API Introduction

Invoke Method

Content	Description
Protocal	HTTP/HTTPS
HTTP Method	POST (application/json)
Message Type	JSON
Charset	UTF-8
Signature Algorithm	MD5
Signature Verify	Request and Response are both required

Message Structure

Request and response message are both use following structure

No.	Parameter	Format	Reference	Description	Memo
1	action	String	inquiry	Transaction Type	The specific interfaces with detail description
2	deviceNo	String	POS01	Cashier Terminal No.	To identify the Cashier Terminal used in this transaction
3	shopNo	String	CN123456	Shop No.	To identify the merchant / shop in this transaction
4	brand	String	663	Brand No.	To identify the Brand used in this transaction, allocated by e-Buy
5	body	String	BASE64	Message Detail	The specific interfaces with detail description
6	mwVersion	String	20161010	Middleware Version	The Middleware is compatible to process with different backend system
7	ptlVersion	String	20161010	Protocal Version	The protocol is compatible to process with different backend system
8	posVersion	String	20161010	e-Buy POS Version	The e-Buy POS is compatible to process with different backend system
9	timestamp	String	1483372334	Unix Format	Unit : second□□□□□□
10	sign	String	7E65B60DCFA42B04	Signature	See detail : Signature Algorithm

example


```
{
    "action" : "inquiry",
    "deviceNo" : "POS01",
    "shopNo" : "CN123456",
    "brand" : "663",
    "body" :
"ewogICAgICAgICAidHJhY2VObyI6ICI5OTAwMDAwOTEwMDAxMDEwMTczMjEyMyIsCiAgICAgICAgICJvcmlnaW5hbFRyYWNlTm8iOiAiOTkwMDAwMDkxMDAwMTAxMDE3MzIxMjQiCiAgICAgfQ==",
    "mwVersion" : "20161010",
    "ptlVersion" : "20161010",
    "posVersion" : "20161010",
    "timestamp" : "1483372334",
    "sign" : "7E65B60DCFA42B04"
}
```

Signature process

- Combine message parameter All parameter must be by ASCII sequential order, except sign and body. Take the “ parameter number 1 = parameter value 1, parameter number 2 = parameter value 2” to connect all parameters.
- Combine mwMacKey Sign in KEY will place to the last parameter with agreed key “KEY=xxxxx”. Before downloadKey，we do not have mwMacKey，so we use assigned key when invoke downloadKey. Before signin，we do not have mwMacKey，so we use mwTmk when invoke signin.
- Signature Sign in value calculate per MD5-32 encryption and then converted to HEX in capital letter, named Sign-in value, as sign parameter.
- PS when parameter value is null or blank, the parameter is not required to stated

example


```
{
    "action" : "inquiry",
    "deviceNo" : "POS01",
    "shopNo" : "CN123456",
    "brand" : "663",
    "body" :
"ewogICAgICAgICAidHJhY2VObyI6ICI5OTAwMDAwOTEwMDAxMDEwMTczMjEyMyIsCiAgICAgICAgICJvcmlnaW5hbFRyYWNlTm8iOiAiOTkwMDAwMDkxMDAwMTAxMDE3MzIxMjQiCiAgICAgfQ==",
    "mwVersion" : "20161010",
    "ptlVersion" : "20161010",
    "posVersion" : "20161010",
    "timestamp" : "1483372334",
    "sign" : "F38545F4D74B5C10A9EBBC053ED9D1CF"
}
```


mwMacKey


```
94365019BBF9CEEAB0DF658E67754A70
```


Combine message parameter


```
action=inquiry&body=ewogICAgICAgICAidHJhY2VObyI6ICI5OTAwMDAwOTEwMDAxMDEwMTczMjEyMyIsCiAgICAgICAgICJvcmlnaW5hbFRyYWNlTm8iOiAiOTkwMDAwMDkxMDAwMTAxMDE3MzIxMjQiCiAgICAgfQ==&brand=663&de
```

viceNo=POS01&mwVersion=20161010&posVersion=20161010&ptlVersion=20161010&shopNo=CN123456&timestamp=1483372334

Combine KEY

action=inquiry&body=ewogICAgICAgICAidHJhY2VObyI6ICI5OTAwMDAwOTEwMDAxMDEwMTczMjEyMyIsCiAgICAgICAgICJvcmlnaW5hbFRyYWNlTm8iOiAiOTkwMDAwMDkxMDAwMTAxMDE3MzIxMjQiCiAgICAgfQ==&brand=663&deviceNo=POS01&mwVersion=20161010&posVersion=20161010&ptlVersion=20161010&shopNo=CN123456&timestamp=1483372334&KEY=94365019BBF9CEEAB0DF658E67754A70

Signature F38545F4D74B5C10A9EBBC053ED9D1CF

Encryption Algorithm

3DES Introduction

- In cryptography, Triple DES (3DES), officially the Triple Data Encryption Algorithm (TDEA or Triple DEA), is a symmetric-key block cipher, which applies the Data Encryption Standard (DES) cipher algorithm three times to each data block.
- The original DES cipher's key size of 56 bits was generally sufficient when that algorithm was designed, but the availability of increasing computational power made brute-force attacks feasible. Triple DES provides a relatively simple method of increasing the key size of DES to protect against such attacks, without the need to design a completely new block cipher algorithm.

Algorithm

Triple DES uses a "key bundle" that comprises three DES keys, K1, K2 and K3, each of 56 bits (excluding parity bits). The encryption algorithm is:

ciphertext = EK3(DK2(EK1(plaintext))) I.e., DES encrypt with K1, DES decrypt with K2, then DES encrypt with K3.

Decryption is the reverse:

plaintext = DK1(EK2(DK3(ciphertext))) I.e., decrypt with K3, encrypt with K2, then decrypt with K1.

Each triple encryption encrypts one block of 64 bits of data.

In each case the middle operation is the reverse of the first and last. This improves the strength of the algorithm when using keying option 2, and provides backward compatibility with DES with keying option 3.

Keying options

The standards define three keying options:

Keying option 1 All three keys are independent. Keying option 2 K1 and K2 are independent, and K3 = K1. Keying option 3 All three keys are identical, i.e. K1 = K2 = K3. Keying option 1 is the strongest, with 3 × 56 = 168 independent key bits.

Keying option 2 provides less security, with 2 × 56 = 112 key bits. This option is stronger than simply DES encrypting twice, e.g. with K1 and K2, because it protects against meet-in-the-middle attacks.

Keying option 3 is equivalent to DES, with only 56 key bits. It provides backward compatibility with DES, because the first and second DES operations cancel out. It is no longer recommended by the National Institute of Standards and Technology (NIST),[5] and is not supported by ISO/IEC 18033-3.

Each DES key is nominally stored or transmitted as 8 bytes, each of odd parity,[12] so a key bundle requires 24 bytes for option 1, 16 for option 2, or 8 for option 3.

Encryption example

Set pinKey: `9D93D15D6A3913AB4151C456A80841EF`:

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

Data M in HEX `3132333435363738` （encryption process）

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

Result C `C63AABF759BDE968`

Decryption example

Set pinKey: `9D93D15D6A3913AB4151C456A80841EF`:

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

Data C in HEX `C63AABF759BDE968` （decryption process）

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

Results M `3132333435363738`

Password Encryption

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