eNETS Merchant Integration Guide
eNETS Open API

Version 1.07
# Contents

1. Introduction ........................................................................................................................................... 4  
1.1. Pre-production .................................................................................................................................... 4  
1.2. Hardware & Software Compatibility ............................................................................................... 5  
1.3. Contact Point ...................................................................................................................................... 5  
1.4. Terms & Conditions ........................................................................................................................... 5  
2. Getting Started ...................................................................................................................................... 7  
3. Key Concept .......................................................................................................................................... 8  
4. API Installation & Setup ....................................................................................................................... 10  
4.1. Obtain Test and Production Keys .................................................................................................... 10  
4.2. Mobile SDK integration setup ........................................................................................................... 10  
5. Four Types of Transaction Flows ........................................................................................................ 11  
5.1. Transaction Flow 1 Web Browser with Credit Card (3D & Non-3D) or eDebit (IDD) ..................... 11  
5.2. Transaction Flow 2 Merchant Mobile Native App with NETS Mobile SDK .................................... 24  
5.3. Transaction Flow 3 Web Browser with QR code ............................................................................... 36  
5.4. Transaction Flow 4 eCredit Server-To-Server (S2S) ...................................................................... 46  
6. Transaction Query ............................................................................................................................... 57  
6.1. Description ......................................................................................................................................... 57  
6.2. Message Format ................................................................................................................................. 60  
7. Recurring Payment .............................................................................................................................. 62  
8. IPP Payment .......................................................................................................................................... 64  
9. Card Prefix ............................................................................................................................................ 66  
Appendix A – Definitions .......................................................................................................................... 67  
Appendix B – Payment Channels ............................................................................................................. 69  
Appendix C – Source of Funds versus Transaction Flows ...................................................................... 70  
Appendix D – Message Format ................................................................................................................. 71  
Appendix E – Response & Action Codes .................................................................................................. 84  
Appendix F – Android Setup .................................................................................................................... 94  
Appendix G – iOS Setup .......................................................................................................................... 100  
Appendix H – FAQ & Exception Handling ............................................................................................ 104
1 Introduction

It’s easy to integrate your apps with eNETS using the eNETS open API. Just follow the simple step-by-step instructions.

Before you start receiving your payments, these are a few things that need to be done:

- **SSL Certificates (if you use HTTPS)**
- **UAT Certificates**
  - CA Root Certificates
  - Intermediate Certificates/Chain Certificates (if any)
- **Production Certificates**
  - CA Root Certificates
  - Intermediate Certificates

You must share the certificates at least 3 weeks before implementing any changes.

1.1. Pre-production

- **eNETS UAT Environment**
  1. Certificates loaded
  2. You must perform at least one round of testing to make sure the transaction flow runs smoothly
  3. The CA Root/Intermediate UAT and Production Certificates should be identical
  4. Successful testing in UAT environment

- **eNETS Production Environment**
1.2. Hardware & Software Compatibility

1.2.1. Using eNETS mobile SDK
Supported Operating System versions:
  • Android 18 or later
  • iOS 9.0 or later

1.2.2. Using eNETS Web Browser
Supported Web Browsers:
  • The current and previous version of Chrome
  • The current and previous version of Firefox
  • Internet explorer 8.0 or later

1.3. Contact Point

Please feel free to address any comments or feedback that you may have on eNETS Merchant Integration guide by:

Post:
eNETS Pte Ltd
298 Tiong Bahru Road
#04-01/06 Central Plaza
Singapore 168730

Email:
info@nets.com.sg

Hotline:
(65) 6274 1212

Operating Hours:
8:30am to 6:00pm Mondays - Fridays.
Closed on Saturdays, Sundays and Public Holidays

1.4. Terms & Conditions

• eNETS reserves the right to reject loading the certificates without getting prior approval from merchants if the CA Root Certificates are suspected or found to be causing abnormalities to the eNETS system. In such scenarios, eNETS will inform the merchant before or after removal, depending on the situation.

• eNETS reserves the rights to remove the Merchant’s CA Root Certificates at any point of time without getting prior approval if the CA Root Certificates are suspected or found to be causing abnormalities to the eNETS system. In such scenarios, eNETS will inform the merchant before or after removal, depending on the situation.

• All the points mentioned above are also applicable for the subsequent
change or renewal of Merchant’s SSL certificates after the integration.
2 Getting Started

Integrating with the new eNETS payment gateway is fast and fun, be it from your mobile app or website. An ever-growing list of payment options makes sure that you’re integrating with a robust and comprehensive payment gateway.

eNETS provides RESTful API where request and response messages use JSON & HTML.

Merchants with website payment

\[\text{eNETS Javascript Plugin}\]

Merchants with native mobile app

\[\text{Client SDK}\]

With the SDK and the Javascript plugin, you can integrate effortlessly without having to worry about security and 3D handling.

Note: Please prompt the payer to enable the Javascript in the browser.
3 Key Concept

No matter what’s the chosen payment option, the following 4 Integration Methods with their transaction flows have it covered:

<table>
<thead>
<tr>
<th>Integration Method 1</th>
<th>Integration Method 2</th>
<th>Integration Method 3</th>
<th>Integration Method 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Browser Flow</td>
<td>Merchant Mobile App Flow</td>
<td>QR Related Flow</td>
<td>Server to Server Flow</td>
</tr>
<tr>
<td>Transaction Flow 1</td>
<td>Transaction Flow 2</td>
<td>Transaction Flow 3</td>
<td>Transaction Flow 4</td>
</tr>
<tr>
<td>(Refer to section 5.1)</td>
<td>(Refer to section 5.2)</td>
<td>(Refer to section 5.3)</td>
<td>(Refer to section 5.4)</td>
</tr>
</tbody>
</table>

For web portals that user access from PC or Mobile via the browser.
For native mobile applications.
For QR code wallets and transactions.
For direct capturing of Credit Card data by Merchant.

The following are the applicable usage for the above flows:

- For those who want to offer eDebit: Refer to 5.1 Transaction Flow 1 Web Browser Flow
- For those who want to offer eCredit: Refer to 5.1 Transaction Flow 1 Web Browser Flow
- For those who want to offer NETSPay, DBS PayLah, OCBC PayAnyOne, UOB Mighty: Refer to 5.2 Transaction Flow 2 Merchant Mobile App Flow
- For those who want to Capture Credit Card yourself: Refer to 5.4 Transaction Flow 4 Server to Server Flow
  (Source Of Fund-SOF supported: Credit 3D, Credit Non-3D)
Please refer to Appendix C for the four transaction flows and their various supported source of funds.
4 API Installation & Setup

4.1 Obtain Test and Production Keys

Please follow the instruction provided in NETS email to download the API KeyId and Secret Key.

- Test – API keyId and Secret Key.
- Production - API keyId and Secret Key.

Both Test and Production MIDs will be sent to you via the same email.

4.2 Mobile SDK integration setup

- **Android:** Refer to Appendix F
- **iOS:** Refer to Appendix G
5 Four Types of Transaction Flows

5.1 Transaction Flow 1
Web Browser with Credit Card (3D & Non-3D) or eDebit (IDD)

The diagram below shows a transaction flow whereby a payer makes a purchase on the merchant site via the browser.

This flow has primarily 2 parts (upon payer checkout):
- In Step 2, merchant sends a transaction request to eNETS Gateway through the browser.
- In Step 4 and 6, merchant receives s2sTxnEnd (Step 4) and b2sTxnEnd (Step 6)

Merchant only needs to wait for either s2sTxnEnd or b2sTxnEnd to confirm whether the transaction response is approved. eNETS will ensure both b2sTxnEnd and s2sTxnEnd have the same value.

Please refer to Appendix A for the definition of s2sTxnEnd and b2sTxnEnd.
5.1.1 Description

The sequence diagram below describes the payer making payment at the Merchant Portal via a Web Browser. In this transaction, the Merchant will be using the Browser-To-Server (B2S) (See Appendix A for definition of B2S) flow. Relevant steps for merchants (2, 4, 6, and 7) are explained in greater details below.

This B2S transaction flow supports eCredit, eDebit (IDD) and uPOP.
Step 2
The following is required:
(a) Create the Transaction Request (txnReq) message
(b) Generate the MAC value of txnReq and
(c) Embed the downloaded keyId, MAC value and txnReq into the HTML Page
and send it to the browser. Please refer to section 4 for the download of secretKey.

(a) txnReq message
The txnReq message is a JSON string. Merchants shall refer to section 5.1.2 to form the actual txnReq message. Below is an example of txnReq message.

```
{"ss":"1","msg":{"netsMid":"UMID_887770001","tid":"","submissionMode":"B","txnAmount":"1000","merchantTxnRef":"20170605 10:26:51.98","merchantTxnDtm":"20170605 10:26:51.989","paymentType":"SALE","currencyCode":"SGD","paymentMode":"","merchantTimeZone":"+8:00","b2sTxnEndURL":"https://sit2.enets.sg/MerchantApp/sim/b2sTxnEndURL.jsp","b2sTxnEndURLParam":"","s2sTxnEndURL":"https://sit2.enets.sg/MerchantApp/rest/s2sTxnEnd","s2sTxnEndURLParam":"","clientType":"W","supMsg":"","netsMidIndicator":"U","ipAddress":"127.0.0.1","language":"en"}}
```

The txnReq format is described below.
- “ss”: 1 – is default to 1 and the value is used by eNETS GW internally.
- “msg”: { <is a nested json string - refer to Appendix D – Message Format TxnReq> }

Please refer to Appendix D – Message Format description and replace the field values in the sample codes.
For easy readability the above json message is as follows.
<replace> There are 6 fields that require you to replace their values
< default value1,2> If your requirement defer from the default value, please read the Appendix D Message Format to understand the value to put in. Default 1 may be required to change. Default 2 is unlikely to be changed.
To display eNETS payment selection option page in a separate web page hosted at eNETS server instead of displaying inline (div section) embedded in Merchant web page, change the clientType value to "clientType" : "H".

