ALIBABA CLOUD

阿里云

智能语音交互
录音文件识别极速版

文档版本：20220329

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.接口说明

录音文件识别极速版支持使用者通过HTTPS POST方式上传一段短音频并在短时间内（一般来说，30分钟的音频可以在10秒内完成识别）同步获取识别结果，满足音视频字幕、准实时质检等场景下对语音文件识别时效性要求。

功能介绍

- 音频格式：支持AAC、MP3、OPUS、WAV格式编码的音频。
- 使用限制：支持100 MB以内且时长不超过2小时的音频文件的识别，时长超过2小时的文件请使用录音文件识别普通版。
- 模型类型：8000（电话）和16000（非电话）。

> 说明
>
> 服务端根据请求参数中的采样率对不符合要求的音频自动进行采样率调整。

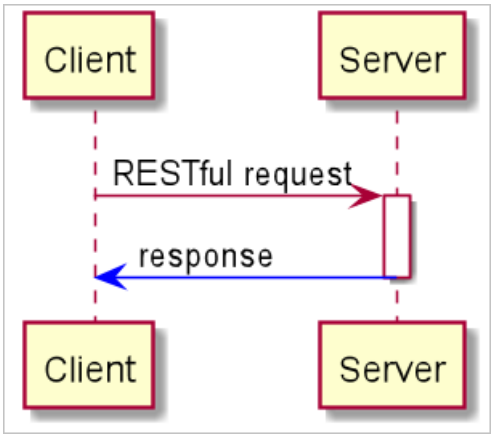
- 支持设置返回结果：支持设置是否将中文数字转为阿拉伯数字输出，支持对多声道音频只处理首个声道。
- 支持控制台配置项目热词、定制语言模型。
- 目前支持中文电话通用、中文非电话通用和英文非电话通用三种场景，可在控制台编辑项目进行模型配置。更多信息，请参见管理项目。

前提条件

- 已获取项目Appkey，更多信息，请参见创建项目。
- 已获取Access Token，更多信息，请参见获取Token。

交互流程

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



> 说明
>
> 服务端的错误响应会在返回信息中包含表示本次合成任务的task_id参数，请记录该值，如果出现错误，请将task_id和错误信息提交到工单。

服务地址

访问类型	说明	URL	Host
外网访问	所有服务器均可使用外网访问URL。	https://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/FlashRecognizer	nls-gateway.cn-shanghai.aliyuncs.com
阿里云上海ECS内网访问	使用阿里云上海ECS（即ECS地域为华东2（上海）），可使用内网访问URL。 ECS的经典网络不能访问AnyTunnel，即不能在内网访问语音服务；如果希望使用AnyTunnel，需要创建专有网络在其内部访问。 说明 • 使用内网访问方式，将不会产生ECS实例的公网流量费用。 • 关于ECS的网络类型请参见网络类型。	http://nls-gateway.cn-shanghai-internal.aliyuncs.com/stream/v1/FlashRecognizer	nls-gateway.cn-shanghai-internal.aliyuncs.com

注意

以下将以使用外网访问URL的方式进行介绍。如果您使用阿里云上海ECS，并需要通过内网访问URL，则使用HTTPS协议，并替换外网访问的URL和Host。

上传音频文件

请求HTTPS报文示例。

```
POST /stream/v1/FlashRecognizer?appkey=23f5****&token=450372e4279bcc2b3c793****&format=wav&
sample_rate=16000 HTT
Content-type: application/octet-stream
Content-Length: 94616
Host: nls-gateway.cn-shanghai.aliyuncs.com

[audio data]
```

一个完整的文件识别极速版RESTful API请求需包含以下要素：HTTPS请求行、HTTPS请求头部和HTTPS请求体。

- HTTPS请求行

  HTTPS请求行指定了**URL**和**请求参数**，由URL和请求参数组成的完整请求链接如下：

  ```
  https://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/FlashRecognizer?appkey=Yu1******uncS&format=wav&sample_rate=16000&vocabulary_id=a17******d6b&token=1234*****
  ```

  - URL

协议	URL	方法
HTTP/1.1	nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/FlashRecognizer	POST

  - 请求参数

参数	类型	是否必选	描述
appkey	String	是	应用Appkey。
format	String	是	音频编码格式。支持格式：WAV/OPUS/AAC/MP3。
token	String	是	鉴权Token。
sample_rate	Integer	否	表示语音识别模型的采样率，上传的音频如果不符合其取值会被自动升/降采样率至8000或16000。取值：16000（非电话）/8000（电话）。默认：16000。
vocabulary_id	String	否	添加热词表ID。默认：不添加。
customization_id	String	否	添加自学习模型ID。默认：不添加。
enable_inverse_text_normalization	Boolean	否	是否在后处理中执行ITN。取值：true或false。默认：false（不开启）。

参数	类型	是否必选	描述
enable_word_level_result	Boolean	否	是否返回词级别信息。取值：true或false。默认：false（不开启）。
enable_timestamp_alignment	Boolean	否	是否启用时间戳校准功能，取值：true或false，默认：false（不开启）。
first_channel_only	Boolean	否	是否只识别首个声道，取值：true或false。默认：false（不开启）。 **说明** 如果为多声道将会叠加计费。例如，双声道为双倍计费。
speech_noise_threshold	Float	否	噪音参数阈值，取值范围：[-1, 1]。取值说明如下： ■ 取值越趋于-1，噪音被判定为语音的概率越大。 ■ 取值越趋于+1，语音被判定为噪音的概率越大。 **注意** 该参数属高级参数，请慎重调整并进行重点测试。

- HTTPS请求头部

  HTTPS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，设置内容如下：

名称	类型	是否必选	描述

名称	类型	是否必选	描述
Content-type	String	是	取值“application/octet-stream”，表明HTTPS请求体的数据为二进制流。
Content-Length	long	是	HTTPS请求体中请求数据的长度，即音频文件的长度。
Host	String	是	取值“nls-gateway.cn-shanghai.aliyuncs.com”，为HTTPS请求的服务器域名。

