语音识别

SDK文档

腾讯云

文档目录

SDK文档
在线 SDK 文档
SDK 概览

最近更新时间：2023-10-26 17:54:51

SDK 说明

腾讯云语音识别 ASR SDK 提供服务端、客户端、前端以及小程序 SDK ，给您提供了一种方便、快捷、灵活的方式，将语音识别功能集成到您的服务。
目前语音识别 SDK 支持的功能：

- 录音文件识别
- 实时语音识别
- 语音流异步识别
- 录音文件识别极速版
- 一句话识别

SDK 接入

类型	平台/语言	服务	SDK 集成说明
客户端	iOS	实时语音识别、一句话识别、录音文件识别极速版	一分钟跑通集成 SDK
	Android	实时语音识别、一句话识别、录音文件识别极速版	一分钟跑通集成 SDK
	Flutter	实时语音识别、一句话识别、录音文件识别极速版	一分钟跑通集成 SDK
小程序	小程序	实时语音识别、一句话识别	一分钟跑通集成 SDK
前端	JS	实时语音识别	Github
服务端	GO	实时语音识别、录音文件识别极速版	Github
		录音文件识别、语音流异步识别、一句话识别	Github
	JAVA	实时语音识别、录音文件识别极速版	Github
		录音文件识别、语音流异步识别、一句话识别	Github
	C++	实时语音识别	Github
		录音文件识别、语音流异步识别、一句话识别	Github
	Python	实时语音识别、录音文件识别极速版	Github
		录音文件识别、语音流异步识别、一句话识别	Github
	PHP	录音文件识别、语音流异步识别、一句话识别	Github
	Node.js	录音文件识别、语音流异步识别、一句话识别	Github
	C#	实时语音识别	Github

说明
录音文件识别、语音流异步识别、一句话识别可直接使用 开发者工具 自动生成多种语言（Python、 Java、PHP、Go、NodeJS、.NET、C++）SDK demo。

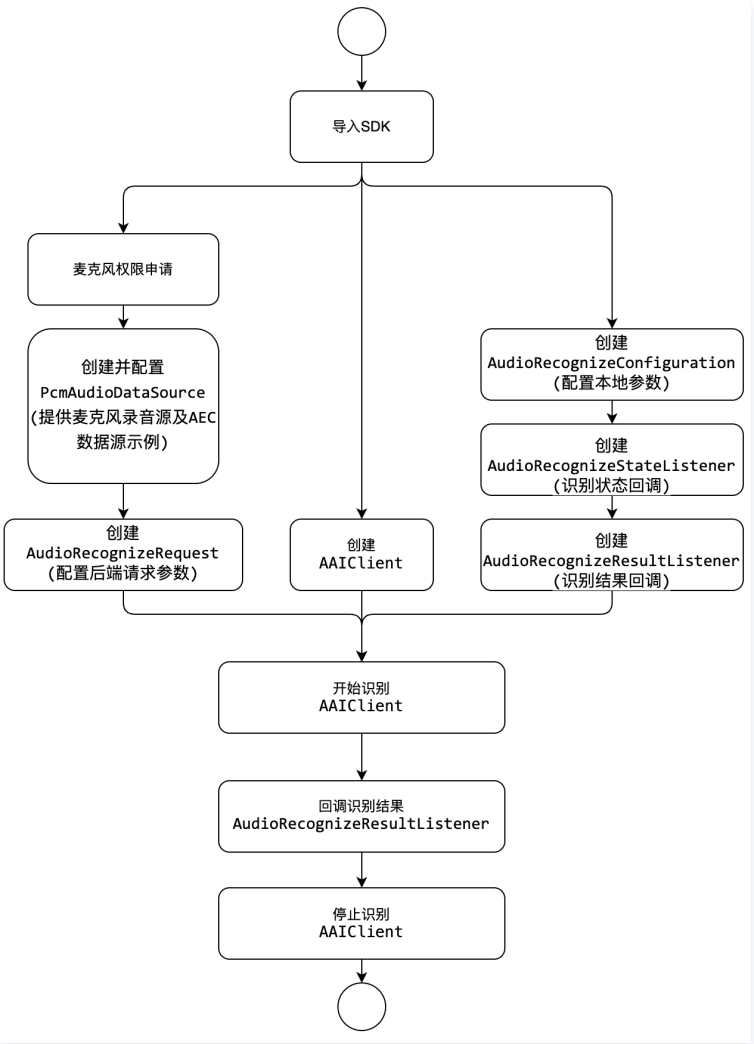
快速体验

目前腾讯云提供了小程序的语音识别体验，扫码即可体验语音识别小程序 SDK 能力。

客户端 SDK
Android
实时语音识别

最近更新时间：2025-06-11 14:20:32

1. 接入流程



2. 接入准备

2.1 SDK 获取

实时语音识别 Android SDK 及 Demo 下载地址：接入 SDK 下载 。

2.2 接入须知

- 开发者在调用前请先查看实时语音识别的 接口说明，了解接口的**使用要求**和**使用步骤**。
- 该接口需要手机能够连接网络（3G、4G、5G 或 Wi-Fi 等），且系统为 **Android 5.0** 及其以上版本。
- 运行 Demo 必须设置 AppID、SecretID、SecretKey，可在 API 密钥管理 中获取。

2.3 开发环境

- **添加实时语音识别 SDK aar**
  将 **asr-realtime-release.aar** 放在 libs 目录下，在 App 的 build.gradle 文件中添加以下代码。

```
implementation(name: 'asr-realtime-release', ext: 'aar')
```

- 添加其他依赖，在 App 的 build.gradle 文件中添加以下代码。

```
implementation 'com.squareup.okhttp3:okhttp:4.2.2'
```

- 在 AndroidManifest.xml 添加如下权限：

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

2.4 混淆规则

```
-keepclasseswithmembernames class * { # 保持 native 方法不被混淆
native <methods>;
}
-keep public class com.tencent.aai.*
-keep public class com.qq.wx.voice.*
```

3. 快速接入

3.1 启动实时语音识别

```
int appid = XXX;
int projectid = 0; //此参数固定为0;
String secretId = "XXX";

final AAIClient aaiClient;
try {
    /**直接鉴权**/
    // 1. 签名鉴权类，sdk中给出了一个本地的鉴权类，您也可以自行实现CredentialProvider接口，在您的服务器上实现鉴权签名
    aaiClient = new AAIClient(MainActivity.this, appid, projectId, secretId ,new
LocalCredentialProvider(secretKey));

    /** 使用临时密钥鉴权
     *  (1).通过sts 获取到临时证书 （secretId secretKey token） ,此步骤应在您的服务器端实现，见
https://cloud.tencent.com/document/product/598/33416
     *  (2).通过临时密钥调用接口
     */
    // aaiClient = new AAIClient(MainActivity.this, appid, projectId,"临时secretId", "临时secretKey","对应的
token");

    // 2、初始化语音识别请求。
    AudioRecognizeRequest.Builder builder = new AudioRecognizeRequest.Builder();
    final AudioRecognizeRequest audioRecognizeRequest = builder
        //设置数据源，数据源要求实现PcmAudioDataSource接口，您可以自己实现此接口来定制您的自定义数据源，例如从第三方推流
中获
        .pcmAudioDataSource(new AudioRecordDataSource(false)) // 使用SDK内置录音器作为数据源,false:不保存音频
        .setEngineModelType("16k_zh") // 设置引擎参数("16k_zh" 通用引擎，支持中文普通话+英文)
        .setFilterDirty(0)  // 0 ：默认状态 不过滤脏话 1：过滤脏话
        .setFilterModal(0) // 0 ：默认状态 不过滤语气词  1：过滤部分语气词 2:严格过滤
        .setFilterPunc(0) // 0 ：默认状态 不过滤句末的句号 1：滤句末的句号
        .setConvert_num_mode(1) //1：默认状态 根据场景智能转换为阿拉伯数字；0：全部转为中文数字。
        .setNeedvad(1) //0：关闭 vad，1：默认状态 开启 vad。语音时长超过一分钟需要开启,如果对实时性要求较高,并且时间较
短的输入,建议关闭
```

```
                    // .setHotWordId("")//热词 id。用于调用对应的热词表，如果在调用语音识别服务时，不进行单独的热词
id 设置，自动生效默认热词；如果进行了单独的热词 id 设置，那么将生效单独设置的热词 id。
                    //.setCustomizationId("")//自学习模型 id。如果设置了该参数，那么将生效对应的自学习模型
        .build();

    // 3、初始化语音识别结果监听器。
    final AudioRecognizeResultListener audioRecognizeResultlistener = new AudioRecognizeResultListener() {

        @Override
        public void onSliceSuccess(AudioRecognizeRequest request, AudioRecognizeResult result, int seq) {
            //返回分片的识别结果，此为中间态结果，会被持续修正
        }

        @Override
        public void onSegmentSuccess(AudioRecognizeRequest request, AudioRecognizeResult result, int seq)
{
            //返回语音流的识别结果，此为稳定态结果，可做为识别结果用与业务
        }

        @Override
        public void onSuccess(AudioRecognizeRequest request, String result) {
            //识别结束回调，返回所有的识别结果
        }

        @Override
        public void onFailure(AudioRecognizeRequest request, final ClientException clientException, final
ServerException serverException,String response) {
            // 识别失败
        }
     };

    // 4、自定义识别配置
    final AudioRecognizeConfiguration audioRecognizeConfiguration = new
AudioRecognizeConfiguration.Builder()
          //分片默认40ms，可设置40-5000，如果您不了解此参数不建议更改
          //.sliceTime(40)
          // 是否使能静音检测，
          .setSilentDetectTimeOut(false)
          // 静音检测超时停止录音可设置>2000ms，setSilentDetectTimeOut为true有效，超过指定时间没有说话将关闭识别；需要
大于等于sliceTime，实际时间为sliceTime的倍数，如果小于sliceTime，则按sliceTime的时间为准
          .audioFlowSilenceTimeOut(5000)
           // 音量回调时间，需要大于等于sliceTime，实际时间为sliceTime的倍数，如果小于sliceTime，则按sliceTime的时间
为准
          .minVolumeCallbackTime(80)
          .build();

    // 5、启动语音识别
    new Thread(new Runnable() {
        @Override
        public void run() {
            if (aaiClient!=null) {
                aaiClient.startAudioRecognize(audioRecognizeRequest,
                                              audioRecognizeResultlistener,
                                              audioRecognizeStateListener,
                                              audioRecognizeConfiguration);
            }
        }
    }).start();
    // 6、log组件设置
    // 将log落盘到本地磁盘，needLogFile字段默认为false，接入调试期间建议设置为true，上线后此接口调用可删除。
    AAILogger.setNeedLogFile(true, getApplicationContext());
```

```
    // 设置日志级别，默认为ERROR_LEVEL，接入调试期间建议设置为DEBUG_LEVEL。
    AAILogger.setLogLevel(AAILogger.DEBUG_LEVEL);
    // 设置日志监听器，用于监听日志信息。
    AAILogger.setLoggerListener(new LoggerListener() {
      @Override
      public void onLogInfo(String s) {

       }
    });
} catch (ClientException e) {
    e.printStackTrace();
}
```

3.2 停止实时语音识别

```
new Thread(new Runnable() {
    @Override
    public void run() {
        if (aaiClient!=null){
            //停止语音识别，等待最终识别结果
            aaiClient.stopAudioRecognize();
        }
    }
}).start();
```

3.3 取消实时语音识别

```
new Thread(new Runnable() {
    @Override
    public void run() {
        if (aaiClient!=null){
            //取消语音识别，丢弃当前任务，丢弃最终结果
            aaiClient.cancelAudioRecognize();
        }
    }
}).start();
```

4. 主要接口类和方法说明

4.1 初始化 AAIClient

AAIClient 是语音服务的核心类，用户可以调用该类来开始、停止以及取消语音识别。

```
public AAIClient(Context context, int appid, int projectId, String secretId, AbsCredentialProvider
credentialProvider) throws ClientException
```

参数名称	类型	是否必填	参数描述
context	Context	是	上下文
appid	Int	是	腾讯云注册的 AppID
projectId	Int	否	此参数固定为0
secretId	String	是	用户的 secretId
credentialProvider	AbsCredentialProvider	是	鉴权类

示例：

```
try {
    AAIClient aaiClient = new AAIClient(context, appid, projectId, secretId, credentialProvider);
} catch (ClientException e) {
    e.printStackTrace();
}
```

如果 aaiClient 不再需要使用，请调用 release() 方法释放资源：

```
aaiClient.release();
```

4.2 配置全局参数

用户调用 ClientConfiguration 类的静态方法来修改全局配置。

方法	方法描述	默认值	有效范围
setAudioRecognizeSliceTimeout	HTTP 读超时时间	5000ms	500 – 10000ms
setAudioRecognizeConnectTimeout	HTTP 连接超时时间	5000ms	500 – 10000ms
setAudioRecognizeWriteTimeout	HTTP 写超时时间	5000ms	500 – 10000ms

示例：

```
ClientConfiguration.setAudioRecognizeSliceTimeout(2000)
ClientConfiguration.setAudioRecognizeConnectTimeout(2000)
ClientConfiguration.setAudioRecognizeWriteTimeout(2000)
```

4.3 设置结果监听器

AudioRecognizeResultListener 可以用来监听语音识别的结果，共有如下四个接口：

- 语音分片的语音识别结果回调接口

```
void onSliceSuccess(AudioRecognizeRequest request, AudioRecognizeResult result, int order);
```

参数	参数类型	参数描述
request	AudioRecognizeRequest	语音识别请求
result	AudioRecognizeResult	语音分片的语音识别结果
order	Int	该语音分片所在语音流的次序

- 语音流的语音识别结果回调接口

```
void onSegmentSuccess(AudioRecognizeRequest request, AudioRecognizeResult result, int seq);
```

参数	参数类型	参数描述
request	AudioRecognizeRequest	语音识别请求
result	AudioRecognizeResult	语音分片的语音识别结果
seq	Int	该语音流的次序

- 返回所有的识别结果

```
void onSuccess(AudioRecognizeRequest request, String result);
```

参数	参数类型	参数描述
request	AudioRecognizeRequest	语音识别请求
result	String	所有的识别结果

- 语音识别请求失败回调函数

```
void onFailure(AudioRecognizeRequest request, final ClientException clientException, final
ServerException serverException,String response);
```

参数	参数类型	参数描述
request	AudioRecognizeRequest	语音识别请求
clientException	ClientException	客户端异常
serverException	ServerException	服务端异常
response	String	服务端返回的 json 字符串

示例代码详见 入门示例 。

4.4 设置语音识别参数

通过构建 AudioRecognizeConfiguration 类，可以设置语音识别时的配置：

参数名称	类型	是否必填	参数描述	默认值
setSilentDetectTimeOut	Boolean	否	是否开启静音检测，开启后检测到超时不说话将停止识别	false
audioFlowSilenceTimeOut	Int	否	配置 setSilentDetectTimeOut 时间超时时间	5000ms
minVolumeCallbackTime	Int	否	音量检测回调时间	80ms

示例：

```
AudioRecognizeConfiguration audioRecognizeConfiguration = new AudioRecognizeConfiguration.Builder()
    .setSilentDetectTimeOut(true)// 是否开启静音检测，开启后检测到超时不说话将停止识别
    .audioFlowSilenceTimeOut(5000) // 静音检测超时停止录音
    .minVolumeCallbackTime(80) // 音量回调时间
    .build();
```

4.5 设置状态监听器

AudioRecognizeStateListener 可以用来监听语音识别的状态：

方法	方法描述
onStartRecord	开始录音
onStopRecord	结束录音
onVoiceDb	音量分贝（取值范围：0~100，集中分布在40~80）
onNextAudioData	返回音频流，用于返回宿主层做录音缓存业务。new AudioRecordDataSource(true) 传递 true 时生效
onSilentDetectTimeOut	静音检测超时回调，此时任务还未中止，仍会等待最终识别结果

示例：

```
AudioRecognizeStateListener audioRecognizeStateListener = new AudioRecognizeStateListener() {
    @Override
    public void onStartRecord(AudioRecognizeRequest audioRecognizeRequest) {
        // 开始录音
    }
    @Override
    public void onStopRecord(AudioRecognizeRequest audioRecognizeRequest) {
        // 结束录音
    }

    @Override
    public void onVoiceVolume(AudioRecognizeRequest audioRecognizeRequest, int i) {
        // 音量回调
    }

    /**
      * 返回音频流，
      * 用于返回宿主层做录音缓存业务。
      * 由于方法跑在sdk线程上，这里多用于文件操作，宿主需要新开一条线程专门用于实现业务逻辑
      * new AudioRecordDataSource(true) 有效，否则不会回调该函数
      * @param audioDatas
      */
    @Override
    public void onNextAudioData(final short[] audioDatas, final int readBufferLength){
    }

     /**
      * 静音检测超时回调
      * 注意：此时任务还未中止，仍然会等待最终识别结果
      */
     @Override
    void onSilentDetectTimeOut(){
        //触发了静音检测事件
    }
};
```

4.6 其他重要类说明

4.6.1 AudioRecognizeRequest

参数名称	类型	是否必填	参数描述	默认值
pcmAudioDataSource	PcmAudioDataSource	是	音频数据源	无
setEngineModelType	String	否	设置引擎参数	"16k_zh"
setFilterDirty	int	否	0：不过滤脏话 1：过滤脏话	0
setFilterModal	int	否	0：不过滤语气词 1：过滤部分语气词 2:严格过滤	0
setFilterPunc	int	否	0：不过滤句末的句号 1：滤句末的句号	0
setConvert_num_mode	int	否	1：根据场景智能转换为阿拉伯数字；0：全部转为中文数字。	1

setVadSilenceTime	int	否	语音断句检测阈值，静音时长超过该阈值会被认为断句（需配合 needvad = 1 使用）默认不传递该参数，不建议更改	无
setNeedvad	int	否	0：关闭 vad，1： 开启 vad。语音时长超过一分钟需要开启,如果对实时性要求较高。	1
setHotWordId	String	否	热词 id。用于调用对应的热词表，如果在调用语音识别服务时，不进行单独的热词 id 设置，自动生效默认热词；如果进行了单独的热词 id 设置，那么将生效单独设置的热词 id。	无
setWordInfo	int	否	是否显示词级别时间戳。0：不显示；1：显示，不包含标点时间戳，2：显示，包含标点时间戳。时间戳信息需要自行解析 AudioRecognizeResult.resultJson 获取	0
setCustomizationId	String	否	自学习模型 id。如不设置该参数，自动生效最后一次上线的自学习模型；如果设置了该参数，那么将生效对应的自学习模型。	无
setNoiseThreshold	float	否	噪音参数阈值，默认为0，取值范围：[-1,1],详情见API文档	无
setMaxSpeakTime	int	否	强制断句功能，取值范围 5000-90000（单位:毫秒），在连续说话不间断情况下，该参数将实现强制断句。	默认值0(不开启)
setApiParam	Object	否	自定义请求参数,用于在请求中添加SDK尚未支持的参数	无

4.6.2 AudioRecognizeResult

语音识别结果对象，和 AudioRecognizeRequest 对象相对应，用于返回语音识别的结果。

参数名称	类型	参数描述
sliceType	Int	0表示一小段话开始，1表示在小段话的进行中，2表示小段话的结束
message	String	识别提示信息
text	String	识别结果
seq	Int	当前一段话结果在整个音频流中的序号，从0开始逐句递增
voiceId	String	该语音分片所在语音流的 ID
startTime	int	当前一段话结果在整个音频流中的起始时间
endTime	int	当前一段话结果在整个音频流中的结束时间
resultJson	String	后端返回的 json 原文本,可解析出上面列出的参数内容，如有需求，您可以自行解析获取更多信息

4.6.3 PcmAudioDataSource

用户可以实现这个接口来识别单通道、采样率16k的 PCM 音频数据。主要包括如下几个接口：

- 向语音识别器添加数据，将长度为 length 的数据从下标0开始复制到 audioPcmData 数组中，并返回实际的复制的数据量的长度。

```
int read(short[] audioPcmData, int length);
```

- 启动识别时回调函数，用户可以在这里做些初始化的工作。

```
void start() throws AudioRecognizerException;
```

- 结束识别时回调函数，用户可以在这里进行一些清理工作。

```
void stop();
```

- 是否保存语音源文件的开关，打开后，音频数据将通过 onNextAudioData 回调返回给调用层。

```
boolean  isSetSaveAudioRecordFiles();
```

4.6.4 AudioRecordDataSource

PcmAudioDataSource 接口的实现类，可以直接读取麦克风输入的音频数据，用于实时识别，其中 demo 也提供了一份录音器源码作为数据源的示例，源码与 SDK 内置录音器 AudioRecordDataSource 一致，您可以参考此源代码自由定制修改，详情查阅 SDK 包内 DemoAudioRecordDataSource.java 内注释。

4.6.5 AAILogger

用户可以利用 AAILogger 来控制日志的输出，可以选择性的输出 debug、info、warn 以及 error 级别的日志信息。

```
public static void disableDebug();
public static void disableInfo();
public static void disableWarn();
public static void disableError();
public static void enableDebug();
public static void enableInfo();
public static void enableWarn();
public static void enableError();
```

5. 错误码

- 后端错误码，详情请参见 API 文档 。
- 客户端错误码如下：

错误码	名称	描述
-100	AUDIO_RECORD_INIT_FAILED	录音器初始化失败
-101	AUDIO_RECORD_START_FAILED	录音器启动失败
-102	AUDIO_RECORD_MULTIPLE_START	录音器重复启动
-103	AUDIO_RECOGNIZE_THREAD_START_FAILED	创建线程失败，录音线程无法启动
-104	AUDIO_SOURCE_DATA_NULL	数据源为空
-105	AUDIO_RECOGNIZE_REQUEST_NULL	请求参数为空
-106	WEBSOCKET_NETWORK_FAILED	websocket 网络连接失败
-1	UNKNOWN_ERROR	未知异常，详见 message 信息

6. 常见问题指引

6.1 音频数据本地缓存指引

宿主层可根据自身业务需求选择将音频保存到本地或者不保存。若需要保存到本地可按照如下步骤进行操作：

1. `new AudioRecordDataSource(isSaveAudioRecordFiles)` 初始化时，`isSaveAudioRecordFiles` 设置为 true。
2. `AudioRecognizeStateListener.onStartRecord` 回调函数内添加创建本次录音的文件逻辑。路径、文件名可支持自定义。
3. `AudioRecognizeStateListener.onStopRecord` 回调函数内添加关流逻辑。（可选）将 PCM 文件转存为 WAV 文件。
4. `AudioRecognizeStateListener.onNextAudioData` 回调函数内添加将音频流写入本地文件的逻辑。
5. 由于回调函数均跑在 sdk 线程中。为了避免写入业务耗时问题影响 sdk 内部运行流畅度，建议将上述步骤放在单独线程池里完成，详情见 Demo 工程中的 `MainActivity` 类中的示例代码。

6.2 回音消除指引

本小节主要介绍如何通过Android原生API实现回音消除，下面将分章节详细展开（详情可参见Demo工程中的DemoAudioRecordDataSource类里的start方法）。

6.2.1 设置音源的方式

```
/**
 * 注：部分android机型可以通过该方式解决回音消除失效的问题
 * https://blog.csdn.net/wyw0000/article/details/125195997
 */
// 1. 设置音频模式为AudioManager.MODE_IN_COMMUNICATION可以起到回音消除的作用
AudioManager audioManager = (AudioManager)context.getSystemService(Context.AUDIO_SERVICE);
audioManager.setMode(AudioManager.MODE_IN_COMMUNICATION);

// 2. 音频源使用MediaRecorder.AudioSource.VOICE_COMMUNICATION可以起到回音消除的作用
int audioSource = MediaRecorder.AudioSource.VOICE_COMMUNICATION;
AudioRecord audioRecord = new AudioRecord(audioSource, sampleRate, channel, audioFormat, bufferSize);
```

6.2.2 尝试开启AEC和噪音抑制

```
/**
 * 注：以下两个能力(AcousticEchoCanceler和NoiseSuppressor)和手机硬件能力相关，有些机型(比如小米11)即使
isAvailable()==true，回音消除也不生效
 */
// AcousticEchoCanceler回音消除
if (AcousticEchoCanceler.isAvailable()) {
    Log.d(TAG, "AcousticEchoCanceler isAvailable.");
    AcousticEchoCanceler acousticEchoCanceler = AcousticEchoCanceler
            .create(audioRecord.getAudioSessionId());
    int resultCode = acousticEchoCanceler.setEnabled(true);
    if (AudioEffect.SUCCESS == resultCode) {
        Log.d(TAG, "AcousticEchoCanceler AudioEffect SUCCESS");
    }
}

// NoiseSuppressor噪音抑制
if (NoiseSuppressor.isAvailable()) {
    Log.d(TAG, "NoiseSuppressor isAvailable.");
    NoiseSuppressor noiseSuppressor = NoiseSuppressor
            .create(audioRecord.getAudioSessionId());
    int resultCode = noiseSuppressor.setEnabled(true);
    if (AudioEffect.SUCCESS == resultCode) {
        Log.d(TAG, "NoiseSuppressor AudioEffect SUCCESS");
    }
}
```

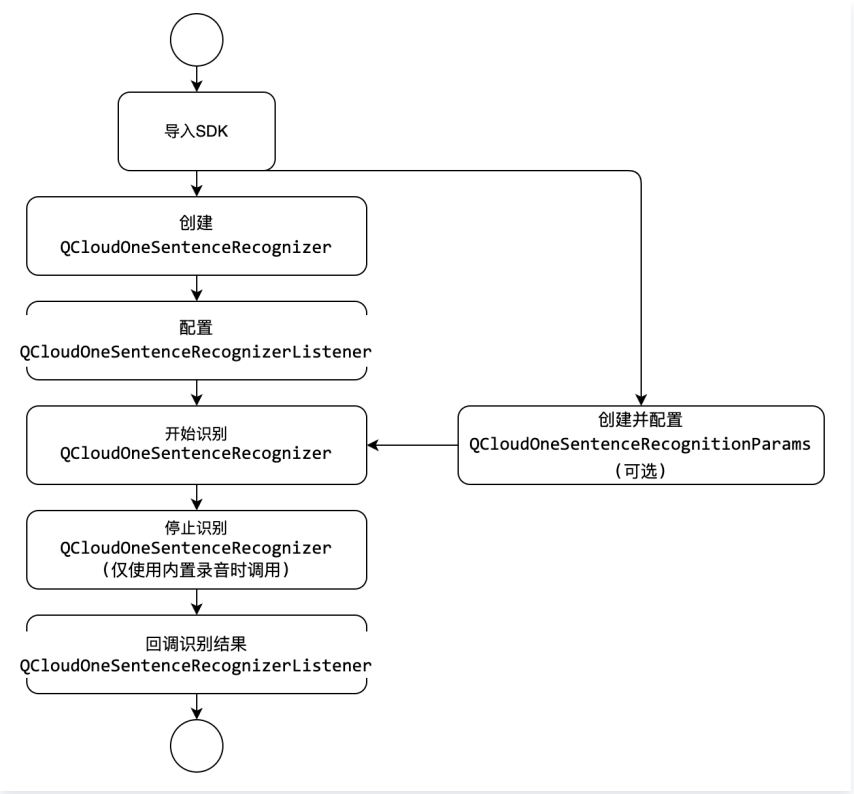
6.2.3 回音消除方案适配的机型列表

回声消除适配还会受到机型及系统的影响，SDK包内的文档列举了已测试机型和系统的适配情况，可前往控制台 下载SDK 。

一句话识别

最近更新时间：2025-06-11 14:20:32

接入流程



接入准备

SDK 获取

一句话识别 Android SDK 及 Demo 下载地址：接入 SDK 下载 。

接入须知

- 开发者在调用前请先查看一句话识别的 接口说明 ，了解接口的使用要求和使用步骤。
- 该接口需要手机能够连接网络（3G、4G、5G 或 Wi-Fi 等），且系统为 Android 5.0 及其以上版本。
- 运行 Demo 必须设置 AppID、SecretID、SecretKey，可在 API 密钥管理 中获取。

开发环境

1. **添加一句话识别 SDK aar**
   将 asr-one-sentence-release.aar 放在 libs 目录下，在 App 的 build.gradle 文件中添加以下代码。

```
implementation(name: 'asr-one-sentence-release', ext: 'aar')
```

2. **添加其他依赖，在 App 的 build.gradle 文件中添加以下代码。**

```
implementation 'com.google.code.gson:gson:2.8.5'
```

3. **在 AndroidManifest.xml 添加如下权限：**

```
< uses-permission android:name="android.permission.RECORD_AUDIO"/>
```

```
< uses-permission android:name="android.permission.INTERNET"/>
```

混淆规则

```
-keepclasseswithmembernames class * { # 保持 native 方法不被混淆
native <methods>;
}
-keep public class com.tencent.cloud.qcloudasrsdk.*
```

快速接入

开发流程及接入示例

1. **创建 QCloudOneSentenceRecognizer 示例**

```
QCloudOneSentenceRecognizer recognizer = new QCloudOneSentenceRecognizer(this, appid, secretId,
secretKey);
```

2. **设置识别结果回调**

```
recognizer.setCallback(this);
```

3. **调用示例**

- **通过语音 URL 调用**

```
QCloudOneSentenceRecognitionParams params =
(QCloudOneSentenceRecognitionParams)QCloudOneSentenceRecognitionParams.defaultRequestParams();

params.setSourceType(QCloudSourceType.QCloudSourceTypeUrl); //调用方式:URL
params.setUrl("http://liqiansunvoice-1255628450.cosgz.myqcloud.com/test.wav"); // 设置音频文件的URL下载地址
(请替换为您自己的地址)
params.setVoiceFormat("wav"); //设置音频文件格式，支持wav、pcm、ogg-opus、speex、silk、mp3、m4a、aac。

params.setFilterDirty(0);// 0 ：默认状态 不过滤脏话 1：过滤脏话
params.setFilterModal(0);// 0 ：默认状态 不过滤语气词  1：过滤部分语气词 2:严格过滤
params.setFilterPunc(0); // 0 ：默认状态 不过滤句末的句号 1：过滤句末的句号
params.setConvertNumMode(1);//1：默认状态 根据场景智能转换为阿拉伯数字；0：全部转为中文数字。
// 热词id。用于调用对应的热词表，如果在调用语音识别服务时，不进行单独的热词id设置，自动生效默认热词；如果进行了单独的热词
id设置，那么将生效单独设置的热词id。
//params.setHotwordId("****************************************");

//设置识别引擎，默认16k_zh,见 https://cloud.tencent.com/document/product/1093/35646
params.setEngSerViceType("16k_zh");

recognizer.recognize(params);
```

