智能语音交互

快速入门

快速入门

欢迎开启智能语音交互（Natural Language Service）之旅！阿里云智能语音交互可提供语音识别（ASR），语音合成(TTS)，智能问答(QA)等服务。

- ASR ：语音识别服务，提供语音转文本服务。
- TTS ：文本转语音服务，提供将文本转为普通话语音的语音合成功能。
- QA ：智能对话平台，用户能够快速搭建自己的对话机器人。目前为1.0版，支持单轮智能问答。

账号和服务申请步骤：

进入阿里云官网，申请阿里云账号。若已有阿里云账号，请忽略之。

到“智能语音交互服务”页面，点击『立即开通』。

在跳转后的页面，点击『立即购买』，购买语音服务。目前为公测阶段，10路并发以下的试用不收费。

在数加-Access Key页面创建并获取您的Access Key ID 和 Secret。您可以在开发调试阶段使用这个Key，不过为了您的账号安全，强烈建议您按照下面一节的步骤创建子账号，使用子账号的Access Key 调用智能语音服务。

账号安全

上一步数加-Access Key页面上创建的Access Key对应您的主账号，有权使用所有您主账号上开通的服务。强烈建议不要直接使用这个Key，而应该在RAM系统创建子账号，使用子账号的Access Key访问语音服务。

具体操作步骤如下：

1. 访问RAM子账号管理页面
2. 点击右上角“新建用户”按钮，在弹出的对话框中填写子账号用户名，其他为非必填项
3. 勾选底部“为该用户自动生成AccessKey”复选框，点击“确定按钮”
4. 在新弹出的对话框中点击“保存AK信息”，把该子账号的Access Key ID和Secret信息保存到本地。
   注意：这是查看和保存 Access Key Secret的唯一机会
5. 下面就可以使用此 Key 调用智能语音服务了

“智能语音交互服务”调用过程

包括以下3个步骤：

- Step1: 账号和服务申请
- Step2: 根据具体的语音使用场景，确定需要调用的服务类别（ASR、TTS、NLP），例如是语音识别成文字，还是文字合成语音或其他。
- Step3: 下载不同开发平台下的SDK和DEMO（Java、C++、Android、iOS、RESTful API等），按照API文档步骤，在DEMO工程中填写对应参数，开始进行服务调用。

下面以最常用的”一句话识别”服务给大家做一个入门级的示例，为您介绍如何快速使用“智能语音交互”的各种服务。

1. 账号和服务申请，获取Access Key ID和Access Key Secret两个数加认证参数。

2. 语音识别服务下的”一句话识别”服务支持的app_key如下表，选择“社交领域”的app_key为nls-service。

一句话识别 app_key	语音数据格式	领域
nls-service	16kHz采样 16bit	社交聊天
nls-service-streaming	16kHz采样 16bit	社交聊天
nls-service-tv	16kHz采样 16bit	家庭娱乐
nls-service-shopping	16kHz采样 16bit	电商购物领域
nls-service-care	16kHz采样 16bit	智能客服服务领域

注：

(1) “SDK支持的结果返回方式”式包括“非流式”和“流式”两种模式，“非流式”简单来说就是用户整句话说完后返回识别结果，“流式”模式下用户一边说话一边返回识别结果。

(2) “一句话识别”支持的领域包括：社交聊天、家庭娱乐、电商购物、智能客服等。用户可针对具体的使用场景选择对应领域的app_key。

3. 下载JAVA SDK和DEMO，开始语音识别。

3.1 运行demo来测试语音服务

到“一句话识别”服务的Java SDK页面下载对应的下载包，包括JAVA SDK和DEMO工程。

打开Java DEMO工程中的AsrDemo.java方法，将其中的app_key、Access Key ID、Access Key Secret 替换成自己的账号信息，然后直接 run->java application即可。

```
    NlsRequest req = new NlsRequest();
req.setApp_key("app_key"); // 替换为选定的app_key
req.setAsr_sc("pcm"); // 这里为测试准备的语音文件是pcm语音文件。
```

```
req.authorize("Access Key ID", "Access Key Secret"); // 替换为在数加平台申请到的"Access Key ID"和 "Access Key Secret"
```

