语音降噪接口

一、语音降噪优势

语音降噪技术提供了一种先进的噪声抑制解决方案。笔声语音降噪技术利用了深度学习和人工智能算法，可以实时地降低背景噪声，使语音通信更清晰。

笔声语音降噪的技术有以下几个关键特点：

1. 双向降噪：不仅可以降低说话者周围的噪声，还可以降低听话者端的噪声。这意味着无论是从麦克风采集到的语音信号，还是从扬声器播放的语音信号，都可以得到有效的降噪处理。
2. 实时性：具有低延迟和实时处理的能力。这使得它非常适用于需要即时交流的场景，例如语音通话、视频会议和实时语音聊天等。
3. 适用广泛：可以应用于各种设备和应用程序，包括电脑、智能手机、VoIP软件、游戏通话等。用户可以通过安装适配笔声语音降噪技术的应用程序或将其集成到现有的通信应用程序中来获得语音降噪的功能。

二、语音降噪API文档

1、接口说明

本接口支持使用restful接口，处理语音文件；有关实时接口，请联系我司商务。

1.1 支持的音频格式有 WAV/MP3/WMA/FLAC/AMR/OPUS/M4A/AAC

1.2 支持音频详情：

当前只支持16KHz 单声道音频文件的降噪，多种格式的支持下一版本补充

- 音频时长：不超过5小时
- 音频大小：不超过600 MB
- 音频声道：单声道、双声道
- 采样率：支持8000、16000、44100和48000 Hz
- 采样精度：16 bits、8 bits
- 编码格式：文件格式和音频解码需结合查看，如下表第一行：只支持pcm封装或wav封装的pcm音频，不支持其他文件格式封装的pcm音频。 不同的音频解码格式需在请求体中填写不同的audio_encode.

文件格式	音频解码	audio_encode
.wav	pcm	pcm
.m4a .aac	aac	aac
.mp3	mpeg2	mpeg2
.opus	opus	opus

文件格式	音频解码	audio_encode
.flac	flac	flac

注意
a. opus编码的音频不支持44100 Hz的采样率
b. 仅 wav封装的pcm格式音频支持 8 bits采样精度

2 语音降噪接口API

- 接口描述： 上传音频文件的url，返回降噪后的音频文件url
- 请求格式：multipart/form-data格式
- 请求方式：POST
- URL: https://denoise-prod.abcpen.com/v1/denoise/create

2. 1 请求参数

2.1.1 Header

Header中输入正确的账号、时间戳及签名（X-App-Key、X-Timestamp、X-App-Signature），鉴权方式

2.1.2 Body体

Body

参数名	类型	是否必须	描述
audio_url	string	是	文件的URL，如：https://zos.abcpen.com/tts/zmeet/20221023/3058bca8-52cb-11ed-961e-00155dc6cbed.mp3
callback	string	否	支持http/https。填写后可通过回调获取结果。为空可使用查询获取结果。

2.1.3 返回数据实例

- 创建任务返回结果：返回包含任务task_id的JSON字符串

```
{"code":"0","data":{"task_id":"3950fc47-dc2c-4d17-9ef4-
d54e8ad55f9e"},"msg":"success"}
```

3、查询任务

- 接口描述： 传入task_id, 查询降噪的结果
- 请求格式： application/json

- 请求方式：POST
- URL: https://denoise-prod.abcpen.com/v1/denoise/query

3. 1 请求参数

3.1.1 Header

Header中输入正确的账号、时间戳及签名（X-App-Key、X-Timestamp、X-App-Signature），鉴权方式参考下述“signa生成"

3.1.2 Body体

Body

参数名	类型	是否必须	描述
task_id	string	是	降噪的任务id号，如 “8ba73ba7-08e3-4590-a379-5f77f7b6508d”

3.1.3 返回结果实例

- 返回的是JSON字符串：


```
{"code":"0","data":
{"audio_url":"https://zos.abcpen.com/zos/wav/denoise1.wav","audio_denoise_url":"http
s://zos.abcpen.com/denoise/abcpen/20230711/bf76ddc9-2f3c-453a-aa26-
8ea8ae879b62.wav","denoise_status":1},"msg":"success"}
```


4、 signa生成

a. python示例（）

```python
import hashlib
import hmac
import time
import base64

def get_signature_flytek(ts, app_id, app_secret):
    tt = (app_id + ts).encode('utf-8')
    md5 = hashlib.md5()
    md5.update(tt)
    baseString = md5.hexdigest()
    baseString = bytes(baseString, encoding='utf-8')

    apiKey = app_secret.encode('utf-8')
    signa = hmac.new(apiKey, baseString, hashlib.sha1).digest()
    signa = base64.b64encode(signa)
    signa = str(signa, 'utf-8')
    return signa
```

b. Java示例

- Java示例(具体参考github Java目录代码)

```
    // 生成握手参数
    public static String getHandShakeParams(String appId, String secretKey) {
        String ts = System.currentTimeMillis() / 1000 + "";
        String signa = "";
        try {
            signa = EncryptUtil.HmacSHA1Encrypt(EncryptUtil.MD5(appId + ts),
secretKey);
            return "?appid=" + appId + "&ts=" + ts + "&signa=" +
URLEncoder.encode(signa, "UTF-8");
        } catch (Exception e) {
            e.printStackTrace();
        }

        return "";
    }
```

