智能语音V2.X 一句话识别RESTful API使用文档

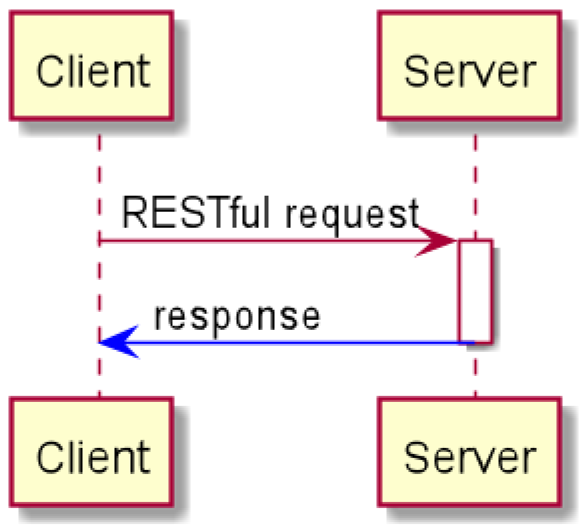
功能介绍

一句话识别RESTful API支持以POST方式整段上传不超过一分钟的语音文件。识别结果将以JSON格式在请求响应中一次性返回，开发者需要保证在识别结果返回之前连接不被中断。

- 支持音频编码格式：pcm（无压缩的pcm文件或wav文件）、opus，16bit采样位数的单声道(mono)；
- 支持音频采样率：8000、16000；
- 支持对返回结果进行设置：是否在后处理中添加标点，是否将中文数字转为阿拉伯数字输出；
- 支持配置项目热词和自学习模型训练；
- 支持多种方言识别。

交互流程

客户端向服务端发送带有音频数据的HTTP REST POST请求，服务端返回带有识别结果的HTTP响应。



上传音频文件

一句话识别请求HTTP报文实例：

```
POST /stream/v1/asr?appkey=default&format=pcm&sample_rate=16000&enable_punct
uation_prediction=true&enable_inverse_text_normalization=true HTTP/1.1
X-NLS-Token: default
Content-type: application/octet-stream
Content-Length: 94616
Host: gateway所在IP:8101

[audio data]
```

一个完整的一句话识别RESTful API请求需包含以下要素：

1.HTTP 请求行

HTTP的请求行指定了URL和请求参数。

URL：

协议	URL	方法
HTTP/1.1	http://gateway所在IP:8101/stream/v1/asr	POST

请求参数：

Parameter	Type	Description
appkey	String	应用appkey，必填，专有云为default
format	String	音频编码格式，可选， 支持的格式： pcm（无压缩的pcm文件或wav文件）、opus，默认是pcm
sample_rate	Integer	音频采样率，可选， 16000或者8000，默认是16000
enable_punctuation_prediction	Boolean	是否在后处理中添加标点， 可选，true或者false， 默认false不开启
enable_inverse_text_normalization	Boolean	是否在后处理中执行ITN， 可选，true或者false， 默认false不开启
enable_voice_detection	Boolean	是否启动语音检测，可选， true或者false， 默认false不开启。**说明： 如果开启语音检测， 服务端会对上传的音频进行静音检测， 切除静音部分和之后的语音内容，不再对其进行识别； 不同的模型表现结果不同。**
model	String	模型名称，可选， 若不设置则使用默认模型
customization_id	String	定制模型ID，可选
vocabulary_id	String	泛热词ID，可选

如上URL和请求参数组成的完整请求链接为：

```
http://gateway所在IP:8101/stream/v1/asr?appkey=default&format=pcm&sample_rate=16000&enable_punctuation_prediction=true&enable_inverse_text_normalization=true&enable_voice_detection=true
```

2.HTTP 请求头部

HTTP 请求头部由“关键字/值”对组成，每行一对，关键字和值用英文冒号“:”分隔，设置内容为如下表格：

名称	类型	需求	描述
X-NLS-Token	String	必填	服务鉴权Token， 专有云为 default
Content-type	String	必填	必须为“application/octet-stream”， 表明HTTP body的数据为二进制流
Content-Length	long	必填	HTTP body中请求数据的长度，即音频文件的长度
Host	String	必填	HTTP请求的服务器域名， 必须为“gateway所在IP:8101”

