Tài liệu tích hợp Exu Charging

1. Mobile Charging

- Hệ thống charging số tiền trực tiếp trên tài khoản điện thoại và xác thực giao dịch bằng OTP
- Mô tả quy trình:
    - **Bước 1:** Khách hàng truy cập các website/wapsite của merchant và nhập số điện thoại để thực hiện giao dịch. Merchant gửi yêu cầu sang Exu để chuyển đến nhà mạng. Nếu số điện thoại hợp lệ (được nhà mạng chấp nhận) Exu sẽ trả thông báo gửi yêu cầu thành công đồng thời Merchant chuyển người dùng sang trang xác thực OTP.
    - **Bước 2:** Khách hàng nhận được mã OTP từ nhà mạng, tiến hành nhập mã xác thực giao dịch OTP trên website/wapsite của merchant. Merchant gửi yêu cầu sang Exu để xác thực mã OTP này và nhận kết quả giao dịch. Nếu giao dịch thành công, merchant tiến hành cung cấp dịch vụ cho người dùng. Nếu giao dịch thất bại, merchant thông báo kết quả đến người dùng theo bảng mã mô tả mà Exu cung cấp.

1.1. Hàm API Exu.vn cung cấp để nhận yêu cầu từ merchant

- Tên hàm API: **process**
- Địa chỉ Webservice: http://exu.vn/ charging.mobile.api.php?wsdl

Danh sách các tham số đầu vào		
Tên tham số	Kiểu dữ liệu	Ghi chú
merchant_id	string	Mã merchant được cổng thanh toán cung cấp cho các merchant
function	string	Tên hàm xử lý. Gồm 2 hàm là : - sendRequest - verifyRequest
params	string	Đây là một mảng các tham số được mã hóa json thành một chuỗi. Chuỗi json được mã hóa bảo mật bằng thuật toánTripldes với Cipher Mode là ECB và Padding là PKCS5/PKCS5. Key dùng để mã hóa là mật khẩu merchant khai báo trên hệ thống cổng thanh toán. Thông tin chi tiết về tham số này được mô tả ở mỗi hàm kết nối

		dưới đây.

Danh sách các tham số đầu ra		
Tên tham số	Kiểu dữ liệu	Ghi chú
result	string	Đây là một chuỗi có cấu trúc: **result_code|result_data** Với **result_code** là mã trả về. Chi tiết xem bảng mã trả về trong mục 3.2 **result_data** là một mảng các tham số trả về được mã hóa json thành một chuỗi. Chuỗi json được mã hóa bảo mật bằng thuật toánTripledes với Cipher Mode là ECB và Padding là PKCS5/PKCS5. Key dùng để mã hóa là mật khẩu merchant khai báo trên hệ thống cổng thanh toán. Thông tin chi tiết về tham số này được mô tả ở mỗi hàm kết nối dưới đây.

1.1.1. Hàm xử lý sendRequest

- Dùng để gửi yêu cầu charging

Mô tả chi tiết tham số params		
Tên tham số	Kiểu dữ liệu	Ghi chú
game_code	string(20)	Mã Game Code merchant đăng ký với Exu.vn
refer_code	string(50)	Mã tham chiếu merchant gửi sang.
mobile	string(20)	Số điện thoại sử dụng để charging
price	int	Số tiền muốn charging. Số tiền phải nằm trong bảng danh sách các mệnh giá charging. Xem chi tiết trong bảng danh sách mệnh giá charging trong mục 3.1
message	string(160)	Nội dung tin nhắn merchant muốn gửi tới người dùng sau khi charging thành công

Mô tả chi tiết tham số result_data		
Tên tham số	Kiểu dữ liệu	Ghi chú
request_id	int	ID yêu cầu được ghi nhận trên

		cổng thanh toán.