- HTTPS请求体

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

使用音频文件链接

如果希望使用非本地音频文件直接进行识别，可以使用音频链接的方式请求，此种调用方式无需上传音频文件。请求HTTPS报文示例如下：

```
POST /stream/v1/FlashRecognizer?appkey=23f5****&token=450372e4279bcc2b3c793****&format=wav&sample_rate=16000&audio_address=https://gw.alipayobjects.com/os/bmw-prod/0574ee2e-f494-45a5-820f-63aee583045a.wa
HTTP Content-type: application/text
Host: nls-gateway.cn-shanghai.aliyuncs.com
```

一个完整的文件识别极速版RESTful API请求需包含以下要素：HTTPS请求行和HTTPS请求头部。

- HTTPS请求行

  HTTPS请求行指定了**URL**和**请求参数**，由URL和请求参数组成的完整请求链接如下：

  ```
  https://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/FlashRecognizer?appkey=Yu1******uncS&format=wav&sample_rate=16000&vocabulary_id=a17******d6b&token=1234*****
  ```

- URL

协议	URL	方法
HTTP/1.1	nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/FlashRecognizer	POST

- 请求参数

参数	类型	是否必选	描述
appkey	String	是	应用Appkey。
format	String	是	音频编码格式。支持格式：WAV、OPUS、AAC、MP3。
token	String	是	鉴权Token。
sample_rate	Integer	否	表示语音识别模型的采样率，上传的音频如果不符合其取值会被自动升/降采样率至8000或16000。取值：16000（非电话）或8000（电话）。默认：16000。
vocabulary_id	String	否	添加热词表ID。默认：不添加。
customization_id	String	否	添加自学习模型ID。默认：不添加。
enable_inverse_text_normalization	Boolean	否	是否在后处理中执行ITN。取值：true或false。默认：false（不开启）。
enable_word_level_result	Boolean	否	是否返回词级别信息。取值：true或false。默认：false（不开启）。
enable_timestamp_alignment	Boolean	否	是否启用时间戳校准功能，取值：true或false，默认：false（不开启）。

参数	类型	是否必选	描述
first_channel_only	Boolean	否	是否只识别首个声道，取值：true或false。默认：false（不开启）。 **说明** 如果为多声道将会叠加计费。例如，双声道为双倍计费。
speech_noise_threshold	Float	否	噪音参数阈值，取值范围：[-1, 1]。取值说明如下： ■ 取值越趋于-1，噪音被判定为语音的概率越大。 ■ 取值越趋于+1，语音被判定为噪音的概率越大。 **注意** 该参数属高级参数，请慎重调整并进行重点测试。
audio_address	String	是	使用音频文件链接时必填，存放录音文件的地址，推荐使用阿里云OSS。

- HTTPS请求头部

  HTTPS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，设置内容如下：

名称	类型	是否必选	描述
Content-type	String	是	取值“application/text”。
Host	String	是	取值“nls-gateway.cn-shanghai.aliyuncs.com”，为HTTPS请求的服务器域名。

响应结果

发送上传音频的HTTPS请求后，会收到服务端的响应，识别结果以JSON字符串的形式保存在响应结果中。

- 成功响应


```
{
    "task_id":"a819f959441b49****",
    "status":20000000,
    "message":"SUCCESS",
    "flash_result":{
        "duration":22,
        "sentences":[
            {
                "text":"喂，",
                "words":[
                    {
                        "text":"喂",
                        "begin_time":"1010",
                        "end_time":"1520",
                        "punc":"，"
                    }
                ],
                "begin_time":1010,
                "end_time":1520,
                "channel_id":1
            },
            {
                "text":"哎，",
                "words":[
                    {
                        "text":"哎",
                        "begin_time":"3540",
                        "end_time":"3570",
                        "punc":"，"
                    }
                ],
                "begin_time":3540,
                "end_time":3570,
                "channel_id":1
            },
            {
                "text":"那五点五十现在是多少点啊，",
                "words":[
                    {
                        "text":"那",
                        "begin_time":"8810",
                        "end_time":"8840",
                        "punc":""
                    },
                    {
                        "text":"五点",
                        "begin_time":"12750",
                        "end_time":"13200",
                        "punc":""
                    },
                    {
                        "text":"五十",
```

```
                        "begin_time":"13200",
                        "end_time":"13890",
                        "punc":""
                    },
                    {
                        "text":"现在",
                        "begin_time":"14330",
                        "end_time":"14600",
                        "punc":""
                    },
                    {
                        "text":"是",
                        "begin_time":"14600",
                        "end_time":"14720",
                        "punc":""
                    },
                    {
                        "text":"多少",
                        "begin_time":"14720",
                        "end_time":"14960",
                        "punc":""
                    },
                    {
                        "text":"点",
                        "begin_time":"19570",
                        "end_time":"21220",
                        "punc":""
                    },
                    {
                        "text":"啊",
                        "begin_time":"21220",
                        "end_time":"22330",
                        "punc":""
                    }
                ],
                "begin_time":8810,
                "end_time":22330,
                "channel_id":1
            }
        ]
    }
}
```


响应字段说明如下：

参数	类型	描述
task_id	String	32位任务ID。请您记录该值，以便排查错误。
status	Integer	服务状态码。

参数	类型	描述
message	String	服务状态描述。

flash_result对象参数说明：

参数	类型	描述
duration	Integer	音频时长。
latency	String	服务端处理时长，单位：毫秒。
sentences.text	String	句子级别的识别结果。
sentences.begin_time	Integer	句子的开始时间，单位：毫秒。
sentences.end_time	Integer	句子的结束时间，单位：毫秒。
sentences.channel_id	Integer	多个声道的音频文件会区分返回识别结果，声道ID从0计数。
sentences.words.text	String	当前句子包含的词信息。
sentences.words.punc	String	当前词尾的标点信息，没有标点则为空。
sentences.words.begin_time	Integer	当前词开始时间，单位：毫秒。
sentences.words.end_time	Integer	当前词结束时间，单位：毫秒。

- 失败响应

  以鉴权token无效为例：

```
{
    "task_id": "8bae3613dfc54ebfa811a17d8a7a****",
    "result": "",
    "status": 40000001,
    "message": "Gateway:ACCESS_DENIED:The token 'c0c1e860f3*******de8091c68a' is invalid!
"
}
```

