经分助手-短信服务接口对接帮助手册

API概览

- **发送短信** 支持在一次请求中向多个不同的手机号码发送同样内容的短信
- **批量发送短信** 支持在一次请求中分别向多个不同的手机号码发送不同的短信。手机号码等参数均为JSON格式，字段个数相同，一一对应，短信服务根据字段在JSON中的顺序判断发往指定手机号码的短信内容

准备工作

登录企业运营平台。 使用企业平台用户登录企业平台，地址：https://opass.infocloud.cc/

大体分为如下步骤

1，根据自己需求创建短信模板

2，获取apiKey和AccessKey

第一步：创建短信模板

1，点击富媒体消息

2，点击左侧 短信发送--》模板管理--》新建模板

创建成功后，会生成模板编号

该模板编号后面短信发送要用到

图片加载失败

第一类，固定内容短信

图片加载失败

第二类，部分带变量的模板

第三类，全变量短信

图片加载失败

第二步，获取接口对接信息（必须），因为涉嫌加密

1，点击富媒体消息

2，点击接口配置-》接口信息

**保存以下三个信息，后面短信发送会用到，很重要

账号

apiKey

AccessKey**


图片加载失败

发送短信

接口签名

【客户端】

1. 签名规则

- 接口授权 apikey、accesskey为企业开户后，可到企业平台进行申请，每个账号对应一个apikey、accesskey
- 数据有效性 加入timestamp（时间戳,13位），以保证在特定的时间内数据提交的数据有效
- 防止重复提交 加入nonce，防止重复提交数据，长度至少10位，需要保证特定时间内的唯一性，以避免重复提交
- 数据防串改 加入sign，结合timestamp、nonce对所有数据进行签名，防止数据被串改。其中apikey、timestamp、nonce、sign这四个字段放入请求头中

2. 请求头部分

- x-api-key: api接入key
- x-sign-method ： 签名方法，支持HmacMD5、HmacSHA1、HmacSHA224、HmacSHA384、HmacSHA512
- x-nonce: 防重复提交(数字)
- x-timestamp: 时间戳(13位)
- x-sign: 签名

3. 签名生成

- 签名字符串　将请求体参数及参数值、请求头部信息（除x-sign）按照数据字典排序后得到一个用来签名的字符串；排序后的字符串eg (必须这个顺序):

phones=手机号1,手机号2&templateCode=审核通过的模板编号&templateParam=["参数1","参数2","参数3"]&x-api-key=你的apiKey&x-nonce=H95a7XT1h6&x-sign-method=HmacSHA256&x-timestamp=1641350184346

- 签名　使用加密算法加secretkey对关键数据签名串进行加密处理，得到签名；
- 最终HTTP请求　将签名相关的所有头加入到原始HTTP请求中，得到最终HTTP请求；

> PostMan自动签名并发送短信
> Java 代码签名并发送短信

接口域名

`https://opassapi.infocloud.cc`
或
推荐使用 `https://opassapi.infocloud.com.cn`

发送短信

`{接口域名}/message/send` 接口是短信发送接口，支持在一次请求中向多个不同的手机号码发送同样内容的短信。
如果您需要在一次请求中分别向多个不同的手机号码发送模版内容变量值不同的短信，请使用批量发送接口。
调用该接口发送短信时，请注意：

- 在一次请求中，最多可以向1000个手机号码发送同样内容的短信。

请求参数

> 请求方式： `post`

名称	类型	是否必选	示例值	描述
phones	String	是	`"phones":"135xxxxxxxx,1588xxxxxx"`	接收短信的手机号码。支持对多个手机号码发送短信，手机号码之间以英文逗号（,）分隔。上限为1000个手机号码。批量调用相对于单条调用及时性稍有延迟。
templateCode	String	是	`"templateCode":"675871161745 17xxxx"`	短信模板编号。**说明** 必须是已添加、并通过审核的短信。``
templateParam	String	否	`"templatePara m":"[\"变量值1 \",\"变量值2 \"]"`	短信模板变量对应的实际值。

upExtendCode	String	否	`"upExtendCode":"32"`	上行短信扩展码，上行短信，指发送给通信服务提供商的短信，用于定制某种服务、完成查询，或是办理某种业务等，需要收费的，按运营商普通短信资费进行扣费。**说明** 无特殊需要此字段的用户请忽略此字段。默认取值范围：1~99
bid	String	否	`"bid":"xxxxxxx"`	客户端业务ID，长度不超过64字符

返回数据

名称	类型	示例值	描述
code	String	200	请求状态码。返回200代表请求成功。其他错误码详见。
message	String	OK	状态码的描述。
data.msgId	String		每一批短信编号。该值不为空，说明平台已经接收提交的短信任务
data.desc	String		状态描述
data.failPhones	string		提交失败的手机号。示例1813757832089,1813759932088,18737 57932087
requestId	String		问题排查时，提供该字段值给短信平台

示例

不带变量

请求示例

```
JSON
1  {"phones":"xxxxx,xxxxx","templateCode":"574005827011772416","upExtendCode":
   "123"}
```

正常返回示例

`JSON` 格式

```
JSON
1  {
2      "code": 200,
3      "data": {
4          "msgId": "574730770464800768"
5      },
6    "requestId": "176e2bfc-a9ab-4096-ae5e-9681b9856992"
7  }
```

带变量

请求示例

```
JSON
1  {"phones":"xxxx,xxxx","templateCode":"574277763210051584","templateParam":"
   [\"你好\"]","upExtendCode":"123"}
```

带变量 – 内容带"\"特殊字符,需要转为"\\"

请求实例

```
JSON
1  {"phones": "xxxx,xxxx",
2      "templateCode": "817436511033106432",
3      "templateParam" : "[\"测\\试一\\下\"]"}
```

正常返回示例

`JSON` 格式

```JSON
{
    "code": 200,
    "data": {
        "msgId": "574730770464800768"
    },
  "requestId": "176e2bfc-a9ab-4096-ae5e-9681b9856992"
}
```

批量发送

`{接口域名}/message/sendBatch` 接口是短信批量发送接口，支持在一次请求中分别向多个不同的手机号码发送不同的短信。手机号码等参数均为JSON格式，字段个数相同，一一对应，短信服务根据字段在JSON中的顺序判断发往指定手机号码的短信内容。

如果您需要往多个手机号码中发送同样内容的短信，请使用 `message/send` 接口实现。

调用该接口发送短信时，请注意：

- 在一次请求中，最多可以向1000个手机号码分别发送短信。

请求参数

> 请求方式： `post`

名称	类型	是否必选	示例值	描述
phonesJson	String	是	`"phonesJson":"[\"1590***0000***\",\"***13500***000\"]"`	接收短信的手机号码，JSON数组格式。手机号码格式：国内短信：11位手机号码。**说明** 验证码类型短信，建议使用接口SendSms单独发送。

templateCode	String	是	`"templateCode":"6758711617 4517xxxx"`	短信模板编号。请在控制台**模板管理**页面**模板编号**一列查看。必须是已添加、并通过审核的模板编号。
templateParamJson	String	否	`"templateParamJson":"[[\"明\"],[\"张三\"]]"`	短信模板变量对应的实际值，JSON格式。关于模板中无参数的情况下该字段传参格式示例：["",""]（""个数和群发的号码个数对应）。如果JSON中需要带换行符，请参照标准的JSON协议处理；且模板变量值的个数必须与手机号码个数相同、内容一一对应，表示向指定手机号码中发对应的短信，且短信模板中的变量参数替换为对应的值。
upExtendCode	String	否	`"upExtendCode":"32"`	上行短信扩展码，上行短信，指发送给通信服务提供商的短信，用于定制某种服务、完成查询，或是办理某种业务等，需要收费的，按运营商普通短信资费进行扣费。**说明** 无特殊需要此字段的用户请忽略此字段。默认取值范围：1~99
bid	String	否	`"bid":"xxxxxxx"`	客户端业务ID，长度不超过64字符

返回数据

名称	类型	示例值	描述

code	String	200	请求状态码。返回200代表请求成功。其他错误码详见。
message	String	OK	状态码的描述。
data.msgid	String		每一批短信编号。该值不为空，说明平台已经接收提交的短信任务
data.desc	String		状态描述
data.failPhones	string		提交失败的手机号
requestId	String		问题排查时，提供该字段值给短信平台

示例

请求示例

JSON

```
{"phonesJson":"[\"xxxxx\",\"xxxx\"]","templateCode":"574277763210051584","templateParamJson":"[[\"明\"],[\"张三\"]]"}
```

正常返回示例

`JSON` 格式

JSON

```
{
    "code": 200,
    "data": {
        "msgId": "574729305331499008"
    },
  "requestId": "176e2bfc-a9ab-4096-ae5e-9681b9856992"
}
```

短信回执状态推送

概述

短信状态 API 提供了获取短信发送状态信息。

> 注意：
>
> - 短信发送状态提供回调的获取方式（经分平台主动推送短信状态）
> - 回调接口必须在1秒内响应，否则服务端将自动断开，连续推送三次回调接口响应均超过1秒，服务端将不在继续推送

短信状态报文格式

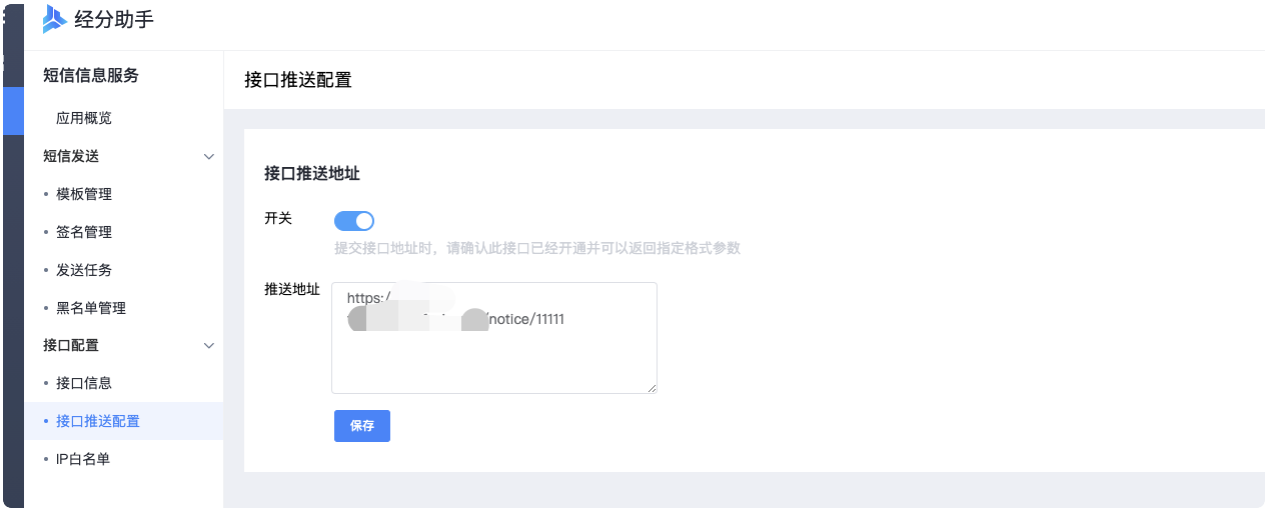
JSON

```
{
  "apiKey": "8767fd835ca3a4eb6f69129d71971760",
  "data": [{
    "msgId": "577951956854276096",--消息id
    "tel": "18551067101",--手机号
    "arriveResult": "UNDELIV",--到达结果。详情参见《短信网关回执短信状态》
    "arriveTime": "2104091735",--到达时间yyMMddHHmm
    "bid": "xxxx" -- 客户端业务ID，发送短信接口提供，此处透传
  }]
}
```