(b) **Calculating the MAC value of txnReq using secretKey**
To calculate the MAC value a secretKey is required. Please refer to section 4 for the download of secretKey.
txnReq="{"ss":"1","msg":{"netsMid":"UMID_887770001","tid":"","submissionMode":"B","txnAmount ":"1000","merchantTxnRef":"20170605 10:26:51.98","merchantTxnDtm":"20170605 10:26:51.989","paymentType":"SALE","currencyCode":"SGD","paymentMode":""","merchantTimeZone":"+8:00","b2sTxnEndURL":"https://sit2.enets.sg/MerchantApp/sim/b2sTxnEndURL.jsp","b2sTxnEndURLParam":"","s2sTxnEndURL":"https://sit2.enets.sg/MerchantApp/rest/s2sTxnEndURL","s2sTxnEndURLParam":"","clientType":"W","supMsg":"","netsMidIndicator":"U","ipAddress":"127.0.0.1","language":"en"}}

//pseudocode
MAC value = Base64Encode(SHA256(txnReq+secretKey))

//JAVA SAMPLE
public static String generateSignature(String txnReq, String secretKey) throws Exception{
    String concatPayloadAndSecretKey = txnReq + secretKey;
    String hmac = encodeBase64(hashSHA256ToBytes(concatPayloadAndSecretKey.getBytes()));
    System.out.println("hmac" + hmac);
    return hmac;
}

public static byte[] hashSHA256ToBytes(byte[] input) throws Exception {
    byte[] byteData = null;
    MessageDigest md = MessageDigest.getInstance("SHA-256");
    md.update(input);
    byteData = md.digest();
    return byteData;
}

public static String encodeBase64(byte[] data) throws Exception {
    return
DatatypeConverter.printBase64Binary(data);
}
(c) Embed the downloaded keyId, MAC value and txnReq into the HTML Page and send it to the browser.

```html
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Merchant Page</title>
<script src="https://uat2.enets.sg/GW2/js/jquery-3.1.1.min.js" type="text/javascript"></script>
<script src="https://uat2.enets.sg/GW2/pluginpages/env.jsp"></script>
<script type="text/javascript" src="https://uat2.enets.sg/GW2/js/apps.js"></script>
</head>
<body>
<input type="hidden" id="txnReq" name="txnReq" value="${txnReq}'">
<input type="hidden" id="keyId" name="keyId" value="${KEY_ID}'">
<input type="hidden" id="hmac" name="hmac" value="${HMAC}'">
<div id="anotherSection">
<fieldset>
<div id="ajaxResponse"></div>
</fieldset>
</div>
<script>
window.onload = function() {
    var txnReq = document.forms[0].txnReq.value;
    var keyId = document.forms[0].keyId.value; // once api key is available, assign a value
    var hmac = document.forms[0].hmac.value; // once hmac is available, assign a value
    sendPayLoad(txnReq, hmac, keyId);
};
</script>
</body>
</html>
```
### Step 4
The Merchant Portal provides a URL for eNETS Gateway to return with the MAC Value and TxnRes message. The portal verifies the MAC Value using the secretKey and processes the TxnRes. If transaction status is successful, the transaction is considered approved.

<table>
<thead>
<tr>
<th>HTTP HEADER</th>
<th>keyld: &lt;echo from txnReq&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>hmac: &lt;generated by eNETS GW&gt;</td>
</tr>
</tbody>
</table>
```java
@RequestMapping(consumes = MediaType.APPLICATION_JSON_VALUE,
    value = "/s2sTxnEnd", method = RequestMethod.POST)

public ResponseEntity<Void> receiveS2STxnEnd(@RequestBody String txnRes,
    HttpServletRequest request) {

    log.debug("MERCHANT APP : in receiveS2STxnEnd :" +
    txnRes); // json message received as string

    try {
        String generatedHmac =
        Util.generateSignature(txnRes,
            "f49015ce-84fd-4e9a-a24e-8aeb30d870d6"); // generate mac

        String macFromGW = request.getHeader("hmac");

        log.info("MERCHANT APP : header hmac received :" + macFromGW); //

        log.info("MERCHANT APP : header hmac generated :");

        if (generatedHmac.equalsIgnoreCase(macFromGW)) {
            // parse message

            SoapiS2S txnResObj = mapper.readValue(txnRes,
                SoapiS2S.class);

            log.info("MERCHANT APP : in receiveS2STxnEnd :");

            txnResObj; // Please handle success or failure response code

        }

        else{
            log.error("signature not matched.");

        }
    }

```
//handle exception flow

}  
} catch (Exception e) {
    // TODO handle exception
    log.error(e);  
}
  
  return new ResponseEntity<Void>(HttpStatus.OK);

*Please refer to Appendix D for the message format and Appendix E to interpret the response message. Below is an example of txnRes message for CREDIT non-3D and 3D.

```
{"ss":1,"msg":{"netsMid":"UMID_887770001","merchantTxnRef":"20170821
17:07:51.69","merchantTxnDtm":"20170821
17:07:51.690","paymentType":"SALE","currencyCode":"SGD","netsTxnRef":"20170821170826
664","netsTxnDtm":"20170821
17:08:27.000","paymentMode":"CC","merchantTimeZone":"+8:00","netsTxnStatus":"0","netsTxnMsg":"Approval","netsAmountDeducted":"1000","maskPan":"4111XXXXXXXXX1111","bank
AuthId":"014089","stageRespCode":"0005-00000","txnRand":"20170821170803492","actionCode":"0","netsMidIndicator":"U"}}
```

For easy readability the above json message is as follows.
{
  "ss":1,
  "msg":{
    "netsMid":"UMID_887770001",
    "merchantTxnRef":"20170821 17:07:51.69",
    "merchantTxnDtm":"20170821 17:07:51.690",
    "paymentType":"SALE",
    "currencyCode":"SGD",
    "netsTxnRef":"20170821170826664",
    "netsTxnDtm":"20170821 17:08:27.000",
    "paymentMode":"CC",
    "merchantTimeZone":"+8:00",
    "netsTxnStatus":"0",
    "netsTxnMsg":"Approval",
    "netsAmountDeducted":"1000",
    "maskPan":"4111XXXXXXXX1111",
    "bankAuthId":"014089",
    "stageRespCode":"0005-00000",
    "txnRand":"20170821170803492",
    "actionCode":"0",
    "netsMidIndicator":"U"
  }
}
Step 6
The Merchant Portal receives MAC Value and TxnRes. If no response is received in step 4, the Merchant Portal can expect the MAC value and TxnRes message from the browser and can verify that the MAC value is correct using its secretKey. If transaction status is successful, the transaction is considered approved.

The MAC value and TxnRes from step 4 (S2S) and step 6 are the same. Merchant only needs to process the MAC value and TxnRes once.

Data as http request parameters
- keyId:<echoed back from request>
- MAC:<generated by eNETS GW>
- TxnRes: <as shown in the below sample>

SAMPLE – Receives message and writes to the output.

```java
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@ page import="java.util.*,java.io.*,java.net.*"%>

String header = request.getParameter("hmac");
System.out.println("MerchantApp:b2sTxnEndUrl : hmac: " + header);
Object message = request.getParameter("message"); // contains TxnRes message
System.out.println("MerchantApp:b2sTxnEndUrl : data, message: " + message);
String txnRes = String.valueOf(message);
try {
    txnRes = URLDecoder.decode(txnRes + ",", "UTF-8");
    if (generatedHmac.equalsIgnoreCase(header)) {
        Soapi txnResObj = mapper.readValue(txnRes, Soapi.class);
        log.info("MERCHAND APP : in receiveb2sTxnEnd :");
        txnResObj;
    }
}

// please handle the success or failure response
```
Step 7
The Merchant Portal sends the receipt page to the browser.
5.2 Transaction Flow 2

**Merchant Mobile Native App with eNETS Mobile SDK**

The diagram below shows a transaction flow whereby a payer makes an in-app purchase from a merchant app. The merchant app is compiled with a SDK provided by eNETS.

NETSPay (see Appendix A for definition) is used in this example to make payment after payer check-out. This flow is also applicable for other similar payment methods such as DBS PayLah, OCBC PayAnyOne and UOB Mighty which has a mobile app.

This flow has primarily 2 parts (upon payer checkout):

1. In Step 2, merchant sends a transaction request to eNETS Gateway through the Merchant mobile app and eNETS SDK
2. In Step 5a and 6, merchant receives s2sTxnEnd (Step 6) and b2sTxnEnd (Step 5a)

Merchant only needs to wait for either s2sTxnEnd or b2sTxnEnd to confirm whether the transaction response is approved. eNETS will ensure both b2sTxnEnd and s2sTxnEnd have the same value.

---

**Diagram:**

- **Merchant:**
  - Check Out
  - Sends keyld, MAC Value, TxnReq
  - Returns keyld, Mac Value, TxnRes-s2sTxnEnd

- **Mobile Native App:**
  - eNETS SDK
  - NETSPay
  - Backend Interaction
  - Returns TxnComplete

- **Gateway:**
  - Action Require Flow
  - Automated Flow
5.2.1 Description

The sequence diagram below describes the Mobile SDK transaction flow with NETSPay App. Relevant steps for merchants (2, 3, 5, 6, 7) are explained in greater details below.

1. Payer chooses "Checkout"

2. Merchant Portal: Sends (keyId, MAC Value & TxnReq)

3. eNETS Mobile SDK: Sends (keyId, MAC Value & TxnReq)

4. eNETS Mobile SDK, NETSPay App and Gateway perform a series of background activities such as Present/Select Payment Option, Render/Submit QR code

5. Returns TxnComplete

5a. Returns keyId, MAC Value, and TxnRes-b2sTxnEnd

6. Returns keyId, MAC Value and TxnRes-s2sTxnEnd

7. Sends receipt page
Step 2
Merchant shall create (a) the Transaction Request (txnReq) message and (b) generate the MAC value of txnReq. Together with the downloaded keyId, the merchant shall send them to the Merchant App. The keyId can be downloaded from the NETS Admin Portal.

