智能语音V2.X 实时语音识别SDK使用文档

功能介绍

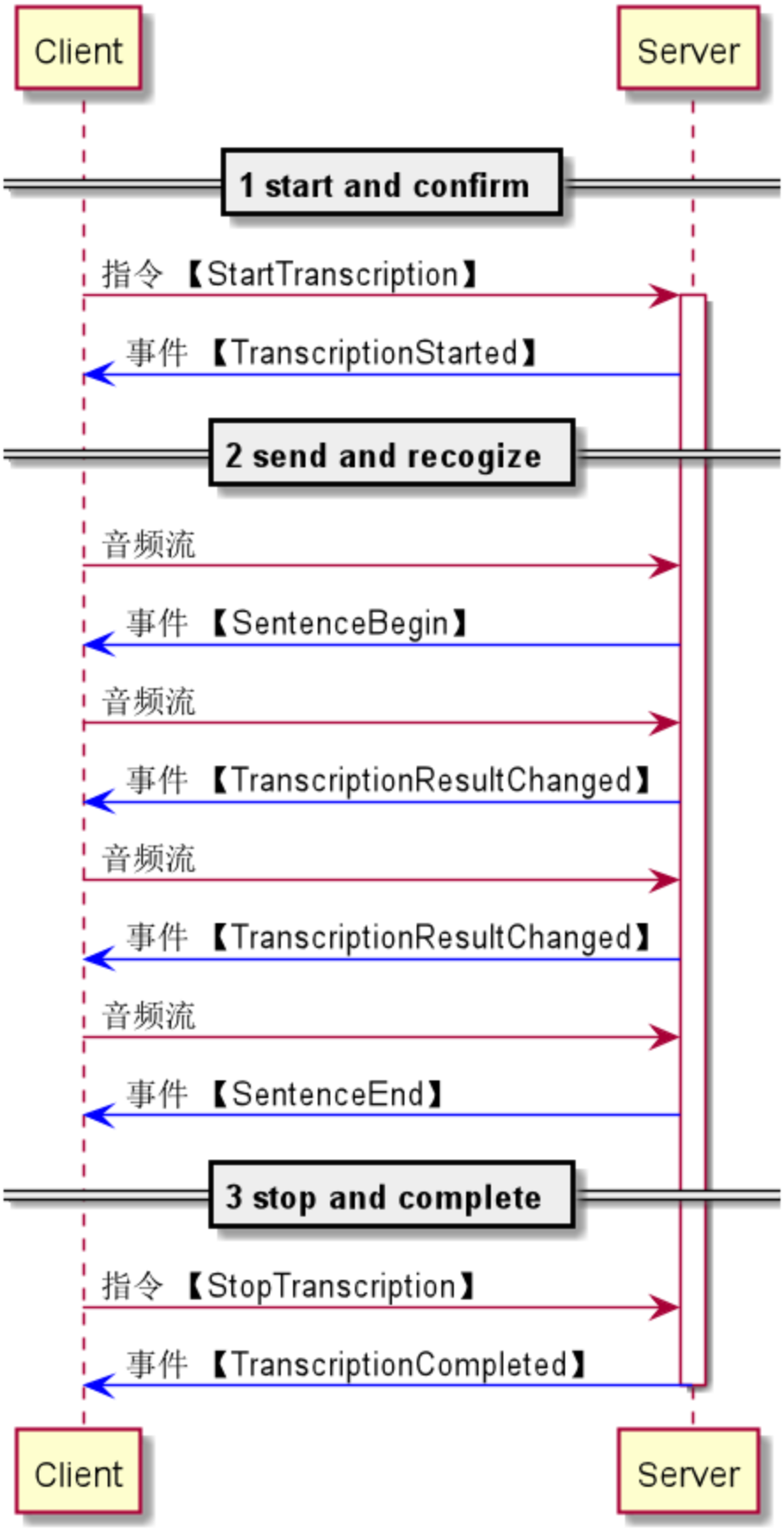
对长时间的语音数据流进行识别，适用于会议演讲、视频直播等长时间不间断识别的场景。