注册短信发送状态回调

接口配置->接口推送配置

> 将回调地址注册到经分助手



短信状态参考

短信网关回执短信状态

- DELIVRD Message is delivered to destination
- EXPIRED Message validity period has expired
- DELETED Message has been deleted.
- UNDELIV Message is undeliverable
- ACCEPTD Message is in accepted state(i.e. has been manually read on behalf of the subscriber by customer service)
- UNKNOWN Message is in invalid state
- REJECTD Message is in a rejected state

短信回执状态查询

接口

{接口域名}/sms/arriveRes/qry

描述

回执信息查询只能查询最近6小时内的回执信息, 返回过的数据不会再次返回, 建议查询频次1分钟，接口单次查询最大返回的回执条数为600

POST

header:

- x-api-key: api接入key
- x-sign-method ： 签名方法，支持HmacSHA256、HmacMD5、HmacSHA1、HmacSHA224、HmacSHA384、HmacSHA512
- x-nonce: 防重复提交(数字)
- x-timestamp: 时间戳
- x-sign: 签名

输入参数

参数	必传	类型	说明
msgId	否	string	msgId 从短信发送接口返回报文获取。多个msgid用英文逗号分隔
bid	否	string	客户端业务ID，调用短信发送接口时传入

输出参数

code		int	请求状态码。返回200代表请求成功。其他错误码详见。
message		string	状态码的描述。
data		array<object>	

	msgId	string	
	tel	string	
	arrive	string	除DELIVRD外，其他都是未送达
	arriveTime	string	到达时间；格式 yyMMddHHmm
	bid	string	客户端业务ID，短信发送时传入，此处透传
requestId		string	请求唯一标识

返回样例

```json
{
    "code":200,
    "message":"查询成功",
    "data":[
        {
            "msgId":"640614750879358976",
            "tel":"187****8801",
            "arrive":"DELIVRD",
            "arriveTime":"2308071315",
            "bid":"202303231335001220"
        },
        {
            "msgId":"640614750879358976",
            "tel":"187****8802",
            "arrive":"DELIVRD",
            "arriveTime":"2308071315",
            "bid":"202303231335001220"
        },
        {
            "msgId":"640614750879358976",
            "tel":"187****8805",
            "arrive":"DELIVRD",
            "arriveTime":"2308071315",
            "bid":"202303231335001220"
        },
        {
            "msgId":"640614750879358976",
            "tel":"187****8806",
            "arrive":"DELIVRD",
            "arriveTime":"2308071315",
            "bid":"202303231335001220"
        }
    ],
    "requestId":"2fffb7d9-6569-4501-98f0-cee3656a6c34"
}
```

短信上行报告推送

概述

短信上行 API 提供了获取短信上行信息。

> 注意：
>
> - 短信发送上行消息提供回调的获取方式（经分平台主动推送短信上行消息），不支持使用短信上行消息 API 获取（三方平台主动从经分平台拉取短信上行消息）。
> - 回调接口必须在1秒内响应，否则服务端将自动断开，连续推送三次回调接口响应均超过1秒，服务端将不在继续推送
> - 推送方式：POST

短信上行消息报文格式

JavaScript

```
{
  "apiKey": "8767fd835ca3a4eb6f69129d71971760",
  "data": [{
    "srcId":"接入号",--可能含有尾号和自定义接入号
    "tel": "18551067101",--手机号
    "content": "上行内容",--上行内容
    "arriveTime": "2104091735"--到达时间yymmddhhmm
  }]
}
```

注册短信上行消息回调

短信应用->接口配置->接口推送配置

上行消息接收

开关

提交接口地址时，请确认此接口已经开通并可以返回指定格式参数

推送地址

https:/ [redacted] /upstream

保存

查询短信发送统计

查询短信发送量详情

{接口域名}/sms/send/statistics

描述

可以通过本接口查询短信发送量详情，包括短信发送时间、短信发送成功条数、接收回执条数等。如果指定日期短信发送量较大，可以分页查看。指定每页显示的短信详情数量和查看的页数，即可分页查看发送记录。

> 单用户QPS限制4次/分钟。超过限制，API调用会被限流

POST

header:

- x-api-key: api接入key
- x-sign-method ： 签名方法，支持HmacSHA256、HmacMD5、HmacSHA1、HmacSHA224、HmacSHA384、HmacSHA512
- x-nonce: 防重复提交(数字)
- x-timestamp: 时间戳
- x-sign: 签名

输入参数

参数	必传	类型	说明
startDate	是	string	开始日期，格式：yyyy-MM-dd
endDate	是	string	结束日期，格式：yyyy-MM-dd
page	是	int	当前页码，默认1
pageSize	是	int	每页显示条数。取值1~50，默认10

输出参数

code		int	请求状态码。返回200代表请求成功。其他错误码详见。
message		string	状态码的描述。
data		array<object>	
	totalCount	long	发送成功的短信条数。
	successCount	long	接收到回执成功的短信条数
	failCount	long	接收到回执失败的短信条数
	noReceiptCount	long	未回执的短信数
	sendDate	string	短信发送日期，格式：yyyy-MM-dd
totalSize	long		返回数据的总条数
requestId		string	请求唯一标识

查询账号短信剩余量

签名

【客户端】

1. 签名规则

- 接口授权 apikey、accesskey为企业开户后，可到企业平台进行申请，每个账号对应一个apikey、accesskey
- 数据有效性 加入timestamp（时间戳,13位），以保证在特定的时间内数据提交的数据有效
- 防止重复提交 加入nonce，防止重复提交数据，长度至少10位，需要保证特定时间内的唯一性，以避免重复提交
- 数据防串改 加入sign，结合timestamp、nonce对所有数据进行签名，防止数据被串改。其中apikey、timestamp、nonce、sign这四个字段放入请求头中

2. 请求头部分

- x-api-key: api接入key
- x-sign-method ： 签名方法，支持HmacMD5、HmacSHA1、HmacSHA224、HmacSHA384、HmacSHA512
- x-nonce: 防重复提交(数字)
- x-timestamp: 时间戳(13位)
- x-sign: 签名

3. 签名生成

- 签名字符串　将请求体参数及参数值、请求头部信息（除x-sign）按照数据字典排序后得到一个用来签名的字符串。例如:x-api-key=你的apiKey&x-nonce=H95a7XT1h6&x-sign-method=HmacSHA256&x-timestamp=1641350184346
- 签名　使用加密算法加secretkey对关键数据签名串进行加密处理，得到签名；
- 最终HTTP请求　将签名相关的所有头加入到原始HTTP请求中，得到最终HTTP请求；

账号短信剩余量查询

- {接口域名}/sms/residueValue/qry 查询账号短信剩余可用数量。
- 需要注意限制调用的频率，默认限制15秒一次。

请求参数

无

返回数据

名称	类型	示例值	描述
code	String	200	请求状态码。返回200代表请求成功。其他错误码详见。
message	String	OK	状态码的描述。
data	Array		短信剩余量信息数组对象
itemName	String		短信类别名称：行业短信、推广短信、会员短信
residueValue	int		短信当前剩余可用数量

示例

正常返回示例

`JSON` 格式

JSON

```
{
  "code" : 200,
  "data" : [ {
    "itemName" : "行业短信",
    "residueValue" : 9784
  }, {
    "itemName" : "会员短信",
    "residueValue" : 10007
  }, {
    "itemName" : "推广短信",
    "residueValue" : 6506
  } ],
  "requestId" : "28284840-5224-4167-a57d-3b76d11f5705"
}
```

Python调用短信接口

1、send/sendBacth接口短信发送类

Python

```python
import hashlib
import hmac
import time
import random
import requests


class SMSSendUtils():

    def __init__(self, api_key, secret_key):
        """
        需要到企业账号下获取
        Args:
            api_key: 公钥
            secret_key: 加密私钥
        """
        self.api_key = api_key
        self.secret_key = secret_key

    @property
    def x_timestamp(self):
        """获取13位时间戳"""
        t = time.time()
        timestamp = int(round(t * 1000))
        return timestamp

    @property
    def nonce(self):
        """获取10位随机数"""
        rand = random.randint(int(1), int(1e10 - 1))
        return rand

    def get_sign_content(self, parameters):
        """参数排序，返回格式为：key1=value1&key2=value2&key3=value3"""
        # 第一步：把字典按Key的字母顺序排序
        sorted_params = dict(sorted(parameters.items()))
        query = ''
        # 第二步：把所有参数名和参数值串在一起
        for key, value in sorted_params.items():
            if key and value:
                query += key + '=' + value + '&'
        content = query[:-1]

        return content