若执行成功，将会输出【” result” ：“你好小云” 】的log信息，至此，您已成功完成了一次语音识别服务的调用。

3.2 DEMO工程的简要介绍

3.2.1 发送语音请求

由AsrDemo.java的方法startAsr()进行处理，首先提取语音数据，然后创建语音识别请求，将提取的语音分批发送至服务端。

分批发送的好处是可以在您进行语音收集的同时，服务器就开始处理识别，这样当语音结束的时候可以最快地得到识别结果并返回。

3.2.2 接收语音识别结果

由AsrDemo.java的回调方法onMessageReceived()进行处理，用于监听服务器的返回，在这个demo中，服务端返回的识别结果json字段参数如下：

```
{
"status" : "1",// 服务器状态，0为失败，非零为成功
"id" : "",
"finish" : "1",// 0为未结束，非零为结束，识别是否已经结束
"results" : {
"asr_out" : {
"result" : "你好小云",// 语音识别结果
"status" : 1,
"finish" : 1,
"version" : "4.0"
},
"out" : {}//保留字段
},
"bstream_attached" : false,
"version" : "4.0"
}
```

参数中您可以重点关注字段 **asr_out**，其中的” result” ：“你好小云” 即为语音识别结果。

鉴权方法说明

通过RESTful API调用智能语音服务，首先您需要拥有AccessKey（ 点击这里创建和管理您的AccessKey ），AccessKey相当于您访问阿里云产品的口令，拥有您完整的权限，请您妥善保管、避免泄露，并定期更换您的AccessKey！

1. 鉴权请求构造

- 智能语音服务采用的鉴权方式与阿里云数加鉴权相似，但实现上稍有区别，主要是**请求URL不参与鉴权。**
- 鉴权签名是指构造一个Authorization 字段，并设置到HTTP的Header当中。服务端通过验证Authorization是否正确来确认请求来自授权用户。

1.1 整体校验规则

```
Authorization =  Dataplus AccessKeyId + ":" + Signature
Signature = Base64( HMAC-SHA1( AccessSecret, UTF-8-Encoding-Of(StringToSign) ) ) //具体实现参考1.2
StringToSign =
//HTTP协议header
Method + "\n" + // POST|GET|PUT...
Accept + "\n" +
Body-Digest + "\n" + // Base64(Md5(body))
Content-Type + "\n" +
Date ;
//注意官方鉴权中的url在NLS服务不参与鉴权签名
```

1.2 签名计算方法

API请求使用标准的Authorization头来签名自己的请求，请求格式如下:

```
Authorization: Dataplus AccessKeyId:Signature
```

签名算法遵循RFC 2104HMAC-SHA1规范，要签名的元素是请求自身的一些参数，由于每个API请求基本不同，所以签名的结果也不尽相同。

在鉴权过程中参与的字段信息，所有字符的编码格式请采用UTF-8,并按顺序排列；若值不存在则以”\n”补齐。

参与签名字段如下：

值	类型名	描述
method	String	Http请求类型，可以是 GET, POST, DELETE, PUT
accept	String	定义http接收的返回值类型
content-type	String	定义http发送的数据类型
bodyString	String	请求数据，POST请求应该放置于http body中
date	String	HTTP 1.1协议中规定的GMT时间，例如：Wed, 05 Sep. 2012 23:00:00 GMT，请注意同步网络时间，上下在5秒钟以内

1.2.1 计算body的MD5及Base64编码

```
//以java为例
String bodyMd5 = MD5Base64(body);
```

POST/PUT请求

- body为string 类型非空字符串：计算body的MD5，再将MD5值做Base64编码。
- body为空：将bodyMd5设为”“，即空字符串。
- body为REST ASR中二进制语音：需要对语音做两次MD5Base64操作，bodyMd5 = MD5Base64(MD5Base64(body))。

GET/DELETE请求

- 将bodyMd5设为”“，即空字符串。

1.2.2 拼装请求验证串

```
String stringToSign = method + "\n" + accept + "\n" + bodyMd5 + "\n" + content_type + "\n" + date ;
```

- REST ASR接口content_type为『”audio/“+audioFormat+”;samplerate=”+sampleRate;』；
- 批量测试接口content_type为『multipart/form-data』。

1.2.3 计算Authorization

把1.2.2 中得到的stringToSign与Access Secret一起计算HMAC-SHA1后取Base64的值，然后与Access Id一起拼装为Authorization。Authorization会放在HTTP的Header中。

```
// 1.计算 HMAC-SHA1
String signature = HMACSha1(stringToSign, ak_secret);
// 2.得到 authorization
String authorization = "Dataplus " + ak_id + ":" + signature;
```

