ALIBABA CLOUD

阿里云

智能语音交互
录音文件识别

文档版本：20210115

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

录音文件识别是针对已经录制完成的录音文件，进行识别的服务。录音文件识别是非实时的，识别的文件需要提交基于HTTP可访问的URL地址，不支持提交本地文件。

使用须知

- 支持单轨/双轨的WAV格式、MP3格式录音文件识别。
- 支持调用方式：轮询方式/回调方式。
- 支持语言模型定制。更多信息，请参见语言模型定制。
- 支持热词。更多信息，请参见热词。
- 支持汉语普通话、方言、欧美英语等多种模型识别。

调用限制

- 录音文件访问权限需要设置为公开，URL中只能使用域名不能使用IP地址、不可包含空格，请尽量避免使用中文。

可用URL	不可用URL
https://aliyun-nls.oss-cn-hangzhou.aliyuncs.com/asr/fileASR/examples/nls-sample-16k.wav	○ http://127.0.0.1/sample.wav ○ D:\files\sample.wav

- 文件大小需控制在512 MB以下。
- 提交录音文件识别请求后，免费用户的识别任务在24小时内完成并返回识别文本； 付费用户的识别任务在6小时内完成并返回识别文本。识别结果在服务端可保存72小时。

> **说明**
>
> 一次性上传大规模数据（半小时内上传超过500小时时长的录音）的除外。有大规模数据识别需求的用户，请联系售前专家。

- 免费用户每日可识别不超过2小时时长的录音文件。

使用步骤

i. 了解您的录音文件格式和采样率，根据业务场景在管控台选择合适的场景模型。

ii. 将录音文件存放至OSS。

如果文件访问权限为公开，可参见公共读Object，获取文件访问链接；如果文件访问权限为私有，可参见私有Object，通过SDK生成有有效时间的访问链接。

> **说明**
>
> 您也可以把录音文件存放在自行搭建的文件服务器，提供文件下载。请保证HTTP的响应头（Header）中 `Content-Length` 的长度值和Body中数据的真实长度一致，否则会导致下载失败。

iii. 客户端提交录音文件识别请求。

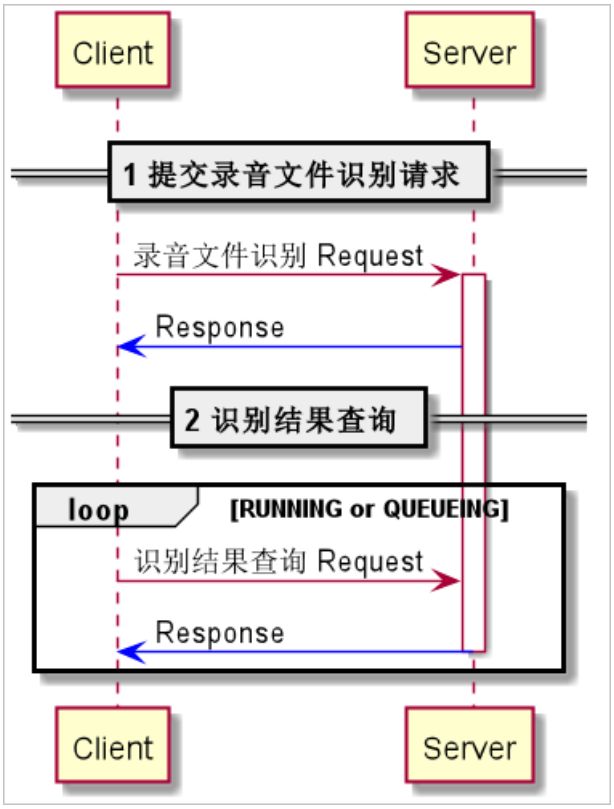
正常情况下，服务端返回该请求任务的ID，用以查询识别结果。

iv. 客户端发送识别结果查询请求。

通过步骤3获取的请求任务ID查询录音文件识别的结果，目前识别的结果在服务端可保存72小时。

交互流程

客户端与服务端的交互流程如图所示。



> 说明
>
> 所有服务端的响应都会在返回信息的header包含表示本次识别任务的TaskId参数，请记录该值。如果发生错误，请将该值和错误信息提交到工单。

接口调用方式

录音文件识别服务是以RPC风格的POP API方式提供录音文件识别接口，将参数封装到每一个请求中，每个请求即对应一个方法，执行的结果放在response中。需要识别的录音文件必须存放在某服务上（推荐阿里云OSS），可以通过URL访问。

录音文件识别POP API包括两部分：POST方式的“录音文件识别请求调用接口”（用户级别QPS（queries per second）限制为200）、GET方式的“录音文件识别结果查询接口”（用户级别QPS限制为500）。

- 识别请求调用接口：
  - 当采用轮询方式时，提交录音文件识别任务，获取任务ID，供后续轮询使用。
  - 当采用回调方式时，提交录音文件识别任务和回调URL，任务完成后会把识别结果POST到回调地址，要求回调地址可接收POST请求。

> **说明**
>
> 由于历史原因，早期发布的录音文件识别服务（默认为2.0版本）的回调方式和轮询方式的识别结果在JSON字符串的风格和字段上均有不同，下文将作说明。录音文件识别服务在4.0版本对回调方式做了优化，使得回调方式的识别结果与轮询方式的识别结果保持一致，均为驼峰风格的JSON格式字符串。
>
> 如果您已接入录音文件识别服务，即没有设置录音文件识别服务的版本，默认为2.0版，可以继续使用；如果您新接入录音文件识别服务，请设置服务版本为4.0。

输入参数及说明：

提交录音文件识别请求时，需要设置输入参数，以JSON格式的字符串传入请求对象的Body，JSON格式如下：

```
{
  "appkey": "your-appkey",
  "file_link": "https://aliyun-nls.oss-cn-hangzhou.aliyuncs.com/asr/fileASR/examples/nls-sample-16k.wav",
  "auto_split":false,
  "version": "4.0",
  "enable_words": false,
  "enable_sample_rate_adaptive": true,
  // valid_times：获取语音指定时间段的识别内容，若不需要，则无需填写。
  "valid_times": [
    {
      "begin_time": 200,
      "end_time":2000,
      "channel_id": 0
    }
  ]
}
```

参数	值类型	是否必选	说明
appkey	String	是	管控台的项目appkey。
file_link	String	是	存放录音文件的地址，需要在管控台中将对应项目的模型设置为支持该音频场景的模型。

参数	值类型	是否必选	说明
version	String	是	设置录音文件识别服务的版本，请设置为“4.0”，默认为“2.0”。
enable_words	Boolean	否	是否开启返回词信息，默认为false，开启时需要设置version为“4.0”。
enable_sample_rate_adaptive	Boolean	否	是否将大于16kHz采样率的音频进行自动降采样，默认为false，开启时需要设置version为“4.0”。
enable_callback	Boolean	否	是否启用回调功能，默认值为false。
callback_url	String	否	回调服务的地址，enable_callback取值为true时，本字段必选。URL支持HTTP和HTTPS协议，host不可使用IP地址。
auto_split	Boolean	否	是否开启智能分轨（开启智能分轨，即可在两方对话的语音情景下，依据每句话识别结果中的ChannelId，判断该句话的发言人为哪一方。通常先发言一方ChanelId为0。只支持8000Hz采样率单通道语音。）。
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”。

参数	值类型	是否必选	说明
enable_disfluency	Boolean	否	是否打开声音顺滑，默认值false，开启时需要设置version为“4.0”。
enable_punctuation_prediction	Boolean	否	是否给句子加标点。默认值true（加标点）。
valid_times	List< ValidTime >	否	有效时间段信息，用来排除一些不需要的时间段。
max_end_silence	Integer	否	允许的最大结束静音，默认值450，单位是毫秒。
max_single_segment_time	Integer	否	允许单句话最大结束时间，最小值10000，默认值20000。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。
class_vocabulary_id	String	否	创建的类热词表ID，默认不添加。
vocabulary_id	String	否	创建的泛热词表ID，默认不添加。

其中，ValidTime对象参数说明如下表所示。

参数	值类型	是否必选	说明
begin_time	Int	是	有效时间段的起始点时间偏移（单位：ms）。
end_time	Int	是	有效时间段的结束点时间偏移（单位：ms）。

参数	值类型	是否必选	说明
channel_id	Int	是	有效时间段的作用音轨序号（从0开始）。

输出参数及说明：

服务端返回录音文件识别请求的响应，响应的输出参数为JSON格式的字符串：

```
{
    "TaskId": "4b56f0c4b7e611e88f34c33c2a60****",
    "RequestId": "E4B183CC-6CFE-411E-A547-D877F7BD****",
    "StatusText": "SUCCESS",
    "StatusCode": 21050000
}
```

返回HTTP状态：200表示成功，更多状态码请查阅HTTP状态码。

属性	值类型	是否必选	说明
TaskId	String	是	识别任务ID。
RequestId	String	是	请求ID，仅用于联调。
StatusCode	Int	是	状态码。
StatusText	String	是	状态说明。

- 识别结果查询接口：

  提交完录音文件识别请求后，按照如下参数设置轮询识别结果。

  输入参数：

  通过提交录音文件识别请求获得的任务ID作为识别结果查询接口参数，获取识别结果。在接口调用过程中，需要设置一定的查询时间间隔。

  > **注意**
  >
  > 查询接口有QPS限制（100QPS），超过QPS之后则可能报错：`Throttling.User : Request was denied due to user flow control.`。建议查询接口的轮询间隔时长适当延长，不宜过短。

属性	值类型	是否必选	说明
TaskId	String	是	识别任务ID。

输出参数及说明：

服务端返回识别结果查询请求的响应，响应的输出参数为JSON格式的字符串。

- 正常返回：以录音文件nls-sample-16k.wav（文件为单轨）识别结果为例。

```
{
    "TaskId": "d429dd7dd75711e89305ab6170fe****",
    "RequestId": "9240D669-6485-4DCC-896A-F8B31F94****",
    "StatusText": "SUCCESS",
    "BizDuration": 2956,
    "SolveTime": 1540363288472,
    "StatusCode": 21050000,
    "Result": {
        "Sentences": [{
            "EndTime": 2365,
            "SilenceDuration": 0,
            "BeginTime": 340,
            "Text": "北京的天气。",
            "ChannelId": 0,
            "SpeechRate": 177,
            "EmotionValue": 5.0
        }]
    }
}
```

如果开启enable_callback/callback_url，且设置服务版本为4.0，回调识别结果为：

```
{
    "Result": {
        "Sentences": [{
            "EndTime": 2365,
            "SilenceDuration": 0,
            "BeginTime": 340,
            "Text": "北京的天气。",
            "ChannelId": 0,
            "SpeechRate": 177,
            "EmotionValue": 5.0
        }]
    },
    "TaskId": "36d01b244ad811e9952db7bb7ed2****",
    "StatusCode": 21050000,
    "StatusText": "SUCCESS",
    "RequestTime": 1553062810452,
    "SolveTime": 1553062810831,
    "BizDuration": 2956
}
```


> **说明**
>
> - RequestTime为时间戳（单位为毫秒，例如1553062810452转换为北京时间为2019/3/20 14:20:10），表示录音文件识别提交请求的时间。
> - SolveTime为时间戳（单位为毫秒），表示录音文件识别完成的时间。

- 排队中：


```
{
    "TaskId": "c7274235b7e611e88f34c33c2a60****",
    "RequestId": "981AD922-0655-46B0-8C6A-5C836822****",
    "StatusText": "QUEUEING",
    "StatusCode": 21050002
}
```

- 识别中：

```
{
    "TaskId": "c7274235b7e611e88f34c33c2a60****",
    "RequestId": "8E908ED2-867F-457E-82BF-4756194A****",
    "StatusText": "RUNNING",
    "BizDuration": 0,
    "StatusCode": 21050001
}
```

- 异常返回：以文件下载失败为例。

```
{
    "TaskId": "4cf25b7eb7e711e88f34c33c2a60****",
    "RequestId": "098BF27C-4CBA-45FF-BD11-3F532F26****",
    "StatusText": "FILE_DOWNLOAD_FAILED",
    "BizDuration": 0,
    "SolveTime": 1536906469146,
    "StatusCode": 41050002
}
```