```

```python
    def get_sgin_encrypt(self, encrypt_text, encrypt_key):
        """
        生成sign签名
        Args:
            encrypt_text: 加密文本
            encrypt_key: 加密私钥

        Returns: 生成的签名串

        """
        message = encrypt_text.encode('utf-8')
        key = encrypt_key.encode('utf-8')
        hmac_code = hmac.new(key, message, digestmod=hashlib.sha256).hexdigest()
        return hmac_code

    def sms_api_send(self, phones, templateCode,upExtendCode:str = None, templateParam:str = None):
        """
        短信发送
        Args:
            phones: 手机号
            templateCode: 模板号
            templateParam: 模板带参数
            upExtendCode: 拓展码

        Returns:

        """
        authHost = "https://opassapi.infocloud.cc/message/send"
        currentTime = str(self.x_timestamp)
        headers = {
            "x-sign-method": "HmacSHA256",
            "x-api-key": self.api_key,
            "x-nonce": str(self.nonce),
            "x-timestamp": currentTime,

        }
        # 兼容带参数发送
        if templateParam:
            data = {
                "phones": phones,
                "templateCode": templateCode,
                "templateParam": templateParam,
                "upExtendCode" : upExtendCode
            }
        else:
            data = {
```

```
                "phones": phones,
                "templateCode": templateCode,
                "upExtendCode": upExtendCode
            }

        signDic = dict(data, **headers)
        dicStr = self.get_sign_content(signDic)
        sign = self.get_sgin_encrypt(dicStr, accesskey)
        headers["x-sign"] = sign
        response = requests.post(authHost, headers=headers, json=data)
        result = response.text
        print(f'send发送返回结果: '
              f'{result}')

    def send_api_sendbatch(self,phonesJson,templateCode,templateParamJson
,upExtendCode:str = None):
        """
        批量发送不同手机号发送不同内容
        Args:
            phonesJson: 发送号码'[[\"测试\\1\"],[\"测试\\2\"]]'
            templateCode: 短信模板
            templateParamJson: 模板参数
            upExtendCode:

        Returns:

        """
        authHost_sendbantch = "https://opassapi.infocloud.cc/message/send
Batch"
        currentTime = str(self.x_timestamp)
        headers = {
            "x-sign-method": "HmacSHA256",
            "x-api-key": self.api_key,
            "x-nonce": str(self.nonce),
            "x-timestamp": currentTime

        }
        data = {
            "phonesJson": phonesJson,
            "templateCode": templateCode,
            "templateParamJson": templateParamJson,
            "upExtendCode" : upExtendCode
        }

        signDic = dict(data, **headers)
        dicStr = self.get_sign_content(signDic)
        sign = self.get_sgin_encrypt(dicStr, accesskey)
        headers["x-sign"] = sign
```

```
        response = requests.post(authHost_sendbantch, headers=headers, json=data)
        result = response.text
        print(f'sendbacth发送返回结果：'
              f'{result}')


if __name__ == '__main__':
    #以下为调用演示

    # 企业账号处获取apiKey、accesskey
    apiKey = "198c4a80xxxxdbd66986f"
    accesskey = "93cbd4468xxxxx824f5131aae"


    # 拓展码
    upExtendCode = '15'

    # 实例化一个发送对象
    Send = SMSSendUtils(apiKey, accesskey)

    # 模板带变量发送调用示例
    # 发送手机号
    phones = "136xxxx0982,183xxxx9925"
    # 带变量模板
    templateCode_var = "823xxxxx8112"
    # send接口带变量,举例两变量
    templateParam = '[\"测试1\",\"测试2\"]'
    Send.sms_api_send(phones, templateCode_var,upExtendCode,templateParam)

    # 模板不带变量发送调用示例
    # 不带变量模板
    templateCode = '8181xxxxx10080'
    # send接口不带变量
    Send.sms_api_send(phones, templateCode,upExtendCode)

    # 模板带变量批量发送示例
    # sendbacth带变量参数
    templateParamJson = '[[\"测试1\"],[\"测试2\"]]'
    # 模板带变量批量发送模板
    templateCode_var_sendbatch = "823xxxxxx8112"
    # sendbacth手机号
    phones_send_bacth = '[\"136xxxx0982\",\"183xxxx925\"]'
    Send.send_api_sendbatch(phones_send_bacth,templateCode_var_sendbatch,templateParamJson,upExtendCode)
```

2、返回结果

示例：

```Python
send发送返回结果：{
  "code" : 200,
  "data" : {
    "msgId" : "822515453833289728",
    "desc" : "null"
  },
  "requestId" : "47b98424-907a-43c7-8bc0-4dade41e3e5f"
}
```

PHP调用短信接口

```php
<?php
$apiKey = "a963d8*********ae1093";
$accessKey = "b9935d21*********8610e57d7";

$authHost = "https://opassapi.infocloud.cc/message/send";
$ts = time();
$currentTime = $ts * 1000;
$requestHeaders = array();
//签名算法。支持HmacSHA256、HmacMD5、HmacSHA224、HmacSHA384、HmacSHA512
$requestHeaders['x-sign-method'] = 'HmacMD5';
$requestHeaders['x-api-key'] = $apiKey;
//随机数。每次请求随机数建议不相同
$requestHeaders['x-nonce'] = mt_rand();
//时间戳
$requestHeaders['x-timestamp'] = $currentTime;

$jsArr = array();
$jsArr['phones'] = '15*****0';
$jsArr['templateCode'] = '82*****48';
$jsArr['templateParam'] = '["xxxx","xxxx"]';
$jsArr['upExtendCode'] = '123';
//将headers参与与接口报文参数合并、排序，并生成本次请求的签名信息
$signDic = array_merge($jsArr, $requestHeaders);
ksort($signDic);

$dicStr = '';
foreach ($signDic as $key => $value) {
    $dicStr .= $key . '=' . $value . '&';
}
$dicStr = rtrim($dicStr, '&');
$sign = hash_hmac('md5', $dicStr, $accessKey);
$requestHeaders['Content-Type'] = 'application/json';
$header = array();
foreach ($requestHeaders as $key => $value) {
    $header[] = $key . ': ' . $value;
}
$header[] = 'x-sign: ' . $sign;
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $authHost);
curl_setopt($ch, CURLOPT_HTTPHEADER, $header);
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($jsArr));
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, FALSE);
```

```
46
47  $response = curl_exec($ch);
48  curl_close($ch);
49
50  print_r($response);
```

VB.NET调用短信接口

VB.net

```vb
Imports System.Net.Http
Imports Newtonsoft.Json
Imports System.Text
Imports System.Security.Cryptography

Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        sendMsg(TextBox1.Text, TextBox2.Text)
    End Sub

    Public Sub sendMsg(ByVal number As String, ByVal tempcode As String)

        Dim apiKey = "XXXXX"
        Dim accessKey = "XXXX"
        Dim authHost As String = "https://opassapi.infocloud.cc/message/send"
        Dim ts = DateTime.UtcNow - New DateTime(1970, 1, 1, 0, 0, 0, 0)
        Dim currentTime = Convert.ToInt64(ts.TotalMilliseconds).ToString()
        Dim client As HttpClient = New HttpClient()
        Dim requestHeaders = New Dictionary(Of String, String)()
        requestHeaders.Add("x-sign-method", "HmacSHA256")
        requestHeaders.Add("x-api-key", apiKey)
        requestHeaders.Add("x-nonce", GetRandom().ToString())
        requestHeaders.Add("x-timestamp", currentTime)

        For Each item In requestHeaders
        Console.WriteLine("[{0}]: {1}", item.Key, item.Value)
        client.DefaultRequestHeaders.Add(item.Key, item.Value)
    Next

    Dim jsArr = New Dictionary(Of String, String)()
    jsArr.Add("phones", number)
    jsArr.Add("templateCode", "859119XXXXXXXXXX7056")
    'jsArr.Add("templateParam", "[" & tempcode & "]")
    jsArr.Add("templateParam", "[""" & tempcode & """]")
    Dim signDic = jsArr.Union(requestHeaders).ToDictionary(Function(k) k.Key, Function(v) v.Value)
            Dim dicStr = GetSignContent(signDic)
            Dim sign = Hmacsha256Encrypt(dicStr, accessKey)
            client.DefaultRequestHeaders.Add("x-sign", sign)
            Dim str As HttpContent = New StringContent(JsonConvert.SerializeObject(jsArr))
            str.Headers.Remove("Content-Type")
```

```
            str.Headers.Add("Content-Type", "application/json")
            Dim response As HttpResponseMessage = client.PostAsync(authHost, str).Result
            Dim result As String = response.Content.ReadAsStringAsync().Result
        End Sub

        Private Shared Function GetSignContent(ByVal parameters As IDictionary(Of String, String)) As String
            Dim sortedParams As IDictionary(Of String, String) = New SortedDictionary(Of String, String)(parameters)
            Dim dem As IEnumerator(Of KeyValuePair(Of String, String)) = sortedParams.GetEnumerator()
            Dim query As StringBuilder = New StringBuilder("")

            While dem.MoveNext()
                Dim key As String = dem.Current.Key
                Dim value As String = dem.Current.Value

                If Not String.IsNullOrEmpty(key) AndAlso Not String.IsNullOrEmpty(value) Then
                    query.Append(key).Append("=").Append(value).Append("&")
                End If
            End While

            Dim content As String = query.ToString().Substring(0, query.Length - 1)
            Return content
        End Function

        Private Function Hmacsha256Encrypt(ByVal encryptText As String, ByVal encryptKey As String) As String
            Dim encoding = New System.Text.UTF8Encoding()
            Dim builder As StringBuilder = New StringBuilder()

            Using hmacsha256 = New HMACSHA256(encoding.GetBytes(encryptKey))
                Dim hashmessage As Byte() = hmacsha256.ComputeHash(encoding.GetBytes(encryptText))

                For i As Integer = 0 To hashmessage.Length - 1
                builder.Append(hashmessage(i).ToString("x2"))
            Next

            Console.WriteLine(builder)
        End Using
```

```
79              Return builder.ToString()
80          End Function
81
82
83          Private Shared Function GetRandom() As Integer
84              Dim random = New Random()
85              Dim rand = random.[Next](10000, 999999999)
86      Return rand
87      End Function
        End Class
```

C# 调用短信接口

> C# HTTP请求库

- 通用样例

📎api-demo.zip

C#

```
using System.Security.Cryptography;
using System.Text;
using System.Text.Json;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            var apiKey = "27cf33a91ff9f5930a49e12b66e7xxxx";
            var accessKey = "32c0d2a0f9bc449c93199272a9ef07e0502d68cd903beb3ab39a92116xxx";

            String authHost = "https://opassapi.infocloud.cc/message/send";
            var ts = DateTime.UtcNow - new DateTime(1970, 1, 1, 0, 0, 0, 0);
            var currentTime= Convert.ToInt64(ts.TotalMilliseconds).ToString();
            HttpClient client = new HttpClient();

            var requestHeaders = new Dictionary<string,string>();
            //签名算法。支持HmacSHA256、HmacMD5、HmacSHA224、HmacSHA384、HmacSHA512
            requestHeaders.Add("x-sign-method", "HmacSHA256");
            requestHeaders.Add("x-api-key", $"{apiKey}");
            //随机数。每次请求随机数建议不相同
            requestHeaders.Add("x-nonce", $"{GetRandom()}");
            //时间戳
            requestHeaders.Add("x-timestamp", $"{currentTime}");


            // client.DefaultRequestHeaders.Host = "recognition.image.myqcloud.com";
            //设置headers请求参数
            foreach (var item in requestHeaders)
            {
                Console.WriteLine("[{0}]: {1}", item.Key, item.Value);
                client.DefaultRequestHeaders.Add(item.Key,item.Value);
            }

            //短信接口请求参数。参见接口规范
            var jsArr = new Dictionary<string, string>();
            jsArr.Add("phones","18502820xxx");
```

```
            jsArr.Add("templateCode","8837136760103362xxx");
            jsArr.Add("templateParam","[\"张三\",\"生日祝福\"]");

            //将headers参与与接口报文参数合并、排序，并生成本次请求的签名信息
            var signDic = jsArr.Union(requestHeaders).ToDictionary(k=>k.Key,v=>v.Value);
            var dicStr = GetSignContent(signDic);
            var sign = Hmacsha256Encrypt(dicStr,accessKey);
            //在header中设置本次请求签名信息
            client.DefaultRequestHeaders.Add("x-sign",sign);


            HttpContent str = new StringContent(JsonSerializer.Serialize(jsArr));
            str.Headers.Remove("Content-Type");
            str.Headers.Add("Content-Type", "application/json");
            HttpResponseMessage response = client.PostAsync(authHost, str).Result;
            String result = response.Content.ReadAsStringAsync().Result;
            Console.WriteLine(result);
        }

        /// <summary>
        /// Get Sign Content
        /// </summary>
        /// <param name="parameters"></param>
        /// <returns></returns>
        private static string GetSignContent(IDictionary<string, string> parameters)
        {
            // 第一步：把字典按Key的字母顺序排序
            IDictionary<string, string> sortedParams = new SortedDictionary<string, string>(parameters);
            IEnumerator<KeyValuePair<string, string>> dem = sortedParams.GetEnumerator();

            // 第二步：把所有参数名和参数值串在一起
            StringBuilder query = new StringBuilder("");
            while (dem.MoveNext())
            {
                string key = dem.Current.Key;
                string value = dem.Current.Value;
                if (!string.IsNullOrEmpty(key) && !string.IsNullOrEmpty(value))
                {
                    query.Append(key).Append("=").Append(value).Append("&");
                }
```

```
            }
            string content = query.ToString().Substring(0, query.Length - 1);

            return content;
        }

        /// <summary>
        /// HMACSHA1算法加密
        /// </summary>
        private static string Hmacsha256Encrypt(string encryptText, string encryptKey)
        {
            var encoding = new System.Text.UTF8Encoding();
            StringBuilder builder = new StringBuilder();

             using (var hmacsha256 = new HMACSHA256(encoding.GetBytes(encryptKey)))
                {
                    byte[] hashmessage = hmacsha256.ComputeHash(encoding.GetBytes(encryptText));
                    for (int i = 0; i < hashmessage.Length; i++)
                    {
                        builder.Append(hashmessage[i].ToString("x2"));
                    }
                    Console.WriteLine(builder.ToString());
                 }
              return builder.ToString();
        }
        /// <summary>
        /// 获取随机数
        /// </summary>
        private static int GetRandom()
        {
            var random = new Random();
            var rand = random.Next(10000, 999999999);
            return rand;
        }
    }
}
```

