

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 `8A2UCK1KznjAJ1rUkndmBNVvkM0=`。

• ** Java算法Demo: **

```
1 public static void main(){
2     java.util.Map<String, String> paras = new java.util.HashMap<String, String>();
3     paras.put("accessKey", "abcdes");
4     paras.put("timestamp", "1590030050868");
5
6     java.util.TreeMap<String, String> sortParas = new java.util.TreeMap<String, String>();
7     sortParas.putAll(paras);
8
9     java.util.Iterator<String> it = sortParas.keySet().iterator();
10
11     StringBuilder sortQueryStringTmp = new StringBuilder();
12     while (it.hasNext()) {
13         String key = it.next();
14         sortQueryStringTmp.append(key).append(paras.get(key));
15     }
16
17     String sign = sign("4A09289E1E99", sortQueryStringTmp.toString());
18 }
19
20 public static String sign(String accessSecret, String stringToSign) throws Exception {
21     javax.crypto.Mac mac = javax.crypto.Mac.getInstance("HmacSHA1");
22     mac.init(new javax.crypto.spec.SecretKeySpec(accessSecret.getBytes("UTF-8"), "HmacSHA1"));
23     byte[] signData = mac.doFinal(stringToSign.getBytes("UTF-8"));
24     return new sun.misc.BASE64Encoder().encode(signData);
25 }
```

- C#算法Demo:

```
1 public static void Main(){
2     SortedList<string, string> ts = new SortedList<string, string>();
3     ts.Add("accessKey", "abcdes");
4     ts.Add("timestamp", "1590030050868");
5     StringBuilder sb = new StringBuilder();
6     foreach (var item in ts)
7     {
8         sb.Append(item.Key).Append(item.Value);
9     }
10    string si= HmacSha1("4A09289E1E99", sb.ToString());
11 }
12
13 public static string HmacSha1(string key, string input)
14 {
15     byte[] keyBytes = Encoding.UTF8.GetBytes(key);
16     byte[] inputBytes = Encoding.UTF8.GetBytes(input);
17     System.Security.Cryptography.HMACSHA1 hmac = new System.Security.Cryptography.HMACSHA1(keyBytes);
18     byte[] hashBytes = hmac.ComputeHash(inputBytes);
19     return Convert.ToBase64String(hashBytes);
20 }
```

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- naming	string	中石油油卡需要，持卡人姓名。
id- card	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

```
1 {
2     "access_key": "abcdes",
3     "code": "P0001",
4     "notify_url": "http://localhost:9000/order/callback",
5     "out_trade_no": "20180101001",
```

```

6      "par_value":"10",
7      "phone":"13224011030",
8      "producttype":"1",
9      "sign":"EFXW302NpTByrDUBF8mEwqwTOp0=",
10     "timestamp":"1590030050868"
11 }

```

签名算法代码示例：

```

java.util.Map<String, String> paras = new java.util.HashMap<String, String>(); paras.size() = 7
paras.put("phone", "13224011030");
paras.put("access_key", "abcde");
paras.put("par_value", "10");
paras.put("code", "P0001");
paras.put("producttype", "1");
paras.put("out_trade_no", "20180101001");
paras.put("timestamp", "1590030050868");
java.util.TreeMap<String, String> sortParas = new java.util.TreeMap<>(); sortParas.size() = 7
sortParas.putAll(paras);
java.util.Iterator<String> it = sortParas.keySet().iterator(); it: TreeMap$KeyIterator@1869 sortParas.size() = 7
StringBuilder sortQueryString = new StringBuilder(); sortQueryString.append("access_key=abcdeP0001out_trade_no=20180101001par_value=10phone=13224011030producttype=1timestamp=1590030050868")
while (it.hasNext()) {
    String key = it.next(); it: TreeMap$KeyIterator@1869
    sortQueryString.append(key).append(paras.get(key)); paras.size() = 7
}
String sign = sign(accessSecret: "4A0928921899", sortQueryString.toString()); sign: "EFXW302NpTByrDUBF8mEwqwTOp0="
sortQueryString.append("sign=" + sign); sortQueryString.append("sign=" + sign);
System.out.println(sign); sign: "EFXW302NpTByrDUBF8mEwqwTOp0="

```

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

```

1 {
2     "code": 1,
3     "msg": "收单成功",
4     "data": {
5         "orderid": 200521230215315000,