3.HTTP 请求体

HTTP请求体传入的是二进制音频数据，因此在HTTP请求头部中的Content-Type必须设置为application/octet-stream。

响应结果

发送上传音频的HTTP请求之后，会收到服务端的响应，识别的结果以JSON字符串的形式保存在该响应中。

- 成功响应

```
{
    "task_id": "cf7b0c5339244ee29cd4e43fb97fd52e",
    "result": "北京的天气。",
    "status":20000000,
    "message":"SUCCESS"
}
```

- 失败响应
  以鉴权token错误为例：

```
{
    "task_id": "8bae3613dfc54ebfa811a17d8a7a9ae7",
    "result": "",
    "status": 40000001,
    "message": "Gateway:ACCESS_DENIED:The token 'c0c1e860f3*******de8091c68
a' is invalid!"
}
```

响应字段说明：

Parameter	Type	Description
task_id	String	32位任务ID
result	String	语音识别结果
status	Integer	服务状态码
message	String	服务状态描述

服务状态码说明：
20000000表示成功，4开头的状态码表示客户端的错误，5开头的错误码表示服务端的错误。

服务状态码	服务状态描述	解决方案
20000000	请求成功	
40000000	默认的客户端错误码	查看错误消息或提交工单
40000001	身份认证失败 检查使用的令牌是否正确， 是否过期	
40000002	无效的消息	检查发送的消息是否符合要求

40000003	无效的参数	检查参数值设置是否合理
40000004	空闲超时	确认是否长时间没有发送数据掉服务端
40000005	请求数量过多	检查是否超过了并发连接数或者每秒钟请求数
50000000	默认的服务端错误	如果偶现可以忽略，重复出现请提交工单
50000001	内部GRPC调用错误	如果偶现可以忽略，重复出现请提交工单

快速测试

音频文件下载链接：nls-sample-16k.wav

使用cURL命令行可以快速进行一句话识别的RESTful API测试。

```
curl -X POST -H "X-NLS-Token: default" http://gateway所在IP:8101/stream/v1/as
r?appkey=default --data-binary @${audio_file}

示例：
curl -X POST -H "X-NLS-Token: default" http://gateway所在IP:8101/stream/v1/as
r?appkey=default --data-binary @./nls-sample-16k.wav
```

代码示例

音频文件说明：音频文件下载链接：nls-sample-16k.wav，16KHz，PCM编码。

以下各语言中需要设置的appkey和token均为default。
链接地址为 http://gateway所在IP:8101/stream/v1/asr， host为"gateway所在IP:8101"

Java Demo

依赖：

```
<dependency>
    <groupId>com.squareup.okhttp3</groupId>
    <artifactId>okhttp</artifactId>
    <version>3.9.1</version>
</dependency>

<!-- http://mvnrepository.com/artifact/com.alibaba/fastjson -->
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.42</version>
</dependency>
```

