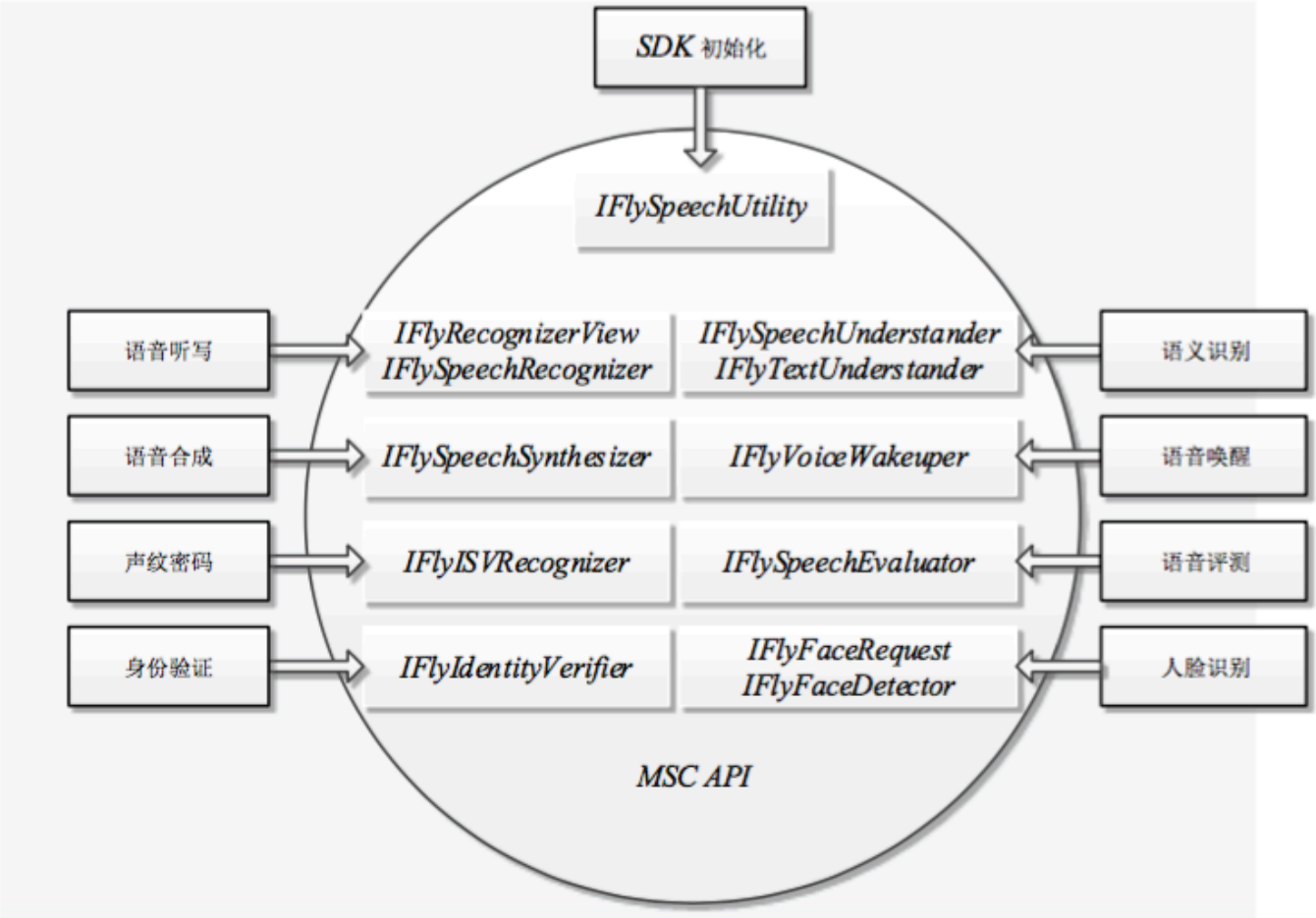
科大讯飞语音识别 v1.146

1. 简述：

(1) 科大讯飞语音 SDK 提供的主要功能有：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。

(2) iflyMSC 主要功能接口如下图所示：

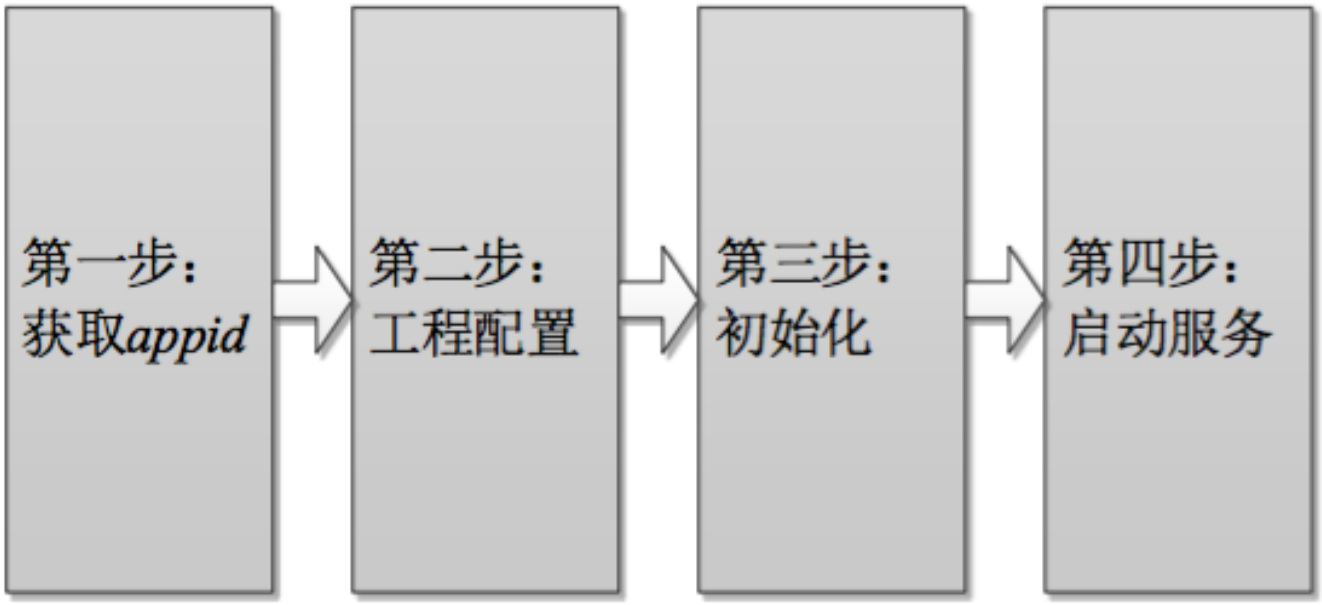


(3) iflyMSC 功能名词解释

名词	解释
语音合成	将一段文字转换成语音，可根据需要将文字合成出不同音色、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，把语音中包含的文字信息提取出来，并可以优先识别用户手机中特有的联系人和个性化数据。
语法识别	判断用户所说的内容是否与预定义的语法相符合，主要用于识别用户是否下达某项指令。使用语法识别前，需要先定义语法。
语义理解	在语音听写基础上，分析理解用户的说话意图，返回结构化的指令信息。开发者可在语义开放平台定义专属的问答格式。
语音评测	通过智能语音技术自动对发音水平进行评价，给出用户综合得分和发音信息。

声纹密码	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测、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。
身份验证	在本方案中，开发者可根据应用场景灵活的选择身份验证方式，如单人脸验证、单声纹验证以及人脸+声纹的融合验证方式。这样既解决了单生物特征识别暴露的局限性，也提供了更精准、更安全的识别和检测方案。身份验证方案还会持续增加更多的常用特征，达到更广泛的市场应用前景。

2. 集成流程



(1) 获取 appid

appid 是第三方应用集成语音云开放平台 sdk 的身份标示，sdk 静态库是和 appid 绑定的，下载 sdk 时会要求选择对应 appid，否者 sdk 和项目的 appid 不一致会出现 10407 错误码。

(2) 工程配置

1) 添加 iflyMSC 静态库和系统库。注意：添加 iflySMC 静态库时，请检查 BuildSetting 中的 framwork path 的设置，如果出现找不到 framework 的情况，可以将 path 清空，在 Xcode 中删除 framework，然后重新添加。

2) xcode7,8 默认开启了 Bitcode，需要工程所依赖所有库都支持 Bitcode，由于暂时不支持,所以临时关闭,在 Build Settings 中搜索 Bitcode 设为 NO 即可。

