一、签名算法

** 签名规则**

> 将需要参与签名的参数按Key值的升序排序，并按照key1value1key2value2...的顺序拼接在一起组成等签名字符串，用accessSecret对该串进行HmacSHA1加密，将加密后的字节数组Base64编码进行生成签名串。注意整个处理是采用utf-8编码。
>
> ** 凡需要签名参数，即使为空，也需要拼凑（即不做排空处理）**

- ** 例： **
  待签名的源数据为 `{"timestamp":"1590030050868","accessKey":"abcdes"}`，按key排序并组合为 `accessKeyabcdestimestamp1590030050868`，accessSecret 为 `4A09289E1E99`，加密后生成的签名串为 `8A2UCKlKznjAJ1rUkndmBNVvkM0=`。
- ** Java算法Demo： **

```
public static void main(){
    java.util.Map<String, String> paras = new java.util.HashMap<String, Strir
    paras.put("accessKey", "abcdes");
    paras.put("timestamp","1590030050868");

    java.util.TreeMap<String, String> sortParas = new java.util.TreeMap<Strir
    sortParas.putAll(paras);

    java.util.Iterator<String> it = sortParas.keySet().iterator();

    StringBuilder sortQueryStringTmp = new StringBuilder();
    while (it.hasNext()) {
        String key = it.next();
        sortQueryStringTmp.append(key).append(paras.get(key));
    }

    String sign =sign("4A09289E1E99" , sortQueryStringTmp.toString());
}

public static String sign(String accessSecret, String stringToSign) throws E
    javax.crypto.Mac mac = javax.crypto.Mac.getInstance("HmacSHA1");
    mac.init(new javax.crypto.spec.SecretKeySpec(accessSecret.getBytes("UTF-8
    byte[] signData = mac.doFinal(stringToSign.getBytes("UTF-8"));
    return new sun.misc.BASE64Encoder().encode(signData);
}
```

- C#算法Demo：

```
public static void Main(){
    SortedList<string, string> ts = new SortedList<string, string>();
    ts.Add("accessKey", "abcdes");
    ts.Add("timestamp", "1590030050868");
    StringBuilder sb = new StringBuilder();
    foreach (var item in ts)
    {
        sb.Append(item.Key).Append(item.Value);
    }
    string si= HmacSha1("4A09289E1E99", sb.ToString());
}

 public static string HmacSha1(string key, string input)
 {
     byte[] keyBytes = Encoding.UTF8.GetBytes(key);
     byte[] inputBytes = Encoding.UTF8.GetBytes(input);
     System.Security.Cryptography.HMACSHA1 hmac = new System.Security.Cryptog
     byte[] hashBytes = hmac.ComputeHash(inputBytes);
     return Convert.ToBase64String(hashBytes);
 }
```

二、直充产品下单接口

简要描述：

- 话费下单接口
- 参数名称、值对大小写敏感

** 请求规则 **

- 接口默认请求方式为Post，请求数据格式为 Json 方式，响应数据格式统一为：json。

请求URL：

- `http://xx.com/order/insert`

Content-Type：

- application/json; charset=utf-8

请求参数：

参数名	必选	类型	是否参与签名	说明
phone	是	string	是	手机号或充值账号
access_key	是	string	是	商户凭证
par_value	是	int	是	面值
code	是	string	是	产品编码
product type	是	int	是	产品类型 1:快充 0:其它
out_trade_no	是	string	是	客户流水号
notify_url	是	string	否	回调地址
timestamp	是	long	是	时间戳
sign	是	string	否	签名，采用HmacSHA1，见签名算法章节

参数名	必选	类型	是否参与签名	说明
pass	否	string	否	透传参数
extra	否	map	否	扩展参数，若不需要传此参数,即此值为空，在json中请保持 {}而不是""

extra： 当特殊产品时才会需要此参数。

参数名	类型	说明
card_name	string	中石油油卡需要，持卡人姓名。
idcard	string	中石油油卡需要，持卡人完整身份证号。 南网户号电费充值需要，身份证后六位。
count	int	需要多数量充值的产品，通过该参数指定充值数量
type	int	电费充值时需要， 1住宅，2店铺，4企事业
area	string	电费充值时需要， 北京,天津,辽宁,江苏,浙江,山东,湖南,陕西,冀北,新疆,宁夏,安徽,重庆,福建,河南,青海,湖北,蒙东,黑龙江,吉林,上海,甘肃,山西,四川,江西,河北,广东,广西,云南,贵州,海南,香港,澳门。

请求示例

- 以下示例：accessSecret 为 `4A09289E1E99`

```
{
    "access_key":"abcdes",
    "code":"P0001",
    "notify_url":"http://localhost:9000/order/callback",
    "out_trade_no":"20180101001",
```

```
    "par_value":"10",
    "phone":"13224011030",
    "producttype":"1",
    "sign":"EFXW3O2NpTByrDUBF8mEwqwTOp0=",
    "timestamp":"1590030050868"
}
```