- 简洁样例（数据字典拼接）

> 该样例，可在线运行，生成签名

C#

```
using System;
using System.Security.Cryptography;
using System.Text;

namespace HelloWorldApplication
{
   class HelloWorld
   {
      static void Main(string[] args)
      {

                var encoding = new System.Text.UTF8Encoding();
                byte[] keyByte = encoding.GetBytes("你的密钥");
                byte[] messageBytes = encoding.GetBytes("phones=15900582939&templateCode=580467552590397440&x-api-key=22a71bfe4f9cab1dbce7e16bd71138c2&x-nonce=8052433127&x-sign-method=HmacSHA256&x-timestamp=1618810991174");
                                StringBuilder builder = new StringBuilder();


    using (var hmacsha256 = new HMACSHA256(keyByte))
                {
                    byte[] hashmessage = hmacsha256.ComputeHash(messageBytes);
                    for (int i = 0; i < hashmessage.Length; i++)
                    {
                        builder.Append(hashmessage[i].ToString("x2"));
                    }
                    Console.WriteLine(builder);
     Console.ReadKey();
                }
      }
   }
}
```

postman调用短信接口

> postMan接口测试工具下载
> 📎temp.postman_collection.json

配置自动签名

- 设置pre-request-script

JavaScript

```javascript
let apiKey = "421cc0ce7xxxxxxxxdeb7e09bb35";
let accesskey = "3c4bda9d5698a6b17103461a82eexxxxxxxxx0ae4257f29d15b04c3c";
let param = request.data;   //post 参数
let queryParam = pm.request.url.query.members;  //get中的参数

try{
    let json = JSON.parse(param);
    param = json;
}catch(err){
    console.error("非标准json字符串");
}

//将post和get合并，并且移除sign参数
for (let i in queryParam) {
    if (queryParam[i].key == "sign") {
        continue;
    }
    param[queryParam[i].key] = queryParam[i].value;
}

//获取header相关参数
param["x-sign-method"] = "HmacSHA256";
param["x-api-key"] = apiKey;
param["x-nonce"] = randomNum(10);
param["x-timestamp"] = (new Date()).getTime().toString();
pm.environment.set("reqtime", param["x-timestamp"]);
pm.environment.set("signmethod",param["x-sign-method"]);
pm.environment.set("nonce",param["x-nonce"]);
pm.environment.set("apiKey",param["x-api-key"]);
console.log(param);
//排序
param = objSort(param);

//拼接
var kv=[];
for(var key in param){
    var obj =param[key];

    if(obj instanceof Object){
        kv.push(key+'='+JSON.stringify(obj));
    }else{
        kv.push(key+'='+obj);
    }
}
var str = kv.join('&');
```

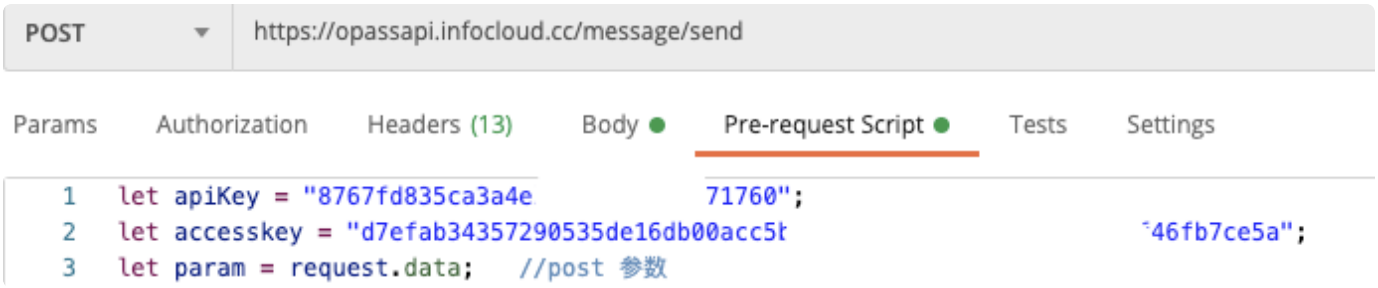
```
console.log(str);

//json，然后生成签名
let sha1Str = CryptoJS.HmacSHA256(str,accesskey);

postman.setGlobalVariable("sign", sha1Str);

//随机数生成
function randomNum(n) {
    var num = "";
    for(var i=0;i<n;i++){
        num+=Math.floor(Math.random()*10);
    }
    return num;
}


//排序方法
function objSort(obj)
{
    let keys = Object.keys(obj).sort();
    let arr = {};
    for (let i in keys) {
        arr[keys[i]] = obj[keys[i]];
    }
    return arr;
}
```

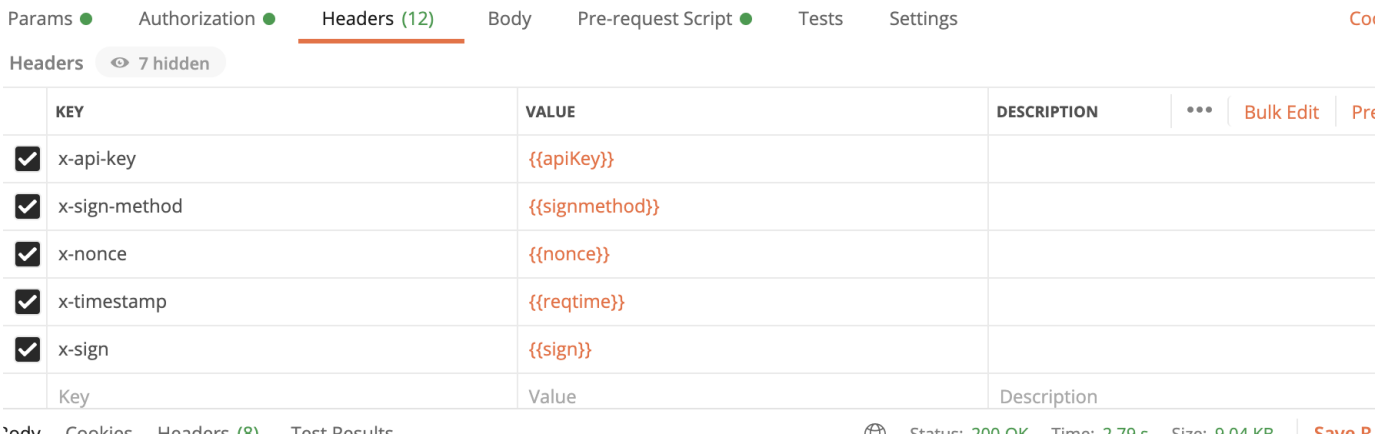


> apiKey 、accesskey来源于企业运营平台接口授权

- 设置请求Headers

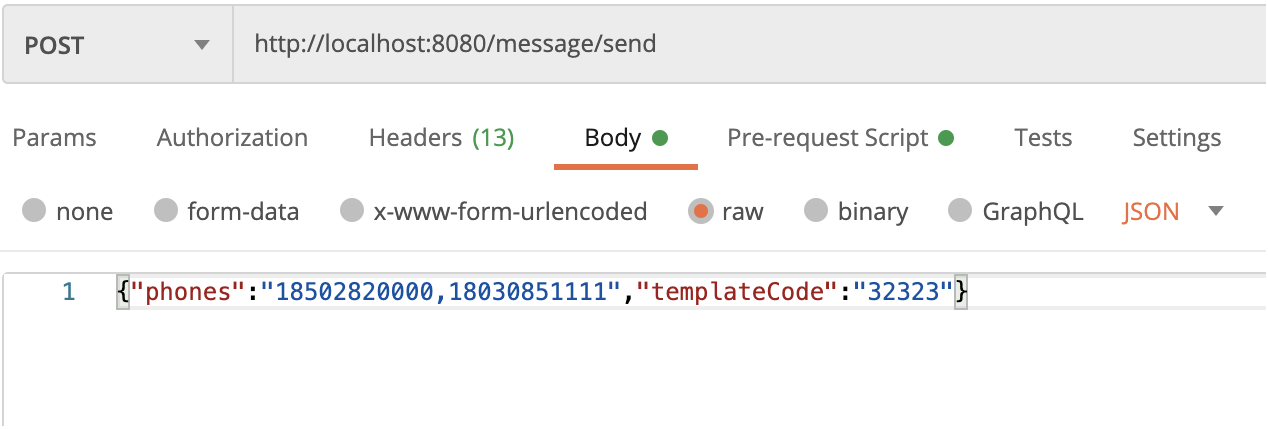
JSON

```
x-api-key:{{apiKey}}
x-sign-method:{{signmethod}}
x-nonce:{{nonce}}
x-timestamp:{{reqtime}}
x-sign:{{sign}}
```



请求接口

> 接口地址：https://opassapi.infocloud.cc



java 调用短信接口

通用样例 📎api-sign-demo.zip，该demo包含以下样例代码

依赖引入

XML

```xml
<dependency>
  <groupId>org.apache.httpcomponents</groupId>
  <artifactId>fluent-hc</artifactId>
  <version>4.5.13</version>
</dependency>
<dependency>
  <groupId>com.alibaba</groupId>
  <artifactId>fastjson</artifactId>
  <version>1.2.73</version>
</dependency>
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <version>1.18.18</version>
  <scope>compile</scope>
</dependency>
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-simple</artifactId>
  <version>1.7.25</version>
</dependency>
<dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>commons-lang3</artifactId>
  <version>3.12.0</version>
</dependency>
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.11</version>
  <scope>test</scope>
</dependency>
```