1.1.2. Hàm xử lý verifyRequest

- Dùng để xác nhận yêu cầu charging

Mô tả chi tiết tham số params		
Tên tham số	Kiểu dữ liệu	Ghi chú
request_id	int	ID yêu cầu được ghi nhận trên cổng thanh toán (được trả lại sau khi gọi hàm sendRequest)
otp	string(6)	Mã OTP nhà mạng gửi cho người dùng đến số điện thoại yêu cầu charging

Mô tả chi tiết tham số result_data		
Tên tham số	Kiểu dữ liệu	Ghi chú
request_id	int	ID yêu cầu được ghi nhận trên cổng thanh toán
game_code	string(20)	Mã Game Code merchant đăng ký với Exu.vn
telco	string(10)	Mã nhà mạng. Bao gồm các giá trị: - VIETTEL - VMS - VNP
mobile	string(20)	Số điện thoại sử dụng để charging
price	int	Số tiền muốn charging. Số tiền phải nằm trong bảng danh sách các mệnh giá charging. Xem chi tiết trong bảng danh sách mệnh giá charging trong mục 3.1
amount	int	Số tiền merchant thực nhận sau khi trừ đi phí
command_code	string(20)	Mã lệnh Exu.vn gửi sang nhà mạng. Giá trị mặc định là NAP
refer_code	string(50)	Mã tham chiếu merchant gửi sang ở hàm sendRequest

1.2. Hàm API Merchant xây dựng để đón thông báo kết quả giao dịch

1.2.1. Hàm notifyMobileTransaction

- Dùng để đón thông báo kết quả giao dịch thành công với yêu cầu charging gửi sang
- Tên hàm: notifyMobileTransaction

Mô tả chi tiết tham số trả về

Tên tham số	Kiểu dữ liệu	Ghi chú
request_id	int	ID yêu cầu được ghi nhận trên cổng thanh toán
game_code	string(20)	Mã Game Code merchant đăng ký với Exu.vn
telco	string(10)	Mã nhà mạng. Bao gồm các giá trị: - VIETTEL - VMS - VNP
mobile	string(20)	Số điện thoại sử dụng để charging
price	int	Số tiền muốn charging. Số tiền phải nằm trong bảng danh sách các mệnh giá charging. Xem chi tiết trong bảng danh sách mệnh giá charging trong mục 3.1
amount	int	Số tiền merchant thực nhận sau khi trừ đi phí
command_code	string(20)	Mã lệnh Exu.vn gửi sang nhà mạng. Giá trị mặc định là NAP
refer_code	string(50)	Mã tham chiếu merchant gửi sang ở hàm sendRequest

2. Sms Charging

- Hệ thống Charging sử dụng tin nhắn SMS đến đầu số 9029
- Cấu trúc tin nhắn:
    - Với nhà mạng Mobifone:
      PS [game_code] [content_id] [account] [Thông tin khác]

- **game_code** : là mã merchant khai báo trên hệ thống Exu.vn
- **content_id** : mã nội dung trừ tiền
  Mã nội dung bao gồm các thông tin loại giao dịch, giá trị thanh toán và các thông tin khác theo yêu cầu của CP.
  content_id = **command_code** + giá cước
  command_code bao gồm: ACTIVE, NAP, DK
  giá cước: Xem chi tiết trong mục 3.1
  Ví dụ:
    Để active tài khoản chơi game **content_id** như sau: ACTIVE1
    Để nạp tiền vào tài khoản game **content_id** như sau: NAP15
    Để đăng ký thuê bao tài khoản Vip trong game **content_id** như sau: DK15

- **account** : username tài khoản chơi game
- Thông tin khác: các thông tin cần thiết khác tùy theo yêu cầu của merchant

2.1. Hàm API Merchant xây dựng để đón thông báo kết quả giao dịch

2.1.1. Hàm notifySmsTransaction

- Dùng để đón thông báo kết quả giao dịch thành công với yêu cầu sms charging gửi sang
- Tên hàm: notifySmsTransaction

Mô tả chi tiết tham số trả về		
Tên tham số	Kiểu dữ liệu	Ghi chú
request_id	int	ID yêu cầu được ghi nhận trên cổng thanh toán
game_code	string(20)	Mã Game Code merchant đăng ký với Exu.vn
telco	string(10)	Mã nhà mạng. Bao gồm các giá trị: - VIETTEL - VMS - VNP
mobile	string(20)	Số điện thoại sử dụng để charging
price	int	Số tiền muốn charging. Số tiền phải nằm trong bảng danh sách các mệnh giá charging. Xem chi tiết trong bảng danh sách mệnh giá charging trong mục 3.1
amount	int	Số tiền merchant thực nhận sau khi trừ đi phí
command_code	string(20)	Mã lệnh Exu.vn gửi sang nhà mạng. Giá trị mặc định là NAP
sms_message	string(160)	Nội dung tin nhắn người dùng gửi sang nhà mạng