> **说明**
>
> 更多异常情况请查看下面的服务状态码的错误状态码及解决方案。

返回HTTP状态：200表示成功，更多状态码请查阅HTTP状态码。

属性	值类型	是否必选	说明
TaskId	String	是	识别任务ID。
StatusCode	Int	是	状态码。
StatusText	String	是	状态说明。
RequestId	String	是	请求id，用于调试。
Result	Object	是	识别结果对象。

属性	值类型	是否必选	说明
Sentences	List< SentenceResult >	是	识别的结果数据。当StatuxText为SUCCEED时存在。
Words	List< WordResult >	否	词信息，获取时需设置enable_words为true，且设置服务version为”4.0”。
BizDuration	Long	是	识别的音频文件总时长，单位为毫秒。
SolveTime	Long	是	时间戳，单位为毫秒，录音文件识别完成的时间。

其中，单句结果SentenceResult参数如下。

属性	值类型	是否必选	说明
ChannelId	Int	是	该句所属音轨ID。
BeginTime	Int	是	该句的起始时间偏移，单位为毫秒。
EndTime	Int	是	该句的结束时间偏移，单位为毫秒。
Text	String	是	该句的识别文本结果。
EmotionValue	Int	是	情绪能量值，取值为音量分贝值/10。取值范围：[1,10]。值越高情绪越强烈。
SilenceDuration	Int	是	本句与上一句之间的静音时长，单位为秒。

属性	值类型	是否必选	说明
SpeechRate	Int	是	本句的平均语速，单位：字数/分钟。

- 开启返回词信息：

  如果enable_words设置为true，且设置服务version为"4.0"，服务端的识别结果将包含词信息。使用轮询方式和回调方式获得的词信息相同，以轮询方式的识别结果为例：

```
{
    "StatusCode": 21050000,
    "Result": {
        "Sentences": [{
            "SilenceDuration": 0,
            "EmotionValue": 5.0,
            "ChannelId": 0,
            "Text": "北京的天气。",
            "BeginTime": 340,
            "EndTime": 2365,
            "SpeechRate": 177
        }],
        "Words": [{
            "ChannelId": 0,
            "Word": "北京",
            "BeginTime": 640,
            "EndTime": 940
        },{
            "ChannelId": 0,
            "Word": "的",
            "BeginTime": 940,
            "EndTime": 1120
        },{
            "ChannelId": 0,
            "Word": "天气",
            "BeginTime": 1120,
            "EndTime": 2020
        }]
    },
    "SolveTime": 1553236968873,
    "StatusText": "SUCCESS",
    "RequestId": "027B126B-4AC8-4C98-9FEC-A031158F****",
    "TaskId": "b505e78c4c6d11e9a213e11db149****",
    "BizDuration": 2956
}
```


Word对象说明：

属性	值类型	是否必选	说明
BeginTime	Int	是	词开始时间，单位毫秒。
EndTime	Int	是	词结束时间，单位毫秒。
ChannelId	Int	是	该词所属音轨ID。
Word	String	是	词信息。

服务状态码

正常状态码：

状态码	状态描述	状态含义	解决方案
21050000	SUCCESS	成功	POST方式的识别请求接口调用成功，或者GET方式的识别结果查询接口调用成功。
21050001	RUNNING	录音文件识别任务运行中	请稍后再发送GET方式的识别结果查询请求。
21050002	QUEUEING	录音文件识别任务排队中	请稍后再发送GET方式的识别结果查询请求。
21050003	SUCCESS_WITH_NO_VALID_FRAGMENT	识别结果查询接口调用成功，但是没有识别到语音	检查录音文件是否有语音，或者语音时长太短。

错误状态码：

> 说明
>
> 状态码以“4”开头表示客户端错误，以“5”开头表示服务端错误。

状态码	描述	含义	解决方案
41050001	USER_BIZDURATION_QUOTA_EXCEED	单日时间超限	如业务量较大，请联系商务洽谈，邮件地址：nls_support@service.aliyun.com。
41050002	FILE_DOWNLOAD_FAILED	文件下载失败	检查录音文件路径是否正确，是否可以外网访问和下载。
41050003	FILE_CHECK_FAILED	文件格式错误	检查录音文件是否是单轨/双轨的WAV格式、MP3格式。
41050004	FILE_TOO_LARGE	文件过大	检查录音文件大小是否超过512 MB。
41050005	FILE_NORMALIZE_FAILED	文件归一化失败	检查录音文件是否有损坏，是否可以正常播放。
41050006	FILE_PARSE_FAILED	文件解析失败	检查录音文件是否有损坏，是否可以正常播放。
41050007	MKV_PARSE_FAILED	MKV解析失败	检查录音文件是否有损坏，是否可以正常播放。
41050008	UNSUPPORTED_SAMPLE_RATE	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。
41050009	UNSUPPORTED_ASR_GROUP	ASR分组不支持	确认ak和appkey是否一致。
41050010	FILE_TRANS_TASK_EXPIRED	录音文件识别任务过期	TaskId不存在，或者已过期。
41050011	REQUEST_INVALID_FILE_URL_VALUE	请求file_link参数非法	确认file_link参数格式是否正确。

状态码	描述	含义	解决方案
41050012	REQUEST_INVALID_CALLBACK_VALUE	请求callback_url参数非法	确认callback_url参数格式是否正确，是否为空。
41050013	REQUEST_PARAMETER_INVALID	请求参数无效	确认请求task值为有效JSON格式字符串。
41050014	REQUEST_EMPTY_APPKEY_VALUE	请求参数appkey值为空	确认是否设置了appkey参数值。
41050015	REQUEST_APPKEY_UNREGISTERED	请求参数appkey未注册	确认请求参数appkey值是否设置正确，或者是否与阿里云账号的AccessKey ID同一个账号。
41050021	RAM_CHECK_FAILED	RAM检查失败	检查您的RAM用户是否已经授权调用语音服务的API，请参见RAM用户鉴权配置。
41050023	CONTENT_LENGTH_CHECK_FAILED	content-length 检查失败	检查下载文件时，HTTP response中的content-length与文件实际大小是否一致。
41050024	FILE_404_NOT_FOUND	需要下载的文件不存在	检查需要下载的文件是否存在。
41050025	FILE_403_FORBIDDEN	没有权限下载需要的文件	检查是否有权限下载录音文件。
41050026	FILE_SERVER_ERROR	请求的文件所在的服务不可用	检查请求的文件所在的服务是否可用。
50000000	INTERNAL_ERROR	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。
51050000	INTERNAL_ERROR	内部通用错误	如果偶现可以忽略，重复出现请提交工单。

状态码	描述	含义	解决方案
51050001	VAD_FAILED	VAD失败	如果偶现可以忽略，重复出现请提交工单。
51050002	RECOGNIZE_FAILED	内部alisr识别失败	如果偶现可以忽略，重复出现请提交工单。
51050003	RECOGNIZE_INTERRUPT	内部alisr识别中断	如果偶现可以忽略，重复出现请提交工单。
51050004	OFFER_INTERRUPT	内部写入队列中断	如果偶现可以忽略，重复出现请提交工单。
51050005	FILE_TRANS_TIMEOUT	内部整体超时失败	如果偶现可以忽略，重复出现请提交工单。
51050006	FRAGMENT_FAILED	内部分断失败	如果偶现可以忽略，重复出现请提交工单。

历史版本说明

如果您已接入录音文件识别，即没有设置服务版本为4.0，将默认使用录音文件识别2.0版本。回调方式的识别结果与轮询方式的识别结果在JSON字符串的风格和字段上有所不同， 如果开启enable_callback/callback_url，回调识别结果为：

```
{
    "result": [{
        "begin_time": 340,
        "channel_id": 0,
        "emotion_value": 5.0,
        "end_time": 2365,
        "silence_duration": 0,
        "speech_rate": 177,
        "text": "北京的天气。"
    }],
    "task_id": "3f5d4c0c399511e98dc025f34473****",
    "status_code": 21050000,
    "status_text": "SUCCESS",
    "request_time": 1551164878830,
    "solve_time": 1551164879230,
    "biz_duration": 2956
}
```

2.Java SDK

本文介绍如何使用阿里云智能语音服务提供的Java SDK，包括SDK的安装方法及SDK代码示例。

前提条件

- 使用SDK前，请先阅读接口说明，详情请参见接口说明。
- 本文中的SDK只适用于2.0版语音服务，如果您当前为1.0版需要首先开通2.0版服务，详情请参见开通服务。

SDK说明

录音文件识别的Java示例使用了阿里云Java SDK的CommonRequest提交录音文件识别请求和识别结果查询，采用的是RPC风格的POP API调用。阿里云Java SDK CommonRequest的使用方法请参见使用CommonRequest进行调用。

> **注意**
>
> 阿里云Java SDK不支持Android开发。

Java依赖

您只需依赖阿里云Java SDK的核心库与阿里开源库fastjson。阿里云Java SDK的核心库版本支持3.5.0及以上（如果版本在4.0.0及以上，需要根据错误提示补充对应的第三方依赖）。

```
<dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-core</artifactId>
  <version>3.7.1</version>
</dependency>
<dependency>
  <groupId>com.alibaba</groupId>
  <artifactId>fastjson</artifactId>
  <version>1.2.49</version>
</dependency>
```

示例说明

> **说明**
>
> 下载nls-sample-16k.wav。示例中使用的WAV录音文件为PCM编码格式16000Hz采样率，模型设置为通用模型。

鉴权

通过传入阿里云账号的AccessKey ID和AccessKey Secret（获取方式请参见开通服务），调用阿里云Java SDK得到client，示例如下：

```
final String accessKeyId = "您的AccessKey Id";
final String accessKeySecret = "您的AccessKey Secret";
/**
 * 地域ID
 */
final String regionId = "cn-shanghai";
final String endpointName = "cn-shanghai";
final String product = "nls-filetrans";
final String domain = "filetrans.cn-shanghai.aliyuncs.com";

IAcsClient client;
// 设置endpoint
DefaultProfile.addEndpoint(endpointName, regionId, product, domain);
// 创建DefaultAcsClient实例并初始化
DefaultProfile profile = DefaultProfile.getProfile(regionId, accessKeyId, accessKeySecret);
client = new DefaultAcsClient(profile);
```

录音文件识别请求调用接口

Java 示例采用轮询的方式，提交录音文件识别请求，获取任务ID，供后续轮询使用。

> **说明**
>
> 参数设置请参见接口说明，只需设置JSON字符串中的参数，其他方法的参数值保持不变。

```
/**
 * 创建CommonRequest 设置请求参数
 */
CommonRequest postRequest = new CommonRequest();
postRequest.setDomain("filetrans.cn-shanghai.aliyuncs.com"); // 设置域名，固定值。
postRequest.setVersion("2018-08-17");      // 设置API的版本号，固定值。
postRequest.setAction("SubmitTask");       // 设置action，固定值。
postRequest.setProduct("nls-filetrans");   // 设置产品名称，固定值。
// 设置录音文件识别请求参数，以JSON字符串的格式设置到请求Body中。
JSONObject taskObject = new JSONObject();
taskObject.put("appkey", "您的appkey");  // 项目的Appkey
taskObject.put("file_link", "您的录音文件访问链接"); // 设置录音文件的链接
taskObject.put(KEY_VERSION, "4.0"); // 新接入请使用4.0版本，已接入（默认2.0）如需维持现状，请注释掉该参数设置。
String task = taskObject.toJSONString();
postRequest.putBodyParameter("Task", task); // 设置以上JSON字符串为Body参数。
postRequest.setMethod(MethodType.POST);   // 设置为POST方式请求。
/**
 * 提交录音文件识别请求
 */
String taskId = "";  // 获取录音文件识别请求任务的ID，以供识别结果查询使用。
CommonResponse postResponse = client.getCommonResponse(postRequest);
if (postResponse.getHttpStatus() == 200) {
  JSONObject result = JSONObject.parseObject(postResponse.getData());
  String statusText = result.getString("StatusText");
  if ("SUCCESS".equals(statusText)) {
    System.out.println("录音文件识别请求成功响应： " + result.toJSONString());
    taskId = result.getString("TaskId");
  }
  else {
    System.out.println("录音文件识别请求失败： " + result.toJSONString());
    return;
  }
}
else {
  System.err.println("录音文件识别请求失败，Http错误码：" + postResponse.getHttpStatus());
  System.err.println("录音文件识别请求失败响应：" + JSONObject.toJSONString(postResponse));
  return;
}
```