- 支持音频编码格式：pcm（无压缩的pcm文件或wav文件），16bit采样位数的单声道(mono)；
- 支持音频采样率：8000Hz、16000Hz；
- 支持对返回结果进行设置：是否返回中间识别结果，是否在后处理中添加标点，是否将中文数字转为阿拉伯数字输出。
- 支持普通话、方言、英语等多种语言的识别。
- 支持自学习模型和热词。

服务地址

实时语音识别的WebSocket服务地址为：`ws://gateway所在IP:8101/ws/v1` 。gateway默认的端口号为`8101`，如果部署时修改过gateway的端口号，请使用修改后的端口号。

交互流程



说明：所有服务端的响应都会在返回信息的header包含task_id参数，用于表示本次识别任务的ID，请记录下这个值，如果发生错误，请将task_id和错误信息提交到工单或者语音接口人。

0.鉴权

客户端在与服务端建立WebSocket链接的时候，需要使用token进行鉴权。在专有云下，token的值为`default`。

1.start and confirm(设置请求参数)

客户端发起请求，服务端确认请求有效。其中在请求消息中需要进行参数设置，各参数由SDK中SpeechTranscriber对象的相关set方法设置，各参数含义如下：

普通参数设置:

参数	类型	是否必需	说明
appkey	Sring	是	业务方或者业务场景的标记， 专有云默认为default
format	String	否	音频编码格式， 支持的格式： pcm（无压缩的pcm文件或wav文件）
sample_rate	Integer	否	音频采样率， 默认是16000Hz， 请确保采样率与模型类型保持一致
enable_intermediate_result	Boolean	否	是否返回中间识别结果，默认是False
enable_punctuation_prediction	Boolean	否	是否在后处理中添加标点，默认是False
enable_inverse_text_normalization	Boolean	否	是否在后处理中执行ITN，设置为true时， 中文数字将转为阿拉伯数字输出， 默认是False

高级参数设置:

参数	类型	是否必需	说明
model	String	否	模型名称
customization_id	String	否	定制模型ID
vocabulary_id	String	否	定制泛热词ID
class_vocabulary_id	Map	否	定制类热词ID， Map的Key是类名， Value是类热词词表ID 。 目前只有law_politics(政法)， customer_service_8k(客服质检) 模型支持人名(PERSON)、地名(ADDRESS)类热词
max_sentence_silence	Integer	否	语音断句检测阈值， 静音时长超过该阈值会被认为断句， 合法参数范围200～2000(ms)， 默认值800ms， 开启语义断句(enable_semantic_sentence_detection)后，此参数无效
enable_semantic_sentence_detection	Boolean	否	是否开启语语义断句， 可选，默认是False； 开启后会走后后处理服务， 后处理默认会打开顺滑、标点、ITN

vad_silence_duration	Integer	否	语义断句长静默的静默时长参数， 用户触发此参数阈值后会自动断句， 开启语义断句后生效， 合法参数范围[2000 10000]ms， 默认值2000ms
enable_words	Boolean	否	是否开启返回词信息， 默认False不开启， 在vad断句时生效
disfluency	Boolean	否	顺滑，默认False关闭
enable_ignore_sentence_timeout	Boolean	否	默认False， 开启后忽略实时转写处理过程中的asr超时错误

2.send and recognize

循环发送语音数据，持续接收识别结果：

- SentenceBegin事件表示服务端检测到了一句话的开始。实时语音识别服务的智能断句功能会判断出一句话的开始与结束，如：

```
{
    "header": {
        "message_id": "05450bf69c53413f8d88aed1ee600e93",
        "task_id": "640bc797bb684bd69601856513079df5",
        "namespace": "SpeechTranscriber",
        "name": "SentenceBegin",
        "status": 20000000,
        "status_message": "GATEWAY|SUCCESS|Success."
    },
    "payload": {
        "index": 1,
        "time": 320
    }
}
```

header对象参数说明：

参数	类型	说明
namespace	String	消息所属的命名空间
name	String	消息名称， SentenceBegin表示一个句子的开始
status	Integer	状态码，表示请求是否成功， 见服务状态码
status_text	String	状态消息
task_id	String	任务全局唯一ID，请记录该值， 便于排查问题
message_id	String	本次消息的ID