- **通过语音数据调用**

```
AssetManager am = getResources().getAssets();
is = am.open("test1.mp3");
int length = is.available();
byte[] audioData = new byte[length];
is.read(audioData);

//配置识别参数,详细参数说明见： https://cloud.tencent.com/document/product/1093/35646
```

```
QCloudOneSentenceRecognitionParams  params =
(QCloudOneSentenceRecognitionParams)QCloudOneSentenceRecognitionParams.defaultRequestParams();

params.setSourceType(QCloudSourceType.QCloudSourceTypeData); //调用方式:通过语音数据调用
params.setData(audioData);
params.setVoiceFormat("mp3");//识别音频的音频格式，支持wav、pcm、ogg-opus、speex、silk、mp3、m4a、aac。

params.setFilterDirty(0);// 0 ：默认状态 不过滤脏话 1：过滤脏话
params.setFilterModal(0);// 0 ：默认状态 不过滤语气词  1：过滤部分语气词 2:严格过滤
params.setFilterPunc(0); // 0 ：默认状态 不过滤句末的句号 1：滤句末的句号
params.setConvertNumMode(1);//1：默认状态 根据场景智能转换为阿拉伯数字；0：全部转为中文数字。
// 热词id。用于调用对应的热词表，如果在调用语音识别服务时，不进行单独的热词id设置，自动生效默认热词；如果进行了单独的热词
id设置，那么将生效单独设置的热词id。
//params.setHotwordId("");
//默认16k_zh，更多引擎参数详见https://cloud.tencent.com/document/product/1093/35646 内的EngSerViceType字段
params.setEngSerViceType("16k_zh");

recognizer.recognize(params);
```

- **通过 SDK 内置录音器并识别**

```
/**
* setDefaultParams 默认参数param
* @param filterDirty    0 ：默认状态 不过滤脏话 1：过滤脏话
* @param filterModal    0 ：默认状态 不过滤语气词  1：过滤部分语气词 2:严格过滤
* @param filterPunc     0 ：默认状态 不过滤句末的句号 1：滤句末的句号
* @param convertNumMode 1：默认状态 根据场景智能转换为阿拉伯数字；0：全部转为中文数字。
* @param hotwordId  热词id，不使用则传null
* @param engSerViceType  引擎模型类型，传null默认使用“16k_zh”
*/
recognizer.setDefaultParams(filterDirty, filterModal, filterPunc,
convertNumMode,hotwordId,engSerViceType);
recognizer.recognizeWithRecorder();
```

4. SDK Log 组件设置说明

```
log组件设置
// 将log落盘到本地磁盘，needLogFile字段默认为false，接入调试期间建议设置为true，上线后此接口调用可删除。
AAILogger.setNeedLogFile(true, getApplicationContext());
// 设置日志级别，默认为ERROR_LEVEL，接入调试期间建议设置为DEBUG_LEVEL。
AAILogger.setLogLevel(AAILogger.DEBUG_LEVEL);
// 设置日志监听器，用于监听日志信息。
AAILogger.setLoggerListener(new LoggerListener() {
    @Override
    public void onLogInfo(String s) {
    }
});
```

关键类说明

QCloudOneSentenceRecognizer：一句话识别入口类

```
/**
 * 初始化方法-直接鉴权，关于 AppId, SecretId, SecretKey 的获取见一句话识别接口说明中的使用步骤
 * @param activity app activity
 * @param appId 腾讯云appid
```

```
 * @param secretId 腾讯云secretId
 * @param secretKey 腾讯云secretKey
 */
public QCloudOneSentenceRecognizer(AppCompatActivity activity, String appId, String secretId, String
secretKey);


/**
 * 初始化方法-使用STS临时证书鉴权，详见https://cloud.tencent.com/document/product/598/33416
 * @param activity app activity
 * @param appId 腾讯云appid
 * @param secretId 腾讯云 临时的secretId
 * @param secretKey 腾讯云 临时的secretKey
 * @param token 腾讯云 token
 */
public QCloudOneSentenceRecognizer(Activity activity, String appId, String secretId, String secretKey,
String token);

 /**
  * 通过语音url进行一句话识别的快捷入口， 本地参数校验不通过抛出异常
  * @param audioUrl 资源url 如http://www.qq.music/hello.mp3
  * @param audioFormat 语音数据格式，QCloudAudioFormat
  * @param frequence 引擎模型类型，QCloudAudioFrequence枚举类获取对应模型名称，也可直接传符串，此参数与API文档
EngSerViceType对应
  */
 public void recognize(String audioUrl, QCloudAudioFormat audioFormat, String frequence) throws Exception;

/**
  * 通过语音数据进行一句话识别的快捷入口， 本地参数校验不通过抛出异常
  * @param audioData 语音数据
  * @param audioFormat 语音数据格式，QCloudAudioFormat
  * @param frequence 语音数据采样率，QCloudAudioFrequence
  */
 public void recognize(byte[] audioData, QCloudAudioFormat audioFormat, QCloudAudioFrequence frequence)
throws Exception;
/**
 * 通过QCloudOneSentenceRecognitionParams调用一句话识别， 调用[QCloudCommonParams defaultRequestParams]方法获取
默认参数，
 * 然后根据需求设置参数
 * @param params请求参数
 */
public void recognize(QCloudOneSentenceRecognitionParams params) throws Exception;
/**
 * 通过sdk内置录音器开启一句话识别
 */
public void recognizeWithRecorder() throws Exception;
```

QCloudOneSentenceRecognizerListener：开始录音、结束录音以及识别结果回调。

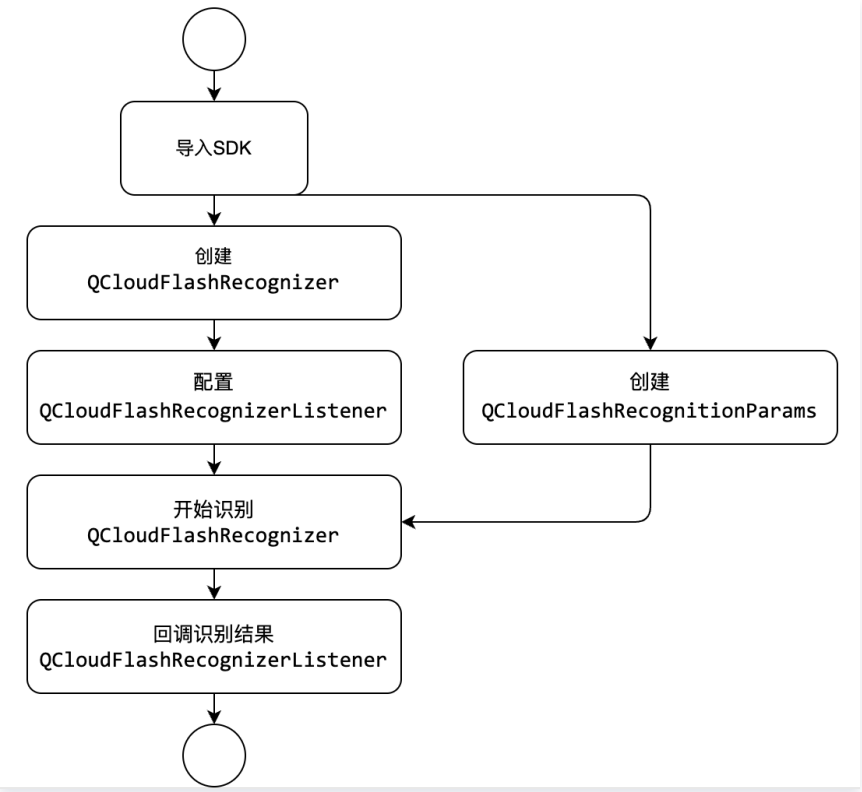
```
public interface QCloudOneSentenceRecognizerListener {
 /**
  * 开始录音回调
  */
 public abstract void didStartRecord();
 /**
  * 结束录音回调
  */
 public abstract void didStopRecord();
 /**
  * 识别结果回调
```

```
   */
  public abstract void recognizeResult(QCloudOneSentenceRecognizer recognizer, String result, Exception
exception);
}
```

录音文件识别极速版

最近更新时间：2025-06-11 14:20:32

接入流程



开发准备

SDK 下载

录音文件识别 Android SDK 及 Demo 下载地址：接入 SDK 下载 。

接入须知

- 开发者在调用前请先查看录音文件识别极速版的 接口说明，了解接口的**使用要求**和**使用步骤**。
- 该接口需要手机能够连接网络（3G、4G、5G 或 Wi-Fi 等），且系统为 **Android 5.0** 及其以上版本。
- 运行 Demo 必须设置 AppID、SecretID、SecretKey，可在 API 密钥管理 中获取。

运行环境配置

1. 添加录音文件识别 SDK aar，将 **asr-file-recognize-release.aar** 放在 libs 目录下，在 App 的 build.gradle 文件中添加。

```
implementation(name: 'asr-file-recognize-release', ext: 'aar')
```

2. 添加其他依赖，在 App 的 build.gradle 文件中添加。

```
implementation 'com.google.code.gson:gson:2.8.5'
```

3. 在 AndroidManifest.xml 添加如下权限。

```
< uses-permission android:name="android.permission.INTERNET"/>
```

混淆规则

```
-keepclasseswithmembernames class * { # 保持 native 方法不被混淆
native <methods>;
}
-keep public class com.tencent.cloud.qcloudasrsdk.*
```

快速接入

开发流程及接入示例

1. 创建 QCloudFileRecognizer 示例

```
QCloudFlashRecognizer fileFlashRecognizer = new QCloudFlashRecognizer(this, appid, secretId,
secretKey);

/**
也可以使用临时密钥鉴权
1.通过sts 获取到临时证书 （secretId secretKey  token） ,此步骤应在您的服务器端实现，见
https://cloud.tencent.com/document/product/598/33416
2.通过临时密钥调用接口
**/
QCloudFlashRecognizer fileFlashRecognizer = new QCloudFlashRecognizer(DemoConfig.appId, "临时secretId",
"临时secretKey","对应的token");
```

2. 设置识别结果回调

```
fileFlashRecognizer.setCallback(this);
```

3. 调用方式示例

```
  InputStream is = null;
  AssetManager am = getResources().getAssets();
  is = am.open("test1.mp3");
  int length = is.available();
  byte[] audioData = new byte[length];
  is.read(audioData);

  QCloudFlashRecognitionParams params = (QCloudFlashRecognitionParams)
  QCloudFlashRecognitionParams.defaultRequestParams();
//支持传音频文件数据或者音频文件路径，如果同时调用setData和setPath，sdk内将忽略setPath的值
 params.setData(audioData);
//  params.setPath("/sdcard/test2.mp3"); //支持100MB以内音频文件的识别
 params.setVoiceFormat("mp3"); //音频格式。支持 wav、pcm、ogg-opus、speex、silk、mp3、m4a、aac。

/**以下参数不设置将使用默认值**/
 params.setEngineModelType("16k_zh");//引擎模型类型，默认16k_zh。8k_zh：8k 中文普通话通用；16k_zh：16k 中文普通
话通用；16k_zh_video：16k 音视频领域。
 params.setFilterDirty(0);// 0 ：默认状态 不过滤脏话 1：过滤脏话
 params.setFilterModal(0);// 0 ：默认状态 不过滤语气词  1：过滤部分语气词 2:严格过滤
 params.setFilterPunc(0);// 0 ：默认状态 不过滤句末的句号 1：滤句末的句号
 params.setConvertNumMode(1);//1：默认状态 根据场景智能转换为阿拉伯数字；0：全部转为中文数字。
 params.setSpeakerDiarization(0); //是否开启说话人分离（目前支持中文普通话引擎），默认为0，0：不开启，1：开启。
 params.setFirstChannelOnly(1); //是否只识别首个声道，默认为1。0：识别所有声道；1：识别首个声道。
 params.setWordInfo(0); //是否显示词级别时间戳，默认为0。0：不显示；1：显示，不包含标点时间戳，2：显示，包含标点时间
戳。
 params.setCustomizationID(""); //自学习模型 id。如设置了该参数，将生效对应的自学习模型。
 params.setHotwordID(""); //热词表 id。如不设置该参数，自动生效默认热词表；如设置了该参数，那么将生效对应的热词表。
```

```
 fileFlashRecognizer.recognize(params);

// log组件设置
// 将log落盘到本地磁盘，needLogFile字段默认为false，接入调试期间建议设置为true，上线后此接口调用可删除。
AAILogger.setNeedLogFile(true, getApplicationContext());
// 设置日志级别，默认为ERROR_LEVEL，接入调试期间建议设置为DEBUG_LEVEL。
AAILogger.setLogLevel(AAILogger.DEBUG_LEVEL);
// 设置日志监听器，用于监听日志信息。
AAILogger.setLoggerListener(new LoggerListener() {
    @Override
    public void onLogInfo(String s) {
    }
});
```

关键类说明

QCloudFlashRecognizer：录音文件识别入口类

```
/**
 * 初始化方法
 * @param activity app activity
 * @param appId 腾讯云 appid
 * @param secretId 腾讯云 secretId
 * @param secretKey 腾讯云 secretKey
 */
public QCloudFlashRecognizer(String appId, String secretId, String secretKey);

 * 通过 url 或语音数据调用录音文件识别
 * @param params 请求参数
 * @return 返回本次请求的唯一标识别 requestId
 */
public long recognize(QCloudFlashRecognitionParams params) throws Exception;
```

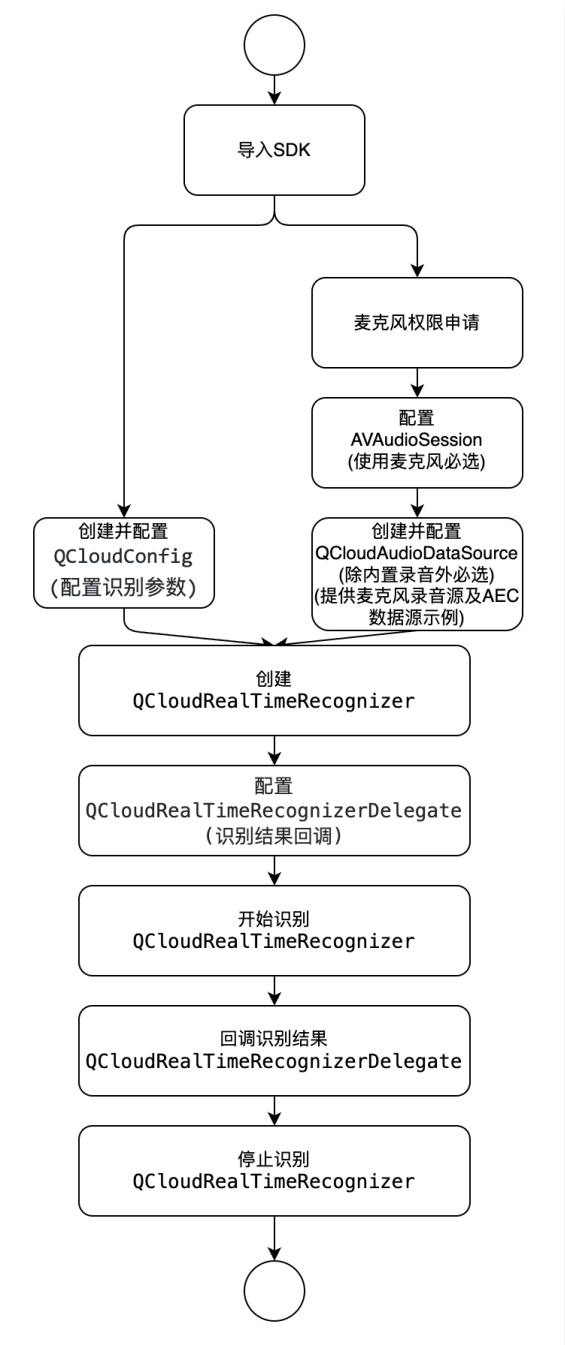
QCloudFlashRecognizerListener：识别结果回调

```
public interface QCloudFlashRecognizerListener {
    /**
     * 识别结果回调
     * @param recognizer 录音文件识别实例
     * @param result 服务器返回的识别结果 api文档 https://cloud.tencent.com/document/product/1093/52097
     * @param exception 异常信息
     *
     */
    void recognizeResult(QCloudFlashFileRecognizer recognizer,String result, int status,Exception
exception);
}
```

iOS
实时语音识别

最近更新时间：2025-01-02 14:18:42

1. 接入流程



2. 接入准备

2.1 SDK 获取

实时语音识别的 iOS SDK 以及 Demo 的下载地址：接入 SDK 下载 。

2.2 接入须知

- 开发者在调用前请先查看实时语音识别的 接口说明 ，了解接口的**使用要求**和**使用步骤**。

- 该接口需要手机能够连接网络（3G、4G、5G 或 Wi-Fi 网络等），且系统为 iOS 9.0 及以上版本。
- 运行 Demo 必须设置 AppID、SecretID、SecretKey，可在 API 密钥管理 中获取。

2.3 SDK 导入

2.3.1 直接导入

1. 下载并解压 iOS SDK 压缩包，压缩包中包含 Demo 和 SDK，其中 QCloudRealTime.xcframework 为实时语音识别 framework 包。
2. XcodeFile > Add Files to "Your Project"，在弹出 Panel 选中所下载 SDK 包 QCloudRealTime.xcframework > Add（选中“Copy items if needed”）。

2.3.2 CocoaPods 导入

使用 CocoaPods 导入时,需将以下内容添加到 Podfile 中。

```
pod 'QCloudRealTime'
```

2.4 工程配置

在工程 `info.plist` 申请系统麦克风权限，添加如下内容：

```
    <key>NSMicrophoneUsageDescription</key>
    <string>需要使用您的麦克风采集音频</string>
```

在工程中添加依赖库，在 build Phases Link Binary With Libraries 中添加以下库：

- QCloudRealTime.xcframework
- libc++.tbd
- AVFoundation.framework
- AudioToolbox.framework

3. 快速接入

下面分别介绍使用内置录音器采集语音识别和调用者提供语音数据接入流程和示例。

3.1 使用内置录音器采集语音识别示例

1. 引入SDK 的头文件：

```
#import <QCloudRealTime/QCloudRealTimeRecognizer.h>
#import <QCloudRealTime/QCloudConfig.h>
#import <QCloudRealTime/QCloudRealTimeResult.h>
#import <QCloudRealTime/QCloudAudioDataSource.h>
```

2. 创建 QCloudConfig 实例：

```
 //1.创建 QCloudConfig 实例
 QCloudConfig *config = [[QCloudConfig alloc] initWithAppId:kQDAppId
                              secretId:kQDSecretId
                             secretKey:kQDSecretKey
                             projectId:0];

//以下为可选配置参数
 config.requestTimeout = 10;     //请求超时时间（秒）
 //config.sliceTime = 40;            //语音分片时长默认40ms（无特殊需求不建议更改）
 config.enableDetectVolume = YES;      //是否检测音量
 config.endRecognizeWhenDetectSilence = YES;     //是否检测到静音停止识别
 config.shouldSaveAsFile = YES; //仅限使用SDK内置录音器有效，是否保存录音文件到本地 默认关闭
 config.saveFilePath = [NSTemporaryDirectory() stringByAppendingPathComponent:@"recordaudio.wav"]; //开
启shouldSaveAsFile后音频保存的路径，仅限使用SDK内置录音器有效，默认路径为[NSTemporaryDirectory()
```

```
stringByAppendingPathComponent:@"recordaudio.wav"]

//以下为API参数配置，参数描述见API文档：https://cloud.tencent.com/document/product/1093/48982
 config.engineType = @"16k_zh";//设置引擎，不设置默认16k_zh
 config.filterDirty = 0;        //是否过滤脏词，具体的取值见API文档的filter_dirty参数
 config.filterModal = 0;        //过滤语气词具体的取值见API文档的filter_modal参数
 config.filterPunc = 0;         //过滤句末的句号具体的取值见API文档的filter_punc参数
 config.convertNumMode = 1;     //是否进行阿拉伯数字智能转换。具体的取值见API文档的convert_num_mode参数
 //config.hotwordId = @"";          //热词id。具体的取值见API文档的hotword_id参数
 //config.customizationId = @"";   //自学习模型id,详情见API文档
 //config.vadSilenceTime = -1;     //语音断句检测阈值,详情见API文档
 config.needvad = 1;    //默认1 0：关闭 vad，1：开启 vad。 如果语音分片长度超过60秒，用户需开启 vad。
 config.wordInfo = 0; //是否显示词级别时间戳。,详情见API文档
 config.noiseThreshold = 0.5; // 噪音参数阈值，默认为0，取值范围：[-1,1],详情见API文档
 config.noiseThreshold = 0; // 噪音参数阈值，默认为0，取值范围：[-1,1]
 config.maxSpeakTime = 1000 * 5; // 强制断句功能，取值范围 5000-90000(单位:毫秒)，默认值0(不开启)。 在连续说话不间断情况下，该参数将实现强制断句（此时结果变成稳态，slice_type=2）。如：游戏解说场景，解说员持续不间断解说，无法断句的情况下，将此参数设置为10000，则将在每10秒收到 slice_type=2的回调。
 [config setApiParam:@"noise_threshold" value:@(0.5)]; // 设置自定义请求参数,用于在请求中添加SDK尚未支持的参数
```

3. **创建 QCloudRealTimeRecognizer 实例：**

```
 QCloudRealTimeRecognizer *recognizer = [[QCloudRealTimeRecognizer alloc] initWithConfig:config];
```

4. **设置 delegate，实现 QCloudRealTimeRecognizerDelegate 方法：**

```
recognizer.delegate = self;
```

5. **开始识别：**

```
  //使用内置录音器前需要先设置AVAudioSession状态为可录音的模式
  NSError *error = nil;
  [[AVAudioSession sharedInstance] setCategory:AVAudioSessionCategoryRecord error:&error];
  if (error) {
     //错误处理
  }
  [[AVAudioSession sharedInstance] setActive:YES error:nil];
//启动识别
 [recognizer start];
```

6. **结束识别：**

```
 [recognizer stop];
```

3.2 调用者提供语音数据示例

1. **引入 SDK 的头文件：**

```
#import <QCloudRealTime/QCloudRealTimeRecognizer.h>
#import <QCloudRealTime/QCloudConfig.h>
#import <QCloudRealTime/QCloudRealTimeResult.h>
#import <QCloudRealTime/QCloudAudioDataSource.h>
```

2. **创建 QCloudConfig 实例：**

```
 //1.创建 QCloudConfig 实例
 QCloudConfig *config = [[QCloudConfig alloc] initWithAppId:kQDAppId
```

```
                              secretId:kQDSecretId
                            secretKey:kQDSecretKey
                            projectId:0];

//以下为可选配置参数
 config.requestTimeout = 10;     //请求超时时间（秒）
 //config.sliceTime = 40;            //语音分片时长默认40ms（无特殊需求不建议更改）
 config.enableDetectVolume = YES;      //是否检测音量
 config.endRecognizeWhenDetectSilence = YES;     //是否检测到静音停止识别
 config.shouldSaveAsFile = YES; //仅限使用SDK内置录音器有效，是否保存录音文件到本地 默认关闭
 config.saveFilePath = [NSTemporaryDirectory() stringByAppendingPathComponent:@"recordaudio.wav"]; //开
启shouldSaveAsFile后音频保存的路径，仅限使用SDK内置录音器有效,默认路径为[NSTemporaryDirectory()
stringByAppendingPathComponent:@"recordaudio.wav"]

//以下为API参数配置，参数描述见API文档：https://cloud.tencent.com/document/product/1093/48982
 config.engineType = @"16k_zh";//设置引擎，不设置默认16k_zh
 config.filterDirty = 0;        //是否过滤脏词，具体的取值见API文档的filter_dirty参数
 config.filterModal = 0;        //过滤语气词具体的取值见API文档的filter_modal参数
 config.filterPunc = 0;         //过滤句末的句号具体的取值见API文档的filter_punc参数
 config.convertNumMode = 1;     //是否进行阿拉伯数字智能转换。具体的取值见API文档的convert_num_mode参数
 //config.hotwordId = @"";           //热词id。具体的取值见API文档的hotword_id参数
 //config.customizationId = @"";   //自学习模型id,详情见API文档
 //config.vadSilenceTime = -1;     //语音断句检测阈值,详情见API文档
 config.needvad = 1;    //默认1 0：关闭 vad，1：开启 vad。 如果语音分片长度超过60秒，用户需开启 vad。
 config.wordInfo = 0; //是否显示词级别时间戳。,详情见API文档
 config.noiseThreshold = 0.5; // 噪音参数阈值，默认为0，取值范围：[-1,1],详情见API文档
 config.noiseThreshold = 0; // 噪音参数阈值，默认为0，取值范围：[-1,1]
 [config setApiParam:@"noise_threshold" value:@(0.5)]; // 设置自定义请求参数,用于在请求中添加SDK尚未支持的参数
```

3. **自定义 QCloudDemoAudioDataSource，QCloudDemoAudioDataSource 实现 QCloudAudioDataSource 协议：**

```
//QCloudDemoAudioDataSource 具体源代码相见SDK demo目录
QCloudDemoAudioDataSource *dataSource = [[QCloudDemoAudioDataSource alloc] init];
```

4. **创建 QCloudRealTimeRecognizer 实例：**

```
 QCloudRealTimeRecognizer *recognizer = [[QCloudRealTimeRecognizer alloc] initWithConfig:config
dataSource:dataSource];
```

5. **设置 delegate，实现 QCloudRealTimeRecognizerDelegate 方法：**

```
 recognizer.delegate = self;
```

6. **开始识别：**

```
 [recognizer start];
```

7. **结束识别：**

```
 [recognizer stop];
```

4 主要接口类和方法说明说明

4.1 QCloudRealTimeRecognizer 识别类说明

QCloudRealTimeRecognizer 是实时语音识别类，提供两种初始化方法。

1. initWithConfig:(QCloudConfig *)config

初始化方法，调用者使用内置录音器采集音频。

名称	类型	描述
config	QCloudConfig*	配置类,用于初始化QCloudRealTimeRecognizer

2. initWithConfig:(QCloudConfig *)config dataSource:(id<QCloudAudioDataSource>)dataSource

初始化方法,调用者使用自定义数据源提供音频。

名称	类型	描述
config	QCloudConfig*	配置类,用于初始化QCloudRealTimeRecognizer
dataSource	id<QCloudAudioDataSource>	自定义数据源

示例

```
QCloudConfig *config = [[QCloudConfig alloc] initWithAppId:kQDAppId secretId:kQDSecretId
secretKey:kQDSecretKey projectId:[kQDProjectId integerValue]];
QCloudDemoAudioDataSource *dataSource = [[QCloudDemoAudioDataSource alloc] init];
_realTimeRecognizer = [[QCloudRealTimeRecognizer alloc] initWithConfig:config dataSource:dataSource];
```

4.2 QCloudConfig 配置说明

配置类,用于QCloudRealTimeRecognizer初始化。

1. initWithAppId:(NSString *)appid secretId:(NSString *)secretId secretKey:(NSString *)secretKey projectId:(NSInteger)projectId;

初始化方法-直接鉴权。

名称	类型	描述
appid	NSString *	腾讯云appId
secretId	NSString *	腾讯云secretId
secretKey	NSString *	腾讯云 secretKey
projectId	NSString *	腾讯云 projectId

2. initWithAppId:(NSString *)appid secretId:(NSString *)secretId secretKey:(NSString *)secretKey token:(NSString *)token projectId:(NSInteger)projectId;

初始化方法-通过 STS 临时证书鉴权，详见 获取联合身份临时访问凭证 。

名称	类型	描述
appid	NSString *	腾讯云appId
secretId	NSString *	腾讯云临时secretId
secretKey	NSString *	腾讯云临时secretKey
token	NSString *	临时token
projectId	NSString *	腾讯云 projectId

3. setApiParam:(NSString* _Nonnull)key value:(NSObject* _Nullable)value

设置自定义参数,该方法会在控制请求后端时的参数。

名称	类型	描述
key	NSString *	腾讯云appId
value	NSObject* _Nullable	nil会删除已添加参数,否则会在请求中添加参数

