

---

---

# 文本无关短语音声纹云 HTTP 协议接口文档

---

---



二零二四年四月

## 目录

目录.....	2
一.文档概述.....	3
二.系统架构.....	4
三.声纹识别原理.....	5
四.接口调用流程.....	6
4.1、声纹登记流程.....	6
4.2、声纹认证流程.....	7
五.协议具体交互描述.....	8
5.1、注册用户.....	8
5.2、删除用户.....	8
5.3、添加语音.....	9
5.4、检测用户.....	12
5.5、检测模型.....	12
5.6、训练模型.....	13
5.7、声纹确认.....	13
5.8、声纹辨认.....	14
六.返回值说明.....	16
6.1、ret 返回值.....	16
6.2、errCode 返回值.....	16
七.CURL 使用例子.....	17
八.JAVA DEMO 例子.....	18
八.技术支持.....	23

# 一.文档概述

**文档简介：**天聪文本无关短语音声纹识别通过 REST API 的方式给开发者提供一个通用的 HTTP 接口。

**支持语种：**普通话。

**适用范围：**任意操作系统，任意编程语言，只要能对声纹识别服务器发起 http 请求，均可使用本接口。

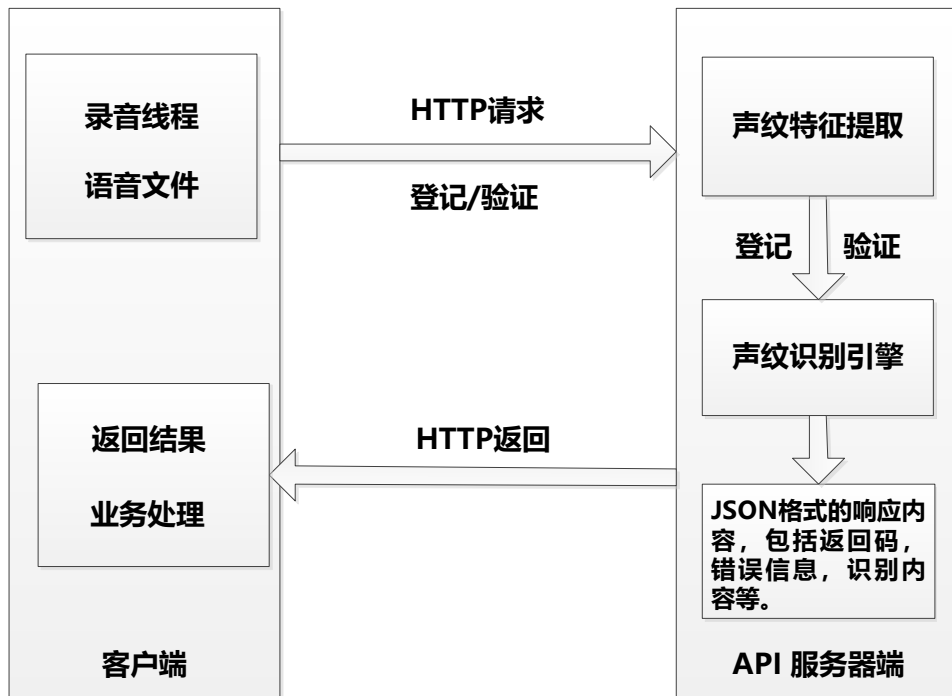
**语音格式：**wav（不压缩，pcm 编码）格式，采样率 16000，16bit 采样精度的单声道语音。