payload对象参数说明：

参数	类型	说明
index	Integer	句子编号，从1开始递增
time	Integer	当前已处理的音频时长， 单位是毫秒

- TranscriptionResultChanged事件表示识别结果发生了变化。仅当enable_intermediate_result=true时会返回此消息，且多次返回，即一句话的中间识别结果，如：

```
{
    "header": {
        "message_id": "05450bf69c53413f8d88aed1ee600e93",
        "task_id": "640bc797bb684bd69601856513079df5",
        "namespace": "SpeechTranscriber",
        "name": "TranscriptionResultChanged",
        "status": 20000000,
        "status_message": "GATEWAY|SUCCESS|Success."
```

```
    },
    "payload": {
        "index": 1,
        "time": 1800,
        "result": "今年双十一",
        "confidence":1.0
    }
}
```

header对象参数同上述表格说明，name为TranscriptionResultChanged表示句子的中间识别结果。payload对象参数说明：

参数	类型	说明
index	Integer	句子编号，从1开始递增
time	Integer	当前已处理的音频时长， 单位是毫秒
result	String	当前句子的识别结果
confidence	Double	当前句子识别结果的置信度， 取值范围[0.0，1.0]， 值越大表示置信度越高

- SentenceEnd事件表示服务端检测到了一句话的结束，并附带返回该句话的识别结果，如：

```
{
    "header": {
        "message_id": "05450bf69c53413f8d88aed1ee600e93",
        "task_id": "640bc797bb684bd69601856513079df5",
        "namespace": "SpeechTranscriber",
        "name": "SentenceEnd",
        "status": 20000000,
        "status_message": "GATEWAY|SUCCESS|Success."
    },
    "payload": {
        "index": 1,
        "time": 3260,
        "begin_time": 1800,
        "result": "今年双十一我要买电视",
        "confidence": 1.0
```

```
        }
    }
```

header对象参数同上述表格说明，name为SentenceEnd表示识别到句子的结束。
payload对象参数说明：

参数	类型	说明
index	Integer	句子编号，从1开始递增
time	Integer	当前已处理的音频时长， 单位是毫秒
begin_time	Integer	当前句子对应的SentenceBegin事件的时间，单位是毫秒
result	String	当前的识别结果
confidence	Double	当前句子识别结果的置信度， 取值范围[0.0，1.0]， 值越大表示置信度越高

3.stop and complete

通知服务端语音数据发送完成，服务端识别结束后通知客户端识别完毕

服务状态码

在服务的每一次响应中，都包含status字段，即服务状态码，状态码各种取值含义如下：

通用错误：

错误码	原因	解决办法
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
40000003	令牌过期或无效的参数	首先检查使用的令牌是否过期， 然后检查参数值设置是否合理

40000004	空闲超时	确认是否长时间没有发送数据到服务端
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
40000000	默认的客户端错误码	查看错误消息或提交工单
50000000	默认的服务端错误	如果偶现可以忽略， 重复出现请提交工单
50000001	内部GRPC调用错误	如果偶现可以忽略， 重复出现请提交工单

网关错误：

错误码	原因	解决办法
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

Meta错误：

错误码	原因	解决办法
40020105	应用不存在	检查应用appKey是否正确， 是否与令牌归属同一个账号

实时语音识别：

错误码	原因	解决办法
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的实时率来发包
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来发包
41040204	客户端调用方法异常	客户端应该先调用发送请求接口， 在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的范围在[200-2000]
51040101	服务端内部错误	未知错误
51040102	保留参数	ASR_SERVICE_UNAVAILABLE
51040103	实时语音识别服务不可用	需要查看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	具体需要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

SDK

专有云只提供了Java和C++SDK，如果您需要使用移动端的Android或iOSSDK，请参考公共云使用方式：

- Android: https://help.aliyun.com/document_detail/84705.html
- iOS: https://help.aliyun.com/document_detail/84615.html

Java SDK 2.0

1.下载安装

实时语音识别的目录为nls-example-transcriber，Demo依赖的jar包存放在lib目录，请参考pom文件添加的jar包依赖。

2.关键接口

- NlsClient：语音处理client，相当于所有语音相关处理类的factory，全局创建一个实例即可。线程安全。
- SpeechTranscriber：实时语音识别类，设置请求参数，发送请求及声音数据。非线程安全。
- SpeechTranscriberListener：实时语音识别结果监听类，监听识别结果。非线程安全。

