接口说明

- 调用方法
- 报文结构
- 签名算法
- 数据加密算法
- 密码加密算法
- 密钥使用流程

调用方法

内容	说明
通讯方式	采用 HTTPS 方式访问
提交方法	POST (application/json)
交互数据格式	JSON
字符编码	UTF-8
签名算法	MD5
签名验证	提交和返回数据都需要验证签名

报文结构

请求报文和返回报文都采用如下的报文结构

序号	参数	类型	参考数据	参数说明	备注
1	action	String	inquiry	交易类型	具体接口有详细说明
2	deviceNo	String(8)	POS01	收银机编号	用于区别本次交易请求的收银机
3	shopNo	String(15)	CN123456	门店编号	用于区别本次交易请求的门店
4	brand	String	663	品牌编号	用于区别本次交易请求的品牌，易百分配
5	body	String	BASE64	报文内容	具体接口有详细说明，本域需要采用BAS
6	mwVersion	String	20161010	中间件版本号	用于兼容不同的中间件对应后台不同的处理
7	ptlVersion	String	20161010	协议版本号	用于兼容不同的报文协议对应后台不同的处
8	posVersion	String	20161010	易百POS版本号	用于兼容不同的易百POS对应后台不同
9	timestamp	String	1483372334	Unix时间戳	单位：秒，中间件需要在收到返回时同步该
10	sign	String	7E65B60DCFA42B04	数据签名	详见：签名算法

报文举例

```
 {
    "action" : "inquiry",
    "deviceNo" : "POS01",
    "shopNo" : "CN123456",
    "brand" : "663",
    "body" :
"ewogICAgICAgICAidHJhY2VObyI6ICI5OTAwMDAwOTEwMDAxMDEwMTczMjEyMyIsCiAgICAgICAgICJvcmlnaW5hbFR
yYWNlTm8iOiAiOTkwMDAwMDkxMDAwMTAxMDE3MzIxMjQiCiAgICAgfQ==",
    "mwVersion" : "20161010",
    "ptlVersion" : "20161010",
    "posVersion" : "20161010",
    "timestamp" : "1483372334",
    "sign" : "7E65B60DCFA42B04"
 }
```

签名算法

签名过程

- 拼接报文参数
  除sign参数外的所有参数按照ASCII顺序排序后，以"参数名1=参数值1&参数名2=参数值2"的方式拼接所有参数
- 拼接签名 mwMacKey
  最后在尾部拼接双方约定的密钥"KEY=xxxxx"，
  因调用downloadKey接口前，并未获得mwMacKey，故使用指定KEY签名
  因调用signin接口前，并未获得mwMacKey，故使用mwTmk签名
- 签名值计算
  MD5-32加密后再转成HEX大写格式，即为签名值，并赋给sign参数。
- 注
  当参数值为null或为空时不参与签名串拼接

示例报文

```
 {
    "action" : "inquiry",
    "deviceNo" : "POS01",
    "shopNo" : "CN123456",
    "brand" : "663",
    "body" :
"ewogICAgICAgICAidHJhY2VObyI6ICI5OTAwMDAwOTEwMDAxMDEwMTczMjEyMyIsCiAgICAgICAgICJvcmlnaW5hbFRy
yYWNlTm8iOiAiOTkwMDAwMDkxMDAwMTAxMDE3MzIxMjQiCiAgICAgfQ==",
    "mwVersion" : "20161010",
    "ptlVersion" : "20161010",
    "posVersion" : "20161010",
    "timestamp" : "1483372334",
    "sign" : "F38545F4D74B5C10A9EBBC053ED9D1CF"
 }
```

签名**mwMacKey**密钥

```
94365019BBF9CEEAB0DF658E67754A70
```

拼接报文参数

```
action=inquiry&body=ewogICAgICAgICAidHJhY2VObyI6ICI5OTAwMDAwOTEwMDAxMDEwMTczMjEyMyIsCiAgICAgICAgICAgICJvcmlnaW5hbFRyYWNlTm8iOiAiOTkwMDAwMDkxMDAwMTAxMDE3MzIxMjQiCiAgICAgfQ==&brand=663&deviceNo=POS01&mwVersion=20161010&posVersion=20161010&ptlVersion=20161010&shopNo=CN123456&timestamp=1483372334
```

□□□KEY

```
action=inquiry&body=ewogICAgICAgICAidHJhY2VObyI6ICI5OTAwMDAwOTEwMDAxMDEwMTczMjEyMyIsCiAgICAgICAgICAgICJvcmlnaW5hbFRyYWNlTm8iOiAiOTkwMDAwMDkxMDAwMTAxMDE3MzIxMjQiCiAgICAgfQ==&brand=663&deviceNo=POS01&mwVersion=20161010&posVersion=20161010&ptlVersion=20161010&shopNo=CN123456&timestamp=1483372334&KEY=94365019BBF9CEEAB0DF658E67754A70
```