签名算法代码示例：

返回参数说明

参数名	类型	说明
code	int	返回码；参见code值处理说明
msg	string	描述信息
data	map	数据内容

返回参数内容（data）

参数名	类型	说明
orderid	long	订单号
sellprice	decimal	销售金额

返回示例

```
{
    "code": 1,
    "msg": "收单成功",
    "data": {
        "orderid": 200521230215315000,
```

```
        "sellprice": 0.06
    }
}
```

code值处理说明（除下列以外其它值均需要核实订单状态，不可直接做失败处理）

code	说明	是否可以做失败处理
0	未知异常	否，需人工或通过查询接口确认订单状态
1	下单成功	否
2	订单已存在	否，需通过查询接口或通过data中的orderstate确认结果
3	不符合收单条件	是

Java代码：

```
private static void insert() throws Exception {
        java.util.Map<String, Object> paras = new java.util.HashMap<String, O
        paras.put("phone", "13224011030");
        paras.put("access_key", "211394650");
        paras.put("par_value", 10);
        paras.put("code", "SKU100001");
        paras.put("producttype", 1);
        paras.put("out_trade_no", "20180101001");
        paras.put("timestamp", Long.parseLong("1590030050868"));
        java.util.TreeMap<String, Object> sortParas = new java.util.TreeMap<S
        sortParas.putAll(paras);
        java.util.Iterator<String> it = sortParas.keySet().iterator();
        StringBuilder sortQueryStringTmp = new StringBuilder();
        while (it.hasNext()) {
            String key = it.next();
            sortQueryStringTmp.append(key).append(paras.get(key));
        }
        String sign =sign("zHCp7cEUy0u55LMFGYEWuA==" , sortQueryStringTmp.toS
        paras.put("sign", sign);
        paras.put("notify_url", "http://127.0.0.1");
        paras.put("pass", "");
```

```
        paras.put("extra", new HashMap<String,Object>());
        System.out.println(sign);
        //Content-Type: application/json; charset=utf-8
        String result=  HttpClientUtil.post("http://123.56.253.5:8081/order/i
        System.out.println(result);
    }
```

三、订单查询接口

简要描述：

- 订单查询接口
- 参数名称、值对大小写敏感

** 请求规则 **

- 接口请求方式为Post，请求数据格式为 Json 方式，响应数据格式统一为：json 。

请求URL：

- `http://xx.com/order/query`

Content-Type：

- application/json; charset=utf-8

请求参数：

参数名	必选	类型	是否参与签名	说明
access_key	是	string	是	商户凭证
orderid	否	string	是	订单号（避免因为精度问题导致尾号不正确） orderid与out_trade_no必须二选一
out_trade_no	否	string	是	客户流水号 orderid与out_trade_no必须二选一
timestamp	是	long	是	时间戳
sign	是	string	否	签名，采用HmacSHA1，见签名算法章节

请求示例

- 以下示例：accessSecret 为 `4A09289E1E99`

```
{
    "access_key":"abcdes",
    "orderid":"",
    "out_trade_no":"20180101001",
    "sign":"4E7Ku7G0TqxdcTQ7fgue+7Vqlxw=",
    "timestamp":"1590030050868"
}
```

签名算法代码示例：

返回参数说明

参数名	类型	说明
code	int	订单状态；0:未知 1:充值中 2:订单成功 3:订单失败 4:查询请求失败 5:订单不存在（1分钟后查询有效且需要保证自己所传参数准确无误） 除2、3、5以外均不能作最终结果处理
msg	string	描述信息
data	map	数据内容

返回参数内容（data）

参数名	类型	说明
out_trade_no	string	客户流水号
orderid	long	订单号（避免因为精度问题导致尾号不正确）
sellprice	decimal	销售金额
phone	string	充值账号
official_id	string	官方流水号，并非所有订单都有流水号
official_ch	string	充值通道类型，如：手支

返回示例

```
{
    "code": 1,
    "msg": "充值中",
    "data": {
        "out_trade_no": "20180101001",
        "orderid": 200521230215315000,
        "sellprice": 0.06,
        "phone": "13224011030",
        "official_id":"982000165965526",
        "official_ch":"其它"
    }
}
```