属性

属性名称	类型	描述
enableDetectVolume	BOOL	是否检测录音音量的变化, 开启后sdk会实时回调音量变化
endRecognizeWhenDetectSilence	BOOL	是否识别静音，默认YES
endRecognizeWhenDetectSilenceAutoStop	BOOL	识别到静音是否停止本次识别，默认YES
silenceDetectDuration	float	最大静音时间阈值, 超过silenceDetectDuration时间不说话则为静音, 单位:秒
sliceTime	NSInteger	分片时间, 此参数影响语音分片长度, 单位:毫秒,必须为20的整倍数，如果不是，sdk内将自动调整为20的整倍数，例如77将被调整为60，如果您不了解此参数不建议更改
requestTimeout	NSInteger	网络请求超时时间，单位:秒, 取值范围[5-60], 默认20
compression	BOOL	是否压缩音频。默认压缩，压缩音频有助于优化弱网或网络不稳定时的识别速度及稳定性。SDK历史版本均默认压缩且不提供配置开关，如无特殊需求，建议使用默认值
engineType	NSString	引擎识别类型,默认16k_zh
filterDirty	NSInteger	是否过滤脏词，具体的取值见API文档的filter_dirty参数
filterModal	NSInteger	过滤语气词具体的取值见API文档的filter_modal参数
filterPunc	NSInteger	过滤句末的句号具体的取值见API文档的filter_punc参数
convertNumMode	NSInteger	是否进行阿拉伯数字智能转换。具体的取值见API文档的convert_num_mode参数
hotwordId	NSString	热词id。具体的取值见API文档的hotword_id参数
customizationId	NSString	自学习模型id,具体的取值见API文档的customization_id参数
vadSilenceTime	NSInteger	语音断句检测阈值，静音时长超过该阈值会被认为断句（多用在智能客服场景，需配合 needvad = 1 使用），具体的取值见API文档vad_silence_time
needvad	NSInteger	默认1 0：关闭 vad，1：开启 vad。如果语音分片长度超过60秒，用户需开启 vad。具体的取值见API文档
wordInfo	NSInteger	是否显示词级别时间戳。0：不显示；1：显示，不包含标点时间戳，2：显示，包含标点时间戳。默认为0。具体的取值见API文档
noiseThreshold	float	噪音参数阈值，默认为0，取值范围：[-1,1],具体的取值见API文档
maxSpeakTime	NSInteger	强制断句功能，取值范围 5000-90000（单位:毫秒），默认值0(不开启)。 在连续说话不间断情况下，该参数将实现强制断句（此时结果变成稳态，slice_type=2）。如：游戏解说场景，解说员持续不间断解说，无法断句的情况下，将此参数设置为10000，则将在每10秒收到 slice_type=2的回调。具体的取值见API文档
keepMicrophoneRecording	BOOL	默认关闭 开启后 需要调用 stopMicrophone 停止麦克风。使用场景：在停止识别后 需要麦克风继续录音一段时间（录音不会上传服务器 不会识别 也不会保存）只支持内置录音设置
shouldSaveAsFile	BOOL	shouldSaveAsFile：仅限使用SDK内置录音器有效，是否保存录音文件到本地 默认关闭
saveFilePath	NSString	SaveFilePath：开启shouldSaveAsFile后音频保存的路径，仅限使用SDK内置录音器有效，默认路径为[NSTemporaryDirectory() stringByAppendingPathComponent:@"recordaudio.wav"]

示例

```
QCloudConfig *config = [[QCloudConfig alloc] initWithAppId:kQDAppId secretId:kQDSecretId
secretKey:kQDSecretKey projectId:[kQDProjectId integerValue]];
config.sliceTime = 40;                              //语音分片时长40ms
config.enableDetectVolume = _volumeDetectSwitch.on; //是否检测音量
config.endRecognizeWhenDetectSilence = _silenceDetectEndSwitch.on; //是否检测静音
config.endRecognizeWhenDetectSilenceAutoStop = YES;//是否检测到静音停止识别，默认YES
config.silenceDetectDuration = 3.0;
config.requestTimeout = 10;
```

4.3 QCloudRealTimeRecognizerDelegate 回调说明

用于接收识别过程中的识别结果和相关状态回调。

1. (void)realTimeRecognizerOnSliceRecognize:(QCloudRealTimeRecognizer *)recognizer result:(QCloudRealTimeResult *)result;

每个语音包分片识别结果。

名称	类型	描述
recognizer	QCloudRealTimeRecognizer *	QCloudRealTimeRecognizer实例
result	QCloudRealTimeResult *	语音分片的识别结果（非稳态结果，会持续修正）

2. (void)realTimeRecognizerOnSegmentSuccessRecognize:(QCloudRealTimeRecognizer *)recognizer result:(QCloudRealTimeResult *)result

语音流的识别结果,一次识别中可以包括多句话，这里持续返回的每句话的识别结果。

名称	类型	描述
recognizer	QCloudRealTimeRecognizer *	QCloudRealTimeRecognizer实例
result	QCloudRealTimeResult *	语音分片的识别结果（稳态结果）

3. (void)realTimeRecognizerDidFinish:(QCloudRealTimeRecognizer *)recognizer result:(NSString *)result

一次识别任务成功完成后的成功回调。

名称	类型	描述
recognizer	QCloudRealTimeRecognizer *	QCloudRealTimeRecognizer实例
result	QCloudRealTimeResult *	一次识别出的总文本, 实际是由SDK本地处理，将本次识别的realTimeRecognizerOnSegmentSuccessRecognize 识别结果拼接后一次性返回

4. (void)realTimeRecognizerDidError:(QCloudRealTimeRecognizer *)recognizer result:(QCloudRealTimeResult *)result;

一次识别任务失败回调。

名称	类型	描述
recognizer	QCloudRealTimeRecognizer *	QCloudRealTimeRecognizer实例
result	QCloudRealTimeResult *	识别结果信息，错误信息详情看QCloudRealTimeResponse内错误码

5. (void)realTimeRecognizerDidStartRecord:(QCloudRealTimeRecognizer *)recognizer error:(NSError * _Nullable)error

开始录音回调。

名称	类型	描述
recognizer	QCloudRealTimeRecognizer *	QCloudRealTimeRecognizer实例
error	NSError *	开启录音失败，错误信息

6. (void)realTimeRecognizerDidStopRecord:(QCloudRealTimeRecognizer *)recognizer;

结束录音回调。

名称	类型	描述
recognizer	QCloudRealTimeRecognizer *	QCloudRealTimeRecognizer实例

7. (void)realTimeRecognizerDidUpdateVolumeDB:(QCloudRealTimeRecognizer *)recognizer volume:(float)volume;

录音音量(单位为分贝)实时回调,此回调计算音量的分贝值。

名称	类型	描述
recognizer	QCloudRealTimeRecognizer *	QCloudRealTimeRecognizer实例
volume	float	音量分贝，取值范围(0~100)，集中分布在40~80

8. (void)realTimeRecognizerDidSaveAudioDataAsFile:(QCloudRealTimeRecognizer *)recognizer audioFilePath:(NSString *)audioFilePath;

录音停止后回调一次，再次开始录音会清空上一次保存的文件。

名称	类型	描述
recognizer	QCloudRealTimeRecognizer *	QCloudRealTimeRecognizer实例
audioFilePath	NSString *	音频文件路径

9. (void)realTimeRecgnizerLogOutPutWithLog:(NSString *)log;

日志输出回调。

名称	类型	描述
recognizer	QCloudRealTimeRecognizer *	QCloudRealTimeRecognizer实例
log	NSString *	日志信息

4.4 QCloudAudioDataSource 协议说明

调用者不使用 SDK 内置录音器进行语音数据采集，自己提供语音数据需要实现此协议所有方法，可见 Demo 工程中的 QDAudioDataSource 实现。

1. (void)start:(void(^)(BOOL didStart, NSError *error))completion;

SDK会调用start方法，实现此协议的类需要初始化数据源。SDK会根据didStart值判断是否开始 ,YES 往下执行，NO不会往下执行。
运行后需将 running 属性设置为 YES。

2. (void)stop;

SDK 会调用 stop 方法，实现此协议的类需要停止提供数据。
运行后需将 running 属性设置为 NO。

3. (nullable NSData *)readData:(NSInteger)expectLength;

SDK 会调用实现此协议的对象的此方法读取语音数据, 如果语音数据不足 expectLength，read 线程进入休眠。

5. 错误码

- 后端错误码，详情请参见 API 文档 。
- 客户端错误码如下：

错误码	名称	描述
-100	QCloudRealTimeClientErrCode_NetworkError	无网络
-101	QCloudRealTimeClientErrCode_Timeout	手机网络存在问题，请求超时
-102	QCloudRealTimeClientErrCode_MicError	录音过程音频通道被占用，录音失败，例如电话
-103	QCloudRealTimeClientErrCode_AudioInitError	音频源初始化失败（麦克风启动失败，权限拒绝等，如果使用自定义音频源start方法返回错误也会触发）

6. 常见问题指引

6.1 回音消除指引

本小节主要介绍如何通过 iOS 原生 API 实现回音消除，下面将介绍实现方案（完整代码参考 Demo 工程中的 QCloudAECDataSource 类的实现方法）。

6.1.1 回声消除方案介绍

1. 设置 AVAudioSession 支持边播放边录音的模式：

```
AVAudioSession* session = [AVAudioSession sharedInstance];
[session setCategory:AVAudioSessionCategoryPlayAndRecord mode:AVAudioSessionModeDefault
options:AVAudioSessionCategoryOptionDefaultToSpeaker error:&error];
```

2. 通过 AVAudioEngine 添加播放节点构建音频处理图：

```
self.engine = [[AVAudioEngine alloc] init];
self.play_node = [[AVAudioPlayerNode alloc] init];
[self.engine attachNode:self.play_node];
[self.engine connect:self.play_node to:self.engine.outputNode format:nil];
```

3. 通过调用输入节点的 setVoiceProcessingEnabled 开启回声消除：

```
[self.engine.inputNode setVoiceProcessingEnabled:YES error:&error];
```

4. 启动音频处理图：

```
[self.engine startAndReturnError:&error];
```

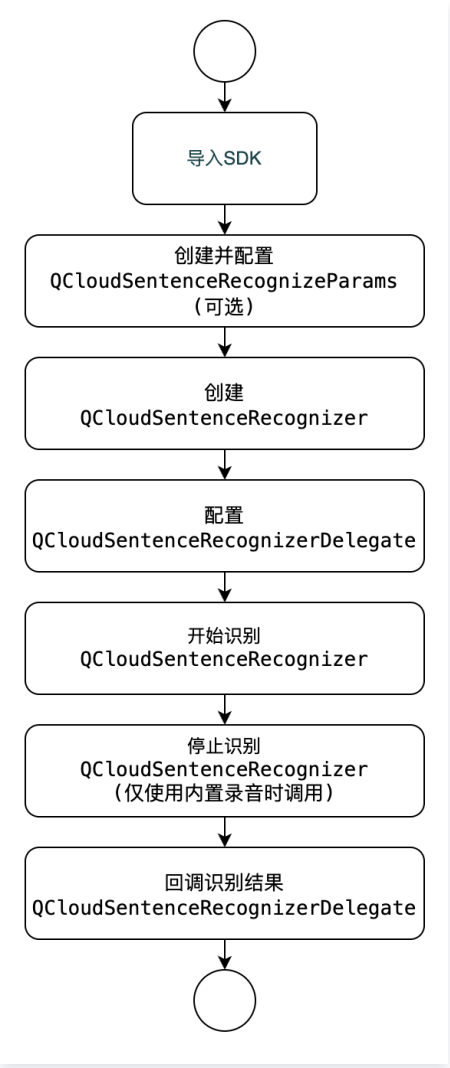
6.1.2 回音消除方案适配的机型列表

回声消除适配还会受到机型及系统的影响，SDK 包内的文档列举了已测试机型和系统的适配情况，可前往控制台下载 SDK。

一句话识别

最近更新时间：2024-12-24 09:54:12

接入流程



接入准备

SDK 获取

一句话识别的 iOS SDK 以及 Demo 的下载地址：接入 SDK 下载 。

接入须知

- 开发者在调用前请先查看实时语音识别的 接口说明，了解接口的**使用要求**和**使用步骤**。
- 该接口需要手机能够连接网络（3G、4G、5G 或 Wi-Fi 网络等），且系统为 **iOS 9.0**及以上版本。
- 运行 Demo 必须设置 AppID、SecretID、SecretKey，可在 API 密钥管理 中获取。

SDK 导入

直接导入

1. 下载并解压 iOS SDK 压缩包，压缩包中包含 Demo 和 SDK，其中 QCloudOneSentence.xcframework 为一句话识别 framework 包。
2. XcodeFile > Add Files to "Your Project"，在弹出 Panel 选中所下载SDK包 QCloudOneSentence.xcframework > Add（选中“Copy items if needed”）。

CocoaPods 导入

使用 CocoaPods 导入时,需将以下内容添加到 Podfile 中。

```
pod 'QCloudOneSentence'
```

工程配置

在工程 `info.plist` 申请系统麦克风权限，添加如下内容：

```
    <key>NSMicrophoneUsageDescription</key>
    <string>需要使用您的麦克风采集音频</string>
```

在工程中添加依赖库，在建阶段链接二进制与库中添加以下库：

- QCloudOneSentence.xcframework
- libc++.tbd
- AVFoundation.framework
- AudioToolbox.framework

快速接入

开发流程及接入示例

1. 创建 QCloudSentenceRecognizer 实例

```
 QCloudSentenceRecognizer *recognizer = [[QCloudSentenceRecognizer alloc] initWithAppId:appId
                                         secretId:secretId
                                         secretKey:secretKey];
 //设置delegate，相关回调方法见QCloudOneSentenceRecognizerDelegate定义
recognizer.delegate = self;
```

2. 实现此 QCloudSentenceRecognizerDelegate 协议方法
3. 调用示例
   - 通过语音 URL 调用

> **说明**
> 支持8K 和16K 的引擎类型，引擎模型类型请参见 EngSerViceType 。

```
//快捷接口
- (void)recognizeWithUrl {
//语音数据url
NSString *url = @"https://asr-audio-1256237915.cos.ap-shanghai.myqcloud.com/30s.wav";
  //指定语音数据url 语音数据格式 识别引擎
  //支持的格式及引擎名称以API文档为准，见https://cloud.tencent.com/document/product/1093/35646
  [_recognizer recoginizeWithUrl:url voiceFormat:@"wav" EngSerViceType:@"16k_zh"];
}

//完整接口，可设置更多参数
- (void)recognizeWithParams {
   NSString *url = @"https://asr-audio-1256237915.cos.ap-shanghai.myqcloud.com/30s.wav";
   //获取一个已设置默认参数params
   QCloudOneSentenceRecognitionParams *params = [_recognizer defaultRecognitionParams];
   //通过语音 url 请求，此4个参数必须设置
   params.url = url;
   //设置语音频数据格式，支持的格式以API文档为准，见https://cloud.tencent.com/document/product/1093/35646
   params.voiceFormat = @"wav";
   //设置语音数据来源，见QCloudAudioSourceType定义
```

```
    params.sourceType = QCloudAudioSourceTypeUrl;
    //设置识别引擎,支持的识别引擎以API文档为准，见https://cloud.tencent.com/document/product/1093/35646
    params.engSerViceType = @"16k_zh";

    //以下为可选项
     //是否过滤脏词（目前支持中文普通话引擎）。0：不过滤脏词；1：过滤脏词；2：将脏词替换为 * 。默认值为 0。
     params.filterDirty = 0;
     //是否过语气词（目前支持中文普通话引擎）。0：不过滤语气词；1：部分过滤；2：严格过滤 。默认值为 0。
     params.filterModal = 0;
     //是否过滤标点符号（目前支持中文普通话引擎）。 0：不过滤，1：过滤句末标点，2：过滤所有标点。默认值为 0。
     params.filterPunc = 0;
     //是否进行阿拉伯数字智能转换。0：不转换，直接输出中文数字，1：根据场景智能转换为阿拉伯数字。默认值为1。
     params.convertNumMode = 1;
     //是否显示词级别时间戳。0：不显示；1：显示，不包含标点时间戳，2：显示，包含标点时间戳。默认值为 0。
     params.wordInfo = 1;
     //params.hotwordId = @"" //热词id

    [_recognizer recognizeWithParams:params];
}
```

- **通过语音数据调用**

> **说明：**
> 支持8K 和16K 的引擎类型，引擎模型类型请参见 EngSerViceType 。

```
//快捷接口
- (void)recognizeWithAudioData {
    //语音数据
    NSString *filePath = [[NSBundle mainBundle] pathForResource:@"recordedFile" ofType:@"wav"];
    NSData *audioData = [[NSData alloc] initWithContentsOfFile:filePath];
    //指定语音数据 语音数据格式 识别引擎
    //支持的格式以API文档为准，见https://cloud.tencent.com/document/product/1093/48982
    [_recognizer recoginizeWithData:audioData voiceFormat:@"wav" frequence:@"16k_zh"];
}

//完整接口，可设置更多参数
- (void)recognizeWithParams {

     NSString *filePath = [[NSBundle mainBundle] pathForResource:@"test2" ofType:@"mp3"];
     NSData *audioData = [[NSData alloc] initWithContentsOfFile:filePath];

     //获取一个已设置默认参数params
     QCloudOneSentenceRecognitionParams *params = [_recognizer defaultRecognitionParams];
     //通过语音数据发起请求， 此4个参数必须设置
     params.data = audioData;
     //设置语音频数据格式，支持的格式以API文档为准，见https://cloud.tencent.com/document/product/1093/35646
     params.voiceFormat = @"mp3";
     //设置语音数据来源，QCloudAudioSourceTypeUrl 或 QCloudAudioSourceTypeAudioData
     params.sourceType = QCloudAudioSourceTypeAudioData;
     //设置识别引擎,支持的识别引擎以API文档为准，见https://cloud.tencent.com/document/product/1093/35646
     params.engSerViceType = @"16k_zh";

    //以下为可选项
     //是否过滤脏词（目前支持中文普通话引擎）。0：不过滤脏词；1：过滤脏词；2：将脏词替换为 * 。默认值为 0。
     params.filterDirty = 0;
     //是否过语气词（目前支持中文普通话引擎）。0：不过滤语气词；1：部分过滤；2：严格过滤 。默认值为 0。
     params.filterModal = 0;
     //是否过滤标点符号（目前支持中文普通话引擎）。 0：不过滤，1：过滤句末标点，2：过滤所有标点。默认值为 0。
     params.filterPunc = 0;
```

```
    //是否进行阿拉伯数字智能转换。0：不转换，直接输出中文数字，1：根据场景智能转换为阿拉伯数字。默认值为1。
    params.convertNumMode = 1;
    //是否显示词级别时间戳。0：不显示；1：显示，不包含标点时间戳，2：显示，包含标点时间戳。默认值为 0。
    params.wordInfo = 1;
    //params.hotwordId = @"" //热词id

    [_recognizer recognizeWithParams:params];
}
```

- **通过 SDK 内置录音器调用**

> **说明**
> 支持16K 的引擎类型，引擎模型类型请参见 EngSerViceType 。

```
//启动录音
  [_recognizer startRecognizeWithRecorder:@"16k_zh"];  //16k_zh > 识别引擎，传nil将默认使用16k_zh，支持的识别引擎以API文档为准，见https://cloud.tencent.com/document/product/1093/35646

//停止录音并上传录音数据开始识别
  [_recognizer stopRecognizeWithRecorder];
```

主要接口类说明

QCloudSentenceRecognizer 初始化说明

QCloudSentenceRecognizer 是一句话识别入口类，提供两种初始化方法。

```
/**
 * 初始化方法，调用者使用内置录音器采集音频
 * @param config 配置参数，详见 QCloudConfig 定义
 */
- (instancetype)initWithConfig:(QCloudConfig *)config;

/**
 * 直接鉴权
 * 通过 appId secretId secretKey 初始化
 * @param appid     腾讯云 appId
 * @param secretId  腾讯云 secretId
 * @param secretKey 腾讯云 secretKey
 */
- (instancetype)initWithAppId:(NSString *)appid secretId:(NSString *)secretId secretKey:(NSString
*)secretKey;

/**
 * 通过STS临时密钥鉴权，详见https://cloud.tencent.com/document/product/598/33416
 * @param appid     腾讯云appId
 * @param secretId  腾讯云临时secretId
 * @param secretKey 腾讯云临时secretKey
 * @param token     对应的token
 */
- (instancetype)initWithAppId:(NSString *)appid secretId:(NSString *)secretId secretKey:(NSString
*)secretKey token:(NSString *)token;
```

QCloudSentenceRecognizerDelegate 协议说明

此 delegate 为一句话识别相关回调，调用者需要实现此 delegate 获取识别结果、开始录音、结束录音事件。

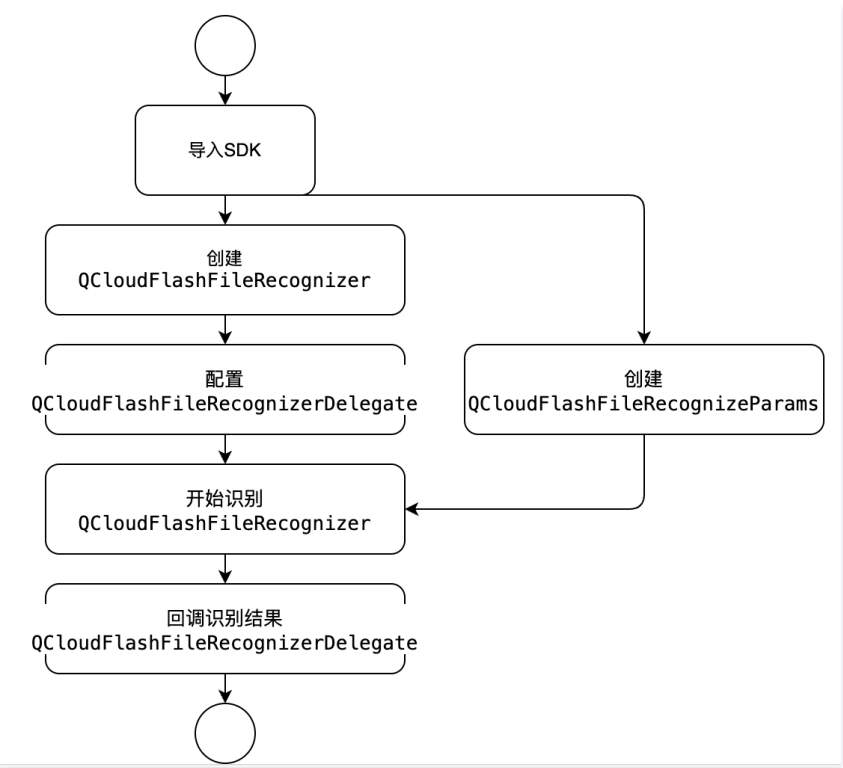
```
@protocol QCloudSentenceRecognizerDelegate <NSObject>
@required
/**
 * 一句话识别回调delegate
 * @param result 识别结果文本，error=nil此字段才存在值
 * @param error 错误信息，详细错误信息见error.domain和error.userInfo字段
 * @param rawData 识别原始数据
 */
- (void)oneSentenceRecognizerDidRecognize:(QCloudSentenceRecognizer *)recognizer text:(nullable NSString
*)text error:(nullable NSError *)error resultData:(nullable NSDictionary *)resultData;
@optional
/**
 * 开始录音回调
 */
- (void)oneSentenceRecognizerDidStartRecord:(QCloudSentenceRecognizer *)recognizer error:(nullable NSError
*)error;
/**
 * 结束录音回调， SDK通过此方法回调后内部开始上报语音数据进行识别
 */
- (void)oneSentenceRecognizerDidEndRecord:(QCloudSentenceRecognizer *)recognizer;
/**
 * 录音音量实时回调用
 * @param recognizer 识别器实例
 * @param volume 声音音量，取值范围（-40-0）
 */
- (void)oneSentenceRecognizerDidUpdateVolume:(QCloudSentenceRecognizer *)recognizer volume:(float)volume;
/**
 * 日志输出
 * @param log 日志
 */
- (void)SentenceRecgnizerLogOutPutWithLog:(NSString *_Nullable)log;

@end
```

录音文件识别极速版

最近更新时间：2025-01-02 11:14:52

接入流程



开发准备

SDK 获取

录音文件识别的 iOS SDK 以及 Demo 的下载地址：接入 SDK 下载。

接入须知

- 开发者在调用前请先查看录音文件识别极速版的 接口说明，了解接口的**使用要求**和**使用步骤**。
- 该接口需要手机能够连接网络（3G、4G、5G 或 Wi-Fi 网络等），且系统为 **iOS 9.0**及以上版本。
- 运行 Demo 必须设置 AppID、SecretID、SecretKey，可在 API 密钥管理 中获取。

SDK 导入

直接导入

1. 下载并解压 iOS SDK 压缩包，压缩包中包含 Demo 和 SDK，其中 QCloudFileRecognizer.xcframework 为录音文件识别极速版 framework 包。
2. XcodeFile > Add Files to "Your Project"，在弹出 Panel 选中所下载 SDK 包 QCloudFileRecognizer.xcframework > Add（选中“Copy items if needed”）。

CocoaPods导入

使用 CocoaPods 导入时,需将以下内容添加到 Podfile 中。

```
pod 'QCloudFileRecognizer'
```

工程配置

在工程中添加依赖库，在 build Phases Link Binary With Libraries 中添加以下库：

- QCloudFileRecognizer.xcframework
- libc++.tbd
- AVFoundation.framework
- AudioToolbox.framework

类说明

QCloudFlashFileRecognizer 初始化说明

QCloudFlashFileRecognizer 是录音文件极速版入口类。

```
/**
  通过 appId secretId secretKey 初始化
  @param appid     腾讯云 appId
  @param secretId  腾讯云 secretId
  @param secretKey 腾讯云 secretKey
 **/
- (instancetype)initWithAppId:(NSString *)appid secretId:(NSString *)secretId secretKey:(NSString
*)secretKey;

/**
  通过 appId 临时secretId 临时secretKey token 初始化
  详见 https://cloud.tencent.com/document/product/598/33416
  @param appid     腾讯云 appId
  @param secretId  腾讯云 临时secretId
  @param secretKey 腾讯云 临时secretKey
  @param token 腾讯云 token
 **/
- (instancetype)initWithAppId:(NSString *)appid secretId:(NSString *)secretId secretKey:(NSString
*)secretKey token:(NSString *)token;
```

QCloudFlashFileRecognizerDelegate 协议说明

此 delegate 为录音文件识别相关回调，调用者需要实现此 delegate 获取识别结果事件。

```
@protocol QCloudFlashFileRecognizerDelegate <NSObject>
@optional

/**
 录音文件识别获取服务器结果成功回调

@param recognizer 录音文件识别器
 @param status 非0时识别失败
 @param text 识别文本，status非0时，此为服务器端返回的错误信息
 @param resultData 原始数据
 */
- (void)FlashFileRecognizer:(QCloudFlashFileRecognizer *_Nullable)recognizer status:(nullable NSInteger *)
status text:(nullable NSString *)text resultData:(nullable NSDictionary *)resultData;

/**
 录音文件识别失败回调
 @param recognizer 录音文件识别器
 @param error 识别错误，出现错误此字段有
 @param resultData 原始数据
 */
- (void)FlashFileRecognizer:(QCloudFlashFileRecognizer *_Nullable)recognizer error:(nullable NSError
*)error resultData:(nullable NSDictionary *)resultData;

/**
 * 日志输出
```

```
 * @param log 日志
 */
- (void)FlashFileRecgnizerLogOutPutWithLog:(NSString *_Nullable)log;

@end
```

示例

1. 创建 QCloudFlashFileRecognizer 实例

```
  QCloudFlashFileRecognizer *recognizer = [[QCloudFlashFileRecognizer alloc] initWithAppId:appId
                                        secretId:secretId secretKey:secretKey];
  //设置 delegate，相关回调方法见 QCloudFlashFileRecognizerDelegate 定义
 recognizer.delegate = self;
```

2. 实现此 QCloudFlashFileRecognizerDelegate 协议方法
3. 调用方式示例

```
 (void)recognizeWithAudioData {
 QCloudFlashFileRecognizeParams *params = [QCloudFlashFileRecognizeParams defaultRequestParams];
 NSString *filePath = [[NSBundle mainBundle] pathForResource:@"test" ofType:@"mp3"];
 NSData *audioData = [[NSData alloc] initWithContentsOfFile:filePath];
 params.audioData = audioData;
 //音频格式。支持 wav、pcm、ogg-opus、speex、silk、mp3、m4a、aac。
 params.voiceFormat = @"mp3";

 //以下参数不设置将使用默认值
 params.engineModelType = @"16k_zh";//引擎模型类型，默认16k_zh。8k_zh：8k 中文普通话通用；16k_zh：16k 中文普通话通用；16k_zh_video：16k 音视频领域。
 params.filterDirty = 0;;// 0 ：默认状态 不过滤脏话 1：过滤脏话
 params.filterModal = 0;// 0 ：默认状态 不过滤语气词  1：过滤部分语气词 2:严格过滤
 params.filterPunc = 0;// 0 ：默认状态 不过滤句末的句号 1：滤句末的句号
 params.convertNumMode = 1;;//1：默认状态 根据场景智能转换为阿拉伯数字；0：全部转为中文数字。
 params.speakerDiarization = 0; //是否开启说话人分离（目前支持中文普通话引擎），默认为0，0：不开启，1：开启。
 params.firstChannelOnly = 1; //是否只识别首个声道，默认为1。0：识别所有声道；1：识别首个声道。
 params.wordInfo = 0; //是否显示词级别时间戳，默认为0。0：不显示；1：显示，不包含标点时间戳，2：显示，包含标点时间戳。
 params.customizationID = @""; //自学习模型 id。如设置了该参数，将生效对应的自学习模型。
 params.hotwordID = @""; // 热词表 id。如不设置该参数，自动生效默认热词表；如设置了该参数，那么将生效对应的热词表。


 [_recognizer recognize:params];
}
```

Flutter
实时语音识别

最近更新时间：2025-06-11 14:20:32

Flutter SDK

SDK 以插件的方式封装了 Android 和 iOS 实时语音识别功能，提供 Flutter 版本的实时语音识别，本文介绍 SDK 的安装方法及示例。

开发环境

- Dart >= 2.18.4
- Flutter >= 3.3.8
- Android API Level >= 16
- iOS >= 9.0

获取安装

请去 控制台 下载 SDK，SDK 内 asr_plugin 目录即为 Flutter 插件，插件内 example 目录下为 demo 示例。

接入指引

此插件仅支持Android和iOS两个平台且包含了平台相关的库,使用时请确保开发环境包含Android Studio及XCode否则集成时会出现编译问题

1. 将项目中asr_plugin目录复制到自己的Flutter工程下
2. 在自己项目的配置文件pubspec.yaml下添加依赖

```
  asr_plugin:
    # 该路径根据asr_plugin存放路径改变
    path: ../asr_plugin
```

3. 在需要使用到的页面,导入asr_plugin的依赖。

```
import 'package:asr_plugin/asr_plugin.dart';
```