**语音要求：**登记有效语音 20s 以上，验证有效语音 5s 以上。

**私有化部署：**系统支持 windows 64 位或者 Linux centos 6.X 64 位操作系统。支持私有云集群化部署。

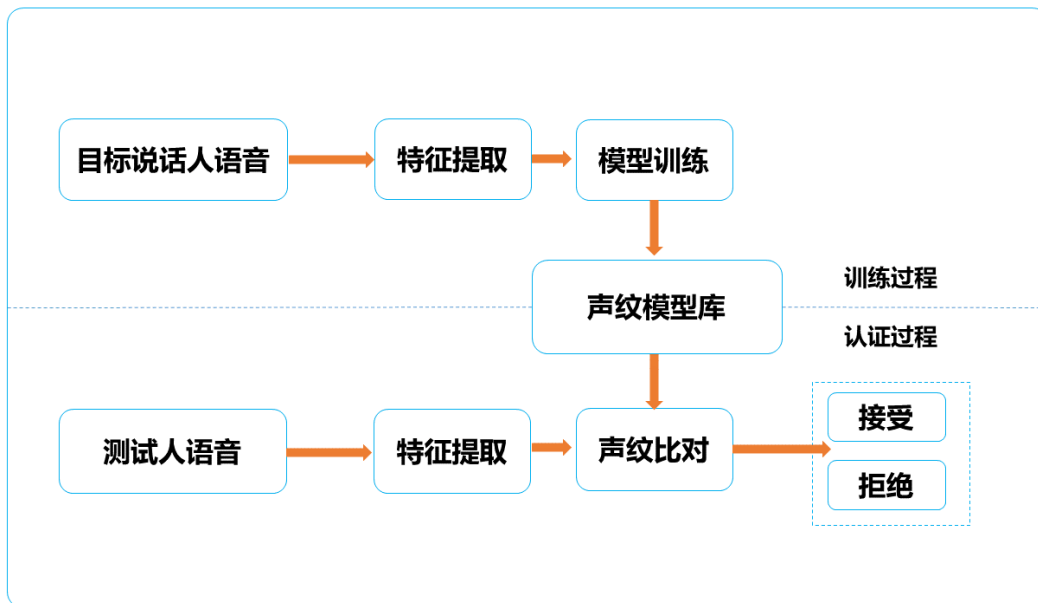
**版本号：V4.0**

## 二.系统架构



**说明:**系统通过 http 的协议来进行调用。客户端采用 http post,发 post 请求到服务器,然后获取服务器的响应, 根据响应的代码, 判断操作是否成功。客户端负责语音的采集, 并将采集后的语音上传到服务端, 由服务端进行语音处理, 包括声纹模型的建立以及声纹比对等, 并将结果返回到客户端。

### 三.声纹识别原理



文本无关短语音声纹识别，允许用户随便说，登记语音和验证语音内容可不一致，但认证有效时长要求 3 秒以上，文本无关声纹模型建立相对困难，但用户使用方便，可应用范围较广，适用于用户不配合或不知情场合。

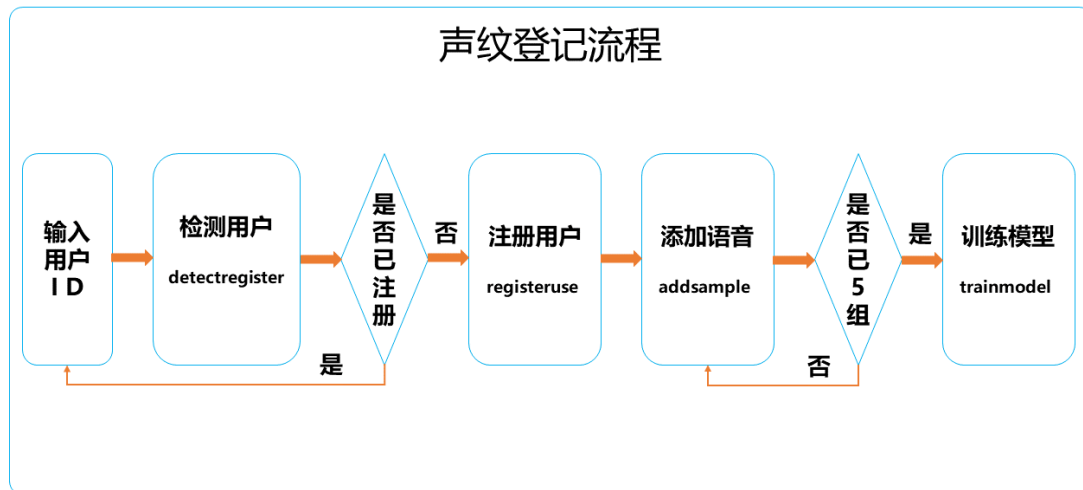
声纹识别包含声纹模型登记和声纹认证。文本无关声纹模型登记，需要采集目标说话人 20s 以上的有效语音。声纹认证的时候需要采集 3s 以上的有效语音。采集过程基于平时说话的语速和声音进行。