短信发送工具类

Java

```java
import com.alibaba.fastjson.JSON;
import org.apache.commons.codec.digest.HmacAlgorithms;
import org.apache.commons.codec.digest.HmacUtils;
import org.apache.commons.lang3.StringUtils;
import org.apache.http.HttpResponse;
import org.apache.http.HttpVersion;
import org.apache.http.client.fluent.Request;
import org.apache.http.entity.ContentType;

import java.io.IOException;
import java.util.*;
import java.util.concurrent.atomic.AtomicBoolean;
import java.util.concurrent.locks.ReentrantLock;
import java.util.stream.Collectors;

/***
 * 短信发送工具类
 */
public class SMSSendUtils {
    /**
     * 短信平台接口配置中申请，替换为企业自己的API_KEY
     */
    private static  String API_KEY;
    /**
     * 短信平台接口配置中申请，替换为企业自己的SECRET_KEY
     */
    private static String SECRET_KEY;

    private static final String DOMAIN = "https://opassapi.infocloud.cc/";
    private static final String MESSAGE_SEND="message/send";
    private static final String MESSAGE_SEND_BATCH = "message/sendBatch";
    public static final String MESSAGE_ARRIVE_QRY = "sms/arrive/qry";

    public static String SMS_API_KEY = "sms.api.key";
    public static String SMS_SECRET_KEY = "sms.secret.key";

    private static ReentrantLock lock = new ReentrantLock(true);
    private static AtomicBoolean init = new AtomicBoolean(false);

    static void init(){
        String apiKey = System.getProperty("sms.api.key");
        String secretKey = System.getProperty("sms.secret.key");
        System.setProperty(SMSSendUtils.SMS_API_KEY,"从短信平台接口配置中获取");
```

```
        System.setProperty(SMSSendUtils.SMS_SECRET_KEY,"从短信平台接口配置中获取");
        if(StringUtils.isAnyEmpty(apiKey,secretKey)){
            throw new NullPointerException("短信发送apiKey、secretKey不能为空。\n解决方式：\n1.代码设置。" +
                    "System.setProperty(SMSSendUtils.SMS_API_KEY,\"从短信平台接口配置中获取\")；" +
                    "System.setProperty(SMSSendUtils.SMS_SECRET_KEY,\"从短信平台接口配置中获取\")；\n" +
                    "2.启动配置。java -Dsms.api.key=xx -Dsms.secret.key=xxx");
        }
        API_KEY = apiKey;
        SECRET_KEY = secretKey;
    }

    private SMSSendUtils(){}

    /**
     * 短信发送。无参短信模板内容发送，即所有手机号接收到的短信内容一样
     * @param templateCode
     * @param phones 手机号。手机号最大个数限制1000
     * @return
     */
    public static HttpResponse send(String templateCode, String ...phones) throws IOException {

        return send(templateCode,null,phones);
    }

    /**
     * 短信发送。不同手机号对应相同的短信模板变量值，即所有手机号接收到的短信内容一样
     * @param templateCode
     * @param params 模板参数变量值
     * @param phones 手机号。手机号最大个数限制1000
     * @return
     */
    public static HttpResponse send(String  templateCode, LinkedList<String> params, String ... phones) throws IOException {
        //手机号先去重，然后再转换
        return sendBatch(templateCode,params!=null&&!params.isEmpty()?JSON.toJSONString(params):null,Arrays.stream(phones).collect(Collectors.toSet()).stream().collect(Collectors.joining(",")), false);
    }

    /**
     * 短信发送。不同手机号对应不同的短信模板变量值，即所有手机号接收到的短信内容不一样
```

```
     * @param templateCode 模板ID
     * @param phonesAndParams 手机号-手机号对应的模板参数。手机号最大个数限制500
     * @return
     */
    public static HttpResponse send(String templateCode, Map<String,LinkedList<String>> phonesAndParams) throws IOException {
        return sendBatch(templateCode,JSON.toJSONString(phonesAndParams.values()),JSON.toJSONString(phonesAndParams.keySet()),true);
    }

    /**
     * 回执信息查询只能查询最近6小时内的回执信息，返回过的数据不会再次返回，建议查询频次1分钟，接口单次查询最大返回的回执条数为600
     * @param msgids 短信发送msgid
     * @return 除DELIVRD外，其他都是未送达
     */
    public static HttpResponse getArriveByIds(String ...msgids) throws IOException {
        if(msgids==null){
            throw new NullPointerException("msgid不能为空");
        }

        Map<String, Object> headers = getHeaders();
        Map<String, Object> bodyParams = new HashMap<>(4);
        bodyParams.put("msgId",Arrays.stream(msgids).collect(Collectors.toSet()).stream().collect(Collectors.joining(",")));

        //排序
        SortedMap<String, Object> sortedMap = new TreeMap<>(bodyParams);
        headers.forEach((k, v) -> sortedMap.put(k, v));
        //生成签名
        headers.put("x-sign", getSignature(SECRET_KEY, sortedMap, HmacAlgorithms.HMAC_SHA_224));
        Request request = Request.Post(DOMAIN+MESSAGE_ARRIVE_QRY)
                .version(HttpVersion.HTTP_1_1)
                //设置连接超时
                .connectTimeout(15000)
                //设置文本读取超时
                .socketTimeout(15000);
        headers.forEach((k,v)->request.addHeader(k,String.valueOf(v)));

        return request.bodyString(JSON.toJSONString(bodyParams), ContentType.APPLICATION_JSON)
                .execute()
                .returnResponse();
    }

    /**
```

```
     * 短信发送
     * @param templateCode 模板ID
     * @param paramsJson 模板参数
     * @param phonesJson 手机号
     * @param batch  true：使用MESSAGE_SEND_BATCH接口发送；false:使用MESSAGE_SEND接口发送
     * @return HttpResponse
     * @throws IOException
     */
    private static HttpResponse sendBatch(String  templateCode,String paramsJson,String phonesJson,boolean batch) throws IOException {
        if(StringUtils.isAnyEmpty(templateCode)){
            throw new NullPointerException("模板id或手机号不能为空");
        }

        Map<String, Object> headers = getHeaders();
        Map<String, Object> bodyParams = new HashMap<>(4);

        if(batch){
            if(StringUtils.isAnyEmpty(paramsJson)){
                throw new NullPointerException("短信发送所有手机号接收到的短信内容不一样时，模板参数值不能为空");
            }
            bodyParams.put("phonesJson", phonesJson);
            //模板变量内容
            bodyParams.put("templateParamJson", paramsJson);
        }else{
            bodyParams.put("phones", phonesJson);
            //模板变量内容
            if(StringUtils.isNotEmpty(paramsJson)) {
                bodyParams.put("templateParam", paramsJson);
            }
        }
        bodyParams.put("templateCode", templateCode);

        //排序
        SortedMap<String, Object> sortedMap = new TreeMap<>(bodyParams);
        headers.forEach((k, v) -> sortedMap.put(k, v));
        //生成签名
        headers.put("x-sign", getSignature(SECRET_KEY, sortedMap, HmacAlgorithms.HMAC_SHA_224));

        Request request = Request.Post(DOMAIN+(batch?MESSAGE_SEND_BATCH:MESSAGE_SEND))
                .version(HttpVersion.HTTP_1_1)
                //设置连接超时
                .connectTimeout(15000)
                //设置文本读取超时
```

```
                .socketTimeout(15000);
        headers.forEach((k,v)->request.addHeader(k,String.valueOf(v)));


        return request.bodyString(JSON.toJSONString(bodyParams), ContentType.APPLICATION_JSON)
                .execute()
```

> 备注：若客户服务所在网络环境不稳定，建议将如下链接超时时间 `connectTimeout(15_000)` 设置为15秒（建议根据实际情况设置）；若是读取内容超时，也可将 `socketTimeout` 设置为合适的时间。

发送短信

```java
import com.github.houbb.junitperf.core.annotation.JunitPerfConfig;
import com.github.houbb.junitperf.core.annotation.JunitPerfRequire;
import com.github.houbb.junitperf.core.report.impl.ConsoleReporter;
import com.github.houbb.junitperf.core.report.impl.HtmlReporter;
import lombok.extern.slf4j.Slf4j;
import org.apache.http.HttpResponse;
import org.apache.http.util.EntityUtils;
import org.junit.Test;

import java.io.IOException;
import java.util.HashMap;
import java.util.LinkedList;
import java.util.Map;

@Slf4j
public class SMSSendTest {
    static{
        /**
         * 登录企业平台-富媒体消息->接口配置->接口信息
         */
        System.setProperty(SMSSendUtils.SMS_API_KEY,"xxx");
        /**
         * 登录企业平台-富媒体消息->接口配置->接口信息
         */
        System.setProperty(SMSSendUtils.SMS_SECRET_KEY,"xxxxx");

    }

    /**
     * 发送不同手机号对应相同的短信内容且不带短信模板变量
     * @throws IOException
     */
    @Test
    public void sendOneNoVar() throws IOException {
        HttpResponse httpResponse = SMSSendUtils.send("712387834644013056
1","xxx","xxx");
        JSONObject result = JSONObject.parseObject(EntityUtils.toString(h
ttpResponse.getEntity()));

        if(result.getInteger("code")==200) {
            Thread.sleep(10_000);
            httpResponse = SMSSendUtils.getArriveByIds(result.getJSONObje
ct("data").getString("msgId"));
            log.info("短信发送状态。 {}",EntityUtils.toString(httpResponse.
getEntity()));
```

```
        }else{
            log.info("业务状态非200. {}",EntityUtils.toString(httpResponse.getEntity()));
        }
    }

    /**
     * 短信发送到达状态查询
     */
    @Test
    public void getArriveByIds() throws IOException {
        HttpResponse httpResponse = SMSSendUtils.getArriveByIds("714908845645774848","714908773243699200");
        log.info("短信发送状态。 {}",EntityUtils.toString(httpResponse.getEntity()));
        // 如果有乱码，可尝试处理
        //String s = EntityUtils.toString(httpResponse.getEntity(), Charset.defaultCharset());
        //String s1 = EntityUtils.toString(httpResponse.getEntity(), "utf-8");

    }

    /**
     * 发送不同手机号对应相同的短信内容且带短信模板变量
     * @throws IOException
     */
    @Test
    public void sendOneVar() throws IOException {
        LinkedList<String> vars = new LinkedList<>();
        vars.add("张三");
        vars.add("02");
        vars.add("10000");
        vars.add("323");
        vars.add("200");
        vars.add("12000");
        HttpResponse httpResponse = SMSSendUtils.send("709402052719181824",vars,"xx","xxx");
        log.info("响应报文:{}", EntityUtils.toString(httpResponse.getEntity()));
    }

    /**
     * 发送不同手机号对应不同的短信内容
     * @throws IOException
     */
    @Test
    public void sendBatchVar() throws IOException {
```

```
        Map<String,LinkedList<String>> phonesAndParams = new HashMap<>(4);
        LinkedList<String> vars = new LinkedList<>();
        vars.add("张三");
        vars.add("02");
        vars.add("10000");
        vars.add("323");
        vars.add("200");
        vars.add("12000");
        phonesAndParams.put("xxxx",vars);

        LinkedList<String> vars2 = new LinkedList<>();
        vars2.add("李四");
        vars2.add("02");
        vars2.add("10000");
        vars2.add("323");
        vars2.add("200");
        vars2.add("12000");
        phonesAndParams.put("xxxx",vars2);
        HttpResponse httpResponse = SMSSendUtils.send("709402052719181824",phonesAndParams);
        log.info("响应报文:{}", EntityUtils.toString(httpResponse.getEntity()));
    }

    /**
     * 配置：62个线程运行。准备时间：1000ms。运行时间：30_000ms。
     * 要求：最快不可低于 210ms，最慢不得低于 250ms，平均不得低于 225ms，每秒运行次数不得低于 4 次。
     * 20% 的数据不低于 220ms，50% 的数据不得低于 230ms;
     *
     */
//    @JunitPerfConfig(threads = 62, warmUp = 1000, duration = 30_000,reporter = {HtmlReporter.class, ConsoleReporter.class})
//    @JunitPerfRequire(min = 210, max = 250, average = 225, timesPerSecond = 4, percentiles = {"20:220", "50:230"})
//    public void send() throws IOException {
//        sendOneNoVar();
//    }

}
```