更多介绍参见API文档链接: Java API接口说明

3.SDK 调用注意事项

1. NlsClient对象创建一次可以重复使用，每次创建消耗性能。NlsClient使用了netty的框架，创建时比较消耗时间和资源，但创建之后可以重复利用。建议调用程序将NlsClient的创建和关闭与程序本身的生命周期结合。
2. SpeechTranscriber对象不能重复使用，一个识别任务对应一个SpeechTranscriber对象。例如有N个音频文件，则要进行N次识别任务，创建N个SpeechTranscriber对象。
3. 实现的SpeechTranscriberListener对象和SpeechTranscriber对象是一一对应的，不能将一个SpeechTranscriberListener对象设置到多个SpeechTranscriber对象中，否则不能区分是哪个识别任务。

4.Demo

Demo文件在SDK目录下的nls-example-transcriber子目录中：

文件名	路径	说明
SpeechTranscriberDemo.java	src/main/java/com/alibaba/nls/client	实时语音识别单线程Demo
SpeechTranscriberMultiThreadDemo.java	src/main/java/com/alibaba/nls/client	实时语音识别多线程Demo

SpeechTranscriberWithMicrophoneDemo.java	src/main/java/com/alibaba/nls/client	实时语音识别-读取麦克风Demo

说明1： 专有云下，appkey为default，token为default，url为ws://gateway所在IP:8101/ws/v1。

说明2： Demo中使用的音频文件为16000Hz采样率，请确认是否与使用的模型匹配，以获取正确的识别结果；如果不匹配，请使用适合模型的音频文件进行测试。

nls-sample-16k.wav

说明3： 高级参数设置方法
专有云需要通过代码设置这些高级参数，这些参数没有对应的set方法，需要通过SpeechTranscriber类中的addCustomedParam方法设置：

```
public void addCustomedParam(String key, Object value)
```

以类热词设置为例，设置方式如下:

```
Map<String,String> classVocabs=new HashMap<>();
classVocabs.put("PERSON","******"); // ******为类热词表id
classVocabs.put("ADDRESS","******");// ******为类热词表id
transcriber.addCustomedParam("class_vocabulary_id",classVocabs);
```

说明4（仅限专有云2.5及以上版本）： Demo默认使用的是静态填写服务器地址的方式，使用VIPServer方式，SDK可动态获取存活的服务器地址，然后直连该服务器进行交互。使用步骤如下（可参考nls-example-vipclient目录下的NlsVipServerClientDemo）：

1. 登录到APES页面获取gateway域名。
2. 项目中引入 nls-vipserver-client 此库外网没有，需从从代码示例nls-example-vipclient/lib中找到该库极其依赖自行导入项目
3. 修改代码 NlsVipServerClient 继承自NlsClient，可以作为NlsClient初始化SpeechSynthesizer等类。SpeechSynthesizer创建成功后使用方式不变。
   NlsVipServerClient初始化方式如下：

```
String vipServer = "您的APES所在服务IP";
String vipKey = "default.gateway.vipserver";
String token = "default";

NlsVipServerClient client = new NlsVipServerClient(vipServer, vipKey, token);
```

```
SpeechTranscriber transcriber = new SpeechTranscriber(client, getSynthesizerListener());
```

登录到APES页面，上面代码中的三个参数vipServer，vipKey分别对应下面截图中的各个部分：

C++ SDK 2.0

C++ SDK 2.0实时语音识别提供了**异步识别方式**和**同步识别方式**。异步识别方式通过设置回调函数来获取识别结果，数据的发送和识别结果的获取运行在不同的线程中；同步识别方式通过get接口即可获取识别结果，数据的发送和识别结果的获取可以运行在同一个线程中。

1.下载安装

C++ SDK的压缩文件包含以下几个部分：

- CMakeLists.txt demo工程的CMakeList文件。
- build 编译目录。
- demo 包含demo.cpp，各语音服务配置文件。各文件描述见下表：