接口说明

接口示例代码为 demo 部分代码，完整代码请参考位于 example 里的 demo 示例。

ASRControllerConfig

配置相关参数用于生成 ASRController。
参数:

```
int appID = 0; // 腾讯云 appID
int projectID = 0; //腾讯云 projectID
String secretID = ""; //腾讯云 secretID
String secretKey = ""; // 腾讯云 projectKey
String token = null; // 腾讯云临时授权

String engine_model_type = "16k_zh"; //设置引擎，不设置默认16k_zh
int filter_dirty = 0; //是否过滤脏词，具体的取值见API文档的filter_dirty参数
int filter_modal = 0; //过滤语气词具体的取值见API文档的filter_modal参数
int filter_punc = 0; //过滤句末的句号具体的取值见API文档的filter_punc参数
int convert_num_mode = 1; //是否进行阿拉伯数字智能转换。具体的取值见API文档的convert_num_mode参数
String hotword_id = ""; //热词id。具体的取值见API文档的hotword_id参数
String customization_id = ""; //自学习模型id,详情见API文档
int vad_silence_time = 0; //语音断句检测阈值,详情见API文档
int needvad = 1; //人声切分,详情见API文档
```

```
  int word_info = 0; //是否显示词级别时间戳，详情见API文档
  int reinforce_hotword = 0; //热词增强功能，详情见API文档

  bool is_compress = true; //是否开启音频压缩，开启后使用opus压缩传输数据
  bool silence_detect = false; //静音检测功能，开启后检测到静音会停止识别
  int silence_detect_duration = 5000; //静音检测时长，开启静音检测功能后生效
  bool is_save_audio_file = false; //是否保存音频，仅对内置录音生效，格式为s16le，16000Hz，mono的pcm，开启后会通过
NOTIFY类型的ASRData返回到上层，其中ASRData中info为以下的JSON格式{"type":"onAudioFile, "code": 0, "message":
"audio file path"}
  String audio_file_path = ""; //is_save_audio_file为true时，会将音频保存在指定位置

  //自定义参数设置函数
  void setCustomParam(String key, dynamic value)
```

方法：

```
Future<ASRController> build() async <--> 创建ASRController
```

示例：

```
var _config = ASRControllerConfig()
_config.filter_dirty = 1;
_config.filter_modal = 0;
_config.filter_punc = 0;
var _controller = await _config.build();
```

ASRController

控制语音识别的流程及获取语音识别的结果。

方法：

```
Stream<ASRData> recognize() async* <--> 开始识别，通过监听Stream可以获得实时语音识别的相关数据
Stream<ASRData> recognizeWithDataSource(Stream<Uint8List>? source) async* <--> 开始识别，可传入自定义数据源进行
识别，有关数据源的要求参考自定义数据源
stop() async <--> 停止识别
release() async <--> 释放资源
```

示例：

```
try {
    if (_controller != null) {
      await _controller?.release();
    }
    _controller = await _config.build();
    setState(() {
      _btn_onclick = stopRecognize;
    });
    await for (final val in _controller!.recognize()) {
      switch (val.type) {
        case ASRDataType.SLICE:
        case ASRDataType.SEGMENT:
          var id = val.id!;
          var res = val.res!;
          if (id >= _sentences.length) {
            for (var i = _sentences.length; i <= id; i++) {
              _sentences.add("");
            }
          }
```

```
          _sentences[id] = res;
          setState(() {
            _result = _sentences.map((e) => e).join("");
          });
          break;
        case ASRDataType.SUCCESS:
          setState(() {
            _btn_onclick = startRecognize;
            _result = val.result!;
            _sentences = [];
          });
          break;
      }
    }
  } on ASRError catch (e) {
    setState(() {
      _btn_onclick = startRecognize;
      _result = "错误码: ${e.code} \n错误信息: ${e.message} \n详细信息: ${e.resp}";
    });
  } catch (e) {
    log(e.toString());
    setState(() {
      _btn_onclick = startRecognize;
    });
  }
}
```

ASRData

识别过程中返回的数据。
参数:

```
ASRDataType type; //数据类型
int? id; //句子的id
String? res; //数据类型为SLICE和SEGMENT时返回部分识别结果
String? result; // 数据类型SUCCESS时返回所有识别结果
```

ASRDataType

ASRData 数据类型。

```
enum ASRDataType {
  SLICE,
  SEGMENT,
  SUCCESS,
}
```

ASRError

识别过程中的错误。
参数:

```
int code; //错误码 iOS参考QCloudRealTimeClientErrCode Android参考ClientException
String message; //错误消息
String? resp; //服务端返回的原始数据
```

自定义数据源

SDK 只负责对输入的语音进行识别，不会进行额外的处理。但调用者可以通过自定义数据源来实现对录音数据的处理(降噪,回声消除等)来满足相应的场景需求。

自定义数据源需要数据以 `Stream<Uint8List>` 的方式传入到 SDK 且需要满足以下的要求。

1. 采样数据格式仅支持单通道16000hz、16bit、小端的 pcm 数据流。
2. 采用数据需要每隔40ms向 stream 推入1280B的数据且数据格式需要满足条件1。

一句话识别

最近更新时间：2023-10-26 15:51:31

Flutter SDK

SDK 以插件的方式封装了一句话识别功能，提供 flutter 版本的一句话识别，本文介绍 SDK 的安装方法及示例。

开发环境

- Dart >= 2.18.4
- Flutter >= 3.3.8
- Android API Level >= 16
- iOS >= 9.0

获取安装

下载SDK SDK 内 asr_plugin 目录即为 flutter 插件,插件内 example 目录下为 demo 示例。

接入指引

此插件仅支持 Android 和 iOS 两个平台且包含了平台相关的库,使用时请确保开发环境包含 Android Studio 及 XCode 否则可能会出现集成时的编译问题。

1. 将项目中 asr_plugin 目录复制到自己的 Flutter 工程下。
2. 在自己项目的配置文件 pubspec.yaml 下添加依赖。

```
  asr_plugin:
    # 该路径根据asr_plugin存放路径改变
    path: ../asr_plugin
```

3. 在需要使用到的页面,导入 asr_plugin 的依赖。

```
import 'package:asr_plugin/asr_plugin.dart';
```

接口说明

接口示例代码为 demo 部分代码,完整代码请参考位于 example 里的 demo 示例。

OneSentenceASRParams

一句话识别请求的相关参数,可参考 一句话识别 的描述。

示例

```
var _params = OneSentenceASRParams();
_params.binary_data = Uint8List.view((await rootBundle.load("assets/30s.wav")).buffer);
_params.voice_format = OneSentenceASRParams.FORMAT_WAV;
```

OneSentenceASRController

控制一句话识别的流程及获取一句话识别的结果

方法

```
Future<OneSentenceASRResult> recognize(OneSentenceASRParams params) async;
```

示例

```
_result = (await _controller.recognize(_params)).response_body;
```

OneSentenceASRResult

返回一句话识别的结果，数据类型与 API 文档描述对应，可参考一句话识别 的描述。
参数

```
String response_body = ""; // 服务端返回原始信息
String? request_id; // 唯一请求 ID
String? result; // 识别结果
int? duration; // 请求的音频时长，单位为ms
int? word_size; // 词时间戳列表的长度
List<SentenceWords>? word_list; //词时间戳列表
Error? error; // 错误信息
```

SentenceWords

SentenceWords 数据类型，可参考语音识别-数据接口 的描述。
参数

```
String? word; // 词结果
int? offset_start_ms; // 词在音频中的开始时间
int? offset_end_ms; // 词在音频中的结束时间
```

Error

服务端返回的错误信息
参数

```
String code; // 错误码
String message; // 错误信息
```

录音文件识别极速版

最近更新时间：2023-10-26 15:51:31

Flutter SDK

SDK 以插件的方式封装了录音文件识别极速版功能，提供 flutter 版本的录音文件识别极速版，本文介绍 SDK 的安装方法及示例。

开发环境

- Dart >= 2.18.4
- Flutter >= 3.3.8
- Android API Level >= 16
- iOS >= 9.0

获取安装

下载SDK SDK 内 asr_plugin 目录即为 flutter 插件，插件内 example 目录下为 demo 示例。

接入指引

此插件仅支持 Android 和 iOS 两个平台且包含了平台相关的库,使用时请确保开发环境包含 Android Studio 及 XCode 否则可能会出现集成时的编译问题。

1. 将项目中 asr_plugin 目录复制到自己的 Flutter 工程下。
2. 在自己项目的配置文件 pubspec.yaml 下添加依赖。

```
  asr_plugin:
    # 该路径根据asr_plugin存放路径改变
    path: ../asr_plugin
```

3. 在需要使用到的页面,导入 asr_plugin 的依赖。

```
import 'package:asr_plugin/asr_plugin.dart';
```

接口说明

接口示例代码为 demo 部分代码,完整代码请参考位于 example 里的 demo 示例。

FlashFileASRParams

录音文件识别极速版请求的相关参数，可参考 录音文件识别极速版 的描述。

示例

```
var _params = FlashFileASRParams();
_params.data = Uint8List.view((await rootBundle.load("assets/30s.wav")).buffer);
_params.voice_format = OneSentenceASRParams.FORMAT_WAV;
```

FlashFileASRController

控制录音文件识别极速版的流程及获取录音文件识别极速版的结果。

方法

```
Future<FlashFileASRResult> recognize(FlashFileASRParams params) async;
```

示例

```
var ret = (await _controller.recognize(_params));
```

FlashFileASRResult

返回录音文件识别极速版的结果，数据类型与 API 文档描述对应，可参考 录音文件识别极速版 的描述。
参数

```
String response_body = ""; // 服务端返回原始信息
late int code; // 0：正常，其他，发生错误
late String message; // code 非0时，message 中会有错误消息
late String request_id; // 请求唯一标识，请您记录该值，以便排查错误
late int audio_duration; // 音频时长，单位为毫秒
List<Result>? flash_result; // 声道识别结果列表
```

Sentence

Sentence 数据类型，可参考 录音文件识别极速版 的描述。
参数

```
String text; // 句子/段落级别文本
int start_time; // 开始时间
int end_time; // 结束时间
int speaker_id; // 说话人 Id（请求中如果设置了 speaker_diarization，可以按照 speaker_id 来区分说话人）
List<Word>? word_list;// 词级别的识别结果列表
```

Result

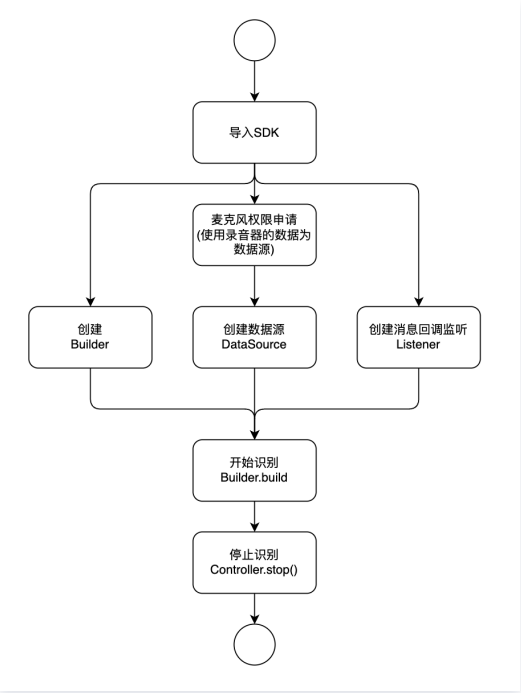
Result 数据类型，可参考 录音文件识别极速版 的描述。
参数

```
int channel_id = 0; // 声道标识，从0开始，对应音频声道数
String text = ""; // 声道音频完整识别结果
List<Sentence>? sentence_list; // 句子/段落级别的识别结果列表
```

HarmonyOS NEXT
实时语音识别

最近更新时间：2025-04-23 18:01:52

1. 接入流程



2. 接入准备

2.1 SDK 获取

实时语音识别 Harmony SDK 以及 Demo 的下载地址：接入 SDK 下载 。

2.2 接入须知

- 开发者在调用前请先查看实时语音识别的接口说明，了解接口的**使用要求**和**使用步骤**。
- 该接口需要手机能够连接网络（3G、4G、5G 或 Wi-Fi 等）。
- 运行 Demo 必须设置 AppID、SecretID、SecretKey，可在 API 密钥管理 中获取。

2.3 开发环境

- **添加实时语音识别 SDK har：**

```
"dependencies": {
    "qcloudrealtime": "file:../qcloudrealtime.har"
}
```

需根据实际路径替换配置。

3. 快速接入

以下为 demo 中的代码片段，完整代码请参考 sdk/demo 工程。

3.1 配置识别任务

```
private _builder: QCloud.RealTime.Builder = new QCloud.RealTime.Builder
```

```
...
this._builder.appID = this._app_id
this._builder.secretID = this._secret_id
this._builder.secretKey = this._secret_key
```

配置识别任务相关参数。

3.2 启动识别任务

使用上面的 Builder 可以创建并启动识别任务：

```
this._controller = this._builder.build(
      this._source, {
      onSlice: (data, raw) => {
        console.log(`----- ${data}`)
        this._tmp_result = data
      },
      onSegment: (data, raw) => {
        console.log(`----- ${data}`)
        this._result += data
        this._tmp_result = ''
      },
      onData(data) {
        console.log(`----- ${data}}`)
      },
      onSilence: () => {
        this._controller?.cancel()
        console.log(`----- silence`)
      },
      onVolume: (db) => {
        this._volume = db
      },
      onLogger(level, message) {
        console.log(`${message}`)
      },
    })
```

3.3 停止识别任务

停止识别任务可以通过控制器的 stop 方法停止：

```
this._controller!.stop()
```

另一种情况当数据源的 empty 返回 true 时识别也会停止。

4. 主要接口类和方法说明

4.1 Builder 类说明

Builder 用于创建语音识别任务。

方法

1. **setApiParam(key: string, value: string | number | null)**

   设置请求后台 websocket 时的参数，后台支持参数请参考 实时语音识别(websocket) 。

名称	类型	描述
key	string	请求后台的参数名称
value	string|number|null	请求后台的参数值,null时将删除已设置的参数,number会转为string后设置

2. **getApiParam(key: string): string | null**

   获取已设置的请求参数。

名称	类型	描述
key	string	参数名称

3. **build(source: DataSource, listener: Listener): Controller**

   创建并启动识别任务同时返回识别任务控制器。

属性

名称	类型	描述
appID	string	腾讯云appId
secretID	string	腾讯云临时secretId
secretKey	string	腾讯云临时secretKey
token	string	腾讯云临时token，为空字符串时表示不使用临时授权，临时授权参考 获取联合身份临时访问凭证
slice	number	分片时间，单位ms，开启opus时必须为40的倍数，不建议更改此参数
enableOpus	boolean	开启opus编码
enableSilence	boolean	开启静音检测，开启后识别到静音后会触发一次静音回调，不会停止任务
enableDetectVolume	boolean	开启音量检测，开启后会通过音量回调返回当前音量
silenceTimeout	number	静音检测时长，开启静音检测后生效
loggerLevel	LoggerLevel	日志等级，日志回调只会返回大于该等级的日志
saveDataPath	string	保存音频数据路径，路径为空不生效，否则将会存储数据源读取到的音频数据到该路径

以下属性是为了简化设置请求 API 参数流程，内部实现 key 值固定的 setApiParam 和 getApiParam 方法。

描述中仅说明 key 值，详细说明请参考 实时语音识别(websocket) 。

名称	类型	描述
engineModelType	string	engine_model_type
needVad	number	needvad
hotwordID	string	hotword_id
customizationID	number	customization_id
filterDirty	number	filter_dirty
filterModal	number	filter_modal
filterPunc	number	filter_punc
filterEmptyResult	number	filter_empty_result
convertNumMode	number	convert_num_mode
wordInfo	number	word_info
vadSilenceTime	number	vad_silence_time
maxSpeakTime	number	max_speak_time

noiseThreshold	number	noise_threshold
hotwordList	string	hotword_list
inputSampleRate	number	input_sample_rate

4.2 Controller 类说明

Controller 用于控制识别任务的流程。

属性

名称	类型	描述
task	Promise<void>	识别任务，识别任务正常结束则此 Promise 正常返回，否则抛出异常
voice_id	string	连接成功后的 voiceid，否则为空字符

方法

1. **stop(): void**

   停止识别任务，调用此方法会向后台发送停止识别的消息，成功停止后 task 会结束。

2. **cancel(): void**

   取消识别任务，调用此方法会取消当前识别任务，成功取消后 task 会抛出 SDKErrorImpl 类型的异常，且此异常的 code 为 CANCEL_ERROR。

4.3 Listener 类说明

Listener 类用于接收识别过程中的信息。

属性

名称	类型	描述
onSlice?	(data: string, raw: string) => void	收到后台 websocket 的消息后且解析到 slice_type 为0或1回调该方法
onSegment?	(data: string, raw: string) => void	收到后台 websocket 的消息后且解析到 slice_type 为2回调该方法
onData?	(data: string) => void	收到后台 websocket 的消息后回调该方法
onLogger?	(level: LoggerLevel, message: string) => void	日志回调，Builder 的 loggerLevel 会影响此回调的内容
onSilence?	() => void	静音回调，开启 Builder 的 enableSilence 后检测到静音后回调
onVolume?	(db: number) => void	音量回调，开启 Builder 的 enableDetectVolume 后回调

4.4 DataSource 类说明

DataSource 类用来提供用于识别的音频数据。

属性

名称	类型	描述
voice_format	string	音频编码，目前仅支持 pcm，即数据类型为16000hz，16bit，小端的pcm数据流
empty	boolean	返回是否还有音频数据
start	() => Promise<void>	SDK 开始识别时会调用此方法，此方法发生的异常会通过task抛出
stop	() => Promise<void>	SDK 停止识别时会调用此方法，此方法发生的异常会通过task抛出
read	(ms: number) => Promise<ArrayBuffer>	SDK 读取音频数据，ms为读取数据的时长，voice_format 为 pcm 时，此方法需返回40 *16 * 2长度的数据，此方法发生的异常会通过 task 抛出

5. 错误码

识别任务抛出类型为 SDKErrorImpl 时的错误码。

错误码	名称	描述
-1	UNKNOWN_ERROR	未知错误
1	NETWORK_ERROR	网络错误
2	SERVER_ERROR	服务端错误，详情通过 message 查看
3	SIGNATURE_ERROR	签名错误
4	PARAMETER_ERROR	参数错误，Builder 设置的参数无法创建识别任务
5	CANCEL_ERROR	取消错误，调用 cancel 方法取消任务时出现
6	CREATE_FILE_ERROR	创建文件失败
7	DATA_LENGTH_ERROR	数据源返回的数据长度不符合要求

一句话识别

最近更新时间：2025-02-20 09:45:52

1. 接入准备

1.1 SDK获取

实时语音识别 Harmony SDK 以及 Demo 的下载地址：接入 SDK 下载 。

1.2 接入须知

- 开发者在调用前请先查看一句话识别的接口说明，了解接口的使用要求和使用步骤。
- 该接口需要手机能够连接网络（3G、4G、5G 或 Wi-Fi 等）
- 运行 Demo 必须设置 AppID、SecretID、SecretKey，可在 API 密钥管理 中获取。

1.3 开发环境

- 添加一句话识别SDK har

```
"dependencies": {
    qcloudonesentence": "file:./sdk/qcloudonesentence.har"
}
```

需根据实际路径替换配置

2. 快速接入

以下为demo中的代码片段,完整代码请参考sdk/demo工程。

2.1 配置识别任务

```
let builder = new QCloud.OneSentence.Builder()
builder.secretID = this._secret_id
builder.secretKey = this._secret_key
builder.token = this._token
builder.setApiParam(QCloud.OneSentence.kEngSerViceType, '16k_zh')
builder.setApiParam(QCloud.OneSentence.kSourceType, 0)
builder.setApiParam(QCloud.OneSentence.kVoiceFormat, 'mp3')
builder.setApiParam(QCloud.OneSentence.kUrl, this._url)
```

配置识别任务相关参数

2.2 启动识别任务

使用上面的Builder可以启动识别任务获取结果

```
this._result = await builder.build().task
```

3. 主要接口类和方法说明

3.1 Builder类说明

Builder 用于创建语音识别任务

方法

i. setApiParam(key: string, value: string | number | null)

设置请求后台一句话识别接口时的参数,后台支持参数请参考一句话接口说明。

名称	类型	描述

key	string	请求后台的参数名称
value	string|number|null	请求后台的参数值，null 时将删除已设置的参数，number 会转为 string 后设置

ii. getApiParam(key: string): string | number | null

获取已设置的请求参数

名称	类型	描述
key	string	参数名称

iii .build(): Controller

创建并启动识别任务同时返回识别任务控制器

属性

名称	类型	描述
secretID	string	腾讯云临时 secretId
secretKey	string	腾讯云临时 secretKey
token	string	腾讯云临时 token，为空字符串时表示不使用临时授权，临时授权参考 获取联合身份临时访问凭证 。

3.2 Controller类说明

Controller 用于控制识别任务的流程

属性

名称	类型	描述
task	Promise<string>	识别任务，识别任务正常结束则此 Promise 正常返回后台返回参数，否则抛出异常

4. 错误码

识别任务抛出类型为 SDKErrorImpl 时的错误码

错误码	名称	描述
-1	UNKNOWN_ERROR	未知错误
1	NETWORK_ERROR	网络错误
2	SERVER_ERROR	服务端错误，详情通过 message 查看

录音文件识别极速版

最近更新时间：2025-02-20 09:45:52

1. 接入准备

1.1 SDK获取

实时语音识别 Harmony SDK 以及 Demo 的下载地址：接入SDK下载

1.2 接入须知

- 开发者在调用前请先查看录音文件识别极速版的接口说明，了解接口的使用要求和使用步骤。
- 该接口需要手机能够连接网络（3G、4G、5G 或 Wi-Fi 等）
- 运行 Demo 必须设置 AppID、SecretID、SecretKey，可在 API 密钥管理 中获取。

1.3 开发环境

- 添加录音文件识别极速版SDK har

```
"dependencies": {
    "qcloudfileflash": "file:./sdk/qcloudfileflash.har"
}
```

需根据实际路径替换配置

2. 快速接入

以下为 demo 中的代码片段,完整代码请参考 sdk/demo 工程。

2.1 配置识别任务

```
let builder = new QCloud.FileFlash.Builder()
builder.appID = this._app_id
builder.secretID = this._secret_id
builder.secretKey = this._secret_key
builder.token = this._token
builder.setApiParam(QCloud.FileFlash.kEngineType, '16k_zh')
builder.setApiParam(QCloud.FileFlash.kVoiceFormat, 'pcm')
```

配置识别任务相关参数

2.2 启动识别任务

使用上面的 Builder 可以启动识别任务获取结果

```
this._result = await builder.build(v).task
```

3. 主要接口类和方法说明

3.1 Builder 类说明

Builder 用于创建语音识别任务

方法

i. setApiParam(key: string, value: string | number | null)

设置请求后台录音文件识别极速版接口时的参数，后台支持参数请参考 录音文件识别极速版接口说明 。

名称	类型	描述
key	string	请求后台的参数名称

value	string|number|null	请求后台的参数值，null 时将删除已设置的参数，number 会转为 string 后设置

ii. **getApiParam(key: string): string | number | null**

获取已设置的请求参数

名称	类型	描述
key	string	参数名称

iii. **build(data: ArrayBuffer): Controller**

创建并启动识别任务同时返回识别任务控制器

名称	类型	描述
data	ArrayBuffer	上传的音频数据

属性

名称	类型	描述
appID	string	腾讯云 appId
secretID	string	腾讯云临时 secretId
secretKey	string	腾讯云临时 secretKey
token	string	腾讯云临时 token，为空字符串时表示不使用临时授权，临时授权参考 获取联合身份临时访问凭证 。

3.2 Controller类说明

Controller 用于控制识别任务的流程

属性

名称	类型	描述
task	Promise<string>	识别任务，识别任务正常结束则此 Promise 正常返回后台返回参数，否则抛出异常

4. 错误码

识别任务抛出类型为 SDKErrorImpl 时的错误码

错误码	名称	描述
-1	UNKNOWN_ERROR	未知错误
1	NETWORK_ERROR	网络错误
2	SERVER_ERROR	服务端错误，详情通过 message 查看

语音识别 SDK 个人信息保护规则

最近更新时间：2024-12-23 11:33:22

更新说明

我们对《腾讯云语音识别 SDK 个人信息保护规则》进行了更新，更新内容主要为：

1. 新增 SDK 在鸿蒙端处理个人信息的相关内容；
2. 明确不涉及扩展业务功能；
3. 明确不涉及可选个人信息；
4. 细化说明 SDK 向 SDK 使用者提供扩展业务功能、可选个人信息配置选项。

引言

腾讯云语音识别 SDK（以下简称“SDK 产品”）由腾讯云计算（北京）有限责任公司以下简称（“我们”）开发，公司注册地为北京市海淀区知春路49号3层西部309。

《腾讯云语音识别 SDK 个人信息保护规则》（以下简称“本规则”）主要向开发者及其终端用户（“终端用户”）告知，为了实现 SDK 产品的相关功能，SDK 产品需收集、使用和处理终端用户个人信息的情况。

请开发者及终端用户认真阅读本规则。如您是开发者，请您确认充分了解并同意本规则后再集成 SDK 产品，**如果您不同意本规则及按照本规则履行对应的用户个人信息保护义务，应立即停止接入及使用 SDK 产品；同时，您应仅在征得终端用户的同意后集成 SDK 产品并处理终端用户的个人信息，在获得终端用户同意前不得启用或初始化本 SDK 产品**。

特别说明

如您是开发者，您应当：

1. **遵守法律、法规收集、使用和处理终端用户的个人信息，包括但不限于制定和公布有关个人信息保护的隐私政策等。**
2. **告知终端用户SDK产品收集、使用和处理终端用户个人信息的情况，并依法征得终端用户同意，在征得终端用户同意后初始化SDK产品。**
3. **在征得终端用户的同意前、以及在用户触发相应功能场景前，除非法律法规另有规定，不应收集任何终端用户的个人信息。**
4. **应按您的应用的具体功能场景，在用户触发具体功能场景时调用 SDK 的相应功能、调用相应权限或处理终端用户的个人信息，未到具体功能场景时不应调用相应的 SDK 功能、调用相应权限或处理终端用户的个人信息。**
5. **向终端用户提供易于操作且满足法律法规要求的用户权利实现机制, 并告知终端用户如何查阅、复制、修改、删除个人信息, 撤回同意, 以及限制个人信息处理、转移个人信息、获取个人信息副本和注销账号。**
6. **遵守本规则的要求，并详细阅读《语音识别 SDK 合规使用指南》查看详细操作指引。**

如开发者和终端用户对本规则内容有任何疑问或建议，可随时通过本规则第八条提供的方式与我们联系。

一、我们收集的信息及我们如何使用信息

（一）为实现 SDK 产品功能所需收集的个人信息

为实现SDK产品的基本业务功能所必须，我们将向终端用户或开发者收集终端用户在使用与SDK产品相关的功能时产生的如下个人信息。SDK目前不涉及提供其他扩展业务功能，如后续提供其他扩展业务功能，将通过本规则进行说明并通过《语音识别SDK合规使用指南》为开发者提供扩展功能的配置方式和示例：

个人信息类型	处理目的	处理方式	是否可选
音频数据	为实现 SDK 语音转文字基础功能，需要上传音频识别为文字	加密传输处理	必选

（二）为实现 SDK 产品功能所需的权限

为实现SDK产品的相应功能所必须,我们会通过开发者的应用在对应的功能场景下申请所需权限。如您是开发者，请您注意，您应按您的应用的具体功能场景，在用户触发具体功能场景时调用SDK的相应功能、调用相应权限或处理终端用户的个人信息，未到具体业务或功能场景时不应调用相应权限，点击此处可查看相关操作指引。

请您注意，对于SDK相应功能的可选权限，SDK不会强制获取，即使没有获取该可选权限，SDK的相应功能也能正常运行，点击此处可查看关于配置可选权限的相关操作指引。

操作系统	权限名称	使用目的	功能场景（申请时机）	是否可选

Android	android.permission.ACCESS_NETWORK_STATE	网络状态：获取网络状态，用于优化当前网络通信	用户在使用第三方开发者应用中与语音识别的功能场景时	必选
	android.permission.INTERNET	网络访问：允许程序访问网络，用于连接网络，进行数据传输	用户在使用第三方开发者应用中与语音识别的功能场景时	必选
	android.permission.RECORD_AUDIO	录音：通过麦克风采集音频，以实现音频转文字功能	用户输入语音进行识别时	可选
iOS	Privacy – Microphone Usage Description	麦克风：通过麦克风采集音频，以实现音频转文字功能	用户输入语音进行识别时	可选
Harmony OS NEXT	ohos.permission.MICROPHONE	麦克风：通过麦克风采集音频，以实现音频转文字功能	用户输入语音进行识别时	可选
	ohos.permission.INTERNET	网络访问：允许程序访问网络，用于连接网络，进行数据传输	用户在使用第三方开发者应用中与语音识别的功能场景时	必选

当开发者下载安装的是语音识别 Flutter 版 SDK 时，开发者需在编译时先确定使用 Flutter 版 SDK 包中的 iOS 版还是 Android 版，选择之后 SDK 再运转对应的子版本，且其收集处理个人信息和申请权限的情况与上述 iOS、Android 版披露的情况一致。
请注意，在不同设备和系统中，权限显示方式及关闭方式会有所不同，需同时参考其使用的设备及操作系统开发方的说明或指引。当终端用户关闭权限即代表其取消了相应的授权，我们和开发者将不会继续收集和使用相关权限所对应的个人信息，也无法为终端用户提供需要终端用户开启权限才能提供的对应的功能。

（三）根据法律法规的规定，以下是征得用户同意的例外情形：

1. 为订立、履行与终端用户的合同所必需。
2. 为履行我们的法定义务所必需。
3. 为应对突发公共卫生事件，或者紧急情况下为保护终端用户的生命健康和财产安全所必需。
4. 为公共利益实施新闻报道、舆论监督等行为，在合理的范围内处理终端用户的个人信息。
5. 依照本法规定在合理的范围内处理终端用户自行公开或者其他已经合法公开的个人信息。
6. 法律行政法规规定的其他情形。