(a) **txnReq message**
The txnReq message is a JSON string. Merchants shall refer to section 5.2.2 to form the actual txnReq message. Below is an example of txnReq message.

```json
{"ss":"1","msg":{"netsMid":"UMID_887770001","tid":"","submissionMode":"B","txnAmount":"1000","merchantTxnRef":"20170605 10:36:51.98","merchantTxnDtm":"20170605 10:36:51.989","paymentType":"SALE","currencyCode":"SGD","paymentMode":"","merchantTimezone":"+8:00","b2sTxnEndURL":"https://sit2.enets.sg/MerchantApp/sim/b2sTxnEndURL.jsp","b2sTxnEndURLParam":"","s2sTxnEndURL":"https://sit2.enets.sg/MerchantApp/rest/s2sTxnEnd","s2sTxnEndURLParam":null,"clientType":"S","supMsg":null,"netsMidIndicator":"U","ipAddress":"127.0.0.1","language":"en","mobileOs":"ANDROID"}}
```

The txnReq format is described below.
- **“ss”:”1”** – is default to 1 and the value is used by eNETS GW internally.
- **“msg”: { <is a nested json string - refer to Appendix D – Message Format TxnReq> }**

Please refer to Appendix D – Message Format description and replace the field values in the sample codes.
For easy readability the above json message is as follows.

<replace>
There are 6 fields that require you to replace their values
</replace>

<default value1,2>
If your requirement defer from the default value, please read the Appendix D Message Format to understand the value to put in.
Default1 may be required to change. Default 2 is unlikely to be changed.
```json
{
  "ss": "1", <Default>
  "msg": {
    "txnAmount": "1000", <replace>
    "merchant_txn_ref": "20170605 10:36:51.98", <replace>
    "b2s_txn_end_url": "https://sit2.enets.sg/MerchantApp/sim/b2sTxnEndURL.jsp", <replace>
    "s2s_txn_end_url": "https://sit2.enets.sg/MerchantApp/rest/s2sTxnEnd", <replace>
    "netsMid": "UMID_887770001", <replace>
    "merchant_txn_dtm": "20170605 10:36:51.989", <replace>
    "mobileOs": "ANDROID", <default value1>
    "submissionMode": "B", <default value1>
    "paymentType": "SALE", <default value1>
    "paymentMode": "", <default value1>
    "clientType": "S", <default value1>
    "currencyCode": "SGD", <default value2>
    "merchantTimezone": "+8:00", <default value2>
    "language": "en", <default value2>
    "netsMidIndicator": "U", <default value2>
  }
}
```
Calculating the MAC value of txnReq using secretKey

To calculate the MAC value a secretKey is required. Please refer to section 4 for the download of secretKey.

```java
//pseudocode
MAC value = Base64Encode(SHA256(txnReq+secretKey))

//JAVA SAMPLE
public static String generateSignature(String txnReq, String secretKey) throws Exception{
    String concatPayloadAndSecretKey = txnReq + secretKey;
    String hmac = encodeBase64(hashSHA256ToBytes(concatPayloadAndSecretKey.getBytes()));
    System.out.println("hmac" + hmac);
    return hmac;
}

public static byte[] hashSHA256ToBytes(byte[] input) throws Exception {
    byte[] byteData = null;
    MessageDigest md = MessageDigest.getInstance("SHA-256");
    md.update(input);
    byteData = md.digest();
    return byteData;
}

public static String encodeBase64(byte[] data) throws Exception {
    return DatatypeConverter.printBase64Binary(data);
}
```
Step 3
The Merchant App upon receiving (a) txnReq, (b) MAC value and (c) keyId shall invoke the send(String txnReq, String MAC value, String keyId, function callback()) function of the NETS Mobile SDK. A callback function should be provided by the Merchant App. The NETS SDK will invoke the callback function at Step 5 when the transaction is complete.

Android:
/*This API_KEY_ID & API_SECRET_KEY can be taken once the Merchant register with NETS Pte Ltd.
String txn = <to be generated by merchant on merchant server>
String hmac = <to be generated by merchant on merchant server>
String key = API KEY_ID;
PaymentCallback interface should be implemented by merchant. */

PaymentRequestManager manager = PaymentRequestManager.getInstance();
try {
    manager.sendPaymentRequest(key, hmac, txn, new PaymentCallback() {
        @Override
        public void onResult(final PaymentResponse response) {
            /*
             * This sample codes check whether the instance belongs to
             * DebitCreditPaymentResponse or NonDebitCreditPaymentResponse. It
             * will extract the data accordingly.
             * If it is DebitCreditPaymentResponse object, response will contain
             * the TXN response, HMAC value and Key ID.
             * If it is NonDebitCreditPaymentResponse object, the response comes
             * from in-app communication and it will contain the status. In this
             * case, it is NETS Pay.
             */
            if (response instanceof DebitCreditPaymentResponse) {
                final DebitCreditPaymentResponse debitCreditResponse =
                        (DebitCreditPaymentResponse) response;
                String txnRes = debitCreditResponse.txnResp;
                String hmac = debitCreditResponse.hmac;
                String keyId = debitCreditResponse.keyId;
            } else if (response instanceof NonDebitCreditPaymentResponse) {
                final NonDebitCreditPaymentResponse nonDebitCreditResponse =
                        (NonDebitCreditPaymentResponse) response;
                String txn_Status = nonDebitCreditResponse.status;
            }
            }
        }
    , context);
} catch (InvalidPaymentRequestException e) {
    Log.e("InvalidPaymentRequestException... ", e.getMessage());
} catch (Exception e) {
    Log.e("Exception... ", e.getMessage());
}
For onResult of QR:
If txn_Status = "00000", the transaction request is considered to be received by the eNETS Gateway but the transaction is not approved yet. Payer should wait for the receipt sent down by Merchant Portal to the Merchant Mobile App.

For onResult of eDebit and eCredit:
Parse txnRes and look for TxnEnd. The transaction request is considered received by eNETS Gateway but the transaction is not approved yet. Merchant should send the b2sTxnEnd response message from Mobile App back to the Merchant Portal to validate the MAC value before the message response data is considered valid. Payer should wait for the receipt sent down by Merchant Portal to the Merchant Mobile App.

For onFailure:
Please refer Appendix E to interpret the error message from the txn_ResponseCode field.

iOS:
At Merchant App checkout page, merchant needs to implement payment request delegate to receive payment response from SDK. Following classes are available for mobile application:
- PaymentRequestManager
- PaymentRequest
- PaymentResponse
- PaymentRequestDelegate
- NETSError

```swift
/* Payment callback from eNETS SDK */
public protocol PaymentRequestDelegate {
    func onResult(response : PaymentResponse)
    func onFailure(error : NETSError)
}
```

The following is an example of the setup.
`UIViewController`
import Foundation
import ENETSLib

class MainViewController: UIViewController, PaymentRequestDelegate {
    
    override func viewDidLoad() {
        super.viewDidLoad()
    }

    // Callback on Payment result
    public func onResult(response: PaymentResponse) {

        // TODO SDK user need to handle payment response (refer to integration guide)
        if response is DebitCreditResponse {
            let debitCreditResponse = response as! DebitCreditResponse
            let hmac       = debitCreditResponse.hmac
            let txnRes     = debitCreditResponse.txnRes
            let keyId      = debitCreditResponse.keyId
        } else if response is NonDebitCreditResponse {
            let nonDebitCreditResponse = response as! NonDebitCreditResponse
            let txn_Status  = nonDebitCreditResponse.status
        }
    }

    // Callback on Payment failure
    func onFailure(error : NETSError) {
        // TODO SDK user need to handle error code (refer to integration guide)
        let responseCode    = error.responseCode
        let actionCode      = error.actionCode
    }

    @IBAction func onCheckOutClicked(_ sender: Any) {
        let apiKey   = "replace_with_apikey";
        let txnReq   = "replace_with_txn_req" /*<to be generated by merchant on merchant
        server>*/
        let hmac     = "replace_with_hmac" /*<to be generated by merchant on merchant
        server>*/

        // Setup Payment request manager
        let paymentManager = PaymentRequestManager()
        paymentManager.paymentDelegate = self

        // Send Payment request to eNETS mobile SDK
        let request = PaymentRequest(hmac:hmac, txnReq :txnReq)
        paymentManager.sendPaymentRequest(apiKey: apiKey, paymentRequest: request,
For onResult (successful) of QR:
If txn_Status = “00000”, the transaction request is considered to be received by the eNETS Gateway but the transaction is not approved yet. Payer must wait for the receipt sent down by Merchant Portal to the Merchant Mobile App.

For onResult (successful) of eDebit and eCredit:
Parse txnRes and look for TxnEnd. The transaction request is considered received by eNETS Gateway but the transaction is not approved yet. Merchant should send the b2sTxnEnd response message from Mobile App back to the Merchant Portal to validate the MAC value before the message response data is considered valid. Payer should wait for the receipt sent down by Merchant Portal to the Merchant Mobile App.

For onFailure:
Please refer Appendix E to interpret the error message from the txn_ResponseCode field.

Step 5 & 5a

For QR Code
The Merchant App receives a nonDebitCreditResponse from NETS SDK via its callback function provided in Step 3 (see sample source code). This nonDebitCreditResponse is simply an acknowledgement that the NETSPay app has received the request. It is not the TxnEnd success/failure response. The dependable transaction response status is only found in Step 6.

For eDebit and eCredit
The Merchant App receives a debitCreditResponse (b2sTxnEnd) from the NETS SDK via its callback function provided in Step 3 (see sample source code) and forward it to the Merchant Portal.
Step 6
The Merchant Portal provides a URL for NETS Server to respond with the TxnRes (s2sTxnEnd) message.