文件名	描述
sdkDemo.cpp	windows专用，默认为一句话识别功能demo，如需可自行替换成其它功能(编码格式：UTF-8 代签名)
speechRecognizerDemo.cpp	一句话异步识别demo
speechRecognizerSyncDemo.cpp	一句话同步识别demo
speechSynthesizerDemo.cpp	语音合成demo
speechTranscriberDemo.cpp	实时语音异步识别demo

speechTranscriberSyncDemo.cpp	实时语音同步识别demo
testX.wav	测试音频

- include 包含sdk头文件，以及部分第三方头文件。各文件描述见下表

文件名	描述
openssl	openssl
pthread	pthread线（windows下使用）
uuid	uuid(linux下使用)
opus	opus
jsoncpp	jsoncpp
curl	nls SDK并不依赖curl，仅用于demo中，用以获取token)
nlsCommonSdknls	nls sdk并不依赖nlsCommonSdk，仅用于demo中，用以获取token
nlsClient.h	SDK实例
nlsEvent.h	事件说明
speechRecognizerRequest.h	一句话异步识别接口
speechRecognizerSyncRequest.h	一句话同步识别接口
speechSynthesizerRequest.h	语音合成接口
speechTranscriberRequest.h	实时语音异步识别接口
speechTranscriberSyncRequest.h	实时语音同步识别接口
iNlsRequest.h	Request基础

- lib 包含sdk，以及第三方依赖库。linux.tar.gz为linux平台lib压缩包。windows.zip为windows平台lib压缩包。其中根据平台不同，可以选择linux版本libnlsCppSdk.so(glibc2.5及以上，Gcc4，Gcc5)，windows(x86/x64)版本nlsCppSdk.dll（VS2013、VS2015）。
- readme.txt SDK说明。

- release.log 版本说明。
- version 版本号。
- build.sh demo编译脚本。

2.依赖库

SDK 依赖 openssl(l-1.0.2j)，opus(1.2.1)，jsoncpp(0.10.6)，uuid(1.0.3)，pthread(2.9.1)。
依赖库放置在 path/to/sdk/lib 下。

注意：
path/to/sdk/lib/linux/uuid仅在linux下使用。path/to/sdk/lib/windwos/1x.0/pthread仅在windows下使用。

3.编译运行

```
运行编译脚本build.sh：
1. 请确认本地系统以安装Cmake，最低版本3.1、Glibc 2.5、Gcc 4.1.2及以上
2. cd path/to/sdk/lib
3. tar -zxvpf linux.tar.gz
4. cd path/to/sdk
5. 执行./build.sh 编译demo
6. 编译完毕，进入path/to/sdk/demo目录。可以看见以生成三个demo可执行程序：srDemo(一句话异步识别)、srSyncDemo(一句话同步识别)、stDemo(实时音频异步识别)、stSyncDemo(实时音频同步识别)、syDemo(语音合成)。
7. 执行[./stDemo service-url]

如果不支持cmake，可尝试手动编译：
1: cd path/to/sdk/lib
2: tar -zxvpf linux.tar.gz
3: cd path/to/sdk/demo
4: g++ -o stDemo speechTranscriberDemo.cpp -I../include -L../lib/linux -lnlsCppSdk -lnlsCommonSdk -lcurl -lssl -lcrypto -lopus -lpthread -luuid -ljsoncpp -D_GLIBCXX_USE_CXX11_ABI=0
5: export LD_LIBRARY_PATH=path/to/sdk/lib/linux/
6: ./stDemo service-url

Windows平台需要用户自己搭建工程。
```

注意：

1. linux环境下，运行环境最低要求：Glibc 2.5及以上， Gcc4、Gcc5
2. windows下，目前支持VS2013，VS2015

4.关键接口

基础接口:

- NlsClient：语音处理Client，相当于所有语音相关处理类的Factory，全局创建一个实例即可。
- NlsEvent：事件对象，用户可以从中获取Request状态码、云端返回结果、失败信息等。

异步识别接口:

- SpeechTranscriberRequest：实时音频流识别请求对象。
- SpeechTranscriberCallback：回调事件函数集合的对象，语音结果、异常等回调的统一入口。