在登记模型的时候，系统会对采集到的语音进行分析，提取其声纹特征，建立模型，并保存到声纹模型库中。在认证的时候，同样对采集到的语音进行分析提取特征，然后与模型库中的声纹模型进行比对，依据一定的判决条件给出最后的识别结果。

支持 1:1 的声纹确认和 1:N 的声纹辨认。

## 四.接口调用流程

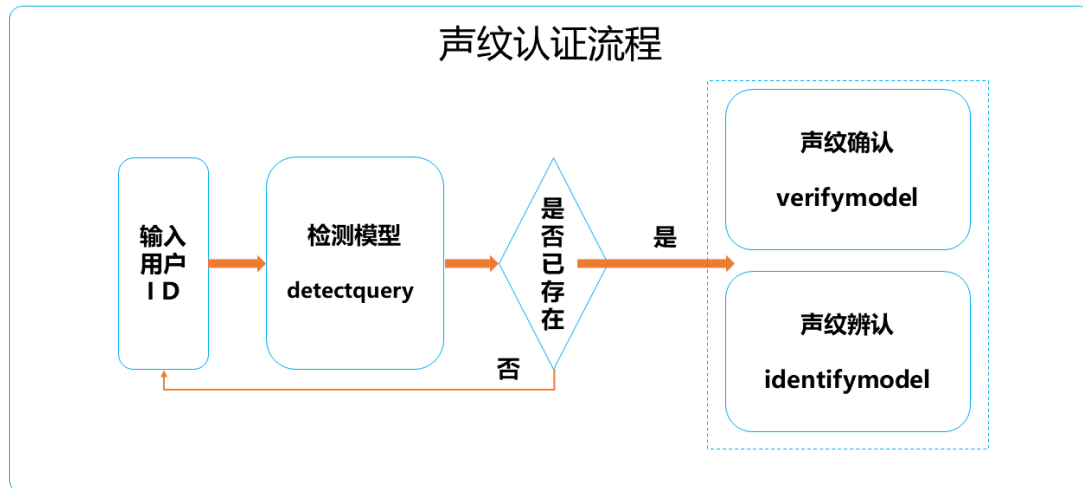
### 4.1、声纹登记流程



登记流程如上图所示，具体流程如下：

- (1)、输入用户 ID（即用户名），用户名支持数字和字母，暂不支持中文汉字。
- (2)、检测用户 ID，调用检测用户 ID 接口（**detectregister**）来检测当前输入的用户 ID 是否合法（符合命名规则）及是否已经注册过。
- (3)、注册用户，在步骤(2)中检测用户 ID 合法且未注册后，便可对当前的用户 ID 进行注册，注册用户调用接口（**registeruse**）。
- (4)、添加语音，对当前新注册的用户进行添加语音，文本无关需要添加一条有效语音达到 20s 以上的完整语音，添加语音调用接口（**addsample**）。
- (5)、训练模型，当添加的语音达到要求之后，即可进行模型训练，训练的模型是以用户 ID 名进行命名的，训练完成之后保存到对应的模型数据库中，训练模型调用接口（**trainmodel**）。

## 4.2、声纹认证流程



声纹认证分为 1:1 的声纹确认和 1:N 的声纹辨认。流程如上图所示，具体步骤如下：

- (1)、输入用户 ID（即用户名），输入已登记声纹模型的用户 ID。
- (2)、检测模型，对输入的用户 ID 进行检测，检测当前用户 ID 是否已存在，并且已完成声纹模型的登记，检测模型调用接口（`detectquery`）。
- (3)、声纹确认，即 1:1 声纹认证。测试语音与当前用户 ID 的模型进行比对，并给出并对得分，声纹确认调用接口（`verifymodel`）。
- (4)、声纹辨认，即 1:N 声纹认证。测试语音与指定的模型集合进行一一比对，给出得分最高的用户模型 ID，声纹辨认调用接口（`identifymodel`）。
- (5)、在声纹认证中，输出的结果为声纹认证的得分。业务层上需要对认证的得分进行判断。即，声纹认证得分要大于某个阈值（阈值可根据实际应用场景进行设置）才可认为认证成功，否则当判断为认证失败。

## 五.协议具体交互描述

说明：需申请 userid 和 token, 然后可以使用.

