语音识别

SDK 文档

产品文档

腾讯云

文档目录

SDK 文档
实时语音识别 SDK
接口说明

最近更新时间：2020-12-18 18:00:35

接口描述

本接口服务对实时音频流进行识别，同步返回识别结果，达到“边说边输出文字”的效果。

- 支持中文普通话、英语、粤语、韩语、日语和上海话方言的识别
- 支持金融领域模型
- 支持 VAD（语音活动检测）功能
- 支持识别结果同步返回或尾包返回

接口说明请观看视频：

点击查看视频

接口要求

使用实时语音识别 SDK 时，需按照以下要求。

内容	说明
支持语言	中文普通话、英语、粤语、韩语、日语、上海话方言
支持行业	通用、金融
音频属性	采样率：16000Hz或8000Hz、采样精度：16bits、声道：单声道，查看音频属性请参见 常见问题
音频格式	wav、pcm、opus、speex、silk、mp3
数据长度	音频流中每个数据包的音频分片建议为200ms，8k采样率对应的音频分片大小为3200字节，16k采样率对应的音频分片大小为6400字节
请求地址	http://asr.cloud.tencent.com/asr/v1/<appid>?{请求参数}
免费额度	每月5小时
请求频率限制	50次/秒

使用步骤

1. 首先获取您的 AppID、SecretID 和 SecretKey。在使用该接口前，需要在 语音识别控制台 开通服务，并进入 API 密钥管理页面 新建密钥，生成 AppID、SecretID 和 SecretKey ，用于 SDK 调用时生成签名，签名将用来进行接口鉴权。
2. 查看您的音频属性和格式，应满足接口支持的属性和格式，否则会请求失败。
3. 通过 SDK 提交实时语音识别的请求。
4. 如果返回的 code = 0，表示请求成功，实时语音识别系统会将识别结果实时返回给客户端；如果返回的 code 不为0，请参见 错误码 详情。
5. 如果识别效果有问题，请参考 识别效果问题排查文档 进行排查。

实时语音识别请求

请求参数

请求参数主要由**请求 URL** 和**请求头部**组成。

请求 URL 示例

```
http://asr.cloud.tencent.com/asr/v1/<125000001>?
projectid=0&
```

```
secretid=<secretid>&
sub_service_type=1&
engine_model_type=16k_0&
result_text_format=0&
res_type=1&
voice_format=1&
needvad=1&
seq=0&
end=0&
source=0&
voice_id=text&
timestamp=1473752207&
expired=1473752300&
timeout=20&
nonce=0&
```

请求 URL 参数说明

参数名称	必选	类型	描述
AppId	是	Int	用户在腾讯云注册账号的 AppId，可以进入 API 密钥管理页面 获取。
projectid	否	Int	腾讯云项目 ID，语音识别目前不区分项目，填 0 即可。
secretid	是	String	用户在腾讯云注册账号 AppId 对应的 SecretId，可以进入 API 密钥管理页面 获取。
sub_service_type	否	Int	子服务类型。1：实时流式识别。
engine_model_type	是	String	引擎模型类型。 电话场景： · 8k_en：电话 8k 英语； · 8k_zh：电话 8k 中文普通话通用； · 8k_zh_finance：电话 8k 金融领域模型； 非电话场景： · 16k_zh：16k 中文普通话通用； · 16k_en：16k 英语； · 16k_ca：16k 粤语； · 16k_ko：16k 韩语； · 16k_ja：16k 日语； · 16k_wuu-SH：16k上海话方言； · 16k_zh-TW：16k 中文普通话繁体。
hotword_id	否	String	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。
result_text_format	否	Int	识别结果文本编码方式。0：UTF-8。
res_type	否	Int	结果返回方式。0：同步返回；1：尾包返回。
voice_format	否	Int	语音编码方式，可选，默认值为 4。1：wav(pcm)；4：speex(sp)；6：silk；8：mp3；10：opus。
needvad	否	Int	0：关闭 vad，1：开启 vad。 如果音频流总时长超过60秒，用户需开启 vad。
vad_silence_time	否	Int	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），取值范围：240-2000，单位 ms，目前仅支持 8k_zh、8k_zh_finance、16k_zh 引擎模型。
seq	是	Int	语音分片的序号，序号从 0 开始，每次请求递增1， 两个seq之间间隔不能超过6秒。
end	是	Int	是否为最后一片，最后一片语音片为 1，其余为 0。
source	否	Int	设置为 0 即可。
voice_id	是	String	16 位 String 串作为每个音频的唯一标识，用户自己生成。

参数名称	必选	类型	描述
timestamp	是	Int	当前 UNIX 时间戳，可记录发起 API 请求的时间。如果与当前时间相差过大，会引起签名过期错误。可以取值为当前请求的系统时间戳即可。
expired	是	Int	签名的有效期，是一个符合 UNIX Epoch 时间戳规范的数值，单位为秒；Expired 必须大于 Timestamp 且 Expired - Timestamp 小于90天。
nonce	是	Int	随机正整数。用户需自行生成，最长10位。
filter_dirty	否	Integer	是否过滤脏词（目前支持中文普通话引擎）。默认为0。0：不过滤脏词；1：过滤脏词；2：将脏词替换为 * 。
filter_modal	否	Integer	是否过语气词（目前支持中文普通话引擎）。默认为0。0：不过滤语气词；1：部分过滤；2：严格过滤 。
filter_punc	否	Integer	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。默认为0。
convert_num_mode	否	Int	是否进行阿拉伯数字智能转换。0：全部转为中文数字；1：根据场景智能转换为阿拉伯数字。
word_info	否	Int	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。支持引擎8k_en，8k_zh，8k_zh_finance，16k_zh，16k_en，16k_ca，16k_zh-TW，16k_ja，16k_wuu-SH，默认为0。

请求头部示例

```
{
"Content-Type":"application/octet-stream",
"Authorization":"UyKZ+Q4xMbdu3gxOmPD7tgnAm1A="
}
```

请求头部参数说明

参数名称	必选	类型	描述
Host	是	String	语音识别服务域名，固定为 asr.cloud.tencent.com
Authorization	是	String	用户的有效签名，用于鉴权
Content-Type	是	String	application/octet-stream
Content-Length	是	Int	请求长度，此处对应语音数据字节数，单位：字节

实时语音识别结果返回

同步返回示例

第一个分片：

```
{"code":0,"message":"success","voice_id":"8qiS3yeVnwHHbQ9F","seq":0,"text":"","result_number":1,"result_list":[{"slice_type":0,"index":1,"start_time":0,"end_time":256,"voice_text_str":""}],"final":0}
```

中间某个分片：

```
{"code":0,"message":"success","voice_id":"8qiS3yeVnwHHbQ9F","seq":2,"text":"吃饭了。","result_number":1,"result_list":[{"slice_type":0,"index":1,"start_time":512,"end_time":768,"voice_text_str":"吃饭了。"}],"final":0}
```

最后一个分片：

```
{"code":0,"message":"success","voice_id":"K3lRwC6tWkKjvnCL","seq":6,"text":"吃饭了吗。","result_number":1,"result_list":[{"slice_type":0,"index":1,"start_time":3072,"end_time":3093,"voice_text_str":"吃饭了吗。"}],"final":1}
```

返回参数说明

参数名称	描述
code	0：正常；不为0：发生错误
message	0：success；不为0：其他
voice_id	表示这通音频的标记，同一个音频流标记一样
seq	语音分片的信号 如果请求参数 needvad 为0的话，表示不需要后台做 vad，这里的 seq 就是发送过来的 seq 的序号。 如果请求参数 needvad 为1的话，表示需要后台做 vad，vad 会重新分片，送入识别的 seq 会和发送过来的 seq 不一样。
text	语音分片的识别结果 如果请求参数 needvad 为0的话，表示不需要后台做 vad， text 为完整识别结果。 如果请求参数 needvad 为1的话，表示需要后台做 vad，text 为空，需要从 result_list 字段中获取分片识别结果。
result_number	表示后面的 result_list 里面有几段结果，如果是0表示没有结果，遇到中间是静音。 如果是1表示 result_list 有一个结果， 在发给服务器分片很大的情况下可能会出现多个结果，正常情况下都是1个结果。
result_list	slice_type：返回分片类型标记， 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 index：表示第几段话 start_time：这个分片在整个音频流中的开始时间 end_time：这个分片在整个音频流中的结束时间 voice_text_str：识别结果
final	0：表示还在整个音频流的中间部分 1：表示是整个音频流的最后一个包。 应用主要是在电信场景中，客户端发送完了之后，要知道是否返回的是最后一个包。
word_size	表示后面的 word_list 的长度，即有多少个词。
word_list	词时间戳列表： word 表示这个词的内容， start_time 表示该词在整个音频中的起始时间， end_time 表示该词在整个音频中的结束时间， stable_flag 表示词的稳态结果，0：该词在后续识别中可能发生变化，1：表示该词在后续识别过程中不会变化。

错误码

数值	说明
100	获取语音分片信息失败
101	语音分片过大
102	参数不合法，具体详情参考 message
103	访问数据库失败
104	AppID 未注册
105	模板不存在
106	模板停用
107	鉴权失败
108	拼接签名串失败
109	l5获取 IP、port 失败
110	后台识别服务器故障，请从 seq = 0重传
111	后台识别模块回包格式错误

数值	说明
112	语音分片为空
113	后台服务器识别超时
114	引擎编号不合法
115	时长计算时音频类型不合法
116	无可使用的免费额度
117	禁止访问
118	请求限流
119	账户欠费停止服务，请及时充值
120	获取 rpcClient 错误
121	后台识别服务器错误，请从seq = 0重传
122	后台识别服务器收到的包格式错误
123	后台识别服务器音频解压失败，请从seq = 0重传
124	后台识别服务器识别失败，请从seq = 0重传
125	后台识别服务器识别失败，请重新尝试
126	后台识别服务器音频分片等待超时，请从seq = 0重传
127	后台识别服务器音频分片重复

</appid>

Android SDK

最近更新时间：2020-06-28 17:04:29

Android SDK 接入请观看视频：

点击查看视频

接入准备

SDK 获取

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

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（GPRS、3G 或 Wi-Fi 等），且系统为 **Android 4.0** 及其以上版本。

开发环境

- 引入 .so 文件
  libWXVoice.so： 腾讯云语音检测 so 库。
- 引入 aar 包
  aai-2.1.5.aar： 腾讯云语音识别 SDK。
- 该接口 SDK 支持本地构建或者远程构建两种方式：
  - 本地构建
    可以直接下载 Android SDK 及 Demo，然后集成对应的 so 文件和 aar 包（均在 sdk-source 目录下），最后将 okhttp3、okio、gson 和 slf4j 4个库也集成到 App 中。
    在 build.gradle 文件中添加：

```
implementation(name: 'aai-2.1.5', ext: 'aar')
```

  - 远程构建
    在 build.gradle 文件中添加：

```
implementation 'com.tencent.aai:aai:2.1.5:@aar'
```

- 添加相关依赖
  okhttp3、okio、gson 和 slf4j 依赖添加，在 build.gradle 文件中添加：

```
implementation 'com.squareup.okhttp3:okhttp:4.0.0-RC1'
implementation 'com.squareup.okio:okio:1.11.0'
implementation 'com.google.code.gson:gson:2.8.5'
implementation 'org.slf4j:slf4j-api:1.7.25'
```

  如果您使用 gradle 来进行工程构建，我们强烈建议使用远程构建的方式来构建您的应用。

- 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.WRITE_SETTINGS" />
< uses-permission android:name="android.permission.READ_PHONE_STATE"/>
< uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

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

快速接入

开发流程介绍

启动实时语音识别

```
int appid = XXX;
int projectid = XXX;
String secretId = "XXX";

// 为了方便用户测试，sdk提供了本地签名，但是为了secretKey的安全性，正式环境下请自行在第三方服务器上生成签名。
AbsCredentialProvider credentialProvider = new LocalCredentialProvider("your secretKey");

final AAIClient aaiClient;
try {
// 1、初始化AAIClient对象。
aaiClient = new AAIClient(this, appid, projectid, secretId, credentialProvider);

// 2、初始化语音识别请求。
final AudioRecognizeRequest audioRecognizeRequest = new AudioRecognizeRequest.Builder()
.pcmAudioDataSource(new AudioRecordDataSource()) // 设置语音源为麦克风输入
.build();

// 3、初始化语音识别结果监听器。
final AudioRecognizeResultListener audioRecognizeResultListener = new AudioRecognizeResultListener() {
@Override
public void onSliceSuccess(AudioRecognizeRequest audioRecognizeRequest, AudioRecognizeResult audioRecognizeResult, int i) {
// 返回语音分片的识别结果
}

@Override
public void onSegmentSuccess(AudioRecognizeRequest audioRecognizeRequest, AudioRecognizeResult audioRecognizeResult, int i)
{
// 返回语音流的识别结果
}

@Override
public void onSuccess(AudioRecognizeRequest audioRecognizeRequest, String s) {
// 返回所有的识别结果
}

@Override
public void onFailure(AudioRecognizeRequest audioRecognizeRequest, ClientException e, ServerException e1) {
// 识别失败
}
};

// 4、启动语音识别
new Thread(new Runnable() {
@Override
public void run() {
if (aaiClient!=null) {
```

```
aaiClient.startAudioRecognize(audioRecognizeRequest, audioRecognizeResultListener);
}
}
}).start();

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

停止实时语音识别

```
// 1、获得请求的 ID
final int requestId = audioRecognizeRequest.getRequestId();
// 2、调用stop方法
new Thread(new Runnable() {
@Override
public void run() {
if (aaiClient!=null){
//停止语音识别，等待当前任务结束
aaiClient.stopAudioRecognize(requestId);
}
}
}).start();
```

取消实时语音识别

```
// 1、获得请求的id
final int requestId = audioRecognizeRequest.getRequestId();
// 2、调用cancel方法
new Thread(new Runnable() {
@Override
public void run() {
if (aaiClient!=null){
//取消语音识别，丢弃当前任务
aaiClient.cancelAudioRecognize(requestId);
}
}
}).start();
```

主要接口类和方法说明

计算签名

调用者需要自己实现 AbsCredentialProvider 接口来计算签名，此方法为 SDK 内部调用，上层不用关心 source 来源。

- **计算签名函数如下：**

```
/**
* 签名函数：将原始字符串进行加密，具体的加密算法见以下说明。
* @param source 原文字符串
* @return 加密后返回的密文
*/
String getAudioRecognizeSign(String source);
```

- **计算签名算法**

先以 SecretKey 对 source 进行 HMAC-SHA1 加密，然后对密文进行Base64编码，获得最终的签名串。即：
sign=Base64Encode(HmacSha1(source，secretKey))。

为方便用户测试，SDK 已提供一个实现类 **LocalCredentialProvider**，但为保证 SecretKey 的安全性，请仅在测试环境下使用，正式版本建议上层实现接口 **AbsCredentialProvider** 中的方法。

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 secreteId, AbsCredentialProvider credentialProvider) thr
ows 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	用户的 projectId
secreteId	String	是	用户的 SecreteId
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();
```

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
setServerProtocolHttps	设置 HTTPS 或 HTTP 协议	true(HTTPS)	false 或 true
setMaxAudioRecognizeConcurrentNumber	语音识别最大并发请求数	2	1 - 5
setMaxRecognizeSliceConcurrentNumber	语音识别分片最大并发数	5	1 - 5
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.setServerProtocolHttps(true);
ClientConfiguration.setMaxAudioRecognizeConcurrentNumber(2)
```

```
ClientConfiguration.setMaxRecognizeSliceConcurrentNumber(5)
ClientConfiguration.setAudioRecognizeSliceTimeout(2000)
ClientConfiguration.setAudioRecognizeConnectTimeout(2000)
ClientConfiguration.setAudioRecognizeWriteTimeout(2000)
```

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 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, ClientException clientException, ServerException serverException);
```