**HTTP HEADER**
- keyId: <echo from txnReq>
- hmac: <generated by eNETS GW>

**SAMPLE.**

```java
@RequestMapping(consumes = MediaType.APPLICATION_JSON_VALUE, value = "/s2sTxnEnd", method = RequestMethod.POST)
public ResponseEntity<Void> receiveS2STxnEnd(@RequestBody String txnRes, HttpServletRequest request) {
    log.debug("MERCHANT APP : in receiveS2STxnEnd:");
    String generatedHmac = Util.generateSignature(txnRes, "f49015ce-84fd-4e9a-a24e-8aeb30d870d6");
    String macFromGW = request.getHeader("hmac");
    log.info("MERCHANT APP : header hmac received:");
    log.info("MERCHANT APP : header hmac generated: ");
    if (generatedHmac.equalsIgnoreCase(macFromGW)) {
        SoapiS2S txnResObj = mapper.readValue(txnRes, SoapiS2S.class);
        log.info("MERCHANT APP : in receiveS2STxnEnd:");
        //Please handle success or failure response code
    } else {
        log.error("signature not matched.");
        //handle exception flow
    }
    return new ResponseEntity<Void>(HttpStatus.OK);
}
```

After receiving the TxnRes message, the Merchant Portal will need to verify that the MAC value is correct using its secretKey. If transaction status is successful, the transaction is considered approved.

*Please refer to Appendix D for message format and Appendix E to interpret the response message. Below is an example of txnRes message for CREDIT non-3D and 3D.*
Step 7
The Merchant Portal sends the receipt page to the browser.
5.3 Transaction Flow 3

Web Browser with QR code

The diagram below shows a transaction flow whereby a payer makes a purchase on the merchant site on the PC using a NETS QR supported app on the mobile. A QR code is generated on the desktop browser and any NETS QR supported app such as NETSPay, DBS PayLah, OCBC PayAnyOne and UOB Mighty can be used to scan the QR code and make payment.

This flow has primarily 2 parts (upon payer checkout):
1. In Step 2, merchant sends a transaction request to eNETS Gateway through the browser
2. In Step 7 and 9, merchant receives s2sTxnEnd (Step 7) and b2sTxnEnd (Step 9)

Merchant only needs to wait for either s2sTxnEnd or b2sTxnEnd to confirm whether the transaction response is approved. NETS will ensure both b2sTxnEnd and s2sTxnEnd have the same value.
5.3.1 Description

The sequence diagram below describes the payer making payment at the Merchant Portal using a Web Browser and any eNETS-supported payment method with QR Code scanning capability. Relevant steps for merchants (2, 3, 5, 7, 8, 9, and 10) are explained in greater details below.

1. **Checkout**
   - Merchant Portal forms HTML page
   - Merchant Portal creates a TxnPeq message
   - Merchant Portal generates a MAC Value

2. **Sends (keyId, MAC Value & TxnReq)**
   - Merchant Portal embeds keyId, MAC Value and TxnReq into the HTML page.

3. **Plug-in forwards (keyId, MAC Value, TxnReq)**
   - QR App (e.g. NETSPay) displays Payment Option Page and performs a series of background interaction such as generating QR Data

4. **Transfer QR Data to QR App via Scanning**
   - Javascript Plug-in generates QR Code

5. **Sends QR Data via QR-SOF Server**
   - QR App returns TxnComplete (Just an acknowledgment)

6. **Returns keyId, MAC Value and TxnRes-s2sTxnEnd**

7. **Returns keyId, MAC Value and TxnRes-b2sTxnEnd**
   - eNETS Gateway (GW) sends receipt page
Step 2
The following is required:
(a) Create the Transaction Request (txnReq) message
(b) Generate the MAC value of txnReq and
(c) Embed the downloaded keyId, MAC value and txnReq into the HTML Page and send it to the browser. The keyId is available at NETS Admin Portal for merchant to download.

(a) txnReq message
The txnReq message is a JSON string. Merchants shall refer to section 5.3.2 to form the actual txnReq message. Below is an example of txnReq message.

```json
{"ss":1,"msg":{"netsMid":"UMID_887770001","tid":","submissionMode":"B","txnAmount":"1000","merchantTxnRef":"20170605 10:26:51.98","merchantTxnDtm":"20170605
10:26:51.989","paymentType":"SALE","currencyCode":"SGD","paymentMode":","merchant
TimeZone":+8:00","b2sTxnEndURL":"https://sit2.enets.sg/MerchantApp/sim/b2sTxnEndURL.jsp","b2sTxnEndURLErrParam":","s2sTxnEndURL":"https://sit2.enets.sg/MerchantApp/rest/s2sTxnEnd","s2sTxnEndURLErrParam":","clientType":"W","supMsg":","netsMidIndicator":"U","ipAddress":"127.0.0.1","language":"en"}}
```

The txnReq format is described below.
- "ss":1 – is default to 1 and the value is used by eNETS GW internally.
- “msg: { <is a nested json string - refer to Appendix D – Message Format TxnReq> }

Please refer to Appendix D – Message Format description and replace the field values in the sample codes.
For easy readability the above json message is as follows.

<replace>
There are 6 fields that require you to replace their values
< default value 1,2>
If your requirement defer from the default value, please read the Appendix D Message Format to understand the value to put in.
Default 1 may be required to change. Default 2 is unlikely to be changed.
To display NETS payment selection option page in a separate web page hosted at NETS server instead of displaying inline (div section) embedded in Merchant web page, change the clientType value to “clientType” : “H”.

(b) Calculating the MAC value of txnReq using secretKey
(c) To calculate the MAC value a secretKey is required. Please refer to section 4 for the download of secretKey.
MAC value = Base64Encode(SHA256(txnReq+secretKey))

//JAVA SAMPLE
public static String generateSignature(String txnReq, String secretKey) throws Exception{
    String concatPayloadAndSecretKey = txnReq + secretKey;
    String hmac = encodeBase64(hashSHA256ToBytes(concatPayloadAndSecretKey.getBytes()));
    System.out.println("hmac" + hmac);
    return hmac;
}

public static byte[] hashSHA256ToBytes(byte[] input) throws Exception{
    byte[] byteData = null;
    MessageDigest md = MessageDigest.getInstance("SHA-256");
    md.update(input);
    byteData = md.digest();
    return byteData;
}

public static String encodeBase64(byte[] data) throws Exception{
    return DatatypeConverter.printBase64Binary(data);
Embed the downloaded keyId, MAC value and txnReq into the HTML Page and send it to the browser.

```html
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
  <title>Merchant Page</title>
  <script src="https://uat2.enets.sg/GW2/js/jquery-3.1.1.min.js" type="text/javascript"></script>
  <script src="https://uat2.enets.sg/GW2/pluginpages/env.jsp"></script>
  <script src="https://uat2.enets.sg/GW2/js/apps.js"></script>
</head>
<body>
  <input type="hidden" id="txnReq" name="txnReq" value="${txnReq}'">
  <input type="hidden" id="keyId" name="keyId" value="${KEY_ID}'">
  <input type="hidden" id="hmac" name="hmac" value="${HMAC}'">
  <div id="anotherSection">
    <fieldset>
      <div id="ajaxResponse"></div>
    </fieldset>
  </div>
  <script>
    window.onload = function() {
      var txnReq = document.forms[0].txnReq.value;
      var keyId = document.forms[0].keyId.value; // once api key is available, assign a value
      var hmac = document.forms[0].hmac.value; // once hmac is available, assign a value
      sendPayLoad(txnReq, hmac, keyId);
    }
  </script>
</body>
</html>
```
**Step 3**
The browser upon receiving (a) keyId, (b) MAC value and (c) txnReq invokes the send() function of the NETS Javascript Plug-in with the following parameters (String keyId, String MAC value, String txnReq).

The NETS Javascript plug-in forwards the parameters to the NETS Gateway which will return a Payment Option page to the browser.

Payer chooses his preferred QR payment app from the list of options returned. Upon scanning the QR code with the selected app, the payer goes through the necessary steps on the payment app and the rest of the interactions are between NETS Plug-in and NETS Gateway.

**Step 7**
The Merchant Portal provides a URL for NETS Server to return with the Transaction Response (TxnRes) message.

```
HTTP HEADER
keyId: <echo from txnReq>
hmac: <generated by eNETS GW>

```

```
SAMP LE.

@RequestMapping(consumes = MediaType.APPLICATION_JSON_VALUE, value = "/s2sTxnEnd", method = RequestMethod.POST)
public ResponseEntity<Void> receiveS2STxnEnd(@RequestBody String txnRes, HttpServletRequest request) {
    log.debug("MERCHANT APP : in receiveS2STxnEnd :" + txnRes);//json message
    received as string
    try {
        String generatedHmac = Util.generateSignature(txnRes, "f49015ce-84fd-4e9a-a24e-8aeb30d870d6");//generate mac
        String macFromGW = request.getHeader("hmac");
        log.info("MERCHANT APP : header hmac received ": + macFromGW);//
        log.info("MERCHANT APP : header hmac generated ": + generatedHmac);
        if(generatedHmac.equalsIgnoreCase(macFromGW)){
            //parse message
            SoapiS2S txnResObj = mapper.readValue(txnRes, SoapiS2S.class);
            log.info("MERCHANT APP : in receiveS2STxnEnd :" + txnResObj);
            //Please handle success or failure response code
        } else{
            log.error("signature not matched.");
            //handle exception flow
        }
    } catch (Exception e) {
        // TODO handle exception
        log.error(e);
    }
    return new ResponseEntity<Void>(HttpStatus.OK);
}
```

Page 42
After receiving the TxnRes message, the Merchant Portal will need to verify that the MAC value is correct using its secretKey. If transaction status is successful, the transaction is considered approved.