Java代码：

```
private static void query() throws Exception {
        java.util.Map<String, String> paras = new java.util.HashMap<String, S
        paras.put("access_key", "211394650");
        paras.put("orderid", "");
        paras.put("out_trade_no", "20180101001");
        paras.put("timestamp", "1590030050868");
        java.util.TreeMap<String, String> sortParas = new java.util.TreeMap<S
        sortParas.putAll(paras);
        java.util.Iterator<String> it = sortParas.keySet().iterator();
        StringBuilder sortQueryStringTmp = new StringBuilder();
        while (it.hasNext()) {
            String key = it.next();
            sortQueryStringTmp.append(key).append(paras.get(key));
        }
        String sign =sign("zHCp7cEUy0u55LMFGYEWuA==" , sortQueryStringTmp.toS
        System.out.println(sign);
        paras.put("sign", sign);
        //Content-Type: application/json; charset=utf-8
        String result=  HttpClientUtil.post("http://123.56.253.5:8071/order/c
        System.out.println(result);

    }
```

```
private static void query() throws Exception {
    java.util.Map<String, String> paras = new java.util.HashMap<String, String>();  paras:  size = 5
    paras.put("access_key", "211394650");
    paras.put("orderid", "");
    paras.put("out_trade_no", "20180101001");
    paras.put("timestamp", "1590030050868");
    java.util.TreeMap<String, String> sortParas = new java.util.TreeMap<String, String>();  sortParas:  size = 4
    sortParas.putAll(paras);
    java.util.Iterator<String> it = sortParas.keySet().iterator();  it: TreeMap$KeyIterator@1790  sortParas:  size = 4
    StringBuilder sortQueryStringTmp = new StringBuilder();  sortQueryStringTmp: "access_key211394650orderidout_trade_no20180101001timestamp1590030050868"
    while (it.hasNext()) {
        String key = it.next();  it: TreeMap$KeyIterator@1790
        sortQueryStringTmp.append(key).append(paras.get(key));
    }
    String sign =sign( accessSecret: "zHCp7cRUy0u55LMPGYRWuA==", sortQueryStringTmp.toString());  sign: "P3ULHGw7+yTh6j/HRdMiL15Fcuw="  sortQueryStringTmp: "access_key211394650orderidout_trade_no20180101001timestamp1590030050868
    System.out.println(sign);
    paras.put("sign", sign);  sign: "P3ULHGw7+yTh6j/HRdMiL15Fcuw="
    //Content-Type: application/json; charset=utf-8
    String result=  HttpClientUtil.post( url: "http://121.89.161.176:8071/order/query", JSONObject.toJSONString(paras), encoding: "UTF-8");  result: "{"code":3,"msg":"充值失败","data":{"out_trade_no":"20180101001","orderid":
    System.out.println(result);  result: "{"code":3,"msg":"充值失败","data":{"out_trade_no":"20180101001","orderid":200524005936037000,"sellprice":10,"phone":"13224011030","official_id":null,"official_ch":null}}"
```

四、订单回调功能

简要描述：

- 当平台有最终充值结果后，会通过此接口异步返回结果给会员（下游）。
- 参数名称、值对大小写敏感

请求规则

- 接口默认请求方式为Post，请求数据格式为 Json 方式，响应数据格式统一为：json。

请求URL：

- 回调地址为下单接口中的“notify_url”参数。

Content-Type：

- application/json; charset=utf-8

请求参数：

参数名	类型	是否参与签名	说明
access_key	string	是	商户凭证
orderid	long	是	订单号 orderid与out_trade_no必须二选一
out_trade_no	string	是	客户流水号 orderid与out_trade_no必须二选一
timestamp	long	是	时间戳 （1970-01-01至当前时间的总毫秒数）
code	int	是	订单状态；2:订单成功 3:订单失败 ，注：若返回其它值，请与我方平台核实后再作处理
sign	string	否	签名，采用HmacSHA1，见签名算法章节
data	map	否	数据内容

请求参数内容（data）

参数名	类型	说明
sellprice	decimal	销售金额
phone	string	充值账号
pass	string	透传参数，与收单中的透传参数一致
official_id	string	官方流水号，并非所有订单都有流水号
official_ch	string	充值通道类型，如：手支

请求示例

- 以下示例：accessSecret 为 `4A09289E1E99`

```
{
    "access_key":"211394650",
    "code":2,
    "data":{
        "sellprice":10,
        "phone":"13224011031",
        "pass":null,
        "official_id":null,
        "official_ch":null
    },
    "orderid":200528105302018000,
    "out_trade_no":"20200528105258",
    "sign":"7M3x1eICIQSUaCHujev17dIFKCo=",
    "timestamp":1590634458337
}
```

返回参数说明

参数名称	类型	说明
code	int	0:接收成功

返回示例：

`{"code":0}`

回调备注：

- 会员（下游）成功接收到通知，需要返回code为0的数据，我方平台判定为会员已经接收成功。
- 会员已成功接收通知，我方平台会停止当前订单的自动回调操作。
- 我方平台发起回调操作失败后（用户接口异常或未按指定格式返回数据），会根据订单回调策略进行再次回调。
- 订单回调策略：

  > *回调最大次数为6次，第一次回调失败（请求响应超时、返回码指定数据、其他异常等）后，服务系统将依次间隔10秒，1分钟，10分钟，1小时，10小时进行推送，推送成功或推送6次都失败都将结束推送任务。*

五、余额查询接口

简要描述：

- 会员（下游）余额查询接口

请求规则

- 接口请求方式为Post，请求数据格式为 Json 方式，响应数据格式统一为：json 。

请求URL：

`http://xx.com/member/account`

Content-Type：

`application/json; charset=utf-8`

请求参数：

参数名	必选	类型	是否参与签名	说明
access_key	是	string	是	商户凭证，请注意参数名是“**access_key**”，不是签名算法demo中的“~~accessKey~~”
timestamp	是	long	是	时间戳
sign	是	string	否	签名，采用HmacSHA1，见签名算法章节

请求示例

- 以下示例：accessSecret 为 `4A09289E1E99`

```
{
    "access_key":"abcdes",
    "sign":"lNdSBH5EP2uBEHyNV4jlBmfTJ14=",
    "timestamp":"1590030050868"
}
```