录音文件识别结果查询

使用获取的任务ID，查询录音文件识别的结果。

```
/**
 * 创建CommonRequest，设置任务ID。
 */
CommonRequest getRequest = new CommonRequest();
getRequest.setDomain("filetrans.cn-shanghai.aliyuncs.com"); // 设置域名，固定值。
getRequest.setVersion("2018-08-17");        // 设置API版本，固定值。
getRequest.setAction("GetTaskResult");      // 设置action，固定值。
getRequest.setProduct("nls-filetrans");     // 设置产品名称，固定值。
getRequest.putQueryParameter("TaskId", taskId); // 设置任务ID为查询参数。
getRequest.setMethod(MethodType.GET);       // 设置为GET方式的请求。
/**
 * 提交录音文件识别结果查询请求
 * 以轮询的方式进行识别结果的查询，直到服务端返回的状态描述为“SUCCESS”、“SUCCESS_WITH_NO_VALID_
FRAGMENT”，或者为错误描述，则结束轮询。
 */
String statusText = "";
while (true) {
  CommonResponse getResponse = client.getCommonResponse(getRequest);
  if (getResponse.getHttpStatus() != 200) {
    System.err.println("识别结果查询请求失败，Http错误码： " + getResponse.getHttpStatus());
    System.err.println("识别结果查询请求失败： " + getResponse.getData());
    break;
  }
  JSONObject result = JSONObject.parseObject(getResponse.getData());
  System.out.println("识别查询结果：" + result.toJSONString());
  statusText = result.getString("StatusText");
  if ("RUNNING".equals(statusText) || "QUEUEING".equals(statusText)) {
    // 继续轮询
    Thread.sleep(3000);
  }
  else {
    break;
  }
}
if ("SUCCESS".equals(statusText) || "SUCCESS_WITH_NO_VALID_FRAGMENT".equals(statusText)) {
  System.out.println("录音文件识别成功！");
}
else {
  System.err.println("录音文件识别失败！");
}
```

示例代码

```
import com.alibaba.fastjson.JSONObject;
import com.aliyuncs.CommonRequest;
import com.aliyuncs.CommonResponse;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.http.MethodType;
import com.aliyuncs.profile.DefaultProfile;
public class FileTransJavaDemo {
  // 地域ID，常量，固定值。
  public static final String REGIONID = "cn-shanghai";
  public static final String ENDPOINTNAME = "cn-shanghai";
  public static final String PRODUCT = "nls-filetrans";
  public static final String DOMAIN = "filetrans.cn-shanghai.aliyuncs.com";
  public static final String API_VERSION = "2018-08-17";
  public static final String POST_REQUEST_ACTION = "SubmitTask";
  public static final String GET_REQUEST_ACTION = "GetTaskResult";
  // 请求参数
  public static final String KEY_APP_KEY = "appkey";
  public static final String KEY_FILE_LINK = "file_link";
  public static final String KEY_VERSION = "version";
  public static final String KEY_ENABLE_WORDS = "enable_words";
  // 响应参数
  public static final String KEY_TASK = "Task";
  public static final String KEY_TASK_ID = "TaskId";
  public static final String KEY_STATUS_TEXT = "StatusText";
  public static final String KEY_RESULT = "Result";
  // 状态值
  public static final String STATUS_SUCCESS = "SUCCESS";
  private static final String STATUS_RUNNING = "RUNNING";
  private static final String STATUS_QUEUEING = "QUEUEING";
  // 阿里云鉴权client
  IAcsClient client;
  public FileTransJavaDemo(String accessKeyId, String accessKeySecret) {
    // 设置endpoint
    try {
      DefaultProfile.addEndpoint(ENDPOINTNAME, REGIONID, PRODUCT, DOMAIN);
    } catch (ClientException e) {
      e.printStackTrace();
```

```
    }
    // 创建DefaultAcsClient实例并初始化
    DefaultProfile profile = DefaultProfile.getProfile(REGIONID, accessKeyId, accessKeySecret);
    this.client = new DefaultAcsClient(profile);
  }
  public String submitFileTransRequest(String appKey, String fileLink) {
    /**
     * 1. 创建CommonRequest，设置请求参数。
     */
    CommonRequest postRequest = new CommonRequest();
    // 设置域名
    postRequest.setDomain(DOMAIN);
    // 设置API的版本号，格式为YYYY-MM-DD。
    postRequest.setVersion(API_VERSION);
    // 设置action
    postRequest.setAction(POST_REQUEST_ACTION);
    // 设置产品名称
    postRequest.setProduct(PRODUCT);
    /**
     * 2. 设置录音文件识别请求参数，以JSON字符串的格式设置到请求Body中。
     */
    JSONObject taskObject = new JSONObject();
    // 设置appkey
    taskObject.put(KEY_APP_KEY, appKey);
    // 设置音频文件访问链接
    taskObject.put(KEY_FILE_LINK, fileLink);
    // 新接入请使用4.0版本，已接入（默认2.0）如需维持现状，请注释掉该参数设置。
    taskObject.put(KEY_VERSION, "4.0");
    // 设置是否输出词信息，默认为false，开启时需要设置version为4.0及以上。
    taskObject.put(KEY_ENABLE_WORDS, true);
    String task = taskObject.toJSONString();
    System.out.println(task);
    // 设置以上JSON字符串为Body参数。
    postRequest.putBodyParameter(KEY_TASK, task);
    // 设置为POST方式的请求。
    postRequest.setMethod(MethodType.POST);
    /**
     * 3. 提交录音文件识别请求，获取录音文件识别请求任务的ID，以供识别结果查询使用。
     */
    String taskId = null;
    try {
```

```
    try {
      CommonResponse postResponse = client.getCommonResponse(postRequest);
      System.err.println("提交录音文件识别请求的响应：" + postResponse.getData());
      if (postResponse.getHttpStatus() == 200) {
        JSONObject result = JSONObject.parseObject(postResponse.getData());
        String statusText = result.getString(KEY_STATUS_TEXT);
        if (STATUS_SUCCESS.equals(statusText)) {
          taskId = result.getString(KEY_TASK_ID);
        }
      }
    } catch (ClientException e) {
      e.printStackTrace();
    }
    return taskId;
  }
  public String getFileTransResult(String taskId) {
    /**
     * 1. 创建CommonRequest，设置任务ID。
     */
    CommonRequest getRequest = new CommonRequest();
    // 设置域名
    getRequest.setDomain(DOMAIN);
    // 设置API版本
    getRequest.setVersion(API_VERSION);
    // 设置action
    getRequest.setAction(GET_REQUEST_ACTION);
    // 设置产品名称
    getRequest.setProduct(PRODUCT);
    // 设置任务ID为查询参数
    getRequest.putQueryParameter(KEY_TASK_ID, taskId);
    // 设置为GET方式的请求
    getRequest.setMethod(MethodType.GET);
    /**
     * 2. 提交录音文件识别结果查询请求
     * 以轮询的方式进行识别结果的查询，直到服务端返回的状态描述为“SUCCESS”或错误描述，则结束轮询。
     */
    String result = null;
    while (true) {
      try {
        CommonResponse getResponse = client.getCommonResponse(getRequest);
        System.err.println("识别查询结果：" + getResponse.getData());
```

```
            if (getResponse.getHttpStatus() != 200) {
              break;
            }
            JSONObject rootObj = JSONObject.parseObject(getResponse.getData());
            String statusText = rootObj.getString(KEY_STATUS_TEXT);
            if (STATUS_RUNNING.equals(statusText) || STATUS_QUEUEING.equals(statusText)) {
              // 继续轮询，注意设置轮询时间间隔。
              Thread.sleep(10000);
            }
            else {
              // 状态信息为成功，返回识别结果；状态信息为异常，返回空。
              if (STATUS_SUCCESS.equals(statusText)) {
                result = rootObj.getString(KEY_RESULT);
                // 状态信息为成功，但没有识别结果，则可能是由于文件里全是静音、噪音等导致识别为空。
                if(result == null) {
                  result = "";
                }
              }
              break;
            }
          } catch (Exception e) {
            e.printStackTrace();
          }
        }
        return result;
      }
      public static void main(String args[]) throws Exception {
        if (args.length < 3) {
          System.err.println("FileTransJavaDemo need params: <AccessKey Id> <AccessKey Secret> <app-key>")
;
        }
        final String accessKeyId = args[0];
        final String accessKeySecret = args[1];
        final String appKey = args[2];
        String fileLink = "https://aliyun-nls.oss-cn-hangzhou.aliyuncs.com/asr/fileASR/examples/nls-sample-16k.
wav";
        FileTransJavaDemo demo = new FileTransJavaDemo(accessKeyId, accessKeySecret);
        // 第一步：提交录音文件识别请求，获取任务ID用于后续的识别结果轮询。
        String taskId = demo.submitFileTransRequest(appKey, fileLink);
        if (taskId != null) {
          System.out.println("录音文件识别请求成功，task_id: " + taskId);
```

```
    }
    else {
      System.out.println("录音文件识别请求失败！");
      return;
    }
    // 第二步：根据任务ID轮询识别结果。
    String result = demo.getFileTransResult(taskId);
    if (result != null) {
      System.out.println("录音文件识别结果查询成功：" + result);
    }
    else {
      System.out.println("录音文件识别结果查询失败！");
    }
  }
}
```

> **说明**
>
> 如果使用回调方式，请设置如下 `enable_callback` 、 `callback_url` 参数：
>
> ```
> taskObject.put("enable_callback", true);
> taskObject.put("callback_url", "回调地址");
> ```

回调服务示例：该服务用于回调方式获取识别结果，假设设置回调地址为`http://ip:port/filetrans/callback/result`。

```
package com.example.filetrans;
import com.alibaba.fastjson.JSONObject;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
import javax.servlet.ServletInputStream;
import javax.servlet.http.HttpServletRequest;
import java.io.IOException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
@RequestMapping("/filetrans/callback")
@RestController
public class FiletransCallBack {
  // 以4开头的状态码是客户端错误。
  private static final Pattern PATTERN_CLIENT_ERR = Pattern.compile("4105[0-9]*");
```

```
  // 以5开头的状态码是服务端错误。
  private static final Pattern PATTERN_SERVER_ERR = Pattern.compile("5105[0-9]*");
  // 必须是post方式
  @RequestMapping(value = "result", method = RequestMethod.POST)
  public void GetResult(HttpServletRequest request) {
    byte [] buffer = new byte[request.getContentLength()];
    ServletInputStream in = null;
    try {
      in = request.getInputStream();
      in.read(buffer, 0 ,request.getContentLength());
      in.close();
      // 获取JSON格式的文件识别结果。
      String result = new String(buffer);
      JSONObject jsonResult = JSONObject.parseObject(result);
      // 解析并输出相关结果内容。
      System.out.println("获取文件中转写回调结果:" + result);
      System.out.println("TaskId: " + jsonResult.getString("TaskId"));
      System.out.println("StatusCode: " + jsonResult.getString("StatusCode"));
      System.out.println("StatusText: " + jsonResult.getString("StatusText"));
      Matcher matcherClient = PATTERN_CLIENT_ERR.matcher(jsonResult.getString("StatusCode"));
      Matcher matcherServer = PATTERN_SERVER_ERR.matcher(jsonResult.getString("StatusCode"));
      // 以2开头状态码为正常状态码，回调方式正常状态只返回“21050000”。
      if("21050000".equals(jsonResult.getString("StatusCode"))) {
        System.out.println("RequestTime: " + jsonResult.getString("RequestTime"));
        System.out.println("SolveTime: " + jsonResult.getString("SolveTime"));
        System.out.println("BizDuration: " + jsonResult.getString("BizDuration"));
        System.out.println("Result.Sentences.size: " +
          jsonResult.getJSONObject("Result").getJSONArray("Sentences").size());
        for (int i = 0; i < jsonResult.getJSONObject("Result").getJSONArray("Sentences").size(); i++) {
          System.out.println("Result.Sentences[" + i + "].BeginTime: " +
            jsonResult.getJSONObject("Result").getJSONArray("Sentences").getJSONObject(i).getString("B
eginTime"));
          System.out.println("Result.Sentences[" + i + "].EndTime: " +
            jsonResult.getJSONObject("Result").getJSONArray("Sentences").getJSONObject(i).getString("E
ndTime"));
          System.out.println("Result.Sentences[" + i + "].SilenceDuration: " +
            jsonResult.getJSONObject("Result").getJSONArray("Sentences").getJSONObject(i).getString("Si
lenceDuration"));
          System.out.println("Result.Sentences[" + i + "].Text: " +
            jsonResult.getJSONObject("Result").getJSONArray("Sentences").getJSONObject(i).getString("T
```

```
ext"));
            System.out.println("Result.Sentences[" + i + "].ChannelId: " +
              jsonResult.getJSONObject("Result").getJSONArray("Sentences").getJSONObject(i).getString("C
hannelId"));
            System.out.println("Result.Sentences[" + i + "].SpeechRate: " +
              jsonResult.getJSONObject("Result").getJSONArray("Sentences").getJSONObject(i).getString("S
peechRate"));
            System.out.println("Result.Sentences[" + i + "].EmotionValue: " +
              jsonResult.getJSONObject("Result").getJSONArray("Sentences").getJSONObject(i).getString("E
motionValue"));
          }
        }
        else if(matcherClient.matches()) {
          System.out.println("状态码以4开头表示客户端错误.....");
        }
        else if(matcherServer.matches()) {
          System.out.println("状态码以5开头表示服务端错误.....");
        }
        else {
        }
      } catch (IOException e) {
        e.printStackTrace();
      }
    }
}
```