发送请求与响应：

```
package com.alibaba.nls.client.example;

import com.alibaba.fastjson.JSONPath;
import com.alibaba.nls.client.example.utils.HttpUtil;

import java.util.HashMap;

public class SpeechRecognizerRESTfulDemo {
    private String accessToken;
    private String appkey;
    private String url;

    public SpeechRecognizerRESTfulDemo(String appkey, String token, String ur
l) {
        this.appkey = appkey;
        this.accessToken = token;
        this.url = url;
    }

    public void process(String fileName, String format, int sampleRate,
                        boolean enablePunctuationPrediction,
                        boolean enableInverseTextNormalization,
                        boolean enableVoiceDetection) {

        /**
         * 设置HTTP REST POST请求
         * 1.使用http协议
         * 2.语音识别服务域名：gateway所在IP:8101
         * 3.语音识别接口请求路径：/stream/v1/asr
         * 4.设置必须请求参数：appkey、format、sample_rate，
         * 5.设置可选请求参数：enable_punctuation_prediction、enable_inverse_text
_normalization、enable_voice_detection
         */
        String request = this.url;
        request = request + "?appkey=" + appkey;
        request = request + "&format=" + format;
        request = request + "&sample_rate=" + sampleRate;
        if (enablePunctuationPrediction) {
            request = request + "&enable_punctuation_prediction=" + true;
        }
        if (enableInverseTextNormalization) {
            request = request + "&enable_inverse_text_normalization=" + true;
        }
        if (enableVoiceDetection) {
            request = request + "&enable_voice_detection=" + true;
        }

        // 指定自学习模型ID，需要时打开
        //request = request + "&customization_id=" + "您的自学习模型ID";
        // 指定泛热词ID，需要时打开
        //request = request + "&vocabulary_id=" + "您的泛热词ID";

        System.out.println("Request: " + request);

        /**
         * 设置HTTP 头部字段
         * 1.鉴权参数
         * 2.Content-Type: application/octet-stream
         */
        HashMap<String, String> headers = new HashMap<String, String>();
        headers.put("X-NLS-Token", this.accessToken);
        headers.put("Content-Type", "application/octet-stream");

        /**
         * 发送HTTP POST请求，返回服务端的响应
         */
        String response = HttpUtil.sendPostFile(request, headers, fileName);

        if (response != null) {
            System.out.println("Response: " + response);
            String result = JSONPath.read(response, "result").toString();
            System.out.println("识别结果: " + result);
        }
        else {
            System.err.println("识别失败!");
        }

    }

    public static void main(String[] args) {
        if (args.length < 1) {
            System.err.println("SpeechRecognizerRESTfulDemo need params: <gat
eway所在ip>");
            System.exit(-1);
        }

        String ip = args[0];
        String token = "default";
        String appkey = "default";
        String port = "8101";

        String url = "http://" + ip + ":" + port + "/stream/v1/asr";

        SpeechRecognizerRESTfulDemo demo = new SpeechRecognizerRESTfulDemo(a
ppkey, token, url);

        String fileName = SpeechRecognizerRESTfulDemo.class.getClassLoader().
getResource("./nls-sample-16k.wav").getPath();
        String format = "pcm";
        int sampleRate = 16000;
        boolean enablePunctuationPrediction = true;
        boolean enableInverseTextNormalization = true;
        boolean enableVoiceDetection = false;

        demo.process(fileName, format, sampleRate, enablePunctuationPredictio
n, enableInverseTextNormalization, enableVoiceDetection);
    }
}
```

HttpUtils 类:

```
package com.alibaba.nls.client.example.utils;

import okhttp3.Headers;
import okhttp3.MediaType;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;

import java.io.File;
import java.io.IOException;
import java.net.SocketTimeoutException;
import java.util.HashMap;
import java.util.Map;
import java.util.concurrent.TimeUnit;

public class HttpUtil {

    private static String getResponseWithTimeout(Request q) {
        String ret = null;

        OkHttpClient.Builder httpBuilder = new OkHttpClient.Builder();
        OkHttpClient client = httpBuilder.connectTimeout(10, TimeUnit.SECOND
S)
                .readTimeout(60, TimeUnit.SECONDS)
                .writeTimeout(60, TimeUnit.SECONDS)
                .build();

        try {
            Response s = client.newCall(q).execute();
            ret = s.body().string();
            s.close();
        } catch (SocketTimeoutException e) {
            ret = null;
            System.err.println("get result timeout");
        } catch (IOException e) {
            System.err.println("get result error " + e.getMessage());
        }

        return ret;
    }

    public static String sendPostFile(String url, HashMap<String, String> heade
rs, String fileName) {
        RequestBody body;

        File file = new File(fileName);
        if (!file.isFile()) {
            System.err.println("The filePath is not a file: " + fileName);
            return null;
        } else {
            body = RequestBody.create(MediaType.parse("application/octet-stre
am"), file);
        }

        Headers.Builder hb = new Headers.Builder();
        if (headers != null && !headers.isEmpty()) {
            for (Map.Entry<String, String> entry : headers.entrySet()) {
                hb.add(entry.getKey(), entry.getValue());
            }
        }

        Request request = new Request.Builder()
                .url(url)
                .headers(hb.build())
                .post(body)
                .build();

        return getResponseWithTimeout(request);
    }

    public static String sendPostData(String url, HashMap<String, String> heade
rs, byte[] data) {
        RequestBody body;

        if (data.length == 0) {
            System.err.println("The send data is empty.");
            return null;
        } else {
            body = RequestBody.create(MediaType.parse("application/octet-stre
am"), data);
        }

        Headers.Builder hb = new Headers.Builder();
        if (headers != null && !headers.isEmpty()) {
            for (Map.Entry<String, String> entry : headers.entrySet()) {
                hb.add(entry.getKey(), entry.getValue());
            }
        }

        Request request = new Request.Builder()
                .url(url)
                .headers(hb.build())
                .post(body)
                .build();
            return getResponseWithTimeout(request);
        }
    }
```