样例代码为无变量模板短信发送。有变量模板短信发送请参考接口规范

常见问题

java.net.SocketException: Connection reset 异常原因分析和解决方法

1. 方法一，跳过SSL

```java
                SSLContext sslcontext = SSLContexts.custom()
                        .loadTrustMaterial((x509Certificates, s) -> true)
                        .build();
                SSLConnectionSocketFactory sslConnectionSocketFactory = new SSLConnectionSocketFactory(sslcontext);

                connectionManager = new PoolingHttpClientConnectionManager(RegistryBuilder.<ConnectionSocketFactory>create()
                        .register("http", PlainConnectionSocketFactory.getSocketFactory())
                        .register("https",sslConnectionSocketFactory).build(), (HttpConnectionFactory)null,(SchemePortResolver)null, (DnsResolver)null,30, TimeUnit.SECONDS);
```

2. 方法二，参见：https://www.cnblogs.com/zhouj850/p/13957713.html

java.net.SocketTimeoutException: Read timed out

参见：https://my.oschina.net/boonya/blog/3046914

返回：API授权失败或授权参数不完整

1. 登录经分平台, 检查代码里 apiKey 和 accesskey 是否和平台一致；

2. 检查http请求,请求头x-sign-method 值 和 生成签名x-sign 使用的算法是否一致 (签名算法没特殊需求,直接使用样例即可);

3. 是中文乱码, message/send 接口为例,templateParam 值尝试使用全英文或数字 ,如果能成功 ,但换成中文不行,就是乱码 .

解决方案：首先检查http请求工具类请求头,请求体相关的字符集编码设置(UTF-8),建议直接使用样例代码,

本地可以发中文,部署到服务器不行,需要检查服务器配置,用到容器的话,容器要同步设置

Java sdk(勿用,特殊环境下会导致发送失败)

> 部分用户以war、jar部署到tomcat时，存在接口认证失败或参数错误的问题

使用说明

~~引入依赖包~~

~~1：maven方式：~~

```xml
<dependency>
    <groupId>io.github.silencelwy</groupId>
    <artifactId>jf-sms-send-sdk</artifactId>
    <version>1.0.3.3-RELEASE</version>
</dependency>
```

~~2：导入jar方式：~~

~~https://repo1.maven.org/maven2/io/github/silencelwy/jf-sms-send-sdk/1.0.3.3-RELEASE/jf-sms-send-sdk-1.0.3.3-RELEASE.jar~~

~~外部依赖jar包~~

~~https://repo1.maven.org/maven2/com/alibaba/fastjson/1.2.73/fastjson-1.2.73.jar~~

~~https://repo1.maven.org/maven2/org/apache/httpcomponents/fluent-hc/4.5.13/fluent-hc-4.5.13.jar~~

~~https://repo1.maven.org/maven2/net/jodah/failsafe/2.4.4/failsafe-2.4.4.jar~~

~~1：短信发送示例~~

~~短信发送一般分为~~

- ~~1.2 固定内容短信模板~~
- ~~1.3 带变量的模板的短信发送~~
- ~~1.4 带变量且每个号码发送内容不一样~~

~~1.1 初始化短信发送api~~

```java
//        String apiKey = "f96d8488c789fc7341e9875d4c631b7";
static String apiKey = "你自己的apiKey";
//        String accessKey = "2cfd72a0d9ad4c257d802a34364781b88ea09471120a1d23686584b958d227b";
static String accessKey = "你自己的AccessKey";
//方式1：默认域名获取示例(推荐)
private SendSmsApi sendSmsApi = SendSmsApi.getInstance(apiKey, accessKey);
//方式2：如果是内网环境，无法访问外网，可以让运维开通公网访问ip和端口，以下方式实现初始化
//private SendSmsApi sendSmsApi= SendSmsApi.getInstance(DomainEnum.IP_PORT.getUrl(), apiKey, accessKey);
//方式3：如果是内网环境，无法访问外网，如果使用ip域名转发，访问url可以自定义，示例如下
//private SendSmsApi sendSmsApi= SendSmsApi.getInstance("http://199.12.25.32:25172/", apiKey, accessKey);
```

~~1.2：固定内容短信模板~~

- ~~即模板内容固定，同时每个号码收到的内容也是一致的。~~
- ~~即 固定模板内容，模板不包含任何变量~~

Java

```java
    /**
     * 没有变量的模板发送，即模板内容固定
     * 如模板号：7671218232469217792
     * 模板内容：你有一个事务需要处理。
     */
    @Test
    public void testNoVar() {
        // 注意如果传入的模板有变量，使用该方法会报错
        //模板号
//        String templateCode = "833337332607283200";
        String templateCode = "你自己的模板id";
        //手机号码，最多不超过1000位
        Set<String> phoneSet = new HashSet<>();
//        phoneSet.add("1811654xxxx");
//        phoneSet.add("1821654xxxx");
        phoneSet.add("手机号1");
        phoneSet.add("手机号2");

        // 以下可以修改默认数据加密方式，一般不需要修改
        // AccessKeyUtils.defaultHmacAlgorithms = HmacAlgorithms.HMAC_SHA_256;
        SmsResponse<MessageSendResVo> send = sendSmsApi.send(templateCode, phoneSet);
        //如果有扩展号或者业务id需求：多传入两个字段 扩展号和自身业务id 两个字段均非必传 如果无须传入，传null即可
        // 扩展号为数字；业务id最长长度为64位
//        SmsResponse<MessageSendResVo> send = sendSmsApi.send(templateCode, phoneSet,"xx","xxxxyyy");
        System.out.println(JSON.toJSONString(send));
        //如果发送有问题，提供该值给平台咨询
        String requestId = send.getRequestId();
        if (200 == send.getCode()){
            System.out.println("提交短信成功");
            MessageSendResVo data = send.getData();
            //每一批短信编号。该值不为空，说明平台已经接收提交的短信任务
            String msgId = data.getMsgId();
            //状态描述
            String desc = data.getDesc();
            //提交失败的手机号，手机号不正确，格式不规范，黑名单等
            String failPhones = data.getFailPhones();
        }
    }
```

~~1.3：带变量的模板的短信发送~~

- ~~模板内容带有变量，通过传入的参数替换其中的变量构成完整短信。~~
- ~~如果是动态签名，需多传入一个签名变量参数即可。~~
- ~~如果自由发送短信，可以创建一个大变量，即所有模板内容只有一个变量。传入这个变量的参数即为全部短信内容~~
- ~~所有号码收到的短信内容一致的。~~

Java

```java
    /**
     * 普通带变量的模板发送，模板内容有变量
     * 如模板号：7671218232469217791
     * 模板内容：你有一个事务编号为{事务编号,30}需要处理。
     * 如果存在动态签名，增加一个变量传入即可，
     * 即增加一个变量传入签名
     */
    @Test
    public void testHasVar() {
        //模板号
//        String templateCode = "833337635977097216";
        String templateCode = "你自己的模板id";
        //变量参数集合，按照顺序传入
        LinkedList<String> paramList = new LinkedList<>();
        //模板内容：你有一个事务编号为{事务编号,30}需要处理。
        //如果存在动态签名，即多传入一个变量，如下。一般不需要传入
        //paramList.add("xxx公积金");
        //这里输入变量内容
        paramList.add("1xhsf1282abhx5");

        //手机号码，最多不超过1000位
        Set<String> phoneSet = new HashSet<>();
//        phoneSet.add("1811654xxxx");
//        phoneSet.add("1821654xxxx");
        phoneSet.add("手机号1");
        phoneSet.add("手机号2");

        // 以下可以修改默认数据加密方式，一般不需要修改
        // AccessKeyUtils.defaultHmacAlgorithms = HmacAlgorithms.HMAC_SHA_256;
        SmsResponse<MessageSendResVo> send = sendSmsApi.sendHasVar(templateCode, paramList, phoneSet);
        //如果有扩展号或者业务id需求：多传入两个字段 扩展号和自身业务id 两个字段均非必传 如果无须传入，传null即可
        // 扩展号为数字；业务id最长长度为64位
        // SmsResponse<MessageSendResVo> send = sendSmsApi.sendHasVar(templateCode, paramList, phoneSet, "xx","xxxxyyyy");
        System.out.println(JSON.toJSONString(send));
        //如果发送有问题，提供该值给平台咨询
        String requestId = send.getRequestId();
        if (200 == send.getCode()){
            System.out.println("提交短信成功");
            MessageSendResVo data = send.getData();
            //每一批短信编号。该值不为空，说明平台已经接收提交的短信任务
            String msgId = data.getMsgId();
```

```
            //状态描述
            String desc = data.getDesc();
            //提交失败的手机号，手机号不正确，格式不规范，黑名单等
            String failPhones = data.getFailPhones();
        }
    }
```