3.C++ Demo

本文介绍了如何使用阿里云智能语音服务提供的C++ SDK，包括SDK的安装方法及SDK代码示例。

> 说明
>
> - 当前最新版本：1.2.2。发布日期：2018年11月14日。
> - 使用SDK前，请先阅读接口说明，详情请参见接口说明。
> - 本文中的SDK只适用于2.0版语音服务。关于开通2.0版服务并获取阿里云账号AccessKey ID和AccessKey Secret，请参见开通服务。

示例说明

录音文件识别示例使用了nlsCommonSDK的 `AlibabaNlsCommon::FileTrans` 提交识别请求和查询识别结果，采用的是RPC风格的POP API调用方式。

下载安装

下载nlsCommonSDK，文件包含如下几部分：

- CMakeLists.txt：示例代码工程的CMakeList文件。
- readme.txt：SDK说明。
- release.log：更新记录。
- version：版本号。
- build.sh：Demo编译脚本。
- demo：示例文件。

文件名	描述
fileTransDemo.cpp	录音文件识别示例。

- include：包含SDK头文件，以及部分第三方头文件。各文件如下表所示。

文件名	描述
openssl	OpenSSL头文件目录，Linux下使用。
curl	curl头文件目录。
uuid	UUID头文件目录，Linux下使用。
json	jsoncpp头文件目录。

文件名	描述
Global.h	全局声明头文件。
FileTrans.h	录音文件识别头文件。

- lib：包含curl、jsoncpp动态库。

  根据平台不同，使用如下版本软件加载库文件：

  - linux（Glibc：2.5及以上，Gcc4或Gcc5）
  - windows（VS2013、VS2015）

编译运行操作步骤：

> 说明
>
> - Linux下安装工具要求如下：
>   - Glibc 2.5及以上
>   - Gcc4或Gcc5
> - Windows下需要您自己搭建示例工程（请将示例文件修改为含有BOM的 `UTF-8` 编码格式），依赖库如下：
>   - openssl-1.0.2j
>   - libuuid-1.0.3
>   - curl-7.60.0
>   - jsoncpp-0.10.6

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：

  确认本地系统已安装Cmake 2.4及以上版本。

  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` 。

  iv. 执行编译脚本：`./build.sh` 。

  v. 切换目录：`cd path/to/sdk/demo` 。

  vi. 执行文件识别示例程序：`./fileTransDemo your-AccessKeyId your-AccessKeySecret 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 fileTransDemo fileTransDemo.cpp -I path/to/sdk/include/ -L path/to/sdk/lib/linux -ljsoncpp -lssl -lcrypto -lcurl -luuid -lnlsCommonSdk -D_GLIBCXX_USE_CXX11_ABI=0` 。

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

vi. 执行文件识别示例程序：`./fileTransDemo your-AccessKeyId your-AccessKeySecret your-appkey` 。

关键接口

FileTrans：录音文件识别对象。您可以从中获取录音文件识别结果、失败信息等。

代码示例

> 说明
>
> 下载nls-sample-16k.wav。
>
> 该示例录音文件为PCM编码格式16000Hz采样率，管控台设置的模型为通用模型。如果使用其他录音文件，请填入对应的编码格式和采样率，并在管控台设置对应的模型。关于模型设置，请参见管理项目。

```
#include <iostream>
#include <string>
#include "FileTrans.h"

#ifdef _WIN32
#include <windows.h>
#endif // _WIN32

using std::string;
using std::cout;
using std::endl;
using namespace AlibabaNlsCommon;

#if defined _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

// 录音文件识别
int fileTrans(const char* accessKeyId, const char* accessKeySecret, const char* appKey, const char* fileLink)
{
  FileTrans request;

  /*设置阿里云账号AccessKey Id*/
  request.setAccessKeyId(accessKeyId);
  /*设置阿里云账号AccessKey Secret*/
  request.setKeySecret(accessKeySecret);
  /*设置阿里云AppKey*/
  request.setAppKey(appKey);
  /*设置音频文件url地址*/
  request.setFileLinkUrl(fileLink);

  /*开始文件识别, 成功返回0, 失败返回-1*/
  int ret = request.applyFileTrans();
  if (-1 == ret) {
    cout << "FileTrans failed: " << request.getErrorMsg() << endl; /*获取失败原因*/
    return -1;
  } else {
    string result = request.getResult();

#ifdef _WIN32
    result = UTF8ToGBK(result);
#endif
```

```
            cout << "FileTrans successed: " << result << endl;

            return 0;
    }
}

int main(int argc, char* argv[]) {
    if (argc < 4) {
        cout << "FileTransDemo need params : <AccessKey Id> <AccessKey Secret> <app - key>" << endl;
        return -1;
    }

    const char* accessKeyId = argv[1];
    const char* accessKeySecret = argv[2];
    const char* appKey = argv[3];
    const char* fileLink = "https://aliyun-nls.oss-cn-hangzhou.aliyuncs.com/asr/fileASR/examples/nls-sample-16k.wav";

    fileTrans(accessKeyId, accessKeySecret, appKey, fileLink);

    return 0;
}
```

4.Go Demo

本文介绍了如何使用阿里云智能语音服务提供的Go SDK，包括SDK的安装方法及SDK代码示例。

前提条件

- 使用SDK前，请先阅读接口说明，详情请参见接口说明。
- 本文中的SDK只适用于2.0版语音服务。关于开通2.0版服务并获取阿里云账号AccessKey ID和AccessKey Secret，请参见开通服务。

示例说明

录音文件识别示例使用Go SDK的CommonRequest提交识别请求和查询识别结果，采用RPC风格的POP API调用方式。

关于阿里云Go SDK请参见使用阿里云Go SDK。

Go SDK CommonRequest的使用方法请参见使用CommonRequest进行调用。

SDK安装

阿里云Go SDK支持Go 1.7及以上版本，您可以通过如下方式安装：

- 使用Glide（推荐）

```
glide get github.com/aliyun/alibaba-cloud-sdk-go
```

- 使用govendor：

```
go get -u github.com/aliyun/alibaba-cloud-sdk-go/sdk
```

调用步骤

1. 创建并初始化阿里云鉴权对象。

   鉴权使用阿里云账号的AccessKey ID和AccessKey Secret。
2. 创建录音文件识别请求，并设置请求参数。
3. 提交录音文件识别请求，处理服务端返回的响应同时获取任务ID。
4. 创建识别结果查询请求，设置查询参数为任务ID。
5. 轮询识别结果。

代码示例

> 说明
>
> 下载nls-sample-16k.wav。该录音文件为PCM编码格式16000Hz采样率，管控台设置的模型为通用模型。如果使用其他录音文件，请填入对应的编码格式和采样率，并在管控台设置对应的模型，模型设置请参见管理项目。

```
package main
import (
  "github.com/aliyun/alibaba-cloud-sdk-go/sdk/requests"
```

```
  "github.com/aliyun/alibaba-cloud-sdk-go/sdk"
  "fmt"
  "encoding/json"
  "time"
)
func main() {
  // 地域ID，固定值。
  const REGION_ID string = "cn-shanghai"
  const ENDPOINT_NAME string = "cn-shanghai"
  const PRODUCT string = "nls-filetrans"
  const DOMAIN string = "filetrans.cn-shanghai.aliyuncs.com"
  const API_VERSION string = "2018-08-17"
  const POST_REQUEST_ACTION string = "SubmitTask"
  const GET_REQUEST_ACTION  string = "GetTaskResult"
  // 请求参数
  const KEY_APP_KEY string = "appkey"
  const KEY_FILE_LINK string = "file_link"
  const KEY_VERSION string = "version"
  const KEY_ENABLE_WORDS string = "enable_words"
  // 响应参数
  const KEY_TASK string = "Task"
  const KEY_TASK_ID string = "TaskId"
  const KEY_STATUS_TEXT string = "StatusText"
  const KEY_RESULT string = "Result"
  // 状态值
  const STATUS_SUCCESS string = "SUCCESS"
  const STATUS_RUNNING string = "RUNNING"
  const STATUS_QUEUEING string = "QUEUEING"
  var accessKeyId string = "您的AccessKey Id"
  var accessKeySecret string = "您的AccessKey Secret"
  var appKey string = "您的Appkey"
  var fileLink string = "https://aliyun-nls.oss-cn-hangzhou.aliyuncs.com/asr/fileASR/examples/nls-sample-16
k.wav"
  client, err := sdk.NewClientWithAccessKey(REGION_ID, accessKeyId, accessKeySecret)
  if err != nil {
    panic(err)
  }
  postRequest := requests.NewCommonRequest()
  postRequest.Domain = DOMAIN
  postRequest.Version = API_VERSION
  postRequest.Product = PRODUCT
```

```
postRequest.Product = PRODUCT
postRequest.ApiName = POST_REQUEST_ACTION
postRequest.Method = "POST"
mapTask := make(map[string]string)
mapTask[KEY_APP_KEY] = appKey
mapTask[KEY_FILE_LINK] = fileLink
// 新接入请使用4.0版本，已接入（默认2.0）如需维持现状，请注释掉该参数设置。
mapTask[KEY_VERSION] = "4.0"
// 设置是否输出词信息，默认为false。开启时需要设置version为4.0。
mapTask[KEY_ENABLE_WORDS] = "false"
task, err := json.Marshal(mapTask)
if err != nil {
  panic(err)
}
postRequest.FormParams[KEY_TASK] = string(task)
postResponse, err := client.ProcessCommonRequest(postRequest)
if err != nil {
  panic(err)
}
postResponseContent := postResponse.GetHttpContentString()
fmt.Println(postResponseContent)
if (postResponse.GetHttpStatus() != 200) {
  fmt.Println("录音文件识别请求失败，Http错误码: ", postResponse.GetHttpStatus())
  return
}
var postMapResult map[string]interface{}
err = json.Unmarshal([]byte(postResponseContent), &postMapResult)
if err != nil {
  panic(err)
}
var taskId string = ""
var statusText string = ""
statusText = postMapResult[KEY_STATUS_TEXT].(string)
if statusText == STATUS_SUCCESS {
  fmt.Println("录音文件识别请求成功响应!")
  taskId = postMapResult[KEY_TASK_ID].(string)
} else {
  fmt.Println("录音文件识别请求失败!")
  return
}
getRequest := requests.NewCommonRequest()
```

```
    getRequest.Domain = DOMAIN
    getRequest.Version = API_VERSION
    getRequest.Product = PRODUCT
    getRequest.ApiName = GET_REQUEST_ACTION
    getRequest.Method = "GET"
    getRequest.QueryParams[KEY_TASK_ID] = taskId
    statusText = ""
    for true {
      getResponse, err := client.ProcessCommonRequest(getRequest)
      if err != nil {
        panic(err)
      }
      getResponseContent := getResponse.GetHttpContentString()
      fmt.Println("识别查询结果：", getResponseContent)
      if (getResponse.GetHttpStatus() != 200) {
        fmt.Println("识别结果查询请求失败，Http错误码：", getResponse.GetHttpStatus())
        break
      }
      var getMapResult map[string]interface{}
      err = json.Unmarshal([]byte(getResponseContent), &getMapResult)
      if err != nil {
        panic(err)
      }
      statusText = getMapResult[KEY_STATUS_TEXT].(string)
      if statusText == STATUS_RUNNING || statusText == STATUS_QUEUEING {
        time.Sleep(10 * time.Second)
      } else {
        break
      }
    }
    if statusText == STATUS_SUCCESS {
      fmt.Println("录音文件识别成功！")
    } else {
      fmt.Println("录音文件识别失败！")
    }
}
```

