声纹识别

声纹是指人的声音特征，它是人的喉部和嘴唇在说话时产生的声音波形在声音频谱上的表现。每个人的喉部和嘴唇的形状、大小、肌肉运动等都不同，因此人的声音波形也具有独特的特征，就像人的指纹一样，具有独特性和可辨识性。声纹识别技术是利用人的声音特征，通过语音信号的分析和比对，来实现身份验证、辨识、监控等功能的一种生物识别技术。声纹识别技术可以用于语音信号的录制、存储、比对和识别，可以识别人的身份、性别、年龄等信息，具有较高的安全性和可靠性。

一、产品优势

1. 深度学习技术：笔声智能基于深度学习技术，提出了一种新的端到端的声纹识别模型，可以有效提高声纹识别的准确率和稳定性。
2. 鲁棒性强：笔声智能的声纹识别技术具有较高的鲁棒性，可以应对噪声、不同话者、语音质量等因素的影响。
3. 大规模应用：笔声智能的声纹识别技术已经在多个场景得到了广泛应用，例如金融、电信、安防等行业，具备良好的可扩展性和稳定性。
4. 安全可靠：笔声智能的声纹识别技术采用了多种安全机制，例如声纹数据加密、数据隐私保护等，保证了数据的安全性和可靠性。
5. 自主研发：笔声智能在声纹识别领域具有自主研发能力，可以根据客户的需求进行定制化开发，提供个性化的声纹识别解决方案。

综上所述，笔声智能的声纹识别技术具有较高的准确率、鲁棒性和安全性，已经在多个场景得到了广泛应用，并具备良好的可扩展性和稳定性

二、产品功能

1. 声纹识别身份验证：用于身份验证和认证，可以取代传统的密码、PIN码等验证方式，保障信息的安全性。
2. 声纹识别支付：可以实现声纹支付，消费者只需通过语音进行身份验证，即可完成支付。
3. 声纹识别客服：可以将声纹识别技术与客服系统相结合，实现自然语言理解和语音交互，提升客户服务的效率和质量。
4. 声纹识别监控：可以将声纹识别技术应用于监控场景，例如监狱、银行、金融等场所，实现对声音的实时监测和识别，提高安全性和管理效率。
5. 声纹识别辨识：可以将声纹识别技术应用于辨识场景，例如语音信号的录制、分析和比对等，可以用于侦破案件、鉴定录音等。

三、应用场景

1. 身份验证：声纹识别技术可以用于身份验证，取代传统的密码、PIN码等验证方式，保障信息的安全性。
2. 支付认证：声纹识别技术可以用于支付认证，消费者只需通过语音进行身份验证，即可完成支付。
3. 客服服务：声纹识别技术可以将声纹识别技术与客服系统相结合，实现自然语言理解和语音交互，提升客户服务的效率和质量。

4. 监控场景：声纹识别技术可以将声纹识别技术应用于监控场景，例如监狱、银行、金融等场所，实现对声音的实时监测和识别，提高安全性和管理效率。
5. 音频取证：声纹识别技术可以将声纹识别技术应用于音频取证场景，例如侦破案件、鉴定录音等。
6. 个性化服务：声纹识别技术可以用于个性化服务，例如在智能家居、车载系统等场景中，通过声纹识别技术实现个性化的服务和体验。
7. 医疗辅助：声纹识别技术可以用于医疗辅助，例如对于声带受损或无法正常交流的人士，可以通过声纹识别技术实现沟通和交流。

三、API文档

1、接口说明

- 集成同声传译时，需按照以下要求:

内容	说明
传输方式	http[s] (为提高安全性，强烈推荐https)
请求地址前缀	http[s]: //voiceid.abcpen.com/voiceid*注：服务器IP不固定，为保证您的接口稳定，请勿通过指定IP的方式调用接口，使用域名方式调用*
接口鉴权	签名机制，详情请参照下方[鉴权说明]
字符编码	UTF-8
响应格式	统一采用JSON格式
开发语言	任意，只要可以向笔声云服务发起HTTP请求的均可

2. 鉴权说明

在调用业务接口时，请求方需要对请求进行签名，服务端通过签名来校验请求的合法性。
鉴权根据application_id, application_secret, timestamp 和signature这几个参数做组合计算，其中application_id, application_secret请在笔声开放平台自主申请。鉴权如下:

- **注意，目前线上版本鉴权尚没有加入；加入后的验证规则和下面的实现代码没有差别。**

(1). Python示例代码

```
def generate_signature(application_key: str, application_secret: str) -> str:
    timestamp: str = str(int(time.time()))
    message = f"{application_key}{timestamp}"
    signature = hmac.new(application_secret.encode("utf-8"), message.encode("utf-
8"), hashlib.sha256).hexdigest()
    return signature
```

(2) . Java示例代码

```
import java.nio.charset.StandardCharsets;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.time.Instant;

public class SignatureGenerator {
    public static String generateSignature(String applicationKey, String
applicationSecret) {
        String timestamp = String.valueOf(Instant.now().getEpochSecond());
        String message = applicationKey + timestamp;
        try {
            Mac sha256Hmac = Mac.getInstance("HmacSHA256");
            SecretKeySpec secretKey = new
SecretKeySpec(applicationSecret.getBytes(StandardCharsets.UTF_8), "HmacSHA256");
            sha256Hmac.init(secretKey);
            byte[] hmacDigest =
sha256Hmac.doFinal(message.getBytes(StandardCharsets.UTF_8));
            return bytesToHex(hmacDigest);
        } catch (NoSuchAlgorithmException | InvalidKeyException e) {
            e.printStackTrace();
            return null;
        }
    }

    private static String bytesToHex(byte[] bytes) {
        StringBuilder hexString = new StringBuilder();
        for (byte b : bytes) {
            String hex = Integer.toHexString(0xff & b);
            if (hex.length() == 1) {
                hexString.append('0');
            }
            hexString.append(hex);
        }
        return hexString.toString();
    }
}
```