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

示例代码详见入门示例。

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，开启后说话前的静音部分不进行识别	true
audioFlowSilenceTimeOut	Int	否	开启检测说话启始超时，开启后超时会自动停止录音	5000ms
minAudioFlowSilenceTime	Int	否	两个语音流最短分割时间	2000ms
minVolumeCallbackTime	Int	否	音量回调时间	80ms
sensitive	float	否	语音识别敏感度，越小越敏感(范围1 - 5)	3

示例：

```
AudioRecognizeConfiguration audioRecognizeConfiguration = new AudioRecognizeConfiguration.Builder()
.setSilentDetectTimeOut(true)// 是否使能静音检测，false 表示不检查静音部分
.audioFlowSilenceTimeOut(5000) // 静音检测超时停止录音
.minAudioFlowSilenceTime(2000) // 语音流识别时的间隔时间
.minVolumeCallbackTime(80) // 音量回调时间
.sensitive(2.8) // 识别敏感度
.build();

// 启动语音识别
new Thread(new Runnable() {
@Override
public void run() {
if (aaiClient!=null) {
aaiClient.startAudioRecognize(audioRecognizeRequest, audioRecognizeResultListener, audioRecognizeConfiguration);
}
}
}).start();
```

设置状态监听器

AudioRecognizeStateListener 可以用来监听语音识别的状态，一共有如下七个接口：

方法	方法描述
onStartRecord	开始录音
onStopRecord	结束录音
onVoiceFlowStart	检测到语音流的起点
onVoiceFlowStartRecognize	语音流开始识别
onVoiceFlowFinishRecognize	语音流结束识别
onVoiceVolume	音量

设置超时监听器

AudioRecognizeTimeoutListener 可以用来监听语音识别的超时，一共有如下两个接口：

方法	方法描述
onFirstVoiceFlowTimeout	检测第一个语音流超时

方法	方法描述
onNextVoiceFlowTimeout	检测下一个语音流超时

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 onVoiceFlowStart(AudioRecognizeRequest audioRecognizeRequest, int i) {
// 语音流开始
}

@Override
public void onVoiceFlowFinish(AudioRecognizeRequest audioRecognizeRequest, int i) {
// 语音流结束
}

@Override
public void onVoiceFlowStartRecognize(AudioRecognizeRequest audioRecognizeRequest, int i) {
// 语音流开始识别
}

@Override
public void onVoiceFlowFinishRecognize(AudioRecognizeRequest audioRecognizeRequest, int i) {
// 语音流结束识别
}

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

AudioRecognizeTimeoutListener audioRecognizeTimeoutListener = new AudioRecognizeTimeoutListener() {
@Override
public void onFirstVoiceFlowTimeout(AudioRecognizeRequest audioRecognizeRequest) {
// 检测语音起始超时
}

@Override
public void onNextVoiceFlowTimeout(AudioRecognizeRequest audioRecognizeRequest) {
// 检测语音结束超时
}
};
```

```
// 启动语音识别
new Thread(new Runnable() {
@Override
public void run() {
if (aaiClient!=null) {
aaiClient.startAudioRecognize(audioRecognizeRequest, audioRecognizeResultListener, audioRecognizeStateListener,audioRecogni
zeTimeoutListener, audioRecognizeConfiguration);
}
}
}).start();
```

其他重要类说明

AudioRecognizeRequest

templateName 和 customTemplate 都设置时，优先使用 templateName 的设置。

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
templateName	String	否	用户控制台设置的模板名称	无
customTemplate	AudioRecognizeTemplate	否	用户自定义的模板	(1, 0, 1)

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
code	Int	识别状态码
message	String	识别提示信息
text	String	识别结果
seq	Int	该语音分片的序号
voiceId	String	该语音分片所在语音流的 ID
cookie	String	cookie 值

AudioRecognizeTemplate

自定义的语音模板，需要设置的参数包括：

参数名称	类型	是否必填	参数描述
engineModelType	Int	是	引擎模型类型
resultTextFormat	Int	是	识别文本结果的编码形式，可选值包括：UTF-8，GB2312，GBK，BIG5
resType	Int	是	结果返回方式

示例：

```
AudioRecognizeTemplate audioRecognizeTemplate = new AudioRecognizeTemplate(1,0,1);
```

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();
```

-获取 sdk Pcm 格式录音源文件路径。

```
void savePcmFileCallBack(String filePath);
```

-获取 sdk wav 格式录音源文件路径。

```
void saveWaveFileCallBack(String filePath);
```

- 设置语音识别器每次最大读取数据量。

```
int maxLengthOnceRead();
```

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。

AudioFileDataSource

PcmAudioDataSource 接口的实现类，可以直接读取单通道、采样率16k的 PCM 音频数据的文件。

> ⚠ 注意：
>
> 其他格式的数据无法正确识别。

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();
```

iOS SDK

最近更新时间：2020-11-03 17:21:50

iOS SDK 接入请观看视频：

点击查看视频

接入准备

SDK 获取

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

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（GPRS、3G 或 Wi-Fi 网络等），且系统为 iOS 9.0 及以上版本。

开发环境

在工程 info.plist 添加以下设置：

- 设置 NSAppTransportSecurity 策略，添加如下内容：

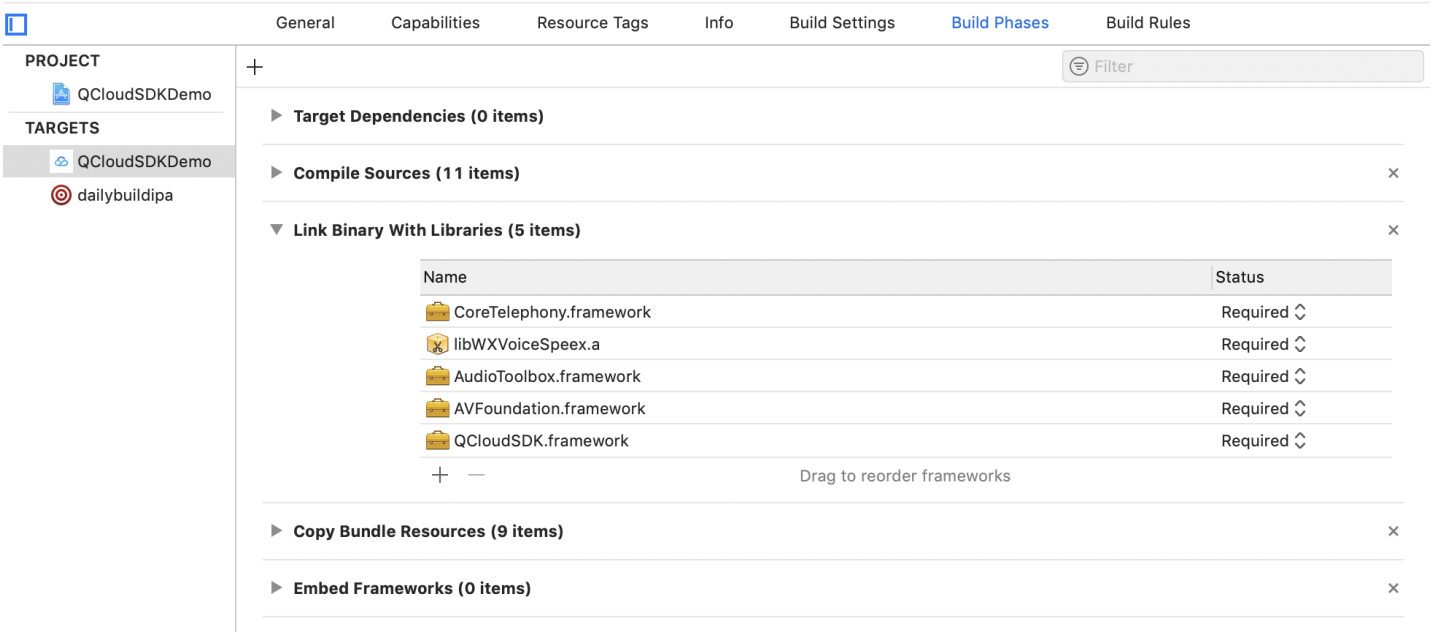
```
<key>NSAppTransportSecurity</key>
<dict>
<key>NSExceptionDomains</key>
<dict>
<key>qcloud.com</key>
<dict>
<key>NSExceptionAllowsInsecureHTTPLoads</key>
<true/>
<key>NSExceptionMinimumTLSVersion</key>
<string>TLSv1.2</string>
<key>NSIncludesSubdomains</key>
<true/>
<key>NSRequiresCertificateTransparency</key>
<false/>
</dict>
</dict>
</dict>
```

- 申请系统麦克风权限，添加如下内容：

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

- 在工程中添加依赖库，在 build Phases Link Binary With Libraries 中添加以下库：
  - AVFoundation.framework
  - AudioToolbox.framework
  - QCloudSDK.framework
  - CoreTelephony.framework
  - libWXVoiceSpeex.a

添加完后如下图所示：



快速接入

开发流程及接入示例

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。

使用内置录音器采集语音识别示例

1. **引入 QCloudSDK 的头文件，将使用 QCloudSDK 的文件名后缀由 .m->.mm**

```
#import<QCloudSDK/QCloudSDK.h>
```

2. **创建 QCloudConfig 实例**

```
//1.创建 QCloudConfig 实例
QCloudConfig *config = [[QCloudConfig alloc] initWithAppId:kQDAppId
secretId:kQDSecretId
secretKey:kQDSecretKey
projectId:kQDProjectId];
config.sliceTime = 600; //语音分片时常600ms
config.enableDetectVolume = YES; //是否检测音量
config.endRecognizeWhenDetectSilence = YES; //是否检测到静音停止识别
```

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**

```
[recognizer start];
```

6. 结束识别

```
[recognizer stop];
```

调用者提供语音数据示例

1. 引入 QCloudSDK 的头文件，将使用 QCloudSDK 的文件名后缀由 .m->.mm

```
#import<QCloudSDK/QCloudSDK.h>
```

2. 创建 QCloudConfig 实例

```
//1.创建 QCloudConfig 实例
QCloudConfig *config = [[QCloudConfig alloc] initWithAppId:kQDAppId
secretId:kQDSecretId
secretKey:kQDSecretKey
projectId:kQDProjectId];
config.sliceTime = 600; //语音分片时常600ms
config.enableDetectVolume = YES; //是否检测音量
config.endRecognizeWhenDetectSilence = YES; //是否检测到静音停止识别
```

3. 自定义 QCloudDemoAudioDataSource，QCloudDemoAudioDataSource 实现 QCloudAudioDataSource 协议

```
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];
```

主要接口类说明

QCloudRealTimeRecognizer 初始化说明

QCloudRealTimeRecognizer 是实时语音识别类，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;

/**
* 初始化方法，调用者传递语音数据调用此初始化方法
* @param config 配置参数，详见QCloudConfig定义
```

```
* @param dataSource 语音数据数据源，必须实现QCloudAudioDataSource协议
*/
- (instancetype)initWithConfig:(QCloudConfig *)config dataSource:(id<QCloudAudioDataSource>)dataSource;
```

QCloudConfig 初始化方法说明

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

QCloudRealTimeRecognizerDelegate 方法说明

```
/**
* 一次实时录音识别，分为多个flow，每个 flow 可形象的理解为一句话，一次识别中可以包括多句话。
* 每个 flow 包含多个 seq 语音数据包，每个 flow 的 seq 从0开始
*/
@protocol QCloudRealTimeRecognizerDelegate <NSObject>

@required
/**
* 每个语音包分片识别结果
* @param response 语音分片的识别结果
*/
- (void)realTimeRecognizerOnSliceRecognize:(QCloudRealTimeRecognizer *)recognizer response:(QCloudRealTimeResponse *)respon
se;

@optional
/**
* 一次识别成功回调
@param recognizer 实时语音识别实例
@param result 一次识别出的总文本
*/
- (void)realTimeRecognizerDidFinish:(QCloudRealTimeRecognizer *)recognizer result:(NSString *)result;
/**
* 一次识别失败回调
* @param recognizer 实时语音识别实例
* @param error 错误信息
*/
- (void)realTimeRecognizerDidError:(QCloudRealTimeRecognizer *)recognizer error:(NSError *)error;


///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/**
* 开始录音回调
* @param recognizer 实时语音识别实例
```

```
* @param error 开启录音失败，错误信息
*/
- (void)realTimeRecognizerDidStartRecord:(QCloudRealTimeRecognizer *)recognizer error:(NSError *)error;
/**
* 结束录音回调
* @param recognizer 实时语音识别实例
*/
- (void)realTimeRecognizerDidStopRecord:(QCloudRealTimeRecognizer *)recognizer;
/**
* 录音音量实时回调用
* @param recognizer 实时语音识别实例
* @param volume 声音音量，取值范围（-40-0）
*/
- (void)realTimeRecognizerDidUpdateVolume:(QCloudRealTimeRecognizer *)recognizer volume:(float)volume;


//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/**
* 语音流的开始识别
* @param recognizer 实时语音识别实例
* @param voiceId 语音流对应的 voiceId，唯一标识
* @param seq flow 的序列号
*/
- (void)realTimeRecognizerOnFlowRecognizeStart:(QCloudRealTimeRecognizer *)recognizer voiceId:(NSString *)voiceId seq:(NSInteger)seq;
/**
* 语音流的结束识别
* @param recognizer 实时语音识别实例
* @param voiceId 语音流对应的 voiceId，唯一标识
* @param seq flow的序列号
*/
- (void)realTimeRecognizerOnFlowRecognizeEnd:(QCloudRealTimeRecognizer *)recognizer voiceId:(NSString *)voiceId seq:(NSInteger)seq;

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/**
* 语音流开始识别
* @param recognizer 实时语音识别实例
* @param voiceId 语音流对应的 voiceId，唯一标识
* @param seq flow 的序列号
*/
- (void)realTimeRecognizerOnFlowStart:(QCloudRealTimeRecognizer *)recognizer voiceId:(NSString *)voiceId seq:(NSInteger)seq;
/**
* 语音流结束识别
* @param recognizer 实时语音识别实例
* @param voiceId 语音流对应的 voiceId，唯一标识
* @param seq flow 的序列号
*/
- (void)realTimeRecognizerOnFlowEnd:(QCloudRealTimeRecognizer *)recognizer voiceId:(NSString *)voiceId seq:(NSInteger)seq;

@end
```

QCloudAudioDataSource 协议说明

调用者不适用 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 工程里的 QDAudioDataSource 实现

```
/**
* 语音数据数据源，如果调用者需要自己提供语音数据，调用者实现此协议中所有方法
* 提供符合以下要求的语音数据：
* 采样率：16k
* 音频格式：pcm
* 编码：16bit位深的单声道
*/
@protocol QCloudAudioDataSource <NSObject>

@required

/**
* 标识 data source是否开始工作，执行完 start 后需要设置成 YES， 执行完 stop 后需要设置成 NO
*/
@property (nonatomic, assign) BOOL running;

/**
* SDK 会调用 start 方法，实现此协议的类需要初始化数据源。
*/
- (void)start:(void(^)(BOOL didStart, NSError *error))completion;
/**
* SDK 会调用 stop 方法，实现此协议的类需要停止提供数据
*/
- (void)stop;
/**
* 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，则直接返回 nil。
* @param expectLength 期望读取的字节数，如果返回的 NSData 不足 expectLength个字节，SDK 会抛出异常。
*/
- (nullable NSData *)readData:(NSInteger)expectLength;

@end
```

PHP SDK

最近更新时间：2020-11-03 17:20:59

接入准备

SDK 获取

实时语音识别 PHP SDK 以及 Demo 的下载地址：PHP SDK。

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**。

开发环境

- **环境依赖**
  该 SDK 适用于 PHP 5.4.16 及以上版本。
- **安装 SDK**
  根据下载地址下载源码，将源码中的 * .php 复制到项目中即可使用。