5..NET Demo

本文介绍了如何使用阿里云智能语音服务提供的.NET SDK，包括SDK的安装方法及SDK代码示例。

前提条件

- 使用SDK前，请先阅读接口说明，详情请参见接口说明。
- 本文中的SDK只适用于2.0版语音服务。关于开通2.0版服务并获取阿里云账号AccessKey ID和AccessKey Secret，请参见开通服务。

示例说明

录音文件识别示例使用.Net SDK的CommonRequest提交识别请求和查询识别结果，采用RPC风格的POP API调用方式。

关于阿里云.NET SDK的详细介绍请参见使用.NET SDK。

.NET SDK CommonRequest的使用方法请参见使用CommonRequest进行调用。

SDK安装

您只需安装阿里云.NET SDK核心库，有如下两种安装方式：

- 添加核心库的DLL引用
  i. 进入.NET SDK发布列表，在SDK核心库处单击右侧**适用于.NET 4.0及以上的DLL引用**。
  ii. 在Visual Studio的解决方案资源管理器，右键单击**我的项目**，然后单击**引用**。
  iii. 在弹出的菜单中单击**添加引用**。
  iv. 在弹出的对话框中，单击**浏览**，选择下载的DLL文件，单击**确定**。
- 项目引入方式
  i. 在命令行执行以下命令，在GitHub中下载SDK的源码。

  ```
  git clone https://github.com/aliyun/aliyun-openapi-net-sdk.git
  ```

  在目录aliyun-openapi-net-sdk中的aliyun-net-sdk-core/aliyun-net-sdk-core.vs2017.csproj文件（适用于VS2017），即.NET项目文件。
  ii. 在Visual Studio的界面中，右键单击**我的解决方案**。
  iii. 单击**添加 > 现有项目**。
  iv. 在弹出的对话框中，选择源码中相应的.NET项目文件（如 aliyun-net-sdk-core.vs2010.csproj），单击**打开**。
  v. 右键单击您的项目，单击**引用 > 添加引用**。

调用步骤

1. 创建并初始化AcsClient。
2. 创建录音文件识别请求，并设置请求参数。
3. 提交录音文件识别请求，处理服务端返回的响应，获取任务ID。
4. 创建识别结果查询请求，设置查询参数为任务ID。
5. 轮询识别结果。

代码示例

> 说明
>
> 下载nls-sample-16k.wav。该录音文件为PCM编码格式16000Hz采样率，管控台设置的模型为通用模型；如果使用其他录音文件，请在请求参数中填入对应的编码格式和采样率，并在管控台设置对应的模型，模型设置请参见管理项目。

```
using System;
using Newtonsoft.Json.Linq;
using Aliyun.Acs.Core;
using Aliyun.Acs.Core.Exceptions;
using Aliyun.Acs.Core.Profile;
using Aliyun.Acs.Core.Http;
namespace FileTrans
{
  class FileTrans
  {
    // 地域ID，固定值。
    public const string REGIONID = "cn-shanghai";
    public const string PRODUCT = "nls-filetrans";
    public const string DOMAIN = "filetrans.cn-shanghai.aliyuncs.com";
    public const string API_VERSION = "2018-08-17";
    public const string POST_REQUEST_ACTION = "SubmitTask";
    public const string GET_REQUEST_ACTION = "GetTaskResult";
    // 请求参数
    public const string KEY_APP_KEY = "appkey";
    public const string KEY_FILE_LINK = "file_link";
    public const string KEY_VERSION = "version";
    public const string KEY_ENABLE_WORDS = "enable_words";
    // 响应参数
    public const string KEY_TASK = "Task";
    public const string KEY_TASK_ID = "TaskId";
    public const string KEY_STATUS_TEXT = "StatusText";
    // 状态值
    public const string STATUS_SUCCESS = "SUCCESS";
    public const string STATUS_RUNNING = "RUNNING";
    public const string STATUS_QUEUEING = "QUEUEING";
    static void Main(string[] args)
    {
      if (args.Length < 3)
      {
```

```
        {
            System.Console.WriteLine("FileTrans Demo need params: <AccessKey Id> <AccessKey Secret> <app-key>");
            return;
        }
        string accessKeyId = args[0];
        string accessKeySecret = args[1];
        string appKey = args[2];
        string fileLink = "https://aliyun-nls.oss-cn-hangzhou.aliyuncs.com/asr/fileASR/examples/nls-sample-16k.wav";
        /**
         * 创建阿里云鉴权对象
         */
        IClientProfile profile = DefaultProfile.GetProfile(
            REGIONID,
            accessKeyId,
            accessKeySecret
            );
        DefaultAcsClient client = new DefaultAcsClient(profile);
        try
        {
            /**
             * 创建录音文件识别请求，设置请求参数。
             */
            CommonRequest request = new CommonRequest();
            request.Domain = DOMAIN;
            request.Version = API_VERSION;
            request.Action = POST_REQUEST_ACTION;
            request.Product = PRODUCT;
            request.Method = MethodType.POST;
            // 设置task，以JSON字符串形式设置到请求Body中。
            JObject obj = new JObject();
            obj[KEY_APP_KEY] = appKey;
            obj[KEY_FILE_LINK] = fileLink;
            // 新接入请使用4.0版本，已接入（默认2.0）如需维持现状，请注释掉该参数设置。
            obj[KEY_VERSION] = "4.0";
            // 设置是否输出词信息，默认为false。开启时需要设置version为4.0。
            obj[KEY_ENABLE_WORDS] = false;
            string task = obj.ToString();
            request.AddBodyParameters(KEY_TASK, task);
            /**
```

```
        * 提交录音文件识别请求，处理服务端返回的响应。
        */
        CommonResponse response = client.GetCommonResponse(request);
        System.Console.WriteLine(response.Data);
        if (response.HttpStatus != 200)
        {
          System.Console.WriteLine("录音文件识别请求失败： " + response.HttpStatus);
          return;
        }
        // 获取录音文件识别请求任务ID，以供识别结果查询使用。
        string taskId = "";
        JObject jsonObj = JObject.Parse(response.Data);
        string statusText = jsonObj[KEY_STATUS_TEXT].ToString();
        if (statusText.Equals(STATUS_SUCCESS))
        {
          System.Console.WriteLine("录音文件识别请求成功响应!");
          taskId = jsonObj[KEY_TASK_ID].ToString();
        }
        else
        {
          System.Console.WriteLine("录音文件识别请求失败!");
          return;
        }
        /**
        * 创建识别结果查询请求，并设置查询参数为任务ID。
        */
        CommonRequest getRequest = new CommonRequest();
        getRequest.Domain = DOMAIN;
        getRequest.Version = API_VERSION;
        getRequest.Action = GET_REQUEST_ACTION;
        getRequest.Product = PRODUCT;
        getRequest.Method = MethodType.GET;
        getRequest.AddQueryParameters(KEY_TASK_ID, taskId);
        /**
        * 提交录音文件识别结果查询请求
        * 以轮询的方式进行识别结果的查询，直到服务端返回的状态描述为“SUCCESS”、“SUCCESS_WITH_NO_
VALID_FRAGMENT”，
        * 或者为错误描述，则结束轮询。
        */
        statusText = "";
        while (true)
```

```
        {
          CommonResponse getResponse = client.GetCommonResponse(getRequest);
          System.Console.WriteLine(getResponse.Data);
          if (getResponse.HttpStatus != 200)
          {
            System.Console.WriteLine("识别结果查询请求失败，Http错误码：" + getResponse.HttpStatus);
            break;
          }
          JObject jsonObj2 = JObject.Parse(getResponse.Data);
          statusText = jsonObj2[KEY_STATUS_TEXT].ToString();
          if (statusText.Equals(STATUS_RUNNING) || statusText.Equals(STATUS_QUEUEING))
          {
            // 继续轮询
            System.Threading.Thread.Sleep(10 * 1000);
          }
          else
          {
            // 退出轮询
            break;
          }
        }
        if (statusText.Equals(STATUS_SUCCESS))
        {
          System.Console.WriteLine("录音文件识别成功！");
        }
        else {
          System.Console.WriteLine("录音文件识别失败！");
        }
      }
      catch (ServerException ex)
      {
        System.Console.WriteLine(ex.ToString());
      }
      catch (ClientException ex)
      {
        System.Console.WriteLine(ex.ToString());
      }
    }
  }
}
```

6.Node.js Demo

本文介绍如何使用阿里云智能语音服务提供的Node.js SDK，包括SDK的安装方法及SDK代码示例。

前提条件

- 使用SDK前，请先阅读接口说明，详情请参见接口说明。
- 本文中的SDK只适用于2.0版语音服务。关于开通2.0版服务并获取阿里云账号AccessKey ID和AccessKey Secret，请参见开通服务。

示例说明

录音文件识别示例使用Node.js SDK提交识别请求和查询识别结果，采用的是RPC风格的POP API调用方式。

关于阿里云Node.js SDK请参见快速开始。

SDK安装

> **说明**
>
> 阿里云Node.js SDK适用于Node.js 4.x和Node.js 6.x 两个LTS版本。您可以通过执行命令 `node -v` 查看Node.js版本。

所有阿里云官方的Node.js SDK均位于 `@alicloud` 下。录音文件识别的Node.js示例依赖 `@alicloud/nls-filetrans-2018-08-17` ，在示例文件所在目录，执行如下命令安装Node.js依赖模块：

```
npm install @alicloud/nls-filetrans-2018-08-17 --save
```

调用步骤

1. 创建并初始化阿里云鉴权对象。
2. 设置请求参数，提交录音文件识别请求，处理服务端返回的响应并获取任务ID。
3. 设置查询参数为任务ID，轮询该任务的识别结果。