返回参数

参数名	类型	说明
code	int	0:请求失败 1:请求成功
msg	string	描述信息
balance	decimal	余额

返回示例

```
{
    "code":1,
```

```
    "msg":"请求成功",
    "balance":-150
}
```

Java代码：

```
private static void account() throws Exception {
        java.util.Map<String, String> paras = new java.util.HashMap<String, S
        paras.put("access_key", "211394650");
        paras.put("timestamp", "1590030050868");
        java.util.TreeMap<String, String> sortParas = new java.util.TreeMap<S
        sortParas.putAll(paras);
        java.util.Iterator<String> it = sortParas.keySet().iterator();
        StringBuilder sortQueryStringTmp = new StringBuilder();
        while (it.hasNext()) {
            String key = it.next();
            sortQueryStringTmp.append(key).append(paras.get(key));
        }
        String sign =sign("zHCp7cEUy0u55LMFGYEWuA==" , sortQueryStringTmp.toS
        paras.put("sign", sign);
        System.out.println(sign);
        //Content-Type: application/json; charset=utf-8
        String result=  HttpClientUtil.post("http://123.56.253.5:8071/member/
        System.out.println(result);
    }
```

六、联调流程说明、地址、账号信息

流程

会员先用测试环境联调通过后，再使用生产环境账号及产品正式跑单。

测试通过后，请联系我司商务开通正式环境账及及配置产品价格、上架。

七、常见问题解答

- **1、收单接口提示50X错误。**

请检查收单接口中的“extra”参数，该参数是map对象（**{}**），不是字符串（**""**）。

extra	否	map	否	扩展参数

```
paras.put("notify_url", "http://127.0.0.1");
paras.put("pass", "");
paras.put("extra", new HashMap<String,Object>());
System.out.println(sign);  sign: "bFqdsWaxChqS/SFvb3uANdvf7j
```

- **2、收单接口怎么样都调不通。**

请直接用各接口中给出代码示例测试，不要过渡自信认为自己写的代码没有任何问题，没有必须把时间浪费在给自已代码找茬上，直接用别人现成的代码不好吗？

- **3、接口提示缺少参数。**
  请检查是否真的缺少参数，另外建议直接复制接口文档中的参数名，很多对接的小伙伴遇到问题后才发现，自己大小写搞错了，有的参数还少个下划线。如果以上检查都没有问题，那么需要检查一下contenttype。

  Content-Type:

  - application/json; charset=utf-8

- **4、提示产品未上架或配置有问题。**

请用测试地址、测试账号、测试产品编码进行测试，直接用生产信息测试会有很多问题（可能是您的代码有问题，也可能是您的账号没有绑定ip，也可能是您的余额没加上，也可能是没有给您配上您待测试的产品）。

八、卡密产品下单

简要描述:

- 卡密产品下单接口
- 参数名称、值对大小写敏感

** 请求规则 **

- 接口默认请求方式为Post，请求数据格式为 Json 方式，响应数据格式统一为：json
  。

请求URL：

- `http://xx.com/order/insertcard`

Content-Type：

- application/json; charset=utf-8

请求参数：

参数名	必选	类型	是否参与签名	说明
phone	否	string	是	手机号
access_key	是	string	是	商户凭证
code	是	string	是	产品编码
out_trade_no	是	string	是	客户流水号
notify_url	是	string	否	回调地址
timestamp	是	long	是	时间戳
public_key	是	string	是	公钥，用于对卡密进行加密，以便下游对卡密进行解密
sign	是	string	否	签名，采用HmacSHA1，见签名算法章节
pass	否	string	否	透传参数
extra	否	map	否	扩展参数

请求示例

- 以下示例：accessSecret 为 `zHCp7cEUy0u55LMFGYEWuA==`

```
{    "public_key":"MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCcciQPzzMllKD6RlP58d
    "code":"SKU100001",
    "out_trade_no":"20180101001",
    "phone":"13224011030",
    "pass":"",
    "access_key":"211394650",
    "extra":{
    },
    "sign":"QXLprzhx5ro5QGNkrXVv+2dBagk=",
    "notify_url":"http://127.0.0.1",
    "timestamp":1590030050868
}
```