```

```
6         "sellprice": 0.06
7     }
8 }
```

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：

```
1 private static void insert() throws Exception {
2     java.util.Map<String, Object> paras = new java.util.HashMap<String, Object>();
3     paras.put("phone", "13224011030");
4     paras.put("access_key", "211394650");
5     paras.put("par_value", 10);
6     paras.put("code", "SKU100001");
7     paras.put("producttype", 1);
8     paras.put("out_trade_no", "20180101001");
9     paras.put("timestamp", Long.parseLong("1590030050868"));
10    java.util.TreeMap<String, Object> sortParas = new java.util.TreeMap<String, Object>();
11    sortParas.putAll(paras);
12    java.util.Iterator<String> it = sortParas.keySet().iterator();
13    StringBuilder sortQueryStringTmp = new StringBuilder();
14    while (it.hasNext()) {
15        String key = it.next();
16        sortQueryStringTmp.append(key).append(paras.get(key));
17    }
18    String sign =sign("zHCp7cEUy0u55LMFGYEWuA==" , sortQueryStringTmp.toString());
19    paras.put("sign", sign);
20    paras.put("notify_url", "http://127.0.0.1");
21    paras.put("pass", "");
}
```

```

22     paras.put("extra", new HashMap<String,Object>());
23     System.out.println(sign);
24     //Content-Type: application/json; charset=utf-8
25     String result= HttpClientUtil.post("http://123.56.253.5:8081/order/i
26     System.out.println(result);
27 }

```

```

private static void insert() throws Exception {
    java.util.Map<String, Object> paras = new java.util.HashMap<String, Object>(); paras: size = 11
    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("1590830050868"));
    java.util.TreeMap<String, Object> sortParas = new java.util.TreeMap<String, Object>(); sortParas: size = 7
    sortParas.putAll(paras);
    java.util.Iterator<String> it = sortParas.keySet().iterator(); it: TreeMap$KeyIterator@1790 sortParas: size = 7
    StringBuilder sortQueryStringTmp = new StringBuilder(); sortQueryStringTmp: "access_key211394650codeSKU100001out_trade_no20180101001par_value10phone13224011030producttype1timestamp1590830050868"
    while (it.hasNext()) {
        String key = it.next(); it: TreeMap$KeyIterator@1790
        sortQueryStringTmp.append(key).append(paras.get(key));
    }
    String sign = sign(accessSecret: "zHCp7cBUy0u5SLMPCYEWuA==", sortQueryStringTmp.toString()); sign: "bFqdsWacChqS/SFvb3uMdvITjg=" sortQueryStringTmp: "access_key211394650codeSKU100001out_trade_no20180101001par_val
    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: "bFqdsWacChqS/SFvb3uMdvITjg="
    //Content-Type: application/json; charset=utf-8
    String result= HttpClientUtil.post(url: "http://121.89.161.176:8081/order/insert", JSONObject.toJSONString(paras), encoding: "UTF-8"); result: "{\"code\":2,\"msg\":\"订单已存在\",\"data\":null}" paras: size = 11
    System.out.println(result); result: "{\"code\":2,\"msg\":\"订单已存在\",\"data\":null}"
}

```

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

```
1 {
2     "access_key":"abcdes",
3     "orderid":"",
4     "out_trade_no":"20180101001",
5     "sign":"4E7Ku7G0TqxdcTQ7fgue+7Vqlxw=",
6     "timestamp":"1590030050868"
7 }
```

签名算法代码示例：

```
java.util.Map<String, String> paras = new java.util.HashMap<String, String>(); paras.size() = 4
paras.put("access_key", "abcedes");
paras.put("orderid", "");
paras.put("out_trade_no", "20180101001");
paras.put("timestamp", "1590030050868");
java.util.TreeMap<String, String> sortParas = new java.util.TreeMap<>(); sortParas.size() = 4
sortParas.putAll(paras);
java.util.Iterator<String> it = sortParas.keySet().iterator(); it: TreeMap$KeyIterator@1869 sortParas.size() = 4
StringBuilder sortQueryStringTmp = new StringBuilder(); sortQueryStringTmp: "access_keyabcedesorderid_trade_no20180101001timestamp1590030050868"
while (it.hasNext()) {
    String key = it.next(); it: TreeMap$KeyIterator@1869
    sortQueryStringTmp.append(key).append(paras.get(key)); paras.size() = 4
}
String sign = sign(accessSecret: "4A0928981E99", sortQueryStringTmp.toString()); sign: "4B7Eu7G0TqdcTQ7fgue+VqLcw=" sortQueryStringTmp: "access_keyabcedesorderid_trade_no20180101001timestamp1590030050868"
System.out.println(sign); sign: "4B7Eu7G0TqdcTQ7fgue+VqLcw="
```