3) 为适配 ios9，info.plist 如下配置

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

4) 用户隐私权限设置

iOS 10 发布以来，苹果为了用户信息安全，加入隐私权限设置机制，让用户来选择是否允许。隐私权限配置可在 *info.plist* 新增相关 *privacy* 字段，*MSC SDK* 中需要用到的权限主要包括麦克风权限、联系人权限和地理位置权限：

```
<key>NSMicrophoneUsageDescription</key>
<string></string>
<key>NSLocationUsageDescription</key>
<string></string>
<key>NSLocationAlwaysUsageDescription</key>
<string></string>
<key>NSContactsUsageDescription</key>
<string></string>
```

即在 *Info.plist* 中增加下图设置：

图 2-5 privacy 配置

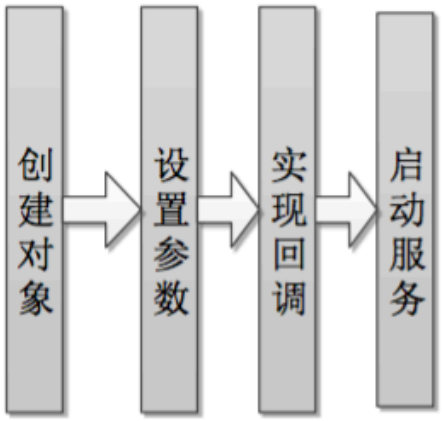
(3) 初始化

```
- (void)initIFlySetting {
    //设置sdk的log等级，log保存在下面设置的工作路径中
    [IFlySetting setLogFile:LVL_ALL];

    //打开输出在console的log开关
    [IFlySetting showLogcat:NO];

    //设置sdk的工作路径
    NSArray *paths = NSSearchPathForDirectoriesInDomains(NSCachesDirectory, NSUserDomainMask, YES);
    NSString *cachePath = [paths objectAtIndex:0];
    [IFlySetting setLogFilePath:cachePath];

    //创建语音配置,appid必须要传入，仅执行一次则可
    NSString *initString = @"appid=584445ae";

    //所有服务启动前，需要确保执行createUtility
    [IFlySpeechUtility createUtility:initString];
}
```