C++ Demo

C++ Demo使用了第三方函数库curl处理HTTP的请求和响应。Demo相关文件在nls-cpp-example/nls-cpp-restful目录下：

目录说明：

- CMakeLists.txt Demo工程的CMakeList文件；
- demo

文件名	描述
restfulAsrDemo.cpp	一句话RESTful API Demo

- include

文件名	描述
curl	curl库头文件目录

- lib 根据平台不同，可以选择linux版本（glibc:2.5及以上，Gcc4，Gcc5）、windows版本（VS2013、VS2015）。
- readme.txt 说明
- release.log 更新记录
- version 版本号
- build.sh demo编译脚本

注意：

1. Linux环境下，运行环境最低要求：Glibc 2.5及以上， Gcc4、Gcc5。
2. Windows下需要用户自己搭建demo工程。
3. C++ demo的下载包里自带了测试音频sample.pcm。

编译运行:

```
1. 请确认本地系统以安装Cmake，最低版本2.4
2. cd path/to/sdk/lib
3. tar -zxvpf linux.tar.gz
4. cd path/to/sdk
5. 执行[./build.sh]编译demo
6. 编译完毕，进入demo目录，执行[./restfulAsrDemo <gateway所在IP>]

如果不支持cmake，可以尝试手动编译：
1: cd path/to/sdk/lib
2: tar -zxvpf linux.tar.gz
3: cd path/to/sdk/demo
4: g++ -o restfulAsrDemo restfulAsrDemo.cpp -I path/to/sdk/include -L path/t
o/sdk/lib/linux -lssl -lcrypto -lcurl -D_GLIBCXX_USE_CXX11_ABI=0
5: export LD_LIBRARY_PATH=path/to/sdk/lib/linux/
6: ./restfulAsrDemo <gateway所在IP>

Windows平台需要用户自己搭建工程。
```