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

```
1 {
2     "code": 1,
3     "msg": "充值中",
4     "data": {
5         "out_trade_no": "20180101001",
6         "orderid": 200521230215315000,
7         "sellprice": 0.06,
8         "phone": "13224011030",
9         "official_id": "982000165965526",
10        "official_ch": "其它"
11    }
12 }
```

Java代码:

```
1 private static void query() throws Exception {
2     java.util.Map<String, String> paras = new java.util.HashMap<String, String>();
3     paras.put("access_key", "211394650");
4     paras.put("orderid", "");
5     paras.put("out_trade_no", "20180101001");
6     paras.put("timestamp", "1590030050868");
7     java.util.TreeMap<String, String> sortParas = new java.util.TreeMap<String, String>();
8     sortParas.putAll(paras);
9     java.util.Iterator<String> it = sortParas.keySet().iterator();
10    StringBuilder sortQueryStringTmp = new StringBuilder();
11    while (it.hasNext()) {
12        String key = it.next();
13        sortQueryStringTmp.append(key).append(paras.get(key));
14    }
15    String sign = sign("zHCp7cEUy0u55LMFGYEWuA==" , sortQueryStringTmp.toString());
16    System.out.println(sign);
17    paras.put("sign", sign);
18    //Content-Type: application/json; charset=utf-8
19    String result= HttpClientUtil.post("http://123.56.253.5:8071/order/create");
20    System.out.println(result);
21
22 }
```

```
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$KeyIterator@1790 sortParas.size = 4
    StringBuilder sortQueryStringTmp = new StringBuilder(); sortQueryStringTmp: "access_key211394650orderid_trade_no20180101001timestamp1590030050868"
    while (it.hasNext()) {
        String key = it.next(); it: TreeMap$KeyIterator@1790
        sortQueryStringTmp.append(key).append(paras.get(key));
    }
    String sign = sign(accessSecret: "sHCp7cWY0u55LNFGY8WuA==", sortQueryStringTmp.toString()); sign: "P3UUEGw7tyTh6j/EEJNLL15Fcum" sortQueryStringTmp: "access_key211394650orderid_trade_no20180101001timestamp1590030050868"
    System.out.println(sign);
    paras.put("sign", sign); sign: "P3UUEGw7tyTh6j/EEJNLL15Fcum"
    //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\":\"msg\":\"充值失败\",\"data\":{\"out_trade_no\":\"20180101001\",\"orderid\":1209534003936037000,\"sellprice\":\"10\",\"phone\":\"18224011030\",\"official_id\":\"null\",\"official_ch\":\"null\"}}\"
    System.out.println(result);
}
```

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

```

1 {
2     "access_key":"211394650",
3     "code":2,
4     "data":{
5         "sellprice":10,
6         "phone":"13224011031",
7         "pass":null,
8         "official_id":null,
9         "official_ch":null
10    },
11    "orderid":200528105302018000,
12    "out_trade_no":"20200528105258",
13    "sign":"7M3x1eICIQSuaCHujev17dIFKCo=",
14    "timestamp":1590634458337
15 }
```

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

```
1 {
2     "access_key": "abcdes",
3     "sign": "lNdSBH5EP2uBEHyNV4j1BmfTJ14=",
4     "timestamp": "1590030050868"
5 }
```

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

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

```

3      "msg": "请求成功",
4      "balance": -150
5  }

```

Java代码:

```

1 private static void account() throws Exception {
2     java.util.Map<String, String> paras = new java.util.HashMap<String, String>();
3     paras.put("access_key", "211394650");
4     paras.put("timestamp", "1590030050868");
5     java.util.TreeMap<String, String> sortParas = new java.util.TreeMap<String, String>();
6     sortParas.putAll(paras);
7     java.util.Iterator<String> it = sortParas.keySet().iterator();
8     StringBuilder sortQueryStringTmp = new StringBuilder();
9     while (it.hasNext()) {
10        String key = it.next();
11        sortQueryStringTmp.append(key).append(paras.get(key));
12    }
13    String sign = sign("zHCp7cEUy0u55LMFGYEWuA==" , sortQueryStringTmp.toString());
14    paras.put("sign", sign);
15    System.out.println(sign);
16    //Content-Type: application/json; charset=utf-8
17    String result= HttpClientUtil.post("http://123.56.253.5:8071/member/account", paras);
18    System.out.println(result);
19 }