(3). Kotlin示例代码

```kotlin
import java.security.MessageDigest
import javax.crypto.Mac
import javax.crypto.spec.SecretKeySpec
import java.time.Instant

fun generateSignature(applicationKey: String, applicationSecret: String): String {
    val timestamp = Instant.now().epochSecond.toString()
    val message = "$applicationKey$timestamp"
    val hmac = Mac.getInstance("HmacSHA256")
    hmac.init(SecretKeySpec(applicationSecret.toByteArray(Charsets.UTF_8),
"HmacSHA256"))
    val hmacDigest = hmac.doFinal(message.toByteArray(Charsets.UTF_8))
    return bytesToHex(hmacDigest)
}

private fun bytesToHex(bytes: ByteArray): String {
    val hexChars = CharArray(bytes.size * 2)
    for (i in bytes.indices) {
        val v = bytes[i].toInt() and 0xFF
        hexChars[i * 2] = hexArray[v.ushr(4)]
        hexChars[i * 2 + 1] = hexArray[v and 0x0F]
    }
    return String(hexChars)
}

private val hexArray = "0123456789abcdef".toCharArray()
```

(4). nodejs示例代码

```nodejs
const crypto = require('crypto');

function generateSignature(applicationKey, applicationSecret) {
  const timestamp = Math.floor(Date.now() / 1000).toString();
  const message = applicationKey + timestamp;
  const hmac = crypto.createHmac('sha256', applicationSecret);
  hmac.update(message);
  const signature = hmac.digest('hex');
  return signature;
}

// 示例使用
const applicationKey = 'your_application_key';
const applicationSecret = 'your_application_secret';
const signature = generateSignature(applicationKey, applicationSecret);
console.log(signature);
```

(5). go示例代码

```go
package main

import (
    "crypto/hmac"
    "crypto/sha256"
    "encoding/hex"
    "fmt"
    "time"
)

func generateSignature(applicationKey string, applicationSecret string) string {
    timestamp := fmt.Sprintf("%d", time.Now().Unix())
    message := applicationKey + timestamp
    hmacKey := []byte(applicationSecret)
    hmacData := []byte(message)
    hmacSha256 := hmac.New(sha256.New, hmacKey)
    hmacSha256.Write(hmacData)
    signature := hex.EncodeToString(hmacSha256.Sum(nil))
    return signature
}

func main() {
    applicationKey := "your_application_key"
    applicationSecret := "your_application_secret"
    signature := generateSignature(applicationKey, applicationSecret)
    fmt.Println(signature)
}
```

(6) Rust示例代码

```rust
use hmac::{Hmac, Mac, NewMac};
use sha2::Sha256;
use std::time::{SystemTime, UNIX_EPOCH};

fn generate_signature(application_key: &str, application_secret: &str) -> String {
    let timestamp = SystemTime::now()
        .duration_since(UNIX_EPOCH)
        .unwrap()
        .as_secs()
        .to_string();
    let message = format!("{}{}", application_key, timestamp);
    let mut hmac =
        Hmac::<Sha256>::new_varkey(application_secret.as_bytes()).expect("HMAC
initialization failed");
    hmac.update(message.as_bytes());
```

```
    let signature = hex::encode(hmac.finalize().into_bytes());
    signature
}

fn main() {
    let application_key = "your_application_key";
    let application_secret = "your_application_secret";
    let signature = generate_signature(application_key, application_secret);
    println!("{}", signature);
}
```

(7) vue示例代码

```
<template>
  <div>
    <button @click="generateSignature">Generate Signature</button>
    <p>Timestamp: {{ timestamp }}</p>
    <p>Signature: {{ signature }}</p>
  </div>
</template>

<script>
import crypto from 'crypto';

export default {
  name: 'SignatureGenerator',
  data() {
    return {
      applicationKey: 'test1',
      applicationSecret: '2258ACC4-199B-4DCB-B6F3-C2485C63E85A',
      timestamp: null,
      signature: null,
    };
  },
  methods: {
    generateSignature() {
      const timestamp = Math.floor(Date.now() / 1000).toString();
      const message = this.applicationKey + timestamp;
      const hmac = crypto.createHmac('sha256', this.applicationSecret);
      hmac.update(message);
      const signature = hmac.digest('hex');
      this.timestamp = timestamp;
      this.signature = signature;
    },
  },
};
</script>
```

2、鉴权访问

根据上述鉴权说明，生成X-App-Key， X-App-Signature和X-Timestamp这三个参数；将这三个参数放入http(s)请求体头部（header），向服务端发起请求。具体参考后面的示例代码。

注意：

- 下面的API路径中，目前去掉了“/v1”路径；目前去掉了appid，appsecret验证。正式版本这两个都会加上。
- 所有输入参数中，目前有app_id参数，正式版本中都会被appid, appsecret的联合验证方式所替代。
- 别的不会发生变化。

3、接口访问

（1）声纹注册

- 将客户上传的语音片段文件注册到声纹数据库（该机构下的个人ID下）
- POST表单方式提交， multipart/form-data方式
- API路径

/voiceid/register

- 输入参数定义
  - 常规参数，属于鉴权信息，生成X-App-Key， X-App-Signature和X-Timestamp这三个参数
  - 具体参数定义