3. Thông tin các bảng mã

3.1. Bảng danh sách mệnh giá charging

Số giá cước	Mệnh giá charging
1	1000
2	2000
3	3000
4	4000
5	5000
10	10000
15	15000
20	20000
30	30000

40	40000
50	50000
100	100000

3.2. Bảng mã trả về

Mã trả về	Mô tả
00	Thành công
01	Lỗi không xác định
02	Merchant không tồn tại hoặc bị khóa
03	Tham số đầu vào mã hóa không đúng hoặc sai mật khẩu kết nối
04	Hàm xử lý không tồn tại
05	Thiếu tham số đầu vào
06	game_code không tồn tại hoặc đang bị khóa
07	refer_code không hợp lệ
08	mobile không hợp lệ
09	price không hợp lệ
10	mobile không được hỗ trợ
11	Số tiền giao dịch bị từ chối bởi telco (số tiền quá lớn hoặc quá giới hạn giao dịch trong ngày)
12	message không hợp lệ
13	request_id không hợp lệ
14	otp không hợp lệ
15	request_id đã hết hạn
16	request_id chưa xác nhận giao dịch
17	request_id đang xác nhận giao dịch
20	MO không hợp lệ
21	Command code không hợp lệ
30	Lỗi kết nối đến telco
31	Lỗi trên hệ thống telco cần đối soát
32	Lỗi không gửi được OTP xác nhận sang nhà mạng tiền chưa bị trừ
33	Hệ thống telco quá tải tiền chưa bị trừ
34	Tài khoản không đủ để thực hiện trừ tiền
35	Nhập sai mã OTP
36	Lỗi gửi tin nhắn báo trừ tiền
37	OTP timeout
50	Lỗi khi nhận yêu cầu
52	Lỗi khi xử lý yêu cầu
60	Lỗi kết nối đến merchant

61	Gọi merchant lỗi

4. Các đoạn code mã hóa Tripledes mẫu

4.1. Ngôn ngữ .NET C#

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Text;
public static string Encrypt(string key, string data)
{
        data = data.Trim();
        byte[] keydata = Encoding.ASCII.GetBytes(key);
        string md5String = BitConverter.ToString(new
        MD5CryptoServiceProvider().ComputeHash(keydata)).Replace("-", "").ToLower();
        byte[] tripleDesKey = Encoding.ASCII.GetBytes(md5String.Substring(0, 24));
        TripleDES tripdes = TripleDESCryptoServiceProvider.Create();
        tripdes.Mode = CipherMode.ECB;
        tripdes.Padding = PaddingMode.PKCS7;
        tripdes.Key = tripleDesKey;
        tripdes.GenerateIV();
        MemoryStream ms = new MemoryStream();
        CryptoStream encStream = new CryptoStream(ms, tripdes.CreateEncryptor(),
        CryptoStreamMode.Write);
        encStream.Write(Encoding.ASCII.GetBytes(data), 0,
        Encoding.ASCII.GetByteCount(data));
        encStream.FlushFinalBlock();
        byte[] cryptoByte = ms.ToArray();
        ms.Close();
        encStream.Close();
        return Convert.ToBase64String(cryptoByte, 0, cryptoByte.GetLength(0)).Trim();
}

public static string Decrypt(string key, string dataen)
{
        byte[] toEncryptArray = Convert.FromBase64String(dataen);
        byte[] keydata = Encoding.ASCII.GetBytes(key);
        string md5String = BitConverter.ToString(new
        MD5CryptoServiceProvider().ComputeHash(keydata)).Replace("-", "").ToLower();
        byte[] tripleDesKey = Encoding.ASCII.GetBytes(md5String.Substring(0, 24));
        TripleDES tripdes = TripleDESCryptoServiceProvider.Create();
        tripdes.Mode = CipherMode.ECB;
        tripdes.Padding = PaddingMode.PKCS7;
        tripdes.Key = tripleDesKey;
        tripdes.GenerateIV();
        ICryptoTransform ict = tripdes.CreateDecryptor();
        byte[] resultArray = ict.TransformFinalBlock(toEncryptArray, 0, toEncryptArray.Length);
        tripdes.Clear();
```