*Please refer to Appendix D for message format and Appendix E to interpret the response message. Below is an example of txnRes message for QR.

```json
{"ss":"1","msg":{"netsMid":"UMID_877858000","merchantTxnRef":"2017082210:06:50.60","netsMidIndicator":"U","netsTxnRef":"20170822100709200","paymentMode":"QR","submissionMode":"B","currencyCode":"SGD","merchantTxnDtm":"2017082210:06:50.605","merchantTimeZone":"+8:00","paymentType":"SALE","clientType":"W","bankId":"232","stageRespCode":"3099-00000","txnRand":"20170822100709200","actionDate":"0","netsTxnDtm":"20170822 10:07:14.422","netsTxnStatus":"0","netsTxnMsg":"Successful","netsAmountDeducted":1000}}
```

For easy readability the above json message is as follows.

```json
{
    "ss":"1",
    "msg":{
        "netsMid":"UMID_877858000",
        "merchantTxnRef":"20170822 10:06:50.60",
        "netsMidIndicator":"U",
        "netsTxnRef":"20170822100709200",
        "paymentMode":"QR",
        "submissionMode":"B",
        "currencyCode":"SGD",
        "merchantTxnDtm":"20170822 10:06:50.605",
        "merchantTimeZone":"+8:00",
        "paymentType":"SALE",
        "clientType":"W",
        "bankId":"232",
        "stageRespCode":"3099-00000",
        "txnRand":"20170822100709200",
        "actionDate":"0",
        "netsTxnDtm":"20170822 10:07:14.422",
        "netsTxnStatus":"0",
        "netsTxnMsg":"Successful",
        "netsAmountDeducted":1000
    }
}
```
Step 9
If no response is received in step 7, the Merchant Portal can expect the MAC value and TxnRes message from the browser and can verify that the MAC value is correct using its secretKey. If transaction status is successful, the transaction is considered approved.

The MAC value and TxnRes from step 7 (S2S) and step 8 are the same. Merchant only needs to process the MAC value and TxnRes once.

---

**Data as http request parameters**

keyId:<echoed back from request>
MAC:<generated by eNETS GW>
TxnRes: <as shown in the below sample>

---

**SAMPLE – Receives message and writes to the output.**

```html
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1" %>
<%@ page import="java.util.*,java.io.*,java.net.*" %>

String header = request.getParameter("hmac");
System.out.println("MerchantApp:b2sTxnEndUrl : hmac: "+header);
Object message = request.getParameter("message"); //contains TxnRes message
System.out.println("MerchantApp:b2sTxnEndUrl : data, message: "+message);
String txnRes = String.valueOf(message);
try {
    txnRes = URLDecoder.decode(txnRes + ", " + "UTF-8");
    if(generatedHmac.equalsIgnoreCase(header)){
        Soapi txnResObj = mapper.readValue(txnRes, Soapi.class);
        log.info("MERCHANT APP : in receiveb2sTxnEnd : " + txnResObj);
        //please handle the success or failure response code.
    } else{
        log.error("signature not matched.");
        //handle exception flow
    }
} catch (Exception ex) {
    ex.printStackTrace();
}
```
Step 10
The Merchant Portal sends the receipt page to the browser.

```java
// Only for testing - writing to output stream
// out.write("decoded message :"+header +"-----------
"+decodedMsg);
```
5.4 Transaction Flow 4

eCredit Server-To-Server (S2S)

This flow is for Merchants, who wish to capture credit cards directly, which also subjects the merchant to PCI certification as a requirement. In the S2S flow, the merchant has more control but will need to manage parts of the transaction flow such as 3DSecure and the construction of the ACS redirection.

This flow has primarily 2 parts (upon payer checkout):

- In Step 4, merchant sends a transaction request to eNETS Gateway through the browser
- In Step 8 and 10, merchant receives s2sTxnEnd (Step 8) and b2sTxnEnd (Step 10). Merchant only needs to wait for either s2sTxnEnd or b2sTxnEnd to confirm whether the transaction response is approved. eNETS will ensure both b2sTxnEnd and s2sTxnEnd have the same value.
5.4.1 Description
The sequence diagram below describes the payer making payment at the Merchant Portal using a Server-To-Server mode of payment. Relevant steps for merchants (4, 5, 6, 8, and 10) are explained in greater details below.

Merchant Portal

1. Checkout
2. Merchant presents Credit Card Capture Page
3. Payer submit credit card details
4. Sends (keyId, MAC Value & TxnReq)
5. Returns PAREq, keyId, MAC Value, 3D info
5a. Sends Receipt Page
6. Sends (keyId, MAC Value & TxnReq)
7. Returns PARes, eNETS URL
8. Returns keyId, MAC Value and TxnRes-s2sTxnEnd
9. Returns keyId, MAC Value and TxnRes-b2sTxnEnd
10. Returns keyId, MAC Value and TxnRes-s2sTxnEnd
11. Sends receipt page
Step 1, 2, 3
The first 3 steps are handled by the merchant. It starts from payer checkout page to the point where credit card details are captured.

Step 4
The following is required:
(a) Create the Transaction Request (txnReq) message
(b) Generate the MAC value of txnReq and
(c) Send keyId, MAC value and txnReq eNETS Gateway. The keyId is available at eNETS Admin Portal for merchant to download.

(a) txnReq message
The txnReq message is a JSON string. Merchants shall refer to section 5.4.2 to form the actual txnReq message. Below is an example of txnReq message.

```json
{"ss":"1","msg":{"b2sTxnEndURL":"https://sit2.enets.sg/MerchantApp/sim/b2sTxnEndURL.jsp","b2sTxnEndURLParam":"","cardHolderName":"Ah Hoa","currencyCode":"SGD","cvv":"232","expiryDate":"1901","ipAddress":"127.0.0.1","merchantTimeZone":"+8:00","merchantTxnDtm":"20170605 10:40:17.312","merchantTxnRef":"20170605 10:40:17.31","netsMid":"UMID_887770001","netsMidIndicator":"U","pan":"411111111111111","param1":"","param2":"","param3":"","param4":"","param5":"","paymentMode":"CC","paymentType":"SALE","s2sTxnEndURL":"https://sit2.enets.sg/MerchantApp/rest/s2sTxnEnd","s2sTxnEndURLParam":"","submissionMode":"S","tid":"127.0.0.1","txnAmount":"1000"}}
```

The txnReq format is described below.
- "ss": "1" – is default to 1 and the value is used by eNETS GW internally.
- "msg": { <is a nested json string - refer to Appendix D – Message Format TxnReq> }

Please refer to Appendix D – Message Format description and replace the field values in the sample codes.
For easy readability the above json message is as follows.
<replace>
<default value 1,2>
There are 6 fields that require you to replace their values
If your requirement defer from the default value, please read the Appendix D Message Format to understand the value to put in. Default 1 may be required to change. Default 2 is unlikely to be changed.
For easy readability the above json message is as follows.

```json
{
    "ss": "1",
    "msg": {
        "txnAmount": "1000",
        "merchantTxnRef": "20170605 10:40:17.31",
        "b2sTxnEndURL": "https://sit2.enets.sg/MerchantApp/sim/b2sTxnEndURL.jsp",
        "s2sTxnEndURL": "https://sit2.enets.sg/MerchantApp/rest/s2sTxnEnd",
        "netsMid": "UMID_887770001",
        "merchantTxnDtm": "20170605 10:40:17.312",
        "cardHolderName": "Ah Hoa",
        "cvv": "232",
        "expiryDate": "1901",
        "pan": "4111111111111111",
        "submissionMode": "S",
        "paymentType": "SALE",
        "paymentMode": "CC",
        "currencyCode": "SGD",
        "merchantTimeZone": "+8:00",
        "netsMidIndicator": "U"
    }
}
```

(b) **Calculating the MAC value of txnReq using secretKey**

To calculate the MAC value a secretKey is required. Please refer to section 4 for the download of secretKey.
//pseudocode
MAC value = Base64Encode(SHA256(txnReq+secretKey))

//JAVA SAMPLE
public static String generateSignature(String txnReq, String secretKey) throws Exception{
    String concatPayloadAndSecretKey = txnReq+ secretKey;
    String hmac = encodeBase64(hashSHA256ToBytes(concatPayloadAndSecretKey.getBytes ()));
    System.out.println("hmac" + hmac);
    return hmac;
}

public static byte[] hashSHA256ToBytes(byte[] input) throws Exception{
    byte[] byteData = null;
    MessageDigest md = MessageDigest.getInstance("SHA-256");
    md.update(input);
    byteData = md.digest();
    return byteData;
}

public static String encodeBase64(byte[] data) throws Exception {

Step 5
The Merchant Portal receives a TxnEnd message. If TxnEnd.netsTxnStatus is not equal to 5, merchant must return a receipt page to the browser. If TxnEnd.netsTxnStatus is equal 5, the eNETS GW will provide the 3D info (eci, pareq, termUrl, md, acsUrl) and the merchant will send them (the 3D info: eci, pareq, termUrl, md) to the acsUrl via browser redirection.