参数	类型	是否必须	默认值	备注
spk_name	string	是	无	声纹名称
app_id	string	是	无	开发者app_id, 目前任意填入字母数字串；生产环境会被appid/appsecret替换。
org_id	string	是	无	组织或机构Id
tag_id	string	否	“”	个人id
convert_format	Bool	否	"1"	默认转换格式
denoise_audio	Bool	否	"1"	默认对输入语音进行降噪；如果针对非人类发出的语音，如潜艇声纳、某武器声音、某乐器声音等做声纹注册，请设置为“0”

参数	类型	是否必须	默认值	备注
audio	File	是		输入的语音文件，默认支持：wav，AIFF/AIFC， FLAC，Ogg/Vorbis， MP3，Raw这些语音格式；对于默认不支持的语音文件，请打开convert_format（设置为“1”）

- 返回参数定义

参数	类型	备注
data	string	插入的条数
code	string	状态码, 如“0”， 具体参考[错误码]
msg	string	状态码对应字符串，如"success"

如:

```
{"code":"0","msg":"success","data":1}
```

(2) 声纹识别

- 根据客户上传的语音片段，在该机构ID和个人ID下，搜索出该声纹对应的名称
- POST表单方式提交， multipart/form-data方式
- API路径

/voiceid/recognize

- 输入参数定义
  - 常规参数，属于鉴权信息，生成X-App-Key， X-App-Signature和X-Timestamp这三个参数
- 具体参数定义

参数	类型	是否必须	默认值	备注
org_id	string	是	无	组织或机构Id
app_id	string	是	无	开发者app_id, 目前任意填入字母数字串；生产环境会被appid/appsecret替换。

参数	类型	是否必须	默认值	备注
tag_id	string	否	无	个人id
convert_format	Bool	否	"1"	默认转换格式
denoise_audio	Bool	否	"1"	默认对输入语音进行降噪；如果针对非人类发出的语音，如潜艇声纳、某武器声音、某乐器声音等做声纹注册，请设置为“0”
audio	File	是		输入的语音文件，默认支持：wav，AIFF/AIFC，FLAC，Ogg/Vorbis， MP3，Raw这些语音格式；对于默认不支持的语音文件，请打开convert_format（设置为“1”）

- 返回参数定义

参数	类型	备注
data	string	实际返回的具体数据，参考下面的返回实例
code	string	状态码, 如“0”， 具体参考[错误码]
msg	string	状态码对应字符串，如"success"

- 返回结果实例

{"code":"0","msg":"success","data":[{"spk_name":"1bb2af16-337e-46c0-8526-1aa5506eb1e5","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/b902df21-5cb5-4f83-92ef-0b04bbd47640.flac","score":100.0,"tag":"tech12"},{"spk_name":"1b5236ee-23ef-4079-80e0-871d9198ba64","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/883ab2bf-b886-4f0b-9c4b-4ffecf04acb4.flac","score":39.535439014434814,"tag":"tech1"},{"spk_name":"a5c5597e-135c-4384-8726-9f725ab71cb5","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/c5e2c68a-f91a-4646-916a-023e2de547bd.flac","score":-47.278785705566406,"tag":"tech6"},{"spk_name":"5986eba9-48bc-4708-bef0-120a2bf96d8c","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/86444365-6f08-4d6d-b4d3-b1d4a3f73573.flac","score":-71.01327180862427,"tag":"tech32"},{"spk_name":"420fe553-dd6c-4867-8c12-ea493745248d","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/3dd366bb-e469-497d-bfe1-129a945bd35f.flac","score":-73.00647497177124,"tag":"tech21"},{"spk_name":"b3aa3d30-ac96-4b33-ba25-a90c3e7c9197","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/19991dc2-1eba-4954-9e59-81020645705a.flac","score":-81.18548393249512,"tag":"tech23"}]}

(3) 删除声纹

- 删除机构ID下某个个人ID下的某个声纹
- HTTP GET方式提交
- API路径

/voiceid/del-spk-name

- 输入参数定义
  - 常规参数，属于鉴权信息，生成X-App-Key， X-App-Signature和X-Timestamp这三个参数
  - 具体参数定义

参数	类型	是否必须	默认值	备注
spk_name	string	是	无	声纹名称
app_id	string	是	无	开发者app_id, 目前任意填入字母数字串；生产环境会被appid/appsecret替换。
org_id	string	是	无	组织或机构Id

参数	类型	是否必须	默认值	备注
tag_id	string	否	“”	个人id

- 返回参数定义

参数	类型	备注
data	string	实际返回的具体数据，参考下面的返回实例
code	string	状态码, 如“0”， 具体参考[错误码]
msg	string	状态码对应字符串，如"success"

- 返回结果实例

```
{"code":"0","msg":"success","data":0}
```

（4）声纹列表

/voiceid/list

- 常规参数，属于鉴权信息，生成X-App-Key， X-App-Signature和X-Timestamp这三个参数
- HTTP GET方式提交
- 具体参数定义

参数	类型	是否必须	默认值	备注
app_id	string	是	无	开发者app_id, 目前任意填入字母数字串；生产环境会被appid/appsecret替换。
org_id	string	是	无	组织id
tag_id	string	否	“”	个人id
offset	int	否	0	返回数据量大的时候的偏移值
limit	int	否	20	返回数据量打的时候，返回的数量