服务状态码

服务状态码	服务状态描述	解决方案
20000000	请求成功	无。
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。
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。
40270001	不支持的音频格式	请求音频格式不在支持列表。
40270002	无效的音频	从音频中没有识别出有效文本。
40270003	音频解码错误	按请求格式对音频解码时遇到错误。
40270004	无有效音频流	多声道的音频中未抽取到有效音频流。
50000000	默认的服务端错误	偶现可以忽略，重复出现请提交工单。
50000001	内部GRPC调用错误	偶现可以忽略，重复出现请提交工单。

快速测试

> **说明**
>
> 音频示例WAV文件，使用通用模型。若使用其他音频文件，请填入相应的编码格式和采样率，并在管控台设置对应的模型。

1. 下载音频文件nls-sample-16k.wav。
2. 使用如下cURL命令行进行极速版识别的RESTful API测试。

```
curl -X POST  'https://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/FlashRecognizer?appkey=${appkey}&token=${token}&format=wav&sample_rate=16000' --data-binary @${audio_file}
```

举例如下：

```
curl -X POST 'https://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/FlashRecognizer?appkey=tt43P2u****&token=4a036531cfdd****&format=wav&sample_rate=16000' --data-binary @./nls-sample-16k.wav
```

Java示例

依赖文件如下：

```
<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>1.2.68.noneautotype</version>
</dependency>
```

发送请求与响应：

```
import java.util.HashMap;
import com.alibaba.fastjson.JSONPath;
public class SpeechFlashRecognizerDemo {
    private String appkey;
    public SpeechFlashRecognizerDemo(String appkey) {
        this.appkey = appkey;
    }
    public void process(String fileName, String token,String format, int sampleRate) {
        /**
         * 设置HTTPS REST POST请求
         * 1.使用http协议
         * 2.语音识别服务域名：nls-gateway.cn-shanghai.aliyuncs.com
         * 3.语音识别接口请求路径：/stream/v1/FlashRecognizer
         * 4.设置必须请求参数：appkey、token、format、sample rate
```

```
         */
        String url = "https://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/FlashRecognize
r";
        String request = url;
        request = request + "?appkey=" + appkey;
        request = request + "&token=" + token;
        request = request + "&format=" + format;
        request = request + "&sample_rate=" + sampleRate;

        System.out.println("Request: " + request);
        /**
         * 设置HTTPS头部字段
         *
         * 1.Content-Type：application/octet-stream
         */
        HashMap<String, String> headers = new HashMap<String, String>();
//         headers.put("Content-Type", "application/octet-stream");
        /**
         * 发送HTTPS POST请求，返回服务端的响应。
         */
        long start = System.currentTimeMillis();
        String response = HttpUtil.sendPostFile(request, headers, fileName);
        System.out.println("latency = " + (System.currentTimeMillis() - start) + " ms");
        if (response != null) {
            System.out.println("Response: " + response);
        }
        else {
            System.err.println("识别失败!");
        }
    }
    public static void main(String[] args) {
        if (args.length < 2) {
            System.err.println("SpeechRecognizerRESTfulDemo need params: <token> <app-key>"
);
            System.exit(-1);
        }
        String token = args[0];
        String appkey = args[1];


        SpeechFlashRecognizerDemo demo = new SpeechFlashRecognizerDemo(appkey);
        //String fileName = SpeechRecognizerRestfulDemo.class.getClassLoader().getResource(
"./nls-sample-16k.wav").getPath();
        // 重要：此处用一个本地文件来模拟发送实时流数据，实际使用时，您可以从某处实时采集或接收语音流并
发送到ASR服务端。
        String fileName = "./nls-sample-16k.wav";
        String format = "wav";
        int sampleRate = 16000;

        demo.process(fileName,token, format, sampleRate);
    }
}
```

其中，HttpUtils类如下：

```
import okhttp3.*;
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.SECONDS)
                .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> headers, String f
ileName) {
        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-stream"), 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> headers, byte[] d
ata) {
        RequestBody body;

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

C++示例使用第三方函数库curl处理HTTPS请求和响应，下载curl库和示例文件。

目录说明如下：

文件/文件夹	说明
CMakeLists.txt	示例工程的CMakeList文件。
demo	其中的restfulFlashRecognizerDemo.cpp是极速版RESTful API示例。

文件/文件夹	说明
include	其中curl文件夹是curl库头文件目录。
lib	包含curl动态库，版本为curl-7.60。根据平台不同选如下版本： • Linux：Glibc 2.5、Gcc 4/Gcc 5。 • Windows：VS2013、VS2015。
readme.txt	说明。
release.log	更新记录。
version	版本号。
build.sh	示例编译脚本。

编译运行操作步骤：

假设示例文件已解压至 `path/to` 路径下，在Linux终端依次执行如下命令编译运行程序。

- 支持Cmake：
  i. 确认本地系统已安装Cmake 2.4及以上版本。
  ii. 切换目录：`cd path/to/sdk/lib` 。
  iii. 解压缩文件：`tar -zxvpf linux.tar.gz` 。
  iv. 切换目录：`cd path/to/sdk` 。
  v. 执行编译脚本：`./build.sh` 。
  vi. 切换目录：`cd path/to/sdk/demo` 。
  vii. 执行示例程序：`./restfulFlashRecognizerDemo <your-token> <your-appkey>` 。
- 不支持Cmake：
  i. 切换目录：`cd path/to/sdk/lib` 。
  ii. 解压缩文件：`tar -zxvpf linux.tar.gz` 。
  iii. 切换目录：`cd path/to/sdk/demo` 。

iv. 使用g++编译命令编译示例程序：

```
 g++ -o restfulFlashRecognizerDemo restfulFlashRecognizerDemo.cpp -I
path/to/sdk/include -L path/to/sdk/lib/linux -lssl -lcrypto -lcurl -
D_GLIBCXX_USE_CXX11_ABI=0
```

。

v. 指定库路径：`export LD_LIBRARY_PATH=path/to/sdk/lib/linux/` 。

vi. 执行示例程序：`./restfulFlashRecognizerDemo <your-token> <your-appkey>` 。

```
#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*)wszGBK, len);
    len = WideCharToMultiByte(CP_ACP, 0, (wchar_t*)wszGBK, -1, NULL, 0, NULL, NULL);
    char *szGBK = new char[len + 1];
    memset(szGBK, 0, len + 1);
    WideCharToMultiByte(CP_ACP, 0, (wchar_t*)wszGBK, -1, szGBK, len, NULL, NULL);
    string strTemp(szGBK);
    delete[] szGBK;
    delete[] wszGBK;
    return strTemp;
}