签名算法代码示例：

```
java.util.Map<String, Object> paras = new java.util.HashMap<>();  paras:  size = 10
paras.put("phone", "13224011030");
paras.put("access_key", "211394650");
paras.put("code", "SKU100001");
paras.put("out_trade_no", "20180101001");
paras.put("timestamp", Long.parseLong( s: "1590030050868"));
paras.put("public_key", "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCcciQPzzMllKD6RlP58dnwp/Pllt9xYb/EXk3IJuL4Oo+MYWYI8P8NfBPa+ncYymIKLX7xv19Wa3Ebi0SxJnoJgYOPE4879YrT2EbXNrlG0P2DI4imCvDrnw4xCwpoLYRINrHobaB1Y
java.util.TreeMap<String, Object> sortParas = new java.util.TreeMap<>();  sortParas:  size = 6
sortParas.putAll(paras);
java.util.Iterator<String> it = sortParas.keySet().iterator();  it: TreeMap$KeyIterator@1103  sortParas:  size = 6
StringBuilder sortQueryStringTmp = new StringBuilder();  sortQueryStringTmp: "access_key211394650codeSKU100001out_trade_no20180101001phone13224011030public_keyMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCcciQPzzMllKD6RlP58dnwp/Pllt9xYb/EXk
while (it.hasNext()) {
    String key = it.next();  it: TreeMap$KeyIterator@1103
    sortQueryStringTmp.append(key).append(paras.get(key));
}
String sign =sign( accessSecret: "zHCp7cEUy0u55LMFGYEWuA==", sortQueryStringTmp.toString());  sign: "QXLprzhx5ro5QGNkrXVv+2dBagk="  sortQueryStringTmp: "access_key211394650codeSKU100001out_trade_no20180101001phone13224011030publi
paras.put("sign", sign);
paras.put("notify_url", "http://127.0.0.1");
paras.put("pass", "");
paras.put("extra", new HashMap<String,Object>());
System.out.println(sign);  sign: "QXLprzhx5ro5QGNkrXVv+2dBagk="
```

返回参数说明

参数名	类型	说明
code	int	返回码；参见code值处理说明
msg	string	描述信息
data	map	数据内容

返回参数内容（data）

参数名	类型	说明
orderid	long	订单号
sellprice	decimal	销售金额

返回示例

```
{
    "code": 1,
    "msg": "收单成功",
    "data": {
        "orderid": 200521230215315000,
        "sellprice": 0.06
    }
}
```

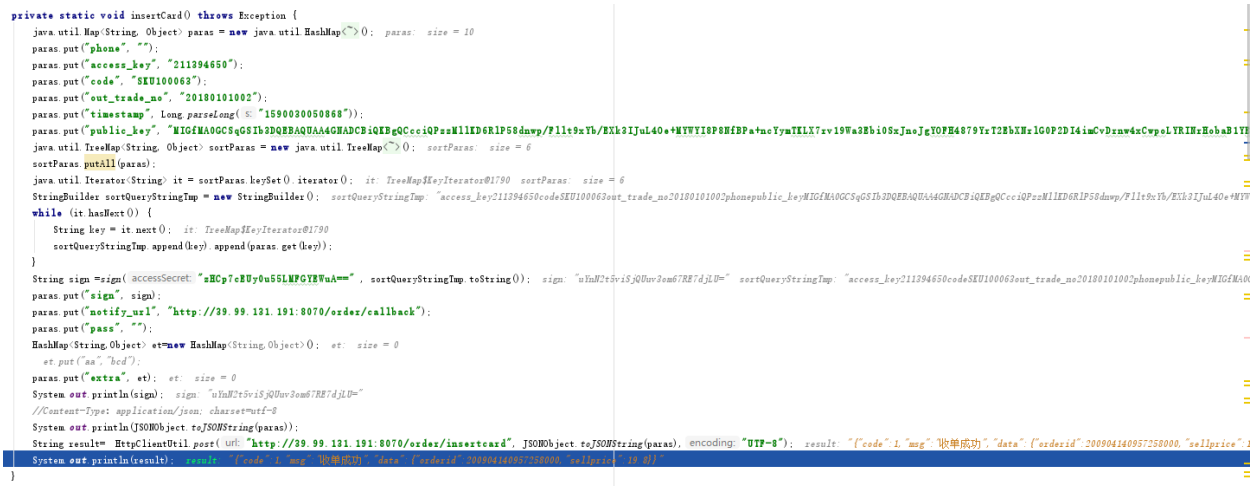
code值处理说明（除下列以外其它值均需要核实订单状态，不可直接做失败处理）

code	说明	是否可以做失败处理
0	未知异常	否，需人工或通过查询接口确认订单状态
1	下单成功	否
2	订单已存在	否，需通过查询接口或通过data中的orderstate确认结果
3	不符合收单条件	是

Java代码：

```
   private static void insertCard() throws Exception {
        java.util.Map<String, Object> paras = new java.util.HashMap<String, O
        paras.put("phone", "");
        paras.put("access_key", "211394650");
        paras.put("code", "SKU100063");
        paras.put("out_trade_no", "20180101002");
        paras.put("timestamp", Long.parseLong("1590030050868"));
```

```
        paras.put("public_key", "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCcciQP
        java.util.TreeMap<String, Object> sortParas = new java.util.TreeMap<S
        sortParas.putAll(paras);
        java.util.Iterator<String> it = sortParas.keySet().iterator();
        StringBuilder sortQueryStringTmp = new StringBuilder();
        while (it.hasNext()) {
            String key = it.next();
            sortQueryStringTmp.append(key).append(paras.get(key));
        }
        String sign =sign("zHCp7cEUy0u55LMFGYEWuA==" , sortQueryStringTmp.toS
        paras.put("sign", sign);
        paras.put("notify_url", "http://123.56.253.5:8070/order/callback");
        paras.put("pass", "");
        HashMap<String,Object> et=new HashMap<String,Object>();
//        et.put("aa","bcd");
        paras.put("extra", et);
        System.out.println(sign);
        //Content-Type: application/json; charset=utf-8
        System.out.println(JSONObject.toJSONString(paras));
        String result=  HttpClientUtil.post("http://123.56.253.5:8070/order/i
        System.out.println(result);
    }
```



