

---

title: 一卡通v5第三方消费接口文档V1.2 date: 2019-05-06 tags:

---

## 1.1 文档说明

本文档用于描述了一卡通对第三方应用提供的通用支付api接口文档，供第三方系统（前置和应用）对接参考。

## 1.2 约定

1. 传输编码统一为utf-8
2. RSA 加密算法为本接口采用的非对称加密算法，`SIGN_ALGORITHMS ="SHA1WithRSA"`;
3. RSA公钥由本系统提供，本系统返回信息的签名数据，第三方系统得到后通过公钥校验算法，验证返回数据的合法性
4. HMAC加密算法为用户请求本系统采用的加密算法，`SIGN_ALGORITHMS ="HMAC-SHA1"`;
5. 具体签名生成参考后面的签名章节
6. retcode等于0表示成功，非0表示失败，失败具体信息查看retmsg
7. 密码加密的原始串为timestamp:pwd

## 2.1 通用账户查询

请求URL:

- `https://ip:port/epayapi/services/thirdparty/common/accountquery`

请求方式/格式:

- POST
- application/x-www-form-urlencoded

请求参数:

参数名	类型	必选	说明
partner_id	String	是	本系统分配给各个接入应用的合作伙伴id号
cardphyid	String	是	查询对象的卡物理id
timestamp	String	是	时间戳格式为yyyyMMddhh24miss
sign	String	是	签名
sign_method	String	是	参数的加密方法选择，可选值是：HMAC 加密方式为HMAC-SHA1

请求内容示例:

```
URI: https://ip:port/epayapi/services/thirdparty/common/accountquery

partner_id:10000
stuempno:09893092
timestamp:20150119130901
```

```
sign:5195f9b9116e4adf67eeebc9935d33dc683f677d
sign_method:HMAC
```

#### 返回示例:

- 成功

```
{
  "retcode": "0",
  "retmsg": "query success",
  "onceTimeLimit": "5000",
  "dayTotalLimit": "10000",
  "expiredate": "20191010",
  "balance": 4850,
  "frozenbal": 0,
  "status": "normal"
}
```

- 失败

```
{
  "retcode": "1",
  "retmsg": "account not exist"
}
```

#### 返回参数说明:

参数名	类型	说明
retcode	String	返回码 (0=成功, 其他为失败)
retmsg	String	返回消息
balance	Integer	余额(分)
frozenbal	Integer	冻结余额(分)
expiredate	String	卡有效期
status	String	状态(正常)
onceTimeLimit	String	单次限额 (分)
dayTotalLimit	String	日累计限额 (分)

## 2.2 通用支付

#### 请求URL:

- <https://ip:port/epayapi/services/thirdparty/common/pay>

#### 请求方式/格式:

- POST
- application/x-www-form-urlencoded

#### 请求参数:

参数名	类型	必选	说明
partner_id	String	是	本系统分配给各个接入应用的合作伙伴id号
cardphyid	String	是	查询对象的卡物理id
tradeno	String(32)	是	第三方系统唯一流水号
tradenname	String(60)	是	交易名称
amount	Integer	是	消费金额（分）
timestamp	String	是	时间戳格式为yyyyMMddhh24miss
sign	String	是	签名
sign_method	String	是	参数的加密方法选择，可选值是：HMAC 加密方式为HMAC-SHA1

#### 请求内容示例:

```
URI: https://ip:port/epayapi/services/thirdparty/common/pay

partner_id:10000
stuempno:09893092
tradeno:20160607000001
tradenname:print fee
amount: 2000
timestamp:20150119130901
sign:5195f9b9116e4adf67eeebc9935d33dc683f677d
sign_method:HMAC
```

#### 返回示例:

- 成功

```
{
    "retcode":"0",
    "retmsg":"success",
    "tradeno":"20160607000001",
    "balance":4850,
    "refno":"20160605190200000001"
}
```

- 失败

```
{
    "retcode": "1",
    "retmsg": "账户余额不足",
    "tradeno": "20160607000001"
}
```

#### 返回参数说明:

参数名	类型	说明
retcode	String	返回码 (0=成功, 其他为失败)
retmsg	String	返回消息
tradeno	String	第三方流水号
refno	String	本系统生成流水号
balance	Integer	余额(分)

## 附录A-用户请求HMAC签名算法

- 签名方式: **hmac-sha1**
- 签名密钥由本系统统一线下提供
- 签名校验的通用步骤如下:

\*\*第一步,\*\*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。

假设传送的参数如下:

```
partner_id:10000
stuempno:09893092
tradeno:20160607000001
trandename:printfee
amount: 2000
timestamp:20150119130901
sign:5195f9b9116e4adf67eeebc9935d33dc683f677d
sign_method:HMAC
```

对参数按照key=value的格式, 并按照参数名ASCII字典序排序如下:

```
amount=2000&partner_id=10000&sign_method=HMAC&stuempno=09893092
&timestamp=20150119130901&tradenno=20160607000001&trandename=printf
ee
```

**\*\*特别注意以下重要规则：\*\***

- 参数名ASCII码从小到大排序（字典序）；
- 如果参数的值为空不参与签名；
- 参数名区分大小写；
- 传送的sign参数不参与签名，用该sign值作校验。

**\*\*第二步，\*\***用密钥secretkey对stringA字符串，进行hmac-sha1签名，得到sign值signValue。signValue最后采用十六进制小写hex编码生成签名字符串。

## 附录B-服务端返回数据RSA签名校验算法

- 签名方式：SHA1withRSA
- 签名校验的公钥key为本系统统一线下提供。
- 签名校验的通用步骤如下：

**\*\*第一步，\*\***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。

**特别注意以下重要规则：**

- 参数名ASCII码从小到大排序（字典序）；
- 如果参数的值为空不参与签名；
- 参数名区分大小写；
- 传送的sign参数不参与签名，用该sign值作校验。

**\*\*第二步，\*\***对sign值进行base64解码，用本系统提供的公钥key对sign签名值解码后的数据基于stringA字符串，进行SHA1withRSA签名验证

举例：

假设传送的参数如下：

```
retcode:1
retmsg:账户余额不足
timestamp:20160513155100
sign_mehtod:RSA
```

对参数按照key=value的格式，并按照参数名ASCII字典序排序如下：

```
retcode=1&retmsg=账户余额不足&timestamp=20160513155100&sign_mehtod=RSA
```