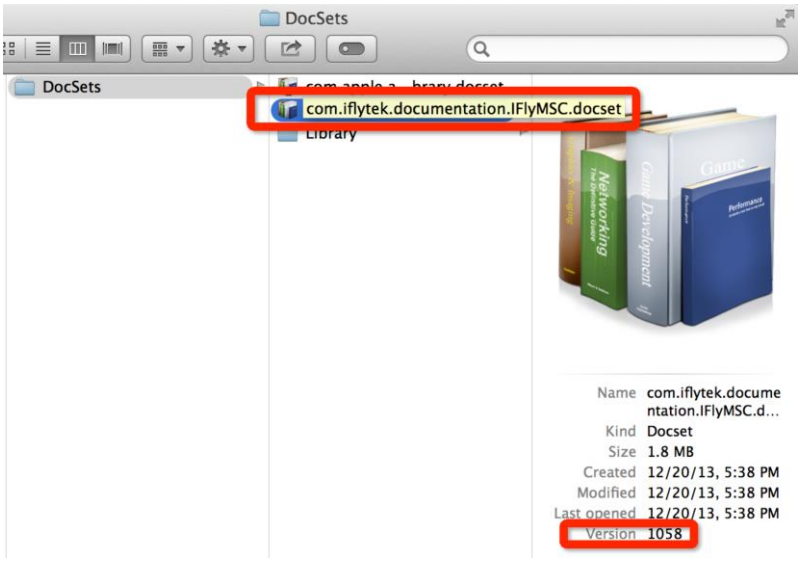
打开终端（termainl 或 iterm），cd 到压缩包的 doc 目录，执行以下命令：
注：不同的 Xcode 版本，对应的 docset 路径可能有变化，需要根据实际路径来操作。
（例如：/Applications/Xcode.app/Contents/Developer/Documentation/DocSets/）

```
cp -R -f -a com.iflytek.documentation.IFlyMSC.docset
~/Library/Developer/Shared/Documentation/DocSets/
```

然后执行命令

```
open ~/Library/Developer/Shared/Documentation/DocSets/
```

请核对文档的版本为最新下载的版本



打开 Xcode 的帮助文档就可以看到已经集成的文档