示例代码:

```
#include <iostream>
#include <string>
#include <fstream>
#include <sstream>
#include "curl/curl.h"

using namespace std;

#ifdef _WIN32
string UTF8ToGBK(const string& strUTF8) {
    int len = MultiByteToWideChar(CP_UTF8, 0, strUTF8.c_str(), -1, NULL, 0);
    unsigned short * wszGBK = new unsigned short[len + 1];
    memset(wszGBK, 0, len * 2 + 2);

    MultiByteToWideChar(CP_UTF8, 0, (char*)strUTF8.c_str(), -1, (wchar_t*)wsz
GBK, len);

    len = WideCharToMultiByte(CP_ACP, 0, (wchar_t*)wszGBK, -1, NULL, 0, NUL
L, NULL);

    char *szGBK = new char[len + 1];
    memset(szGBK, 0, len + 1);
    WideCharToMultiByte(CP_ACP, 0, (wchar_t*)wszGBK, -1, szGBK, len, NULL, N
ULL);

    string strTemp(szGBK);
    delete[] szGBK;
    delete[] wszGBK;

    return strTemp;
}

#endif

/**
* 一句话识别RESTful API HTTP请求的响应回调函数
* 识别结果为JSON格式的字符串
*/
size_t responseCallback(void* ptr, size_t size, size_t nmemb, void* userData)
{
    string* srResult = (string*)userData;

    size_t len = size * nmemb;
    char *pBuf = (char*)ptr;
    string response = string(pBuf, pBuf + len);
#ifdef _WIN32
    response = UTF8ToGBK(response);
#endif
    cout << "current result: " << response << endl;

    *srResult += response;
    cout << "total result: " << *srResult << endl;

    return len;
}

int sendAsrRequest(const char* request, const char* token, const char* fileNa
me, string* srResult) {
    CURL* curl = NULL;
    CURLcode res;

    /**
    * 读取音频文件
    */
    ifstream fs;
    fs.open(fileName, ios::out | ios::binary);
    if (!fs.is_open()) {
        cerr << "The audio file is not exist!" << endl;
        return -1;
    }
    stringstream buffer;
    buffer << fs.rdbuf();
    string audioData(buffer.str());

    curl = curl_easy_init();
    if (curl == NULL) {
        return -1;
    }

    /**
    * 设置HTTP请求行
    */
    curl_easy_setopt(curl, CURLOPT_CUSTOMREQUEST, "POST");
    curl_easy_setopt(curl, CURLOPT_URL, request);

    /**
    * 设置HTTP请求头部
    */
    struct curl_slist* headers = NULL;
    // token
    string X_NLS_Token = "X-NLS-Token:";
    X_NLS_Token += token;
    headers = curl_slist_append(headers, X_NLS_Token.c_str());
    // Content-Type
    headers = curl_slist_append(headers, "Content-Type:application/octet-strea
m");
    // Content-Length
    string content_Length = "Content-Length:";
    ostringstream oss;
    oss << content_Length << audioData.length();
    content_Length = oss.str();
    headers = curl_slist_append(headers, content_Length.c_str());

    curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headers);

    /**
    * 设置HTTP请求数据
    */
    curl_easy_setopt(curl, CURLOPT_POSTFIELDS, audioData.c_str());
    curl_easy_setopt(curl, CURLOPT_POSTFIELDSIZE, audioData.length());

    /**
```

```
     * 设置HTTP请求的响应回调函数
     */
    curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, responseCallback);
    curl_easy_setopt(curl, CURLOPT_WRITEDATA, srResult);

    /**
     * 发送HTTP请求
     */
    res = curl_easy_perform(curl);

    // 释放资源
    curl_slist_free_all(headers);
    curl_easy_cleanup(curl);

    if (res != CURLE_OK) {
        cerr << "curl_easy_perform failed: " << curl_easy_strerror(res) << e
ndl;
        return -1;
    }

    return 0;
}

int process(const char* request, const char* token, const char* fileName) {
    // 全局只初始化一次
    curl_global_init(CURL_GLOBAL_ALL);

    string srResult = "";
    int ret = sendAsrRequest(request, token, fileName, &srResult);

    curl_global_cleanup();

    return ret;
}

int main(int argc, char* argv[]) {
    if (argc < 2) {
        cerr << "params is not valid. Usage: ./demo <gateway所在IP>" << endl;
        return -1;
    }

    string ip = argv[1];
    string token = "default";
    string appKey = "default";
    string port = "8101";
    string url = "http://" + ip + ":" + port + "/stream/v1/asr";
    string format = "pcm";
    int sampleRate = 16000;
    bool enablePunctuationPrediction = true;
    bool enableInverseTextNormalization = true;
    bool enableVoiceDetection = false;
    string fileName = "sample.pcm";

    /**
    * 设置RESTful请求参数
    */
    ostringstream oss;
    oss << url;
    oss << "?appkey=" << appKey;
    oss << "&format=" << format;
    oss << "&sample_rate=" << sampleRate;
    if (enablePunctuationPrediction) {
        oss << "&enable_punctuation_prediction=" << "true";
    }
    if (enableInverseTextNormalization) {
        oss << "&enable_inverse_text_normalization=" << "true";
    }
    if (enableVoiceDetection) {
        oss << "&enable_voice_detection=" << "true";
    }

    // 指定自学习模型ID，需要时打开
    // oss << "&customization_id=" << "您的自学习模型ID";
    // 指定泛热词ID，需要时打开
    // oss << "&vocabulary_id=" << "您的泛热词ID";

    string request = oss.str();
    cout << "request: " << request << endl;

    process(request.c_str(), token.c_str(), fileName.c_str());

    return 0;
}
```