Sample TxnEnd when netsTxnStatus is “5”

```json
{"ss":"1","msg":{"netsMid":"887770001","merchantTxnRef":"20170703 11:51:11.97","merchantTxnDtm":"20170703 11:51:11.975","netsTxnRef":"20170703115143837","netsTxnDtm":"20170703 11:51:43.000","paymentMode":"CC","merchantTimeZone":"+8:00","pareq":"eJxVUtUtuwjAM/R XEB5CE3mjrRiorG0hLhzaQ9loVay2jF9J06/b1Szo6WFr7icx24d28eBXS4XfjM O0kcBLZt+oaT4hBOf9Ezn2fOpZPvcXcpcyZchhYRh8o26KuOJvR2RzICLWAzPK0UhzS7LzcJN x2 2MKygVwglCg3MXc9nzHKXJcOB8gvDnVallfYqlFhaw8cZHVVXKfnFpaqqEUAntxQmkDQtjc 083Q GQzI8X7al2G4ei1ZhhXKWq/IExCQDuXa47UzcUavG+OHARR5/iGPXJd2SJWPVt++TNnjioUwubmAw4 pAr5nDkPetSaMBY4Ld1bAMPAPw64myY5wKgMTWi25dBTrEqtsHgHtEgHI1TV6gNPxXww HbjCf1 JNa3LBqHdFFQnkOszd2pifKe3nujfrbrV+8N/yotIltUiJxFWJyWMSYQoU2kvmUDZUMAGkl yGXT5PIndPTv/wA0Em58Q==","termUrl":"https://sit2.enets.sg/GW2/credit/paResSubmitFor m","md":"20170703115143837","acsUrl":"https://pit.3dsecure.net/VbVTestSuiteService/pit1/a csService/paReq?summary=ZDE5MDBIbNDYiNYWY1Zi00ZjEo0LiZjiZnjdYhWyU0MWUxZTYy","nets TxnStatus":"5","netsTxnMsg":"Approval","txnRand":"20170703115143635"}
```
Sample 3D redirection
After receiving the TxnRes message, the Merchant Portal will need to verify the MAC value is correct using its secretKey. If transaction status is successful, the transaction is considered approved.
Step 8
The Merchant Portal provides a URL for NETS Server to return with the Transaction Response (TxnRes) message.

| HTTP HEADER | keyId: <echo from txnReq>  
|             | hmac: <generated by eNETS GW> |

SAMPLE.

```java
@RequestMapping(consumes = MediaType.APPLICATION_JSON_VALUE, value = "/s2sTxnEnd", method = RequestMethod.POST)
public ResponseEntity<Void> receiveS2STxnEnd(@RequestBody String txnRes, HttpServletRequest request) {
    log.debug("MERCHANT APP : in receiveS2STxnEnd :" + txnRes); // json message
    received as string

    try {
        String generatedHmac = Util.generateSignature(txnRes, "f49015ce-84fd-4e9a-a24e-8aebe3d870d6");

        // generate mac
        String macFromGW = request.getHeader("hmac");

        // log.info("MERCHANT APP : header hmac received :" + macFromGW);
        // log.info("MERCHANT APP : header hmac generated :" + generatedHmac);
        if(generatedHmac.equalsIgnoreCase(macFromGW)){
            // parse message
            SoapiS2S txnResObj = mapper.readValue(txnRes, SoapiS2S.class);
            log.info("MERCHANT APP : in receiveS2STxnEnd :" + txnResObj);
            // Please handle success or failure response codes
        } else{
            log.error("signature not matched.");
            // handle exception flow
        }
    } catch (Exception e) {
        // TODO handle exception
        log.error(e);
    }

    return new ResponseEntity<Void>(HttpStatus.OK);
}
```

After receiving the TxnRes message, the Merchant Portal will need to verify that the MAC value is correct using its secretKey. If transaction status is successful, the transaction is considered approved.

*Please refer to Appendix D for message format and Appendix E to interpret the response message. Below is an example of txnRes message for CREDIT non-3D and 3D.

```json
{"ss":"1","msg":{"netsMid":"887770001","tid":"127.0.0.1","merchantTxnRef":"20170821
17:20:23.56","merchantTxnDtm":"20170821
17:20:23.560","paymentType":"SALE","currencyCode":"SGD","netsTxnRef":"2017082117203066
6","netsTxnDtm":"20170821
17:20:30.000","paymentMode":"CC","merchantTimeZone":"+8:00","netsTxnStatus":"0","netsTxn
Msg":"Approval","netsAmountDeducted":"1000","bankAuthId":"014092","stageRespCode":"0
005-00000","txnRand":"20170821172030821","actionCode":"0","netsMidIndicator":"U"}
```
Step 10
If no response is received in step 8, the Merchant Portal can expect the MAC value and TxnRes message from the browser and can verify that the MAC value is correct using its secretKey. If transaction status is successful, the transaction is considered approved.

The MAC value and TxnRes from step 8 (S2S) and step 10 are the same. Merchant only needs to process the MAC value and TxnRes once. Upon processing, the Merchant receipt page can be shown and the transaction flow is complete.
```java
8859-1
pageEncoding="ISO-8859-1">
<%@ page import="java.util.*,java.io.*,java.net.*"%>
<% String header = request.getParameter("hmac");
    System.out.println("MerchantApp:b2sTxnEndUrl : hmac: "+header);
    Object message = request.getParameter("message"); // contains
TxnRes message
    System.out.println("MerchantApp:b2sTxnEndUrl : data,
message: "+ message);
    String txnRes = String.valueOf(message);
    try {
        txnRes = URLDecoder.decode(txnRes + "", "UTF-8");
        if (generatedHmac.equalsIgnoreCase(header)) {
            Soapi txnResObj = mapper.readValue(txnRes, Soapi.class);
            log.info("MERCHANT APP : in receiveb2sTxnEnd :" + txnResObj);
            // please handle the success or failure response code.
        } else {
            log.error("signature not matched.");
            // handle exception flow
        }
    } catch (Exception ex) {
        ex.printStackTrace();
    }
    // Only for testing - writing to output stream
    // out.write("decoded message :"+header +"------------"+decodedMsg);
%>
```
6 Transaction Query

6.1 Description
The transaction query URL allows merchants to query the status of a transaction in the event where there is no response from eNETS Gateway or if the status is unknown.

- **Transaction Query Request**
  - Send with MAC value, keyId via HTTP header
  - Query is in HTTP REST call
  - Input is in JSON format

- **Transaction Query Response**
  - Response data contains MAC value and keyId in HTTP header

- **URL**
  - Production URL: https://api.nets.com.sg/GW2/TxnQuery

(a) **TxnQueryReq message**
The txnReq message is a JSON string. Please refer to section 7.2 to form the actual TxnQueryReq message. An example of txnReq message as follows.

```json
{"ss":"1","msg":{"netsMid":"UMID_878163000","merchantTxnRef":"20170609 14:13:57.79","netsMidIndicator":"U"}}
```

For easy readability the above json message is as follows.

```json
{
  "ss":"1",
  "msg":{
    "netsMid":"UMID_878163000",
    "merchantTxnRef":"20170609 14:13:57.79",
    "netsMidIndicator":"U"
  }
}
```

(b) **Calculating the MAC value of txnReq using secretKey**
To calculate the MAC value a secretKey is required. Please refer to section 4 for the download of secretKey.


```
txnReq="ss":"1","msg":{"netsMid":"UMID_878163000","merchantTxnRef":"20170609
14:13:57.79","netsMidIndicator":"U"}

//pseudocode
MAC value = Base64Encode(SHA256(txnReq+secretKey))

//JAVA SAMPLE
public static String generateSignature(String txnReq,String secretKey)
throws Exception{
    String concatPayloadAndSecretKey = txnReq + secretKey;
    String hmac = encodeBase64(hashSHA256ToBytes(concatPayloadAndSecretKey.getBytes()));
    System.out.println("hmac" + hmac);
    return hmac;
}

public static byte[] hashSHA256ToBytes(byte[] input) throws Exception
{
    byte[] byteData = null;
    MessageDigest md = MessageDigest.getInstance("SHA-256");
    md.update(input);
    byteData = md.digest();
    return byteData;
}

public static String encodeBase64(byte[] data) throws Exception
{
    return DatatypeConverter.printBase64Binary(data);
}
```

(c) **Send TxnQueryReq to GW**

POST to the REST URL with the request and the GW will respond with the TxnQueryResponse.
```java
/**
 * KEY-ID - provided by eNETS
 * SECRET-KEY - provided by eNETS
 * @param strGWPostURL - https://<domain-name>/GW2/TxnQuery
 * @param soapiTxnQueryReq - pojo contains getter setter based on
 * message format
 * @throws Exception
 */

public void sendTxnQueryReqToGW(String strGWPostURL, String secretKey, String keyId, SoapiS2S soapiTxnQueryReq) throws Exception {
    String soapiToGW = mapper.writeValueAsString(soapiTxnQueryReq);
    String signatureForReq = Util.generateSignature(soapiToGW, secretKey);
    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    headers.set("keyId", keyId);
    headers.set("hmac", signatureForReq);
    HttpEntity<String> entity = new HttpEntity<String>(soapiToGW, headers);
    ResponseEntity<String> response = getRestTemplate().exchange(strGWPostURL, HttpMethod.POST, entity, String.class);
    log.info("S2S response status : " + response.getStatusCodeValue());
    String stringBody = response.getBody();
    String hmacResponseFromGW = response.getHeaders().getFirst("hmac");
    String hmacForResponseGenerated = Util.generateSignature(stringBody, secretKey);

    if (hmacForResponseGenerated.equalsIgnoreCase(hmacResponseFromGW)){
        //parse json response
        SoapiS2S soapi2Res = mapper.readValue(stringBody, SoapiS2S.class);
        //handle business logic based on response codes
    } else {
        log.error("signature not matched.");
        //handle exception flow
    }
```
(d) **TxnQueryRes message**

```
{"ss":"1","msg":{"netsMid":"UMID_878163000","merchantTxnRef":"20170609 14:13:57.79","netsMidIndicator":"U","stageRespCode":"0000_-2018"}}
```

For easy readability the above json message is as follows.

```
{
    "ss":"1",
    "msg":{
        "netsMid":"UMID_878163000",
        "merchantTxnRef":"20170609 14:13:57.79",
        "netsMidIndicator":"U",
        "stageRespCode":"0000_-2018"
    }
}
```

### 1.1 Message Format

<table>
<thead>
<tr>
<th><strong>TxnQueryReq</strong></th>
<th>Field Name</th>
<th>Field Type</th>
<th>Max Length</th>
<th>Description</th>
<th>MOC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>netsMid</td>
<td>AN</td>
<td>20</td>
<td>The Merchant ID used in the transaction to be queried.</td>
<td>M</td>
</tr>
<tr>
<td></td>
<td>merchantTxnRef</td>
<td>AN</td>
<td>20</td>
<td>Unique reference code used in the transactions to be queried.</td>
<td>M</td>
</tr>
<tr>
<td></td>
<td>paymentType</td>
<td>A</td>
<td>5</td>
<td>Payment type used in the transaction to be queried.</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td>netsMidIndicator</td>
<td>A</td>
<td>1</td>
<td>“U” for UMID</td>
<td>M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>TxnQueryRes</strong></th>
<th>Field Name</th>
<th>Field Type</th>
<th>Max Length</th>
<th>Description</th>
<th>MOC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>netsMid</td>
<td>AN</td>
<td>20</td>
<td>Echo from TxnQueryReq</td>
<td>M</td>
</tr>
<tr>
<td></td>
<td>merchantTxnRef</td>
<td>AN</td>
<td>20</td>
<td>Echo from TxnQueryReq transactions to be queried.</td>
<td>M</td>
</tr>
<tr>
<td></td>
<td>paymentType</td>
<td>A</td>
<td>5</td>
<td>Echo from TxnQueryReq</td>
<td>C</td>
</tr>
<tr>
<td>netsMidIndicator</td>
<td>A</td>
<td>1</td>
<td>Echo from TxnQueryReq</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>----</td>
<td>---</td>
<td>-----------------------------------------------</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>stageRespCode</td>
<td>NS</td>
<td>13</td>
<td>The response code or error code returned by individual payment instruments.</td>
<td>M</td>
<td></td>
</tr>
</tbody>
</table>
| netsTxnStatus     | NS | 5 | Status of the Transaction.  
0: Transaction is successful  
1: Transaction failed  
5: Merchant to redirect payer browser to acsURL  
9: Payer cancelled the transaction. | M |
## 7 Recurring Payment

Note:
To register an RPI via eNETS Online Transaction Mode, the following fields are to be populated.