快速接入

1. 进入 API 密钥管理页面 获取 AppID、SecretId、SecretKey 并将`php_realtime_asr_sdk/Config.php`中的配置项按需改成自己的值。
2. 参考`php_realtime_asr_sdk/RasrClient.php` 接入：

```
<?php
require ('RASRsdk.php');
# 1. 先修改好 Config.php 文件中的配置
# 2. 然后开始调用：
//调用 RASRsdk 中的 sendvoice 函数获得识别结果
$filepath = "test_wav/8k/8k.wav";
$result = sendvoice($filepath, false);
echo "<br>8K Result is: ".$result;


# ----------------------------------------------------------------
# 3. 若需中途调整参数值，可直接修改，然后继续发请求即可。例如：
Config::$ENGINE_MODEL_TYPE = "16k_0";

$filepath = "test_wav/16k/16k.wav";
$result = sendvoice($filepath, false);
echo "<br>16K Result is: ".$result;

?>
```

C++ SDK

最近更新时间：2020-11-03 17:20:22

接入准备

SDK 获取

实时语音识别 C++ SDK 以及 Demo 的下载地址：C++ SDK。

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**。

开发环境

- **编译 Demo**，如失败需确认以下环境：

```
//下载sdk
tar -xzf c++_realtime_asr_sdk.tar.gz
cd c++_realtime_asr_sdk
make clean
make
//如果编译并未报错则跳过以下环境检测，否则可根据错误类型去校验库
```

- **安装 gcc g++**

```
1.RedHat 系列系统：
yum install -y gcc gcc-c++ make automake
//安装 gcc 等必备程序包（已安装则略过此步）
yum install -y wget
2.Debian系列系统：
apt-get install gcc g++
```

- **安装 CMake 工具**

```
// cmake 版本要大于3.5
wget https://cmake.org/files/v3.5/cmake-3.5.2.tar.gz
tar -zxvf cmake-3.5.2.tar.gz
cd cmake-3.5.2
sudo ./bootstrap --prefix=/usr
sudo make
sudo make install
```

- **依赖库安装及编译**
  客户需自行安装版本大于7.44.0的 **curl**。下载 curl 文件，解压并进入源码目录执行如下命令：

```
sudo ./configure
sudo cmake ./
sudo make
sudo make install
```

- **openssl**
  本 SDK 提供，目录为：c++*realtime*asr_sdk/lib。如果不适合客户系统，请客户自行安装方法，版本1.0.2f，下载 wget 源码并执行以下命令：

```
1. 更新 zlib
RedHat系列:yum install -y zlib
Debian系列:sudo apt-get install zlib1g zlib1g.dev
2. 安装
tar zxf openssl-1.0.2f.tar.gz
cd openssl-1.0.2f
sudo ./config shared zlib
sudo make
sudo make install
自行替换 c++_realtime_asr_sdk/lib 下面的库文件
```

- **speex**
  本 SDK 提供，目录为：c++_realtime_asr_sdk/lib，如果不适合客户系统，自行安装方法：下载 speex 源码，解压进入源码目录并执行以下命令：

```
sudo ./configure
sudo make
sudo make install
自行替换 c++_realtime_asr_sdk/lib 下面的库文件
```

快速接入

开发流程介绍

配置用户信息

- 进入 API 密钥管理页面 获取 AppID、SecretId、SecretKey 信息，并按如下步骤配置用户信息和请求 URL 参数。

```
#需要配置成用户账号信息 c++_record_asr_sdk/config/TCloudRecordASRConfig.ini
[tcloud_config]
#用户Appid
appid=1256*******
#用户SecretId
secretid=AKID****************************
#用户SecretKey，此处若担心存在密钥泄露风险，可在调用接口中传入，参考下面的“设置用户密钥”
secretkey=670m****************************
#需要配置成用户实际的参数，具体参见接口说明中的请求 URL 参数说明
[tcloud_asr]
projectid=0
sub_service_type=1
engine_model_type=16k_0
source=0
result_text_format=0
res_type=0
voice_format=4
```

初始化请求参数

- 调用 Init 接口初始化请求参数
- 请参考 开发流程介绍

设置用户密钥

- 调用 SetSecretKey 接口设置密钥
- 请参考 主要接口方法说明

Speex 压缩(可选)

- 调用 TSpeexEncoder::Encode 接口对 pcm 格式音频压缩
- 建议选择压缩，减少带宽

传入音频数据

- 调用 SetData 接口获取结果

SDK 已提供各个接口源码，用户可根据自身需要进行更改。

主要接口方法说明

TCloudRealtimeASR::Init

```
/*
** 初始化公共请求参数，此类参数较稳定不变
** config 公共请求参数结构体
** Output int 返回结果
*/
int Init(TCloudRealtimeASRConfig config);
```

TCloudRealtimeASR::SetSecretKey

```
/*
** 设置用户密钥
** strSecretKey 用户密钥
** Output int 返回结果
*/
int SetSecretKey(string strSecretKey);
```

TCloudRealtimeASR::BuildRequest

```
/*
** 构造一个完整的请求
** Output int 返回结果
*/
int BuildRequest();
```

TCloudRealtimeASR::SetData

```
/*
** 传入语音数据
** pPCMData pcm格式语音数据
** length 音频长度，字节
** voiceID 音频对应的voice_id
** seq 音频序号，含义参考参数
** bEnd 是否是最后一片
** Output TCloudRealtimeASRResponse 返回结果
*/
TCloudRealtimeASRResponse SetData(char* pPCMData, int length, string voiceID, int seq, bool bEnd);
```

TSpeexEncoder::Encode

```
/*
** pcm格式音频speex压缩
** inputBuffer 需压缩音频数据
** inputSize 需压缩音频长度 字节
** strRsp 压缩完成的音频串
** Output bool 返回结果
*/
bool Encode(const char* inputBuffer, int inputSize, string& strRsp);
```

请求 demo

```
//首先确认./bin/test.wav是否存在
make
./run.sh
```

入门示例

参考 c++_realtime_asr_sdk/src/demo.cpp，此例为 speex 压缩请求，建议使用 speex 压缩方式，可以减少带宽。

```
int RunRealtimeASR()
{
TCloudRealtimeASR realtimeASR;
int ret = realtimeASR.Init("./config/TCloudRealtimeASRConfig.ini");
if(ret != 0)
{
printf("Realtime ASR Init Error, Code:%d\n", ret);
return ret;
}

FILE* fp = fopen("bin/test.wav", "rb");
if (fp == NULL)
{
printf("file open failed\n");
}

char* buffer = new char[FILE_BUFFER_LEN];
int seq = 0;
string voiceID = GetVoiceID();
TSpeexEncoder encoder;
encoder.Init();
string strSpeex = "";
while(!feof(fp))
{
memset(buffer, 0, FILE_BUFFER_LEN);
int readLength = fread(buffer, 1, FILE_BUFFER_LEN, fp);
if(realtimeASR.GetVoiceFormat() == "4"){
if(seq == 0){
encoder.Encode(buffer + 44, readLength - 44, strSpeex);
}else{
encoder.Encode(buffer, readLength, strSpeex);
}
}
TCloudRealtimeASRResponse response = realtimeASR.SetData((char*)strSpeex.c_str(), strSpeex.size(), voiceID, seq, readLength
!= FILE_BUFFER_LEN);
```

```
seq++;
}

delete[] buffer;
buffer = NULL;
fclose(fp);
return 0;
}
```

Python SDK

最近更新时间：2020-11-03 17:19:39

接入准备

SDK 获取

实时语音识别 Python SDK 以及 Demo 的下载地址：Python3 SDK、Python2 SDK。

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。

开发环境

- **环境依赖**
  该接口支持 Python3 和 Python2.7 版本,请用户根据需要选择。
- **安装 requests**
  方法1：pip install requests 。
  方法2：先下载 requests，然后进入下载目录执行：python setup.py install 。

快速接入

1. 进入 API 密钥管理页面 获取 AppID、SecretId、SecretKey 并将Python_realtime_asr_sdk/src/Config.py中的配置项按需改成自己的值。
2. 参考 Python_realtime_asr_sdk/src/RasrClient.py 接入：

```
# 说明：请先将 Config.py 中的配置项按需改成自己的值，然后再开始使用。
import RASRsdk
import os
import Config
# ------------------------------ 调用方法1 ------------------------------
# 音频文件路径
filepath = "../../test_wavs/8k.wav"
# 调用语音识别函数获得识别结果，参数2标识是否打印中间结果到控制台
result = RASRsdk.sendVoice(filepath, True)
print("Final result: " + result)

# ----------------------------------------------------------------------
# 若需中途调整参数值，可直接修改，然后继续发请求即可。例如：
Config.config.ENGINE_MODEL_TYPE = '8k_0'

# ------------------------------ 调用方法2 ------------------------------
def requestExample():
# 将音频文件分解成小的切片数据（即：切分成长度较小的多个字符串）发出。模拟不断发送数据接收回复，最终收完整句话的识别结果。
filepath = "../../test_wavs/8k.wav"
file_object = open(filepath, 'rb')
file_object.seek(0, os.SEEK_END)
datalen = file_object.tell()
file_object.seek(0, os.SEEK_SET)

# 发送请求时需要用户自行维护3个变量：voiceId：创建后保持不变； seq：递增； endFlag：前面为0，发送尾部分片的请求时设置为1
voiceId = RASRsdk.randstr(16)
seq = 0
endFlag = 0
while (datalen > 0):
if (datalen < Config.config.CUT_LENGTH):
```

```
endFlag = 1
content = file_object.read(datalen)
else:
content = file_object.read(Config.config.CUT_LENGTH)
res = RASRsdk.sendRequest(content, voiceId, seq, endFlag)
print(res) # 打印收到的结果到控制台。示例用途。
seq = seq + 1
datalen = datalen - Config.config.CUT_LENGTH
file_object.close()

# 使用方法2
requestExample()
```

Java SDK

最近更新时间：2020-12-28 11:04:07

本文介绍如何使用腾讯云语音实时识别服务提供的 Java SDK，包括 SDK 的安装方法及 SDK 代码示例。

依赖环境

1. 依赖环境：JDK 1.8版本及以上
2. 从 腾讯云控制台 开通相应产品。
3. 获取 SecretID、SecretKey 。

获取安装

安装 Java SDK 前，先获取安全凭证。在第一次使用 SDK 之前，用户首先需要在腾讯云控制台上申请安全凭证，安全凭证包括 SecretID 和 SecretKey，SecretID 是用于标识 API 调用者的身份，SecretKey 是用于加密签名字符串和服务器端验证签名字符串的密钥，SecretKey 必须严格保管，避免泄露。

通过 Maven 安装

从 maven 服务器下载最新版本 SDK

```
<dependency>
<groupId>com.tencentcloudapi</groupId>
<artifactId>tencentcloud-speech-sdk-java</artifactId>
<version>1.0.11</version>
</dependency>
```

ASR SDK 说明

关键类说明

- SpeechClient 用于创建 SpeechRecognizer 语音识别器的客户端，通过 SpeechClient.newInstance 创建该实例，newInstance 为单例实现。
- SpeechRecognizer 语音识别器，通过客户端 speechClient.newSpeechRecognizer 创建实例。
- SpeechRecognitionRequest 用于配置请求参数，可通过 SpeechRecognitionRequest.initialize() 方法进行初始化。
- SpeechRecognitionResponse 请求响应。
- SpeechRecognitionListener 请求回调。包含识别开始，识别结束等回调方法。

SDK 使用说明

1.创建 SpeechClient 实例。
2.创建 SpeechRecognitionRequest，这里配置请求相关参数包含切片大、引擎模型类型、文件格式等，具体参考官网请求参数 。
3.创建 SpeechRecognizer 实例，该实例是语音识别的处理者。
4.调用 SpeechRecognizer 的 start 方法，开始识别。
5.调用 SpeechRecognizer 的 write 方法开始发送语音数据。
6.调用 SpeechRecognizer 的 stop 方法，结束识别。

示例

参考案例

```
package com.tencentcloud.asr;
import com.tencent.SpeechClient;
import com.tencent.asr.model.*;
import com.tencent.asr.service.SpeechRecognitionListener;
import com.tencent.asr.service.SpeechRecognizer;
import com.tencent.core.model.GlobalConfig;
import com.tencent.core.utils.ByteUtils;
import com.tencent.core.utils.JsonUtil;
```

```
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.List;
import java.util.Properties;
public class NewSpeechRecognitionHttpByteArrayExample {
public static void main(String[] args) throws InterruptedException, IOException {
GlobalConfig.ifLog = true;
//默认使用 websocket 协议，可通过该配置指定协议类型
//SpeechRecognitionSysConfig.requestWay= AsrConstant.RequestWay.Http;
Properties props = new Properties();
//从配置文件读取密钥
props.load(new FileInputStream("../config.properties"));
String appId = props.getProperty("appId");
String secretId = props.getProperty("secretId");
String secretKey = props.getProperty("secretKey");

//1.创建 client 实例 client 为单例
final SpeechClient speechClient = SpeechClient.newInstance(appId, secretId, secretKey);
//2.创建 SpeechRecognizerRequest，这里配置请求相关参数包含切片大小、文件格式等
final SpeechRecognitionRequest request = SpeechRecognitionRequest.initialize();
//必须手动设置 EngineModelType
request.setEngineModelType("16k_zh");
//根据文件格式设置 VoiceFormat
request.setVoiceFormat(1);
//3.创建 SpeechRecognizer 实例，该实例是语音识别的处理者。
SpeechRecognizer speechRecognizer = speechClient.newSpeechRecognizer(request, new MySpeechRecognitionListener());
//案例使用文件模拟实时获取语音流，用户使用可直接调用 recognize(data) 传入字节数据
FileInputStream fileInputStream = new FileInputStream(new File("8k.wav"));
List<byte[]> speechData = ByteUtils.subToSmallBytes(fileInputStream, 3200);
//4.调用 SpeechRecognizer 的 start 方法，开始识别。
speechRecognizer.start();
for (byte[] item : speechData) {
// 休眠用于模拟语音时长，方便测试，休眠时间根据传输数据选择对应值。实际使用不用休眠
// 参考时长：8k 3200字节 对应200ms 16k 6400字节对应200ms
Thread.sleep(200);
//5.调用 SpeechRecognizer 的 recognize 方法开始发送语音数据。
speechRecognizer.write(item);
}
//6.调用 SpeechRecognizer 的 end 方法，结束识别。
speechRecognizer.stop();
fileInputStream.close();
}

public static class MySpeechRecognitionListener extends SpeechRecognitionListener {
@Override
public void onRecognitionResultChange(SpeechRecognitionResponse response) {
//System.out.println("识别结果:"+JsonUtil.toJson(response));
}
@Override
public void onRecognitionStart(SpeechRecognitionResponse response) {
System.out.println("开始识别:" + JsonUtil.toJson(response));
}
@Override
public void onSentenceBegin(SpeechRecognitionResponse response) {
```

```
System.out.println("一句话开始:" + JsonUtil.toJson(response));
}
@Override
public void onSentenceEnd(SpeechRecognitionResponse response) {
System.out.println("一句话结束:" + JsonUtil.toJson(response));
}
@Override
public void onRecognitionComplete(SpeechRecognitionResponse response) {
System.out.println("识别结束:" + JsonUtil.toJson(response));
}
@Override
public void onFail(SpeechRecognitionResponse response) {
System.out.println("错误:" + JsonUtil.toJson(response));
}
}
}
```