```

```

private static void account() throws Exception {
    java.util.Map<String, String> paras = new java.util.HashMap<>(); paras: size = 2
    paras.put("access_key", "211394650");
    paras.put("timestamp", "1590030050868");
    java.util.TreeMap<String, String> sortParas = new java.util.TreeMap<>(); sortParas: size = 2
    sortParas.putAll(paras);
    java.util.Iterator<String> it = sortParas.keySet().iterator(); it: TreeMap$KeyIterator@1790 sortParas: size = 2
    StringBuilder sortQueryStringTmp = new StringBuilder(); sortQueryStringTmp: "access_key211394650timestamp1590030050868"
    while (it.hasNext()) {
        String key = it.next(); it: TreeMap$KeyIterator@1790
        sortQueryStringTmp.append(key).append(paras.get(key));
    }
    String sign = sign("accessSecret: zHCp7cEUy0u55LMFGYEWuA==", sortQueryStringTmp.toString()); sign: "qT7BbPbLEHNG61aJcZe8P2hF1cZk=" sortQueryStringTmp: "access_key211394650timestamp1590030050868"
    paras.put("sign", sign);
    System.out.println(sign); sign: "qT7BbPbLEHNG61aJcZe8P2hF1cZk="
    //Content-Type: application/json; charset=utf-8
    String result= HttpClientUtil.post("url: http://121.89.161.176:8071/member/account", JSonObject.toJSonString(paras), encoding: "UTF-8"); result: "{code:1,msg:请求成功,balance:-1140} paras: size = 2
    System.out.println(result); result: "{code:1,msg:请求成功,balance:-1140}"
}

```

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。

请直接使用各接口中给出代码示例测试，不要过度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己代码找茬上，直接使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==`

```
1 {    "public_key": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCCciQPzzM1lKD6RlP58c
2    "code": "SKU100001",
3    "out_trade_no": "20180101001",
4    "phone": "13224011030",
5    "pass": "",
6    "access_key": "211394650",
7    "extra": {
8    },
9    "sign": "QXLprzhx5ro5QGNkrXVv+2dBagk=",
10   "notify_url": "http://127.0.0.1",
11   "timestamp": 1590030050868
12 }
```

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("1590030050868"));
paras.put("public_key", "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCCciQPzzM1lKD6RlP58dwp/P1lt9zYh/8Xk3IJuL40+MYWY18P8WfBP+ncYmTELX7rv19Wz8Bbi0SxJncJgY0PB4879YzT28bXWz1G0P2D14imCvDxrw4xCwpoLYRlNzHobaB1V");
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$KeyIterator@1103 sortParas.size() = 6
StringBuilder sortQueryStringTmp = new StringBuilder(); sortQueryStringTmp: "access_key211394650codeSKU100001out_trade_no20180101001phone13224011030public_keyMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCCciQPzzM1lKD6RlP58dwp/P1lt9zYh/8Xk3IJuL40+MYWY18P8WfBP+ncYmTELX7rv19Wz8Bbi0SxJncJgY0PB4879YzT28bXWz1G0P2D14imCvDxrw4xCwpoLYRlNzHobaB1V"
while (it.hasNext()) {
    String key = it.next(); it: TreeMap$KeyIterator@1103
    sortQueryStringTmp.append(key).append(paras.get(key));
}
String sign = sign(accessSecret: "zHCp7cEUy0u55LMFGYEWuA==", sortQueryStringTmp.toString()); sign: "QXLprzhx5ro5QGNkrXVv+2dBagk="
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

```
1 {
2     "code": 1,
3     "msg": "收单成功",
4     "data": {
5         "orderid": 200521230215315000,
6         "sellprice": 0.06
7     }
8 }
```