1.3 鉴权签名范例

为了方便用户校验自己的鉴权模块是否正确，我们给出一个鉴权签名范例。我们以POST请求为例，传入一个String类型字符串”Alibaba”作为body值参与鉴权。我们设置鉴权的几个字段值如下：

```
String method = "POST";
String accept = "application/json";
String content_type = "application/json";
String Date = "Wed, 31 May 2017 08:51:26 GMT";
```

1.3.1 计算body的MD5及Base64编码

计算”Alibaba”的MD5Base64的值为：”AsdYv2nI4ijTfKYmKX4h/Q==”。

```
public static String MD5Base64(String s) throws UnsupportedEncodingException {
if (s == null) {
return null;
}
if (s == ""){
return "";
}
String encodeStr = "";

//string 编码必须为utf-8
byte[] utfBytes = s.getBytes("UTF-8");

MessageDigest mdTemp;
try {
mdTemp = MessageDigest.getInstance("MD5");
mdTemp.update(utfBytes);
byte[] md5Bytes = mdTemp.digest();
BASE64Encoder b64Encoder = new BASE64Encoder();
encodeStr = b64Encoder.encode(md5Bytes);
} catch (Exception e) {
throw new Error("Failed to generate MD5 : " + e.getMessage());
}
return encodeStr;
}
```

1.3.2 拼装请求验证串

拼装stringToSign的值为：“POST\napplication/json\nAsdYv2nI4ijTfKYmKX4h/Q==\napplication/json\nWed, 31 May 2017 08:51:26 GMT”。

```
String stringToSign = method + "\n" + accept + "\n" + bodyMd5 + "\n" + content_type + "\n" + date;
```

1.3.3 计算Authorization

计算stringToSign和Access secret 的鉴权值为：signature = “nPuAlFx4pPPsGT55oXeigM60paw=”。

```
String signature = HMACSha1(stringToSign, ak_secret);

public static String HMACSha1(String data, String key) {
String result;
try {

SecretKeySpec signingKey = new SecretKeySpec(key.getBytes(), "HmacSHA1");
Mac mac = Mac.getInstance("HmacSHA1");
mac.init(signingKey);
byte[] rawHmac = mac.doFinal(data.getBytes());
result = (new BASE64Encoder()).encode(rawHmac);
} catch (Exception e) {
throw new Error("Failed to generate HMAC : " + e.getMessage());
}
```

```
return result;
}
```

2. 请求与响应

2.1 请求

NLS REST服务基于标准的HTTP协议，可设置的值包含Header和Body等。

2.1.1 设置Header

名称	含义	类型	是否必须
Authorization	用于验证请求合法性的认证信息，采用AccessKeyId:Signature的形式。	字符串	是
Content-Type	RFC 2616中定义的HTTP请求内容类型。	字符串	是
Date	HTTP 1.1协议中规定的GMT时间，例如：Wed, 05 Sep. 2012 23:00:00 GMT	字符串	是
Accept	客户端需要的返回值类型。	字符串	是
Content-Length	body长度	int	否，REST ASR服务中必须设置

- REST ASR接口content_type为『”audio/“+audioFormat+”;samplerate=”+sampleRate;』；
- 批量测试接口content_type为『multipart/form-data』。
- REST TTS接口content_type为『”audio/“+audioFormat+”;samplerate=”+sampleRate;』；

2.1.2 设置Body

根据Content-Type的不同，body的参数也各有不同：

类型	说明
application/json	String类型json
text/plain	String类型文本，tts合成使用
audio/*	byte[]二进制语音数据
multipart/form-data	form表单

2.2 响应

- HTTP状态200表示请求成功
- HTTP状态4XX表示客户端错误，并在body中写明具体错误码和错误消息（error_code，error_message）
- HTTP状态5XX表示服务端错误，并在body中写明具体错误码和错误消息（error_code，error_message）
- 若响应格式为JSON格式，JSON结构中会包含request_id字段，用来区分每次请求（一些老的API使用id做为请求标识）

错误消息示例

```
{
"id": "be053bf9af0e406dafa8249631372d53",
"request_id": "be053bf9af0e406dafa8249631372d53",
"error_code": 080101,
"error_message": "REQUEST_PARSE_ERROR(Failed to parse json object!)"
}
```