Node.js SDK

最近更新时间：2020-08-21 10:03:15

接入准备

SDK 获取

实时语音识别 Node.js SDK 以及 Demo 的下载地址： Node.js SDK

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**。

开发环境

- **环境依赖**
  该接口支持 Node.js 7.10.1 及以上版本。
- **安装 SDK**
  下载文件后解压缩，添加文件到代码编辑器（WebStorm、VSCode 或其他编辑器），在 nodejs_realtime_asr_sdk_v1 文件夹下执行`npm install`命令安装依赖。

快速接入

demo 文件夹下：localRecordFile.js 为整个文件的识别请求 demo，oneFragmentation.js 为某个分片的识别请求 demo。

1. 通过下面的实时语音识别请求中的两种接入方式的 demo 或是运行 demo文件夹 下的 js 文件快速请求，进入 API 密钥管理页面 获取 AppID、SecretId、SecretKey，并在代码中对应位置配置好用户需要的参数。
2. 然后通过运行 demo 文件夹 下的 js 文件 或 在项目中使用以下的 demo，来快速获取识别结果。

实时语音识别请求

整个文件识别请求

```
const fs = require("fs");
const path = require('path');

//引入 sdk 和相关模块
const tencentcloud = require("../../nodejs_realtime_asr_sdk_v1");
const Asr = tencentcloud.asrRealtime;
const Config = tencentcloud.config;


//Config 实例的三个参数分别为 SecretId, SecretKey, appId。请前往控制台获取后修改下方参数
let config = new Config("Your SecretId","Your SecretKey",1200000000);



//设置接口需要参数，具体请参考 实时语音识别接口文档
let query = {
engineModelType : '16k_zh',
resultTextFormat : 0,
resType : 0,
voiceFormat : 1,
cutLength : 6400, // 整个文件识别时需要将文件分片，16k模型分片大小为6400，8k模型分片大小为3200
// 以下为非必填参数，可跟据业务自行修改
// hotwordId : '08003a00000000000000000000000000',
// wordInfo : 1,
// needvad: 0,
```

```
// filterDirty: 0,
// filterModal: 0,
// filterPunc: 0,
// convertNumMode : 0
}
//创建调用实例
const asrReq = new Asr(config, query);

//调用方式1:识别整个文件
let filePath = path.resolve('./test_long.wav');
let data = fs.readFileSync(filePath);

//发送识别请求，sendVoice 函数最后一个参数为文件分片请求返回时触发的回调，可根据业务修改
//error 为请求错误，response 为请求响应，data 为请求结果
asrReq.sendVoice(data, (error, response, data) => {
if(error){
console.log(error);
return;
}
console.log(data);
});
```

分片识别请求

```
const fs = require("fs");
const path = require('path');

//引入 sdk 和相关模块
const tencentcloud = require("../../nodejs_realtime_asr_sdk_v1");
const Asr = tencentcloud.asrRealtime;
const Config = tencentcloud.config;


//Config 实例的三个参数分别为 SecretId，SecretKey，appId。请前往控制台获取后修改下方参数
let config = new Config("Your SecretId","Your SecretKey",1200000000);



//设置接口需要参数，具体请参考 接口文档
let query = {
engineModelType : '16k_zh',
resultTextFormat : 0,
resType : 0,
voiceFormat : 1,
// 以下为非必填参数，可跟据业务自行修改
// hotwordId : '08003a00000000000000000000000000',
// wordInfo : 1,
// needvad: 0,
// filterDirty: 0,
// filterModal: 0,
// filterPunc: 0,
// convertNumMode : 0
}
//创建调用实例
const asrReq = new Asr(config, query);
```

```
//调用方式2:识别某个分片，test_short.wav 为示例分片
//发送请求时需要用户自行维护3个变量：voiceId：创建后保持不变； seq：递增； endFlag：前面为0，发送尾部分片的请求时设置为1

let filePathTestOne = path.resolve('./test_short.wav');
let dataTest = fs.readFileSync(filePathTestOne);
let vioceId = asrReq.randStr(16);
let seq = 0;
let endFlag = 1;

//发送识别请求，sendRequest 函数最后一个参数为请求返回时触发的回调，可根据业务修改
asrReq.sendRequest(dataTest, vioceId, seq, endFlag, (error, response, data) => {
if(error){
console.log(error);
return;
}
console.log(data);
});
```

录音文件识别 SDK
接口说明

最近更新时间：2020-12-23 17:55:06

接口描述

本接口服务对时长5小时以内的录音文件进行识别，异步返回识别全部结果， HTTP RESTful 形式。

- 支持中文普通话、英语、粤语、日语和上海话方言
- 支持通用、音视频领域
- 支持 wav、mp3、m4a、flv、mp4 格式
- 支持语音 URL 和本地语音文件两种请求方式
- 语音 URL 的音频时长不能长于5小时，文件大小不超过512MB
- 本地语音文件上传的文件不能大于5MB

> 说明：
> 提交录音文件识别请求后，在5小时内完成识别（半小时内发送超过1000小时录音或者2万条识别任务的除外），识别结果在服务端可保存7天。

接口说明请观看视频：

点击查看视频

接口要求

使用录音文件识别 SDK 时，需按照以下要求。

内容	说明
音频属性	采样率16k或8k、采样精度16bits、单声道或双声道，查看音频属性请参见 常见问题
音频格式	支持 wav、mp3、m4a
数据长度	语音 URL 的音频时长不能长于5小时，文件大小不超过512MB 本地语音文件不能大于5MB
免费额度	每月10小时

使用步骤

1. 首先获取您的 AppID、SecretID 和 SecretKey。在使用该接口前，需要在 语音识别控制台 开通服务，并进入 API 密钥管理页面 新建密钥，生成 AppID、SecretID 和 SecretKey ，用于 SDK 调用时生成签名，签名将用来进行接口鉴权。
2. 查看您的音频属性和格式，应满足接口支持的属性和格式，否则会请求失败。
3. 通过 SDK 提交录音文件识别的请求，请求参数可以查看 录音文件识别请求 接口文档，正常情况下服务器端会返回该请求的状态。
4. 如果返回的 code = 0，表示请求成功，用户可以选择轮询或者回调的方式获得识别结果：
   - **轮询方式**请参考 录音文件识别结果查询 接口文档；
   - **回调方式**需要用户在调用语音识别请求时给出回调 URL ，当语音识别系统完成识别后，会将结果通过 HTTP POST 请求的形式通知到用户，用户需要在自身业务服务器上搭建服务接收回调，具体见下面介绍；
5. 如果返回的 code 不为 0，请参见 录音文件识别请求 接口文档中的错误码详情。
6. 如果识别效果有问题，请参考 识别效果问题排查文档 进行排查。

录音文件识别请求

请求参数具体参见：录音文件识别请求 接口文档 。

录音文件识别结果回调

录音文件识别结果回调包括：**服务端回调结果**和**用户客户端确认**。

识别结果回调

服务端结果返回示例：

```
code=0&requestId=500&appid=12000001&projectid=0&text=%e6%82%a8%e5%a5%bd&audioUrl=http%3a%2f%2ftest.qq.com%2fvoice_url&audioTime=2.5&message=success
```

> 注意：
> 为了防止某些字段中，出现诸如 “&” 等特殊字符，导致解包失败，所有字段的 value 值都将进行 url_encode 之后发送给用户业务服务器，在获取 value 之后，需要先对 value 进行 url_decode 以获取原始 value 值。

服务端返回参数说明

字段	类型	描述
code	Int	服务器错误码，0 为成功，其他：失败
message	String	成功或者失败原因文字描述
requestId	Int	请求 ID，与后台任务 ID 一一对应
AppId	Int	腾讯云应用 ID
projectid	Int	腾讯云项目 ID
audioUrl	String	语音下载 URL。如果语音源非公网可下载 URL，则不包含该字段
text	String	识别出的结果文本
audioTime	Double	语音总时长

客户端确认

语音识别系统发起请求，收到请求后，用户侧需要以 JSON 格式回以响应。

客户端确认示例：

```
{ "code" : 0, "message" : "成功" }
```

客户端确认参数说明

参数名称	类型	描述
code	Int	错误码，0为成功，其他值代表失败
message	String	失败原因说明，例如业务服务器过载。 如果业务服务器返回失败，会间隔一段时间重新通知

回调错误码

数值	说明
10000	语音非标准格式，转码失败
10001	识别不成功
10002	语音时长过长
10003	语音时长过长
10004	无效的语音文件
10005	其他失败
10006	音轨个数不匹配

Android SDK

最近更新时间：2020-06-28 17:05:36

Android SDK 接入请观看视频：

点击查看视频

开发准备

SDK 下载

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

开发前

1. 开发者使用录音文件识别功能前，需要先在 腾讯云控制台 注册账号，
   并获得 APPID、SecretId 和 SecretKey 信息。
2. 手机必须要有网络（GPRS、3G 或 Wi-Fi 等）。
3. 支持 Android 4.0 及其以上版本。

运行环境配置

添加录音文件识别 SDK aar

将 **qcloudasrsdk_2.0_release.aar** 放在 libs 目录下，在 App 的 build.gradle 文件中添加。

```
implementation(name: 'qcloudasrsdk_2.0_release', ext: 'aar')
```

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'
implementation 'com.squareup.okhttp3:okhttp:4.0.0-RC1'
implementation 'com.squareup.okio:okio:1.11.0'
implementation 'org.slf4j:slf4j-api:1.7.25'
```

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.WRITE_SETTINGS" />
< uses-permission android:name="android.permission.READ_PHONE_STATE"/>
< uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
< uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
```

示例

创建 QCloudFileRecognizer 示例

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

设置识别结果回调

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

调用方式示例

- **通过语音 url 调用**

```
QCloudFileRecognitionParams params = (QCloudFileRecognitionParams) QCloudFileRecognitionParams.defaultRequestParams();
params.setUrl("http://client-sdk-1255628450.cossh.myqcloud.com/test%20audio/voice_WGVNG_800.mp3");
params.setSourceType(QCloudSourceType.QCloudSourceTypeUrl);
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.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.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。
fileRecognizer.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);

QCloudFileRecognitionParams params = (QCloudFileRecognitionParams) QCloudFileRecognitionParams.defaultRequestParams();
params.setData(audioData);
params.setSourceType(QCloudSourceType.QCloudSourceTypeData);
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.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.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。
fileRecognizer.recognize(params);
```

关键类说明

QCloudFileRecognizer 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 QCloudFileRecognizer(AppCompatActivity activity, 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(QCloudFileRecognitionParams params) throws Exception;
```

QCloudFileRecognizerListener 识别结果回调

```
public interface QCloudFileRecognizerListener {
/**
```

```
 * 识别结果回调
 * @param recognizer 录音文件识别实例
 * @param requestId 请求唯一标识别
 * @param result 识别文本
 * @param status 任务状态码：0：任务等待 1：任务执行中 2:任务成功 3：任务失败
 * @param exception 异常信息
 *
 */
void recognizeResult(QCloudFileRecognizer recognizer, final long requestId, String result, int status,Exception exception);
}
```

iOS SDK

最近更新时间：2020-12-22 11:14:41

iOS SDK 接入请观看视频：

点击查看视频

开发准备

SDK 获取

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

使用须知

- QCloudSDK 支持 **iOS 9.0** 及以上版本。
- 录音文件识别，需要手机能够连接网络（GPRS、3G 或 Wi-Fi 网络等）。
- 从控制台获取 AppID、SecretID、SecretKey、ProjectId 。
- 运行 Demo 必须设置 AppID、SecretID、SecretKey、ProjectId 。
- 进入 API 密钥管理页面 ，获取 AppID、SecretID 与 SecretKey 。

SDK 导入

iOS SDK 压缩包名称为：QCloudSDK_v2.0.7.zip，压缩包中包含 Sample Code 和 QCloudSDK 。

工程配置

在工程 `info.plist` 添加以下设置：

1. **设置 NSAppTransportSecurity 策略，添加如下内容：**

```
<key>NSAppTransportSecurity</key>
<dict>
<key>NSExceptionDomains</key>
<dict>
<key>qcloud.com</key>
<dict>
<key>NSExceptionAllowsInsecureHTTPLoads</key>
<true/>
<key>NSExceptionMinimumTLSVersion</key>
<string>TLSv1.2</string>
<key>NSIncludesSubdomains</key>
<true/>
<key>NSRequiresCertificateTransparency</key>
<false/>
</dict>
</dict>
</dict>
```

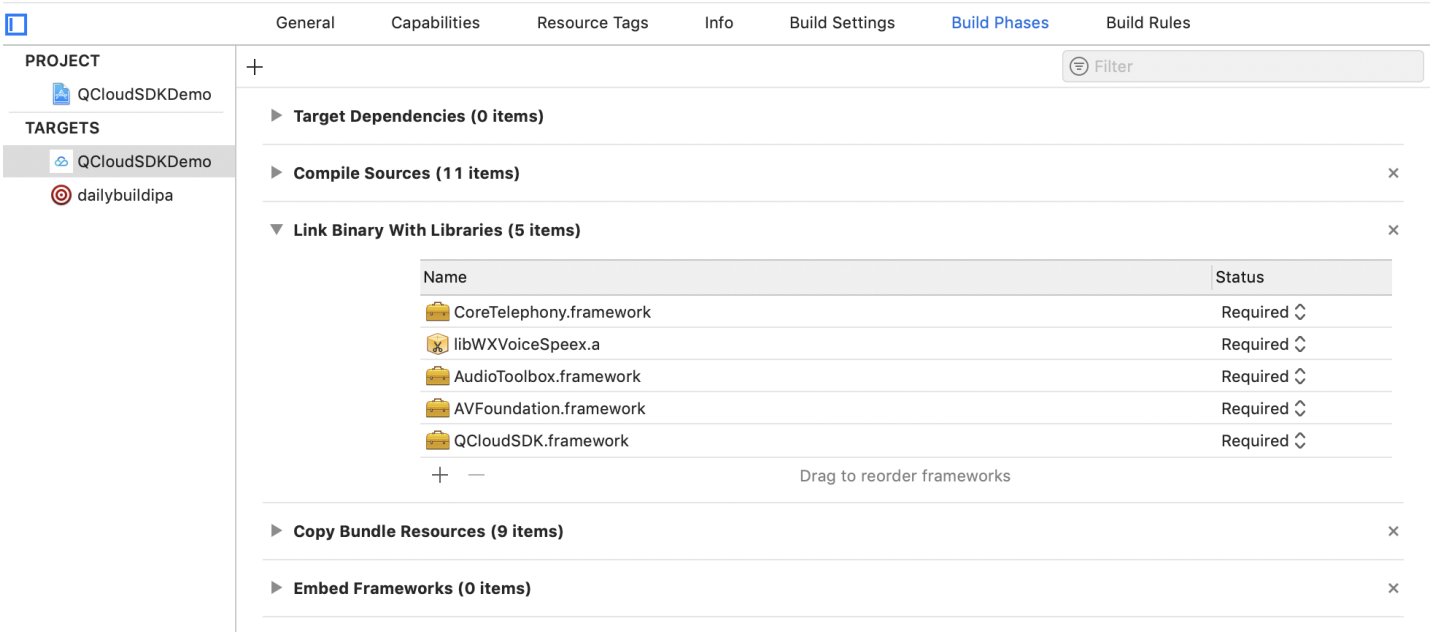
2. **申请系统麦克风权限，添加如下内容：**

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

3. **在工程中添加依赖库，在 build Phases Link Binary With Libraries 中添加以下库：**
   - AVFoundation.framework
   - AudioToolbox.framework
   - QCloudSDK.framework

- CoreTelephony.framework
- libWXVoiceSpeex.a

添加完如图所示。



类说明

QCloudFileRecognizer 初始化说明

QCloudFileRecognizer 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;
/**
* 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;
```