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：

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

```

8      paras.put("public_key", "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCcciQF
9      java.util.TreeMap<String, Object> sortParas = new java.util.TreeMap<S
10     sortParas.putAll(paras);
11     java.util.Iterator<String> it = sortParas.keySet().iterator();
12     StringBuilder sortQueryStringTmp = new StringBuilder();
13     while (it.hasNext()) {
14         String key = it.next();
15         sortQueryStringTmp.append(key).append(paras.get(key));
16     }
17     String sign =sign("zHCp7cEUy0u55LMFGYEWuA==" , sortQueryStringTmp.toS
18     paras.put("sign", sign);
19     paras.put("notify_url", "http://123.56.253.5:8070/order/callback");
20     paras.put("pass", "");
21     HashMap<String,Object> et=new HashMap<String,Object>();
22 //      et.put("aa", "bcd");
23     paras.put("extra", et);
24     System.out.println(sign);
25     //Content-Type: application/json; charset=utf-8
26     System.out.println(JSONObject.toJSONString(paras));
27     String result= HttpClientUtil.post("http://123.56.253.5:8070/order/i
28     System.out.println(result);
29 }

```

```

private static void insertCard() throws Exception {
    java.util.Map<String, Object> paras = new java.util.HashMap<>(); paras.size = 10
    paras.put("phone", "");
    paras.put("access_key", "211394650");
    paras.put("code", "SKU100063");
    paras.put("out_trade_no", "201801010002");
    paras.put("timestamp", Long.parseLong("1590030050868"));
    paras.put("public_key", "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCcciQPaW11ED6R1P58dwp/F11t9xYh/EXk3IJuL40e+MYWYI8P8HfPa+mcTymTELX7+v19Wz8Bi0SxJaoJgY0P84879YzT2BbXNz1G0P3D14imCvDzxw4xCwp0LYRIMzHobab1YH
    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$KeyIterator@1790 sortParas.size = 6
    StringBuilder sortQueryStringTmp = new StringBuilder(); sortQueryStringTmp: "access_key211394650codeSKU100063out_trade_no20180101002phonepublic_keyMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCcciQPaW11ED6R1P58dwp/F11t9xYh/EXk3IJuL40e+MYW
    while (it.hasNext()) {
        String key = it.next(); it: TreeMap$KeyIterator@1790
        sortQueryStringTmp.append(key).append(paras.get(key));
    }
    String sign =sign("accessSecret: "zHCp7cEUy0u55LMFGYEWuA==" , sortQueryStringTmp.toString()); sign: "uYhN2t5vISjQUv3om67R87dJL0=" sortQueryStringTmp: "access_key211394650codeSKU100063out_trade_no20180101002phonepublic_keyMIGfMA0G
    paras.put("sign", sign);
    paras.put("notify_url", "http://39.99.131.191:8070/order/callback");
    paras.put("pass", "");
    HashMap<String,Object> et=new HashMap<String,Object>(); et.size = 0
    et.put("aa", "bcd");
    paras.put("extra", et); et.size = 0
    System.out.println(sign); sign: "uYhN2t5vISjQUv3om67R87dJL0="
    //Content-Type: application/json; charset=utf-8
    System.out.println(JSONObject.toJSONString(paras));
    String result= HttpClientUtil.post("url: http://39.99.131.191:8070/order/insertcard", JSONObject.toJSONString(paras), encoding: "UTF-8"); result: {"code":1,"msg":"收单成功","data":{"order_id":"2009041095735900","sellprice":1
    System.out.println(result); result: {"code":1,"msg":"收单成功","data":{"order_id":"2009041095735900","sellprice":1
}

```

九、卡密加解密RSA算法

Java:

• Demo:

```

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

```

```

3      System.out.println( "加密前: "+ sour );
4      String str=  publicKeyEncrypt(sour,pkey);
5      System.out.println( "公钥加密后: "+ str );
6      str= prikeydecrypt(prikey,str);
7      System.out.println( "私钥解密后: "+ str );
8  }
9  static String pkey="MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCcciQPzzM1lKD6R1
10 static String prikey="MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBAJxy
11
12 public static String prikeydecrypt(String prikey, String str) throws E
13     String res= RSATools.decrypt(RSAApplication.prikey,str);
14     return res;
15 }
16
17 public static String publicKeyEncrypt(String sour, String pkey) throws E
18     String str= RSATools.encrypt(RSAApplication.pkey,sour);
19     return str;
20 }

```

• RSATools

```

1 package com.xxx.testappc;
2
3 import org.apache.commons.lang3.ArrayUtils;
4 import sun.misc.BASE64Decoder;
5 import sun.misc.BASE64Encoder;
6 import javax.crypto.BadPaddingException;
7 import javax.crypto.Cipher;
8 import javax.crypto.IllegalBlockSizeException;
9 import javax.crypto.NoSuchPaddingException;
10 import java.math.BigInteger;
11 import java.security.*;
12 import java.security.interfaces.RSAPrivateCrtKey;
13 import java.security.interfaces.RSAPrivateKey;
14 import java.security.interfaces.RSAPublicKey;
15 import java.security.spec.InvalidKeySpecException;
16 import java.security.spec.PKCS8EncodedKeySpec;
17 import java.security.spec.X509EncodedKeySpec;
18
19 public class RSATools {
20
21     /**