为了兼容旧API，错误响应中同时包含request_id和id，值相同，后续版本会移除id字段。

2.3 参数类型和参数值类型

参数类型表示该参数出现的位置

类型名	描述
Header	表明该参数是出现在HTTP请求的Header中
Path	表明该参数是url路径的一部分，例如/customizations/{customization_id}
Query	表明该参数是url的的查询参数，例如/customizations?base_model=ime
Body	表明该参数为HTTP请求的body部分

参数值类型表示该参数的数据类型

类型名	描述
String	字符串类型
Integer	整数类型
Double	浮点数类型
Object	对象类型，可以有自定义属性
Type[]	数组类型，如String[]，Object[]

附录 各语言编程范例

Java

```
package com.alibaba.vo;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.URL;
import java.net.URLConnection;
import java.security.MessageDigest;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;
import javax.crypto.spec.SecretKeySpec;
import sun.misc.BASE64Encoder;
import javax.crypto.Mac;
@SuppressWarnings("restriction")
public class AESDecode {
/*
* 计算MD5+BASE64
*/
public static String MD5Base64(String s) {
if (s == null)
return null;
String encodeStr = "";
//编码为utf-8
byte[] utfBytes = s.getBytes("UTF-8");
MessageDigest mdTemp;
try {
mdTemp = MessageDigest.getInstance("MD5");
mdTemp.update(utfBytes);
byte[] md5Bytes = mdTemp.digest();
BASE64Encoder b64Encoder = new BASE64Encoder();
encodeStr = b64Encoder.encode(md5Bytes);
} catch (Exception e) {
throw new Error("Failed to generate MD5 : " + e.getMessage());
}
return encodeStr;
}
/*
* 计算 HMAC-SHA1
*/
public static String HMACSha1(String data, String key) {
String result;
try {
SecretKeySpec signingKey = new SecretKeySpec(key.getBytes(), "HmacSHA1");
Mac mac = Mac.getInstance("HmacSHA1");
mac.init(signingKey);
byte[] rawHmac = mac.doFinal(data.getBytes());
result = (new BASE64Encoder()).encode(rawHmac);
```

```
} catch (Exception e) {
throw new Error("Failed to generate HMAC : " + e.getMessage());
}
return result;
}
/*
* 等同于javaScript中的 new Date().toUTCString();
*/
public static String toGMTString(Date date) {
SimpleDateFormat df = new SimpleDateFormat("E, dd MMM yyyy HH:mm:ss z", Locale.UK);
df.setTimeZone(new java.util.SimpleTimeZone(0, "GMT"));
return df.format(date);
}
/*
* 发送POST请求
*/
public static String sendPost(String url, String body, String ak_id, String ak_secret) throws Exception {
PrintWriter out = null;
BufferedReader in = null;
String result = "";
int statusCode = 200;
try {
URL realUrl = new URL(url);
/*
* http header 参数
*/
String method = "POST";
String accept = "json";
String content_type = "application/json";
String path = realUrl.getFile();
String date = toGMTString(new Date());
String bodyMd5 = "";
// 1.对body做MD5+BASE64加密
if(body != null && body != ""){
bodyMd5 = MD5Base64(body);
}
String stringToSign = method + "\n" + accept + "\n" + bodyMd5 + "\n" + content_type + "\n" + date ;
// 2.计算 HMAC-SHA1
String signature = HMACSha1(stringToSign, ak_secret);
// 3.得到 authorization header
String authHeader = "Dataplus " + ak_id + ":" + signature;
// 打开和URL之间的连接
URLConnection conn = realUrl.openConnection();
// 设置通用的请求属性
conn.setRequestProperty("accept", accept);
conn.setRequestProperty("content-type", content_type);
conn.setRequestProperty("date", date);
conn.setRequestProperty("Authorization", authHeader);
// 发送POST请求必须设置如下两行
conn.setDoOutput(true);
conn.setDoInput(true);
// 获取URLConnection对象对应的输出流
out = new PrintWriter(conn.getOutputStream());
// 发送请求参数
out.print(body);
// flush输出流的缓冲
```

```
out.flush();
// 定义BufferedReader输入流来读取URL的响应
statusCode = ((HttpURLConnection)conn).getResponseCode();
if(statusCode != 200) {
in = new BufferedReader(new InputStreamReader(((HttpURLConnection)conn).getErrorStream()));
} else {
in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
}
String line;
while ((line = in.readLine()) != null) {
result += line;
}
} catch (Exception e) {
e.printStackTrace();
} finally {
try {
if (out != null) {
out.close();
}
if (in != null) {
in.close();
}
} catch (IOException ex) {
ex.printStackTrace();
}
}
if (statusCode != 200) {
throw new IOException("\nHttp StatusCode: "+ statusCode + "\nErrorMessage: " + result);
}
return result;
}
/*
* GET请求
*/
public static String sendGet(String url, String ak_id, String ak_secret) throws Exception {
String result = "";
BufferedReader in = null;
int statusCode = 200;
try {
URL realUrl = new URL(url);
/*
* http header 参数
*/
String method = "GET";
String accept = "json";
String content_type = "application/json";
String path = realUrl.getFile();
String date = toGMTString(new Date());
// 1.对body做MD5+BASE64加密
// String bodyMd5 = MD5Base64(body);
String stringToSign = method + "\n" + accept + "\n" + "" + "\n" + content_type + "\n" + date ;
// 2.计算 HMAC-SHA1
String signature = HMACSha1(stringToSign, ak_secret);
// 3.得到 authorization header
String authHeader = "Dataplus " + ak_id + ":" + signature;
// 打开和URL之间的连接
```

```
URLConnection connection = realUrl.openConnection();
// 设置通用的请求属性
connection.setRequestProperty("accept", accept);
connection.setRequestProperty("content-type", content_type);
connection.setRequestProperty("date", date);
connection.setRequestProperty("Authorization", authHeader);
connection.setRequestProperty("Connection", "keep-alive");
// 建立实际的连接
connection.connect();
// 定义 BufferedReader输入流来读取URL的响应
statusCode = ((HttpURLConnection)connection).getResponseCode();
if(statusCode != 200) {
in = new BufferedReader(new InputStreamReader(((HttpURLConnection)connection).getErrorStream()));
} else {
in = new BufferedReader(new InputStreamReader(connection.getInputStream()));
}
String line;
while ((line = in.readLine()) != null) {
result += line;
}
} catch (Exception e) {
e.printStackTrace();
} finally {
try {
if (in != null) {
in.close();
}
} catch (Exception e) {
e.printStackTrace();
}
}
if (statusCode != 200) {
throw new IOException("\nHttp StatusCode: "+ statusCode + "\nErrorMessage: " + result);
}
return result;
}
public static void main(String[] args) throws Exception {
// 发送POST请求示例
String ak_id1 = "NMV.....5jv"; //用户ak
String ak_secret1 = "Fgs.....3zu"; // 用户ak_secret
String url = "https://shujuapi.aliyun.com/org_code/service_code/api_name";
String body = "{\"param1\": \"xxx\", \"param2\":\"xxx\"}";
System.out.println("response body:" + sendPost(url, body, ak_id, ak_secret));
// 发送GET请求
String ak_id1 = "NMV.....5jv"; //用户ak
String ak_secret1 = "Fgs.....3zu"; // 用户ak_secret
String url1 = "https://shujuapi.aliyun.com/org_code/service_code/api_name?param1=xxx&param2=xxx";
System.out.println("response body:" + sendGet(url1, ak_id1, ak_secret1));
}
}
```