QCloudConfig 初始化方法说明

腾讯云 AppId，腾讯云 secretId，腾讯云 secretKey，腾讯云 projectId 从控制台获取。

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

QCloudFileRecognizerDelegate 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、开始录音、结束录音事件。

```
@protocol QCloudFileRecognizerDelegate <NSObject>
@optional
/**
录音文件识别成功回调
@param recognizer 录音文件识别器
@param requestId 请求唯一标识 requestId，recognize：接口返回
@param text 识别文本
@param resultData 原始数据
*/
- (void)fileRecognizer:(QCloudFileRecognizer *_Nullable)recognizer requestId:(NSInteger)requestId 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 requestId 请求唯一标识 requestId，recognize：接口返回
@param error 识别错误，出现错误此字段有
@param resultData 原始数据
*/
- (void)fileRecognizer:(QCloudFileRecognizer *_Nullable)recognizer requestId:(NSInteger)requestId error:(nullable NSError
*)error resultData:(nullable NSDictionary *)resultData;
@end
```

示例

1. 创建 QCloudFileRecognizer 实例

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

2. 实现此 QCloudFileRecognizerDelegate 协议方法

3. 调用方式示例

- 通过语音 url 调用

```
(void)recognizeWithUrl {
QCloudFileRecognizeParams *params = [QCloudFileRecognizeParams defaultRequestParams];
params.audioUrl = @"https://asr-audio-1256237915.cos.ap-shanghai.myqcloud.com/30s.wav";
[_recognizer recognize:params];
}
```

- 通过语音数据调用

```
(void)recognizeWithAudioData {
QCloudFileRecognizeParams *params = [QCloudFileRecognizeParams defaultRequestParams];
```

```
NSString *filePath = [[NSBundle mainBundle] pathForResource:@"recordedFile" ofType:@"wav"];
NSData *audioData = [[NSData alloc] initWithContentsOfFile:filePath];
params.audioData = audioData;
params.sourceType = QCloudAudioSourceTypeAudioData;
[_recognizer recognize:params];
}
```

C++ SDK

最近更新时间：2020-08-04 15:26:20

接入准备

SDK 获取

录音文件识别 C++ SDK 以及 Demo 的下载地址：C++ SDK

接入须知

开发者在调用前请先查看录音文件识别的 接口说明，了解接口的使用要求和使用步骤。

开发环境

需要64位 Linux 系统，如果用户使用的是32位系统，则需自行安装编译 extern 中的依赖库。

- 编译 demo，如果失败则参考对应的提示安装依赖工具。

```
//下载 sdk 并解压
unzip c++_record_asr_sdk.zip
cd c++_record_asr_sdk
cmake ./
make clean
make
//如果编译并未报错则跳过以下环境检测，否则可根据错误类型去校验库
```

- 安装 CMake 工具

```
// cmake 版本要大于3.5
wget https://cmake.org/files/v3.5/cmake-3.5.2.tar.gz
tar -zxvf cmake-3.5.2.tar.gz
cd cmake-3.5.2
sudo ./bootstrap --prefix=/usr
sudo make
sudo make install
```

- 安装 gcc g++

```
1.RedHat 系列系统:
yum install -y gcc gcc-c++ make automake
//安装 gcc 等必备程序包（已安装则略过此步）
yum install -y wget

2.Debian 系列系统：
apt-get install gcc g++
```

编译安装 extern 依赖库（编译 demo 正常则跳过）

- 安装 curl 下载地址

```
解压，并进入目录
sudo ./configure
sudo cmake ./
```

```
sudo make
sudo make install
```

- 安装 openssl 下载地址

```
1.更新 zlib
RedHat 系列:yum install -y zlib
Debian 系列:sudo apt-get install zlib1g zlib1g.dev

2.安装
tar zxf openssl-1.0.2f.tar.gz
cd openssl-1.0.2f
sudo ./config shared zlib
sudo make
sudo make install
自行替换 c++_record_asr_sdk/lib 下面的库文件
```

快速接入

开发流程介绍

- 配置用户信息，进入 API 密钥管理页面 获取 AppID、SecretId、SecretKey 信息，并配置用户信息

```
#用户配置信息文件：tcloud_auth.ini
AppId=Your AppId
SecretId=Your SecretId
SecretKey=Your SecretKey
```

- 配置请求参数

```
#用户请求参数配置文件：request_parameter.ini
#引擎类型。
EngineModelType = 16k_zh
#语音声道数。1：单声道；2：双声道（仅在电话 8k 通用模型下支持）。
ChannelNum = 1
#识别结果文本编码方式。0：UTF-8。
ResTextFormat = 0
#语音数据来源。0：语音 URL；1：语音数据（post body）。
SourceType = 1
#回调 URL，用户自行搭建的用于接收识别结果的服务器地址，长度小于2048字节。
CallbackUrl = http://test.qq.com
```

- 创建和初始化请求

```
//创建一个具体的请求
TCloudRecordASR asrReq;
//初始化请求参数
asrReq.InitCommonParam("./conf/request_parameter.ini");
//初始化用户信息
asrReq.InitAuth("./conf/tcloud_auth.ini");
```

- 传入音频发送请求，有本地音频上传和音频 URL 上传两种方式

```
//本地音频方式
//将sourcetype设置为1
asrReq.SetSourceType(1);
//传入本地音频文件路径
asrReq.CreateRecTask("./test.wav");
```

```
//音频URL方式
//将sourcetype设置为0
asrReq.SetSourceType(0);
//传入本地音频文件路径
asrReq.CreateRecTask("http://***.wav");
```

- 获取请求结果，支持轮询或者回调

```
//获取请求结果
string taskRsp = asrReq.GetResponse();
//成功请求则返回 taskId,可根据 taskId 查询识别结果
```

- 查询识别结果（轮询方式）

```
//根据 taskId 轮询查询识别结果
string strRsp;
asrReq.DescribeTaskStatus(taskId, strRsp);
```

- 获取识别结果（回调方式）

```
此方式需要客户自行搭建接收识别结果的服务，并将 服务 URL 传入参数 CallbackUrl
```

主要接口说明

SDK 已提供各个接口源码，用户可根据自身需要进行更改。

- TCloudRecordASR::InitCommonParam

```
/* 初始化公共请求参数，此类参数较稳定不变
** configPath 参数文件路径
** time 鉴权的有效时间
** Output int 返回结果
*/
int InitCommonParam(string configPath);
```

- TCloudRecordASR::InitAuth

```
/* 初始化用户信息
** stAuth 用户信息结构体
** time 鉴权的有效时间
```

```
** Output int 返回结果
*/
int InitAuth(TCloudRecordASRAuth stAuth);

/* 初始化用户信息
** configPath 参数文件路径
** time 鉴权的有效时间
** Output int 返回结果
*/
int InitAuth(string configPath);
```

- TCloudRecordASR::SetSourceType

```
/* 设置 SourceType
** type 0 音频url,  1本地音频
** Output void
*/
void SetSourceType(int type);
```

- TCloudRecordASR::CreateRecTask

```
/* 创建识别请求
** fileURI 音频的本地地址或者音频的 URL 链接
** Output int 返回结果
*/
int CreateRecTask(string fileURI);
```

- TCloudRecordASR::GetResponse

```
/* 获取任务结果
** Output string 任务返回 json 包
*/
string GetResponse();
```

- TCloudRecordASR::DescribeTaskStatus

```
/* 轮询查询识别结果
** taskId 任务 Id
** rsp 查询结果
** Output int 返回结果
*/
int DescribeTaskStatus(int taskId, string &rsp);
```

接入示例

```
//进入 demo 文件夹
//编译
./compile
//创建任务 参数：sourceType fileURI
./run_create_task 1 test.wav
```

```
//查询识别结果 参数：taskId
./run_describe_task 12345678
```

参考代码

识别请求 demo

```
#include "TCloudRecordASR.h"
#include <iostream>

using namespace std;
int main(int argc, char ** argv){
if(argc < 3){
cout << "use ./run_create_task source_type uri" << endl;
}
int sourceType = atoi(argv[1]);
string uri = string(argv[2]);
TCloudRecordASR asrReq;
//初始化请求参数
asrReq.InitCommonParam("./conf/request_parameter.ini");
//初始化用户信息
asrReq.InitAuth("./conf/tcloud_auth.ini");
asrReq.SetSourceType(sourceType);
asrReq.CreateRecTask(uri);
string taskRsp = asrReq.GetResponse();
cout << "rsp: " << taskRsp << endl;
return 0;
}
```

轮询查询结果 demo

```
#include "TCloudRecordASR.h"
#include <iostream>

using namespace std;
int main(int argc, char ** argv){
if(argc < 2){
cout << "use ./run_describe_task taskId" << endl;
}
int taskId = atoi(argv[1]);
TCloudRecordASR asrReq;
//初始化请求参数
asrReq.InitCommonParam("./conf/request_parameter.ini");
//初始化用户信息
asrReq.InitAuth("./conf/tcloud_auth.ini");
string strRsp;
asrReq.DescribeTaskStatus(taskId, strRsp);
cout << "rsp: " << strRsp << endl;
return 0;
}
```

Java SDK

最近更新时间：2020-08-04 15:26:26

接入准备

SDK获取

录音文件识别 Java SDK 获取，请参考：Java SDK 依赖环境及获取安装。

接入须知

开发者在调用前请先查看录音文件语音识别的 接口说明，了解接口的**使用要求**和**使用步骤**。

快速接入

以下分别是通过**语音 URL** 和**本地语音上传**请求方式的 demo 以及**轮询**识别结果的 demo ，来帮助客户快速接入。

- **通过语音 URL 方式请求**

```
import com.tencentcloudapi.common.Credential;
import com.tencentcloudapi.common.profile.ClientProfile;
import com.tencentcloudapi.common.profile.HttpProfile;
import com.tencentcloudapi.common.exception.TencentCloudSDKException;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.Base64;

import com.tencentcloudapi.asr.v20190614.AsrClient;

import com.tencentcloudapi.asr.v20190614.models.CreateRecTaskRequest;
import com.tencentcloudapi.asr.v20190614.models.CreateRecTaskResponse;

public class CreateRecTask
{
public static void main(String [] args) throws IOException {
//采用音频 URL 方式
try{
//重要，此处<Your SecretId><Your SecretKey>需要替换成客户自己的账号信息，获取方法：
//请参考接口说明（https://cloud.tencent.com/document/product/1093/37139）中的使用步骤 1 进行获取。
Credential cred = new Credential("Your SecretId", "Your SecretKey");

HttpProfile httpProfile = new HttpProfile();
httpProfile.setEndpoint("asr.tencentcloudapi.com");

ClientProfile clientProfile = new ClientProfile();
clientProfile.setHttpProfile(httpProfile);

AsrClient client = new AsrClient(cred, "ap-shanghai", clientProfile);

String params = "{\"EngineModelType\":\"16k_zh\",\"ChannelNum\":1,\"ResTextFormat\":0,\"SourceType\":0,\"Url\":\"https://asr-audio-1300466766.cos.ap-nanjing.myqcloud.com/test16k.wav\"}";
CreateRecTaskRequest req = CreateRecTaskRequest.fromJsonString(params, CreateRecTaskRequest.class);
```

```
CreateRecTaskResponse resp = client.CreateRecTask(req);

System.out.println(CreateRecTaskRequest.toJsonString(resp));
} catch (TencentCloudSDKException e) {
System.out.println(e.toString());
}
}
}
```

- **通过本地语音上传方式请求**

```
import com.tencentcloudapi.common.Credential;
import com.tencentcloudapi.common.profile.ClientProfile;
import com.tencentcloudapi.common.profile.HttpProfile;
import com.tencentcloudapi.common.exception.TencentCloudSDKException;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.Base64;

import com.tencentcloudapi.asr.v20190614.AsrClient;

import com.tencentcloudapi.asr.v20190614.models.CreateRecTaskRequest;
import com.tencentcloudapi.asr.v20190614.models.CreateRecTaskResponse;

public class CreateRecTask
{
public static void main(String [] args) throws IOException {
//通过本地音频方式
try{
//重要，此处<Your SecretId><Your SecretKey>需要替换成客户自己的账号信息，获取方法：
//请参考接口说明（https://cloud.tencent.com/document/product/1093/37139）中的使用步骤 1 进行获取。
Credential cred = new Credential("Your SecretId", "Your SecretKey");

HttpProfile httpProfile = new HttpProfile();
httpProfile.setEndpoint("asr.tencentcloudapi.com");

ClientProfile clientProfile = new ClientProfile();
clientProfile.setHttpProfile(httpProfile);

AsrClient client = new AsrClient(cred, "ap-shanghai", clientProfile);

String params = "{\"EngineModelType\":\"16k_zh\",\"ChannelNum\":1,\"ResTextFormat\":0,\"SourceType\":1}";
CreateRecTaskRequest req = CreateRecTaskRequest.fromJsonString(params, CreateRecTaskRequest.class);

File file = new File("/Users/ruskinli/eclipse-workspace/TencentSentence/src/test.wav");
FileInputStream inputFile = new FileInputStream(file);
byte[] buffer = new byte[(int)file.length()];
req.setDataLen(file.length());
inputFile.read(buffer);
```

```
inputFile.close();
String encodeData = Base64.getEncoder().encodeToString(buffer);
req.setData(encodeData);
CreateRecTaskResponse resp = client.CreateRecTask(req);
System.out.println(CreateRecTaskRequest.toJsonString(resp));
} catch (TencentCloudSDKException e) {
System.out.println(e.toString());
}

}

}
```

- **查询录音文件语音识别结果**

```
import com.tencentcloudapi.common.Credential;
import com.tencentcloudapi.common.profile.ClientProfile;
import com.tencentcloudapi.common.profile.HttpProfile;
import com.tencentcloudapi.common.exception.TencentCloudSDKException;

import com.tencentcloudapi.asr.v20190614.AsrClient;

import com.tencentcloudapi.asr.v20190614.models.DescribeTaskStatusRequest;
import com.tencentcloudapi.asr.v20190614.models.DescribeTaskStatusResponse;

public class DescribeTaskStatus
{
public static void main(String [] args) {
try{

//重要，此处<Your SecretId><Your SecretKey>需要替换成客户自己的账号信息，获取方法：
//请参考接口说明（https://cloud.tencent.com/document/product/1093/37139）中的使用步骤 1 进行获取。
Credential cred = new Credential("Your SecretId", "Your SecretKey");

HttpProfile httpProfile = new HttpProfile();
httpProfile.setEndpoint("asr.tencentcloudapi.com");

ClientProfile clientProfile = new ClientProfile();
clientProfile.setHttpProfile(httpProfile);

AsrClient client = new AsrClient(cred, "ap-shanghai", clientProfile);

String params = "{\"TaskId\":123456}";
DescribeTaskStatusRequest req = DescribeTaskStatusRequest.fromJsonString(params, DescribeTaskStatusRequest.class);

DescribeTaskStatusResponse resp = client.DescribeTaskStatus(req);

System.out.println(DescribeTaskStatusRequest.toJsonString(resp));
} catch (TencentCloudSDKException e) {
System.out.println(e.toString());
}

}
```

```
}
```

PHP SDK

最近更新时间：2020-08-04 15:26:36

接入准备

SDK获取

录音文件语音识别 PHP SDK 获取，请参考： PHP SDK 安装及相关环境说明。

接入须知

开发者在调用前请先查看录音文件语音识别的 接口说明，了解接口的**使用要求**和**使用步骤**。

快速接入

以下分别是通过**语音 URL** 和**本地语音上传**请求方式的 demo，来帮助客户快速接入。

1. 通过下面的录音文件识别请求中的两种接入方式的 demo快速请求，进入 API 密钥管理页面 获取 AppID、SecretId、SecretKey，并在代码中对应的位置配置好用户参数。
2. 然后在项目中使用以下的 demo，来快速获取识别结果。