```

```
22     * 公钥加密过程
23     *
24     * @param publicKey
25     *         公钥
26     * @param plainTextData
27     *         明文数据
28     * @return
29     * @throws Exception
30     *         加密过程中的异常信息
31     */
32     public static byte[] encrypt(RSAPublicKey publicKey, byte[] plainTextDat
33         if (publicKey == null) {
34             throw new Exception("加密公钥为空, 请设置");
35         }
36         Cipher cipher = null;
37         try {
38             // 使用默认RSA
39             cipher = Cipher.getInstance("RSA");
40             cipher.init(Cipher.ENCRYPT_MODE, publicKey);
41             byte[] output = cipher.doFinal(plainTextData);
42             return output;
43         } catch (NoSuchAlgorithmException e) {
44             throw new Exception("无此加密算法");
45         } catch (NoSuchPaddingException e) {
46             e.printStackTrace();
47             return null;
48         } catch (InvalidKeyException e) {
49             throw new Exception("加密公钥非法, 请检查");
50         } catch (IllegalBlockSizeException e) {
51             e.printStackTrace();
52             throw new Exception("明文长度非法");
53         } catch (BadPaddingException e) {
54             throw new Exception("明文数据已损坏");
55         }
56     }
57
58     /**
59     * 公钥加密过程
60     *
61     * @param
62     *
63     * @param plainTextData
64     *         明文数据
```

```

65     * @return
66     * @throws Exception
67     *         加密过程中的异常信息
68     */
69     public static String encrypt(String pubKeyString, String plainTextData)
70     {
71         RSAPublicKey publicKey = loadPublicKeyByStr(pubKeyString);
72         byte[] encData = encrypt(publicKey, plainTextData.getBytes("utf-8"));
73         return new BASE64Encoder().encode(encData);
74     }
75     /**
76     * 从字符串中加载公钥
77     *
78     * @param publicKeyStr
79     *         公钥数据字符串
80     * @throws Exception
81     *         加载公钥时产生的异常
82     */
83     public static RSAPublicKey loadPublicKeyByStr(String publicKeyStr) throw
84     try {
85         byte[] buffer = new BASE64Decoder().decodeBuffer(publicKeyStr);
86         KeyFactory keyFactory = KeyFactory.getInstance("RSA");
87         X509EncodedKeySpec keySpec = new X509EncodedKeySpec(buffer);
88         return (RSAPublicKey) keyFactory.generatePublic(keySpec);
89     } catch (NoSuchAlgorithmException e) {
90         throw new Exception("无此算法");
91     } catch (InvalidKeySpecException e) {
92         throw new Exception("公钥非法");
93     } catch (NullPointerException e) {
94         throw new Exception("公钥数据为空");
95     }
96 }
97
98     public static String decrypt(String priKeyString, String cipherText) thr
99     {
100         RSAPrivateKey privateKey = loadPrivateKeyByStr(priKeyString);
101         byte[] cipherData = new BASE64Decoder().decodeBuffer(cipherText);
102         byte[] decDdatas = decrypt(privateKey, cipherData);
103         return new String(decDdatas);
104     }
105     public static RSAPrivateKey loadPrivateKeyByStr(String privateKeyStr) th
106     try {
107         byte[] buffer = new BASE64Decoder().decodeBuffer(privateKeyStr);
108         PKCS8EncodedKeySpec keySpec = new PKCS8EncodedKeySpec(buffer);