```
        return Encoding.ASCII.GetString(resultArray);
    }
```

4.2. Ngôn ngữ Java

```
public static String getMD5(String sMessage)
{
        byte[] defaultBytes = sMessage.getBytes();
        try {
                MessageDigest algorithm = MessageDigest.getInstance("MD5");
                algorithm.reset();
                algorithm.update(defaultBytes);
                byte messageDigest[] = algorithm.digest();
                StringBuffer hexString = new StringBuffer();
                for (int i = 0; i < messageDigest.length; i++) {
                        String hex = Integer.toHexString(0xFF & messageDigest[i]);
                        if (hex.length() == 1) {
                                hexString.append('0');
                        }
                        hexString.append(hex);
                 }
                return hexString.toString();
        } catch (NoSuchAlgorithmException nsae) {
                return null;
        }
}

public static String Encrypt(String key,String data) throws Exception
{
        Cipher cipher=Cipher.getInstance("TripleDES");
        String keymd5 =getMD5(key).substring(0,24);
        SecretKeySpec keyspec = new SecretKeySpec(keymd5.getBytes(),"TripleDES");
        cipher.init(Cipher.ENCRYPT_MODE,keyspec);
        byte[] stringBytes=data.getBytes();
        byte[] raw=cipher.doFinal(stringBytes);
        BASE64Encoder encoder = new BASE64Encoder();
        String base64 = encoder.encode(raw);
        return base64;
}

public static String Decrypt(String key,String data) throws Exception
 {
        Cipher cipher=Cipher.getInstance("TripleDES");
        String keymd5 =getMD5(key).substring(0,24);
        SecretKeySpec keyspec = new
        SecretKeySpec(keymd5.getBytes(),"TripleDES");
        cipher.init(Cipher.DECRYPT_MODE,keyspec);
        BASE64Decoder decoder = new BASE64Decoder();
        byte[] raw = decoder.decodeBuffer(data);
        byte[] stringBytes = cipher.doFinal(raw);
```

```
        String result = new String(stringBytes);
        return result;
}
```

4.3. Ngôn ngữ PHP

```
function Encrypt($input, $key_seed)
{
        $input = trim($input);
        $block = mcrypt_get_block_size('tripledes', 'ecb');
        $len = strlen($input);
        $padding = $block - ($len % $block);
        $input .= str_repeat(chr($padding),$padding);
        // generate a 24 byte key from the md5 of the seed
        $key = substr(md5($key_seed),0,24);
        $iv_size = mcrypt_get_iv_size(MCRYPT_TRIPLEDES, MCRYPT_MODE_ECB);
        $iv = mcrypt_create_iv($iv_size, MCRYPT_RAND);
        // encrypt
        $encrypted_data = mcrypt_encrypt(MCRYPT_TRIPLEDES, $key, $input,
        MCRYPT_MODE_ECB, $iv);
        // clean up output and return base64 encoded
        return base64_encode($encrypted_data);
}

function Decrypt($input, $key_seed)
{
        $input = base64_decode($input);
        $key = substr(md5($key_seed),0,24);
        $text=mcrypt_decrypt(MCRYPT_TRIPLEDES, $key, $input,
        MCRYPT_MODE_ECB,'12345678');
        $block = mcrypt_get_block_size('tripledes', 'ecb');
        $packing = ord($text{strlen($text) - 1});
        if($packing and ($packing < $block)){
                for($P = strlen($text) - 1; $P >= strlen($text) - $packing; $P--){
                        if(ord($text{$P}) != $packing){
                                $packing = 0;
                        }
                }
        }
        $text = substr($text,0,strlen($text) - $packing);
        return $text;
}
```