- **通过语音 URL 方式请求**

```
<?php
require_once './tencentcloud-sdk-php/TCloudAutoLoader.php';
use TencentCloud\Common\Credential;
use TencentCloud\Common\Profile\ClientProfile;
use TencentCloud\Common\Profile\HttpProfile;
use TencentCloud\Common\Exception\TencentCloudSDKException;
use TencentCloud\Asr\V20190614\AsrClient;
use TencentCloud\Asr\V20190614\Models\CreateRecTaskRequest;
//通过音频 URL 方式请求
try {
//此处<Your SecretId><Your SecretKey>需要替换成客户自己的账号信息
$cred = new Credential("Your SecretId", "Your SecretKey");
$httpProfile = new HttpProfile();
$httpProfile->setEndpoint("asr.tencentcloudapi.com");

$clientProfile = new ClientProfile();
$clientProfile->setHttpProfile($httpProfile);
$clientProfile->setSignMethod("TC3-HMAC-SHA256");
$client = new AsrClient($cred, "ap-shanghai", $clientProfile);

$req = new CreateRecTaskRequest();

$params = '{"EngineModelType":"16k_zh","ChannelNum":1,"ResTextFormat":0,"SourceType":0,"Url":"https://asr-audio-1300466766.
cos.ap-nanjing.myqcloud.com/test16k.wav"}';
$req->fromJsonString($params);

$resp = $client->CreateRecTask($req);

print_r($resp->toJsonString());
}
catch(TencentCloudSDKException $e) {
echo $e;
}
```

- **通过本地语音上传方式请求**

```php
<?php
require_once './tencentcloud-sdk-php/TCloudAutoLoader.php';
use TencentCloud\Common\Credential;
use TencentCloud\Common\Profile\ClientProfile;
use TencentCloud\Common\Profile\HttpProfile;
use TencentCloud\Common\Exception\TencentCloudSDKException;
use TencentCloud\Asr\V20190614\AsrClient;
use TencentCloud\Asr\V20190614\Models\CreateRecTaskRequest;
//通过本地音频方式请求
try {
//此处<Your SecretId><Your SecretKey>需要替换成客户自己的账号信息
$cred = new Credential("Your SecretId", "Your SecretKey");
$httpProfile = new HttpProfile();
$httpProfile->setEndpoint("asr.tencentcloudapi.com");

$clientProfile = new ClientProfile();
$clientProfile->setHttpProfile($httpProfile);
$clientProfile->setSignMethod("TC3-HMAC-SHA256");
$client = new AsrClient($cred, "ap-shanghai", $clientProfile);

$req = new CreateRecTaskRequest();

$params = '{"EngineModelType":"16k_zh","ChannelNum":1,"ResTextFormat":0,"SourceType":1}';
$req->fromJsonString($params);
$data = file_get_contents('./test.wav');
$encodeData = base64_encode($data);
$req->Data = $encodeData;
$req->DataLen = strlen($data);

$resp = $client->CreateRecTask($req);

print_r($resp->toJsonString());
}
catch(TencentCloudSDKException $e) {
echo $e;
}
```

- **查询识别结果**

```php
<?php
require_once '../../../TCloudAutoLoader.php';
use TencentCloud\Common\Credential;
use TencentCloud\Common\Profile\ClientProfile;
use TencentCloud\Common\Profile\HttpProfile;
use TencentCloud\Common\Exception\TencentCloudSDKException;
use TencentCloud\Asr\V20190614\AsrClient;
use TencentCloud\Asr\V20190614\Models\DescribeTaskStatusRequest;
try {
//此处<Your SecretId><Your SecretKey>需要替换成客户自己的账号信息
$cred = new Credential("Your SecretId", "Your SecretKey");
$httpProfile = new HttpProfile();
$httpProfile->setEndpoint("asr.tencentcloudapi.com");
```

```
$clientProfile = new ClientProfile();
$clientProfile->setHttpProfile($httpProfile);
$client = new AsrClient($cred, "ap-shanghai", $clientProfile);

$req = new DescribeTaskStatusRequest();

$params = '{"TaskId":123456}';
$req->fromJsonString($params);


$resp = $client->DescribeTaskStatus($req);

print_r($resp->toJsonString());
}
catch(TencentCloudSDKException $e) {
echo $e;
}
```

Python SDK

最近更新时间：2020-07-24 14:56:06

接入准备

SDK获取

录音文件语音识别 Python SDK 获取，请参考： Python SDK 安装及相关环境说明

接入须知

开发者在调用前请先查看录音文件语音识别的 接口说明，了解接口的**使用要求**和**使用步骤**。

快速接入

以下分别是通过**语音 URL**和**本地语音上传**请求方式的 demo，以及**轮询接口**查询识别结果，来帮助客户快速接入。

1. 通过下面的录音文件识别请求中的两种接入方式的 demo快速请求，进入 API 密钥管理页面 获取 AppID、SecretId、SecretKey，并在代码中对应的位置配置好用户参数。
2. 然后在项目中使用以下的 demo，来快速获取识别结果。

- **通过语音 URL 方式请求**

```
# -*- coding: utf-8 -*-
from tencentcloud.common import credential
from tencentcloud.common.profile.client_profile import ClientProfile
from tencentcloud.common.profile.http_profile import HttpProfile
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.asr.v20190614 import asr_client, models
import base64
import io
import sys
if sys.version_info[0] == 3:
sys.stdout = io.TextIOWrapper(sys.stdout.buffer,encoding='utf-8')

#采用 URL 方式请求
try:
#重要，此处<Your SecretId><Your SecretKey>需要替换成客户自己的账号信息，获取方法：
#https://cloud.tencent.com/product/asr/getting-started
cred = credential.Credential("Your SecretId", "Your SecretKey")
httpProfile = HttpProfile()
httpProfile.endpoint = "asr.tencentcloudapi.com"
clientProfile = ClientProfile()
clientProfile.httpProfile = httpProfile
clientProfile.signMethod = "TC3-HMAC-SHA256"
client = asr_client.AsrClient(cred, "ap-shanghai", clientProfile)

req = models.CreateRecTaskRequest()
params = {"ChannelNum":1,"ResTextFormat":0,"SourceType":0}
req._deserialize(params)
req.EngineModelType = "16k_zh"
req.Url = "https://asr-audio-1300466766.cos.ap-nanjing.myqcloud.com/test16k.wav"
resp = client.CreateRecTask(req)
print(resp.to_json_string())
```

```
except TencentCloudSDKException as err:
print(err)
```

- **通过本地语音上传方式请求**

```
# -*- coding: utf-8 -*-
from tencentcloud.common import credential
from tencentcloud.common.profile.client_profile import ClientProfile
from tencentcloud.common.profile.http_profile import HttpProfile
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.asr.v20190614 import asr_client, models
import base64
import io
import sys
if sys.version_info[0] == 3:
sys.stdout = io.TextIOWrapper(sys.stdout.buffer,encoding='utf-8')

#本地文件方式请求
try:
#重要，此处<Your SecretId><Your SecretKey>需要替换成客户自己的账号信息，获取方法：
#https://cloud.tencent.com/product/asr/getting-started
cred = credential.Credential("Your SecretId", "Your SecretKey")
httpProfile = HttpProfile()
httpProfile.endpoint = "asr.tencentcloudapi.com"
clientProfile = ClientProfile()
clientProfile.httpProfile = httpProfile
clientProfile.signMethod = "TC3-HMAC-SHA256"
client = asr_client.AsrClient(cred, "ap-shanghai", clientProfile)
#读取文件以及 base64
#此处可以下载测试音频 https://asr-audio-1300466766.cos.ap-nanjing.myqcloud.com/test16k.wav
with open('./test16k.wav', "rb") as f:
if sys.version_info[0] == 2:
content = base64.b64encode(f.read())
else:
content = base64.b64encode(f.read()).decode('utf-8')

req = models.CreateRecTaskRequest()
params = {"ChannelNum":1,"ResTextFormat":0,"SourceType":1}
req._deserialize(params)
req.EngineModelType = "16k_zh"
req.Data = content
resp = client.CreateRecTask(req)
print(resp.to_json_string())

except TencentCloudSDKException as err:
print(err)
```

- **查询录音文件识别结果**

```
from tencentcloud.common import credential
from tencentcloud.common.profile.client_profile import ClientProfile
from tencentcloud.common.profile.http_profile import HttpProfile
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
```

```
from tencentcloud.asr.v20190614 import asr_client, models
try:
#此处<Your SecretId><Your SecretKey>需要替换成客户自己的账号信息
cred = credential.Credential("Your SecretId", "Your SecretKey")
httpProfile = HttpProfile()
httpProfile.endpoint = "asr.tencentcloudapi.com"

clientProfile = ClientProfile()
clientProfile.httpProfile = httpProfile
client = asr_client.AsrClient(cred, "ap-shanghai", clientProfile)

req = models.DescribeTaskStatusRequest()
params = '{"TaskId":123456}'
req.from_json_string(params)

resp = client.DescribeTaskStatus(req)
print(resp.to_json_string())

except TencentCloudSDKException as err:
print(err)
```

Node.js SDK

最近更新时间：2020-08-20 16:29:12

接入准备

SDK获取

录音文件识别 Node.js SDK 以及 Demo 的下载地址：Node.js SDK

接入须知

开发者在调用前请先查看录音文件识别的 接口说明，了解**接口要求**和**接口使用步骤**。

开发环境

- **环境依赖**
  该接口支持 Node.js 7.10.1 及以上版本。
- **安装 SDK**
  下载文件后解压缩，添加文件到代码编辑器（WebStorm、VSCode 或其他编辑器），在`nodejs_record_asr_sdk_v3` 文件夹下执行`npm install`命令安装依赖。

快速接入

`demo` 文件夹下：`localRecordFileRequest.js` 以及 `urlRequest.js` 为录音识别请求 demo，`recognizeResult.js` 为轮询识别结果的 demo。

1. 通过下面的录音文件识别请求中的两种接入方式的 demo 或是直接运行 demo 文件夹下的 js 文件快速请求，进入 API 密钥管理页面 获取 AppID、SecretId、SecretKey，并在代码中对应的位置配置好用户参数。

2. 然后通过运行 demo 文件夹下的 js 文件 或 在项目中使用以下的 demo，来快速获取识别结果。

录音识别请求

通过语音 URL 请求

```
//将 require 中路径替换为项目中 SDK 的真实路径
const tencentcloud = require("../../nodejs_record_asr_sdk_v3");

// 导入对应产品模块的 client models 以及需要用到的模块
const AsrClient = tencentcloud.asr.v20190614.Client;
const models = tencentcloud.asr.v20190614.Models;
const Credential = tencentcloud.common.Credential;
const ClientProfile = tencentcloud.common.ClientProfile;
const HttpProfile = tencentcloud.common.HttpProfile;

// Your SecretId、Your SecretKey 需要替换成客户自己的账号信息
let cred = new Credential("Your SecretId", "Your SecretKey");

let httpProfile = new HttpProfile();
httpProfile.reqMethod = "POST";
httpProfile.reqTimeout = 30;
httpProfile.endpoint = "asr.tencentcloudapi.com";

let clientProfile = new ClientProfile();
clientProfile.httpProfile = httpProfile;
clientProfile.signMethod = "TC3-HMAC-SHA256";
// 实例化要请求产品(asr)的 client 对象
let client = new AsrClient(cred, "", clientProfile);
```

```
//通过语音 URL 方式调用

// 实例化一个请求对象
let req = new models.CreateRecTaskRequest();
// 设置接口需要的参数，参考 接入须知 中 [请求接口说明]
req.EngineModelType = '16k_zh';
req.ChannelNum = 1;
req.ResTextFormat = 0;
req.SourceType = 0;
req.Url = 'https://asr-audio-1300466766.cos.ap-nanjing.myqcloud.com/test16k.wav';

// 非必填参数，可根据业务自行添加
// req.HotwordId = '08003a00000000000000000000000000';
// req.FilterDirty = 0;
// req.FilterModal = 0;
// req.ConvertNumMode = 0;
// req.CallbackUrl = '';


// 通过 client 对象调用想要访问的接口，需要传入请求对象以及响应回调函数
client.CreateRecTask(req, function(errMsg, response) {
if (errMsg) {
console.log(errMsg);
return;
}

console.log(response.to_json_string());
});
```

通过上传本地录音文件请求

```
//将 require 中路径替换为项目中 SDK 的真实路径
const tencentcloud = require("../../nodejs_record_asr_sdk_v3");

// 导入对应产品模块的 client models 以及需要用到的模块
const AsrClient = tencentcloud.asr.v20190614.Client;
const models = tencentcloud.asr.v20190614.Models;
const Credential = tencentcloud.common.Credential;
const ClientProfile = tencentcloud.common.ClientProfile;
const HttpProfile = tencentcloud.common.HttpProfile;

// Your SecretId、Your SecretKey 需要替换成客户自己的账号信息
let cred = new Credential("Your SecretId", "Your SecretKey");

let httpProfile = new HttpProfile();
httpProfile.reqMethod = "POST";
httpProfile.reqTimeout = 30;
httpProfile.endpoint = "asr.tencentcloudapi.com";

let clientProfile = new ClientProfile();
clientProfile.httpProfile = httpProfile;
clientProfile.signMethod = "TC3-HMAC-SHA256";
// 实例化要请求产品(asr)的 client 对象
let client = new AsrClient(cred, "", clientProfile);
```

```
//通过本地语音上传方式调用

const fs = require('fs');
const path = require('path');
//需要将"../demo/test16k.wav"替换为用户录音文件地址
let filePath = path.resolve('../demo/test16k.wav');
let data = fs.readFileSync(filePath);
//将音频文件转为base64格式，注意：转为base64后数据要小于5MB，否则不能成功请求识别
data = Buffer.from(data).toString('base64');
//此数据长度为数据未进行base64编码时的数据长度
let dataLen = fs.statSync('../demo/test16k.wav').size;

// 实例化一个请求对象
let req = new models.CreateRecTaskRequest();
// 设置接口需要的参数，参考 接入须知 中 [请求接口说明]
req.EngineModelType = '16k_zh';
req.ChannelNum = 1;
req.ResTextFormat = 0;
req.SourceType = 1;
req.Data = data;
req.DataLen = dataLen;
// 非必填参数，可根据业务自行添加
// req.HotwordId = '08003a00000000000000000000000000';
// req.FilterDirty = 0;
// req.FilterModal = 0;
// req.ConvertNumMode = 0;
// req.CallbackUrl = '';


// 通过 client 对象调用想要访问的接口，需要传入请求对象以及响应回调函数
client.CreateRecTask(req, function(errMsg, response) {
if (errMsg) {
console.log(errMsg);
return;
}

console.log(response.to_json_string());
});
```

识别结果查询

录音文件识别结果查询

```
//调用录音识别结果查询接口
let reqResult = new models.DescribeTaskStatusRequest();
//reqResult.TaskId 为创建识别任务时 response 里的 TaskId 字段
//示例 TaskId 不可用，需要客户替换成可用 TaskId
reqResult.TaskId = 080387961;

client.DescribeTaskStatus(reqResult, function(errMsg, response) {
if (errMsg) {
console.log(errMsg);
return;
}
```

```
console.log(response.to_json_string());
});
```

一句话识别 SDK
接口说明

最近更新时间：2020-09-17 16:06:34

接口描述

本接口用于对60秒之内的短音频文件进行快速识别。

- 支持中文普通话、英语、粤语、日语和上海话方言的识别
- 支持本地语音上传和语音 URL 上传两种请求方式

接口说明请观看视频：

点击查看视频

接口要求

使用一句话识别 SDK 时，需按照以下要求。

内容	说明
支持语言	中文普通话、英语、粤语、日语、上海话方言
音频属性	采样率：16000Hz或8000Hz、采样精度：16bits、声道：单声道，查看音频属性请参见 常见问题
音频格式	wav、mp3
请求地址	https://asr.tencentcloudapi.com/?{请求参数}
数据长度	一句话识别限制音频时长不超过60s，数据长度不可以超过3MB
免费额度	每月5000次
请求频率限制	25次/秒