#endif

/**
 * RESTful API HTTPS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
    *srResult += response;
    cout << "current result: " << response << endl;
    cout << "total result: " << *srResult << endl;
    return len;
}

int sendAsrRequest(const char* request, const char* token, const char* fileName, string* sr
```

```
Result) {
    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HTTPS请求行
    */
    curl_easy_setopt(curl, CURLOPT_CUSTOMREQUEST, "POST");
    curl_easy_setopt(curl, CURLOPT_URL, request);

    /**
    * 设置HTTPS请求头部
    */
    struct curl_slist* headers = NULL;

    // Content-Type
    headers = curl_slist_append(headers, "Content-Type:application/octet-stream");
    // 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HTTPS请求数据
    */
    curl_easy_setopt(curl, CURLOPT_POSTFIELDS, audioData.c_str());
    curl_easy_setopt(curl, CURLOPT_POSTFIELDSIZE, audioData.length());

    /**
    * 设置HTTPS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HTTPS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) << endl;
        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 < 3) {
        cerr << "params is not valid. Usage: ./demo your_token your_appkey" << endl;
        return -1;
    }
    string token = argv[1];
    string appKey = argv[2];
    string url = "https://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/FlashRecognizer";
    string format = "wav";
    int sampleRate = 16000;
    string fileName = "nls-sample-16k.wav";

    /**
    * 设置RESTful请求参数
    */
    ostringstream oss;
    oss << url;
    oss << "?appkey=" << appKey;
    oss << "&token=" << token;
    oss << "&format=" << format;
    oss << "&sample_rate=" << sampleRate;

    string request = oss.str();
    cout << "request: " << request << endl;
    process(request.c_str(), token.c_str(), fileName.c_str());
    return 0;
}
```

Python示例

说明

- Python 2.x请使用httplib模块；Python 3.x请使用http.client模块。
- 采用RFC 3986规范进行urlencode编码，Python 2.x请使用urllib模块的urllib.quote，Python 3.x请使用urllib.parse模块的urllib.parse.quote_plus。
- 如果使用内网访问URL，请使用HTTP协议，需要替换如下函数，即将 `HTTPSConnection` 修改为 `HTTPConnection` ：

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

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

```
# -*- 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,  audioFile) :
    # 读取音频文件
    with open(audioFile, mode = 'rb') as f:
        audioContent = f.read()
    host = 'nls-gateway.cn-shanghai.aliyuncs.com'
    # 设置HTTP请求头部
    httpHeaders = {
        'Content-Length': len(audioContent)
        }
    # Python 2.x使用httplib
    # conn = httplib.HTTPConnection(host)

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

    conn.request(method='POST', url=request, body=audioContent, headers=httpHeaders)
    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 = '您的appkey'
token = '您的token'

# 服务请求地址
url = 'https://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/FlashRecognizer'

# 音频文件，下载地址：https://gw.alipayobjects.com/os/bmw-prod/0574ee2e-f494-45a5-820f-63aee583045a.wav
audioFile = 'nls-sample-16k.wav'

format = 'wav'
sampleRate = 16000
print(type(sampleRate))
enablePunctuationPrediction  = True
enableInverseTextNormalization = True
enableVoiceDetection  = False

# 设置RESTful请求参数
request = url + '?appkey=' + appKey
request = request + '&token=' + token
request = request + '&format=' + format
request = request + '&sample_rate=' + str(sampleRate)
print('Request: ' + request)
process(request, audioFile)
```

PHP示例

> **说明**
>
> PHP示例中使用了cURL函数，要求PHP版本在4.0.2以上，并且确保已安装cURL扩展。

```
<?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);
```

```
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 = "您的appkey";
$token = "您的token";
$url = "https://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/FlashRecognizer";
//$audioFile = "/path/to/nls-sample-16k.wav";
$audioFile = "./nls-sample-16k.wav";
$format = "wav";
$sampleRate = 16000;

/**
```

```
* 设置RESTful请求参数
*/
$request = $url;
$request = $request . "?appkey=" . $appkey;
$request = $request . "&token=" . $token;
$request = $request . "&format=" . $format;
$request = $request . "&sample_rate=" . strval($sampleRate);
print "Request: " . $request . "\n";
process($token, $request, $audioFile);
?>
```

2.移动端SDK说明

本文为您介绍传入录音文件，完成音频文件识别并返回结果的流程说明。

使用须知

- 输入格式：WAV/MP3/AAC。
- 时长限制：识别语音文件大小不能超过100 MB。
- 设置多语言识别：在管控台编辑项目中进行模型选择，详情请参见管理项目。

服务地址

访问类型	URL	说明
外网访问	https://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/FlashRecognizer	所有服务器均可使用外网访问URL。

交互流程

下图展示iOS SDK和Android SDK的交互流程。

1. 鉴权和初始化

   客户端与服务端建立连接时，使用Access Token进行鉴权。关于Token获取请参见获取Token。

   初始化参数表

参数	类型	是否必选	说明
url	String	是	语音服务URL地址。

参数	类型	是否必选	说明
app_key	String	是	管控台创建项目的appkey。
token	String	是	请确保该Token可以使用并在有效期内。 说明 Token可以在初始化时设置，也可通过参数设置进行更新。
device_id	String	是	设备标识，唯一表示一台设备（如Mac地址/SN/UniquePsuedoID）。

2. 开始文件识别

客户端发起文件识别请求时，需要通过接口传入JSON格式的params参数。

参数	类型	是否必选	说明
file_path	String	是	需要进行识别的文件绝对路径。
direct_ip	String	否	您可自行完成服务地址DNS解析，传入IP地址进行访问。
nls_config	JsonObject	否	服务细节参数配置。

其中，`nls_config`字段用于设置服务的具体配置参数，多与识别效果相关。

参数	类型	是否必选	说明
format	String	是	音频编码格式，支持WAV/MP3/AAC格式。

参数	类型	是否必选	说明
sample_rate	Integer	否	识别模型使用的采样率，一般不设置，在管控台进行配置。
enable_inverse_text_normalization	Boolean	否	是否打开ITN，中文数字将转为阿拉伯数字输出，默认值为 false，开启时需要设置version为”4.0”，enable_unify_post 必须为 true 。注意：不会对词信息进行ITN转换。
max_end_silence	Integer	否	允许的最大结束静音，默认值450，单位是毫秒。
customization_id	String	否	通过POP API创建的定制模型id，默认不添加。
vocabulary_id	String	否	创建的泛热词表ID，默认不添加。
enable_word_level_result	Boolean	否	是否开启词信息返回，默认false。
first_channel_only	Boolean	否	是否开启只处理第一个声道，默认false。

3. 获取识别结果

   启动任务后一段时间（根据文件长度和网络情况而定）将收到EVENT_FILE_TRANS_RESULT事件，即完整转写结果，示例如下。

