**支付接口**

**目录**

[1. 接口的统一说明 2](#_Toc520293899)

[1.1 参数 2](#_Toc520293900)

[1.2 调用方式 2](#_Toc520293901)

[1.3 域长度计算说明 2](#_Toc520293902)

[1.4 服务器返回报文说明 2](#_Toc520293903)

[1.5 服务器地址 2](#_Toc520293904)

[2. 接口描述 2](#_Toc520293905)

[2.1 服务器公钥 2](#_Toc520293906)

[2.2 支付请求接口 3](#_Toc520293907)

[2.3 支付通知（异步） 4](#_Toc520293908)

[2.4 订单查询接口 4](#_Toc520293909)

[3. 签名验签 5](#_Toc520293910)

[3.1 客户端签名( JAVA demo) 5](#_Toc520293911)

[3.2 客户端验签( JAVA demo ) 5](#_Toc520293912)

[4. 测试用商户号及密钥 6](#_Toc520293913)

[4.1 签名样例 6](#_Toc520293914)

[4.2 Rsa秘钥生成算法 7](#_Toc520293915)

# 接口的统一说明

## 参数

接口传输的所有参数（特别是一些中文，非ASCII字符集字符）都使用UTF-8进行编码。

## 调用方式

接口采用http连接方式进行通讯，所有参数均可以GET、POST提交，以key

,value方式提交。

## 域长度计算说明

所有参数的长度均以字节为单位。

## 服务器返回报文说明

服务器返回报文，以json格式

## 服务器地址

测试环境：https://api.wispay388.com/

生产环境：https://api.wispay388.com/

# 接口描述

## 服务器公钥

服务器返回的报文，请用此公钥验签

生产环境：

308189028181009a85b39acd0ee5aa0ee89b094d92ec0b9c6323191d6d32f95350f9ca4e891307a8ae45841082170435b0b6f054eaed15c50aa67dbdad8e176eb0684826181dd760cec48909310c2d6e93a97e4b8a9226826f8521d90968509b5cca05cc459da43dae29e0c848abf3ddd64cee4966f4e4cb8a19fdb70ba93a7734572310a5b3b30203010001

## 支付请求接口

* 提交接口地址

https://api.wispay388.com/EBankPay

* 请求接口 （商户->平台）

| **参数** | **说明** | **长度** | **必须** | **备注** |
| --- | --- | --- | --- | --- |
| version | 版本号 | 8 | Y | 固定填“20180221” |
| merId | 商户号 | 6 | Y | 商户号 |
| transDate | 商户日期 | 8 | Y | yyyymmdd，商户交易的日期 |
| seqId | 商户订单号 | 32 | Y | 合作伙伴系统中的订单号 |
| transTime | 交易时间 | 6 | Y | HHmmss |
| amount | 订单金额 | 13 | Y | 以分为单位的整数 |
| notifyUrl | 通知URL | 128 | Y | 异步通知接收地址 |
| returnUrl | 同步返回地址 | 128 | Y | 前台同步返回地址 |
| subject | 订单标题 | 50 | Y | 商品名称 |
| body | 商品描述 | 50 | Y | 商品的具体描述 |
| cardType | 支付卡类型 | 8 | Y | 01：储蓄卡  02：信用卡 |
| payType | 支付产品类型 | 2 | Y | 00：网银 01：快捷  10：微信H5 11：支付宝H5  12：京东H5 13：银联H5  20：qq钱包 21：京东钱包  30：银联扫码 31：支付宝扫码 |
| bankId | 银行代码 | 4 | **N** | 银行编码 |
| channel | 来源类型 | 3 | **N** | 01（PC端）、 02（手机端） |
| remark | 备注 | 128 | **N** | 备注 |
| sign | 签名 | 128 | Y | 设所有发送或者接收到的数据为集合M，将集合M内非空参数值的参数按照参数名ASCII码从小到大排序（字典序），使用URL键值对的格式（即key1=value1&key2=value2…）拼接成字符串stringA。以此为明文，进行rsa签名 |

## 支付通知（异步）

* 请求接口 平台->商户

| **参数** | **说明** | **类型** | **长度** | **必须** | **备注** |
| --- | --- | --- | --- | --- | --- |
| version | 版本号 | String | 8 | Y | 固定填“20180221” |
| merId | 商户号 | string | 6 | Y | 商户号 |
| stat | 订单状态 | string | 4 | Y | 0000：交易成功，其他表示失败  1111：初始状态 |
| transDate | 商户订单日期 | string | 8 | Y | 商户订单日期 |
| seqId | 商户流水号 | string | 32 | Y | 商户流水号 |
| amount | 订单金额 | string | 13 | Y | 交易金额。以分为单位 |
| sign | 签名 | string | 128 | Y | 明文拼接规则同上，用服务器公钥验签 |