代码示例

> **说明**
>
> 下载nls-sample-16k.wav。该录音文件为PCM编码格式16000Hz采样率，管控台设置的模型为通用模型；如果使用其他录音文件，请填入对应的编码格式和采样率，并在管控台设置对应的模型，模型设置请参见管理项目。

```
'use strict';
const Client = require('@alicloud/nls-filetrans-2018-08-17');
function fileTrans(akID, akSecret, appKey, fileLink) {
  //地域ID，固定值。
  var ENDPOINT = 'http://filetrans.cn-shanghai.aliyuncs.com';
  var API_VERSION = '2018-08-17';
```

```
/**
 * 创建阿里云鉴权client
 */
var client = new Client({
  accessKeyId: akID,
  secretAccessKey: akSecret,
  endpoint: ENDPOINT,
  apiVersion: API_VERSION
});
/**
 * 提交录音文件识别请求，请求参数组合成JSON格式的字符串作为task的值。
 * 请求参数appkey：项目appkey。
 * 请求参数file_link：需要识别的录音文件。
 */
var task = {
  appkey : appKey,
  file_link : fileLink,
  version : "4.0",     // 新接入请使用4.0版本，已接入（默认2.0）如需维持现状，请注释掉该参数设置。
  enable_words : false    // 设置是否输出词信息，默认值为false，开启时需要设置version为4.0。
};
task = JSON.stringify(task);
var taskParams = {
  Task : task
};
var options = {
    method: 'POST'
  };
// 提交录音文件识别请求，处理服务端返回的响应。
client.submitTask(taskParams, options).then((response) => {
  console.log(response);
  // 服务端响应信息的状态描述StatusText。
  var statusText = response.StatusText;
  if (statusText != 'SUCCESS') {
    console.log('录音文件识别请求响应失败!')
    return;
  }
  console.log('录音文件识别请求响应成功!');
  // 获取录音文件识别请求任务的TaskId，以供识别结果查询使用。
  var taskId = response.TaskId;
  /**
   * 以TaskId为查询参数，提交识别结果查询请求
```

```
    * 以taskId为查询参数，提交识别结果查询请求。
    * 以轮询的方式进行识别结果的查询，直到服务端返回的状态描述为"SUCCESS"、SUCCESS_WITH_NO_VALID_FRAGMENT，
    * 或者为错误描述，则结束轮询。
    */
    var taskIdParams = {
      TaskId : taskId
    };
    var timer = setInterval(() => {
      client.getTaskResult(taskIdParams).then((response) => {
        console.log('识别结果查询响应：');
        console.log(response);
        var statusText = response.StatusText;
        if (statusText == 'RUNNING' || statusText == 'QUEUEING') {
          // 继续轮询，注意间隔周期。
        }
        else {
          if (statusText == 'SUCCESS' || statusText == 'SUCCESS_WITH_NO_VALID_FRAGMENT') {
            console.log('录音文件识别成功：');
            var sentences = response.Result;
            console.log(sentences);
          }
          else {
            console.log('录音文件识别失败!');
          }
          // 退出轮询
          clearInterval(timer);
        }
      }).catch((error) => {
        console.error(error);
        // 异常情况，退出轮询。
        clearInterval(timer);
      });
    }, 10000);
  }).catch((error) => {
    console.error(error);
  });
}
var akId = '您的AccessKey Id';
var akSecret = '您的AccessKey Secret';
var appKey = '您的appkey';
```

```
var fileLink = 'https://aliyun-nls.oss-cn-hangzhou.aliyuncs.com/asr/fileASR/examples/nls-sample-16k.wav';
fileTrans(akId, akSecret, appKey, fileLink);
```

7.PHP Demo

本文介绍如何使用阿里云智能语音服务提供的PHP SDK，包括SDK的安装方法及SDK代码示例。

前提条件

- 使用SDK前，请先阅读接口说明，详情请参见接口说明。
- 已开通2.0版本语音服务且已获取AccessKey ID和AccessKey Secret，详情请参见开通服务。

> 说明
>
> - 本文PHP示例基于阿里云新版PHP SDK（Alibaba Cloud SDK for PHP）开发。如果您已接入阿里云旧版PHP SDK（aliyun-openapi-php-sdk），仍然可以继续使用或者更新到新版SDK（推荐）。
> - 后续录音文件识别的PHP示例更新将基于新版PHP SDK。

SDK说明

录音文件识别的PHP示例使用了阿里云的PHP SDK提交录音文件识别请求和查询识别结果，采用RPC风格的POP API调用方式。

关于阿里云PHP SDK的详细介绍请参见PHP SDK。

> 注意
>
> 阿里云PHP SDK适用于PHP的5.5.0或更高版本。

安装PHP SDK

阅读PHP SDK的安装方式，详情请参见安装Alibaba Cloud SDK for PHP。

调用步骤

1. 创建一个全局客户端。
2. 设置请求参数，提交录音文件识别请求；处理服务端返回的响应，获取任务ID，用于后续的识别结果轮询。
3. 根据任务ID，轮询识别结果。

代码示例

> 说明
>
> 下载nls-sample-16k.wav。该录音文件为PCM编码格式16000Hz采样率，管控台设置的模型为通用模型；如果使用其他录音文件，请填入对应的编码格式和采样率，并在管控台设置对应的模型，模型设置请参见管理项目。

```
<?php
require __DIR__ . '/vendor/autoload.php';
use AlibabaCloud\Client\AlibabaCloud;
use AlibabaCloud\Client\Exception\ClientException;
```

```
use AlibabaCloud\Client\Exception\ClientException;
use AlibabaCloud\Client\Exception\ServerException;
class NLSFileTrans {
  // 请求参数
  private const KEY_APP_KEY = "appkey";
  private const KEY_FILE_LINK = "file_link";
  private const KEY_VERSION = "version";
  private const KEY_ENABLE_WORDS = "enable_words";
  // 响应参数
  private const KEY_TASK_ID = "TaskId";
  private const KEY_STATUS_TEXT = "StatusText";
  private const KEY_RESULT = "Result";
  // 状态值
  private const STATUS_SUCCESS = "SUCCESS";
  private const STATUS_RUNNING = "RUNNING";
  private const STATUS_QUEUEING = "QUEUEING";
  function submitFileTransRequest($appKey, $fileLink) {
    // 获取task JSON字符串，包含appkey和file_link参数等。
    // 新接入请使用4.0版本，已接入（默认2.0）如需维持现状，请注释掉该参数设置。
    // 设置是否输出词信息，默认为false，开启时需要设置version为4.0。
    $taskArr = array(self::KEY_APP_KEY => $appKey, self::KEY_FILE_LINK => $fileLink, self::KEY_VERSION => "4
.0", self::KEY_ENABLE_WORDS => FALSE);
    $task = json_encode($taskArr);
    print $task . "\n";
    try {
      // 提交请求，返回服务端的响应。
      $submitTaskResponse = AlibabaCloud::nlsFiletrans()
                    ->v20180817()
                    ->submitTask()
                    ->withTask($task)
                    ->request();
      print $submitTaskResponse . "\n";
      // 获取录音文件识别请求任务的ID，以供识别结果查询使用。
      $taskId = NULL;
      $statusText = $submitTaskResponse[self::KEY_STATUS_TEXT];
      if (strcmp(self::STATUS_SUCCESS, $statusText) == 0) {
        $taskId = $submitTaskResponse[self::KEY_TASK_ID];
      }
      return $taskId;
    } catch (ClientException $exception) {
      // 获取错误消息
```

```
        print_r($exception->getErrorMessage());
    } catch (ServerException $exception) {
        // 获取错误消息
        print_r($exception->getErrorMessage());
    }
  }
  function getFileTransResult($taskId) {
    $result = NULL;
    while (TRUE) {
      try {
        $getResultResponse = AlibabaCloud::nlsFiletrans()
                            ->v20180817()
                            ->getTaskResult()
                            ->withTaskId($taskId)
                            ->request();
        print "识别查询结果: " . $getResultResponse . "\n";
        $statusText = $getResultResponse[self::KEY_STATUS_TEXT];
        if (strcmp(self::STATUS_RUNNING, $statusText) == 0 || strcmp(self::STATUS_QUEUEING, $statusText)
== 0) {
          // 继续轮询
          sleep(10);
        }
        else {
          if (strcmp(self::STATUS_SUCCESS, $statusText) == 0) {
            $result = $getResultResponse;
          }
          // 退出轮询
          break;
        }
      } catch (ClientException $exception) {
        // 获取错误消息
        print_r($exception->getErrorMessage());
      } catch (ServerException $exception) {
        // 获取错误消息
        print_r($exception->getErrorMessage());
      }
    }
    return $result;
  }
}
$accessKeyId = "您的AccessKey Id";
```

```
$accessKeySecret = "您的AccessKey Secret";
$appKey = "您的appkey";
$fileLink = "https://aliyun-nls.oss-cn-hangzhou.aliyuncs.com/asr/fileASR/examples/nls-sample-16k.wav";
/**
 * 第一步：设置一个全局客户端。
 * 使用阿里云RAM账号的AccessKey ID和AccessKey Secret进行鉴权。
 */
AlibabaCloud::accessKeyClient($accessKeyId, $accessKeySecret)
        ->regionId("cn-shanghai")
        ->asGlobalClient();
$fileTrans = new NLSFileTrans();
/**
 * 第二步：提交录音文件识别请求，获取任务ID，用于后续的识别结果轮询。
 */
$taskId = $fileTrans->submitFileTransRequest($appKey, $fileLink);
if ($taskId != NULL) {
  print "录音文件识别请求成功，task_id: " . $taskId . "\n";
}
else {
  print "录音文件识别请求失败!";
  return ;
}
/**
 * 第三步：根据任务ID轮询识别结果。
 */
$result = $fileTrans->getFileTransResult($taskId);
if ($result != NULL) {
  print "录音文件识别结果查询成功: " . $result . "\n";
}
else {
  print "录音文件识别结果查询失败!";
}
```

8.Python Demo

本文介绍如何使用阿里云智能语音服务提供的Python SDK，包括SDK的安装方法及SDK代码示例。

前提条件

- 使用SDK前，请先阅读接口说明，详情请参见接口说明。
- 本文中SDK只适用于2.0版语音服务，开通服务并获取阿里云账号的AccessKey ID和AccessKey Secret，请参见开通服务。

SDK说明

录音文件识别的Python示例使用了阿里云Python SDK的CommonRequest提交录音文件识别请求和查询识别结果，采用RPC风格的POP API调用方式。

关于使用阿里云Python SDK请参见使用Python SDK。

关于Python SDK CommonRequest的使用方法请参见使用CommonRequest进行调用。

SDK安装

运行录音文件识别Python示例，只需安装阿里云Python SDK的核心库。

阿里云Python SDK支持python版本如下，并提供pip和GitHub两种安装方式。

- Python 2.6及以上
- Python 2.7及以上
- Python 3及以上

使用pip安装（推荐）：

执行如下命令，通过pip安装Python SDK，版本为2.13.3：

```
pip install aliyun-python-sdk-core==2.13.3
```

调用步骤

1. 创建并初始化AcsClient实例。
2. 创建录音文件识别请求，设置请求参数。
3. 提交录音文件识别请求，处理服务端返回的响应，获取任务ID。
4. 创建识别结果查询请求，设置查询参数为任务ID。
5. 轮询识别结果。