```
{
    "task_id":"e76f979b33d9443eb6fbf770315a****",
    "status":20000000,
    "message":"SUCCESS",
    "flash_result":{
        "duration":299,  //音频时长
        "completed":true,
        "sentences":[
            {
                "text":"啊，",  //句子级别的识别结果
                "begin_time":3700,  //句子的开始时间，单位：毫秒
                "end_time":3940,  //句子的结束时间，单位：毫秒
                "channel_id":0,  //多个声道的音频文件会区分返回识别结果，声道id从0计数
                "words":[
                    {
                        "text":"啊",  //当前句子包含的词信息
                        "begin_time":3700,  //当前词开始时间，单位：毫秒
                        "end_time":3940,  //当前词结束时间，单位：毫秒
                        "punc":"，"  //当前词尾的标点信息，没有标点则为空
                    }
                ]
            },
            {
                "text":"我在哪呀了。",
                "begin_time":3940,
                "end_time":4720,
                "channel_id":0,
                "words":[
                    {
                        "text":"我",
                        "begin_time":3940,
                        "end_time":4040,
                        "punc":"，"
                    },
                    {
                        "text":"在哪",
                        "begin_time":4040,
                        "end_time":4340,
                        "punc":"，"
                    },
                    {
                        "text":"呀了",
                        "begin_time":4340,
                        "end_time":4720,
                        "punc":"，"
                    }
                ]
            }
        ]
    }
}
```

响应消息为JSON格式，包含如下字段。

字段	说明
task_id	任务ID，任务唯一标识。
status	状态码
message	状态消息
flash_result	识别结果对象

其中，`flash_result`对象包括如下字段。

字段	说明
sentences	句子级别的识别结果，是一个对象数组。

其中，`sentences`数组对象包含如下字段。

字段	说明
text	识别结果。
begin_time	识别结果的语音起点在音频流的时间偏移，单位毫秒。
end_time	识别结果的语音尾点在音频流的时间偏移，单位毫秒。

错误码

识别成功

错误码	错误信息	描述
0	SUCCESS	成功

配置或参数错误

错误码	错误消息	描述	解决方案
240999	DEFAULT_ERROR	内部默认错误。	内部未明确错误，可提工单通过日志文件寻求支持。
240001	NUI_CONFIG_INVALID	配置文件错误。	配置文件错误，请确认传入的资源路径内是否有资源文件。如果是Android平台，请参考代码样例主动使用copyAssets接口。
240002	ILLEGAL_PARAM	非法参数。	请确认传入的格式是否正确，包括字段类型，值范围限制。
240003	ILLEGAL_INIT_PARAM	初始化参数非法。	请确认初始化参数格式错误或缺少必须字段。
240004	NECESSARY_PARAM_LACK	缺少必须参数。	请确认接口调用时的必须参数。
240005	NULL_PARAM_ERROR	参数为空。	确认参数是否为空。
240006	NULL_LISTENER_ERROR	未定义事件回调。	确认回调事件是否正确赋值。
240007	NULL_DIALOG_ERROR	无有效对话实例，一般在内部状态错误时发生。	请确认接口调用前是否为正确状态，可使用cancel接口恢复idle状态。仍不能解决可提工单获得支持。
240008	NULL_ENGINE_ERROR	无有效引擎实例，请检查是否初始化成功。	请确认是否初始化成功。
240009	ILLEGAL_DATA	传入音频数据地址或长度非法。	请确认传入的数据长度值。

SDK状态错误

错误码	错误消息	描述	解决方案
240010	ILLEGAL_REENTRANT	退出后调用SDK接口。	不影响功能时可忽略。

错误码	错误消息	描述	解决方案
240011	SDK_NOT_INIT	SDK未正确初始化。	确认初始化返回值正确再进行其他接口使用。
240012	SDK_ALREADY_INIT	重复调用SDK初始化接口。	确认初始化调用逻辑。
240013	DIALOG_INVALID_STATE	内部对话状态错误。	请阅读SDK流程图，确认是否在错误状态下调用接口。
240014	STATE_INVALID	SDK内部状态错误。	同DIALOG_INVALID_STATE。
240015	ILLEGAL_FUNC_CALL	该模式无法调用接口。	请确认接口调用是否合理。

系统调用错误

错误码	错误信息	描述	解决方法
240020	MEM_ALLOC_ERROR	内存分配错误。	检查内存是否不足。
240021	FILE_ACCESS_FAIL	文件访问错误。	检查文件读写权限是否提供。
240022	CREATE_DIR_ERROR	创建目录错误。	检查是否有写权限。

SDK内部调用错误

错误码	错误消息	描述	解决方法
240030	CREATE_NUI_ERROR	引擎创建失败。	创建实例失败，一般为系统资源不足。
240031	TEXT_DIALOG_START_FAIL	发起文本理解失败。	文本转语义理解失败，检查网络连接或url以及token等信息是否有效。
240032	TEXT_CANCEL_START_FAIL	取消文本理解失败。	可忽略。
240033	WUW_DUPLICATE	动态唤醒词重复。	可忽略。

本地引擎调用错误

错误码	错误消息	描述	解决方法
240040	CEI_INIT_FAIL	本地引擎初始化失败。	请确认本地引擎的模型是否有效，目录是否可读写。可提交工单跟进。
240041	CEI_SET_PARAM_FAIL	引擎参数设置失败。	请提工单。
240042	CEI_COMPILE_GRAMMER_FAIL	语法编译失败。	可忽略。
240043	CEI_STOP_FAIL	停止识别失败。	请提工单。
240044	CEI_CANCEL_FAIL	取消识别失败。	请提工单。
240045	CEI_UNLOAD_KWS_FAIL	取消唤醒词失败。	请提工单。
240046	GET_WUW_ERROR	获取唤醒词失败。	请提工单。

音频错误

错误码	错误消息	描述	解决方法
240050	SELECT_RECORDER_ERROR	选择音频设备错误。	内部错误，请提工单。
240051	UPDATE_AUDIO_ERROR	推送音频错误，一般为输入音频长度大于所需音频。	确认推送的音频长度是否非法。
240052	MIC_ERROR	连续2s未获取到音频。	请确认在音频数据回调中是否正确提供所需长度的音频。

调用超时错误

错误码	错误消息	描述	解决方法
240080	ENGINE_INIT_TIMEOUT	初始化引擎超时。	如果偶现可忽略，否则请提工单。

错误码	错误消息	描述	解决方法