- Java基础工具类

```
package com.abcpen.ai.rtasr.util;

import java.io.UnsupportedEncodingException;
import java.nio.charset.StandardCharsets;
import java.security.InvalidKeyException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.SignatureException;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

import org.apache.commons.codec.binary.Base64;

public class EncryptUtil {

    /**
     * 加密数字签名（基于HMACSHA1算法）
     *
     * @param encryptText
     * @param encryptKey
     * @return
     * @throws SignatureException
     */
    public static String HmacSHA1Encrypt(String encryptText, String encryptKey)
throws SignatureException {
        byte[] rawHmac = null;
        try {
            byte[] data = encryptKey.getBytes(StandardCharsets.UTF_8);
```

```
            SecretKeySpec secretKey = new SecretKeySpec(data, "HmacSHA1");
            Mac mac = Mac.getInstance("HmacSHA1");
            mac.init(secretKey);
            byte[] text = encryptText.getBytes(StandardCharsets.UTF_8);
            rawHmac = mac.doFinal(text);
        } catch (InvalidKeyException e) {
            throw new SignatureException("InvalidKeyException:" + e.getMessage());
        } catch (NoSuchAlgorithmException e) {
            throw new SignatureException("NoSuchAlgorithmException:" +
e.getMessage());
        }
        String oauth = new String(Base64.encodeBase64(rawHmac));

        return oauth;
    }

    public final static String MD5(String pstr) {
        char[] md5String = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a',
'b', 'c', 'd', 'e', 'f'};
        try {
            byte[] btInput = pstr.getBytes();
            MessageDigest mdInst = MessageDigest.getInstance("MD5");
            mdInst.update(btInput);
            byte[] md = mdInst.digest();
            int j = md.length;
            char[] str = new char[j * 2];
            int k = 0;
            for (int i = 0; i < j; i++) { // i = 0
                byte byte0 = md[i]; // 95
                str[k++] = md5String[byte0 >>> 4 & 0xf]; // 5
                str[k++] = md5String[byte0 & 0xf]; // F
            }

            return new String(str);
        } catch (Exception e) {
            return null;
        }
    }
}
```

- 加密规则说明

1.获取baseString，baseString由app_id和当前时间戳ts拼接而成，假如app_id为595f23df，ts为1512041814，则baseString为

> 595f23df1512041814

2.对baseString进行MD5，假如baseString为上一步生成的595f23df1512041814，MD5之后则为

> 0829d4012497c14a30e7e72aeebe565e

3.以app_secret为key对MD5之后的baseString进行HmacSHA1加密，然后再对加密后的字符串进行base64编码。
假如app_secret为d9f4aa7ea6d94faca62cd88a28fd5234，MD5之后的baseString为上一步生成的0829d4012497c14a30e7e72aeebe565e，
则加密之后再进行base64编码得到的signa为

> IrrzsJeOFk1NGfJHW6SkHUoN9CU=

备注：

- app_secret：接口密钥，在应用中添加实时语音转写服务时自动生成，调用方注意保管；
- signa的生成公式：HmacSHA1(MD5(app_id + ts), app_secret)，具体的生成方法参考本git实例代
- 请求示例

```
{'ts': '1688972361', 'appid': 'test1', 'signa': '6b6VF7IR38Yk3DB651juuKM1Stg=',
'audio_url': 'https://zos.abcpen.com/tts/zmeet/20221023/3058bca8-52cb-11ed-961e-00155dc6cbed.mp3', 'audio_encode': 'mpeg2', 'audio_sample_rate': '48000',
'has_participle': 'false'}
```

3、错误码

错误码	描述	说明	处理方式
0	success	成功	
-1	in progress	识别中	请继续重试
-2	audio encode error	音频编码错误	请编码成正确的格式，再提交请求
10105	illegal access	没有权限	检查apiKey，ip，ts等授权参数是否正确
10106	invalid parameter	无效参数	上传必要的参数， 检查参数格式以及编码
10107	illegal parameter	非法参数值	检查参数值是否超过范围或不符合要求
10109	audio url is not valid http(s) url	audio_url不是http[s]链接	长语音识别的时候，audio_url必须是http[s]链接
10110	no license	无授权许可	检查参数值是否超过范围或不符合要求
10700	engine error	引擎错误	提供接口返回值，向服务提供商反馈

错误码	描述	说明	处理方式
10701	Audio encode error, only support pcm, aac, mpeg2, opus and flac	音频编码错误	支持pcm, aac, mpeg2, opus 和 flac这几种编码，请选择其中一种
10702	Audio sample error, only support 8000、16000、44100 and 48000 Hz	音频采样率错误	支持 8000、16000、44100 和 48000 Hz，请选择其中一种
10202	websocket connect error	websocket 连接错误	检查网络是否正常
10204	websocket write error	服务端 websocket 写错误	检查网络是否正常，向服务提供商反馈
10205	websocket read error	服务端 websocket 读错误	检查网络是否正常，向服务提供商反馈
16003	basic component error	基础组件异常	重试或向服务提供商反馈
10800	over max connect limit	超过授权的连接数	确认连接数是否超过授权的连接数