(4) 启动服务

所有的服务皆遵循如下的流程，如下图：

图 2-7 启动服务流程



3. 语音听写

```
@interface MYNotesSpeechSearchViewController ()<IFlySpeechRecognizerDelegate>//识别协议

@property (strong, nonatomic) MYSearchHisttoryModel *searchModel;

@property (weak, nonatomic) IBOutlet UIButton *recognizerButton;
@property (strong, nonatomic) IFlySpeechRecognizer *iFlySpeechRecognizer;//不带界面的识别对象
```

```
#pragma mark private method

-(void)initRecognizer {
  //单例模式，无UI的实例
  if (_iFlySpeechRecognizer == nil) {
    _iFlySpeechRecognizer = [IFlySpeechRecognizer sharedInstance];//创建语音识别对象

    //设置识别参数

    //扩展参数
    [_iFlySpeechRecognizer setParameter:@"" forKey:[IFlySpeechConstant PARAMS]];

    //设置听写模式(应用领域)
    [_iFlySpeechRecognizer setParameter:@"iat" forKey:[IFlySpeechConstant IFLY_DOMAIN]];

    //设置最长录音时间
    [_iFlySpeechRecognizer setParameter:@"30000" forKey:[IFlySpeechConstant SPEECH_TIMEOUT]];
    //设置后端点
    [_iFlySpeechRecognizer setParameter:@"3000" forKey:[IFlySpeechConstant VAD_EOS]];
    //设置前端点
    [_iFlySpeechRecognizer setParameter:@"3000" forKey:[IFlySpeechConstant VAD_BOS]];
    //网络等待时间
    [_iFlySpeechRecognizer setParameter:@"20000" forKey:[IFlySpeechConstant NET_TIMEOUT]];
```

```
  //设置采样率，推荐使用16K
  [_iFlySpeechRecognizer setParameter:@"16000" forKey:[IFlySpeechConstant SAMPLE_RATE]];

  //设置语言
  [_iFlySpeechRecognizer setParameter:@"zh_cn" forKey:[IFlySpeechConstant LANGUAGE]];
  //设置方言
  [_iFlySpeechRecognizer setParameter:@"mandarin" forKey:[IFlySpeechConstant ACCENT]];
```

```
    //设置是否返回标点符号
    [_iFlySpeechRecognizer setParameter:@"0" forKey:[IFlySpeechConstant ASR_PTT]];

    //设置听写结果格式为json
    [_iFlySpeechRecognizer setParameter:@"json" forKey:[IFlySpeechConstant RESULT_TYPE]];

    //设置音频来源为麦克风
    [_iFlySpeechRecognizer setParameter:IFLY_AUDIO_SOURCE_MIC forKey:@"audio_source"];
  }

  _iFlySpeechRecognizer.delegate = self;
}
```

```
- (IBAction)onStartRecognizer:(id)sender {
  [self.audioWaverView setHidden:NO];
   [self showRecongizering];
  [self hidenEmptyResult];

  if(_iFlySpeechRecognizer == nil)
     [self initRecognizer];//初始化识别对象

  if (_iFlySpeechRecognizer.isListening)//正在识别
     return;

  [_iFlySpeechRecognizer cancel];//取消之前服务

  BOOL ret = [_iFlySpeechRecognizer startListening];//启动识别服务

  if (!ret) {
     UIAlertView *alterView = [[UIAlertView alloc] initWithTitle:@"提示" message:@"语音识别失败" delegate:nil cancelButtonTitle:@"确定" otherButtonTitles:nil,
        nil];
     [alterView show];
  }
}
```

```
- (IBAction)onEndRecognizer:(id)sender {
  [self.audioWaverView setHidden:YES];
   [self hideRecongizering];

  self.audioVolum = 0.0f;
  [_iFlySpeechRecognizer stopListening];//关闭识别服务
}
```

```
//识别结果返回代理
- (void)onResults:(NSArray *)results isLast:(BOOL)isLast {
  if (isLast)
     return ;

  NSMutableString *resultString = [[NSMutableString alloc] init];
  NSDictionary *dic = results[0];
  for (NSString *key in dic)
     [resultString appendFormat:@"%@",key];

  NSString *resultFromJson =  [self stringFromJson:resultString];

  [self.titleLabel setText:resultFromJson];

  [self requestResult:resultFromJson];
}

//音量回调函数
- (void)onVolumeChanged:(int)volume {
  self.audioVolum = volume/50.0f;
}

//开始录音回调
- (void)onBeginOfSpeech{}

//停止录音回调
- (void)onEndOfSpeech{}

//会话取消回调
- (void)onCancel{}
```

```
- (void)requestResult:(NSString *)keyWords {
    [self searchManagerLoading];

    NSArray *array = [[MYNotesUtility defaultUtility] filterArrayWithPredicate:[NSPredicate predicateWithFormat:@"(ParentID != 0) AND (Name CONTAINS[c]
        %@)", keyWords]];
     [self hideRecongizering];
    if([array count]){
        [self performSelector:@selector(popViewController) withObject:nil afterDelay:0.33];
    }else{
         [self showEmptyResult];
        return;
    }
}
```

```
//解析听写json格式的数据
- (NSString *)stringFromJson:(NSString*)params
{
    if (params == NULL) {
        return nil;
    }

    NSMutableString *tempStr = [[NSMutableString alloc] init];
    NSDictionary *resultDic  = [NSJSONSerialization JSONObjectWithData:
                            [params dataUsingEncoding:NSUTF8StringEncoding] options:kNilOptions error:nil];

    if (resultDic!= nil) {
        NSArray *wordArray = [resultDic objectForKey:@"ws"];

        for (int i = 0; i < [wordArray count]; i++) {
            NSDictionary *wsDic = [wordArray objectAtIndex: i];
            NSArray *cwArray = [wsDic objectForKey:@"cw"];

            for (int j = 0; j < [cwArray count]; j++) {
                NSDictionary *wDic = [cwArray objectAtIndex:j];
                NSString *str = [wDic objectForKey:@"w"];
                [tempStr appendString: str];
            }
        }
    }
    return tempStr;
}
```

(NSString *) 0x174672ec0

```
{"sn":1,"ls":false,"bg":0,"ed":0,"ws":[{"bg":0,"cw":
[{"sc":0.00,"w":"直线"}]}]}
```

4. 语音合成(文字转语音)

   略。

5. 语义理解(语音听写基础上分析理解说话者意图)

   略。

6. 语法识别(命令词识别)

   略。

7. 声纹识别(声纹密码)

   略。

8. 语音评测(发音水平评测)

略。

9. 语音唤醒(手机锁屏也能识别)

略。

10. 人脸识别

略。

11. 身份验证(可声纹+人脸融合验证)

略。