PHP

```
<?php
```

```
$akId = "*************";
$akSecret = "******************";
//更新api信息
$url = "https://shujuapi.aliyun.com/org_code/service_code/api_name?param1=xxx&param2=xxx";
$options = array(
'http' => array(
'header' => array(
'accept'=> "application/json",
'content-type'=> "application/json",
'date'=> gmdate("D, d M Y H:i:s \G\M\T"),
'authorization' => ''
),
'method' => "GET", //可以是 GET, POST, DELETE, PUT
'content' => '' //如有数据，请用json_encode()进行编码
)
);
$http = $options['http'];
$header = $http['header'];
$urlObj = parse_url($url);
if(empty($urlObj["query"]))
$path = $urlObj["path"];
else
$path = $urlObj["path"]."?".$urlObj["query"];
$body = $http['content'];
if(empty($body))
$bodymd5 = $body;
else
$bodymd5 = base64_encode(md5($body,true));
$stringToSign = $http['method']."\n".$header['accept']."\n".$bodymd5."\n".$header['content-
type']."\n".$header['date'];
$signature = base64_encode(
hash_hmac(
"sha1",
$stringToSign,
$akSecret, true));
$authHeader = "Dataplus "."$akId".":"."$signature";
$options['http']['header']['authorization'] = $authHeader;
$options['http']['header'] = implode(
array_map(
function($key, $val){
return $key.":".$val."\r\n";
},
array_keys($options['http']['header']),
$options['http']['header']));
$context = stream_context_create($options);
$file = file_get_contents($url, false, $context );
echo($file);
?>
```