特别提示：如我们收集的信息无法单独或结合其他信息识别到终端用户的个人身份，其不属于法律意义上的个人信息。

二、信息的公开披露

我们不会与我们的关联公司、合作伙伴及第三方共享（“接收方”）终端用户的个人信息。我们与第三方合作过程中，将遵守法律规定，按照最小必要原则，安全审慎地处理相关数据。我们将按照法律法规的规定，对数据处理涉及的第三方进行严格的限制，要求其严格遵守我们关于个人信息保护的措施与要求。
我们不会将终端用户的个人信息转移给任何公司、组织和个人，但以下情况除外：

1. 事先告知终端用户转移个人信息的种类、目的、方式和范围，并征得终端用户的单独同意。
2. 如涉及合并、分立、解散、被宣告破产等原因需要转移个人信息的，我们会向终端用户告知接收方的名称或者姓名和联系方式，并要求接收方继续履行个人信息处理者的义务。接收方变更原先的处理目的、处理方式的，我们会要求接收方重新取得终端用户的同意。

我们不会公开披露终端用户的个人信息，但以下情况除外：

1. 告知终端用户公开披露的个人信息的种类、目的、方式和范围并征得终端用户的单独同意后。
2. 在法律法规、法律程序、诉讼或政府主管部门强制要求的情况下。

三、终端用户如何管理自己的信息

我们非常重视终端用户对其个人信息管理的权利，并竭力帮助终端用户管理个人信息，包括个人信息查阅、复制、删除、注销账号以及设置隐私功能等，以保障终端用户的权利。如您是开发者，您应当为终端用户提供实现查阅、复制、修改、删除个人信息、撤回同意和注销账号的方式。
基于终端用户的同意而进行的个人信息处理活动，终端用户有权撤回该同意。我们已向开发者提供关闭本 SDK 产品的能力，开发者可以通过不发起对以下

【android 包含（ QCloudFileRecognizer、QCloudFlashRecognizer、AAIClient、QCloudOneSentenceRecognizer）；iOS 包含（initWithAppId 、QCloudRealTimeRecognizer、QCloudFlashFileRecognizer、QCloudFileRecognizer、QCloudSentenceRecognizer）；Flutter 包含(ASRController TTSController); Harmony Next包含(QCloud.RealTime.Controller)】接口的调用来停止收集和处理终端用户的个人信息，请开发者 点击此处 查看操作指引。由于我们与终端用户无直接的交互对话界面，终端用户可以直接联系开发者停止使用本 SDK 产品，也可通过本规则第八条提供的方式与我们联系。如您是终端用户，请您理解，特定的业务功能或服务需要您提供服务所需的信息才能得以完成，当您撤回同意后，我们无法继续为您提供对应的功能或服务，也不再处理您相应的个人信息。您撤回同意的决定，不会影响我们此前基于您的授权而开展的个人信息处理。

四、信息的存储

（一）存储信息的地点

我们遵守法律法规的规定，将在中华人民共和国境内收集和产生的个人信息存储在境内。

（二）存储信息的期限

一般而言，我们仅在为实现目的所必需的最短时间内保留终端用户的个人信息，但下列情况除外：

1. 为遵守适用的法律法规等有关规定。
2. 为遵守法院判决、裁定或其他法律程序的规定。
3. 为遵守相关政府机关执法的要求。

五、信息安全

- 我们为终端用户的个人信息提供相应的安全保障，以防止信息的丢失、不当使用、未经授权访问或披露。
- 我们严格遵守法律法规保护终端用户的个人信息。
- 我们将在合理的安全水平内使用各种安全保护措施以保障信息的安全。
  例如，我们使用加密技术、匿名化处理等手段来保护终端用户的个人信息。
- 我们建立严谨的管理制度、流程和组织确保信息安全。
  例如，我们严格限制访问信息的人员范围，要求他们遵守保密义务，并进行审查。
- 若发生个人信息泄露等安全事件，我们会启动应急预案，阻止安全事件扩大，并以推送通知、公告等形式告知开发者。

六、未成年人保护

本 SDK 产品主要面向成年人。
若您是开发者，如果终端用户是未满14周岁的未成年人（“儿童”），您应当向儿童的父母或其他监护人告知本规则，并在征得儿童的父母或其他监护人同意的前提下处理儿童个人信息。如果我们发现开发者未征得儿童监护人同意向我们提供儿童个人信息的，我们将会采取措施尽快删除。
若您是儿童监护人，当您对您所监护儿童个人信息保护有相关疑问或权利请求时，您可以联系开发者，或通过本规则提供的方式与我们联系。

七、变更

我们会适时修订本规则的内容。
如本规则的修订会导致终端用户在本规则项下权利的实质减损，我们将在变更生效前，通过网站公告等方式进行告知。如您是开发者，当更新后的本规则对处理终端用户的个人信息情况有变动的，您应当适时更新隐私政策，并以弹框形式通知终端用户并且征得其同意，**如果终端用户不同意接受本规则，请停止集成 SDK 产品**。

八、联系我们

我们设立了专门的个人信息保护团队和个人信息保护负责人，如果开发者和/或终端用户对本规则或个人信息保护相关事宜有任何疑问或投诉、建议时，可以通过以下方式与我们联系：

1. 通过 https://kf.qq.com/ 或者 腾讯云工单/客服 与我们联系或 在线咨询 。
2. 将问题发送至 Dataprivacy@tencent.com。
3. 邮寄信件至： 中国广东省深圳市南山区海天二路33号腾讯滨海大厦 数据隐私保护部（收）邮编： 518054。

我们将尽快审核所涉问题，并在15个工作日或法律法规规定的期限内予以反馈。

语音识别 SDK 合规使用指南

最近更新时间：2024-08-09 11:33:41

您可以通过点击以下链接查看语音识别 SDK 合规使用指南：

点击查看 >>

小程序 SDK
小程序插件

最近更新时间：2024-07-05 15:08:31

平台/语言	服务	SDK 集成说明
小程序	实时语音识别、一句话识别	一分钟跑通集成SDK

说明
因为微信录音参数支持问题，小程序 SDK 不支持 PC 端。

Web SDK
JS

最近更新时间：2023-11-17 09:53:03

平台/语言	服务	SDK 集成说明
JS	实时语音识别	Github

服务端 SDK
GO

最近更新时间：2023-11-17 09:53:04

平台/语言	服务	SDK 集成说明
GO	实时语音识别、录音文件识别极速版	Github
	录音文件识别、语音流异步识别、一句话识别	Github

JAVA

最近更新时间：2023-11-17 09:53:04

平台/语言	服务	SDK 集成说明
JAVA	实时语音识别、录音文件识别极速版	Github
	录音文件识别、语音流异步识别、一句话识别	Github

C++

最近更新时间：2023-11-17 09:53:04

平台/语言	服务	SDK 集成说明
C++	实时语音识别	Github
	录音文件识别、语音流异步识别、一句话识别	Github

Python

最近更新时间：2023-11-17 09:53:04

平台/语言	服务	SDK 集成说明
Python	实时语音识别、录音文件识别极速版	Github
	录音文件识别、语音流异步识别、一句话识别	Github

PHP

最近更新时间：2023-11-17 09:53:04

平台/语言	服务	SDK 集成说明
PHP	录音文件识别、语音流异步识别、一句话识别	Github

Node.js

最近更新时间：2023-11-17 09:53:04

平台/语言	服务	SDK 集成说明
Node.js	录音文件识别、语音流异步识别、一句话识别	Github

C#

最近更新时间：2023-11-17 09:53:04

平台/语言	服务	SDK 集成说明
C#	实时语音识别	Github

SDK 更新说明

最近更新时间：2025-05-16 14:03:02

SDK 版本说明

腾讯云语音识别 SDK（Android）

版本：V3.1.24
更新时间：2025-04-30
SDK 介绍：腾讯云语音识别客户端 SDK, 支持实时语音识别、一句话识别、录音文件识别极速版功能。
更新日志：

版本	更新内容
3.1.24	• [实时语音识别] 优化日志组件 • [一句话识别] 优化日志组件 • [录音文件识别极速版] 优化日志组件

服务提供方：腾讯云计算（北京）有限责任公司
合规使用说明：《语音识别 SDK 合规使用指南》
个人信息处理规则：《语音识别 SDK 个人信息保护规则》

腾讯云语音识别 SDK（iOS）

版本：V3.1.25
更新时间：2025-03-06
SDK 介绍：腾讯云语音识别客户端SDK, 支持实时语音识别、一句话识别、录音文件识别极速版功能。
更新日志：

版本	更新内容
3.1.25	• [实时语音识别] 修复了一些已知问题
3.1.24	• [实时语音识别] 增加取消识别功能
3.1.22	• [实时语音识别] 修复数据越界导致的 crash

服务提供方：腾讯云计算（北京）有限责任公司
合规使用说明：《语音识别 SDK 合规使用指南》
个人信息处理规则：《语音识别 SDK 个人信息保护规则》

腾讯云语音识别 SDK（Flutter）

版本：V0.0.12
更新时间：2024-10-25
SDK 介绍：腾讯云语音识别客户端 SDK，支持实时语音识别、一句话识别、录音文件识别极速版功能。
更新日志：

版本	更新内容
0.0.12	• 更新 iOS SDK 至v3.1.23 • 修复自定义音频源缺失数据问题
0.0.10	• 更新 iOS SDK 至v3.1.16 • 更新 Android SDK 至v3.1.18 • 修复回调丢失问题

服务提供方：腾讯云计算（北京）有限责任公司
合规使用说明：《语音识别 SDK 合规使用指南》
个人信息处理规则：《语音识别 SDK 个人信息保护规则》

腾讯云语音识别 SDK（Harmony）

版本：V1.1.2
更新时间：2025-02-19
SDK 介绍：腾讯云语音识别客户端 SDK, 支持实时语音识别、一句话识别、录音文件识别极速版功能。
更新日志：

版本	更新内容
1.1.2	• 兼容鸿蒙包命名新规则以及语法
1.1.0	• 新增一句话识别 • 新增录音文件识别极速版 • 优化实时语音识别参数设置
1.0.0	• 新增实时语音识别

服务提供方：腾讯云计算（北京）有限责任公司
合规使用说明：《语音识别 SDK 合规使用指南》
个人信息处理规则：《语音识别 SDK 个人信息保护规则》

离在线 SDK 文档
产品简介与开通授权

最近更新时间：2024-04-03 17:29:51

产品简介

语音识别离在线 SDK 支持**纯离线识别、离在线混合识别、在线识别**三种识别模式，客户可以根据业务需要，在 SDK 内定义识别模式后使用本产品，三种识别模式的具体介绍请参考下方描述：

识别模式	网络情况	支持语种	支持的接口	涉及的计费项
纯离线识别	无网络	主中文普通话，少量英文混说	仅支持实时语音识别	仅需按设备或按应用购买 离线产品
离在线混合识别	有网络，但部分情况下弱网或无网络。该模式会根据客户的网络情况自动切换在线和离线版本			需要按设备或按应用购买 离线产品，并同时购买 在线产品（实时语音识别） **请注意：离在线混合识别模式下，如果仅购买离线产品，在线部分会因无用量额度而无法正常识别**
在线识别	有网络			若仅有在线识别的需求，推荐优先通过 在线 SDK 接入语音识别服务

注意

- 语音识别离在线 SDK 仅限**经企业认证的**腾讯云账号使用。
- **纯离线识别模式，是无网络情况下的兜底识别版本，仅适用于简单识别的业务场景**。推荐客户使用离在线混合识别模式，SDK 将根据终端网络情况自动切换在线或纯离线识别模式，以此达到最好的识别效果和识别体验。
- 如果客户有离在线混合识别、在线识别的需求，请**务必**提前购买 在线产品（实时语音识别）的资源包，或在 语音技术控制台 开启账号的后付费功能，保证在线服务可以正常使用。

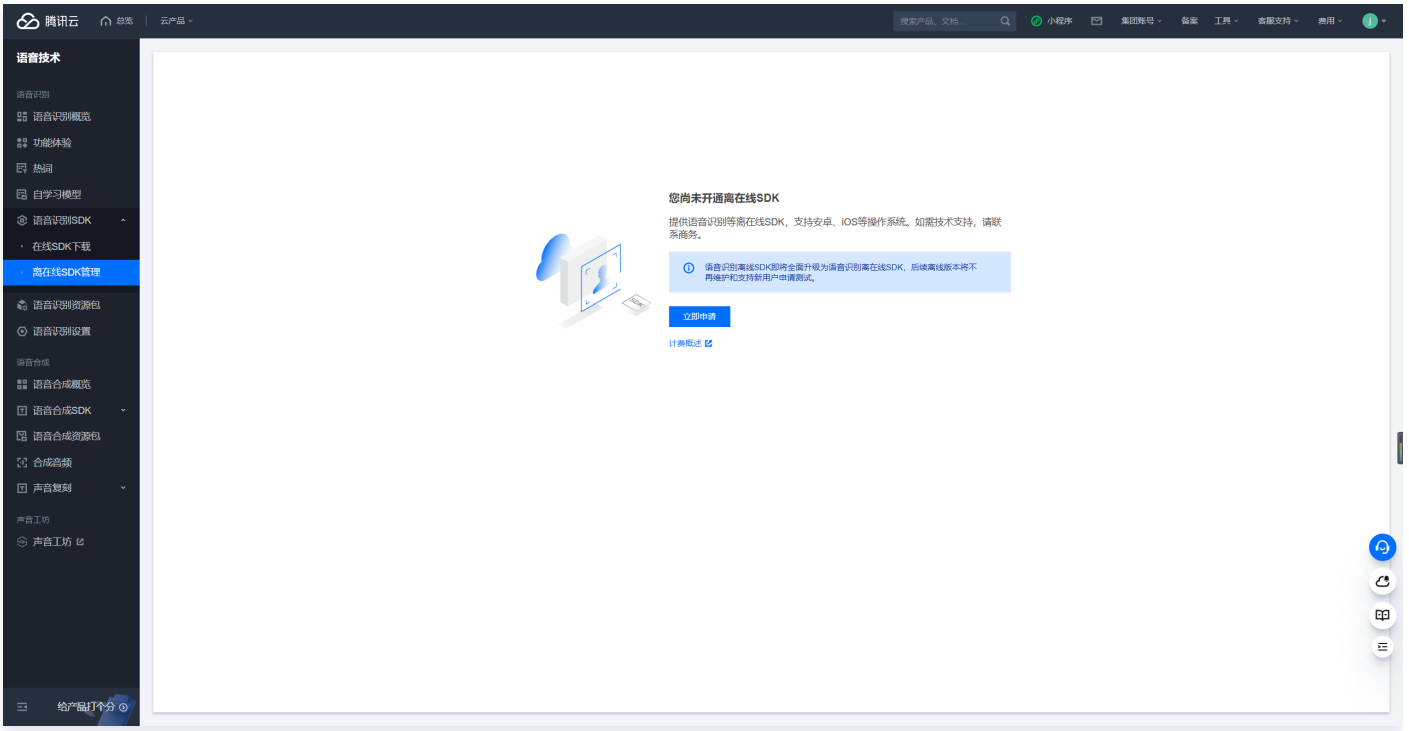
语音离在线SDK-控制台体验指引

测试准备

1. 准备好企业的腾讯云账号
2. 访问链接：登录 语音技术控制台

（备注：目前ASR、TTS、声音工坊控制台已合并为**语音技术**控制台）

3. 打开链接访问，选择**离在线 SDK 管理**（注意：提示“没有开通语音识别服务”，先击开通语音识别服务；提示未进行**企业认证**，请先进行企业认证），如下图所示：

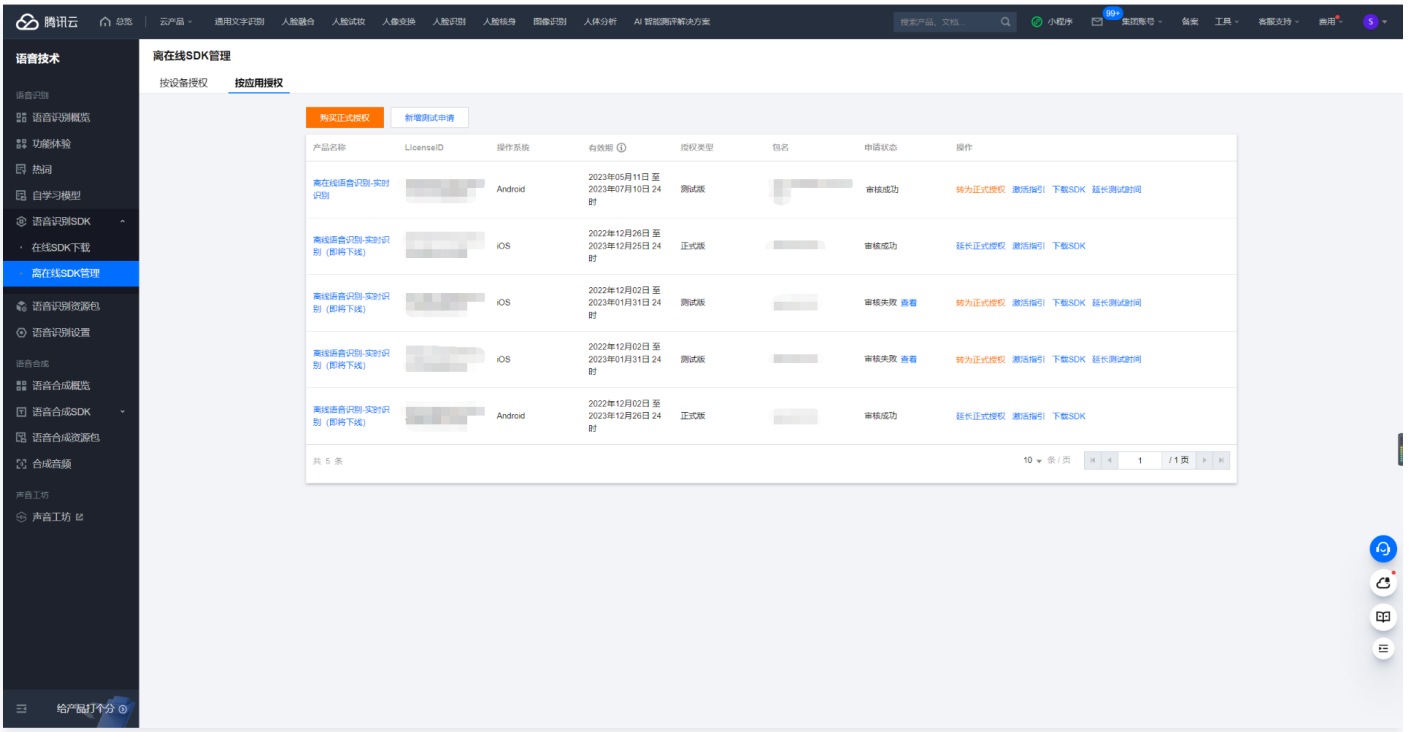


新增测试申请

1. 点击上图所示**立即申请**，弹出**语音识别 SDK 需求表**填写；
2. 填写**语音识别 SDK 需求表**，写完告诉对接的商务，安排需求审核（需求审核的目的是保证上线产品与客户需求匹配、可提供产品最终使用）；
3. 在**离在线 SDK 管理**页点击选择**按设备授权**或**按应用授权**，然后单击 **新增测试申请**，审核通过后即可测试试用。

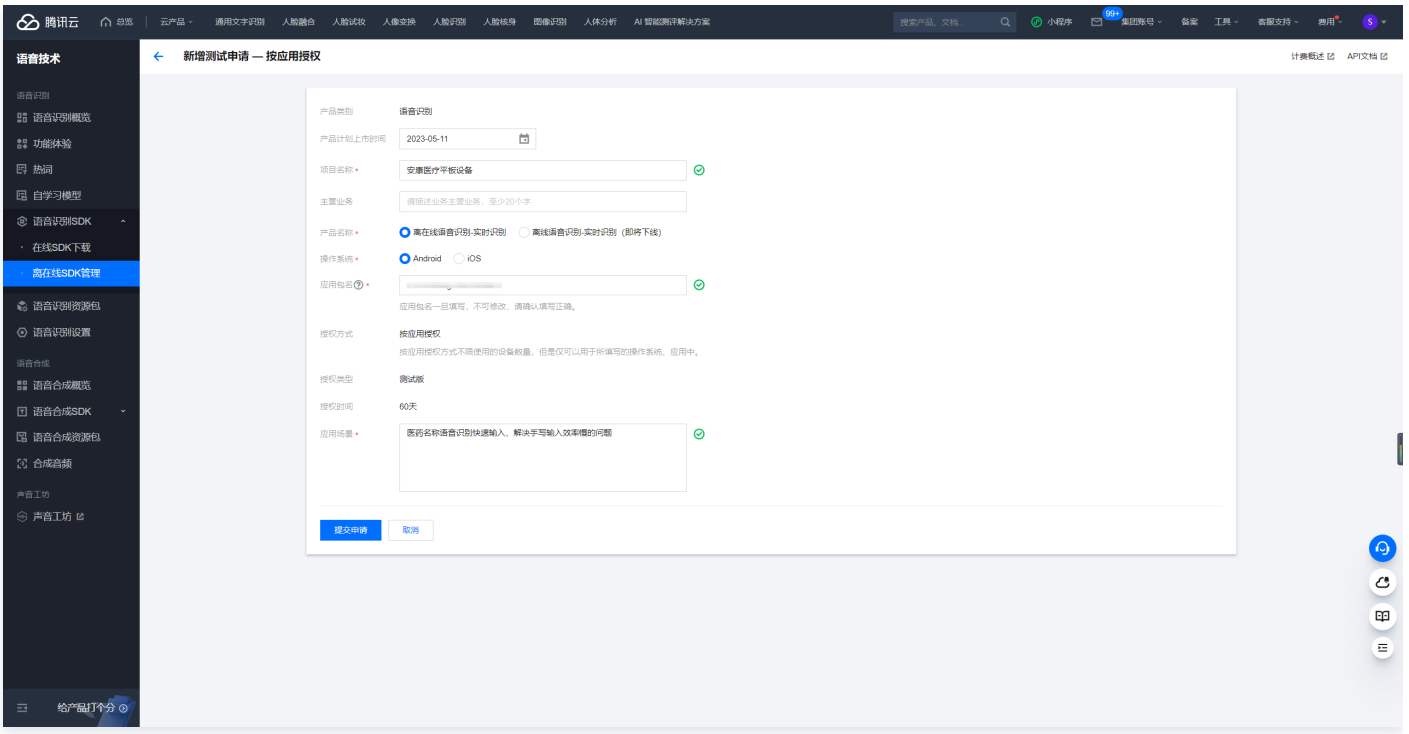
   按设备授权　有效期60天　默认授权测试设备数 5台
   按应用授权　有效期60天　默认授权测试应用数 1个

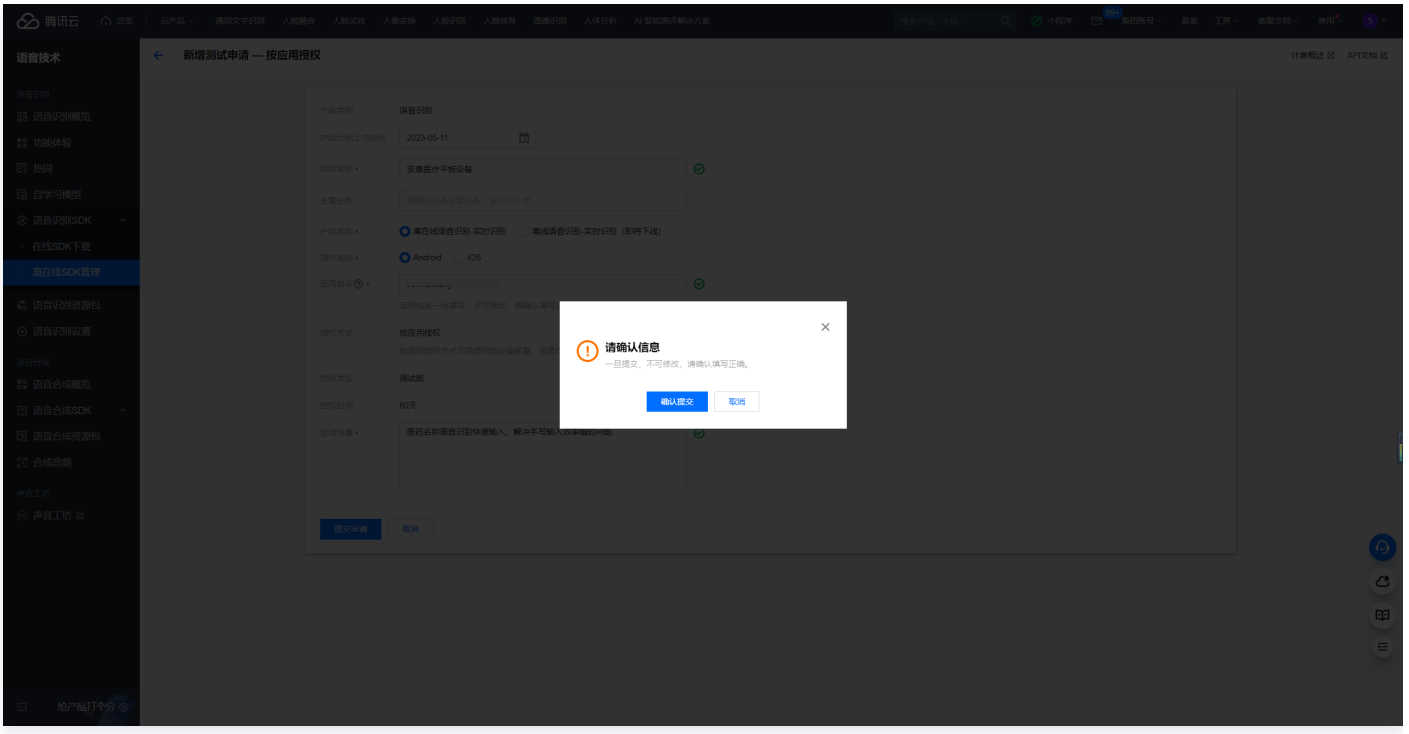
> **注意：**
> 此处用语音识别按**应用授权演示**操作流程（按设备授权或语音合成可同样参考）

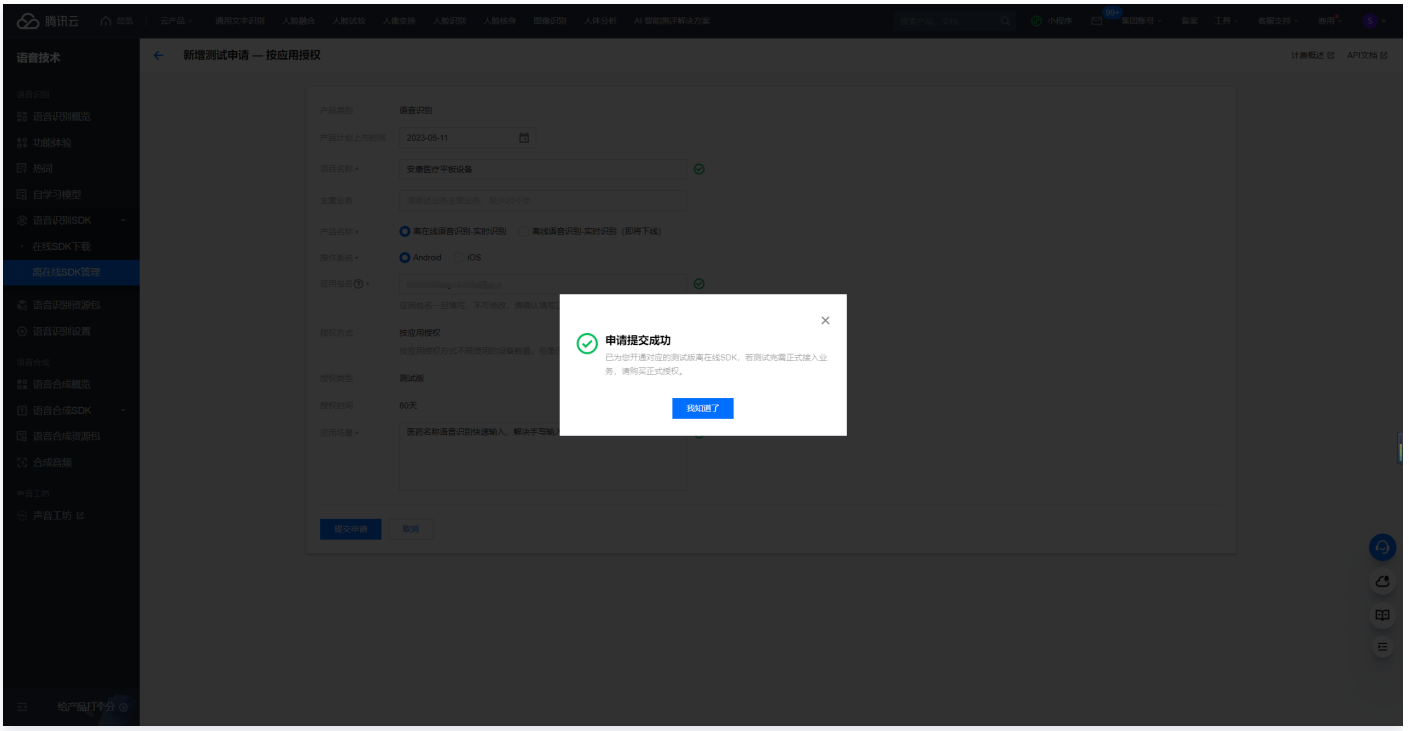


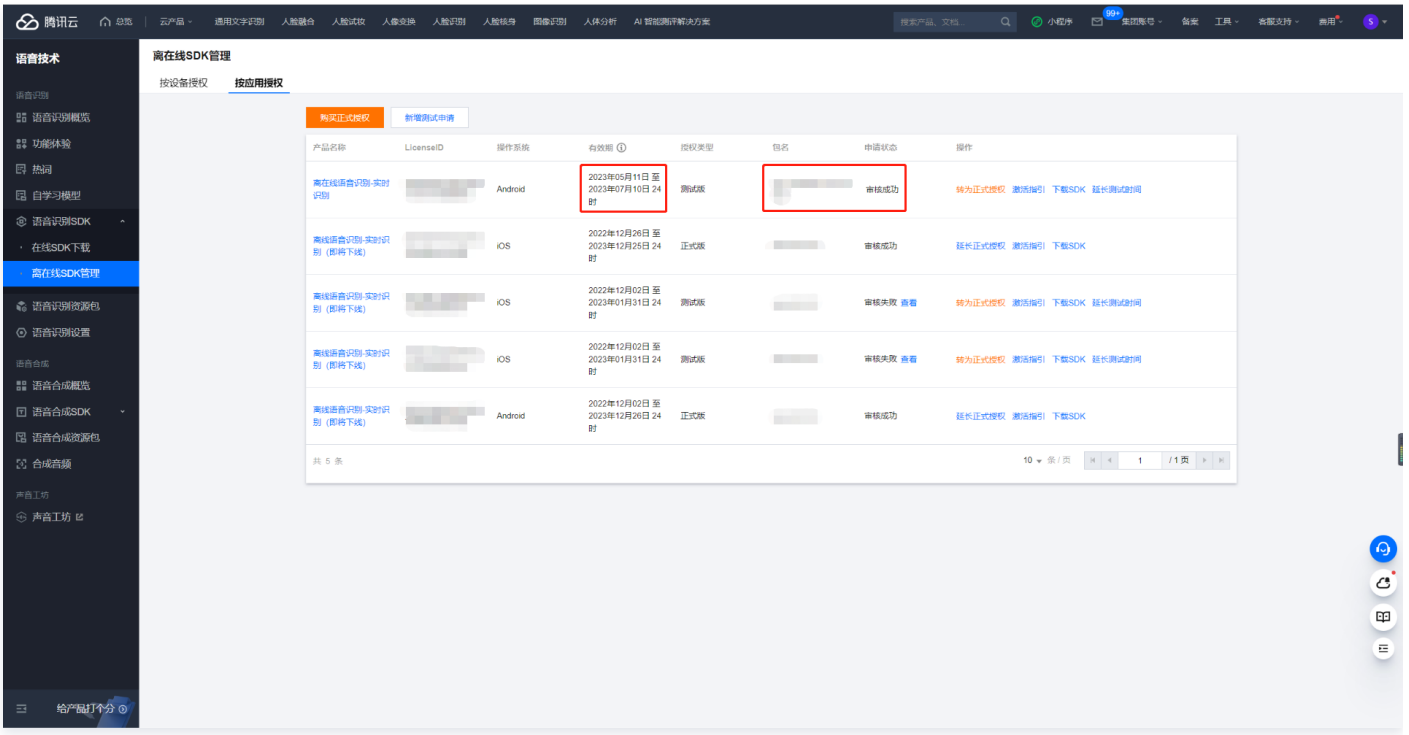
4. 在申请页面填写 SDK 测试所需要的信息，填写**应用包名**命名方式建议参考以下（注意：按设备授权直接选择数量）；