```

```

108         KeyFactory keyFactory = KeyFactory.getInstance("RSA");
109         return (RSAPrivateKey) keyFactory.generatePrivate(keySpec);
110     } catch (NoSuchAlgorithmException e) {
111         throw new Exception("无此算法");
112     } catch (InvalidKeySpecException e) {
113         throw new Exception("私钥非法");
114     } catch (NullPointerException e) {
115         throw new Exception("私钥数据为空");
116     }
117 }
118 /**
119  * 私钥解密过程
120  *
121  * @param privateKey
122  *      私钥
123  * @param cipherData
124  *      密文数据
125  * @return 明文
126  * @throws Exception
127  *      解密过程中的异常信息
128  */
129 public static byte[] decrypt(RSAPrivateKey privateKey, byte[] cipherData) {
130     if (privateKey == null) {
131         throw new Exception("解密私钥为空，请设置");
132     }
133     Cipher cipher = null;
134     try {
135         // 使用默认RSA
136         cipher = Cipher.getInstance("RSA");
137         cipher.init(Cipher.DECRYPT_MODE, privateKey);
138         byte[] output = cipher.doFinal(cipherData); //解密
139         return output;
140     } catch (NoSuchAlgorithmException e) {
141         throw new Exception("无此解密算法");
142     } catch (NoSuchPaddingException e) {
143         e.printStackTrace();
144         return null;
145     } catch (InvalidKeyException e) {
146         throw new Exception("解密私钥非法,请检查");
147     } catch (IllegalBlockSizeException e) {
148         throw new Exception("密文长度非法");
149     } catch (BadPaddingException e) {
150         throw new Exception("密文数据已损坏");

```

```

151     }
152 }
153
154 }
155

```

c#:

```

1  using Org.BouncyCastle.Crypto;
2  using Org.BouncyCastle.Crypto.Parameters;
3  using Org.BouncyCastle.Security;
4  using System;
5  using System.Collections.Generic;
6  using System.IO;
7  using System.Linq;
8  using System.Net;
9  using System.Security.Cryptography;
10 using System.Text;
11 using System.Threading;
12 using System.Threading.Tasks;
13
14 private static void RSADemo()
15 {
16     String sour = @"123456789SABDFDFDEFFK";
17     string prikey = "MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGE
18     string pkey = "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCCciQPzzM1lk
19
20     Console.WriteLine("加密前明文:" + sour);
21     //公钥加密
22     byte[] t_b = System.Text.Encoding.UTF8.GetBytes(sour);
23     BouncyCastleHelper bbe = new BouncyCastleHelper();
24     AsymmetricKeyParameter publicKey = bbe.GetPublicKey(Convert.FromBase64String(pkey));
25     IAsymmetricBlockCipher enginee = new Org.BouncyCastle.Crypto.Encoders.Base64Encoder();
26     enginee.Init(true, publicKey);
27     var tem = enginee.ProcessBlock(t_b, 0, t_b.Length);
28     String t_e = Convert.ToBase64String(tem);
29     Console.WriteLine("加密后密文:" + t_e);
30     //私钥解密:
31     byte[] bytesToDecrypt = Convert.FromBase64String(t_e);
32     BouncyCastleHelper bbd = new BouncyCastleHelper();
33     var engined = new Org.BouncyCastle.Crypto.Encodings.Pkcs1Encoding

```

```
34         engined.Init(false, bbd.GetPrivateKey(Convert.FromBase64String(pr
35         var returnData = engined.ProcessBlock(bytesToDecrypt, 0, bytesToD
36         string t_d = Encoding.UTF8.GetString(returnData);
37         Console.WriteLine("解密后明文: "+ t_d);
38         Console.ReadKey();
39     }
```

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==`

```
1 {
2     "out_trade_no":"20180101010",
3     "orderid":"",
4     "access_key":"211394650",
5     "sign":"DNjuyj6Qj+tk0nrgMGjpKFKF1vI=",
6     "timestamp":"1590030050868"
7 }
```

签名算法代码示例：

```
java.util.Map<String, String> paras = new java.util.HashMap<String, String>(); paras: size = 4
paras.put("access_key", "abcdes");
paras.put("orderid", "");
paras.put("out_trade_no", "20180101001");
paras.put("timestamp", "1590030050868");
java.util.TreeMap<String, String> sortParas = new java.util.TreeMap<>(); sortParas: size = 4
sortParas.putAll(paras);
java.util.Iterator<String> it = sortParas.keySet().iterator(); it: TreeMap$KeyIterator@1869 sortParas: size = 4
StringBuilder sortQueryStringTmp = new StringBuilder(); sortQueryStringTmp: "access_keyabcdesorderid_trade_no20180101001timestamp1590030050868"
while (it.hasNext()) {
    String key = it.next(); it: TreeMap$KeyIterator@1869
    sortQueryStringTmp.append(key).append(paras.get(key)); paras: size = 4
}
String sign = sign(accessSecret: "4A0928981E99", sortQueryStringTmp.toString()); sign: "4B7Eu7G0TqdcTQ7iguc+VqLw=" sortQueryStringTmp: "access_keyabcdesorderid_trade_no20180101001timestamp1590030050868"
System.out.println(sign); sign: "4B7Eu7G0TqdcTQ7iguc+VqLw="
```