九、卡密加解密RSA算法

Java:

- **Demo:**

```
 public static void main(String[] args) throws Exception {
        String sour="123456789SABDFDFDEFFK";
```

```
        System.out.println( "加密前："+ sour  );
        String str=  publickeyEncrypt(sour,pkey);
        System.out.println( "公钥加密后："+ str  );
        str= prikeydecript(prikey,str);
        System.out.println( "私钥解密后："+ str  );
    }
   static String pkey="MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCcciQPzzMllKD6RJ
  static  String prikey="MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBAJxy

    public  static  String  prikeydecript(String prikey, String str) throws E
        String res= RSATools.decrypt(RSAApplication.prikey,str);
        return  res;
    }

    public  static String publickeyEncrypt(String sour, String pkey) throws E
       String str= RSATools.encrypt(RSAApplication.pkey,sour);
       return str;
    }
```

- **RSATools**

```
package com.xxx.testappc;

import org.apache.commons.lang3.ArrayUtils;
import sun.misc.BASE64Decoder;
import sun.misc.BASE64Encoder;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import java.math.BigInteger;
import java.security.*;
import java.security.interfaces.RSAPrivateCrtKey;
import java.security.interfaces.RSAPrivateKey;
import java.security.interfaces.RSAPublicKey;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;

public class RSATools {

    /**
```

```
     * 公钥加密过程
     *
     * @param publicKey
     *            公钥
     * @param plainTextData
     *            明文数据
     * @return
     * @throws Exception
     *             加密过程中的异常信息
     */
    public static byte[] encrypt(RSAPublicKey publicKey, byte[] plainTextDat
        if (publicKey == null) {
            throw new Exception("加密公钥为空, 请设置");
        }
        Cipher cipher = null;
        try {
            // 使用默认RSA
            cipher = Cipher.getInstance("RSA");
            cipher.init(Cipher.ENCRYPT_MODE, publicKey);
            byte[] output = cipher.doFinal(plainTextData);
            return output;
        } catch (NoSuchAlgorithmException e) {
            throw new Exception("无此加密算法");
        } catch (NoSuchPaddingException e) {
            e.printStackTrace();
            return null;
        } catch (InvalidKeyException e) {
            throw new Exception("加密公钥非法,请检查");
        } catch (IllegalBlockSizeException e) {
            e.printStackTrace();
            throw new Exception("明文长度非法");
        } catch (BadPaddingException e) {
            throw new Exception("明文数据已损坏");
        }
    }

    /**
     * 公钥加密过程
     *
     * @param
     *
     * @param plainTextData
     *            明文数据
```

```
     * @return
     * @throws Exception
     *             加密过程中的异常信息
     */
    public static String encrypt(String pubKeyString, String plainTextData)
        RSAPublicKey publicKey = loadPublicKeyByStr(pubKeyString);
        byte[] encData = encrypt(publicKey, plainTextData.getBytes("utf-8"))
        return new BASE64Encoder().encode(encData);
    }

    /**
     * 从字符串中加载公钥
     *
     * @param publicKeyStr
     *            公钥数据字符串
     * @throws Exception
     *             加载公钥时产生的异常
     */
    public static RSAPublicKey loadPublicKeyByStr(String publicKeyStr) throw
        try {
            byte[] buffer =  new BASE64Decoder().decodeBuffer(publicKeyStr);
            KeyFactory keyFactory = KeyFactory.getInstance("RSA");
            X509EncodedKeySpec keySpec = new X509EncodedKeySpec(buffer);
            return (RSAPublicKey) keyFactory.generatePublic(keySpec);
        } catch (NoSuchAlgorithmException e) {
            throw new Exception("无此算法");
        } catch (InvalidKeySpecException e) {
            throw new Exception("公钥非法");
        } catch (NullPointerException e) {
            throw new Exception("公钥数据为空");
        }
    }

    public static String decrypt(String priKeyString, String cipherText) thr
        RSAPrivateKey privateKey = loadPrivateKeyByStr(priKeyString);
        byte[] cipherData =new BASE64Decoder().decodeBuffer(cipherText);
        byte[] decDatas = decrypt(privateKey,cipherData);
        return new String(decDatas);
    }
    public static RSAPrivateKey loadPrivateKeyByStr(String privateKeyStr) th
        try {
            byte[] buffer =new BASE64Decoder().decodeBuffer(privateKeyStr);
            PKCS8EncodedKeySpec keySpec = new PKCS8EncodedKeySpec(buffer);
```

```
            KeyFactory keyFactory = KeyFactory.getInstance("RSA");
            return (RSAPrivateKey) keyFactory.generatePrivate(keySpec);
        } catch (NoSuchAlgorithmException e) {
            throw new Exception("无此算法");
        } catch (InvalidKeySpecException e) {
            throw new Exception("私钥非法");
        } catch (NullPointerException e) {
            throw new Exception("私钥数据为空");
        }
    }
    /**
     * 私钥解密过程
     *
     * @param privateKey
     *            私钥
     * @param cipherData
     *            密文数据
     * @return 明文
     * @throws Exception
     *             解密过程中的异常信息
     */
    public static byte[] decrypt(RSAPrivateKey privateKey, byte[] cipherData
        if (privateKey == null) {
            throw new Exception("解密私钥为空, 请设置");
        }
        Cipher cipher = null;
        try {
            // 使用默认RSA
            cipher = Cipher.getInstance("RSA");
            cipher.init(Cipher.DECRYPT_MODE, privateKey);
            byte[] output = cipher.doFinal(cipherData);//解密
            return output;
        } catch (NoSuchAlgorithmException e) {
            throw new Exception("无此解密算法");
        } catch (NoSuchPaddingException e) {
            e.printStackTrace();
            return null;
        } catch (InvalidKeyException e) {
            throw new Exception("解密私钥非法,请检查");
        } catch (IllegalBlockSizeException e) {
            throw new Exception("密文长度非法");
        } catch (BadPaddingException e) {
            throw new Exception("密文数据已损坏");
```

```
        }
    }

}