使用步骤

1. 首先获取您的 AppID、SecretID 和 SecretKey。在使用该接口前，需要在 语音识别控制台 开通服务，并进入 API 密钥管理页面 新建密钥，生成 AppID、SecretID 和 SecretKey ，用于 SDK 调用时生成签名，签名将用来进行接口鉴权。
2. 查看您的音频属性和格式，应满足接口支持的属性和格式，否则会请求失败。
3. 通过 SDK 提交一句话识别的请求。
4. 如果返回的 code = 0，表示请求成功，同时一句话识别系统会将识别结果返回给客户端；如果返回的 code 不为0，请参见 错误码 查看错误详情。
5. 如果识别效果有问题，请参考 识别效果问题排查文档 进行排查。

一句话识别请求

请求参数

请求参数主要由**请求 URL** 和**请求头部**组成。

请求 URL 示例

用户通过语音 URL（https%3a%2f%2fruskin-1256085166.cos.ap-guangzhou.myqcloud.com%2ftest.wav，这是经过 urlencode 编码后的地址）上传的方式（SourceType 为0）请求一句话识别服务，服务的引擎模型为：16k通用模型（EngSerViceType = 16k_0），一句话语音的采样率为16k，声道数为单声道。

```
https://asr.tencentcloudapi.com/?Action=SentenceRecognition
&ProjectId=0
&SubServiceType=2
&EngSerViceType=16k_0
&SourceType=0
&Url=https%3a%2f%2fruskin-1256085166.cos.ap-guangzhou.myqcloud.com%2ftest.wav
&SecretId=111
&Timestamp=111
```

```
&VoiceFormat=wav
&UsrAudioKey=www
&<公共请求参数>
```

请求 URL 参数说明

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见 公共请求参数。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：SentenceRecognition。
Version	是	String	公共参数，本接口取值：2019-06-14。
Region	否	String	公共参数，本接口不需要传递此参数。
ProjectId	是	Integer	腾讯云项目 ID，可填0，总长度不超过1024字节。
SubServiceType	是	Integer	子服务类型，2：一句话识别。
EngineServiceType	是	String	引擎模型类型。 电话场景： 8k_en：电话 8k 英语； 8k_zh：电话 8k 中文普通话通用； 非电话场景： 16k_zh：16k 中文普通话通用； 16k_en：16k 英语； 16k_ca：16k 粤语； 16k_ja：16k 日语； 16k_wuu-SH：16k上海话方言。
HotwordId	否	String	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。
SourceType	是	Integer	语音数据来源， 0：语音 URL、1：语音数据（post body）。
VoiceFormat	是	String	识别音频的音频格式，wav 或者 mp3。
Url	否	String	语音 URL，公网可下载。当 SourceType 值为 0（语音 URL 上传）时须填写该字段，为 1 时不填、URL 的长度大于 0，小于 2048，需进行 urlencode 编码。音频时间长度要小于60s。
Data	否	String	语音数据，当 SourceType 值为1（本地语音数据上传）时必须填写，当 SourceType 值为0（语音 URL 上传）可不写。要使用 base64 编码(采用 Python 语言时注意读取文件应该为 string 而不是 byte，以 byte 格式读取后要 decode()。编码后的数据不可带有回车换行符)。音频数据要小于600KB。
DataLen	否	Integer	数据长度，单位为字节。当 SourceType 值为1（本地语音数据上传）时必须填写，当 SourceType 值为 0（语音 URL上传）可不写（此数据长度为数据未进行 base64 编码时的数据长度）。
FilterDirty	否	Integer	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 * 。
FilterModal	否	Integer	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 。
FilterPunc	否	Integer	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。默认为 0。
ConvertNumMode	否	Integer	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。

一句话识别结果返回

同步返回示例

```
{
"Response": {
"Result": "腾讯云语音识别欢迎您。",
"AudioDuration": 5000,
```

```
"RequestId": "8984d9a9-343f-4c67-8fd9-5c79510a12da"
}
}
```

返回参数说明

参数名称	类型	描述
Result	String	识别结果
RequestId	String	唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId
AudioDuration	Integer	请求的音频时长，单位为 ms

错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码

错误码	说明
FailedOperation.ServiceIsolate	账号因为欠费停止服务，请在腾讯云账户充值
FailedOperation.UserHasNoFreeAmount	账号本月免费额度已用完
FailedOperation.UserNotRegistered	服务未开通，请在腾讯云官网语音识别控制台开通服务
InternalError	内部错误
InternalError.ErrorConfigure	初始化配置失败
InternalError.ErrorCreateLog	创建日志失败
InternalError.ErrorDownFile	下载音频文件失败
InternalError.ErrorFailNewprequest	新建数组失败
InternalError.ErrorFailWritetodb	写入数据库失败
InternalError.ErrorFileCannotopen	文件无法打开
InternalError.ErrorGetRoute	获取路由失败
InternalError.ErrorMakeLogpath	创建日志路径失败
InternalError.ErrorRecognize	识别失败
InvalidParameter.ErrorContentlength	请求数据长度无效
InvalidParameter.ErrorParamsMissing	参数不全
InvalidParameter.ErrorParsequest	解析请求数据失败
InvalidParameterValue	参数取值错误
InvalidParameterValue.ErrorInvalidAppid	AppId 无效
InvalidParameterValue.ErrorInvalidClientip	ClientIp 无效
InvalidParameterValue.ErrorInvalidEngservice	EngSerViceType 无效
InvalidParameterValue.ErrorInvalidProjectid	ProjectId 无效
InvalidParameterValue.ErrorInvalidRequestid	RequestId 无效
InvalidParameterValue.ErrorInvalidSourcetype	SourceType 无效
InvalidParameterValue.ErrorInvalidSubservicetype	SubserviceType 无效

错误码	说明
InvalidParameterValue.ErrorInvalidUrl	Url 无效
InvalidParameterValue.ErrorInvalidUseraudiokey	UsrAudioKey 无效
InvalidParameterValue.ErrorInvalidVoiceFormat	音频编码格式不支持
InvalidParameterValue.ErrorInvalidVoicedata	音频数据无效

Android SDK

最近更新时间：2020-11-03 17:19:03

Android SDK 接入请观看视频：

点击查看视频

接入准备

SDK 获取

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

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（GPRS、3G 或 Wi-Fi 等），且系统为 Android 4.0.3 及其以上版本。

开发环境

1）添加一句话识别 SDK aar

将 qcloudasrsdk_1.0_release.aar 放在 libs 目录下，在 App 的 build.gradle 文件中添加以下代码。

```
implementation(name: 'qcloudasrsdk_2.0_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'
implementation 'com.squareup.okhttp3:okhttp:4.0.0-RC1'
implementation 'com.squareup.okio:okio:1.11.0'
implementation 'org.slf4j:slf4j-api:1.7.25'
```

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"/>
< uses-permission android:name="android.permission.WRITE_SETTINGS" />
< uses-permission android:name="android.permission.READ_PHONE_STATE"/>
< uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
< uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
```

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**

```
String audioUrl = "https://img.soulapp.cn/audio/2019-07-22/9ed1a797-93b5-4268-be6d-5660cc3e894e.mp3";
recognizer.recognize(audioUrl, QCloudAudioFormat.QCloudAudioFormatMp3, QCloudAudioFrequence.QCloudAudioFrequence16k);
```

- **通过语音数据调用**

```
AssetManager am = getResources().getAssets();
InputStream is = am.open("onesentence.mp3");
int length = is.available();
byte[] audioData = new byte[length];
is.read(audioData);
recognizer.recognize(audioData, QCloudAudioFormat.QCloudAudioFormatMp3, QCloudAudioFrequence.QCloudAudioFrequence16k);
```

- **通过 SDK 内置录音器**

```
recognizer.recognizeWithRecorder();
```

关键类说明

QCloudOneSentenceRecognizer :一句话识别入口类

```
/**
* 初始化方法，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);
/**
* 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 语音数据采样率，QCloudAudioFrequence
*/
public void recognize(String audioUrl, QCloudAudioFormat audioFormat, QCloudAudioFrequence 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);
}
```

iOS SDK

最近更新时间：2020-12-22 11:16:34

iOS SDK 接入请观看视频：

点击查看视频

接入准备

SDK 获取

一句话识别的 iOS SDK 以及 Demo 的下载地址：iOS SDK。

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（GPRS、3G 或 Wi-Fi 网络等），且系统为 **iOS 9.0**及以上版本。

开发环境

在工程 `info.plist` 添加以下设置：

- **设置 NSAppTransportSecurity 策略，添加如下内容：**

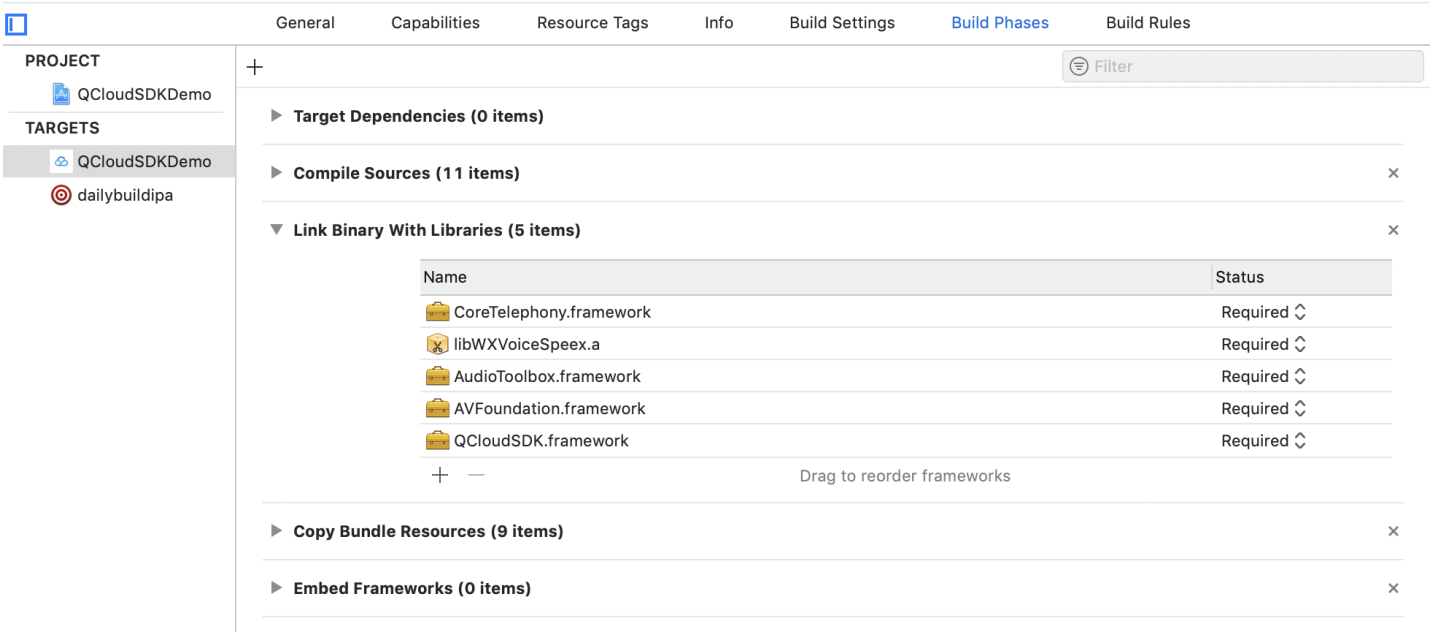
```
<key>NSAppTransportSecurity</key>
<dict>
<key>NSExceptionDomains</key>
<dict>
<key>qcloud.com</key>
<dict>
<key>NSExceptionAllowsInsecureHTTPLoads</key>
<true/>
<key>NSExceptionMinimumTLSVersion</key>
<string>TLSv1.2</string>
<key>NSIncludesSubdomains</key>
<true/>
<key>NSRequiresCertificateTransparency</key>
<false/>
</dict>
</dict>
</dict>
```

- **申请系统麦克风权限，添加如下内容：**

```
<key>NSMicrophoneUsageDescription</key>
<string>需要使用了的麦克风采集音频</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：**
  - AVFoundation.framework
  - AudioToolbox.framework
  - QCloudSDK.framework
  - CoreTelephony.framework
  - libWXVoiceSpeex.a

添加完如下图所示：



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

- a. 通过语音 URL 调用

```
- (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 采样率
[_recognizer recoginizeWithUrl:url voiceFormat:kQCloudVoiceFormatWAV frequence:kQCloudEngSerViceType16k];
}
```

- b. 通过语音数据调用

```
- (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 采样率
[_recognizer recoginizeWithData:audioData voiceFormat:kQCloudVoiceFormatWAV frequence:kQCloudEngSerViceType16k];
}
```

- **c. 通过指定参数调用**

```
- (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，见kQCloudVoiceFormat定义
params.voiceFormat = kQCloudVoiceFormatWAV;
//设置语音数据来源，见QCloudAudioSourceType定义
params.sourceType = QCloudAudioSourceTypeUrl;
//设置采样率，见kQCloudEngSerViceType定义
params.engSerViceType = kQCloudEngSerViceType16k;
[_recognizer recognizeWithParams:params];
}
```

- **d. 通过 SDK 内置录音器调用**

```
- (void)recognizeWithRecorder {
[_recognizer startRecognizeWithRecorder];
}
```

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;
/**
* 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;
```

QCloudConfig 初始化方法说明

参考一句话识别接口说明中的使用步骤，获取 AppID、SecretID 和 SecretKey。

```
/**
* 初始化方法
* @param appid 腾讯云 appId
* @param secretId 腾讯云 secretId
* @param secretKey 腾讯云 secretKey
* @param projectId 腾讯云 projectId
*/
```

```
- (instancetype)initWithAppId:(NSString *)appid
secretId:(NSString *)secretId
secretKey:(NSString *)secretKey
projectId:(NSString *)projectId;
```

QCloudSentenceRecognizerDelegate 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:(null
able 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;
@end
```

PHP SDK

最近更新时间：2020-08-04 15:26:05

接入准备

SDK获取

一句话识别 PHP SDK 获取，请参考：PHP SDK 依赖环境及获取安装 。

接入须知

开发者在调用前请先查看一句话语音识别的 接口说明，了解接口的**使用要求**和**使用步骤**。

快速接入

以下分别是通过**语音 URL** 和**本地语音上传**请求方式的 demo，来帮助用户快速接入。

- **通过语音 URL 方式请求**

```
<?php
require_once './tencentcloud-sdk-php/TCloudAutoLoader.php';
use TencentCloud\Common\Credential;
use TencentCloud\Common\Profile\ClientProfile;
use TencentCloud\Common\Profile\HttpProfile;
use TencentCloud\Common\Exception\TencentCloudSDKException;
use TencentCloud\Asr\V20190614\AsrClient;
use TencentCloud\Asr\V20190614\Models\SentenceRecognitionRequest;

//通过语音URL方式调用
try {
//重要：<Your SecretId>、<Your SecretKey>需要替换成用户自己的账号信息
//请参考接口说明中的使用步骤1进行获取。
$cred = new Credential("Your SecretId", "Your SecretKey");
$httpProfile = new HttpProfile();
$httpProfile->setEndpoint("asr.tencentcloudapi.com");

$clientProfile = new ClientProfile();
$clientProfile->setHttpProfile($httpProfile);
$client = new AsrClient($cred, "ap-shanghai", $clientProfile);

$req = new SentenceRecognitionRequest();

$params = '{"ProjectId":0,"SubServiceType":2,"EngSerViceType":"16k_zh","SourceType":0,"Url":"https://asr-audio-1300466766.c
os.ap-nanjing.myqcloud.com/test16k.wav","VoiceFormat":"wav","UsrAudioKey":"session-123"}';
$req->fromJsonString($params);


$resp = $client->SentenceRecognition($req);

print_r($resp->toJsonString());
}
catch(TencentCloudSDKException $e) {
echo $e;
}
```