### TxnReq.param1

<table>
<thead>
<tr>
<th>S/N</th>
<th>Field Name</th>
<th>Attributes</th>
<th>Length (Max)</th>
<th>Description</th>
<th>MOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Param1</td>
<td>A</td>
<td>3</td>
<td>Fixed: &quot;RPP&quot;m2.</td>
<td>M</td>
</tr>
</tbody>
</table>

### TxnReq.param2 (The sub-fields to be constructed as JSON String)

<table>
<thead>
<tr>
<th>S/N</th>
<th>Sub-Field Name</th>
<th>Attributes</th>
<th>Length (Max)</th>
<th>Description</th>
<th>MOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>rpild</td>
<td>AN</td>
<td>20</td>
<td>RPI ID - ID uniquely identifies the new RPI.</td>
<td>M</td>
</tr>
<tr>
<td>2</td>
<td>netsMid</td>
<td>AN</td>
<td>20</td>
<td>eNETS MID - ID uniquely identifies the eNETS enrolled merchant.</td>
<td>M</td>
</tr>
<tr>
<td>3</td>
<td>subPaymtAmt</td>
<td>N</td>
<td>16</td>
<td>Subsequent Payment Amount - Amount charged for the subsequent payment cycles after the first payment is made. In cents or the lowest currency unit.</td>
<td>M</td>
</tr>
<tr>
<td>4</td>
<td>paymtFreq</td>
<td>N</td>
<td>2</td>
<td>Frequency of Payment - Number of frequency unit.</td>
<td>M</td>
</tr>
<tr>
<td>S/N</td>
<td>Sub-Field Name</td>
<td>Attributes</td>
<td>Length (Max)</td>
<td>Description</td>
<td>MOC</td>
</tr>
<tr>
<td>-----</td>
<td>----------------</td>
<td>------------</td>
<td>--------------</td>
<td>-------------</td>
<td>-----</td>
</tr>
</tbody>
</table>
| 5   | freqUnit       | N          | 1            | Frequency Unit  
‘0’ denotes weekly  
‘1’ denotes monthly | M   |
| 6   | numPaymtTerms  | N          | 3            | Number of Payment Terms - Total number of payment cycles including the first payment.  
‘0’ for infinite | M   |
| 7   | custId         | AN         | 20           | Customer ID - ID uniquely identifies the merchant’s customer associated with this RPI. Assigned by merchant. The merchant may use the customer’s NRIC number as the value of this field. | O   |
8 IPP Payment

Note:
To register an IPP via eNETS Online Transaction Mode, the following fields are to be populated.

### TxnReq.param1

<table>
<thead>
<tr>
<th>S/N</th>
<th>Field Name</th>
<th>Attributes</th>
<th>Length (Max)</th>
<th>Description</th>
<th>MOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Param1</td>
<td>A</td>
<td>3</td>
<td>Fixed: “IPP”</td>
<td>M</td>
</tr>
</tbody>
</table>

### TxnReq.param2 (The sub-fields to be constructed as JSON String)

<table>
<thead>
<tr>
<th>S/N</th>
<th>Sub-Field Name</th>
<th>Attributes</th>
<th>Length (Max)</th>
<th>Description</th>
<th>MOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>fullAmt</td>
<td>N</td>
<td>15</td>
<td>Full amount - Full purchase amount in cents, Eg 10000 implied $100</td>
<td>M</td>
</tr>
<tr>
<td>2</td>
<td>installmentPeriod</td>
<td>N</td>
<td>4</td>
<td>Installment period - In months, The period assigned to merchant during profile setup.</td>
<td>M</td>
</tr>
<tr>
<td>3</td>
<td>mthlyPaymt Amt</td>
<td>N</td>
<td>12</td>
<td>Monthly Payment amount in cents, Eg 10000 implied $100</td>
<td>M</td>
</tr>
<tr>
<td>4</td>
<td>downPaymt Amt</td>
<td>N</td>
<td>12</td>
<td>Down payment amount in cents, Eg 10000 implied $100</td>
<td>M</td>
</tr>
<tr>
<td>5</td>
<td>intRate</td>
<td>N</td>
<td>4</td>
<td>Interest rate can be defined between 0.00% to 99.99% per plan.</td>
<td>M</td>
</tr>
</tbody>
</table>
### intAmt

<table>
<thead>
<tr>
<th>S/N</th>
<th>Sub-Field Name</th>
<th>Attributes</th>
<th>Length (Max)</th>
<th>Description</th>
<th>MOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>intAmt</td>
<td>N</td>
<td>12</td>
<td>Total Interest amount in cents, Eg 10000 implied $100</td>
<td>M</td>
</tr>
</tbody>
</table>
9 Card Prefix

9.1.1 Supplementary Message (aka Card Prefix, KFC)

Supplementary message is optional. If required, it shall be submitted together with the Credit Payment Request.

<table>
<thead>
<tr>
<th>S/N</th>
<th>Field Name</th>
<th>Remark</th>
<th>Max Length</th>
<th>Payment Mode</th>
<th>Inclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Credit Browser</td>
<td>Credit Server</td>
</tr>
<tr>
<td>1</td>
<td>APC</td>
<td>Additional processing code. Allowed values: - 'CCPrefix'</td>
<td>20</td>
<td>C</td>
<td>O</td>
</tr>
</tbody>
</table>

Note: Allowed values shall be assigned by eNETS. Only alphanumeric values are allowed and no special characters allowed.

| 2   | Delimiter  | Fixed as '::' | 1   | C | O | O |
| 3   | Data       | If APC = "CCPrefix", This field shall contain first few digits of a credit card number up to a maximum of 8 digits. | 500 | C | O | O |

APC Field value equals to "CCPrefix"

If the eNETS Gateway receives a "CCPrefix" string in the APC field, it assumes that the Data field contains the first few digits of a credit card number up to a maximum of 8 digits. The eNETS Gateway populates the Data field’s value onto the eNETS Credit card Payment Page Credit Card Number field, and displays it on the browser. The Data field’s value populated on the eNETS Credit card Payment Page Credit Card Number field is NOT editable.
### Appendix A – Definitions

In this document, the terms listed below use the meanings set forth below:

<table>
<thead>
<tr>
<th><strong>Alternate Payment</strong></th>
<th>Anything outside the mainstream eDebit and eCredit payment is considered alternate payment.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Browser-To-Server (B2S)</strong></td>
<td>In B2S mode, the credit card capture page is presented by eNETS Gateway not by merchants.</td>
</tr>
<tr>
<td><strong>b2sTxnEnd</strong></td>
<td>This is the transaction status responded by eNETS Gateway via the browser redirection. b2sTxnEnd, though less reliable than s2sTxnEnd, is a backup transaction response sent by eNETS Gateway in case the s2sTxnEnd did not reach the merchant portal. Merchant does not need to wait for both s2sTxnEnd and b2sTxnEnd to arrive to confirm that the transaction is good. Any of the TxnEnd that arrives is considered good. eNETS will ensure both TxnEnds have the same value.</td>
</tr>
<tr>
<td><strong>Credit Card Capture Page</strong></td>
<td>This is the HTML page that captures the Credit Card number, Expiry and CVV.</td>
</tr>
<tr>
<td><strong>HMAC</strong></td>
<td>In the sample code, hmac label refers to the message digest of the payload concatenated with the SecretKey.</td>
</tr>
<tr>
<td><strong>IDD</strong></td>
<td>Internet Direct Debit which uses Bank’s Internet banking system.</td>
</tr>
<tr>
<td><strong>Mobile SDK</strong></td>
<td>Mobile Software Development Kit is a binary library that is supported on iOS and Android to help merchant to speed up their payment integration with NETS.</td>
</tr>
<tr>
<td><strong>MOC</strong></td>
<td>M=Mandatory, O=Optional, C=Conditional E.g. for Conditional, if clientType = S it is mandatory to specify mobile OS, else mobile OS is not required.</td>
</tr>
<tr>
<td><strong>NETS QR Code</strong></td>
<td>It is one of NETS payment mechanisms mainly for alternate payment such as DBS PayLah, OCBC PayAnyOne or UOB Mighty.</td>
</tr>
<tr>
<td><strong>NETSPay</strong></td>
<td>Refers to the contactless or digitized bank ATM cards used in NETS Payment Network.</td>
</tr>
<tr>
<td><strong>NETS Mobile SDK</strong></td>
<td>It is a mobile software development kit that a merchant can compile into their own merchant mobile native app. It can run on either iOS or Android.</td>
</tr>
<tr>
<td><strong>OTP</strong></td>
<td>One-Time Password is used for authentication.</td>
</tr>
<tr>
<td><strong>PAReq</strong></td>
<td>Payment Authentication Request is used in 3D transaction.</td>
</tr>
</tbody>
</table>
**eNETS Merchant Integration Guide eNETS Open API**