    - 输入一段文本，识别出这是哪种语言。
- 返回参数定义

参数	类型	备注
data	string	实际返回的具体数据，参考下面的返回实例
code	string	状态码, 如“0”， 具体参考[错误码]
msg	string	状态码对应字符串，如"success"

- 返回结果实例：


```
{"code":"0","msg":"success","data":[{"spk_name":"0f591461-6581-4d07-81a7-
fc92494237f1","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/f6666429
-47eb-46f4-a07d-0b3e40c1538b.flac"},{"spk_name":"39d7c511-ec7d-46d1-b52a-
5d0d1a54d088","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/824e86a8
-66db-44b6-833a-4f646b375b36.flac"},{"spk_name":"08e5f7bd-09d9-4615-9d90-
fe14de259b96","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/c2354856
-2fdf-4398-ae7c-649ad39e093c.flac"},{"spk_name":"68667cbc-d28d-4ac9-81ee-
81cd456574de","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/7ad0a6cc
-a589-4ccb-af30-9636ce63bacd.flac"},{"spk_name":"af6a6712-e318-42da-95f4-
6649d928ac50","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/feacd899
-ef77-4291-8039-20696812341c.flac"},{"spk_name":"ef0a3808-a358-4c79-a09c-
139264cc6d30","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/de406c66
-52d4-4399-b55d-cf9842c68791.flac"},{"spk_name":"9abdbfc3-ab27-4bc3-8a4d-
756fa460f6be","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/fb5358dd
-0624-4877-b1f6-f81d37630852.flac"},{"spk_name":"7a73f2c1-86ca-447d-9859-
1469a5a7f908","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/d6e3c82e
-ff94-4341-8c2d-bc191dddbbe8.flac"},{"spk_name":"174278ca-8d6c-4521-921a-
a3c11b9f972c","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/cdf80d62
-2f96-4e1e-8d01-c5615175ab6c.flac"},{"spk_name":"a5d664bc-e7d4-4775-88d2-
fa92274003a9","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/17fb7be2
-44fe-42b7-8d33-05356f9d87bc.wav"},{"spk_name":"28c1f849-6ade-47be-b314-
c89d2695faba","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/e8c97c23
-e7de-490e-9ad1-dda068d9deb1.flac"},{"spk_name":"17ede8be-3876-491e-9f46-
5a67d7c6a6d8","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/f56dd16e
-685c-4d5a-82c7-867b0f8234ab.wav"},{"spk_name":"753e7cb5-2ba6-4c16-af97-
4debbfb5eae1","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/2c0da03e
-b5df-4282-9bab-b1fd27675f41.flac"},{"spk_name":"3fb70ddb-d870-4ba5-9e03-
0e1c440ceaca","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/30bb80e6
-c9ca-45fc-9ed0-e25bc7cf6bb4.wav"},{"spk_name":"6ff7cf32-a7d9-479b-8974-
06e7abd45fd8","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/4f0fb3fb
-7f43-4c81-93e0-00341914fbc7.flac"},{"spk_name":"4da451e7-02e2-48db-9cf2-
2587ffd809ec","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/7c7b7341
-ab74-48b7-ae13-ea11343f760d.flac"},{"spk_name":"04c30219-6492-4c47-8832-
301be275965b","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/2fea7f8e
-bba4-43e5-8a67-7e26a155a8f3.flac"},{"spk_name":"93fde8f0-8a13-405f-a160-
a1dbff8bf121","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/89bb67f2
-2f73-48d9-800c-b01ab5c038be.flac"},{"spk_name":"80ab5934-fc33-4c3f-8339-
d2ff3f9a732b","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/c72a4767
-f711-4efb-ae52-a38a19f2fe98.flac"},{"spk_name":"830d2adc-489b-411c-a350-
d9c94921d8ba","audio_path":"https://zos.abcpen.com/voiceid/abcpen0/20230419/6a7ecff9
-721c-4474-b735-16cded90e9c6.wav"}]}
```

（5）删除所有声纹

- 删除某个机构ID下的某个个人ID下的所有的声纹
- API路径

/voiceid/delete-all

- HTTP GET 提交请求
- 常规参数，属于鉴权信息，生成X-App-Key， X-App-Signature和X-Timestamp这三个参数
- 具体参数定义

参数	类型	是否必须	默认值	备注
org_id	string	是	无	组织或机构Id
tag_id	string	否	""	个人id
app_id	string	是	无	开发者app_id, 目前任意填入字母数字串；生产环境会被appid/appsecret替换。

- 返回参数定义

参数	类型	备注
data	string	实际删除的数量
code	string	状态码, 如“0”， 具体参考[错误码]
msg	string	状态码对应字符串，如"success"