Python2.7

```
#!/usr/bin/python
# -*- coding:utf-8 -*-
from urlparse import urlparse
```

```
import datetime
import base64
import hmac
import hashlib
import json
import urllib2
def get_current_date():
date = datetime.datetime.strftime(datetime.datetime.utcnow(), "%a, %d %b %Y %H:%M:%S GMT")
return date

def to_md5_base64(strBody):
hash = hashlib.md5()
hash.update(body.encode('utf-8'))
return hash.digest().encode('base64').strip()

def to_sha1_base64(stringToSign, secret):
hmacsha1 = hmac.new(secret.encode('utf-8'), stringToSign.encode('utf-8'), hashlib.sha1)
return base64.b64encode(hmacsha1.digest()).decode('utf-8')
ak_id = '<用户的AK_ID>'
ak_secret = '<用户的AK_SECRET>'
options = {
'url': '<请求的url>',
'method': 'POST',
'body': json.dumps({"name": "hello"}, separators=(',', ':')),
'headers': {
'accept': 'application/json',
'content-type': 'application/json',
'date': get_current_date(),
'authorization': ''
}
}
# options = {
# 'url': '<请求的url>',
# 'method': 'GET',
# 'headers': {
# 'accept': 'application/json',
# 'content-type': 'application/json',
# 'date': get_current_date(), # 'Sat, 07 May 2016 08:19:52 GMT', # get_current_date(),
# 'authorization': ''
# }
# }
body = ''
if 'body' in options:
body = options['body']
print body
bodymd5 = ''
if not body == '':
bodymd5 = to_md5_base64(body)
print bodymd5
# REST ASR 接口，需要做两次鉴权
#bodymd5 = to_md5_base64(bodymd5)
stringToSign = options['method'] + '\n' + options['headers']['accept'] + '\n' + bodymd5 + '\n' +
options['headers']['content-type'] + '\n' + options['headers']['date']
signature = to_sha1_base64(stringToSign, ak_secret)
print stringToSign
authHeader = 'Dataplus ' + ak_id + ':' + signature
```

```
options['headers']['authorization'] = authHeader
print authHeader
request = None
method = options['method']
url = options['url']
print method
print url
if 'GET' == method or 'DELETE' == method:
request = urllib2.Request(url)
elif 'POST' == method or 'PUT' == method:
request = urllib2.Request(url, body)
request.get_method = lambda: method
for key, value in options['headers'].items():
request.add_header(key, value)
try:
conn = urllib2.urlopen(request)
response = conn.read()
print response
except urllib2.HTTPError, e:
print e.read()
raise SystemExit(e)
```

Python 3.5

```
 #!/usr/bin/python
# -*- coding:utf-8 -*-
import hashlib
import urllib.request
import hmac
import base64
import datetime
import ssl
class http_proxy:
"""
Http工具类，封装了鉴权
"""
def __init__(self, ak_id, ak_secret):
self.__ak_id = ak_id
self.__ak_secret = ak_secret
def __current_gmt_time(self):
date = datetime.datetime.strftime(datetime.datetime.utcnow(), "%a, %d %b %Y %H:%M:%S GMT")
return date
def __md5_base64(self, strbody):
hash = hashlib.md5()
hash.update(strbody.encode('utf-8'))
print(hash.digest())
return base64.b64encode(hash.digest()).decode('utf-8')
def __sha1_base64(self, str_to_sign, secret):
hmacsha1 = hmac.new(secret.encode('utf-8'), str_to_sign.encode('utf-8'), hashlib.sha1)
return base64.b64encode(hmacsha1.digest()).decode('utf-8')
def send_request(self, url, body):
gmtnow = self.__current_gmt_time()
print(gmtnow)
body_md5 = self.__md5_base64(body)
```

```
print(body_md5)
str_to_sign = "POST\napplication/json\n" + body_md5 + "\napplication/json\n" + gmtnow
print(str_to_sign)
signature = self.__sha1_base64(str_to_sign, self.__ak_secret)
print(signature)
auth_header = "Dataplus " + self.__ak_id + ":" + signature
print(auth_header)
ssl._create_default_https_context = ssl._create_unverified_context
req = urllib.request.Request(url)
req.add_header("Accept", "application/json")
req.add_header("Content-Type", "application/json")
req.add_header("Date", gmtnow)
req.add_header("Authorization", auth_header)
data = body.encode('utf-8')
f = urllib.request.urlopen(req, data)
return f.read().decode('utf-8')
```