240081	SET_PARAM_TIMEOUT	设置参数超时。	如果偶现可忽略，否则请提工单。
240082	SET_WUW_TIMEOUT	设置唤醒词超时。	如果偶现可忽略，否则请提工单。
240083	SELECT_RECORDER_TIMEOUT	选择录音设备超时。	如果偶现可忽略，否则请提工单。
240084	STOP_TIMEOUT	结束对话超时。	如果偶现可忽略，否则请提工单。
240085	ASR_ENGINE_STOP_TIMEOUT	结束引擎超时。	如果偶现可忽略，否则请提工单。
240086	UNLOAD_DYNAMIC_WUW_TIMEOUT	取消动态唤醒词超时。	如果偶现可忽略，否则请提工单。
240087	ADD_DYNAMIC_WUW_TIMEOUT	增加动态唤醒词超时。	如果偶现可忽略，否则请提工单。
240100	WAIT_TIMEOUT	引擎接口调用超时。	如果偶现可忽略，否则请提工单。
240101	HANDLE_API_TIMEOUT	API层接口调用超时。	如果偶现可忽略，否则请提工单。

网络错误

错误码	错误消息	描述	解决方法
240060	CREATE_DA_REQUEST_ERROR	创建对话助手实例失败。	可忽略。
240061	START_DA_REQUEST_ERROR	发起对话助手请求失败。	可忽略。

错误码	错误消息	描述	解决方法
240062	DEFAULT_NLS_ERROR	服务端发生错误。说明该错误同时包含服务端返回错误内容，具体请参见服务端错误码。	请参考服务端返回码进一步定位。
240063	SSL_ERROR	创建ssl实例错误。	偶现请忽略，否则请提交工单。
240064	SSL_CONNECT_FAILED	ssl连接失败。	连接异常，请检查服务url或者本地网络连接是否正常。
240065	HTTP_CONNECT_FAILED	HTTP连接失败。	服务连接错误，可通过日志文件查看HTTP返回值确认原因，或提工单跟进。
240066	DNS_FAILED	DNS解析失败。	请检查本地网络是否正常，DNS服务是否正常。
240067	CONNECT_FAILED	socket连接失败。	检查网络连接。
240068	SERVER_NOT_ACCESS	服务端无法访问。	请检查token是否过期或者url是否正确，仍无法定位请提工单。
240069	SOCKET_CLOSED	socket已关闭。	偶现可忽略，否则提交工单跟进。
240070	AUTH_FAILED	鉴权失败。	请检查是否提供正确的ak_serect，ak_id，app_key，sdk_code，device_id等信息，以及确认是否开通足够配额。
240071	HTTPDNS_FAILED	使用客户端传入的IP连接失败。	如果使用直接传入IP进行访问，请确认IP是否可访问。
240072	HTTP_SEND_FAILED	文件转写HTTP发送失败。	确认网络连接是否正常。
240073	HTTP_RECEIVE_FAILED	文件转写HTTP接收失败。	确认网络连接是否正常。

错误码	错误消息	描述	解决方法
240074	HTTP_RESPONSE_ERROR	文件转写接收内容解析失败。	服务端返回内容错误，请提工单跟进。
240075	HTTP_SERVER_ERROR	文件转写服务错误，详细错误请参考服务错误码。	请参考服务端错误码进一步确认原因。

网络超时错误

错误码	错误消息	描述	解决方法
240090	UPDATE_CONTEXT_TIMEOUT	更新客户端信息超时	偶现请忽略，否则提交工单。
240091	CONNECTION_TIMEOUT	网络连接超时	偶现请忽略，否则提交工单。
240092	PARTIAL_ASR_TIMEOUT	获取中间识别结果超时	偶现请忽略，否则提交工单。
240093	ASR_TIMEOUT	获取最终识别结果超时	偶现请忽略，否则提交工单。
240094	DIALOG_TIMEOUT	获取对话理解结果超时	偶现请忽略，否则提交工单。
240095	WWV_TIMEOUT	获取云端唤醒确认结果超时	偶现请忽略，否则提交工单。

服务端错误码

当收到EVENT_ASR_ERROR事件，并且错误码为DEFAULT_NLS_ERROR（240062）或HTTP_SERVER_ERROR（240075）时，可以通过错误事件header中status字段获取服务端错误码进行进一步问题定位。

错误码	原因	解决方法
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。
403	令牌过期或无效的参数	1. 检查使用的令牌是否过期。 2. 检查参数值设置是否合理。

错误码	原因	解决方法
40000004	空闲超时。	确认是否长时间（10秒）未发送数据到服务端。
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。如果超过并发数，建议从免费版升级到商用版，或者商用版扩容并发资源。
40000000	默认的客户端错误码。	查看错误消息或提交工单。
41010120	客户端超时错误。	客户端连续10秒及以上没有发送数据，导致客户端超时错误。
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	内部调用错误。	如果偶现可以忽略，重复出现请提交工单。
52010001	内部调用错误。	如果偶现可以忽略，重复出现请提交工单。
40010001	不支持的接口。	使用了不支持的接口，如果使用SDK请提交工单。
40010002	不支持的指令。	使用了不支持的指令，如果使用SDK请提交工单。
40010003	无效的指令。	指令格式错误，如果使用SDK请提交工单。
40010004	客户端提前断开连接。	检查是否在请求正常完成之前关闭了连接。
40010005	任务状态错误。	发送了当前任务状态不能处理的指令。
40020105	应用不存在。	解析路由时找不到应用。
40020106	appkey和token不匹配。	检查应用appkey是否正确，是否与令牌归属同一个账号。

错误码	原因	解决方法
40020503	子账户鉴权失败。	使用父账户对调用的子账户授权POP API的访问权限。
41040201	客户端10s内停止发送数据。	检查网络问题，或者检查业务中是否存在不发数据的情况。
41040202	客户端发送数据过快，服务器资源已经耗尽。	检测客户端发包是否过快，是否按照1:1的实时率发包。
41040203	客户端发送音频格式不正确。	请将音频数据的格式转换为SDK目前支持的音频格式。
41040204	客户端调用方法异常。	客户端应该先调用发送请求接口，发送请求完毕后再调用其他接口。
41040205	客户端设置MAXSILENCE_PARAM方法异常。	参数MAXSILENCE_PARAM的范围为200～2000。
41050008	采样率不匹配。	检查调用时设置的采样率和管控台上appkey绑定的ASR模型采样率是否一致。
51040101	服务端内部错误。	未知错误。
51040103	实时语音识别服务不可用	检查实时语音识别服务是否有任务堆积等导致任务提交失败
51040104	请求实时语音识别服务超时。	排查实时语音识别日志。
51040105	调用实时语音识别服务失败。	检查实时语音识别服务是否启动，端口是否正常开启。
51040106	实时语音识别服务负载均衡失败，未获取到实时语音识别服务的IP地址。	检查VPC中的实时语音识别服务机器是否有异常。

3.iOS SDK

本文介绍了如何使用阿里云智能语音服务提供的iOS SDK，包括SDK下载安装、关键接口及代码示例。

前提条件

- 使用SDK前，首先阅读接口说明，详情请参见接口说明。
- 准备好项目Appkey，详情请参见创建项目。
- 已获取Access Token，详情请参见获取Token。

下载安装

1. 下载SDK和示例代码。

   > **说明**
   >
   > 请下载后在样例初始化代码中替换您的阿里云账号信息、Appkey和Token才可运行。为方便集成，2.5.14版本后iOS接口使用纯Object-C接口，不再使用C++混合接口。

2. 解压ZIP包，将zip包中的nuisdk.framework添加到您的工程中，并在工程Build Phases的Link Binary With Libraries中添加nuisdk.framework。请确保在编译配置的General > Frameworks, Libraries, and Embedded Content中配置nuisdk.framework为Embed & Sign。
3. 使用Xcode打开此工程，工程中提供了参考代码以及一些直接可使用的工具类，例如音频播放录制和文件操作，您可以直接复制源码到您的实际工程进行使用。其中录音文件识别极速版示例代码在FileTranscriberViewController类中。

SDK关键接口

- **nui_initialize**：初始化SDK。