- 返回结果实例

```
{"code":"0","msg":"success","data":5}
```

4、接口调用实例

（1）、Python实例

```
import requests
import uuid
import random
import os
from pathlib import Path
from time import perf_counter
```

```
audio_part_16k = "./dataset/16k"
audio_part_complex = "./dataset/mp3"
audio_all = "/mnt/h/dataset/test/LibriSpeech/dev-clean"
TEST_COUNT = 100


def voice_register():
    """注册单条语音"""
    files = {'audio': open('./dataset/16k/777-126732-0003.flac', 'rb')}
    # app_id = f'abcpen{random.randrange(5)}'
    app_id = "abcpen0"
    values = {'spk_name': str(uuid.uuid4()),
              "app_id": app_id, "org_id": "abcpen", "tag_id": "tech",
"denoise_audio": "0"}
    url = "https://voiceid.abcpen.com/voiceid/register"
    r = requests.post(url, files=files, data=values)
    print(f"voice register result: {r.text}\n")




def voice_register_complex_all():
    """批量注册语音，支持各种媒体文件，引擎自动转换，但时间会加大；特别注意convert_format赋值为
1"""
    i = 1000
    for elem in Path(audio_part_complex).rglob('*.*'):
        suffix = Path(elem).suffix
        if (suffix == ".flac" or suffix == ".wav" or suffix == ".mp3"):
            print(f"read file: {elem}")
            files = {'audio': open(elem, 'rb')}
            # app_id = f'abcpen{random.randrange(5)}'
            app_id = "abcpen0"
            values = {'spk_name': str(uuid.uuid4()),
                      "app_id": app_id, "org_id": "abcpen", "tag_id": "tech"+i,
"convert_format": 1, "denoise_audio": "0"}
            url = "https://voiceid.abcpen.com/voiceid/register"
            t1 = perf_counter()
            r = requests.post(url, files=files, data=values)
            print(
                f"voice register all result: {r.text}, time: {perf_counter() -
t1}s\n")
            i = i+1
            if (i > TEST_COUNT+1000):
                break

def voice_search():
    """单条声纹搜索"""
    files = {'audio': open('./dataset/16k/777-126732-0003.flac', 'rb')}
    values = {"app_id": "abcpen0", "org_id": "abcpen", "convert_format": "0",
"denoise_audio": "0"}
```

```
    url = "https://voiceid.abcpen.com/voiceid/recognize"
    r = requests.post(url, files=files, data=values)
    print(f"voice search result: {r.text}\n")




def voice_search_complex_full():
    """批量声纹搜索，支持各种媒体文件，注意convert_format赋值为1"""
    i = 0
    for elem in Path(audio_part_complex).rglob('*.*'):
        suffix = Path(elem).suffix
        if (suffix == ".flac" or suffix == ".wav" or suffix == ".mp3"):
            print(f"read file: {elem}")
            files = {'audio': open(elem, 'rb')}
            # app_id = f'abcpen{random.randrange(5)}'
            app_id = "abcpen0"
            values = {"app_id": app_id, "convert_format": 1, "org_id": "abcpen",
"convert_format": "0", "denoise_audio": "0"}
            t1 = perf_counter()
            url = "https://voiceid.abcpen.com/voiceid/recognize"
            r = requests.post(url, files=files, data=values)
            print(
                f"voice search result: {r.text}, time: {perf_counter() - t1}s\n")
            i = i+1
            if (i > TEST_COUNT):
                break

def voice_list():
    """声纹列表，支持limit和offset，也即翻页和返回单次调用限量"""
    values = random.randrange(10)
    url = f"https://voiceid.abcpen.com/voiceid/list"
    para = {"app_id": "abcpen0", "org_id": "abcpen"}
    r = requests.get(url, para)
    print(f"voice list result : {r.text}\n")

    url = f"https://voiceid.abcpen.com/voiceid/list"
    para = {"app_id": "abcpen0",  "org_id": "abcpen", "limit": 10, "offset": 10}
    r = requests.get(url, para)
    print(f"voice list with limit and offset result: {r.text}\n")

    url = f"https://voiceid.abcpen.com/voiceid/list"
    para = {"app_id": "abcpen0",  "org_id": "abcpen", "limit": 10}
    r = requests.get(url, para)
    print(f"voice list with limit result : {r.text}\n")

    url = f"https://voiceid.abcpen.com/voiceid/list"
    para = {"app_id": "abcpen0", "org_id": "abcpen","offset": 10}
    r = requests.get(url, para)
    print(f"voice list with offset result : {r.text}\n")
```

```python
def voice_count():
    """该机构下的声纹数量"""
    values = random.randrange(10)
    para = {"app_id": "abcpen0", "org_id": "abcpen"}
    url = f"https://voiceid.abcpen.com/voiceid/count"
    r = requests.get(url, para)
    print(f"voice count result: {r.text}\n")


def voice_url():
    """返回某个speaker name注册过的原始媒体文件url"""
    values = random.randrange(10)
    para = {"app_id": "abcpen0",
            "spk_name": '03b8accb-3ee8-4bb9-a15e-cf7ed895d3c2', "org_id": "abcpen"}
    url = f"https://voiceid.abcpen.com/voiceid/voice-url"
    r = requests.get(url, para)
    print(f"voice url result: {r.text}\n")


def voice_del():
    """删除该机构下某个声纹"""
    values = random.randrange(10)
    para = {"app_id": "abcpen0","tag_id": "abcpen0", "org_id": "abcpen",
            "spk_name": '03b8accb-3ee8-4bb9-a15e-cf7ed895d3c2'}
    url = f"https://voiceid.abcpen.com/voiceid/del-spk-name"
    r = requests.get(url, para)
    print(f"voice del result: {r.text}\n")


def voice_del_all():
    """删除该机构下所有的声纹"""
    values = random.randrange(10)
    para = {"app_id": "abcpen2", "tag_id": "abcpen2", "org_id": "abcpen"}
    url = f"https://voiceid.abcpen.com/voiceid/delete-all"
    r = requests.get(url, para)
    print(f"voice delete all result: {r.text}\n")


if __name__ == "__main__":
    try:
        for i in range(1):
            voice_register()
            voice_register_complex_all()
            voice_count()
            voice_list()
            voice_search()

            voice_search_complex_full()
            voice_url()
            voice_del()
            voice_del_all()
```

```
    except Exception as err:
        print(f"meet exception {err}")
```