返回参数说明

参 数 名	类 型	说 明
c o d e	in t	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）
m s g	st ri n g	描述信息
d a t a	m a p	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

```

1 {
2     "code":2,
3     "msg":"充值成功",
4     "data":{
5         "out_trade_no":"20180101010",
6         "orderid":200904161848544000,
7         "sellprice":19.8,
8         "phone":null,
9         "expire":"2020-09-04 16:19:14",
10        "No":"YMDPRs/7FWpAwg19mDRyAs7qnjQ85pGGUePrCsFXRPJ1LU/XbHPDBYb54YmSx2S
11        "pwd":"Iiyw20xhEKgUunTnh/R8wH8DT/p8vms003/cvC4PuXpLZpsjgTLUpLyquJDpf6
12    }
13 }
```

Java代码:

```

1 private static void queryCard() throws Exception {
2     java.util.Map<String, String> paras = new java.util.HashMap<String, String>();
3     paras.put("access_key", "211394650");
4     paras.put("orderid", "");
5     paras.put("out_trade_no", "20180101010");
6     paras.put("timestamp", "1590030050868");
7     java.util.TreeMap<String, String> sortParas = new java.util.TreeMap<String, String>();
8     sortParas.putAll(paras);
9     java.util.Iterator<String> it = sortParas.keySet().iterator();
10    StringBuilder sortQueryStringTmp = new StringBuilder();
11    while (it.hasNext()) {
12        String key = it.next();
13        sortQueryStringTmp.append(key).append(paras.get(key));
14    }
15    String sign =sign("zHCp7cEUy0u55LMFGYEWuA==" , sortQueryStringTmp.toString());
16    System.out.println(sign);
17    paras.put("sign", sign);
18    //Content-Type: application/json; charset=utf-8
19    String result= HttpClientUtil.post("http://123.56.253.5:8070/order/querycard", paras);
20    System.out.println(result);
21    String pwd =JSON.parseObject(result).getJSONObject("data").get("pwd");
22    System.out.println( "激活码，解密后明文: "+RSAApplication.prikeydecrypt(pwd));
23    String No =JSON.parseObject(result).getJSONObject("data").get("No");
24    System.out.println( "激活码，解密后明文: "+RSAApplication.prikeydecrypt(No));
25
26 }

```

```

private static void queryCard() throws Exception {
    Map<String, String> paras = new HashMap<>();
    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<>();
    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.toString());
    System.out.println(sign);
    paras.put("sign", sign);
    //Content-Type: application/json; charset=utf-8
    String result= HttpClientUtil.post("http://39.99.131.191:8070/order/querycard", JSON.toJSONString(paras), encoding: "UTF-8");
    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.prikeydecrypt(RSAApplication.prikey(pwd)));
    String No =JSON.parseObject(result).getJSONObject("data").get("No");
    System.out.println( "激活码，解密后明文: "+RSAApplication.prikeydecrypt(RSAApplication.prikey(No)));
}

```

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
sell price	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==`

```

1 {
2     "access_key":"211394650",
3     "orderid":200904203510437000,
4     "out_trade_no":"20180101015",
5     "timestamp":1599222911324,
6     "code":2,
7     "sign":"I4PQgDp7oSiG6DjfyqcWtDtvPCc=",
8     "data":{
9         "sellprice":19.8,
10        "phone":null,
11        "pass":null,
12        "No":"IBdpjoHIcSmH/xgWdGUp8qrZX1ddMXgHSXeQ0LLnw5VoD5qzoAXb+g7myrH33sJ
13        "pwd":"bFo5NA65IUN8G1Eh4l5gsyy6PQMLZL+VHKeLbkg2QigvG3RG3WnyG00/PWaPX2
14        "expire":"2020-09-04 20:35:11"
15    }
16 }
```

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：

```
1 String str="{\"access_key\": \"211394650\", \"orderid\": 200904203510437000, \"  
2     CallbackParamCard param= JSON.parseObject(str, CallbackParamCard.class)  
3     String pwd= (String) param.getData().get(\"pwd\");  
4     System.out.println( RSAApplication.prikeydecrypt(RSAApplication.prikey
```