```
/**
     * 初始化SDK，SDK为单例，请先释放后再次进行初始化。请勿在UI线程调用，可能引起阻塞。
     * @param parameters: 初始化参数，参见接口说明文档
     * @param level: log打印级别，值越小打印越多
     * @param save_log: 是否保存log为文件，存储目录为parameter中的debug_path字段值
     * @return 参见错误码
     */
-(NuiResultCode) nui_initialize:(const char *)parameters
                  logLevel:(NuiSdkLogLevel)level
                  saveLog:(BOOL)save_log;
```

- **nui_set_params**：以JSON格式设置SDK参数。

```
/**
     * 以JSON格式设置参数
     * @param params: 参数信息请参见接口说明文档
     * @return 参见错误码
     */
   -(NuiResultCode) nui_set_params:(const char *)params;
```

- **nui_file_trans_start**：发起文件识别请求。

```
/**
     * 开始识别
     * @param params：设置识别参数，参考接口说明。
     * @param task_id：开始转写的任务ID，SDK生成随机字符串。
     * @return：参见错误码。
     */
NuiResultCode nui_file_trans_start(const char *params, char *task_id);
```

- **nui_file_trans_cancel**：取消正在工作的识别任务。

```
/**
     * 结束识别
   * @param task_id：需要结束的转写任务ID。
     * @return：参见错误码
     */
NuiResultCode nui_file_trans_cancel(const char *task_id);
```

- **nui_release**：释放SDK。

```
/**
     * 释放SDK资源
     * @return 参见错误码
     */
    -(NuiResultCode) nui_release;
```

- **NeoNuiSdkDelegate** 事件代理

  onFileTransEventCallback：SDK转写事件回调。

```
/**
     * SDK主要事件回调
     * @param nuiEvent: 回调事件，参见如下事件列表
     * @param asrResult: 语音识别结果
     * @param taskId: 一个任务对应的唯一id
     * @param ifFinish: 本轮识别是否结束标志
     * @param retCode: 参见错误码，在出现EVENT_ASR_ERROR事件时有效
     */
    -(void) onFileTransEventCallback:(NuiCallbackEvent)nuiEvent
                                asrResult:(const char *)asr_result
                                taskId:(const char *)task_id
                                ifFinish:(BOOL)finish
                                retCode:(int)code;
```

  NuiCallbackEvent事件列表：

名称	说明
EVENT_FILE_TRANS_CONNECTED	连接文件转写服务成功
EVENT_FILE_TRANS_UPLOADED	上传文件成功

名称	说明
EVENT_FILE_TRANS_RESULT	识别最终结果
EVENT_ASR_ERROR	根据错误码信息判断出错原因

调用步骤

1. 初始化SDK。
2. 根据业务需求设置参数。
3. 调用nui_file_trans_start开始识别。
4. 在EVENT_FILE_TRANS_RESULT事件中获取最终识别结果。
5. 结束调用，使用release接口释放SDK资源。

代码示例

> 说明
> 接口默认采用get_instance方式获得单例，您如果有多例需求，也可以直接alloc对象进行使用。

NUI SDK初始化

```
NSString * initParam = [self genInitParams];
    [_nui nui_initialize:[initParam UTF8String] logLevel:LOG_LEVEL_VERBOSE saveLog:save_log
];
```

其中，genInitParams生成为String JSON字符串，包含资源目录和用户信息。主要包含如下字段。

```
    [dictM setObject:id_string forKey:@"device_id"];
    [dictM setObject:@"" forKey:@"url"];
    [dictM setObject:@"" forKey:@"app_key"];
    [dictM setObject:@"" forKey:@"token"];
```

开始识别

调用nui_file_trans_start接口开启识别。