同步识别接口:

- SpeechTranscriberSyncRequest: 一句话识别同步请求对象。

更多介绍参见api文档链接:C++ API接口说明

5.C++ SDK自定义错误码

错误码	错误描述	解决方案
10000001	SSL: couldn't create a .....!	建议重试
10000002	openssl官方错误描述	根据描述提示处理之后， 建议重试
10000003	系统错误描述	根据系统错误描述提示处理
10000004	URL: The url is empty.	检查是否设置 云端URL地址
10000005	URL: Could not parse WebSocket url	检查是否正确设置 云端URL地址
10000006	MODE: unsupport mode.	检查时都正确设置了语音功能模式
10000007	JSON: Json parse failed.	服务端发送错误响应内容， 请提供task_id， 并反馈给阿里云
10000008	WEBSOCKET: unkown head type.	服务端发送错误WebSocket类型，请提供task_id， 并反馈给阿里云
10000009	HTTP: connect failed.	与云端连接失败，请检查网络， 在重试

HTTP协议官方状态码	HTTP: Got bad status.	根据HTTP协议官方描述提示处理
系统错误码	IP: ip address is not valid.	根据系统错误描述提示处理
系统错误码	ENCODE: convert to utf8 error.	根据系统错误描述提示处理
10000010	please check if the memory is enough	内存不足. 请检查本地机器内存
10000011	Please check the order of execution	接口调用顺序错误 (接收到Failed/complete事件时，SDK内部会关闭连接。 此时在调用send会上报错误。)
10000012	StartCommand/StopCommand Send failed	参数错误. 请检查参数设置是否正确
10000013	The sent data is null or dataSize <= 0.	发送错误. 请检查发送参数是否正确
10000014	Start invoke failed.	start超时错误. 请调用stop，释放资源，重新开始识别流程.
10000015	connect failed等	connect失败. 释放资源，重新开始识别流程.

6.Demo

说明1： 专有云下，appkey为default，token为default，url为ws://gateway所在IP:8101/ws/v1。

说明2： Demo中使用的音频文件为16000Hz采样率，请确认是否与使用的模型匹配，以获取正确的识别结果；如果不匹配，请使用适合模型的音频文件进行测试。

说明3： 高级参数设置方法

专有云需要通过代码设置这些高级参数，这些参数没有对应的set方法，需要通过SpeechTranscriberRequest类中的setPayloadParam方法设置：

```
/**
* @brief 参数设置
* @param value 参数
* @return 成功则返回0，否则返回-1
```

```
/
int setPayloadParam(const char value);
```

以类热词为例：

```
// "*************" 表示类热词表id
request->setPayloadParam("{\"class_vocabulary_id\":\"{\"PERSON\":\"*********
***\", \"ADDRESS\":\"************\"}\"}");
```

说明4（仅限专有云2.5及以上版本，Linux环境）： Demo默认使用的是静态填写服务器地址的方式，使用VIPServer方式，SDK可动态获取存活的服务器地址，然后直连该服务器进行交互。使用步骤如下（请参考vipServerDemo文件）：

1. 登录到APES页面获取gateway域名。
2. 在调用setUrl()之前，使用vipServerGetIp()获取语音服务ip地址即可。

```
/**
 * @brief 获取ip
 * @param vipServerDomain 地址服务器
 * @param port 地址服务器端口
 * @param targetDomain     语音服务域名
 * @param nlsServerUrl     语音服务目标url
 * @return 成功返回0, 失败返回-1
 */
static int vipServerGetIp(const std::string& vipServerDomain, const int vipSer
verPort, const std::string& targetDomain, std::string& nlsServerUrl);
```

Demo修改方式：

```
string nlsUrl;
// 获取gateway服务url
if ( -1 == NlsClient::getInstance()->vipServerGetIp(vipServerDomain, vipServe
rPort, targetDomain, nlsUrl)) {
    return NULL;
}

request->setUrl(nlsUrl.c_str());
........
```

登录到APES页面，上面代码中的参数vipServerDomain，targetDomain分别对应下面截图中的红框部分。参数port可以设置为80。如果80不可用，请咨询部署人员。