Python Demo

注意：Python 2.x请使用httplib模块；Python 3.x请使用http.client模块。

```
# -*- coding: UTF-8 -*-

# Python 2.x 引入httplib模块
# import httplib

# Python 3.x 引入http.client模块
import http.client

import json

def process(request, token, audioFile, host) :

    # 读取音频文件
    with open(audioFile, mode = 'rb') as f:
        audioContent = f.read()


    # 设置HTTP请求头部
    httpHeaders = {
        'X-NLS-Token': token,
        'Content-type': 'application/octet-stream',
        'Content-Length': len(audioContent)
        }


    # Python 2.x 请使用httplib
    # conn = httplib.HTTPConnection(host)

    # Python 3.x 请使用http.client
    conn = http.client.HTTPConnection(host)

    conn.request(method='POST', url=request, body=audioContent, headers=httpH
eaders)

    response = conn.getresponse()
    print('Response status and response reason:')
    print(response.status ,response.reason)

    body = response.read()
    try:
        print('Recognize response is:')
        body = json.loads(body)
        print(body)

        status = body['status']
        if status == 20000000 :
            result = body['result']
            print('Recognize result: ' + result)
        else :
            print('Recognizer failed!')

    except ValueError:
        print('The response is not json format string')

    conn.close()



appKey = 'default'
token = 'default'

# 服务请求地址

host = 'gateway所在IP:8101'

url = 'http://' + host + '/stream/v1/asr'


# 音频文件
audioFile = 'nls-sample-16k.wav'
format = 'pcm'
sampleRate = 16000
enablePunctuationPrediction  = True
enableInverseTextNormalization = True
enableVoiceDetection  = False

# 设置RESTful请求参数
request = url + '?appkey=' + appKey
request = request + '&format=' + format
request = request + '&sample_rate=' + str(sampleRate)

if enablePunctuationPrediction :
    request = request + '&enable_punctuation_prediction=' + 'true'

if enableInverseTextNormalization :
    request = request + '&enable_inverse_text_normalization=' + 'true'
```

```
if enableVoiceDetection :
    request = request + '&enable_voice_detection=' + 'true'

# 指定自学习模型ID，需要时打开
# request = request + '&customization_id=' + '您的自学习模型ID'
# 指定泛热词ID，需要时打开
# request = request + '&vocabulary_id=' + '您的泛热词ID'

print('Request: ' + request)

process(request, token, audioFile, host)
```

PHP Demo

```
<?php

function process($token, $request, $audioFile) {
    /**
     * 读取音频文件
     */
    $audioContent = file_get_contents($audioFile);
    if ($audioContent == FALSE) {
        print "The audio file is not exist!\n";
        return;
    }

    $curl = curl_init();
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl, CURLOPT_TIMEOUT, 120);

    /**
     * 设置HTTP请求行
     */
    curl_setopt($curl, CURLOPT_URL, $request);
    curl_setopt($curl, CURLOPT_POST,TRUE);

    /**
     * 设置HTTP请求头部
     */
    $contentType = "application/octet-stream";
    $contentLength = strlen($audioContent);
    $headers = array(
        "X-NLS-Token:" . $token,
        "Content-type:" . $contentType,
        "Content-Length:" . strval($contentLength)
    );
    curl_setopt($curl, CURLOPT_HTTPHEADER, $headers);

    /**
     * 设置HTTP请求数据
     */
    curl_setopt($curl, CURLOPT_POSTFIELDS, $audioContent);
    curl_setopt($curl, CURLOPT_NOBODY, FALSE);

    /**
     * 发送HTTP请求
     */
    $returnData = curl_exec($curl);

    curl_close($curl);

    if ($returnData == FALSE) {
        print "curl_exec failed!\n";
        return;
    }

    print $returnData . "\n";

    $resultArr = json_decode($returnData, true);

    $status = $resultArr["status"];
    if ($status == 20000000) {
        $result = $resultArr["result"];
        print "The audio file recognized result: " . $result . "\n";
    }
    else {
        print "The audio file recognized failed.\n";
    }

}


$appkey = "default";
$token = "default";

$url = "http://gateway所在IP:8101/stream/v1/asr";
$audioFile = "/path/to/nls-sample-16k.wav";
```

```
$format = "pcm";
$sampleRate = 16000;
$enablePunctuationPrediction = TRUE;
$enableInverseTextNormalization = TRUE;
$enableVoiceDetection = FALSE;

/**
 * 设置RESTful 请求参数
 */
$request = $url;
$request = $request . "?appkey=" . $appkey;
$request = $request . "&format=" . $format;
$request = $request . "&sample_rate=" . strval($sampleRate);
if ($enablePunctuationPrediction) {
    $request = $request . "&enable_punctuation_prediction=" . "true";
}
if ($enableInverseTextNormalization) {
    $request = $request . "&enable_inverse_text_normalization=" . "true";
}
if ($enableVoiceDetection) {
    $request = $request . "&enable_voice_detection=" . "true";
}

// 指定自学习模型ID，需要时打开
// $request = $request . "&customization_id=" . "您的自学习模型ID";
// 指定泛热词ID，需要时打开
// $request = $request . "&vocabulary_id=" . "您的泛热词ID";

print "Request: " . $request . "\n";

process($token, $request, $audioFile);

?>
```