(2) 、Java实例

```
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashMap;
import java.util.Map;
import java.util.UUID;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.ContentType;
import org.apache.http.entity.mime.MultipartEntityBuilder;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

public class VoiceAPI {
private static final String AUDIO_PART_16K = "./dataset/16k";
private static final String AUDIO_PART_COMPLEX = "./dataset/mp3";
private static final String AUDIO_ALL = "/mnt/h/dataset/test/LibriSpeech/dev-clean";
private static final int TEST_COUNT = 100;
public static void main(String[] args) {
    try {
        for (int i = 0; i < 1; i++) {
            voiceRegister();
            voiceRegisterComplexAll();
            voiceCount();
            voiceList();
            voiceSearch();

            voiceSearchComplexFull();
            voiceUrl();
            voiceDel();
            voiceDelAll();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```
public static void voiceRegister() throws IOException {
    String url = "https://voiceid.abcpen.com/voiceid/register";
    Path path = Paths.get(AUDIO_PART_16K, "777-126732-0003.flac");
    File file = path.toFile();
    String app_id = "abcpen0";
    String spk_name = UUID.randomUUID().toString();
    String org_id = "abcpen";
    String tag_id = "tech";
    String denoise_audio = "0";
    HttpPost httpPost = new HttpPost(url);
    MultipartEntityBuilder builder = MultipartEntityBuilder.create();
    builder.addBinaryBody("audio", file, ContentType.DEFAULT_BINARY,
file.getName());
    builder.addTextBody("app_id", app_id);
    builder.addTextBody("spk_name", spk_name);
    builder.addTextBody("org_id", org_id);
    builder.addTextBody("tag_id", tag_id);
    builder.addTextBody("denoise_audio", denoise_audio);
    HttpEntity multipart = builder.build();
    httpPost.setEntity(multipart);
    try (CloseableHttpClient httpClient = HttpClients.createDefault();
            CloseableHttpResponse response = httpClient.execute(httpPost)) {
        System.out.println("voice register result: " +
EntityUtils.toString(response.getEntity()));
    }
}

public static void voiceRegisterComplexAll() throws IOException {
    String url = "https://voiceid.abcpen.com/voiceid/register";
    int i = 1000;
    for (File file : new File(AUDIO_PART_COMPLEX).listFiles()) {
        String suffix = file.getName().substring(file.getName().lastIndexOf("."));
        if (suffix.equals(".flac") || suffix.equals(".wav") ||
suffix.equals(".mp3")) {
            System.out.println("read file: " + file.getName());
            String app_id = "abcpen0";
            String spk_name = UUID.randomUUID().toString();
            String org_id = "abcpen";
            String tag_id = "tech" + i;
            String convert_format = "1";
            String denoise_audio = "0";
            HttpPost httpPost = new HttpPost(url);
            MultipartEntityBuilder builder = MultipartEntityBuilder.create();
            builder.addBinaryBody("audio", file,
            ContentType.DEFAULT_BINARY, file.getName());
            builder.addTextBody("app_id", app_id);
            builder.addTextBody("spk_name", spk_name);
            builder.addTextBody("org_id", org_id);
            builder.addTextBody("tag_id", tag_id);
            builder.addTextBody("convert_format", convert_format);
            builder.addTextBody("denoise_audio", denoise_audio);
```

```
            HttpEntity multipart = builder.build();
            httpPost.setEntity(multipart);
            try (CloseableHttpClient httpClient = HttpClients.createDefault();
            CloseableHttpResponse response = httpClient.execute(httpPost)) {
            System.out.println("voice register all result: " +
EntityUtils.toString(response.getEntity()) + ", time: " +
(System.currentTimeMillis() / 1000));
            }
            i++;
            if (i > TEST_COUNT + 1000) {
            break;
            }
            }
            }
            }
    public static void voiceSearch() throws IOException {
    String url = "https://voiceid.abcpen.com/voiceid/recognize";
    Path path = Paths.get(AUDIO_PART_16K, "777-126732-0003.flac");
    File file = path.toFile();
    String app_id = "abcpen0";
    String org_id = "abcpen";
    String convert_format = "0";
    String denoise_audio = "0";
    HttpPost httpPost = new HttpPost(url);
    MultipartEntityBuilder builder = MultipartEntityBuilder.create();
    builder.addBinaryBody("audio", file, ContentType.DEFAULT_BINARY,
file.getName());
    builder.addTextBody("app_id", app_id);
    builder.addTextBody("org_id", org_id);
    builder.addTextBody("convert_format", convert_format);
    builder.addTextBody("denoise_audio", denoise_audio);
    HttpEntity multipart = builder.build();
    httpPost.setEntity(multipart);
    try (CloseableHttpClient httpClient = HttpClients.createDefault();
            CloseableHttpResponse response = httpClient.execute(httpPost)) {
        System.out.println("voice search result: " +
EntityUtils.toString(response.getEntity()));
    }
}

public static void voiceSearchComplexFull() throws IOException {
    String url = "https://voiceid.abcpen.com/voiceid/recognize";
    int i = 0;
    for (File file : new File(AUDIO_PART_COMPLEX).listFiles()) {
        String suffix = file.getName().substring(file.getName().lastIndexOf("."));
        if (suffix.equals(".flac") || suffix.equals(".wav") ||
suffix.equals(".mp3")) {
            System.out.println("read file: " + file.getName());
            String app_id = "abcpen0";