Android-SDK 应用包名：`com.tencent.cloud.sdkasr`
iOS-SDK 应用包名：`com.tencent.cloud.iai.sdkasr`



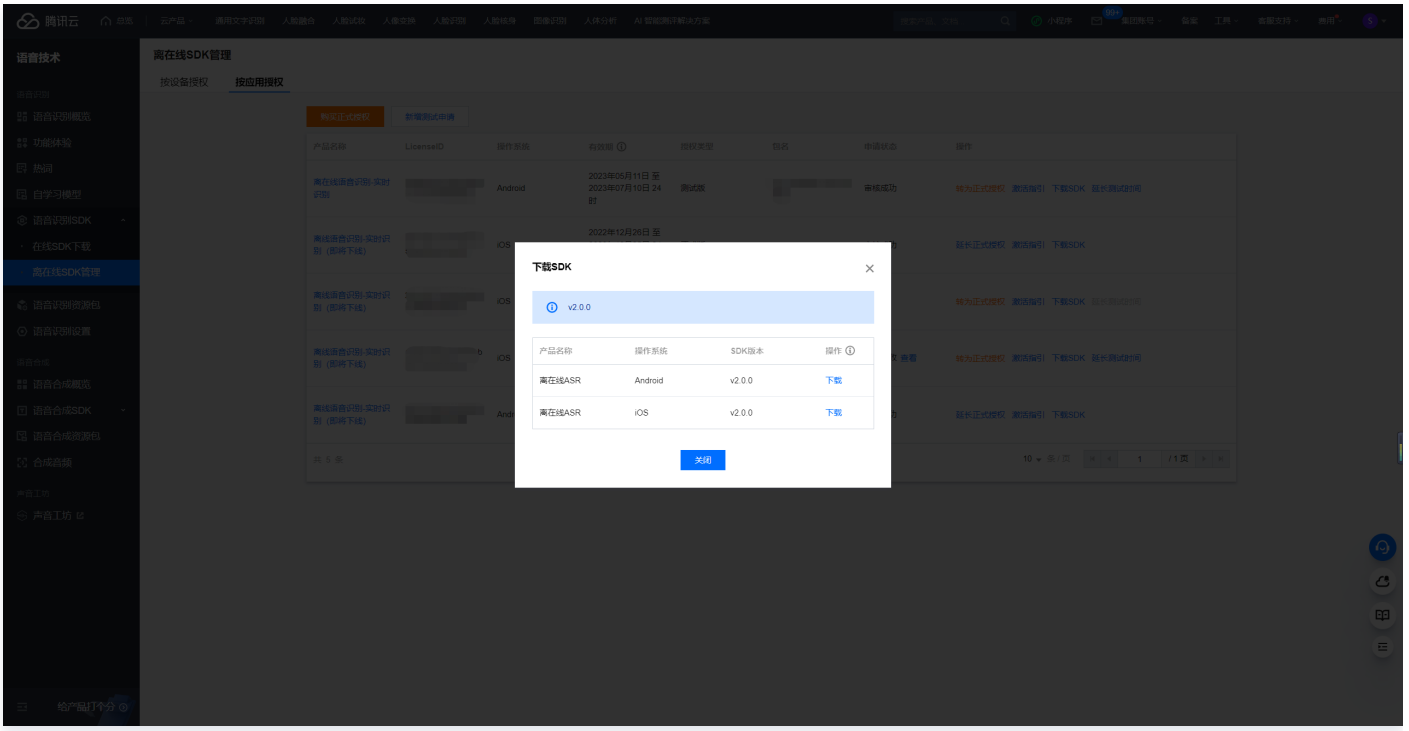




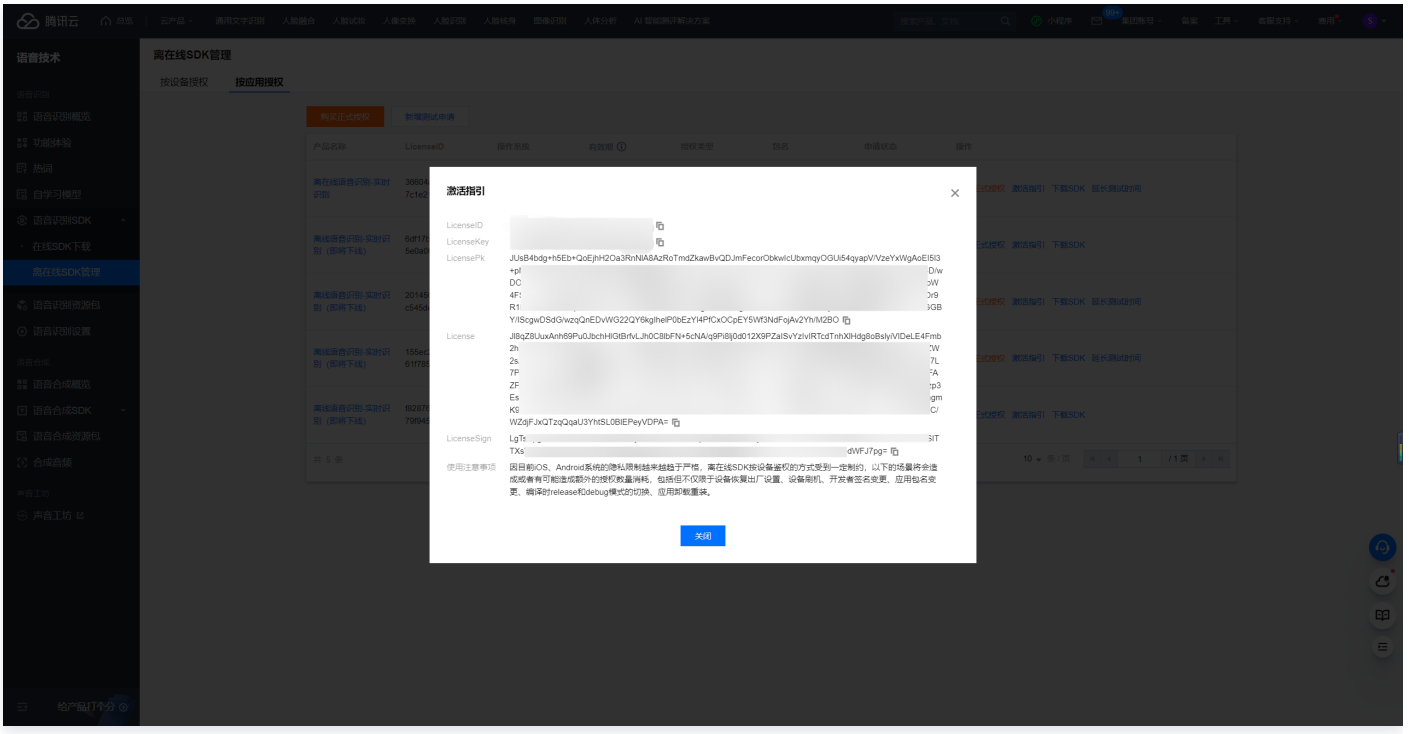


测试下载 SDK、激活指引

1. 单击**下载 SDK**；



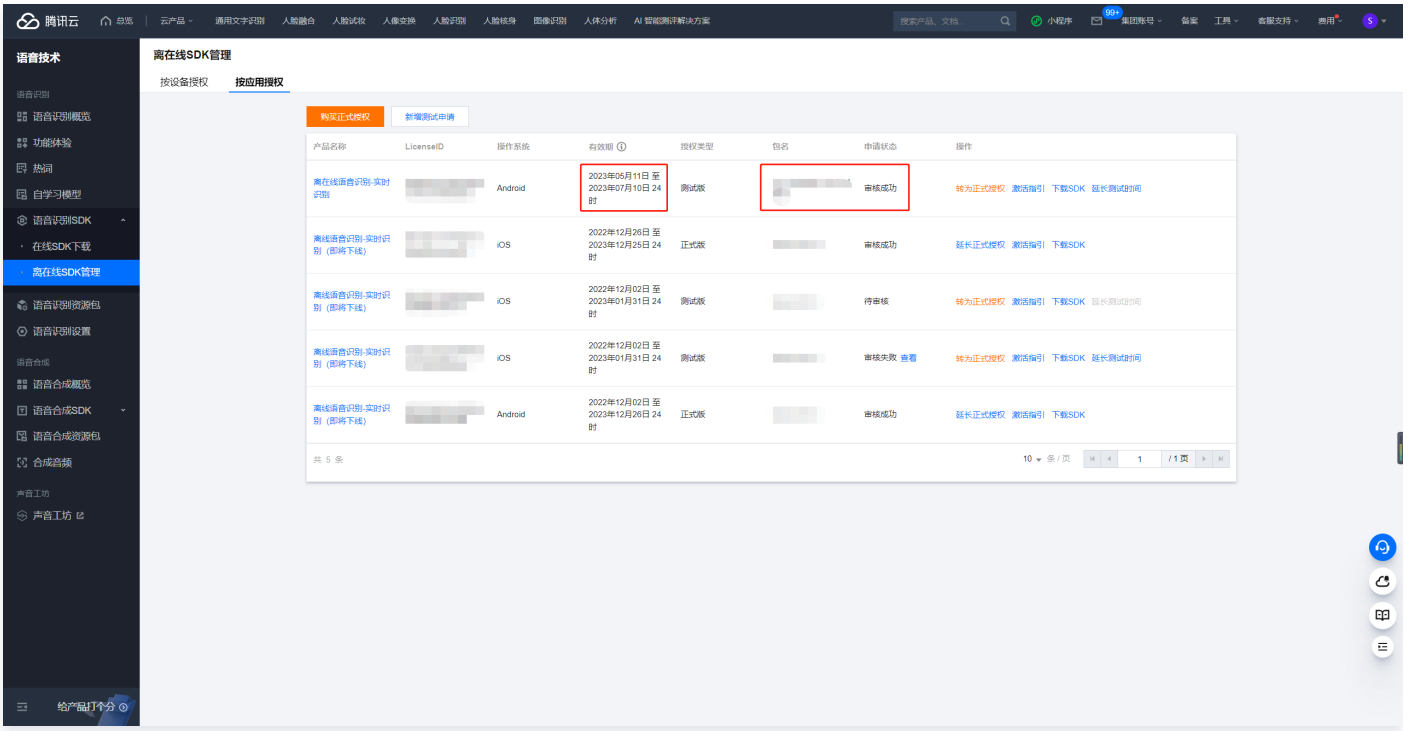
2. 单击**激活指引**；

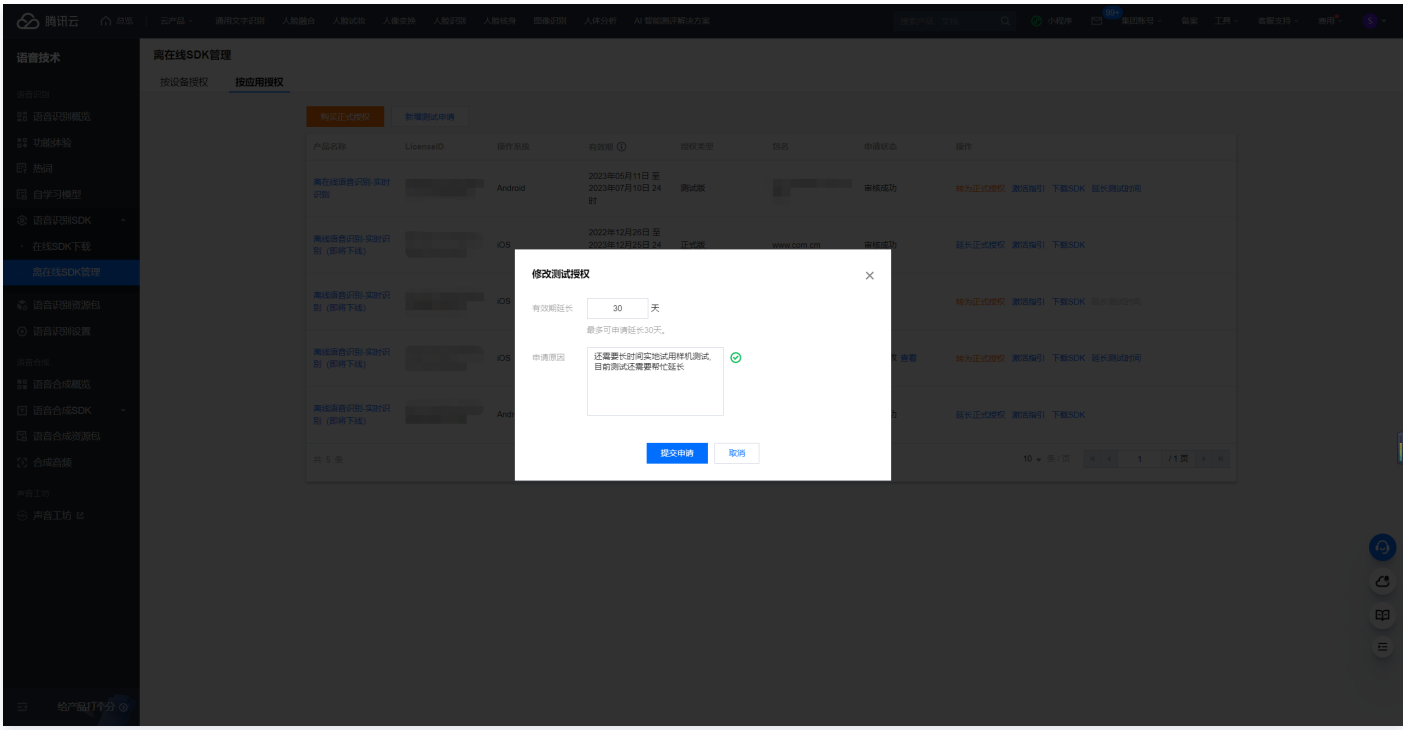


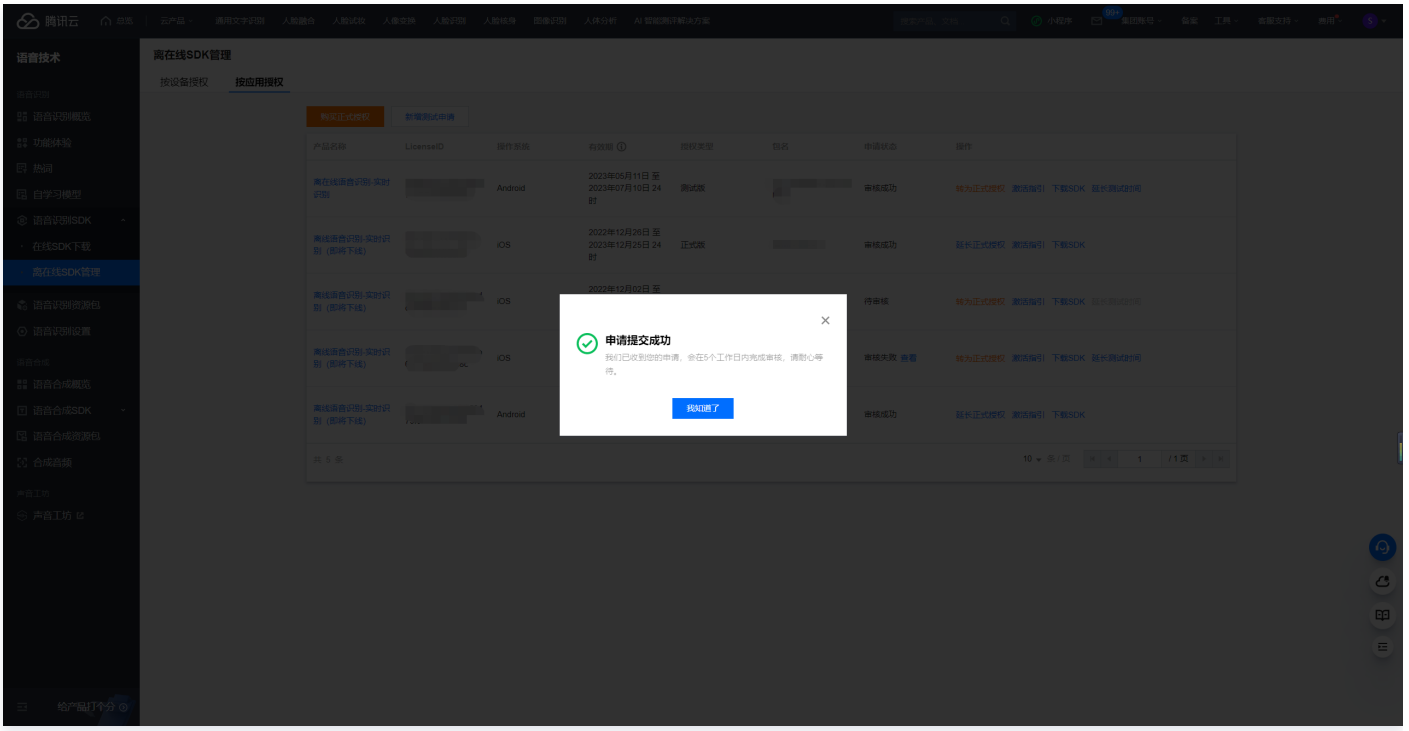
3. 按照**下载 SDK** 压缩包内文档指引，集成 SDK 到 App 后进行密钥输入进行激活。