~~1.4：带变量且每个号码发送内容不一样~~

- ~~同上一个内容相似，区别在于每个手机号对应有一个变量内容。即每一个手机号码收到的内容不一致。~~

Java

```java
    /**
     * 普通带变量的模板发送，模板内容有变量，每个手机号收到的内容不一样
     * 如模板号：7671218232469217791
     * 模板内容：你本次收入${收入,10}元。
     * <p>
     * // 1811654xxxx(手机号1)收到 【企业签名】你本次收入300.09元。
     * // 1876789xxxx(手机号2)收到 【企业签名】你本次收入222元。
     */
    @Test
    public void testBatch() {
        //模板号
//        String templateCode = "833337635977097216";
        String templateCode = "你自己的模板id";
        //多个手机号，收到的内容不一致

        Map<String, LinkedList<String>> phonesAndParams = new HashMap<>();
        // 1811654xxxx收到 【企业签名】你本次收入300.09元。
        String phone1 = "1811654xxxx";
        LinkedList<String> paramList1 = new LinkedList<>();
        //如果存在动态签名，即多传入一个变量，如下。固定签名不需要传入
        //paramList.add("xxx公积金");
        paramList1.add("300.09");
        phonesAndParams.put(phone1, paramList1);
        // 18767892205收到 【企业签名】你本次收入222元。
        String phone2 = "1876789xxxx";
        LinkedList<String> paramList2 = new LinkedList<>();
        //如果存在动态签名，即多传入一个变量，如下。一般不需要传入
        //paramList.add("xxx公积金");
        paramList2.add("222");
        phonesAndParams.put(phone2, paramList2);

        // 以下可以修改默认数据加密方式，一般不需要修改
        // AccessKeyUtils.defaultHmacAlgorithms = HmacAlgorithms.HMAC_SHA_256;
        SmsResponse<MessageSendResVo> send = sendSmsApi.sendPhoneToContent(templateCode, phonesAndParams);
        //如果有扩展号或者业务id需求：多传入两个字段 扩展号和自身业务id 两个字段均非必传 如果无须传入，传null即可
        // 扩展号为数字；业务id最长长度为64位
        //SmsResponse<MessageSendResVo> send = sendSmsApi.sendPhoneToContent(templateCode, phonesAndParams,"xx","xxxxyyyy");
        System.out.println(JSON.toJSONString(send));
        //如果发送有问题，提供该值给平台咨询
        String requestId = send.getRequestId();
        if (200 == send.getCode()){
```

```
            System.out.println("提交短信成功");
            MessageSendResVo data = send.getData();
            //每一批短信编号。该值不为空，说明平台已经接收提交的短信任务
            String msgId = data.getMsgId();
            //状态描述
            String desc = data.getDesc();
            //提交失败的手机号，手机号不正确，格式不规范，黑名单等
            String failPhones = data.getFailPhones();
        }
    }
```

~~1.5：短信发送响应内容：~~

~~响应参数说明~~		~~类型~~	~~备注~~	~~描述~~
~~code~~		~~int~~	~~响应码~~	~~200表示成功。如果不是200，参考第3节最后返回码说明~~
~~message~~		~~string~~		~~状态码的描述~~
~~requestId~~		~~string~~		~~问题排查时，提供该字段给短信平台~~
~~data~~				
	~~msgId~~	~~string~~		~~每一批短信编号。该值不为空，说明平台已经接收提交的短信任务。由此数据可查询是否到达，参考回执查询接口~~
	~~failPhones~~	~~string~~		~~提交失败的手机号，多个用逗号分割~~
	~~desc~~	~~string~~		~~描述信息~~

~~2：回执报告查询~~

~~通过发送接口返回的msgId，查询短信发送情况，（通常会在2-3分钟查询到网关回执结果）~~

~~2.1：初始化回执查询api~~

```java
    //        String apiKey = "f96d8488c89b23fc7a1e3475d4c631b7";
    private String apiKey = "你自己的apiKey";
    //        String accessKey = "2cfd72a0d9ad4c234d802a87fa0ca81b88ea09471120a1d23686584b958d227b";
    private String accessKey = "你自己的accessKey";
    //方式1：默认域名获取示例(推荐)
    private ArriveInfoApi arriveInfoApi = ArriveInfoApi.getInstance(apiKey, accessKey);
    //方式2：如果是内网环境，无法访问外网，可以让运维开通公网访问ip和端口，以下方式实现初始化
    //private ArriveInfoApi arriveInfoApi = ArriveInfoApi.getInstance(DomainEnum.IP_PORT.getUrl(), apiKey, accessKey);
    //方式3：如果是内网环境，无法访问外网，如果使用ip域名转发，访问url可以自定义，示例如下
    //private ArriveInfoApi arriveInfoApi = ArriveInfoApi.getInstance("http://199.12.xx.xx:25172/",apiKey, accessKey);
```

~~2.2：回执查询示例~~

```java
    /**
     * 任务号：836281956443505664，836281566675434496
     * 只支持查询24小时以内记录
     * 查询到达结果，多个查询结果，返回以下示例结构
     */
    @Test
    public void testGetArriveInfo() {
        //任务号，最多不超过600，提交发送成功后返回的msgId
        Set<String> msgIdSet = new HashSet<>();
//        msgIdSet.add("836281956443505664");
//        msgIdSet.add("836281566675434496");
        msgIdSet.add("填入发送时返回的msgId");
        // 以下可以修改默认数据加密方式，一般不需要修改
        // AccessKeyUtils.defaultHmacAlgorithms = HmacAlgorithms.HMAC_SHA_256;
        SmsResponse<List<ArriveInfoResVo>> smsResponse = arriveInfoApi.getArriveInfo(msgIdSet);
        System.out.println(JSON.toJSONString(smsResponse));
        if (200 == smsResponse.getCode()){
            System.out.println("查询回执成功");
            List<ArriveInfoResVo> data = smsResponse.getData();
            if (data !=null && data.size()>0){
                data.forEach(arriveInfoResVo -> {
                    //回执结果，DELIVRD表示到达成功，其他结果表示失败
                    String arrive = arriveInfoResVo.getArrive();
                    //手机号
                    String tel = arriveInfoResVo.getTel();
                    //任务号
                    String msgId = arriveInfoResVo.getMsgId();
                    //短信发送时传入的业务id，发送时没有传入该字段，字段返回为空（可无视）
                    //String bid = arriveInfoResVo.getBid();
                });
            }
        }
    }
```

~~2.3：回执结果~~

~~响应参数说明~~		~~类型~~	~~备注~~	~~描述~~

code		int	响应码	200表示成功。如果不是200，参考第3节返回码说明
message		string		状态码的描述
requestId		string		问题排查时，提供该字段给短信平台
data				
	msgId	string		每一批任务编号。短信发送时返回的字段
	tel	string		手机号码
	arrive	到达结果		除DELIVRD外，其他都是未送达
	arriveTime	到达时间		yyMMddHHmm
	bid	业务id		短信发送时传入的bid，没有使用到可无视该字段即可

3：返回码说明

每次调用接口时，可能获得正确或错误的返回码，开发者可以根据返回码信息调试接口，排查错误。全局返回码说明如下：

响应码	描述
-1	系统繁忙，请稍后再试或者其他原因
200	请求成功
2000	IP白名单限制
2001	API授权错误

6000	访问频繁,请稍后再试
40002	余额不足
40004	参数错误,详情见提示
40005	超出模板参数设置限制
40006	扩展号不匹配或者被企业其他账号占用
60001	模板不存在
60002	批量发送下电话数量和参数数量不一样
60003	短信变量个数和模板变量个数不一样
60004	变量值超过模板设置的变量值最大长度
60005	没有可以发送的内容
60006	模板参数变量格式错误
60007	手机号参数格式错误
60008	手机号数量大于1000

网络环境检测

> 监测客户网络与经分平台网络是否连通。

1. 在服务器创建 `netSpeed.sh` 文件，文件内容如下：

Shell

```shell
#!/bin/bash

#Author by xiaolong.li
#Email 1315812131@qq.com

G=`tput setaf 2`
C=`tput setaf 6`
Y=`tput setaf 7`
Q=`tput sgr0`

#颜色
red(){
    echo -e "\033[31m\033[01m$1\033[0m"
}
green(){
    echo -e "\033[32m\033[01m$1\033[0m"
}
yellow(){
    echo -e "\033[33m\033[01m$1\033[0m"
}
blue(){
    echo -e "\033[34m\033[01m$1\033[0m"
}


#获取要解析的域名地址
function analysis(){

#ping 10次,超时时间为3秒
if ping -c 1 -w 1 www.baidu.com >/dev/null;then
    green "本地网络环境正常"

green "${C}开始获取本地ip:${Q}"
curl ip.p3terx.com

green "${C}获取本地出网IP:${Q}"
#read -p "${C}获取出网ip${Q}"
curl  cip.cc

green "${C}准备开始域名解析:${Q}"
read -p "${C}输入要解析的域名地址:${Q}" domain
dig ${domain}

green "${C}开始测试丢包率:${Q}"
read -p "${C}输入要ping发送的数据包个数:${Q}" count
```

```
ping -c ${count} ${domain}

else
    red "本地网络环境异常，脚本自动退出，请检查网络"
    break
fi

}


#接口地址调用
function interface(){

if ping -c 1 -w 1 www.baidu.com >/dev/null;then
green "本地网络环境正常"

read -p "${C}请输入接口地址(回车默认https://opassapi.infocloud.cc/health/info):${Q}" interface_com
if [ -z "${interface_com}" ];then
   interface_com="https://opassapi.infocloud.cc/health/info"
   fi
   if [[ ${interface_com} =~ /$ ]];
      then echo
        else interface_com="${interface_com}/"
        fi

read -p "${C}请输入并发数:${Q}" concurrent

for ((i=1;i<=${concurrent};i=i+1))
do

RESULT=`curl -k -H 'Content-Type:application/json;charset=utf-8' ${interface_com} -X POST`

yellow "返回结果:"
blue $RESULT;

done
else
    red "本地网络环境异常，脚本自动退出，请检查网络"
    break
fi

}

#主菜单
function start_menu(){
    clear
```

```
    red " 接口并发测试脚本"
    green " "
    green " "
    green " "
  yellow " =======基础网络环境检查============================== "
    green " 1. 获取本机IP"

        yellow " =======接口调用============================== "
    green " 2. 调用经分接口 "
        yellow " ============================================== "
    green " 0. 退出脚本"
    echo
    read -p "请输入数字:" menuNumberInput
    case "$menuNumberInput" in
        1 )
           analysis
;;
        2 )
           interface
;;
        0 )
            exit 1
        ;;
        * )
            clear
            red "请输入正确数字 !"
            start_menu
        ;;
    esac
}


start_menu "first"
```

2. 设置 `netSpeed.sh` 权限

```shell
chmod a+x netSpeed.sh
```

3. 运行脚本，根据提示操作

```shell
./netSpeed.sh
```

返回码说明

每次调用接口时，可能获得正确或错误的返回码，开发者可以根据返回码信息调试接口，排查错误。
全局返回码说明如下：

错误码	描述
–1	系统繁忙，请稍后再试或者其他原因
200	请求成功
2000	IP白名单限制
2001	API授权错误
6000	访问频繁,请稍后再试
40002	余额不足
40004	参数错误
40005	超出模板参数设置限制
40006	扩展号不匹配或者被企业其他账号占用
60001	模板不存在
60002	批量发送下电话数量和参数数量不一样
60003	短信变量个数和模板变量个数不一样
60004	变量值超过模板设置的变量值最大长度
60005	没有可以发送的内容
60006	模板参数变量格式错误
60007	手机号参数格式错误
60008	手机号数量大于1000
60009	短信资源被锁定

短信签名FAQ

是否可申请多个短信签名?