```
char task_id[33] = {0};
[_nui nui_file_trans_start:param:[param_string UTF8String] taskId:task_id];
```

回调处理

onFileTransEventCallback：文件识别事件回调，请勿在事件回调中调用SDK的接口，可能引起死锁。

```
-(void)onFileTransEventCallback:(NuiCallbackEvent)nuiEvent
                asrResult:(const char *)asr_result
                taskId: (const char *)task_id
                 ifFinish:(bool)finish
                  retCode:(int)code {
    TLog(@"onNuiEventCallback event %d finish %d", nuiEvent, finish);
    if (nuiEvent == EVENT_FILE_TRANS_RESULT) {
     TLog(@"trans finish");
    } else if (nuiEvent == EVENT_ASR_ERROR) {
        TLog(@"EVENT_ASR_ERROR error[%d]", code);
    }
    if (finish) {
        [myself showStart];
    }
    return;
}
```

取消识别

```
[_nui nui_file_trans_cancel:[task_id UTF8String]];
```

4.Android SDK

本文为您介绍如何使用阿里云智能语音服务提供的Android SDK，包括SDK下载安装、关键接口及代码示例。

前提条件

- 使用SDK前，首先阅读接口说明，详情请参见接口说明。
- 准备好项目Appkey，详情请参见创建项目。
- 已获取Access Token，详情请参见获取Token。

下载安装

1. 下载SDK和示例代码。
2. 解压ZIP包，在 `app/libs` 目录下获取AAR格式的SDK包，将AAR包集成到您的工程项目中进行依赖。
3. 使用Android Studio打开此工程查看参考代码实现，其中示例代码为FileTranscriberActivity.java文件，替换appkey和token后可直接运行。

SDK关键接口

- initialize：初始化SDK。

```
/**
     * 初始化SDK，SDK为单例，请先释放后再次进行初始化。请勿在UI线程调用，可能会引起阻塞。
     * @param callback：事件监听回调，参见下文回调说明。
     * @param parameters：初始化参数，参见接口说明。
     * @param level：log打印级别，值越小打印越多。
     * @param save_log：是否保存log为文件，存储目录为parameter中的debug_path字段值。
     * @return：参见错误码。
     */
public synchronized int initialize(final INativeFileTransCallback callback,
                                   String parameters,
                                   final Constants.LogLevel level,
                                   final boolean save_log)
```

其中，INativeFileTransCallback类型需要实现的回调是onFileTransEventCallback。

onFileTransEventCallback：文件识别事件回调。

```
/**
     * SDK主要事件回调
     * @param event：回调事件，参见如下事件列表。
     * @param resultCode：参见错误码，在出现EVENT_ASR_ERROR事件时有效。
     * @param arg2：保留参数。
     * @param asrResult：语音识别结果。
     * @param taskId：任务ID。
     */
    void onFileTransEventCallback(NuiEvent event, final int resultCode, final int arg2, A
srResult asrResult, String taskId);
```

事件列表：

名称	说明
EVENT_FILE_TRANS_CONNECTED	连接文件识别服务成功
EVENT_FILE_TRANS_UPLOADED	上传文件成功
EVENT_FILE_TRANS_RESULT	识别最终结果
EVENT_ASR_ERROR	根据错误码信息判断出错原因

- set_params：以JSON格式设置SDK参数。

```
/**
     * 以JSON格式设置参数
     * @param params：参见接口说明
     * @return：参见错误码
     */
    public synchronized int setParams(String params)
```

- startFileTranscriber：开始文件识别。

```
/**
     * 开始识别
     * @param params：识别参数，参见接口说明。
     * @param taskId：开始识别的任务ID，SDK会生成随机字符串。
     * @return：参见错误码
     */
    public synchronized int startFileTranscriber(String params, byte[] taskId)
```

- stopFileTranscriber：结束识别。

```
/**
     * 结束识别
     * @return：参见错误码
     */
    public synchronized int stopFileTranscriber(String taskId)
```

- release：释放SDK。

```
/**
     * 释放SDK资源
     * @return：参见错误码
     */
    public synchronized int release()
```

调用步骤

1. 初始化SDK。

2. 根据业务需求设置参数。
3. 调用startFileTranscriber开始识别。
4. 在EVENT_FILE_TRANS_RESULT事件中获取最终识别结果。
5. 结束调用，使用release接口释放SDK资源。

Proguard配置

如果代码使用了混淆，请在proguard-rules.pro中配置：

```
-keep class com.alibaba.idst.nui.*{*;}
```

代码示例

> 说明
> 接口默认采用GetInstance获得单例，您如果有多例需求，也可以直接new对象进行使用。

NUI SDK初始化

```
CommonUtils.copyAssetsData(this);
int ret = NativeNui.GetInstance().initialize(this, genInitParams(path,path2), Constants.Log
Level.LOG_LEVEL_VERBOSE, true);
```

其中，`genInitParams`生成为String JSON字符串，包含资源目录和用户信息。其中用户信息包含如下字段。

```
private String genInitParams(String workpath, String debugpath) {
        String str = "";
        try{
            JSONObject object;
            object.put("app_key","");
            object.put("token","");
            object.put("device_id",Utils.getDeviceId());
            object.put("url","");
            object.put("workspace", workpath);
            object.put("debug_path",debugpath);
            str = object.toString();
        } catch (JSONException e) {
            e.printStackTrace();
        }
  return str;
}
```

开始识别

调用startFileTranscriber方法开启识别。

```
byte[] task_id = new byte[32];
NativeNui.GetInstance().startFileTranscriber(genDialogParams(), taskId);
```

回调处理

onNuiEventCallback：NUI SDK事件回调，请勿在事件回调中调用SDK的接口，可能引起死锁。

```
public void onFileTransEventCallback(Constants.NuiEvent event, final int resultCode, final
int arg2, AsrResult asrResult, String taskId) {
        Log.i(TAG, "event=" + event);
            if (event == Constants.NuiEvent.EVENT_FILE_TRANS_RESULT) {
            showText(asrView, asrResult.asrResult);
        } else if (event == Constants.NuiEvent.EVENT_ASR_ERROR) {
            ;
        }
    }
```

取消识别

```
NativeNui.GetInstance().cancelDialog(taskId);
```