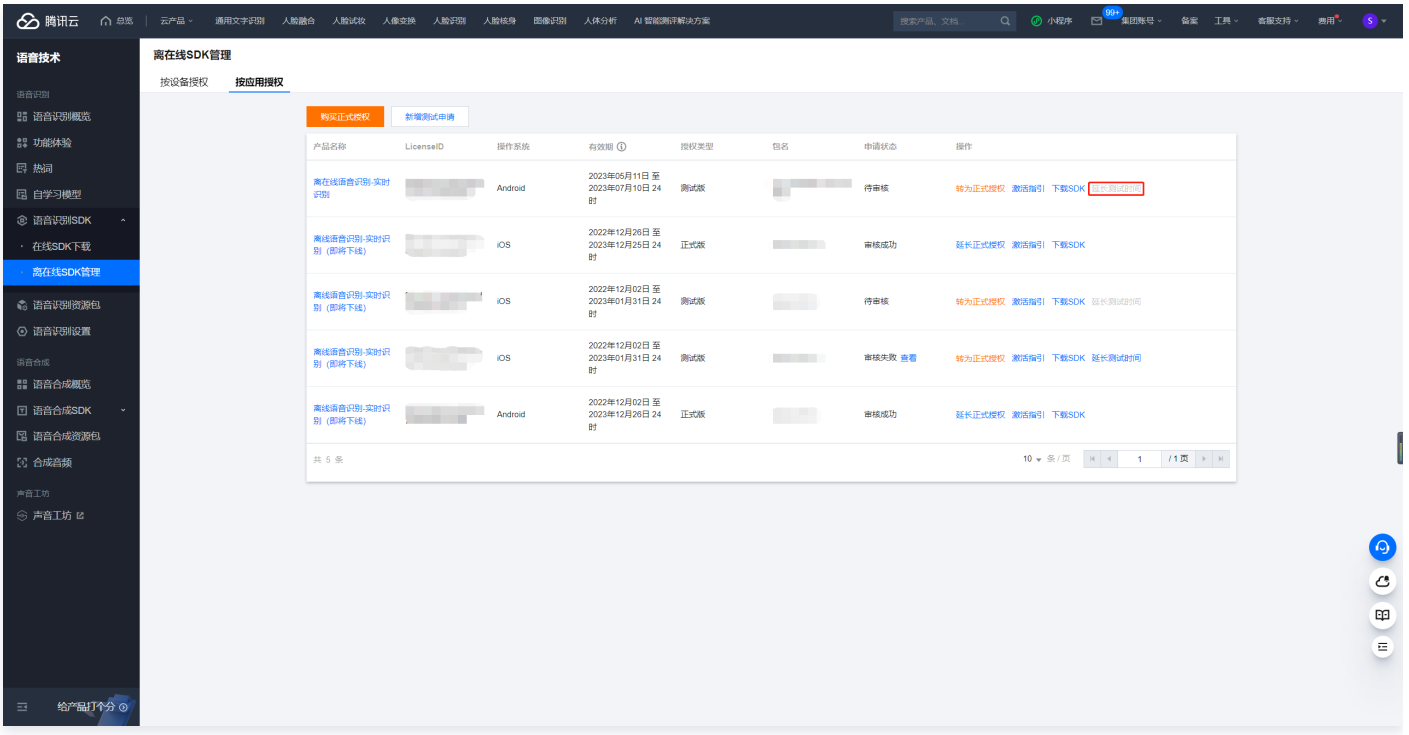
延长测试时间

1. 按应用授权单击**延长测试时间**，或按设备授权单击**修改测试授权**；

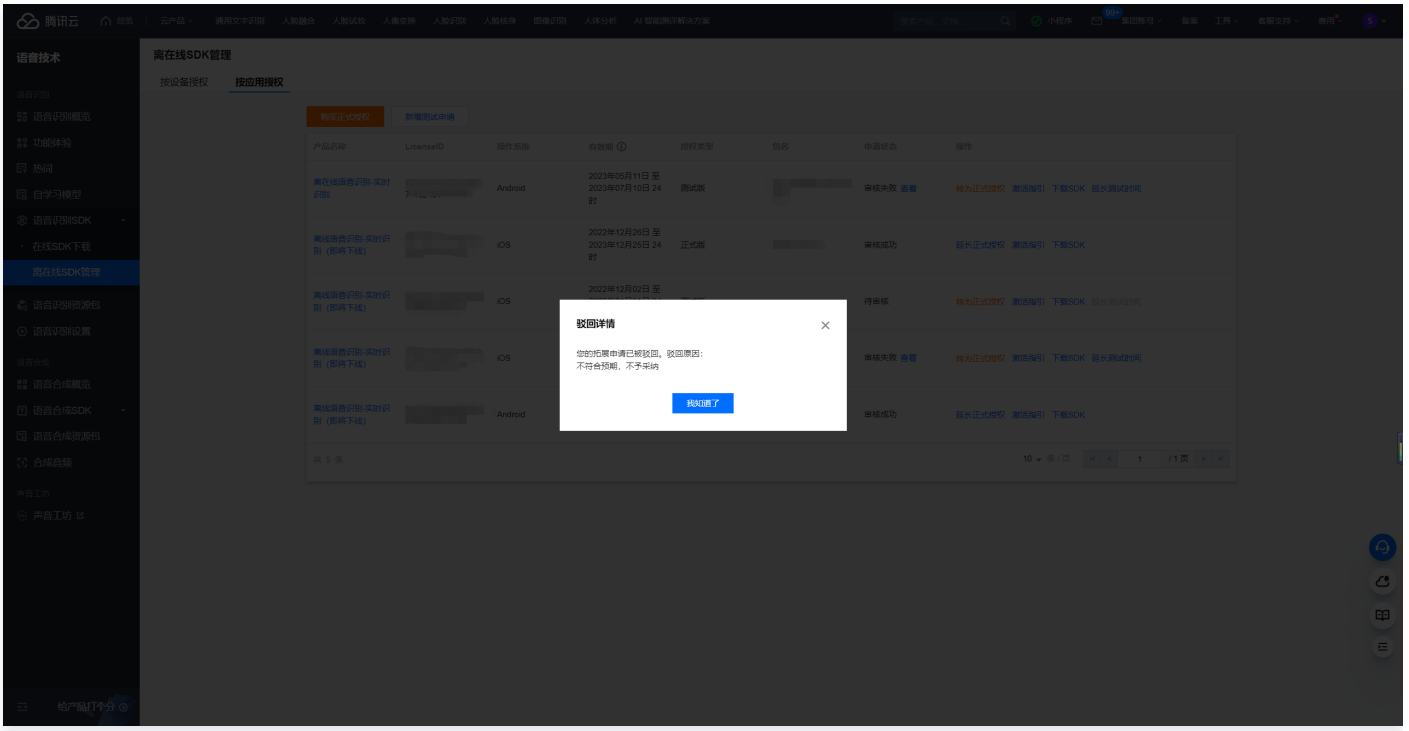





离在线SDK管理
按设备授权
按应用授权
申请提交成功
我们已收到您的申请，会在5个工作日内完成审核，请耐心等待。
我知道了

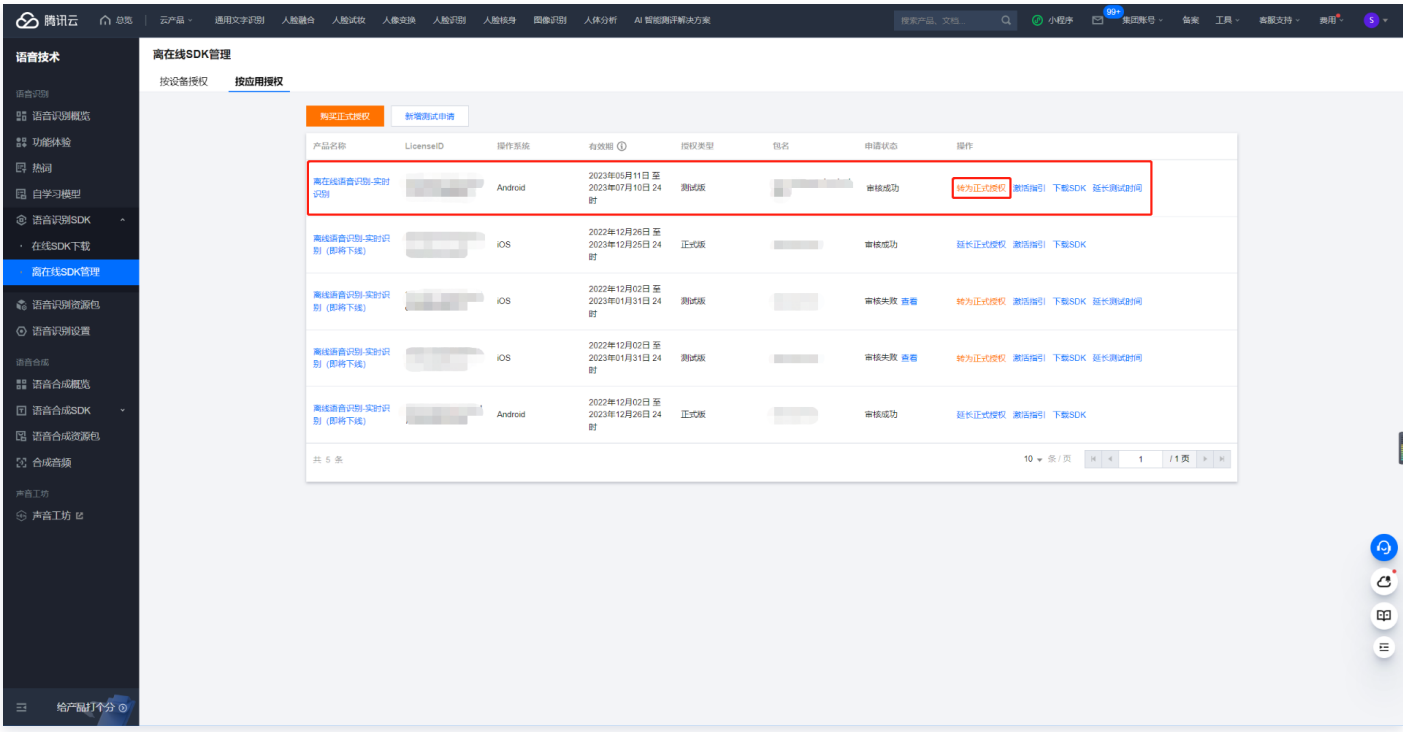

离在线SDK管理
按设备授权
按应用授权
购买正式授权
新增测试申请

2. 待审核通过后，可延长测试时间，若审核失败，单击**查看原因**；

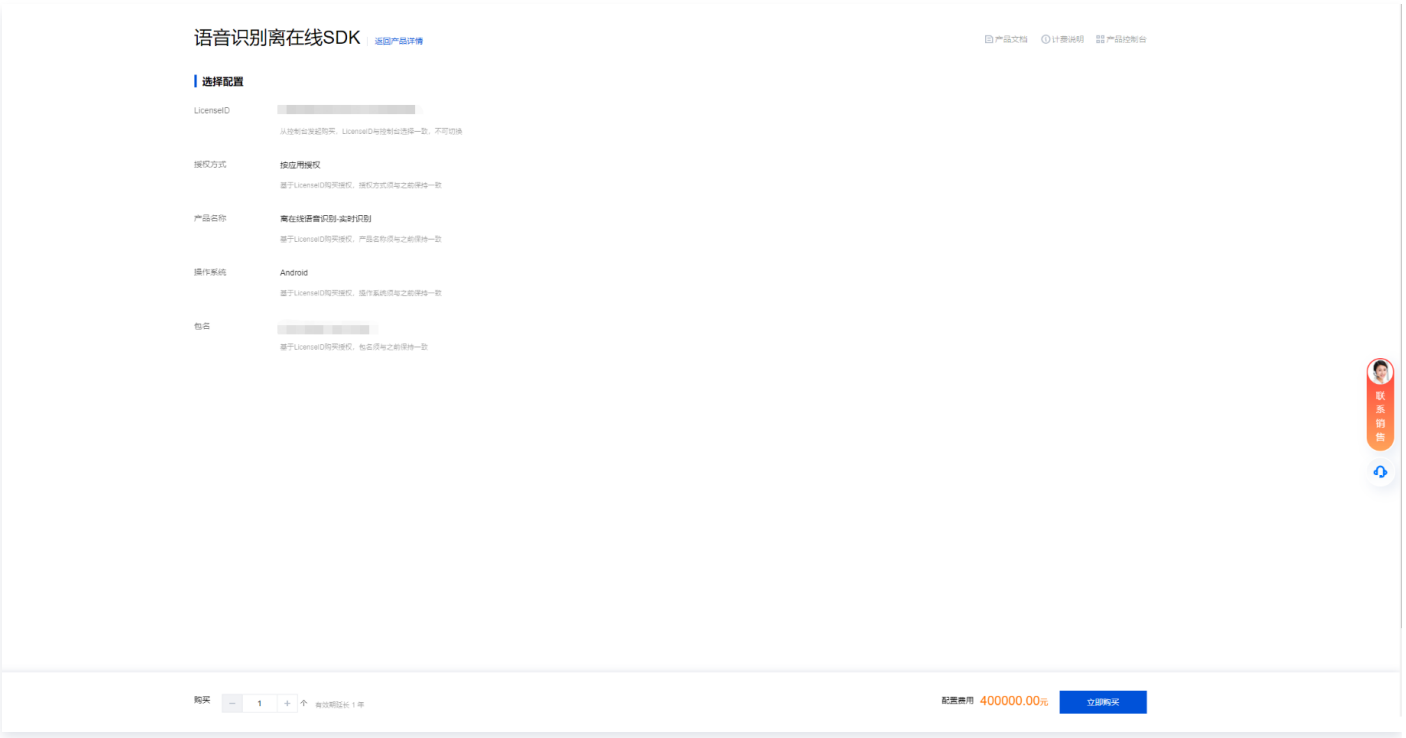


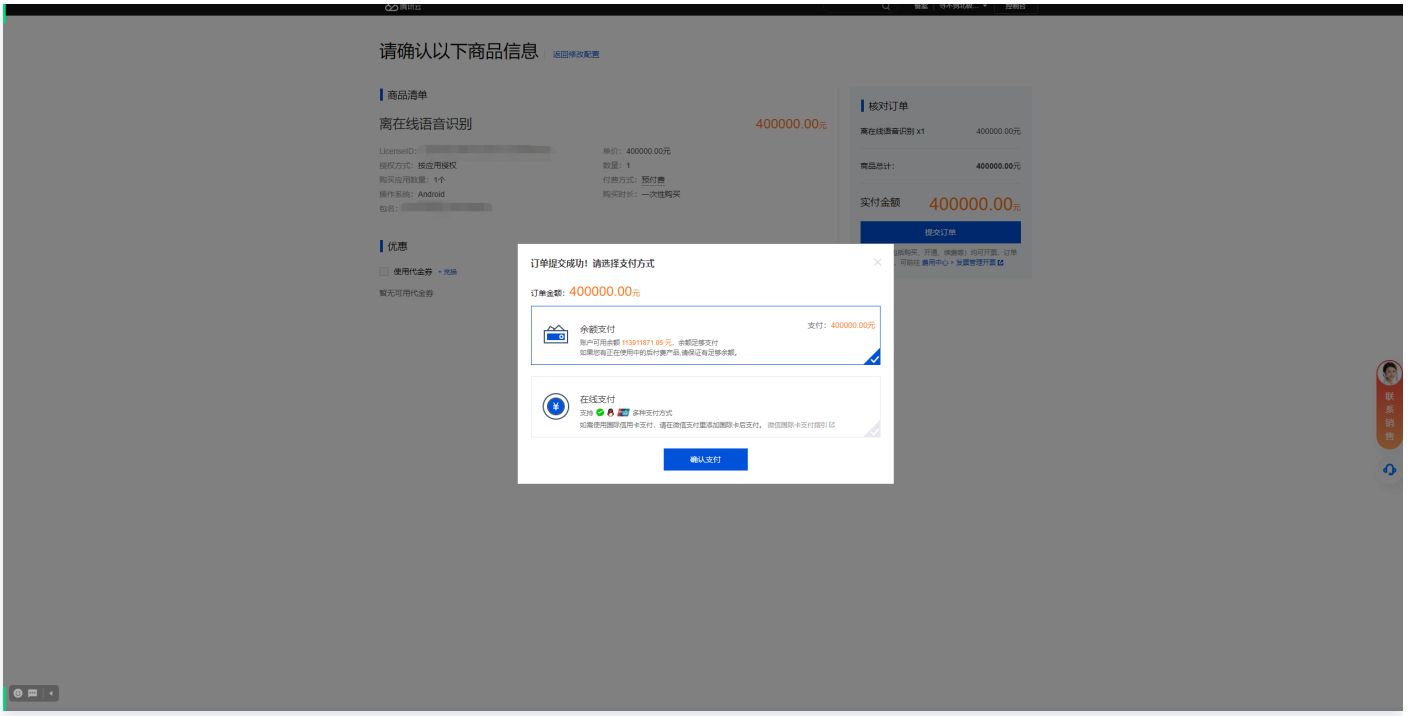
转为正式授权

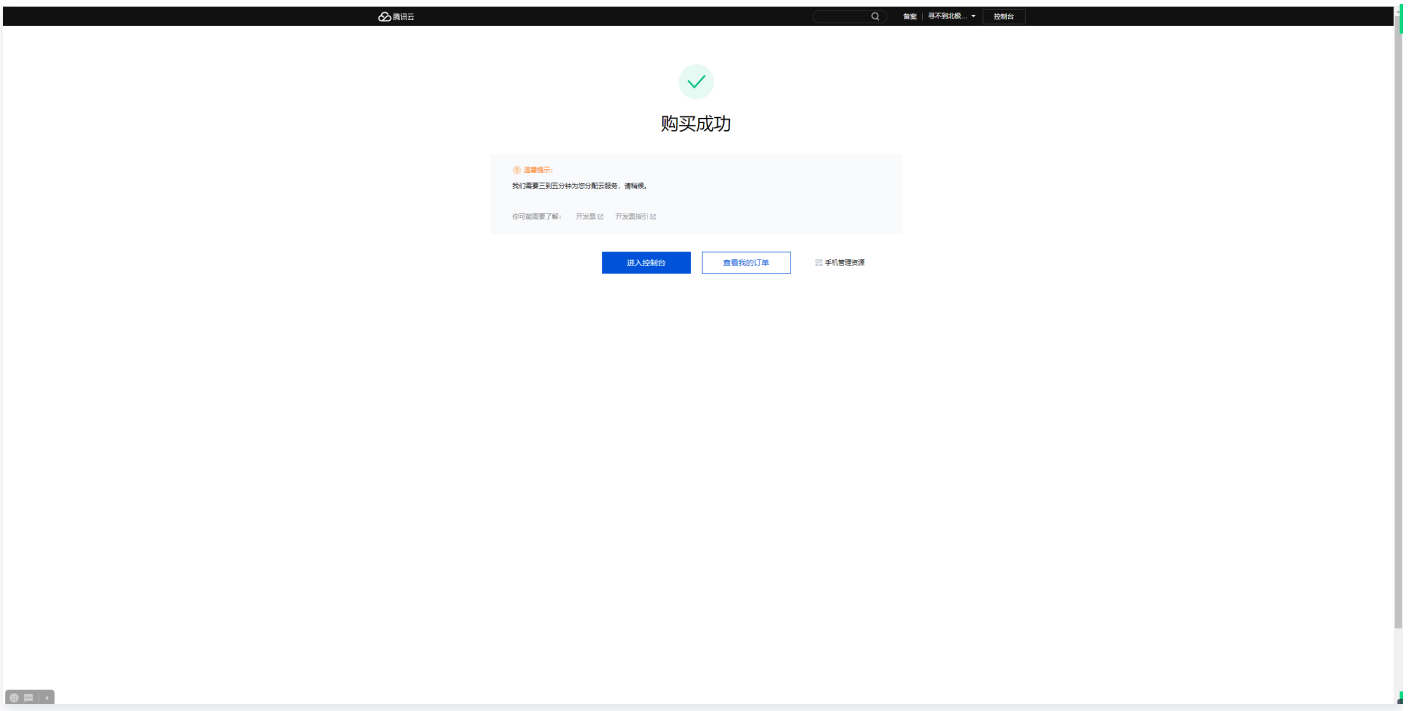
1. 在**离在线 SDK 管理**页单击**转为正式授权**跳到购买页（如不需要测试、或不需要转为正式授权，可直接单击**购买正式授权**，翻到页后看**购买正式授权**流程）；



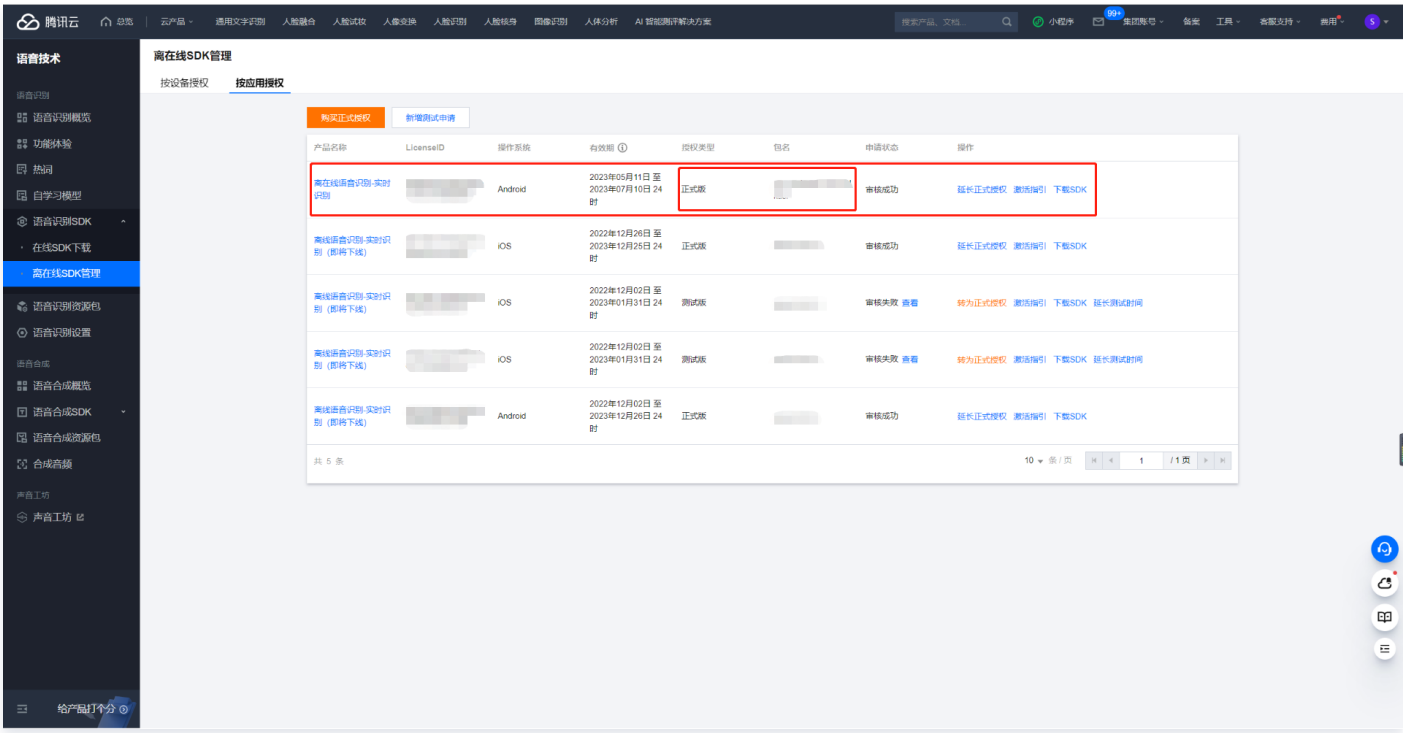
2. 购买页选择**购买XX个**，提交订单并支付成功，单击**进入控制台**；




请确认以下商品信息
返回修改配置
商品清单
离在线语音识别
400000.00元
核对订单
实付金额
400000.00元
提交订单
优惠
订单提交成功！请选择支付方式
订单金额：400000.00元
余额支付
在线支付
确认支付

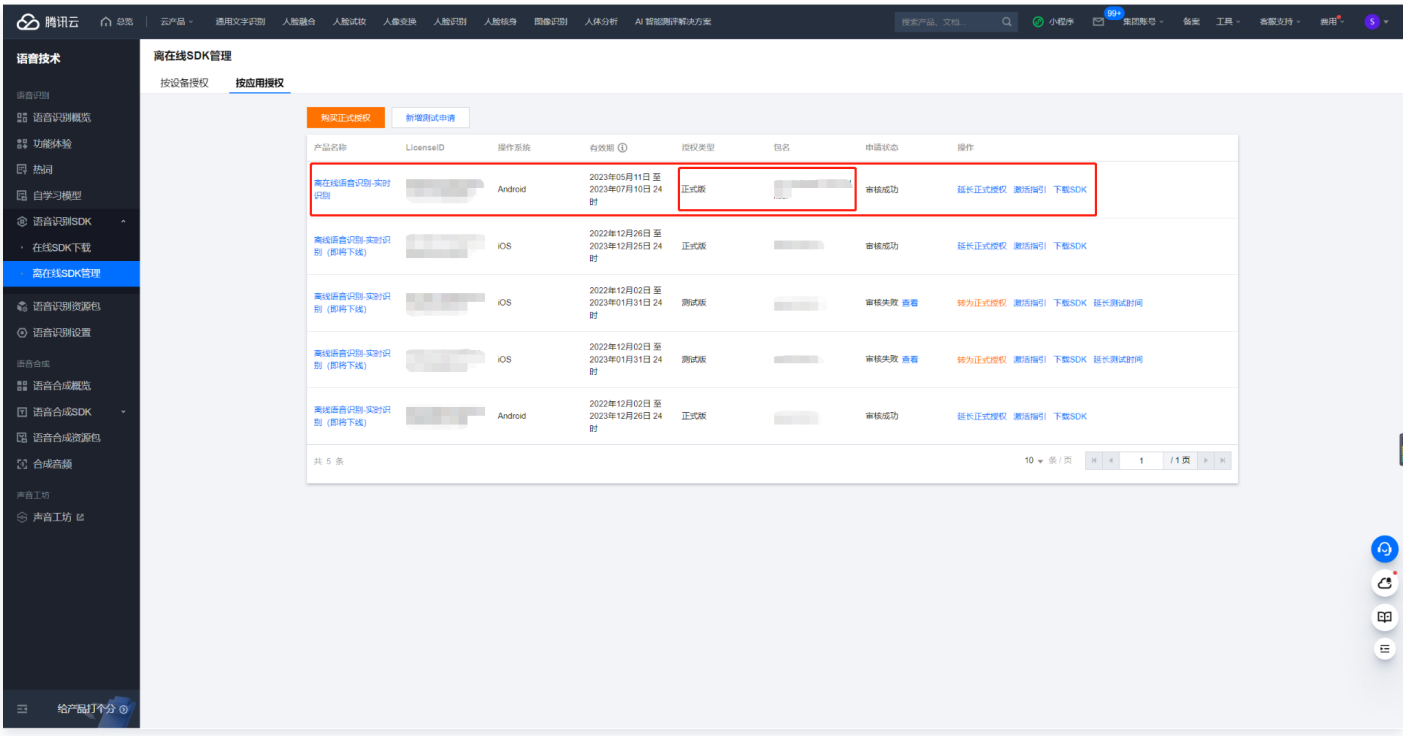

购买成功
进入控制台
查看我的订单

3. 查看**按应用授权**页面，确定**转为正式授权**购买成功。



购买正式授权

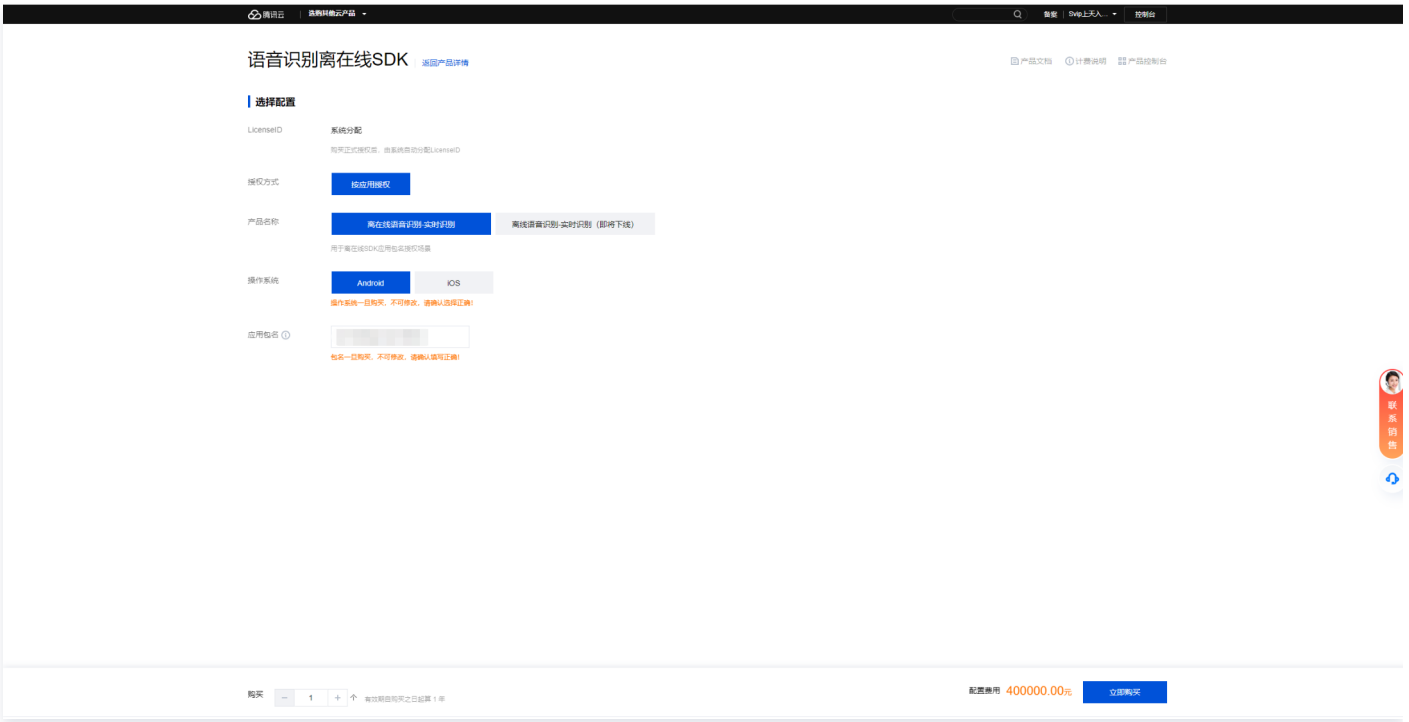
1. 在**离在线 SDK 管理**页，单击**按应用授权**，再单击**购买正式授权**跳到购买页；

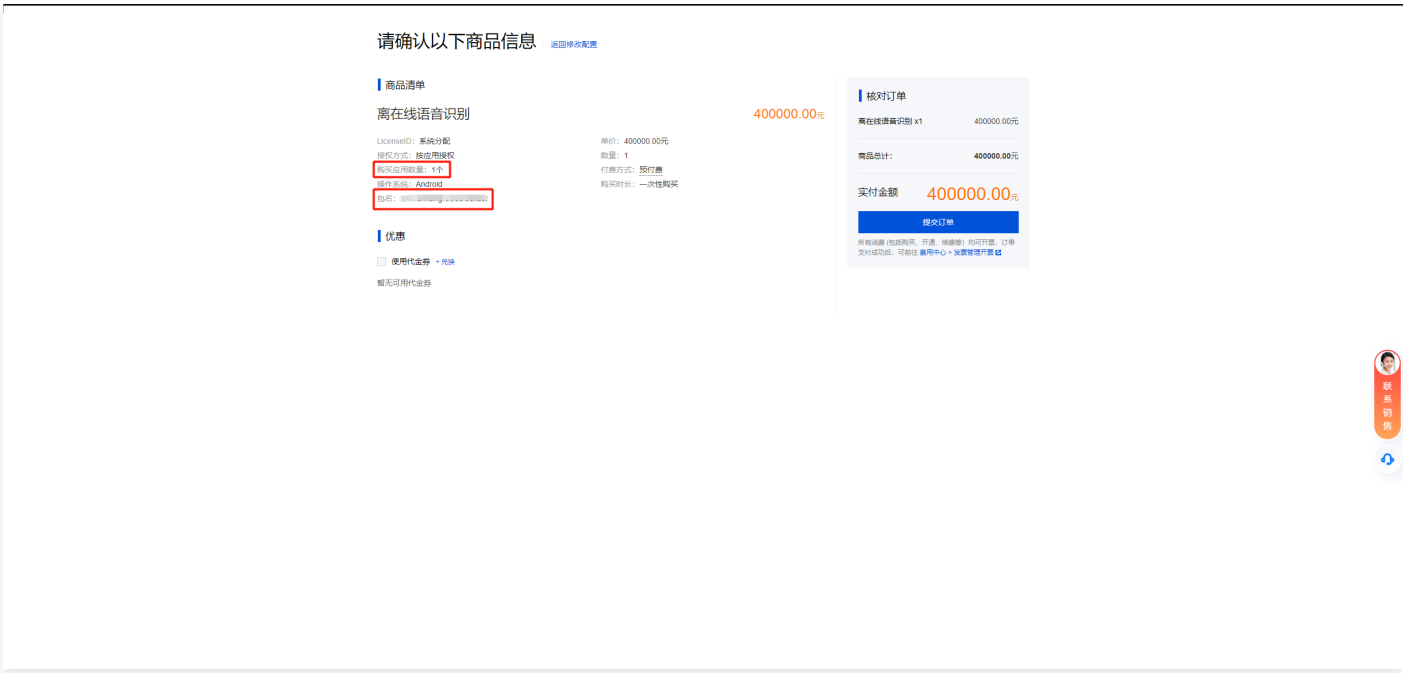


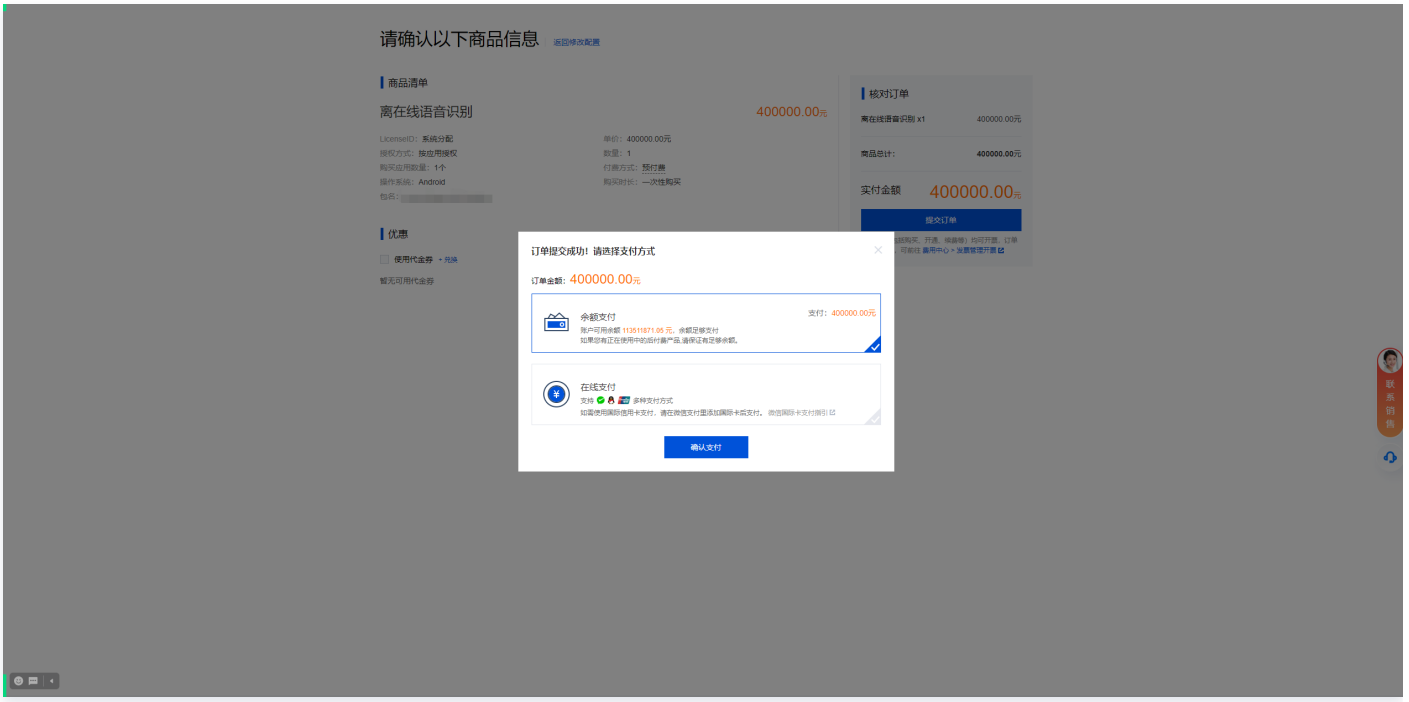
2. 选择购买XX个，填写**应用包名**命名方式建议参考以下（按设备授权直接选择数量），提交订单并支付成功，单击**进入控制台**；