            String convert_format = "1";
            String org_id = "abcpen";
            String denoise_audio = "0";
```

```
            HttpPost httpPost = new HttpPost(url);
            MultipartEntityBuilder builder = MultipartEntityBuilder.create();
            builder.addBinaryBody("audio", file, ContentType.DEFAULT_BINARY,
file.getName());
            builder.addTextBody("app_id", app_id);
            builder.addTextBody("convert_format", convert_format);
            builder.addTextBody("org_id", org_id);
            builder.addTextBody("denoise_audio", denoise_audio);
            HttpEntity multipart = builder.build();
            httpPost.setEntity(multipart);
            try (CloseableHttpClient httpClient = HttpClients.createDefault();
                    CloseableHttpResponse response = httpClient.execute(httpPost)) {
                System.out.println("voice search result: " +
EntityUtils.toString(response.getEntity()) + ", time: " +
(System.currentTimeMillis() / 1000));
            }
            i++;
            if (i > TEST_COUNT) {
                break;
            }
        }
    }
public static void voiceList() throws IOException {
    String url = "https://voiceid.abcpen.com/voiceid/list";
    Random random = new Random();
    int values = random.nextInt(10);
    Map<String, String> para = new HashMap<>();
    para.put("app_id", "abcpen0");
    para.put("org_id", "abcpen");
    try (CloseableHttpClient httpClient = HttpClients.createDefault()) {
        HttpGet httpGet = new HttpGet(url);
        httpGet.setHeader("Content-Type", "application/json");
        httpGet.setHeader("Accept", "application/json");
        httpGet.setURI(URI.create(url + "?" +
URLEncodedUtils.format(para.entrySet().stream()
                .map(e -> new BasicNameValuePair(e.getKey(),
e.getValue())).collect(Collectors.toList()), StandardCharsets.UTF_8)));
        try (CloseableHttpResponse response = httpClient.execute(httpGet)) {
            System.out.println("voice list result : " +
EntityUtils.toString(response.getEntity()));
        }

        para.put("limit", "10");
        para.put("offset", "10");
        httpGet = new HttpGet(url);
        httpGet.setHeader("Content-Type", "application/json");
        httpGet.setHeader("Accept", "application/json");
        httpGet.setURI(URI.create(url + "?" +
URLEncodedUtils.format(para.entrySet().stream()
                .map(e -> new BasicNameValuePair(e.getKey(),
e.getValue())).collect(Collectors.toList()), StandardCharsets.UTF_8)));
        try (CloseableHttpResponse response = httpClient.execute(httpGet)) {
```

```
            System.out.println("voice list with limit and offset result: " +
EntityUtils.toString(response.getEntity()));
        }

        para.put("limit", "10");
        para.remove("offset");
        httpGet = new HttpGet(url);
        httpGet.setHeader("Content-Type", "application/json");
        httpGet.setHeader("Accept", "application/json");
        httpGet.setURI(URI.create(url + "?" +
URLEncodedUtils.format(para.entrySet().stream()
                .map(e -> new BasicNameValuePair(e.getKey(),
e.getValue())).collect(Collectors.toList()), StandardCharsets.UTF_8)));
        try (CloseableHttpResponse response = httpClient.execute(httpGet)) {
            System.out.println("voice list with limit result : " +
EntityUtils.toString(response.getEntity()));
        }

        para.remove("limit");
        para.put("offset", "10");
        httpGet = new HttpGet(url);
        httpGet.setHeader("Content-Type", "application/json");
        httpGet.setHeader("Accept", "application/json");
        httpGet.setURI(URI.create(url + "?" +
URLEncodedUtils.format(para.entrySet().stream()
                .map(e -> new BasicNameValuePair(e.getKey(),
e.getValue())).collect(Collectors.toList()), StandardCharsets.UTF_8)));
        try (CloseableHttpResponse response = httpClient.execute(httpGet)) {
            System.out.println("voice list with offset result : " +
EntityUtils.toString(response.getEntity()));
        }
    }
}

public static void voiceCount() throws IOException {
    String url = "https://voiceid.abcpen.com/voiceid/count";
    Random random = new Random();
    int values = random.nextInt(10);
    Map<String, String> para = new HashMap<>();
    para.put("app_id", "abcpen0");
    para.put("org_id", "abcpen");
    try (CloseableHttpClient httpClient = HttpClients.createDefault()) {
        HttpGet httpGet = new HttpGet(url);
        httpGet.setHeader("Content-Type", "application/json");
        httpGet.setHeader("Accept", "application/json");
        httpGet.setURI(URI.create(url + "?" +
URLEncodedUtils.format(para.entrySet().stream()
                .map(e -> new BasicNameValuePair(e.getKey(),
e.getValue())).collect(Collectors
        para.put("app_id", "abcpen0");
        para.put("org_id", "abcpen");
        para.put("spk_name", "03b8accb-3ee8-4bb9-a15e-cf7ed895d3c2");
```

```
        httpGet = new HttpGet(url);
        httpGet.setHeader("Content-Type", "application/json");
        httpGet.setHeader("Accept", "application/json");
        httpGet.setURI(URI.create(url + "?" +
URLEncodedUtils.format(para.entrySet().stream()
                .map(e -> new BasicNameValuePair(e.getKey(),
e.getValue())).collect(Collectors.toList()), StandardCharsets.UTF_8)));
        try (CloseableHttpResponse response = httpClient.execute(httpGet)) {
            System.out.println("voice url result: " +
EntityUtils.toString(response.getEntity()));
        }
    }
}