服务器地址:SERVER\_ADDR = " http://124.220.2.168:9391"

### 5.1、注册用户

使用 POST 方法

注册 URL/registeruser

参数:userid token

参数说明:

userid:用户名称,可以是字母或者数字

token:公司名称 id

整个 URL:

`http://124.220.2.168:9391/registeruser?userid=xxxx&token=xxxx`

服务器返回:

"1001" 注册用户成功,

"100" 注册用户失败,

"2001" 用户已经存在 (errCode) ,

备注: 只有注册用户成功之后才能进行上传语音, 训练模型, 验证声纹等一系列操作。

Java 例子: `java registeruser 1001`

### 5.2、删除用户

使用 POST 方法

注册 URL/deleteuser

参数:userid token

参数说明:



userid: 用户名称

token: 公司名称 id

整个 URL:

`http://124.220.2.168:9391/deleteuser?userid=xxxx&token=xxxx`

服务器返回:

"1002" 删除用户成功,

"200" 删除用户失败,

"2002" 用户不存在 (errCode),

Java 例子: `java deleteuser 1001`

### 5.3、添加语音

使用 POST 方法

注册 URL /addsample

整个 URL:

`http://124.220.2.168:9391/addsample`

传送字节流

必选字段:

userid: 用户名称

token: 公司名称 id

step: 当前的训练步骤, 对应第 step 录音,  $step \leq 5$

content: 上传的语音内容, 系统固定 id="123456"

file: 文件标识

语音缓冲 (可以合并传, 也可分段传, 但不能有间隔符):

buffer1(录音缓冲区) +

buffer2(录音缓冲区) +

buffer3(录音缓冲区) +

.

.

bufferN(录音缓冲区)

完整字节流(具体参看例子):

```
TWO_HYPHENS + BOUNDARY + END + Content-Disposition: form-data;
name=\"userid\" + END + \"Content-Type: text/plain; charset=\" + CHARSET
+ END + \"Content-Transfer-Encoding: 8bit\" + END + END + userid + END +
TWO_HYPHENS + BOUNDARY + END + Content-Disposition: form-data;
name=\"token\" + END + \"Content-Type: text/plain; charset=\" + CHARSET +
END + \"Content-Transfer-Encoding: 8bit\" + END + END + token + END +
TWO_HYPHENS + BOUNDARY + END + Content-Disposition: form-data;
name=\"content\" + END + \"Content-Type: text/plain; charset=\" + CHARSET
+ END + \"Content-Transfer-Encoding: 8bit\" + END + END + content + END +
TWO_HYPHENS + BOUNDARY + END + Content-Disposition: form-data;
name=\"step\" + END + \"Content-Type: text/plain; charset=\" + CHARSET +
END + \"Content-Transfer-Encoding: 8bit\" + END + END + step + END +
TWO_HYPHENS + BOUNDARY + END + Content-Disposition: form-data;
name=\"sampleRate\" + END + \"Content-Type: text/plain; charset=\" +
CHARSET + END + \"Content-Transfer-Encoding: 8bit\" + END + END + sampleRate
+ END +
TWO_HYPHENS + BOUNDARY + END + Content-Disposition: form-data;
name=\"bits\" + END + \"Content-Type: text/plain; charset=\" + CHARSET +
END + \"Content-Transfer-Encoding: 8bit\" + END + END + bits + END +
TWO_HYPHENS + BOUNDARY + END + Content-Disposition: form-data;
name=\"channels\" + END + \"Content-Type: text/plain; charset=\" + CHARSET
+ END + \"Content-Transfer-Encoding: 8bit\" + END + END + channels + END
+
TWO_HYPHENS + BOUNDARY + END + \"Content-Disposition: form-data;
name=\"file\"; filename=\"\" + \"upload.wav\" + \"\" + END + END +
```

```
buffer1(录音缓冲区) +  
buffer2(录音缓冲区) +  
buffer3(录音缓冲区) +  
. . .  
bufferN(录音缓冲区) +  
  
END +  
TWO_HYPHENS + BOUNDARY + TWO_HYPHENS + END
```