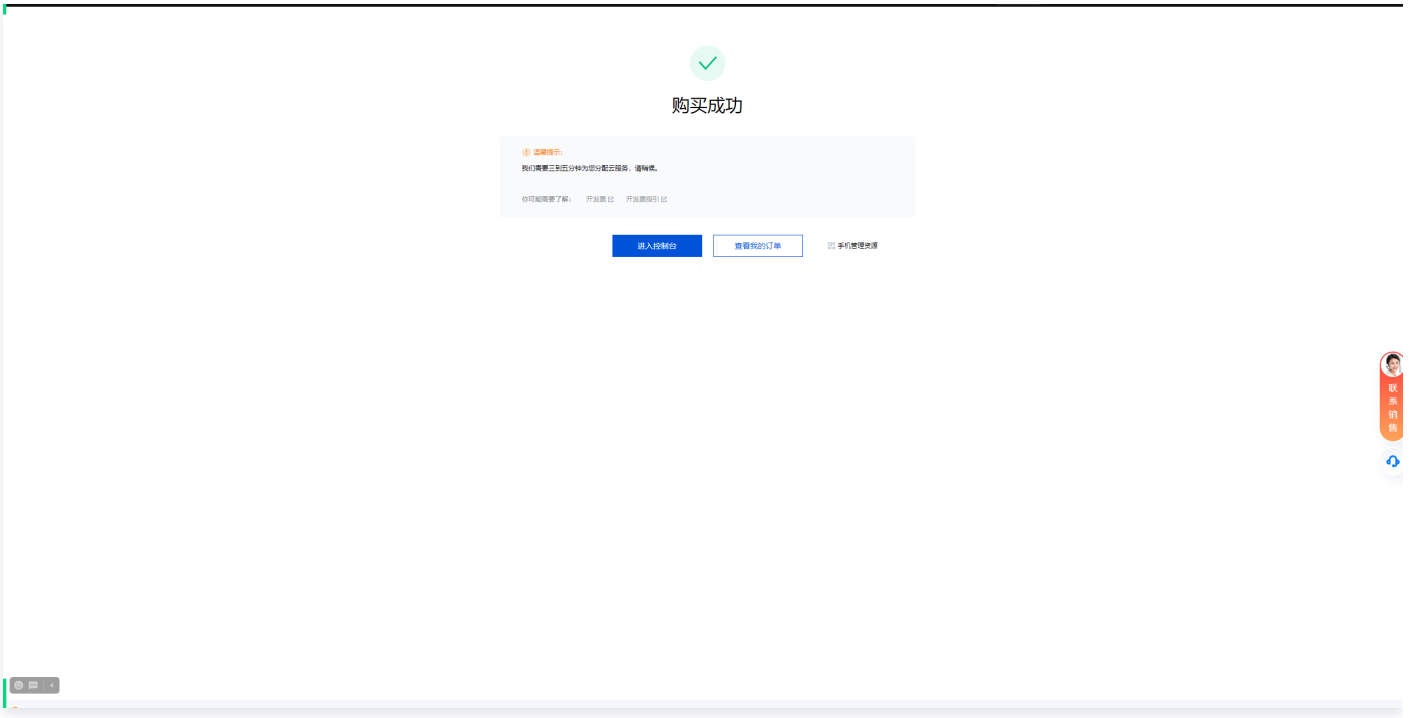
   Android-SDK应用包名：`com.tencent.cloud.sdkasr`

iOS-SDK应用包名： `com.tencent.cloud.iai.sdkasr`

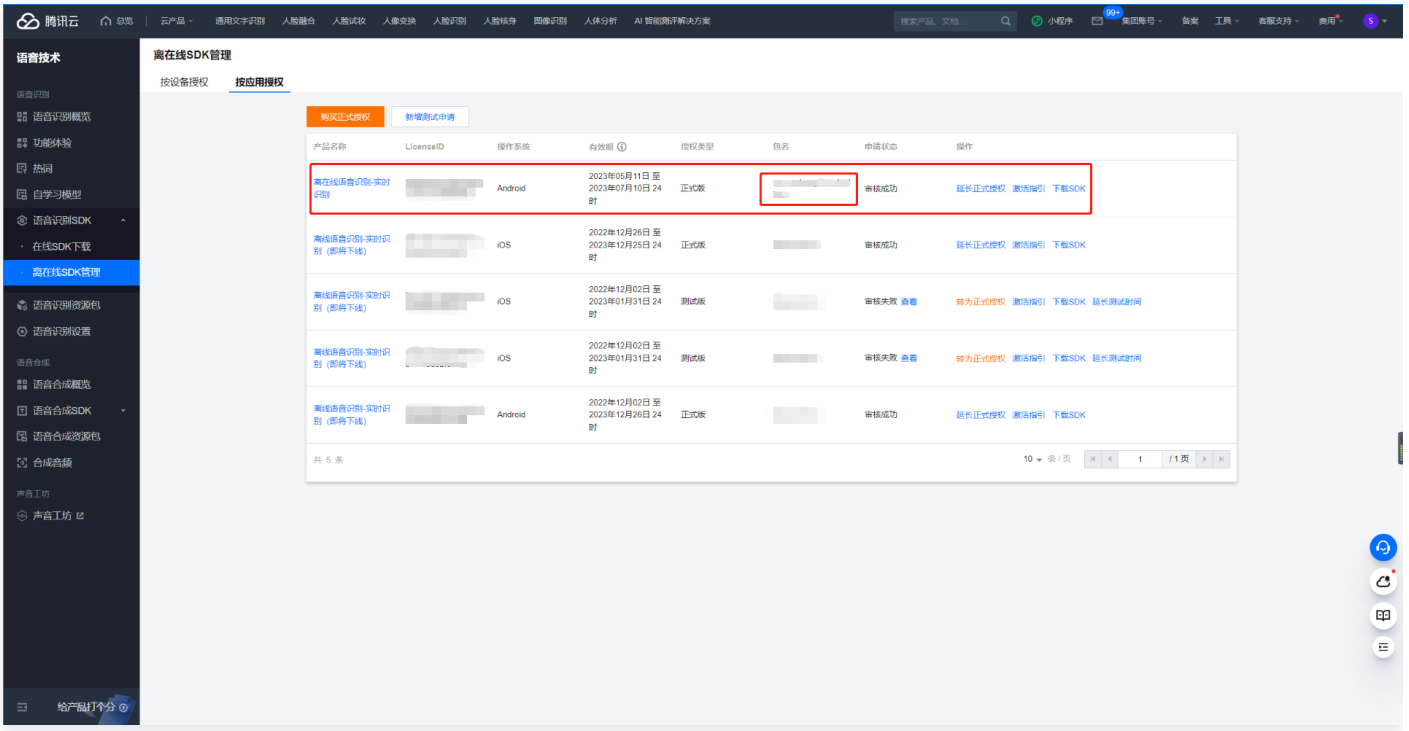





请确认以下商品信息
返回修改配置
商品清单
离在线语音识别
400000.00元
核对订单
离在线语音识别 x1
400000.00元
商品总计:
400000.00元
实付金额
400000.00元
提交订单
优惠
使用代金券
暂无可用代金券
订单提交成功！请选择支付方式
订单金额: 400000.00元
余额支付
支付：400000.00元
在线支付
多种支付方式
确认支付


购买成功
进入控制台
查看我的订单

3. 查看**按应用授权**页面，确定**购买正式授权**购买成功。



Android SDK

最近更新时间：2025-02-12 16:18:32

说明：

- 当前页面为语音识别离在线 SDK 开发文档。新用户可按当前文档接入离在线 SDK。客户可通过当前 SDK 选择使用纯离线版本，或根据网络情况使用自助切换的离在线版本。
- 语音识别离在线 SDK 不仅提供纯离线识别能力，也支持根据网络情况的变化自助切换离线和在线识别版本，从而更好地提升使用体验（请注意，如打开离在线切换开关，则在线识别需要按在线部分独立计费，否则会导致在线部分识别无效，收费标准请见 计费概述（在线版））。
- 我们即将下线离线 SDK 的版本维护和新用户申请，也建议正在使用离线 SDK 的客户及时升级到离在线 SDK，以获取更好的使用体验。

开发准备

- 支持 Android 4.1 以上版本 API LEVEL 16
- 不支持模拟器

硬件要求

- CPU: 支持 armv7,armv8 指令集,主频1.5GHz以上。
- 独占内存: 至少200M
- 独占存储: 至少40M

下载安装 SDK

- SDK 包含 demo 和 SDK 两部分
- 下载语音识别 Android SDK，详情请参见 开通授权 。

简介

SDK 支持离线，在线及混合三种模式。其中 ASRController 为 SDK 的核心类，包含授权设置，参数设置，模式设置和运行控制。使用时需根据业务所需要的模式进行授权设置，参数设置和模式设置(详见接口说明部分)。

运行控制对于三种模式逻辑是一致的，进行识别前需设置数据源和事件监听，start 与 stop 用于控制整体流程，调用 start 后成功开始回调 onStart，失败回调 onError，调用 stop 后成功停止回调 onStop，失败回调 onError，其余回调均为识别过程中的事件消息(详见接口说明部分)。

接口说明

ASRController

语音识别控制类

- **GlobalInstance**

```
static ASRController GlobalInstance()
```

获取语音识别控制类全局实例

- **setMode**

```
void setMode(MODE val)
```

设置控制器模式.支持离线模式、在线模式和混合模式。

- **doAppAuth**

```
ASRControllerError doAppAuth(String license, String licensePk, String licenseSign)
```

按应用授权(仅对离线模式和混合模式生效)

参数

参数名称	描述
license	离线 SDK 授权 License
licensePk	离线 SDK 授权 LicensePk
licenseSign	离线 SDK 授权 LicenseSign

- **doDeviceAuth**

```
ASRControllerError doDeviceAuth(Context context, String secretId, String secretKey, String token,
String licenseKey, String licensePk)
```

按设备授权(仅对离线模式和混合模式生效)

> **说明:**
> **按设备授权会使用到 ANDROID_ID,以下是有关 ANDROID_ID 的说明。**
> 1. **Android 8.0 及以后版本, 如果保证 apk 签名、用户、设备3个条件不变，那么 ANDROID_ID 就不变。**
> 2. **常见变化原因是开发过程中 debug 版 apk 默认使用电脑上 Android SDK 自带的签名文件，这个文件每台电脑都不一样，可能导致 ANDROID_ID 变化超出预期，建议 debug 版也显示指定签名。**

参数

参数名称	描述
context	Android Context
secretId	腾讯云 secret_id，通过 访问管理控制台 获取
secretKey	腾讯云 secret_key，通过 访问管理控制台 获取
token	腾讯云 STS 鉴权 token,使用 STS 临时鉴权时使用，不使用 STS 鉴权传 null
licenseKey	离线 SDK 授权 LicenseKey
licensePk	离线 SDK 授权 LicensePk

- **load**

```
ASRControllerError load(String path, String name)
```

载入模型(仅对离线模式和混合模式生效),需再授权完成后调用。

参数

参数名称	描述
path	模型所在文件目录
name	模型名称

- **unload**

```
ASRControllerError unload()
```

卸载模型(仅对离线模式和混合模式生效),与 load 配对使用。

- **setOnlineAuth**

```
void setOnlineAuth(String appId, String secretId, String secretKey, String token)
```

设置在线认证参数(仅对在线模式和混合模式生效)

参数

参数名称	描述
appId	腾讯云 appID
secretId	腾讯云 secretId
secretKey	腾讯云 secretKey
token	腾讯云临时 token

- **setOnlineParams**

```
void setOnlineParams(String engine_model_type, int needvad, int voice_format, String hotword_id, int
reinforce_hotword, String customization_id, int filter_dirty, int filter_modal, int filter_punc, int
convert_num_mode, int word_info, float noise_threshold)
```

设置在线识别参数(仅对在线模式和混合模式生效)，默认参数请参考 demo 示例。

参数

参数名称	描述
engine_model_type	请参考 API 文档
needvad	请参考 API 文档
voice_format	SDK 仅支持1(pcm)和10(opus),请参考API文档
hotword_id	请参考 API 文档
reinforce_hotword	请参考 API 文档
customization_id	请参考 API 文档
filter_dirty	请参考 API 文档
filter_modal	请参考 API 文档
filter_punc	请参考 API 文档
convert_num_mode	请参考 API 文档
word_info	请参考 API 文档
noise_threshold	请参考 API 文档

- **setOnlineParams**

```
void setCustomOnlineParams(String key,Object value)
```

在线识别用户设置自定义请求参数接口

参数

参数名称	描述
key	自定义参数Key

value	自定义参数Value

- **start**

```
void start()
```

开始识别,请在设置ASRControllerDataSource和ASRControllerListener后调用。

- **stop**

```
void stop()
```

停止识别

ASRControllerError

包含错误码及错误信息

错误码

错误码	返回码	描述
UNKNOWN	-1	未知错误
SUCCESS	0	成功
NETWORK_ERROR	1	网络错误
SERVER_ERROR	2	服务器错误
ENGINE_INIT_ERROR	3	引擎初始化错误
ENGINE_AUTH_ERROR	4	引擎认证错误
DATASOURCE_INVALID	5	数据源错误
STATE_ERROR	6	状态错误
ENGINE_ERROR	7	引擎错误
ONLINE_AUTH_ERROR	8	在线认证错误

ASRControllerListener

用于同步识别过程中的事件

onBegin，onSlice，onSegment 与一段话识别有关，调用有以下几种情况。

1. onBegin->onSlice->onSegment
2. onBegin->onSegment
3. onSegment

- **onBegin**

```
void onBegin(String extra)
```

一段话开始识别

参数

参数名称	描述
extra	在线模式及混合模式处于在线状态时为服务端返回信息,请参考API文档

- **onSlice**

```
void onSlice(String val, String extra)
```

一段话开始识别中

参数

参数名称	描述
val	非稳态识别结果
extra	在线模式及混合模式处于在线状态时为服务端返回信息,请参考API文档

- **onSegment**

```
void onSegment(String val, String extra)
```

一段话开始识别结束

参数

参数名称	描述
val	稳态识别结果
extra	在线模式及混合模式处于在线状态时为服务端返回信息,请参考API文档

- **onStart**

```
void onStart(String extra)
```

识别任务开始

参数

参数名称	描述
extra	在线模式及混合模式处于在线状态时为服务端返回信息,请参考API文档

- **onError**

```
void onError(ASRControllerError val)
```

识别任务结束

参数

参数名称	描述
val	错误信息

- **onStop**

```
void onStop()
```

识别任务停止

- **onSwitch**

```
void onSwitch(boolean is_online)
```

模式切换(仅混合)

参数

参数名称	描述
is_online	true 为在线状态,false 为离线状态

ASRControllerDataSource

提供16000Hz，16bit(le)，pcm 格式，单声道的数据源用于识别。

- **read**

```
long read(ByteBuffer data)
```

参数

参数名称	描述
ByteBuffer	传入数据

返回实际写入长度

iOS SDK

最近更新时间：2024-12-12 22:17:32

说明：

- 当前页面为语音识别离在线 SDK 开发文档。新用户可按当前文档接入离在线 SDK。客户可通过当前 SDK 选择使用纯离线版本，或根据网络情况使用自助切换的离在线版本。
- 语音识别离在线 SDK 不仅提供纯离线识别能力，也支持根据网络情况的变化自助切换离线和在线识别版本，从而更好地提升使用体验，请注意，如打开离在线切换开关，则在线识别需要按在线部分独立计费，否则会导致在线部分识别无效，收费标准请见 计费概述（在线版）。
- 我们即将下线离线 SDK 的版本维护和新用户申请，也建议正在使用离线 SDK 的客户及时升级到离在线 SDK，以获取更好的使用体验。

开发准备

- 不支持 iOS 11.0 以下版本
- 仅支持 iOS arm64 架构
- 不支持模拟器

硬件要求

- CPU: 支持 armv7,armv8 指令集,主频 1.5GHz 以上。
- 独占内存: 至少100M
- 独占存储: 至少30M

下载安装 SDK

- SDK 包含 demo 和 SDK 两部分
- demo 仅支持 iOS 13.0 以上

简介

SDK 支持离线，在线及混合三种模式.其中 ASRController 为 SDK 的核心类，包含授权设置，参数设置，模式设置和运行控制。使用时需根据业务所需要的模式进行授权设置，参数设置和模式设置(详见接口说明部分)。

运行控制对于三种模式逻辑是一致的，进行识别前需设置数据源和事件监听，start 与stop 用于控制整体流程，调用 start 后成功开始回调 onStart，失败回调 onError，调用 stop 后成功停止回调 onStop，失败回调 onError，其余回调均为识别过程中的事件消息(详见接口说明部分)。

配置隐私协议

根据苹果隐私政策规定，集成 SDK 的应用需要在 Xcode 项目的 PrivacyInfo.xcprivacy 中补全条款，若项目中没有，需要根据 官方说明 使用 Xcode 15 及以上的版本，新建一个 PrivacyInfo.xcprivacy 文件，参考以下方式，添加 SDK 依赖的隐私条款：

1. 在 Xcode 中选择 PrivacyInfo.xcprivacy。
2. 将 `Privacy Accessed API Types` 条款添加到 PrivacyInfo.xcprivacy 中，具体配置如下图：

Key	Type	Value
App Privacy Configuration	Dictionary	(1 item)
Privacy Accessed API Types	Array	(1 item)
Item 0	Dictionary	(2 items)
Privacy Accessed API Reasons	Array	(1 item)
Item 0	String	35F9.1: Measure time on-device, per documentation
Privacy Accessed API Type	String	System Boot Time

接口说明

ASRController

语音识别控制类

- GlobalInstance

```
+(instancetype)GlobalInstance;
```

获取语音识别控制类全局实例

- **setMode**

```
-(void)setMode:(MODE)mode;
```

设置控制器模式.支持离线模式、在线模式和混合模式。

- **doAppAuth**

```
-(ASRControllerError*)doAppAuth:(NSString*)lic  licPk:(NSString*)licPk authSign:(NSString*)authSign;
```

按应用授权(仅对离线模式和混合模式生效)

参数

参数名称	描述
lic	离线 SDK 授权 License
licPk	离线 SDK 授权 LicensePk
authSign	离线 SDK 授权 LicenseSign

- **doDeviceAuth**

```
-(ASRControllerError*)doDeviceAuth:(NSString*)secretId secretKey:(NSString*)secretKey token:
(NSString*)token licKey:(NSString*)licKey  licPk:(NSString*)licPk;
```

按设备授权(仅对离线模式和混合模式生效)

参数

参数名称	描述
secretId	腾讯云 secret_id，通过 访问管理控制台 获取
secretKey	腾讯云 secret_key，通过 访问管理控制台 获取
token	腾讯云 STS 鉴权 token,使用 STS 临时鉴权时使用，不使用 STS 鉴权传 null
licKey	离线 SDK 授权 LicenseKey
licPk	离线 SDK 授权 LicensePk

- **load**

```
-(ASRControllerError*)load:(NSString*)path name:(NSString*)name;
```

载入模型(仅对离线模式和混合模式生效),需再授权完成后调用。

参数

参数名称	描述
path	模型所在文件目录
name	模型名称

- **unload**

```
-(ASRControllerError*)unload;
```

卸载模型(仅对离线模式和混合模式生效),与load配对使用。

- **setOnlineAuth**

```
-(void)setOnlineAuth:(NSString*)appId secretId:(NSString*)secretId secretKey:(NSString*)secretKey
token:(NSString*)token;
```

设置在线认证参数(仅对在线模式和混合模式生效)

参数

参数名称	描述
appId	腾讯云 appID
secretId	腾讯云 secretId
secretKey	腾讯云 secretKey
token	腾讯云临时 token

- **setOnlineParams**

```
-(void)setOnlineParams:(NSString*)engine_model_type needvad:(NSInteger)needvad
          voice_format:(NSInteger)voice_format
            hotword_id:(NSString*)hotword_id
    reinforce_hotword:(NSInteger)reinforce_hotword
     customization_id:(NSString*)customization_id
         filter_dirty:(NSInteger)filter_dirty
         filter_modal:(NSInteger)filter_modal
          filter_punc:(NSInteger)filter_punc
     convert_num_mode:(NSInteger)convert_num_mode
            word_info:(NSInteger)word_info
     vad_silence_time:(NSInteger)vad_silence_time
      noise_threshold:(float)noise_threshold;
```

设置在线识别参数(仅对在线模式和混合模式生效),默认参数请参考demo示例。

参数

参数名称	描述
engine_model_type	请参考 API 文档
needvad	请参考 API 文档
voice_format	SDK 仅支持1(pcm)和10(opus),请参考 API 文档
hotword_id	请参考 API 文档
reinforce_hotword	请参考 API 文档
customization_id	请参考 API 文档
filter_dirty	请参考 API 文档
filter_modal	请参考 API 文档
filter_punc	请参考 API 文档

convert_num_mode	请参考 API 文档
word_info	请参考 API 文档
vad_silence_time	传入小于等于0 SDK忽略该参数,请参考 API 文档
noise_threshold	请参考 API 文档

- **start**

```
-(void)start;
```

开始识别,请在设置ASRControllerDataSource和ASRControllerListener后调用。

- **stop**

```
-(void)stop;
```

停止识别

ASRControllerError

包含错误码及错误信息
错误码

错误码	返回码	描述
UNKNOWN	-1	未知错误
SUCCESS	0	成功
NETWORK_ERROR	1	网络错误
SERVER_ERROR	2	服务器错误
ENGINE_INIT_ERROR	3	引擎初始化错误
ENGINE_AUTH_ERROR	4	引擎认证错误
DATASOURCE_INVALID	5	数据源错误
STATE_ERROR	6	状态错误
ENGINE_ERROR	7	引擎错误
ONLINE_AUTH_ERROR	8	在线认证错误

ASRControllerListener

用于同步识别过程中的事件

onBegin，onSlice，onSegment 与一段话识别有关,调用有以下几种情况。
1. onBegin->onSlice->onSegment
2. onBegin->onSegment
3. onSegment

- **onBegin**

```
-(void)onBegin:(NSString*)extra;
```

一段话开始识别

参数

参数名称	描述
extra	在线模式及混合模式处于在线状态时为服务端返回信息,请参考 API 文档

- **onSlice**

```
-(void)onSlice:(NSString*)val extra:(NSString*)extra;
```

一段话开始识别中

参数

参数名称	描述
val	非稳态识别结果
extra	在线模式及混合模式处于在线状态时为服务端返回信息,请参考 API 文档

- **onSegment**

```
-(void)onSegment:(NSString*)val extra:(NSString*)extra;
```

一段话开始识别结束

参数

参数名称	描述
val	稳态识别结果
extra	在线模式及混合模式处于在线状态时为服务端返回信息,请参考 API 文档

- **onStart**

```
-(void)onStart:(NSString*)extra;
```

识别任务开始

参数

参数名称	描述
extra	在线模式及混合模式处于在线状态时为服务端返回信息,请参考API文档

- **onError**

```
-(void)onError:(ASRControllerError*)val;
```

识别任务结束

参数

参数名称	描述
val	错误信息

- **onStop**

```
-(void)onStop;
```

识别任务停止

- **onSwitch**

```
-(void)onSwitch:(bool)is_online;
```

模式切换(仅混合)

参数

参数名称	描述
is_online	true 为在线状态，false 为离线状态

ASRControllerDataSource

提供16000Hz，16bit(le)，pcm格式，单声道的数据源用于识别。

- **read**

```
-(size_t) read:(short*)data len:(size_t)len;
```

参数

参数名称	描述
data	传入缓存
len	缓存长度

返回实际写入长度

离在线语音识别 SDK 个人信息保护规则

最近更新时间：2023-05-15 16:57:14

引言

离在线语音识别 SDK（以下简称“SDK 产品”）由腾讯云计算（北京）有限责任公司以下简称（“我们”）开发，公司注册地为北京市海淀区西北旺东路10号院西区9号楼4层101。
《离在线语音识别 SDK 个人信息保护规则》（以下简称“本规则”）主要向开发者及其终端用户（“终端用户”）告知，为了实现 SDK 产品的相关功能，SDK 产品需收集、使用和处理终端用户个人信息的情况。
请开发者及终端用户认真阅读本规则。如您是开发者，请您确认充分了解并同意本规则后再集成 SDK 产品，**如果您不同意本规则及按照本规则履行对应的用户个人信息保护义务，应立即停止接入及使用 SDK 产品；同时，您应仅在征得终端用户的同意后集成 SDK 产品并处理终端用户的个人信息，在获得终端用户同意前不得启用或初始化本 SDK 产品。**

特别说明

如您是开发者，您应当：

1. **遵守法律、法规收集、使用和处理终端用户的个人信息，包括但不限于制定和公布有关个人信息保护的隐私政策等；**
2. **在集成 SDK产品前，告知终端用户 SDK 产品收集、使用和处理终端用户个人信息的情况，并依法征得终端用户同意；**
3. **在征得终端用户的同意前，除非法律法规另有规定，不应收集任何终端用户的个人信息；**
4. **向终端用户提供易于操作且满足法律法规要求的用户权利实现机制，并告知终端用户如何查阅、复制、修改、删除个人信息，撤回同意，以及限制个人信息处理、转移个人信息、获取个人信息副本和注销账号；**
5. **遵守本规则的要求。**

如开发者和终端用户对本规则内容有任何疑问或建议，可随时通过本规则提供的方式与我们联系。

一、我们收集的信息及我们如何使用信息

（一）为实现 SDK 产品功能所需收集的个人信息

为实现 SDK 产品的相应功能所必须，我们将向终端用户或开发者收集终端用户在使用与 SDK 产品相关的功能时产生的如下个人信息：

个人信息类型	处理目的	处理方式
AndroidID	离线模式且采用按设备授权的方式时，SDK需要采集Android ID发送到设备后台，获取授权证书用于对设备的唯一性识别与校验	设备端本地加密传输处理
音频数据	为实现 SDK 的音频转文字基础功能，需要上传音频以便识别转为文字	1. 离线模式时：本地化处理，即本SDK接收音频数据后在设备端转换成文字，开发者将文字调回； 2. 在线模式：加密传输处理； 3. 本 SDK 优先在线模式，在线模式无法适用时可切换为离线模式。
iOS IDFV	离线模式且采用按设备授权的方式时，SDK需要采集IDFV发送到设备后台，获取授权证书用于对设备的唯一性识别与校验	设备端本地加密传输处理

（二）为实现SDK 产品功能所需的权限

为实现SDK 产品的相应功能所必须，我们会通过开发者的应用申请所需权限。

操作系统	权限名称	使用目的	是否可选
Android	网络访问 android.permission.INTERNET	1. 允许程序访问网络连接，离线模式下按设备授权时向后台发送授权信息获取授权证书； 2. 在线模式下允许程序访问网络连接，用于连接网络，进行数据传输。	必选
	网络状态 android.permission.ACCESS_NETWORK_STATE	1. 用来判断使用离线还是在线模式； 2. 在线模式时需要用于获取网络状态，用于优化当前网络通信。	必选
iOS	网络访问	1. 允许程序访问网络连接，离线模式下按设备授权时向后台发送授权信息获取授权证书；	必选

		2. 在线模式下允许程序访问网络连接，用于连接网络，进行数据传输。	
	网络状态	用来判断使用离线还是在线模式。	必选

请注意，在不同设备和系统中，权限显示方式及关闭方式会有所不同，需同时参考其使用的设备及操作系统开发方的说明或指引。当终端用户关闭权限即代表其取消了相应的授权，我们和开发者将不会继续收集和使用相关权限所对应的个人信息，也无法为终端用户提供需要终端用户开启权限才能提供的对应的功能。

（三）根据法律法规的规定，以下是征得用户同意的例外情形：

为订立、履行与终端用户的合同所必需。

为履行我们的法定义务所必需。

为应对突发公共卫生事件，或者紧急情况下为保护终端用户的生命健康和财产安全所必需。

为公共利益实施新闻报道、舆论监督等行为，在合理的范围内处理终端用户的个人信息。

依照本法规定在合理的范围内处理终端用户自行公开或者其他已经合法公开的个人信息。

法律行政法规规定的其他情形。

特别提示： 如我们收集的信息无法单独或结合其他信息识别到终端用户的个人身份，其不属于法律意义上的个人信息。

二、信息的公开披露

我们不会将终端用户的个人信息转移给任何公司、组织和个人，但以下情况除外：

1. 事先告知终端用户转移个人信息的种类、目的、方式和范围，并征得终端用户的单独同意。
2. 如涉及合并、分立、解散、被宣告破产等原因需要转移个人信息的，我们会向终端用户告知接收方的名称或者姓名和联系方式，并要求接收方继续履行个人信息处理者的义务。接收方变更原先的处理目的、处理方式的，我们会要求接收方重新取得终端用户的同意。

我们不会公开披露终端用户的个人信息，但以下情况除外：

1. 告知终端用户公开披露的个人信息的种类、目的、方式和范围并征得终端用户的单独同意后。
2. 在法律法规、法律程序、诉讼或政府主管部门强制要求的情况下。

三、终端用户如何管理自己的信息

我们非常重视终端用户对其个人信息管理的权利，并竭力帮助终端用户管理个人信息，包括个人信息查阅、复制、删除、注销账号以及设置隐私功能等，以保障终端用户的权利。如您是开发者，您应当为终端用户提供实现查阅、复制、修改、删除个人信息、撤回同意和注销账号的方式。

基于终端用户的同意而进行的个人信息处理活动，终端用户有权撤回该同意。我们已向开发者提供关闭本 SDK 产品的能力，请开发者点击此处查看操作指引开发者可以通过不发起对【Android 包含：ASRController；iOS包含：ASRController】接口的调用来停止收集和处理终端用户的个人信息。由于我们与终端用户无直接的交互对话界面，终端用户可以直接联系开发者停止使用本SDK产品，也可通过本规则第八条提供的方式与我们联系。如您是终端用户，请您理解，特定的业务功能或服务需要您提供服务所需的信息才能得以完成，当您撤回同意后，我们无法继续为您提供对应的功能或服务，也不再处理您相应的个人信息。您撤回同意的决定，不会影响我们此前基于您的授权而开展的个人信息处理。

四、信息的存储

(一) 存储信息的地点

我们遵守法律法规的规定，将在中华人民共和国境内收集和产生的个人信息存储在境内。

(二) 存储信息的期限

一般而言，我们仅在为实现目的所必需的最短时间内保留终端用户的个人信息，但下列情况除外：

- 为遵守适用的法律法规等有关规定；
- 为遵守法院判决、裁定或其他法律程序的规定；
- 为遵守相关政府机关执法的要求。

五、信息安全

我们为终端用户的个人信息提供相应的安全保障，以防止信息的丢失、不当使用、未经授权访问或披露。

我们严格遵守法律法规保护终端用户的个人信息。

我们将在合理的安全水平内使用各种安全保护措施以保障信息的安全。

例如，我们使用加密技术、匿名化处理等手段来保护终端用户的个人信息。

我们建立严谨的管理制度、流程和组织确保信息安全。

例如，我们严格限制访问信息的人员范围，要求他们遵守保密义务，并进行审查。

若发生个人信息泄露等安全事件，我们会启动应急预案，阻止安全事件扩大，并以推送通知、公告等形式告知开发者。

六、未成年人保护

本 SDK 产品主要面向成年人。
若您是开发者，如果终端用户是未满14周岁的未成年人（“儿童”），您应当向儿童的父母或其他监护人告知本规则，并在征得儿童的父母或其他监护人同意的前提下处理儿童个人信息。如果我们发现开发者未征得儿童监护人同意向我们提供儿童个人信息的，我们将会采取措施尽快删除。
若您是儿童监护人，当您对您所监护儿童个人信息保护有相关疑问或权利请求时，您可以联系开发者，或通过本规则提供的方式与我们联系。

七、变更

我们会适时修订本规则的内容。
如本规则的修订会导致终端用户在本规则项下权利的实质减损，我们将在变更生效前，通过网站公告等方式进行告知。如您是开发者，当更新后的本规则对处理终端用户的个人信息情况有变动的，您应当适时更新隐私政策，并以弹框形式通知终端用户并且征得其同意，如果终端用户不同意接受本规则，请停止集成 SDK 产品。

八、联系我们

我们设立了专门的个人信息保护团队和个人信息保护负责人，如果开发者和/或终端用户对本规则或个人信息保护相关事宜有任何疑问或投诉、建议时，可以通过以下方式与我们联系：

1. 通过 腾讯客服 或者 登录 > 腾讯云 与我们联系或 在线支持 > 腾讯云 。
2. 将问题发送至 Dataprivacy@tencent.com。
3. 邮寄信件至： 中国广东省深圳市南山区海天二路33号腾讯滨海大厦 数据隐私保护部（收）邮编： 518054。

我们将尽快审核所涉问题，并在15个工作日或法律法规规定的期限内予以反馈。