public static void voiceDel() throws IOException {
    String url = "https://voiceid.abcpen.com/voiceid/del-spk-name";
    Random random = new Random();
    int values = random.nextInt(10);
    Map<String, String> para = new HashMap<>();
    para.put("app_id", "abcpen0");
    para.put("org_id", "abcpen");
    para.put("tag_id", "abcpen0");
    para.put("spk_name", "03b8accb-3ee8-4bb9-a15e-cf7ed895d3c2");
    try (CloseableHttpClient httpClient = HttpClients.createDefault()) {
        HttpGet httpGet = new HttpGet(url);
        httpGet.setHeader("Content-Type", "application/json");
        httpGet.setHeader("Accept", "application/json");
        httpGet.setURI(URI.create(url + "?" +
URLEncodedUtils.format(para.entrySet().stream()
                .map(e -> new BasicNameValuePair(e.getKey(),
e.getValue())).collect(Collectors.toList()), StandardCharsets.UTF_8)));
        try (CloseableHttpResponse response = httpClient.execute(httpGet)) {
            System.out.println("voice del result: " +
EntityUtils.toString(response.getEntity()));
        }
    }
}

public static void voiceDelAll() throws IOException {
    String url = "https://voiceid.abcpen.com/voiceid/delete-all";
    Random random = new Random();
    int values = random.nextInt(10);
    Map<String, String> para = new HashMap<>();
    para.put("app_id", "abcpen2");
    para.put("org_id", "abcpen");
    para.put("tag_id", "abcpen2");
    try (CloseableHttpClient httpClient = HttpClients.createDefault()) {
        HttpGet httpGet = new HttpGet(url);
        httpGet.setHeader("Content-Type", "application/json");
        httpGet.setHeader("Accept", "application/json");
        httpGet.setURI(URI.create(url + "?" +
URLEncodedUtils.format(para.entrySet().stream()
```

```
                .map(e -> new BasicNameValuePair(e.getKey(),
e.getValue())).collect(Collectors.toList()), StandardCharsets.UTF_8)));
        try (CloseableHttpResponse response = httpClient.execute(httpGet)) {
            System.out.println("voice delete all result: " +
EntityUtils.toString(response.getEntity()));
        }
    }
}

public static void main(String[] args) throws IOException {
    for (int i = 0; i < 1; i++) {
        voiceRegister();
        voiceRegisterComplexAll();
        voiceCount();
        voiceList();
        voiceSearch();
        voiceSearchComplexFull();
        voiceUrl();
        voiceDel();
        voiceDelAll();
    }
}

}

   }
}

public static void voiceList() throws IOException {
    String url = "https://voiceid.abcpen.com/voiceid/list";
    Random random = new Random();
    int values = random.nextInt(10);
    Map<String, String> para = new HashMap<>();
    para.put("app_id", "abcpen0");
    para.put("org_id", "abcpen");

    HttpGet httpGet = new HttpGet(url);
    httpGet.setHeader("Content-Type", "application/json");
    httpGet.setHeader("Accept", "application/json");
    httpGet.setURI(URI.create(url + "?" +
URLEncodedUtils.format(para.entrySet().stream()
            .map(e -> new BasicNameValuePair(e.getKey(),
e.getValue())).collect(Collectors.toList()), StandardCharsets.UTF_8)));
    try (CloseableHttpClient httpClient = HttpClients.createDefault();
CloseableHttpResponse response = httpClient.execute(httpGet)) {
        System.out.println("voice list result : " +
EntityUtils.toString(response.getEntity()));
    }

    para.put("limit", "10");
    para.put("offset", "10");
    httpGet = new HttpGet(url);
```

```
    httpGet.setHeader("Content-Type", "application/json");
    httpGet.setHeader("Accept", "application/json");
    httpGet.setURI(URI.create(url + "?" +
URLEncodedUtils.format(para.entrySet().stream()
            .map(e -> new BasicNameValuePair(e.getKey(),
e.getValue())).collect(Collectors.toList()), StandardCharsets.UTF_8)));
    try (CloseableHttpClient httpClient = HttpClients.createDefault();
CloseableHttpResponse response = httpClient.execute(httpGet)) {
        System.out.println("voice list with limit and offset result: " +
EntityUtils.toString(response.getEntity()));
    }

    para.put("limit", "10");
    para.remove("offset");
    httpGet = new HttpGet(url);
    httpGet.setHeader("Content-Type", "application/json");
    httpGet.setHeader("Accept", "application/json");
    httpGet.setURI(URI.create(url + "?" +
URLEncodedUtils.format(para.entrySet().stream()
            .map(e -> new BasicNameValuePair(e.getKey(),
e.getValue())).collect(Collectors.toList()), StandardCharsets.UTF_8)));
    try (CloseableHttpClient httpClient = HttpClients.createDefault();
CloseableHttpResponse response = httpClient.execute(httpGet)) {
        System.out.println("voice list with limit result : " +
EntityUtils.toString(response.getEntity()));
    }

    para.put("offset", "10");
    para.remove("limit");
    httpGet = new HttpGet(url);
    httpGet.setHeader("Content-Type", "application/json");
    httpGet.setHeader("Accept", "application/json");
    httpGet.setURI(URI.create(url + "?" +
URLEncodedUtils.format(para.entrySet().stream()
            .map(e -> new BasicNameValuePair(e.getKey(),
e.getValue())).collect(Collectors.toList()), StandardCharsets.UTF_8)));
    try (CloseableHttpClient httpClient = HttpClients.createDefault();
CloseableHttpResponse response = httpClient.execute(httpGet)) {
        System.out.println("voice list with offset result : " +
EntityUtils.toString(response.getEntity()));
    }
}

}
```

5. 错误码

错误码	描述	说明	处理方式

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
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
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

四、价格套餐

待定