- **通过本地语音上传方式请求**

```
<?php
require_once './tencentcloud-sdk-php/TCloudAutoLoader.php';
use TencentCloud\Common\Credential;
use TencentCloud\Common\Profile\ClientProfile;
use TencentCloud\Common\Profile\HttpProfile;
use TencentCloud\Common\Exception\TencentCloudSDKException;
use TencentCloud\Asr\V20190614\AsrClient;
use TencentCloud\Asr\V20190614\Models\SentenceRecognitionRequest;

//通过本地语音上传方式调用
try {
//重要：<Your SecretId>、<Your SecretKey>需要替换成用户自己的账号信息
//请参考接口说明中的使用步骤1进行获取。
$cred = new Credential("Your SecretId", "Your SecretKey");
$httpProfile = new HttpProfile();
$httpProfile->setEndpoint("asr.tencentcloudapi.com");

$clientProfile = new ClientProfile();
$clientProfile->setHttpProfile($httpProfile);
$clientProfile->setSignMethod("TC3-HMAC-SHA256");
$client = new AsrClient($cred, "ap-shanghai", $clientProfile);

$req = new SentenceRecognitionRequest();

$params = '{"ProjectId":0,"SubServiceType":2,"EngSerViceType":"16k_zh","SourceType":1,"Url":"","VoiceFormat":"wav","UsrAudi
oKey":"session-123"}';
$req->fromJsonString($params);
$data = file_get_contents('./test.wav');
$encodeData = base64_encode($data);
$req->Data = $encodeData;
$req->DataLen = strlen($data);

$resp = $client->SentenceRecognition($req);

print_r($resp->toJsonString());
}
catch(TencentCloudSDKException $e) {
echo $e;
}
```

C++ SDK

最近更新时间：2020-08-04 15:25:52

接入准备

SDK 获取

一句话识别 C++ SDK 以及 Demo 的下载地址：C++ SDK。

接入须知

开发者在调用前请先查看一句话语音识别的 接口说明，了解接口的**使用要求**和**使用步骤**。

开发环境

- **编译 Demo**，如失败需确认以下环境：

```
//下载sdk
tar -xzf c++_sentence_asr_sdk.tar.gz
cd c++_sentence_asr_sdk
cmake ./
make
//如果编译并未报错则跳过以下环境检测，否则可根据错误类型去校验库
```

- **安装 gcc g++**

```
1.RedHat 系列系统：
yum install -y gcc gcc-c++ make automake
//安装 gcc 等必备程序包（已安装则略过此步）
yum install -y wget
2.Debian系列系统：
apt-get install gcc g++
```

- **安装 CMake 工具**

```
// cmake 版本要大于3.5
wget https://cmake.org/files/v3.5/cmake-3.5.2.tar.gz
tar -zxvf cmake-3.5.2.tar.gz
cd cmake-3.5.2
sudo ./bootstrap --prefix=/usr
sudo make
sudo make install
```

- **依赖库安装及编译**

客户需自行安装版本大于7.44.0的 **curl**。下载 curl 文件，解压并进入源码目录执行如下命令：

```
sudo ./configure
sudo cmake ./
sudo make
sudo make install
```

- **openssl**

本 SDK 提供，目录为：c++*realtime*|asr_sdk/lib。如果不适合客户系统，请客户自行安装方法，版本1.0.2f，下载 wget 源码 并执行以下命令：

```
1. 更新 zlib
RedHat系列:yum install -y zlib
Debian系列:sudo apt-get install zlib1g zlib1g.dev
2. 安装
tar zxf openssl-1.0.2f.tar.gz
cd openssl-1.0.2f
sudo ./config shared zlib
sudo make
sudo make install
自行替换 c++_realtime_asr_sdk/lib 下面的库文件
```

快速接入

开发流程介绍

配置用户信息

- 进入 API 密钥管理页面 获取 AppID、SecretId、SecretKey 信息，并按如下步骤配置用户信息和请求 URL 参数。

```
//需要配置成用户账号信息 c++_sentence_asr_sdk/src/TCloudSASR.h
//请求的用户信息配置
struct TCloudSASRConfig{
string SecretId; //对应用户的SecretId
string SecretKey;//对应用户的SecretKey
...
};
```

初始化请求参数

- 定义请求参数，具体参数参考请求参数说明。
- 请参考 接口说明

设置用户密钥

- 赋值用户的SecretId 和 SecretKey
- 请参考 开发流程介绍

初始化请求

- 调用 TCloudSASR::SetTCloudConfig 接口初始化请求
- 请参考 主要接口方法说明

发送请求并获取结果

- 调用 TCloudSASR::SendVoiceData 接口获取结果
- 请参考 主要接口方法说明

SDK 已提供各个接口源码，用户可根据自身需要进行更改。

主要接口方法说明

TCloudSASR::SetTCloudConfig

```
/*
*设置相关参数
*params：
```

```
* SecretId，SecretKey：官网获得的 SecretId，SecretKey
* EngSerViceTyp：引擎类型引擎模型类型。8k:8k通用，16k:16k通用。
* SourceType：语音数据来源。0：语音 URL；1：语音数据（post body）
* VoiceFormat：识别音频的音频格式（mp3，wav）
*/
bool SetTCloudConfig(const string &SecretId, const string &SecretKey,const string &EngSerViceType="16k",const string &Sourc
eType="0",const string &VoiceFormat="wav");
```

TCloudSASR::SendVoiceData

```
/*
* 发型音频数据
* fileURI：本地音频文件或者远程音频文件地址
* pResponse：识别结果
*
* */
int SendVoiceData(const string fileURI,string &pResponse);
```

请求 demo

```
//在 sdk 的根目录下执行
cd demo
make
./SASRtest
```

快速入门示例

```
//引用 SDK 头文件
#include "TCloudSASR.h"
#include"iostream"
using namespace std;

int main(int argc, const char * argv[]) {
//重要：<Your SecretId>、<Your SecretKey>需要替换成客户自己的账号信息
//请参考接口说明中的使用步骤1进行获取。
string SecretId="Your SecretId";
string SecretKey="Your SecretKey";
//8k or 16k
string EngSerViceType="16k_zh";
//1 or 0
//填1的时候 URL 可以设置为空
string SourceType="0";
//wav or mp3
string VoiceFormat="wav";
string fileURI="https://asr-audio-1300466766.cos.ap-nanjing.myqcloud.com/test16k.wav";
//string fileURI="test.wav";
TCloudSASR *sasrdemo=new TCloudSASR();

//设置请求参数以及校验参数
bool configres=sasrdemo->SetTCloudConfig(SecretId,SecretKey,EngSerViceType,SourceType,VoiceFormat);
if(configres==false){
cout<<"the params are wrong ,please check out.";
return -1;
```

```
}
string pResponse;
//发送请求并获取回包
int res=sasrdemo->SendVoiceData(fileURI,pResponse);
if(res!=0){
cout<<"send voice data may have some wrong.";
return -2;
}
cout<<pResponse<<endl;
delete sasrdemo;
return 0;
}
```

Python SDK

最近更新时间：2020-07-24 14:56:02

接入准备

SDK获取

一句话识别 Python SDK 获取，请参考：Python SDK 依赖环境及获取安装说明 。

接入须知

开发者在调用前请先查看一句话语音识别的 接口说明，了解接口的**使用要求**和**使用步骤**。

快速接入

以下分别是通过**语音 URL** 和**本地语音上传**请求方式的 demo，来帮助用户快速接入。

- **通过语音 URL 方式请求**

```
# -*- coding: utf-8 -*-
from tencentcloud.common import credential
from tencentcloud.common.profile.client_profile import ClientProfile
from tencentcloud.common.profile.http_profile import HttpProfile
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.asr.v20190614 import asr_client, models
import base64
import io
import sys
if sys.version_info[0] == 3:
sys.stdout = io.TextIOWrapper(sys.stdout.buffer,encoding='utf-8')

#采用 URL 方式请求
try:
#重要，此处<Your SecretId><Your SecretKey>需要替换成客户自己的账号信息，获取方法：
#https://cloud.tencent.com/product/asr/getting-started
cred = credential.Credential("Your SecretId", "Your SecretKey")
httpProfile = HttpProfile()
httpProfile.endpoint = "asr.tencentcloudapi.com"
clientProfile = ClientProfile()
clientProfile.httpProfile = httpProfile
clientProfile.signMethod = "TC3-HMAC-SHA256"
client = asr_client.AsrClient(cred, "ap-shanghai", clientProfile)
#发送请求
req = models.SentenceRecognitionRequest()
params = {"ProjectId":0,"SubServiceType":2,"SourceType":0,"UsrAudioKey":"session-123"}
req._deserialize(params)
req.EngSerViceType = "16k_zh"
req.VoiceFormat = "wav"
req.Url = "https://asr-audio-1300466766.cos.ap-nanjing.myqcloud.com/test16k.wav"
resp = client.SentenceRecognition(req)
print(resp.to_json_string())

#windows 如果是 GBK 编码则用下面 print 语句替换上面 print 语句
#print(resp.to_json_string().decode('UTF-8').encode('GBK') )
```

```
except TencentCloudSDKException as err:
print(err)
```

- **通过本地语音上传方式请求**

```
# -*- coding: utf-8 -*-
from tencentcloud.common import credential
from tencentcloud.common.profile.client_profile import ClientProfile
from tencentcloud.common.profile.http_profile import HttpProfile
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.asr.v20190614 import asr_client, models
import base64
import io
import sys
if sys.version_info[0] == 3:
sys.stdout = io.TextIOWrapper(sys.stdout.buffer,encoding='utf-8')

#本地文件方式请求
try:
#重要，此处<Your SecretId><Your SecretKey>需要替换成客户自己的账号信息，获取方法：
#https://cloud.tencent.com/product/asr/getting-started
cred = credential.Credential("Your SecretId", "Your SecretKey")
httpProfile = HttpProfile()
httpProfile.endpoint = "asr.tencentcloudapi.com"
clientProfile = ClientProfile()
clientProfile.httpProfile = httpProfile
clientProfile.signMethod = "TC3-HMAC-SHA256"
client = asr_client.AsrClient(cred, "ap-shanghai", clientProfile)
#读取文件以及 base64
#此处可以下载测试音频 https://asr-audio-1300466766.cos.ap-nanjing.myqcloud.com/test16k.wav
with open('./test16k.wav', "rb") as f:
if sys.version_info[0] == 2:
content = base64.b64encode(f.read())
else:
content = base64.b64encode(f.read()).decode('utf-8')
#发送请求
req = models.SentenceRecognitionRequest()
params = {"ProjectId":0,"SubServiceType":2,"SourceType":1,"UsrAudioKey":"session-123"}
req._deserialize(params)
req.DataLen = len(content)
req.Data = content
req.EngSerViceType = "16k_zh"
req.VoiceFormat = "wav"
resp = client.SentenceRecognition(req)
print(resp.to_json_string())

#windows 如果是 GBK 编码则用下面 print 语句替换上面 print 语句
#print(resp.to_json_string().decode('UTF-8').encode('GBK') )

except TencentCloudSDKException as err:
print(err)
```

Java SDK

最近更新时间：2020-08-04 15:25:58

接入准备

SDK获取

一句话识别 Java SDK 获取，请参考：Java SDK 依赖环境及获取安装说明。

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**。

快速接入

以下分别是通过**语音 URL** 和**本地语音上传**请求方式的 demo，来帮助用户快速接入。

- **通过语音 URL 方式请求**

```
import com.tencentcloudapi.common.Credential;
import com.tencentcloudapi.common.profile.ClientProfile;
import com.tencentcloudapi.common.profile.HttpProfile;
import com.tencentcloudapi.common.exception.TencentCloudSDKException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.Base64;
import com.tencentcloudapi.asr.v20190614.AsrClient;
import com.tencentcloudapi.asr.v20190614.models.SentenceRecognitionRequest;
import com.tencentcloudapi.asr.v20190614.models.SentenceRecognitionResponse;

public class SentenceRecognition
{
public static void main(String [] args) throws IOException {
// 采用语音URL方式调用
try{
//重要：<Your SecretId>、<Your SecretKey>需要替换成用户自己的账号信息
//请参考接口说明中的使用步骤1进行获取。
Credential cred = new Credential("Your SecretId", "Your SecretKey");

HttpProfile httpProfile = new HttpProfile();
httpProfile.setEndpoint("asr.tencentcloudapi.com");

ClientProfile clientProfile = new ClientProfile();
clientProfile.setHttpProfile(httpProfile);

AsrClient client = new AsrClient(cred, "ap-shanghai", clientProfile);

String params = "{\"ProjectId\":0,\"SubServiceType\":2,\"EngSerViceType\":\"16k_zh\",\"SourceType\":0,\"Url\":\"https://asr
-audio-1300466766.cos.ap-nanjing.myqcloud.com/test16k.wav\",\"VoiceFormat\":\"wav\",\"UsrAudioKey\":\"session-123\"}";
SentenceRecognitionRequest req = SentenceRecognitionRequest.fromJsonString(params, SentenceRecognitionRequest.class);

SentenceRecognitionResponse resp = client.SentenceRecognition(req);
```

```
System.out.println(SentenceRecognitionRequest.toJsonString(resp));
} catch (TencentCloudSDKException e) {
System.out.println(e.toString());
}
}

}
```

- **通过本地语音上传方式请求**

```
import com.tencentcloudapi.common.Credential;
import com.tencentcloudapi.common.profile.ClientProfile;
import com.tencentcloudapi.common.profile.HttpProfile;
import com.tencentcloudapi.common.exception.TencentCloudSDKException;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.Base64;

import com.tencentcloudapi.asr.v20190614.AsrClient;

import com.tencentcloudapi.asr.v20190614.models.SentenceRecognitionRequest;
import com.tencentcloudapi.asr.v20190614.models.SentenceRecognitionResponse;

public class SentenceRecognition
{
public static void main(String [] args) throws IOException {

//采用本地语音上传方式调用
try{
//重要，此处<Your SecretId><Your SecretKey>需要替换成客户自己的账号信息，获取方法：
//https://cloud.tencent.com/document/product/441/6203
//具体路径：点控制台右上角您的账号-->选：访问管理-->点左边菜单的：访问密钥-->API 密钥管理
Credential cred = new Credential("Your SecretId", "Your SecretKey");

HttpProfile httpProfile = new HttpProfile();
httpProfile.setEndpoint("asr.tencentcloudapi.com");

ClientProfile clientProfile = new ClientProfile();
clientProfile.setHttpProfile(httpProfile);
clientProfile.setSignMethod("TC3-HMAC-SHA256");
AsrClient client = new AsrClient(cred, "ap-shanghai", clientProfile);

String params = "{\"ProjectId\":0,\"SubServiceType\":2,\"EngSerViceType\":\"16k_zh\",\"SourceType\":1,\"Url\":\"\",\"VoiceF
ormat\":\"wav\",\"UsrAudioKey\":\"session-123\"}";
SentenceRecognitionRequest req = SentenceRecognitionRequest.fromJsonString(params, SentenceRecognitionRequest.class);

File file = new File("/Users/ruskinli/eclipse-workspace/TencentSentence/src/test.wav");
FileInputStream inputFile = new FileInputStream(file);
byte[] buffer = new byte[(int)file.length()];
req.setDataLen(file.length());
```

```
inputFile.read(buffer);
inputFile.close();
String encodeData = Base64.getEncoder().encodeToString(buffer);
req.setData(encodeData);

SentenceRecognitionResponse resp = client.SentenceRecognition(req);

System.out.println(SentenceRecognitionRequest.toJsonString(resp));
} catch (TencentCloudSDKException e) {
System.out.println(e.toString());
}

}

}
```