其中

```
String END = "\r\n";  
String CHARSET = "UTF-8";  
String TWO_HYPHENS = "--";  
String BOUNDARY = "*****";
```

服务器返回:

```
"1006" 上传语音缓冲成功  
"600" 上传语音缓冲失败,  
"1" 第 1 遍录音(step)  
"2" 第 2 遍录音(step)  
"3" 第 3 遍录音(step)  
"4" 第 4 遍录音(step)  
"5" 第 5 遍录音(step)
```

Java 例子: `java addsample 1001 1 123456 wav/upload.wav`

## 5.4、检测用户

检查用户 ID 是否可以使用

使用 POST 方法

注册 URL/detectregister

参数:userid token

参数说明:

userid:用户名称

token:公司名称 id

整个 URL:

<http://124.220.2.168:9391/detectregister?userid=xxxx&token=xxxx>

服务器返回:

"1010" 用户尚未注册, 登记检测通过

"800" 用户已存在

Java例子: `java detectregister 1001`

## 5.5、检测模型

检测用户: 检测该用户是否存在, 并且已经训练模型

使用 POST 方法

注册 URL/ detectquery

参数:userid token

参数说明:

userid: 用户名称

token:公司名称 id

appkey系统分配="xxxx"

整个URL:

<http://124.220.2.168:9391/detectquery?userid=xxxx&token=xxxx>

服务器返回:

“1011” 符合验证声纹的前提，  
“900” 不符合验证声纹的前提，  
“2005” 模型不存在（errCode），  
java 例子：java detectquery 1001

## 5.6、训练模型

使用 POST 方法

注册 URL/trainmodel

参数:useridtoken

参数说明：

userid:用户名称

token:公司名称 id

整个 URL：

<http://124.220.2.168:9391/trainmodel?userid=xxxx&token=xxxx>

服务器返回：

“1004” 训练模型成功，

“400” 训练模型失败，

“2004” 模型已经存在，

“2008” 训练模型失败，

备注：

只有上传一条有效语音达到 20s 以上的录音文件，才能训练模型。

Java 例子：java trainmodel 1001

## 5.7、声纹确认

声纹确认即 1:1 的声纹认证，验证模型并返回识别得分

使用 POST 方法

注册 URL /verifymodel

整个 URL:

<http://124.220.2.168:9391/verifymodel>

传送字节流

必选字段:

userid: 用户名称

token: 公司名称 id

content: 上传的语音内容, 系统固定 id="123456"

file: 文件标识

语音缓冲 (可以合并传, 也可分段传, 但不能有间隔符):

buffer1 (录音缓冲区) +

buffer2 (录音缓冲区) +

buffer3 (录音缓冲区) +

.

.

.

bufferN (录音缓冲区)

服务器返回:

"1007" 验证完成

"700" 验证失败

"900" 验证检测失败

"2005" 模型不存在

"2012" 用户名不合法

Java 例子: `java verifymodel 1001 123456 wav/upload.wav`

## 5.8、声纹辨认

声纹辨认即 1:N 声纹识别, 从一组里面识别出用户模型并返回识别得分

使用 POST 方法

注册 URL /identifymodel

整个 URL:

`http://124.220.2.168:9391/identifymodel`

传送字节流

必选字段:

userid: 用户名称

token: 公司名称 id

content: 上传的语音内容, 系统固定 id="123456"

file: 文件标识

token 可以代表某个公司, 一个公司一个 token, 1:N 的情况下按照 token 分目录识别

不同公司的 token 不一样

语音缓冲 (可以合并传, 也可分段传, 但不能有间隔符):

buffer1 (录音缓冲区) +

buffer2 (录音缓冲区) +

buffer3 (录音缓冲区) +

.

.