```

c#:

```
using Org.BouncyCastle.Crypto;
using Org.BouncyCastle.Crypto.Parameters;
using Org.BouncyCastle.Security;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Security.Cryptography;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

 private static void RSADemo()
        {
            String sour = @"123456789SABDFDFDEFFK";
            string prikey = "MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGE
            string pkey = "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCcciQPzzMllK

            Console.WriteLine("加密前明文:" + sour);
            //公钥加密
            byte[] t_b = System.Text.Encoding.UTF8.GetBytes(sour);
            BouncyCastleHelper bbe = new BouncyCastleHelper();
            AsymmetricKeyParameter publicKey = bbe.GetPublicKey(Convert.FromB
            IAsymmetricBlockCipher enginee = new Org.BouncyCastle.Crypto.Enco
            enginee.Init(true, publicKey);
            var tem = enginee.ProcessBlock(t_b, 0, t_b.Length);
            String t_e = Convert.ToBase64String(tem);
            Console.WriteLine("加密后密文:" + t_e);
            //私钥解密:
            byte[] bytesToDecrypt = Convert.FromBase64String(t_e);
            BouncyCastleHelper bbd = new BouncyCastleHelper();
            var engined = new Org.BouncyCastle.Crypto.Encodings.Pkcs1Encoding
```

```
            engined.Init(false, bbd.GetPrivateKey(Convert.FromBase64String(pr
            var returnData = engined.ProcessBlock(bytesToDecrypt, 0, bytesToD
           string t_d = Encoding.UTF8.GetString(returnData);
            Console.WriteLine("解密后明文："+ t_d);
            Console.ReadKey();
        }
```

十、卡密查询接口

简要描述：

- 订单查询接口
- 参数名称、值对大小写敏感

** 请求规则 **

- 接口请求方式为Post，请求数据格式为 Json 方式，响应数据格式统一为：json 。

请求URL：

- `http://xx.com/order/querycard`

Content-Type：

- application/json; charset=utf-8

请求参数：

参数名	必选	类型	是否参与签名	说明
access_key	是	string	是	商户凭证
orderid	否	string	是	订单号（避免因为精度问题导致尾号不正确） orderid与out_trade_no必须二选一
out_trade_no	否	string	是	客户流水号 orderid与out_trade_no必须二选一
timestamp	是	long	是	时间戳
sign	是	string	否	签名，采用HmacSHA1，见签名算法章节

请求示例

- 以下示例：accessSecret 为 `zHCp7cEUy0u55LMFGYEWuA==`

```
{
    "out_trade_no":"20180101010",
    "orderid":"",
    "access_key":"211394650",
    "sign":"DNjuyj6Qj+tk0nrgMGjpKFKF1vI=",
    "timestamp":"1590030050868"
}
```

签名算法代码示例：

返回参数说明

参数名	类型	说明
code	int	订单状态；0:未知 1:充值中 2:订单成功 3:订单失败 4:查询请求失败 5:订单不存在（1分钟后查询有效且需要保证自己所传参数准确无误）
msg	string	描述信息
data	map	数据内容

返回参数内容（data）

参数名	类型	说明
out_trade_no	string	客户流水号
orderid	long	订单号（避免因为精度问题导致尾号不正确）
sellprice	decimal	销售金额
phone	string	充值账号
expire	string	卡密有效期，并非每一笔卡密都会返回效期，是否存在着效期取决于上游及官方，详情咨询商务人员
No	string	卡号，并非每一种产品都会返回卡号，比如爱奇艺，就有可能不存在着卡号，详情咨询商务人员
pwd	string	激活码，具体格式请咨询商务人员