Node.js Demo

说明：request依赖安装，请在您的Demo文件所在目录执行如下命令：

```
npm install request --save
```

代码示例：

```
const request = require('request');
const fs = require('fs');


function callback(error, response, body) {
    if (error != null) {
        console.log(error);
    }
    else {
        console.log('The audio file recognized result:');
        console.log(body);
        if (response.statusCode == 200) {
            body = JSON.parse(body);
            if (body.status == 20000000) {
                console.log('result: ' + body.result);
                console.log('The audio file recognized succeed!');
            } else {
                console.log('The audio file recognized failed!');
            }
        } else {
            console.log('The audio file recognized failed, http code: ' + resp
onse.statusCode);
        }
    }
}

function process(requestUrl, token, audioFile) {
    /**
     * 读取音频文件
    */
    var audioContent = null;
    try {
        audioContent = fs.readFileSync(audioFile);
    } catch(error) {
        if (error.code == 'ENOENT') {
            console.log('The audio file is not exist!');
        }
        return;
    }

    /**
     * 设置HTTP 请求头部
    */
    var httpHeaders = {
        'X-NLS-Token': token,
        'Content-type': 'application/octet-stream',
        'Content-Length': audioContent.length
    };

    var options = {
        url: requestUrl,
        method: 'POST',
        headers: httpHeaders,
        body: audioContent
    };

    request(options, callback);
}


var appkey = 'default';
var token = 'default';
var url = 'http://gateway所在IP:8101/stream/v1/asr';

var audioFile = '/path/to/nls-sample-16k.wav';
var format = 'pcm';
var sampleRate = '16000';
var enablePunctuationPrediction = true;
var enableInverseTextNormalization = true;
var enableVoiceDetection  = false;

/**
 * 设置RESTful 请求参数
 */
var requestUrl = url;
requestUrl = requestUrl + '?appkey=' + appkey;
requestUrl = requestUrl + '&format=' + format;
requestUrl = requestUrl + '&sample_rate=' + sampleRate;
if (enablePunctuationPrediction) {
    requestUrl = requestUrl + '&enable_punctuation_prediction=' + 'true';
}
if (enableInverseTextNormalization) {
    requestUrl = requestUrl + '&enable_inverse_text_normalization=' + 'true';
}
if (enableVoiceDetection) {
    requestUrl = requestUrl + '&enable_voice_detection=' + 'true';
}

// 指定自学习模型ID，需要时打开
// requestUrl = requestUrl + '&customization_id=' + '您的自学习模型ID';
// 指定泛热词ID，需要时打开
// requestUrl = requestUrl + '&vocabulary_id=' + '您的泛热词ID';

process(requestUrl, token, audioFile);
```