.

bufferN (录音缓冲区)

服务器返回:

"1007" 验证完成

"700" 验证失败

"900" 验证检测失败

"2005" 模型不存在

"2012" 用户名不合法

Java 例子: `java identifymodel 1001 123456 wav/upload.wav`

## 六.返回值说明

### 6.1、ret 返回值

SUCCESS_REGISTER_USER	= 1001 // 注册用户成功
SUCCESS_DELETE_USER	= 1002 // 删除用成功
SUCCESS_CLEAR_SAMPLES	= 1003 // 清除数据库中训练语音缓存成功
SUCCESS_TRAIN_MODEL	= 1004 // 训练模型成功
SUCCESS_DELETE_MODEL	= 1005 // 删除模型成功
SUCCESS_ADDSAMPLE	= 1006 // 添加语音到数据库训练语音缓存成功
SUCCESS_VERIFY_MODEL	= 1007 // 验证完成
SUCCESS_DETECT_REGISTER	= 1010 // 登记检测通过
SUCCESS_DETECT_QUERY	= 1011 // 验证检测通过
FAILED_REGISTER_USER	= 100 // 注册用户失败
FAILED_DELETE_USER	= 200 // 删除用失败
FAILED_CLEAR_SAMPLES	= 300 // 清除数据库中训练语音缓存失败
FAILED_TRAIN_MODEL	= 400 // 训练模型失败
FAILED_DELETE_MODEL	= 500 // 删除模型失败
FAILED_ADDSAMPLE	= 600 // 添加语音到数据库训练语音缓存失败
FAILED_VERIFY_MODEL	= 700 // 验证失败
FAILED_DETECT_REGISTER	= 800 // 登记检测失败
FAILED_DETECT_QUERY	= 900 // 验证检测失败

### 6.2、errCode 返回值

ERROR_USER_EXISTENT	= 2001 // 用户已存在
ERROR_USER_NONEXISTENT	= 2002 // 用户不存在
ERROR_CLEAR_SAMPLES_FAILED	= 2003 // 清除数据库中训练语音缓存失败



ERROR_MODEL_EXISTENT	= 2004 // 模型已存在
ERROR_MODEL_NONEXISTENT	= 2005 // 模型不存在
ERROR_SAMPLES_NOT_ENOUGH	= 2006 // 数据库中训练语音缓存条数不足
ERROR_LENGTH_MISMATCH	= 2007 // 数据库中训练语音缓存条数与数据库中文本内容条数不同
ERROR_TRAIN_MODEL_FAILED	= 2008 // 训练模型失败
ERROR_SAMPLE_IS_NULL	= 2009 // 上传语音为空
ERROR_ADDSAMPLE_FAILED	= 2010 // 添加语音到数据库训练语音缓存失败
ERROR_VERIFY_MODEL_FAILED	= 2011 // 验证失败
ERROR_USER_ILLEGAL	= 2012 // 用户名不合法
ERROR_APP_TOKEN	= 2018 // 权限不合法

## 七.CURL 使用例子

添加语音，假如用户没注册会自动注册

```
curl http://124.220.2.168:9391/addsample -X POST -F "userid=1" -F  
"content=123456" -F token=123 -F step=1 -F "file=@./1.wav"
```

训练模型

```
curl http://124.220.2.168:9391/trainmodel -X POST -F "userid=1" -F  
"content=123456" -F token=123
```

比对声纹

```
curl http://124.220.2.168:9391/verifymodel -X POST -F "userid=1" -F  
"content=123456" -F token=123 -F "file=@./1.wav"
```

声纹识别

```
curl http://124.220.2.168:9391/identifymodel -X POST -F "userid=1" -F  
"content=123456" -F token=123 -F "file=@./1.wav"
```