* 应答 （请务必返回）

| **参数** | **说明** | **类型** | **长度** | **必须** | **备注** |
| --- | --- | --- | --- | --- | --- |
| code | 应答码 | String | 20 | Y | 成功：SUCCESS  失败：ERROR |
| msg | 应答描述 | string | 20 | Y | 根据情况填写 |

应答例子:

{"code":"SUCCESS","msg":"ok"}

## 订单查询接口

* 提交接口地址

https://api.wispay388.com/EBankPayQuery

* 请求接口

| **参数** | **说明** | **类型** | **长度** | **必须** | **备注** |
| --- | --- | --- | --- | --- | --- |
| version | 版本号 | String | 8 | Y | 固定填“20180221” |
| merId | 商户号 | string | 6 | Y | 商户号 |
| transDate | 商户订单日期 | string | 8 | Y | 商户订单日期 |
| seqId | 商户流水号 | string | 32 | Y | 商户流水号 |
| sign | 签名 | string | 128 | Y | 明文拼接规则同上，用服务器公钥验签 |

* 应答接口

| **参数** | **说明** | **类型** | **长度** | **必须** | **备注** |
| --- | --- | --- | --- | --- | --- |
| code | 应答码 | String | 20 | Y | 查询成功：0000  查询失败：非0000均为失败 |
| msg | 应答描述 | string | 20 | Y | 根据情况填写 |
| merId | 商户号 | string | 6 | Y | 商户号 |
| transDate | 商户订单日期 | string | 8 | Y | 商户订单日期 |
| seqId | 商户流水号 | string | 32 | Y | 商户流水号 |
| amount | 订单金额 | string | 13 | Y | 交易金额。以分为单位 |
| stat | 订单状态 | string | 4 | Y | 0000：交易成功,其他表示失败  1111：处理中 |
| sign | 签名 | string | 128 | Y | 明文拼接规则同上，用服务器公钥验签 |

返回例子:

{"code":"0000","message":"SUCCESS","merId":"000012","transDate":"20180107","seqId":"20180101131120","amount":"100","stat":"0000","sign":"xxx"}

# 签名验签

## 客户端签名( JAVA demo)

用商户私钥进行签名，样例：

//reqMap中存放pos参数，

Map<String, String> reqMap = **new** HashMap<>();

reqMap.put("version", "20180221");

reqMap.put("merId", bean.getMerId());

reqMap.put("stat", bean.getStat());

reqMap.put("transDate", bean.getTransDate());

reqMap.put("seqId", bean.getSeqId());

reqMap.put("amount", bean.getAmount());

String signedMsgStr = signUtils.lib.NySignatureStrUtil.*genSignStr*(reqMap);

String sign = **new** signUtils.lib.NySignatureData().sign(signedMsgStr, *MER\_PRI\_KEY*);

## 客户端验签( JAVA demo )

用**服务器公钥（见2.1章）**进行验签，样例：

// paraMap为传入的参数的hashmap。

String signedMsgStr = NySignatureStrUtil.genSignStr(paraMap);//获取排好序的签名明文

**if**(**new** NyVerifySignature().run(signedMsgStr, PUB\_KEY, reqSign)){

TraceLog.*debug*(**this**, "mer verify success"+bean.getSeqId());

**return** **true**;

}**else**{

TraceLog.*error*(**this**, "mer verify error!!"+bean.getSeqId());

respMap.put("respMsg", "商户验签失败");

**return** **false**;

}

# 测试用商户号及密钥

## 签名样例

测试商户号：000012

测试商户密钥：



调用签名：

String sign = **new** signUtils.lib.NySignatureData().sign("123", *PRI\_KEY*);// *PRI\_KEY为上面的私钥*

System.***out***.println(sign);

打印结果：

5BCD7D87A2269DACF823897CBA9BCBBC0958FD2F953877164D79481688445EA86FFF01E85F3C645D7DFA2E1DC51E603207D02A54BB49D9B7C9436859C9BEC0C15BB4CBB2B13989A3CA063DDE24E3CA177A765D73A53A4F30D6A92908EB6811005B4A9BF1E2F3791BF60FCE5F85E3AFFCBA2DF81883DA9EA5AD2D340165D3F72D

## Rsa秘钥生成算法

KeyPairGenerator keygen = KeyPairGenerator.getInstance("RSA");

SecureRandom secrand = new SecureRandom();

secrand.setSeed(key.getBytes());

keygen.initialize(1024, secrand);

KeyPair keys = keygen.genKeyPair();

PublicKey pubkey = keys.getPublic();

PrivateKey prikey = keys.getPrivate();

this.pubKey = NyChangeKeyPair.getRsaPublicKeyDerFromX509(bytesToHexStr(pubkey.getEncoded()));

this.priKey = bytesToHexStr(prikey.getEncoded());