返回示例


```
{
    "code":2,
    "msg":"充值成功",
    "data":{
        "out_trade_no":"20180101010",
        "orderid":200904161848544000,
        "sellprice":19.8,
        "phone":null,
        "expire":"2020-09-04 16:19:14",
        "No":"YMDPRs/7FWpAWg19mDRyAs7qnjQ85pGGUePrCsFXRPJlLU/XbHPDBYb54YmSx2S
        "pwd":"Iiyw20xhEKgUunTnh/R8wH8DT/p8vms003/cvC4PuXpLZpsjgTLUpLyquJDpf0
    }
}
```


Java代码：

```
private static void queryCard() throws Exception {
        java.util.Map<String, String> paras = new java.util.HashMap<String, S
        paras.put("access_key", "211394650");
        paras.put("orderid", "");
        paras.put("out_trade_no", "20180101010");
        paras.put("timestamp", "1590030050868");
        java.util.TreeMap<String, String> sortParas = new java.util.TreeMap<S
        sortParas.putAll(paras);
        java.util.Iterator<String> it = sortParas.keySet().iterator();
        StringBuilder sortQueryStringTmp = new StringBuilder();
        while (it.hasNext()) {
            String key = it.next();
            sortQueryStringTmp.append(key).append(paras.get(key));
        }
        String sign =sign("zHCp7cEUy0u55LMFGYEWuA==" , sortQueryStringTmp.toS
        System.out.println(sign);
        paras.put("sign", sign);
        //Content-Type: application/json; charset=utf-8
        String result=  HttpClientUtil.post("http://123.56.253.5:8070/order/c
        System.out.println(result);
        String pwd =JSON.parseObject(result).getJSONObject("data").get("pwd")
        System.out.println( "激活码，解密后明文："+RSAApplication.prikeydecrip
        String No =JSON.parseObject(result).getJSONObject("data").get("No").t
        System.out.println( "激活码，解密后明文："+RSAApplication.prikeydecrip

    }
```

十一、卡密回调

简要描述：

- 当平台有最终充值结果后，会通过此接口异步返回结果给会员（下游）。

- 参数名称、值对大小写敏感

请求规则

- 接口默认请求方式为Post，请求数据格式为 Json 方式，响应数据格式统一为：json。

请求URL：

- 回调地址为下单接口中的“notify_url”参数。

Content-Type：

- application/json; charset=utf-8

请求参数：

参数名	类型	是否参与签名	说明
access_key	string	是	商户凭证
orderid	long	是	订单号 orderid与out_trade_no必须二选一
out_trade_no	string	是	客户流水号 orderid与out_trade_no必须二选一
timestamp	long	是	时间戳 （1970-01-01至当前时间的总毫秒数）
code	int	是	订单状态；2:订单成功 3:订单失败，注：若返回其它值，请与我方平台核实后再作处理
sign	string	否	签名，采用HmacSHA1，见签名算法章节
data	map	否	数据内容

请求参数内容（data）

参数名	类型	说明
sellprice	decimal	销售金额
phone	string	充值账号
pass	string	透传参数，与收单中的透传参数一致
expire	string	卡密有效期，并非每一笔卡密都会返回效期，是否存在着效期取决于上游及官方，详情咨询商务人员
No	string	卡号，并非每一种产品都会返回卡号，比如爱奇艺，就有可能不存在着卡号，详情咨询商务人员
pwd	string	激活码，具体格式请咨询商务人员

请求示例

- 以下示例：accessSecret 为 `zHCp7cEUy0u55LMFGYEWuA==`


```
{
    "access_key":"211394650",
    "orderid":200904203510437000,
    "out_trade_no":"20180101015",
    "timestamp":1599222911324,
    "code":2,
    "sign":"I4PQgDp7oSiG6DjfyqcWtDtvPCc=",
    "data":{
        "sellprice":19.8,
        "phone":null,
        "pass":null,
        "No":"IBdpjoHIcSmH/xgWdGUp8qrZX1ddMXgHSXeQOLLnw5VoD5qzoAXb+g7myrH33s
        "pwd":"bFo5NA65IUN8G1Eh4l5gsyy6PQMLZL+VHKeLbkg2QigvG3RG3WnyGOO/PWaPX
        "expire":"2020-09-04 20:35:11"
    }
}
```

返回参数说明

参数名称	类型	说明
code	int	0:接收成功

返回示例：

```
{"code":0}
```

回调备注：

- 会员（下游）成功接收到通知，需要返回code为0的数据，我方平台判定为会员已经接收成功。
- 会员已成功接收通知，我方平台会停止当前订单的自动回调操作。
- 我方平台发起回调操作失败后（用户接口异常或未按指定格式返回数据），会根据订单回调策略进行再次回调。
- 订单回调策略：

  > *回调最大次数为6次，第一次回调失败（请求响应超时、返回码指定数据、其他异常等）后，服务系统将依次间隔10秒，1分钟，10分钟，1小时，10小时进行推送，推送成功或推送6次都失败都将结束推送任务。*

部署解密代码：

```
String str="{\"access_key\":\"211394650\",\"orderid\":200904203510437000,\'
      CallBackParamCard param= JSON.parseObject(str,CallBackParamCard.class
      String  pwd= (String) param.getData().get("pwd");
     System.out.println( RSAApplication.prikeydecript(RSAApplication.prikey
```