示例代码

> **说明**
>
> 下载nls-sample-16k.wav。
>
> 示例中使用的录音文件为PCM编码格式16000Hz采样率，管控台设置的模型为通用模型；如果使用其他录音文件，请填入对应的编码格式和采样率，并在管控台设置对应的模型，关于模型设置参见管理项目。

```
# -*- coding: utf8 -*-
import json
import time
from aliyunsdkcore.acs_exception.exceptions import ClientException
from aliyunsdkcore.acs_exception.exceptions import ServerException
from aliyunsdkcore.client import AcsClient
from aliyunsdkcore.request import CommonRequest
def fileTrans(akId, akSecret, appKey, fileLink) :
  # 地域ID，固定值。
  REGION_ID = "cn-shanghai"
  PRODUCT = "nls-filetrans"
  DOMAIN = "filetrans.cn-shanghai.aliyuncs.com"
  API_VERSION = "2018-08-17"
  POST_REQUEST_ACTION = "SubmitTask"
  GET_REQUEST_ACTION = "GetTaskResult"
  # 请求参数
  KEY_APP_KEY = "appkey"
  KEY_FILE_LINK = "file_link"
  KEY_VERSION = "version"
  KEY_ENABLE_WORDS = "enable_words"
  # 是否开启智能分轨
  KEY_AUTO_SPLIT = "auto_split"
  # 响应参数
  KEY_TASK = "Task"
  KEY_TASK_ID = "TaskId"
  KEY_STATUS_TEXT = "StatusText"
  KEY_RESULT = "Result"
  # 状态值
  STATUS_SUCCESS = "SUCCESS"
  STATUS_RUNNING = "RUNNING"
  STATUS_QUEUEING = "QUEUEING"
  # 创建AcsClient实例
  client = AcsClient(akId, akSecret, REGION_ID)
  # 提交录音文件识别请求
  postRequest = CommonRequest()
  postRequest.set_domain(DOMAIN)
  postRequest.set_version(API_VERSION)
  postRequest.set_product(PRODUCT)
  postRequest.set_action_name(POST_REQUEST_ACTION)
  postRequest.set_method('POST')
```

```
  # 新接入请使用4.0版本，已接入（默认2.0）如需维持现状，请注释掉该参数设置。
  # 设置是否输出词信息，默认为false，开启时需要设置version为4.0。
  task = {KEY_APP_KEY : appKey, KEY_FILE_LINK : fileLink, KEY_VERSION : "4.0", KEY_ENABLE_WORDS : False}
  # 开启智能分轨，如果开启智能分轨，task中设置KEY_AUTO_SPLIT为True。
  # task = {KEY_APP_KEY : appKey, KEY_FILE_LINK : fileLink, KEY_VERSION : "4.0", KEY_ENABLE_WORDS : False
, KEY_AUTO_SPLIT : True}
  task = json.dumps(task)
  print(task)
  postRequest.add_body_params(KEY_TASK, task)
  taskId = ""
  try :
    postResponse = client.do_action_with_exception(postRequest)
    postResponse = json.loads(postResponse)
    print (postResponse)
    statusText = postResponse[KEY_STATUS_TEXT]
    if statusText == STATUS_SUCCESS :
      print ("录音文件识别请求成功响应！")
      taskId = postResponse[KEY_TASK_ID]
    else :
      print ("录音文件识别请求失败！")
      return
  except ServerException as e:
    print (e)
  except ClientException as e:
    print (e)
  # 创建CommonRequest，设置任务ID。
  getRequest = CommonRequest()
  getRequest.set_domain(DOMAIN)
  getRequest.set_version(API_VERSION)
  getRequest.set_product(PRODUCT)
  getRequest.set_action_name(GET_REQUEST_ACTION)
  getRequest.set_method('GET')
  getRequest.add_query_param(KEY_TASK_ID, taskId)
  # 提交录音文件识别结果查询请求
  # 以轮询的方式进行识别结果的查询，直到服务端返回的状态描述符为"SUCCESS"、"SUCCESS_WITH_NO_VALID_
FRAGMENT"，
  # 或者为错误描述，则结束轮询。
  statusText = ""
  while True :
    try :
      getResponse = client.do_action_with_exception(getRequest)
```

```
        getResponse = json.loads(getResponse)
        print (getResponse)
        statusText = getResponse[KEY_STATUS_TEXT]
        if statusText == STATUS_RUNNING or statusText == STATUS_QUEUEING :
          # 继续轮询
          time.sleep(10)
        else :
          # 退出轮询
          break
      except ServerException as e:
        print (e)
      except ClientException as e:
        print (e)
    if statusText == STATUS_SUCCESS :
      print ("录音文件识别成功！")
    else :
      print ("录音文件识别失败！")
    return
accessKeyId = "您的AccessKey Id"
accessKeySecret = "您的AccessKey Secret"
appKey = "您的appkey"
fileLink = "https://aliyun-nls.oss-cn-hangzhou.aliyuncs.com/asr/fileASR/examples/nls-sample-16k.wav"
# 执行录音文件识别
fileTrans(accessKeyId, accessKeySecret, appKey, fileLink)
```

9.使用函数计算方式的录音文件识别

本文为您介绍如何使用函数计算方式进行录音文件识别。

概述

对于将音频文件存储在阿里云OSS上的用户，除使用SDK集成录音文件识别的开发方式外，还可以通过函数计算的方式，录音文件识别通过触发器函数自动执行，将识别结果保存回OSS或者其他存储器上，您只需关注最终的识别结果，减少SDK集成开发工作量。对于非开发人员，可以通过该方式快速获取识别结果进行分析。函数计算的详细介绍，请参见什么是函数计算。

前提条件

已开通如下服务，且需要给开通函数计算服务的账号授权OSS服务的读写权限：

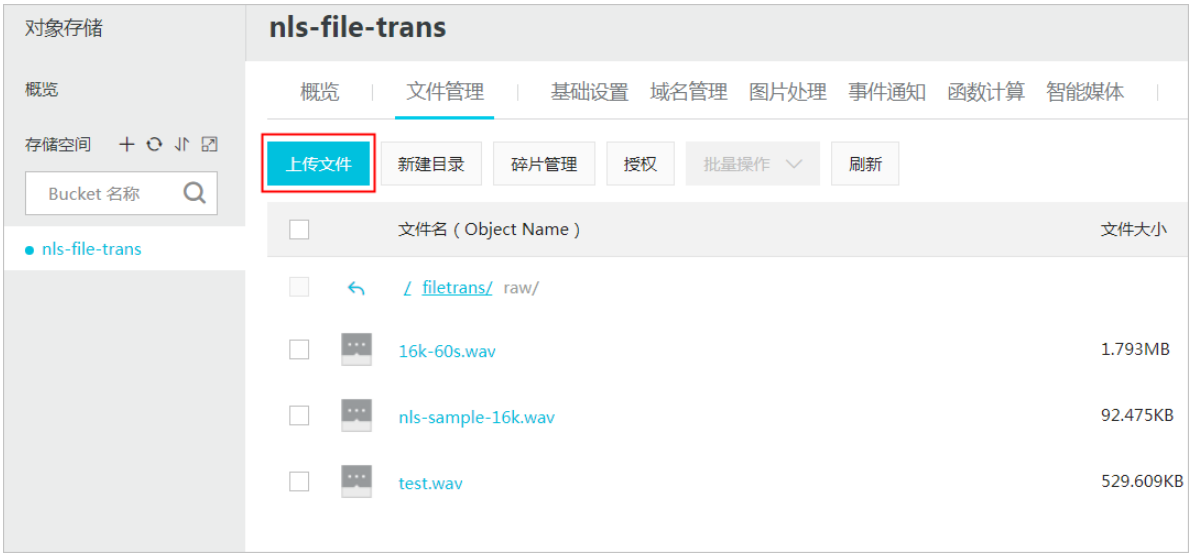
- 已开通OSS服务，有对应的AccessKey ID、AccessKey Secret、OSS EndPoint，详情参见对象存储。
- 已开通函数计算服务，详情参见函数计算。
- 已开通智能语音交互服务，有对应的AccessKey ID、AccessKey Secret、appkey，详情参见智能语音交互。

效果

> **说明**
>
> 本文使用的OSS Bucket为nls-file-trans，音频文件存放路径为filetrans/raw，识别结果存放路径为filetrans/result。识别结果保存在JSON文件中，以 文件名_taskId.json 表示识别成功的结果，文件名_taskId_failed.json 表示识别失败的结果。使用中，请根据实际情况，修改为自己的Bucket和文件存放路径。

- 通过OSS控制台上传音频文件

  单击**上传文件**，上传音频文件到指定的Bucket路径filetrans/raw：



  对应函数计算的触发器：

识别结果存放路径filetrans/result：

上传的nls-sample-16k.wav音频文件识别结果：


```
{
  "Result": {
    "Sentences": [{
      "EndTime": 2365,
      "SilenceDuration": 0,
      "BeginTime": 340,
      "Text": "北京的天气。",
      "ChannelId": 0,
      "SpeechRate": 177,
      "EmotionValue": 5.0
    }]
  },
  "TaskId": "fb0474184c6d11e9a213e11db149****",
  "StatusCode": 21050000,
  "StatusText": "SUCCESS",
  "RequestTime": 1553237085804,
  "SolveTime": 1553237086146,
  "BizDuration": 2956
}
```

- 通过OSS常用工具上传音频文件

  以ossutil为例，与控制台上传音频文件的识别流程相同。

  ```
  ossutil cp nls-sample-16k.wav oss://nls-file-trans/filetrans/raw/nls-sample-16k.wav
  ```

实现方式

需要在函数计算上实现的内容如下：

1. 创建函数计算的服务。
2. 创建生成任务的函数，使用OSS触发器，设置第3步的回调URL，生成录音文件识别任务。
3. 创建接收回调的函数，使用HTTP触发器，生成回调URL，用于将录音文件识别结果写回OSS。

步骤一：创建服务

1. 在函数计算的控制台创建一个服务。

   可以在创建时进行高级配置，也可以创建后再进行设置。

2. 创建服务角色。

   在**高级配置 > 权限配置**中，如果没有已有角色，需要创建一个新角色，**系统模版授权**选择AliyunOSSFullAccess，单击**点击授权**。

3. 添加权限。

   登录RAM访问控制，在左侧导航栏选择**RAM角色管理**，添加 `AliyunOSSFullAccess` 、 `AliyunSTSAssumeRoleAccess` 、 `AliyunNLSFullAccess` 权限（如果已经授权，则不需要添加）。

步骤二：生成任务

生成任务用于提交录音文件识别请求，OSS上传音频文件时将触发OSS触发器，调用函数，提交录音文件识别请求。

1. 编写函数。

   在创建的服务下，创建一个函数，用于生成任务：

   - 函数名称：call_filetrans
   - 函数入口：call_filetrans.handler
   - 运行环境：Python 3
   - 函数执行内存：128 MB（选择最小值128 MB即可，该值与函数计算的计费相关）
   - 超时时间：600s

   函数内容如下，需要您替换如下参数：

   - OSS账号相关的AccessKey ID、AccessKey Secret、OSS EndPoint（选择外网访问）
   - 录音文件识别账号相关的AccessKey ID、AccessKey Secret、appkey、回调URL（创建HTTP触发器时获取）

```
# -*- coding: utf-8 -*-
import json
```

```
import time
from aliyunsdkcore.acs_exception.exceptions import ClientException
from aliyunsdkcore.acs_exception.exceptions import ServerException
from aliyunsdkcore.client import AcsClient
from aliyunsdkcore.auth.credentials import StsTokenCredential
from aliyunsdkcore.request import CommonRequest
import oss2
import logging
ossEndPoint = "your OSS EndPoint"        # oss账号EndPoint，请选择外网访问。
fileTransAppkey = "your appkey"  # 录音文件识别账号appkey。
fileTransCallbackUrl = "您的回调URL" # 在创建HTTP触发器时获取。
def handler(event, context):
  logger = logging.getLogger()
  logger.info(event)
  eventObj = json.loads(event)["events"]
  eventName=eventObj[0]["eventName"]
  bucketName=eventObj[0]["oss"]["bucket"]["name"]
  ossFileName=eventObj[0]["oss"]["object"]["key"]
  logger.info("eventName: %s" % eventName)
  logger.info("bucketName: %s" % bucketName)
  logger.info("ossFileName: %s" % ossFileName)
  appKey = fileTransAppkey
  # 如下取值固定。
  REGION_ID = "cn-shanghai"
  PRODUCT = "nls-filetrans"
  DOMAIN = "filetrans.cn-shanghai.aliyuncs.com"
  API_VERSION = "2018-08-17"
  POST_REQUEST_ACTION = "SubmitTask"
  GET_REQUEST_ACTION = "GetTaskResult"
  KEY_APP_KEY = "appkey"
  KEY_FILE_LINK = "file_link"
  KEY_VERSION = "version"
  KEY_TASK = "Task"
  KEY_TASK_ID = "TaskId"
  KEY_STATUS_TEXT = "StatusText"
  creds = context.credentials
  # 创建AcsClient实例
  sts_token_credential = StsTokenCredential(creds.accessKeyId, creds.accessKeySecret, creds.security
Token)
  client = AcsClient(region_id=REGION_ID, credential=sts_token_credential)
```

```
    # 创建提交录音文件识别请求，并设置请求参数。
    postRequest = CommonRequest()
    postRequest.set_domain(DOMAIN)
    postRequest.set_version(API_VERSION)
    postRequest.set_product(PRODUCT)
    postRequest.set_action_name(POST_REQUEST_ACTION)
    postRequest.set_method('POST')
    filename = ossFileName.split('/')[-1]
    # create file url
    auth = oss2.StsAuth(creds.accessKeyId, creds.accessKeySecret, creds.securityToken)
    bucket = oss2.Bucket(auth, ossEndPoint, bucketName)
    fileLink = bucket.sign_url('GET', ossFileName, 3600)
    logger.info("file link = " + fileLink)
    # 如下回调地址，在第二步中生成。
    callback_url = fileTransCallbackUrl + "/" + filename
    logger.info("callback url = " + callback_url)
    task = {"SecurityToken": creds.securityToken, KEY_APP_KEY : appKey, KEY_FILE_LINK : fileLink, KEY_V
ERSION : "4.0", "enable_words" : False, "enable_callback" : True, "callback_url" : callback_url}
    task = json.dumps(task)
    #logger.info (task)
    postRequest.add_body_params(KEY_TASK, task)
    taskId = ""
    try :
      # 提交录音文件识别请求，处理服务端返回的响应。
      postResponse = client.do_action_with_exception(postRequest)
      postResponse = json.loads(postResponse)
      logger.info(postResponse)
      # 获取录音文件识别请求任务的ID，以供识别结果查询使用。
      statusText = postResponse[KEY_STATUS_TEXT]
      if statusText == "SUCCESS" :
        logger.info("录音文件识别请求成功响应！")
        taskId = postResponse[KEY_TASK_ID]
        logger.info("taskId = " + taskId)
      else :
        logger.info ("录音文件识别请求失败！")
    except ServerException as e:
      logger.error(e)
    except ClientException as e:
      logger.error(e)
    logger.info('hello world')
    logger.info(taskId)
```

```
return taskId
```