企业用户可申请多个签名，签名个数均无限制。

> 签名不要中性，也不要含有联通、电信、移动、工商银行、预警提醒中性等关键字，这样可能导致审核不通过或发送失败（表现为一直收不到短信）

签名规范

类别	规范说明
名称	• 作为短信发送者属性的一种标识，签名必须用于标识公司、产品或业务。建议设置为短信发送方的真实应用名称、网站名称或公司名称。 • 必须具有实际意义，不支持含义模糊的中性签名，如“客服通知”、“客户您好”等。 • 若签名用途为他用或涉及第三方权益，必须获得第三方授权。 • 签名不能含有违法违规或者损害到相关他人权益的内容。
签名长度	建议长度限制为2~20字符
格式	请直接填写，无需添加【】、()、[]等符号，创建模块时会自动为签名增加书名号，列如【XXX】

签名可修改吗？修改签名是否会影响短信发送?

签名可修改，修改签名不会影响已创建的模块短信发送。

是否支持发送时指定签名？怎么申请?

支持接口短信发送时指定签名，可通过经分助手平台创建，格式为：`{动态签名,20}` 其中20表示签名长度，签名长度与变量名之间通过英文逗号分隔，且相互之间不能有空格。

短信模板FAQ

短信模板可以大于500字吗?

可以，但建议不要超过450字，如有特殊需求，请联系销售进行沟通协商。

短信模板可免审核吗?

可以，请联系销售进行沟通协调。

短信模板删除后是否可发送短信?

短信模板删除后，不能发送短信。

短信模板变量规范有什么要求?

类别	公共内容规范
格式	• 模板长度限制：建议1~500个字（含变量） • 为避免与签名混淆，在模板内容任意位置均不能使用【】，在模板内容收尾不能使用[] • 支持中文、英文、数字、符号，但不支持特殊符号，请勿直接从word、网页中直接拷贝内容，避免存在特殊符号

内容	短信模板需明确表述短信发送的实际内容。 • 禁止发送与金融相关的所有内容。 • 地产、留学、招聘、交友、游戏等行业仅支持发送验证码。 • 不支持发送未经许可的信息，主要指邀请注册、邀请成为会员的商业性信息。 • 不支持内容中含有直接或间接访问应用内测分发平台的行为。 • 不支持发送涉及数据排名、关键字搜索、精准拓客引流营销的内容。 • 禁止发送涉及：色情、赌博、毒品、党政、法律维权、众筹、慈善募捐、宗教、迷信、股票、留学移民、面试招聘、商标专利、博彩、贷款、催款还款、信用卡提额、投资理财、中奖、抽奖、一元夺宝、一元秒杀、一元云购、二类电商、A货、整容、烟酒、交友、暴力、恐吓、皮草、返现返利、代开发票、运营商禁止发送的信息、代理注册、代办证件、加群、加QQ或者加微信、贩卖个人信息、运营商业务、流量营销、违反广告法用语、殡葬、刷单、做任务、空包网、邀请好评、转店类、拉新、众包业务、POS机、积分兑换等内容的短信。 • 禁止在关键字或关键信息中出现错别字、变体字、异体字、各类干扰符号等，例如威信；禁止出现各类非正常混合字以及非常用的表达法。
链接	• 请提供可以访问且有ICP备案的网站地址。 • 跳转链接存在安全风险的情况下，可以压缩成短链接。 • 若提供下载链接或内测链接，审核方无法核实业务内容，建议上线后提供应用商城链接再申请。

短信内容可包含链接或短链吗?

可以包含链接，也可以转长链接转换为短链，短链在短信模板创建时添加。

短信模板删除后可恢复吗?

可以，请联系销售进行沟通协调。

1天内同模板是否可以给同一号码发送短信条数限制?

可以，在短信模板创建时配置。

短信模板是否可设置单次发送总人数限制?

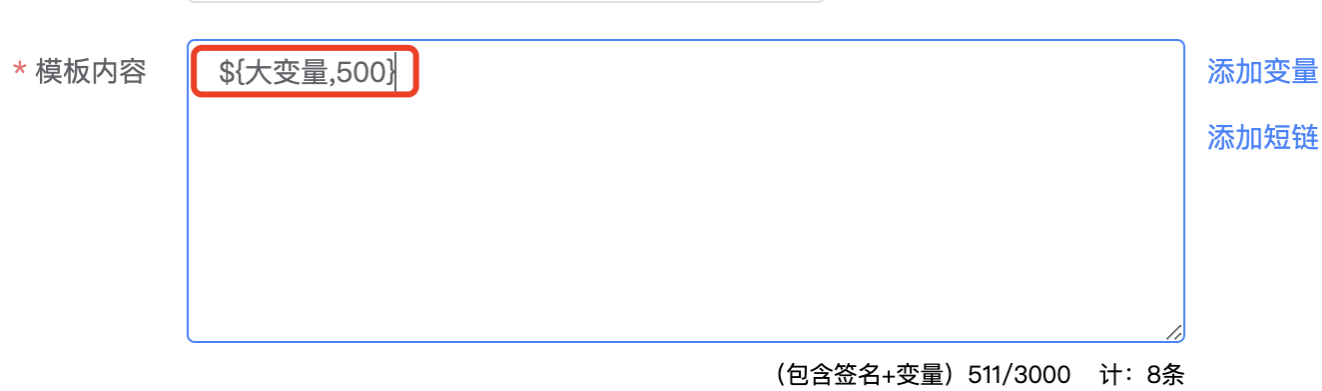
可以，在短信模板创建时配置。

短信模板是否可设置日发送频次限制?

可以，在短信模板创建时配置。

短信内容不固定，发送时传入短信内容是否支持？内容是否会被拦截？

支持，配置短信模板时，配置为大模板变量，但实际发送时，部分敏感关键字可能会被拦截，导致发送失败。



已发布的模板可修改模板内容吗？可修改模板限制吗？

已审核通过发布的短信模板不能修改模板内容；也不可修改模板限制。

短信发送FAQ

请检查传入参数是否完整是什么原因导致?

请参考短信发送API规范及对应的代码示例，确认接口所需要的参数均按要求传入。若还存在问题，请提供接口响应报文中的 `requestId` 给平台技术支持人员排查问题。

API授权失败或授权参数不完整是什么原因导致?

请参考短信发送API规范及对应的代码示例，确认接口所需要的参数均按要求传入。若还存在问题，请提供接口响应报文中的 `requestId` 给平台技术支持人员排查问题。

接口调用时提示'无效的参数（时间戳）'是什么原因导致?

- 请检查调用端所在服务器时间是否是北京时间，若不是，请自行校准服务器时间或使用这个接口

📎经分短信(安全性低)API.zip进行对接

- 当前短信服务端时间戳校准误差是60秒，即调用端服务器时间与短信服务器时间误差在60秒内的请求均可通过验证

短信API接口吞吐量多少?

当前短信API接口吞吐量约300/TPS，基本上能满足当前业务诉求，若存在更大发送要求，请联系销售/运营。

需要批量发送短信是每个手机号调用一次接口吗?

不是，存在大批量短信发送时，建议每次提交多个手机号，不要每个手机号调用一次接口，这样发送执行效率低下，批量发送分两种场景

- 同短信内容不同手机号 建议使用 `message/send` 接口实现，详情见接口文档
- 不同短信内容不同手机号 建议使用 `message/sendBatch` 接口实现，详情见接口文档

接口调用时提示‘短信变量个数和模板变量个数不一样’是什么原因导致?

请参考短信发送API规范及对应的代码示例，确认接口所需要的参数均按要求传入。若还存在问题，请提供接口响应报文中的 `requestId` 给平台技术支持人员排查问题。

接口调用时偶尔提示‘短信资源被锁定’是什么原因导致?

若只是偶尔出现，该问题可能是短信服务平台网络波动导致，针对这类问题，在短信发送时，建议调用端增加重试机制，尽量避免这类问题导致短信提交失败；目前短信服务平台也在完善网络及服务建设，尽量避免网络波动导致此类问题。

接口调用时提示超过了发送速度限制是什么原因导致?

查询类接口默认每个企业每分钟只允许请求三次，超过请求次数，则提示该信息，请合理调整查询类接口请求频次；也可调整短信状态获取方式为服务端推送模式，具体配置请参考 `短信回执状态推送` 或 `短信上行报告推送`

接口短信是否支持定时发送?

不支持

短信发送后是否可撤回?

发送后不支持撤回

未回执的短信会在多久时间更新状态?

“未知状态”短信即发送中的短信。

- 若72小时内，客户终端正常接收到短信，短信状态更新为“发送成功”；未能正常接收到短信，短信状态更新为“发送失败”。
- 若超过72小时，短信状态为“未知状态”。

短信发送成功了，但用户未收到是什么原因?

可能有如下原因：

- 手机长时间未关机，建议将手机关机后重启。
- 手机短信存储空间已满，建议删除一些无用短信再查看是否可以正常接收。
- 手机是双卡双待，建议将卡取出交换一下卡槽。
- 手机安装了安全软件，安全软件将短信拦截到垃圾信箱里。

> 如若上述操作后仍未收到短信，请您将接收短信的SIM卡换到其他手机上查看。

短信发送支持回执失败自动重发吗?

支持，请联系销售沟通协调。

短信发送是否支持发送时间限制?

支持，可在模板创建时设置。

如何判断短信发送成功还是失败?

- 可以在控制台查询发送记录确认是否发送成功，也可以查看API/SDK回执状态消息确认是否发

送成功。

- 通过API `返回码说明` 或 `短信发送状态回执码` ，查看发送失败原因。

短信发送失败的可能原因

- 携号转网的短信发送，多数情况下携号转网的信息也能到达。如果运营商未及时更新携号转网路由信息或携号转网24小时内的情况，可能导致短信发送失败。
- 运营商拦截，短信内容可能触发运营商的审核机制，运营商拦截后短信发送失败。
- 终端状态异常，终端网络信号导致接收短信失败。
- 错误内容，如手机不支持此类语言可能导致接收短信失败。
- 黑名单，机主主动投诉或退订导致发送短信的号码被运营商加入黑名单。
- 国际漫游，取决于运营商之间的国际漫游互联协议，国际漫游不保证一定能接收成功。
- 运营商过滤，部分运营商对短信内容进行“敏感词”过滤。
- 账户余额≤0，国内短信套餐包有余量但账户余额≤0时无法发送短信。

给用户发送短信后，如何获取用户的短信回复?

请联系销售沟通，沟通后，接口用户可直接获取上行内容，详情参见 `短信上行报告推送`

短信发送方的号码可现实同一个号码吗?

可以，若存在特殊号码需求，请联系销售。

短信发送失败，但是没有发送失败记录?

API调用失败或发送号码非手机号等情况时，是没有发送失败记录的；若接口已请求到服务器，此时可提供接口响应报文对应的request id给平台技术支持人员排查问题。

> 请求日志记录只能查询最近4天的，超过4天不能查询。

自定义签名、内容发送时，签名和内容放在一起还是分开两个参数?

按变量处理，签名一个变量、内容一个变量

JSON

```
{
    "phones":"1369941xx,1369941xx",
    "templateCode":"8392985760770416xx",
    "templateParam":"[\"短信签名\",\"短信内容\"]"
}
```