□□□□□

F38545F4D74B5C10A9EBBC053ED9D1CF

Java□□□□

```
Map<String, String> map = new TreeMap<>();
map.put("action", "downloadKey");
map.put("deviceNo", "CN999999");
map.put("shopNo", "CN999999");
map.put("brand", "1458");
map.put("body", "JXU2RDRCJXU4QkQ1JXU1MTg1JXU1QkI5JTIwJXU2RDRCJXU4QkQ1JXU1MTg1JXU1QkI5");
map.put("mwVersion", "20170214");
map.put("ptlVersion", "20170214");
map.put("posVersion", "20170214");
map.put("timestamp", "1483372334");

StringBuilder buffer = new StringBuilder();
for (Map.Entry<String, String> item : map.entrySet()) {
    buffer.append(item.getKey()).append("=").append(item.getValue()).append("&");
}
buffer.append("KEY=").append("F42616614BDC0000161EF06C04061484");
String checkSign = DESCoder.getInstance().encryptMD5(buffer.toString(), "UTF-8").toUpperCase();
System.out.println(buffer.toString());
System.out.println(checkSign);

□□:
```

action=downloadKey&body=JXU2RDRCJXU4QkQ1JXU1MTg1JXU1QkI5JTIwJXU2RDRCJXU4QkQ1JXU1MTg1JXU1QkI5&brand=1458&deviceNo=CN999999&mwVersion=20170214&posVersion=20170214&ptlVersion=20170214&shopNo=CN999999&timestamp=1483372334&KEY=F42616614BDC0000161EF06C04061484

824AE098F6135CF50A824BAE220379C6

数据加密算法

3DES介绍

- 3DES（或称为Triple DES）是三重数据加密算法（TDEA，Triple Data Encryption Algorithm）块密码的通称。它相当于是对每个数据块应用三次DE

加解密过程

- 设Ek()和Dk()代表DES算法的加密和解密过程，K代表DES算法使用的密钥，M代表明文，C代表密文，这样：
- 3DES加密过程为：C=Ek3(Dk2(Ek1(M)))
- 3DES解密过程为：M=Dk1(EK2(Dk3(C)))
- K1、K2、K3决定了算法的安全性，若三个密钥互不相同，本质上就相当于用一个长为168位的密钥进行加密。多年来，它在对付强力攻击时是比较安全的。若数据对安全性要求不那么高，K1可以等于K3。在这种情况下，密钥的有效长度为112位。

加密举例

设pinKey `9D93D15D6A3913AB4151C456A80841EF`，则：

```
K1 = 9D93D15D6A3913AB
K2 = 4151C456A80841EF
K3 = 9D93D15D6A3913AB
```

原始数据M `3132333435363738`，加密如下：

```
DES1_RESULT = Ek(M, K1)
DES2_RESULT = Dk(DES1_RESULT, K2)
C = Ek(DES2_RESULT, K3)
```

最 `C63AABF759BDE968`

解密举例

设pinKey `9D93D15D6A3913AB4151C456A80841EF`，则：

```
K1 = 9D93D15D6A3913AB
K2 = 4151C456A80841EF
K3 = 9D93D15D6A3913AB
```

密文数据C `C63AABF759BDE968`，解密如下：

DES1_RESULT = Dk(C□K3)

DES2_RESULT = Ek(DES1_RESULT□K2)

M = Dk(DES2_RESULT□K1)

□□ 3132333435363738

密码加密算法

ANSI X9.8 Format（带主帐号信息）

- PIN BLOCK 格式：等于 PIN 按位异或主帐号
- PIN 格式：（个人识别码 Personal Identity Number）总共有8个byte长度，分为两个部分；（类似数据包的格式）

  > Byte 1 PIN的长度
  >
  > Byte 2 – Byte 3/4/5/6/7   4--12个PIN(每个PIN占4个BIT)
  >
  > Byte4/5/6/7/8 – Byte 8 FILLER “F” (每个“F“占4个BIT)

- PAN（主帐号 Primary Account Number）同样包含8个byte，格式如下：

  > Byte 1 — Byte 2 0x00 0x00
  >
  > Byte 3 — Byte 8 12个主帐号字符（最后一位为校验位）
  >
  > 12位字符主帐号的取法：取主帐号的右12位（不包括最右边的校验位），不足12位左补“0X00”。

加密举例

- 明文 P `123456`
- 主帐号 P `1234567890123456 78`
- `678901234567` （最后一位校验位8的前12位字符为截取的主帐号）
- 则用于 `0x00 0x00 0x67 0x89 0x01 0x23 0x45 0x67`
- 则 PIN BLOCK （PIN按位异或主帐号PAN）

  > 即是为： 0x06 0x12 0x34 0x56 0xFF 0xFF 0xFF 0xFF
  >
  > 异或上： 0x00 0x00 0x67 0x89 0x01 0x23 0x45 0x67
  >
  > 结果为： 0x06 0x12 0x53 0xDF 0xFE 0xDC 0xBA 0x98

密钥使用流程

一般调用流程：

- 当做 DownloadKey 时，使用 rootKey 来生成签名，返回报文中得到中间件主密钥 mwTmk 的密文，使用 rootKey 对该密文做3DES解密，得到 mwTm
- 当做 SignIn 时，使用上一步中获得的 mwTmk 明文来生成签名，返回报文中得到中间件Mac秘钥(工作秘钥) mwMacKey 的密文，使用 mwTmk 明文对此密文做3
- 当做交易时，使用上一步中获得的 mwMacKey 明文来生成签名。