Go

```
package utils
import (
"fmt"
"net/http"
"io/ioutil"
"time"
"bytes"
"crypto/md5"
"crypto/hmac"
"crypto/sha1"
"encoding/base64"
)
type HttpProxy struct {
AkId string
AkSecret string
}
func (proxy *HttpProxy)gmttime() string {
location,_ := time.LoadLocation("GMT")
now := time.Now().In(location)
return now.Format("Mon, 02 Jan 2006 15:04:05 GMT")
}
func (proxy *HttpProxy)md5base64(strbody string) string {
m := md5.New()
m.Write([]byte(strbody))
return string(base64.StdEncoding.EncodeToString(m.Sum(nil)))
}
func (proxy *HttpProxy)sha1base64(strToSign string, secret string) string {
key := []byte(secret)
h := hmac.New(sha1.New, key)
h.Write([]byte(strToSign))
return string(base64.StdEncoding.EncodeToString(h.Sum(nil)))
}
func (proxy *HttpProxy)SendRequest(url string, reqBody string) string {
gmtnow := proxy.gmttime()
bodyMd5 := proxy.md5base64(reqBody)
```

```
strToSign := "POST\napplication/json\n" + bodyMd5 + "\napplication/json\n" + gmtnow;
signature := proxy.sha1base64(strToSign, proxy.AkSecret)
authHeader := "Dataplus " + proxy.AkId + ":" + signature
client := &http.Client{}
reqBodyBuf := bytes.NewBuffer([]byte(reqBody))
request, _ := http.NewRequest("POST", url, reqBodyBuf)
request.Header.Set("Accept", "application/json")
request.Header.Set("Content-type", "application/json")
request.Header.Set("Date", gmtnow)
request.Header.Set("Authorization", authHeader)
response, _ := client.Do(request)
fmt.Println(response.StatusCode)
respBody,_ := ioutil.ReadAll(response.Body)
return string(respBody)
}
```

Node.js

```
var request = require('request');
var url = require('url');
var crypto = require('crypto');
var date = new Date().toUTCString()
// 这里填写AK和请求
var ak_id = 'NNV.....5jv';
var ak_secret = 'FGs.....3Zu';
var options = {
url : 'https://shujuapi.aliyun.com/org_code/service_code/api_name?param1=xxx&param2=xxx',
method: 'GET',
body: '',
headers: {
'accept': 'application/json',
'content-type': 'application/json',
'date': date,
'Authorization': ''
}
};
// 这里填写AK和请求
md5 = function(buffer) {
var hash;
hash = crypto.createHash('md5');
hash.update(buffer);
return hash.digest('base64');
};
sha1 = function(stringToSign, secret) {
var signature;
return signature = crypto.createHmac('sha1', secret).update(stringToSign).digest().toString('base64');
};
// step1: 组stringToSign [StringToSign = #{method}\\n#{accept}\\n#{data}\\n#{contentType}\\n#{date}]
var body = options.body || '';
var bodymd5;
if(body === void 0 || body === ''){
bodymd5 = '';
} else {
bodymd5 = md5(new Buffer(body));
```

```
}
console.log(bodymd5)
var stringToSign = options.method + "\n" + options.headers.accept + "\n" + bodymd5 + "\n" +
options.headers['content-type'] + "\n" + options.headers.date;
console.log("step1-Sign string:", stringToSign);
// step2: 加密 [Signature = Base64( HMAC-SHA1( AccessSecret, UTF-8-Encoding-Of(StringToSign) ) )]
var signature = sha1(stringToSign, ak_secret);
// console.log("step2-signature:", signature);
// step3: 组authorization header [Authorization = Dataplus AccessKeyId + ":" + Signature]
var authHeader = "Dataplus " + ak_id + ":" + signature;
console.log("step3-authorization Header:", authHeader);
options.headers.Authorization = authHeader;
console.log('authHeader', authHeader);
// step4: send request
function callback(error, response, body) {
if (error) {
console.log("error", error)
}
console.log("step4-response body:", response.statusCode, body)
}
request(options, callback);
```