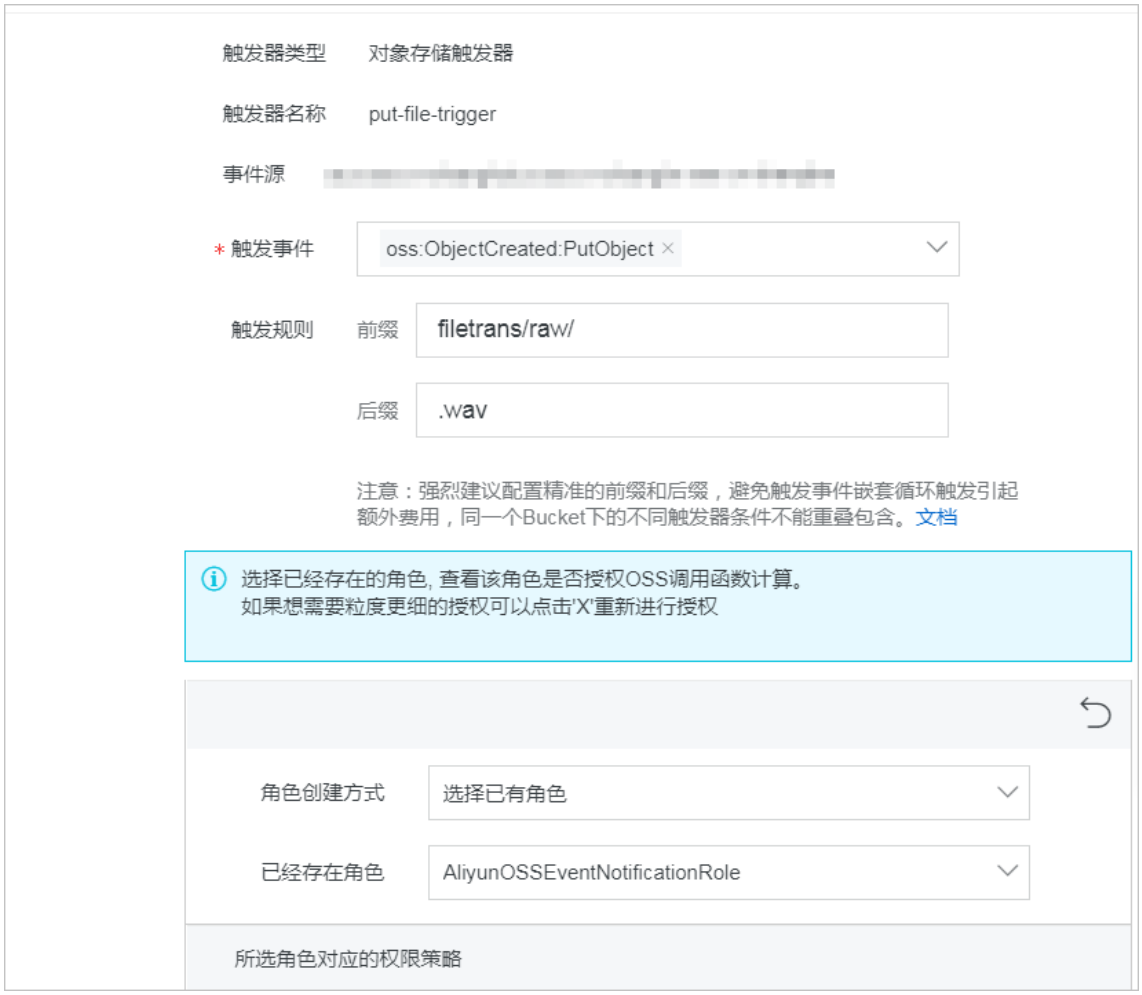
2. 创建并配置OSS触发器。

> **说明**
>
> 由于上传音频文件到OSS的Bucket中，有两种方式：Put和Post。所以为了触发所有的上传事件，需要创建两个OSS触发器。

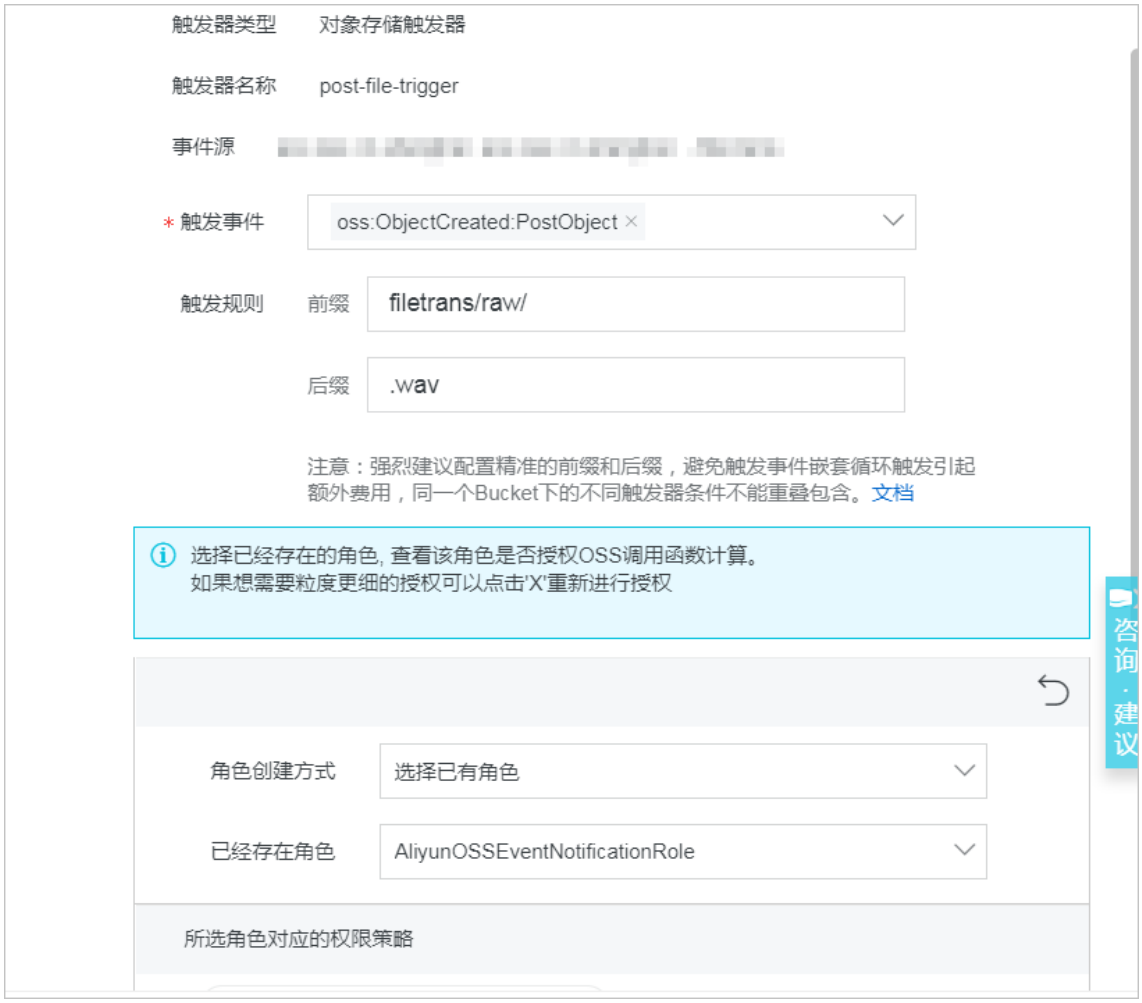
参数	说明
触发器类型	对象存储触发器。
触发器名称	自定义名称。
触发事件	分别选择 `oss:ObjectCreated:PutObject` 和 `oss:ObjectCreated:PostObject` 创建两个触发器。
触发规则：前缀	**filetrans/raw/**
触发规则：后缀	**.wav**
角色	如果已有合适角色，可直接选择；如果没有角色，则选择新建角色。

创建成功后，可以在OSS控制台对应Bucket的函数计算中找到这两个触发器。

- oss:ObjectCreated:PutObject事件：



- oss:ObjectCreated:PostObject事件：



步骤三：接收回调

录音文件识别服务识别完成后，将识别结果通过HTTP触发器，写回OSS的Bucket中。在创建HTTP触发器的时候可以获得回调URL，请设置到生成任务函数的回调URL中。

1. 编写函数。

   在创建的服务下，创建一个函数，用于将识别结果写回OSS：

   - 函数名称：put_http_post_to_oss
   - 函数入口：index.handler
   - 运行环境：Python 3
   - 函数执行内存：128 MB（选择最小值128 MB，该值与函数计算的计费相关）
   - 超时时间：600s

   函数内容如下，需要您替换如下参数：

   - OSS的EndPoint（外网访问）
   - OSS Bucket名称
   - OSS Bucket中的目录，用于存放识别结果文件。

```
# -*- coding: utf-8 -*-
import logging
```

```
import oss2
import json
endpoint = "您的OSS EndPoint" # oss配置
bucketName = "您的OSS Bucket名称" # oss配置
resultOssPath = "您的录音文件识别结果存放目录" # oss路径，用于存储识别结果文件。
B_OK = b"ok"
def handler(environ, start_response):
  logger = logging.getLogger()
  context = environ['fc.context']
  request_uri = environ['fc.request_uri']
  for k, v in environ.items():
   if k.startswith("HTTP_"):
    # process custom request headers
    pass
  logger.info("request_uri = " + request_uri)
  filename = request_uri.split('/')[-1]
  logger.info('filename = ' + filename)
  # get request_body
  try:
    request_body_size = int(environ.get('CONTENT_LENGTH', 0))
  except (ValueError):
    request_body_size = 0
  request_body = environ['wsgi.input'].read(request_body_size)
  # parse result
  postResponse = json.loads(request_body)
  taskId = postResponse['TaskId']
  statusText = postResponse['StatusText']
  # put to oss
  creds = context.credentials
  auth = oss2.StsAuth(creds.accessKeyId, creds.accessKeySecret, creds.securityToken)
  logger.info("creds.accessKeyId = " + creds.accessKeyId)
  logger.info("creds.accessKeySecret = " + creds.accessKeySecret)
  logger.info("creds.securityToken = " + creds.securityToken)
  bucket = oss2.Bucket(auth, endpoint, bucketName)
  if statusText == "SUCCESS" :
   filename = resultOssPath + "/" + filename + "_" + taskId + '.json'
  else :
   filename = resultOssPath + "/" + filename + "_" + taskId + '_failed.json'
  logger.info('filename = ' + filename)
  bucket.put_object(filename, request_body)
  # do something here
```

```
# do something here
status = '200 OK'
response_headers = [('Content-type', 'text/plain')]
start_response(status, response_headers)
return [B_OK]
```

2. 创建并配置HTTP触发器。

   根据下表说明创建成功后，会自动生成路径作为回调URL。请将其设置到生成任务函数的回调URL中。

参数	说明
服务类型	HTTP触发器
触发器名称	http_post_caller
认证方式	anonymous
请求方式	POST