## 八.java demo 例子

上传语音:

```
import java.io. File;
import java.io. FileInputStream;
import java.io. FileOutputStream;
import java.io. BufferedReader;
import java.io. OutputStream;
import java.io. DataInputStream;
import java.io. DataOutputStream;
import java.io. IOException;
import java.io. InputStreamReader;
import java.net. MalformedURLException;
import java.net. URL;
import java.net. URLConnection;
import java.net. HttpURLConnection;

import java.util. ArrayList;
import java.util. Iterator;
import java.util. LinkedList;
import java.util. List;
import java.util. ListIterator;
import java.util. Stack;
import java.util. Vector;
import java.util. Map;
import java.util. HashMap;
```

```
import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

public class addsample {

    public static void main(String[] args) {

        if (args.length != 4)

            {

                System.out.println("Usage: java addsampleuserid step content wavfile");

//addsample 1001 1 87436529 wav/upload.wav

                return;

            }

        String userid = args[0]; //1001

        String step = args[1]; //1~5

        String content = args[2]; //87436529

        String uploadfile = args[3]; //wav/upload.wav

        List<String> list = new ArrayList<String>(); //创建一个空 list.

        list.add(uploadfile); //要上传的语音文件名

        try {

            //定义数据分隔符

            String TWO_HYPHENS = "--";

            String BOUNDARY = "*****";

            String END = "\r\n";

            String CHARSET = "UTF-8";

            //上传语音

            URL url = new URL("http://119.3.22.24:3997/addsample");
```

```
URLConnection conn = (URLConnection) url.openConnection();  
// 发送 POST 请求必须设置如下两行  
conn.setDoOutput(true);  
conn.setDoInput(true);  
conn.setUseCaches(false);  
conn.setRequestMethod("POST");  
conn.setRequestProperty("connection", "Keep-Alive");  
conn.setRequestProperty("user-agent", "Mozilla/4.0 (compatible; MSIE  
6.0; Windows NT 5.1; SV1)");  
conn.setRequestProperty("Charset", "UTF-8");  
conn.setRequestProperty("Content-Type", "multipart/form-data;  
boundary=" + BOUNDARY);
```

```
String token = "test";  
//System.out.println(token);
```

```
HashMap<String, String>param = null;  
param = new HashMap<String, String>();  
param.put("userid", userid); //必选字段  
param.put("token", token); //必选字段  
param.put("step", step); //设置第几遍语音（总共 5 遍），必选字段  
param.put("content", content); //设置语音对应文本，必选字段  
param.put("sampleRate", "16000"); //可选字段  
param.put("bits", "16"); //可选字段  
param.put("channels", "1"); //可选字段
```

```
OutputStream out = new DataOutputStream(conn.getOutputStream());  
byte[] end_data = ("\r\n--" + BOUNDARY + "--\r\n").getBytes(); // 定
```

义最后数据分隔线

```
StringBuildersb = new StringBuilder();
for (Map.Entry<String, String> entry : param.entrySet()) {
    sb.append(TWO_HYPHENS);
    sb.append(BOUNDARY);
    sb.append(END);
    sb.append("Content-Disposition: form-data; name=\""
        + entry.getKey() + "\"");
    sb.append("Content-Type: text/plain; charset=" + CHARSET + END);
    sb.append("Content-Transfer-Encoding: 8bit" + END);
    sb.append(END);
    sb.append(entry.getValue());
    sb.append(END);
}

intleng = list.size();
for(inti=0;i<leng;i++){
    String fname = list.get(i);
    File file = new File(fname);

    sb.append("--");
    sb.append(BOUNDARY);
    sb.append("\r\n");
    sb.append("Content-Disposition:
form-data;name=\"file\";filename=\"" + file.getName() + "\"\r\n");
    sb.append("Content-Type:application/octet-stream\r\n\r\n");

    byte[] data = sb.toString().getBytes();
    out.write(data);
}
```

```
        DataInputStream in = new DataInputStream(new
FileInputStream(file));

        int bytes = 0;

        byte[] bufferOut = new byte[1024];

        while ((bytes = in.read(bufferOut)) != -1) {

            out.write(bufferOut, 0, bytes);

        }

        out.write("\r\n".getBytes()); //多个文件时，二个文件之间加入这个

        in.close();

    }

    out.write(end_data);

    out.flush();

    out.close();

    // 定义 BufferedReader 输入流来读取 URL 的响应

    BufferedReader reader = new BufferedReader(new

InputStreamReader(conn.getInputStream()));

    String line = null;

    while ((line = reader.readLine()) != null) {

        System.out.println(line);

    }

} catch (Exception e) {

    System.out.println("发送 POST 请求出现异常!" + e);

    e.printStackTrace();

}

}

}
```

## 八.技术支持

联系电话：0592-5998812

邮箱：info@talentedsoft.com