<table>
<thead>
<tr>
<th><strong>PARes</strong></th>
<th>Payment Authentication Response is used in 3D transaction.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Payer</strong></td>
<td>The person who is making the payment, or the buyer.</td>
</tr>
<tr>
<td><strong>Payment Method</strong></td>
<td>Examples of Payment Method are VISA/MC, Amex, eDebit-IDD, NETSPay, DBS PayLah, etc.</td>
</tr>
<tr>
<td><strong>Payment Option Page</strong></td>
<td>It is a HTML page that displays a list of payment methods for customer to choose from.</td>
</tr>
<tr>
<td><strong>Server-To-Server (S2S)</strong></td>
<td>In S2S mode, the credit card capture page is presented by the merchant.</td>
</tr>
<tr>
<td><strong>s2sTxnEnd</strong></td>
<td>This is the transaction status responded by eNETS Gateway directly from eNETS Gateway to Merchant Portal. It is more reliable than b2sTxnEnd. Merchant does not need to wait for both s2sTxnEnd and b2sTxnEnd to arrive to confirm transaction is good. Any of the TxnEnd that arrives is considered good. eNETS will ensure both TxnEnds have the same value.</td>
</tr>
<tr>
<td><strong>Source Of Funds (SOF)</strong></td>
<td>It refers to the source of the funding that will be used by payer to make payment. For example, eDebit, eCredit, uPOP, DBS PayLah, OCBC PayAnyOne and UOB Mighty, etc.</td>
</tr>
<tr>
<td><strong>TxnReq</strong></td>
<td>Transaction Request</td>
</tr>
<tr>
<td><strong>TxnRes</strong></td>
<td>Transaction Response</td>
</tr>
</tbody>
</table>
Appendix B – Payment Channels

Transaction Flows that are suitable for each payment channel

Q1. For payment via *merchant’s own mobile native application* use

![Merchant Mobile App Flow](image)

Transaction Flow 2 (Refer to section 5.2)

For native mobile applications.

SOF supported: Credit 3D, Credit Non-3D, eNETS Debit, NETSPay

Q2. For payment via a *merchant’s web portal* through a web PC browser use

![Web Browser Flow](image)

Transaction Flow 1 (Refer to section 5.1)

For web portals that users access from PC or Mobile via the browser.

![QR Related Flow](image)

Transaction Flow 3 (Refer to section 5.3)

For QR code wallets and transactions.

![Server to Server Flow](image)

Transaction Flow 4 (Refer to section 5.4)

For direct capturing of Credit Card data by Merchant.

SOF supported: Credit 3D, Credit Non-3D, eNETS Debit, NETSPay
Q3. For payment using a merchant’s web portal via a web PC browser or mobile browser use

<table>
<thead>
<tr>
<th>SOF / Transaction Flow</th>
<th>Integration Method 1: Web Browser (See section 3.1)</th>
<th>Integration Method 2: Merchant Mobile Native App (See section 3.2)</th>
<th>Integration Method 3: QR Code (See section 3.3)</th>
<th>Integration Method 4: Server-To-Server (S2S) (See section 3.4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>eCredit Non-3D</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>eCredit 3D</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>eDebit</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NETSPay</td>
<td></td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>DBS PayLah</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

SOF supported:
- Web Browser Flow: Credit 3D, Credit Non-3D, eNETS Debit
- Server to Server Flow: Credit 3D, Credit Non-3D

Appendix C – Source of Funds versus Transaction Flows
The table below provides a comprehensive view of the various SOFs and their supported transaction flow.
Appendix D – Message Format

**TxnReq**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Max Length</th>
<th>Description</th>
<th>MOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>netsMid</td>
<td>AN</td>
<td>20</td>
<td>Merchant ID. Uniquely identifies merchant. Value assigned by eNETS.</td>
<td>M</td>
</tr>
<tr>
<td>tid</td>
<td>NS</td>
<td>20</td>
<td>Terminal ID. Identifies the merchant terminal, if applicable, from where transaction originates.</td>
<td>O</td>
</tr>
<tr>
<td>clientType</td>
<td>A</td>
<td>1</td>
<td>S—SDK, W—WEB, M—Mobile Web. This field is only applicable for Transaction Flow 1, Transaction Flow 2, and Transaction Flow 3. It is not applicable for Transaction Flow 4.</td>
<td>C</td>
</tr>
<tr>
<td>mobileOs</td>
<td>A</td>
<td>16</td>
<td>If client type is &quot;S&quot;, ANDROID— for android OS, IOS – for IOS. This field is only applicable for Transaction Flow 1, Transaction Flow 2, and Transaction Flow 3. It is not applicable for Transaction Flow 4.</td>
<td>C</td>
</tr>
<tr>
<td>submission Mode</td>
<td>A</td>
<td>2</td>
<td>S – Server Submission. This is meant for Transaction Flow 4 only. B – Browser Submission. This is meant for the following transaction flows: Transaction Flow 1, Transaction Flow 2, Transaction Flow 3.</td>
<td>M</td>
</tr>
</tbody>
</table>
### Transaction Amount

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>txnAmount</td>
<td>N</td>
<td>15</td>
<td>Transaction amount in cents for all currencies. E.g. for $10.00 in SGD, the txnaAmount will be equal to 1000. For ¥10,000 in JPY, the txnAmount will be equal to 1000000.</td>
</tr>
<tr>
<td>currencyCode</td>
<td>A</td>
<td>3</td>
<td>Identifies the type of currency used. Three character currency code following the ISO 4217 Standard.</td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Length</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------</td>
<td>--------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>merchantTxnRef</td>
<td>AN</td>
<td>20</td>
<td>Unique reference code assigned by the merchant for this transaction. eNETS will reject duplicate merchantTxnRef. This is to prevent multiple submissions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For Credit Transactions: - For SALE and AUTH transactions, it must be a unique Merchant Reference as specified by the merchant storefront, or any ID that is used by the merchant to identify the transaction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For CAPTURE transactions, it must contain the same Merchant Reference corresponding to the prior AUTH transaction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For CREDIT transactions, it must contain the same Merchant Reference corresponding to the prior SALE or CAPTURE transaction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For QR Code Transactions:- For SALE, it must be a unique Merchant Reference as specified by the merchant storefront, or any ID that is used by the merchant to identify the transaction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For REFUND, it must contain the same Merchant Reference corresponding to the prior SALE transaction.</td>
</tr>
<tr>
<td>merchantTxDtm</td>
<td>NS</td>
<td>21</td>
<td>Format: yyyyMMdd HH:mm:ss.SSS Transaction date as recorded by merchant server. SSS - milliseconds.</td>
</tr>
<tr>
<td>merchantTimeZone</td>
<td>NS</td>
<td>6</td>
<td>Format : +8:00</td>
</tr>
<tr>
<td>paymentType</td>
<td>A</td>
<td>5</td>
<td>M</td>
</tr>
<tr>
<td>-------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>When paymentMode equals to DD “eNETS Debit” or QR “NETS QR Code”, the possible values of paymentType is: SALE – Sale This is where the money is deducted from the payer’s account immediately.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>When paymentMode equals to CC, “eNETS Credit Txn”, the possible values of paymentType are: SALE – Sale This is where the money is deducted from the cardholder’s credit account immediately. AUTH – Authorisation CAPT – Capture AUTH and CAPT is where the money is first reserved from the cardholder’s credit account upon successful Authorisation. The money is only deducted when the Merchant captures the Auth transaction. If the Auth transaction is not captured within a stipulated time (determined by bank), the reserved credit will be restored. CRED – Credit Is where the deducted money will be refunded back to the cardholder’s credit account? RAUTH – Reverse Authorisation RSALE – Reverse Sale RCRED – Reverse Credit</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**paymentMode**

| A | 2 | Payment Mode has 3 values: DD, CC or QR.  
If merchant ONLY wants to accept eNETS Debit as source of fund, set paymentType to SALE and set paymentMode to DD.  
If merchant ONLY wants to accept Credit Card adhere to the following conditions.  
a. If **Transaction Flow 1** (Web Browser) is used and if paymentType is AUTH or SALE set paymentMode to CC.  
b. If **Transaction Flow 4** (Server-To-Server) is used and if paymentType is any of the values: SALE, AUTH, CAPT, CRED, RAUTH, RSALE set paymentMode to CC.  
If merchant ONLY wants to accept NETS QR Code as source of fund, set paymentType to SALE and set paymentMode to QR. |

---

**bankId**

| N | 3 | <Available in future phase>  
This is applicable for eNETS Debit (i.e. paymentMode = DD) and merchant wants to specify a bankId which payer has last transacted. |

---

**netsMidIndicator**

| A | 1 | “U” for UMID |

---

**b2sTxnEndURL**

<p>| ANS | 80 | For <strong>Transaction Flow 1</strong>, <strong>Transaction Flow 3</strong> and <strong>Transaction Flow 4 (3D)</strong> only: eNETS will redirect to the merchant b2sTxnEndURL upon completion of transaction |</p>
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>b2sTxnEndURL</td>
<td>ANS</td>
<td>255</td>
<td>If b2sTxnEndURL is specified, eNETS will echo back the data during b2sTxnEnd else no data will be echoed back.</td>
<td>C</td>
</tr>
<tr>
<td>s2sTxnEndURL</td>
<td>ANS</td>
<td>80</td>
<td>For Transaction Flow 1, Transaction Flow 3 and Transaction Flow 4 (3D) only: eNETS will do an s2s call to the merchant server for this provided URL.</td>
<td>C</td>
</tr>
<tr>
<td>pan</td>
<td>N</td>
<td>19</td>
<td>For Transaction Flow 4 only. Credit Card Number For Server submission, this field is mandatory for SALE and for AUTH transactions only.</td>
<td>C</td>
</tr>
<tr>
<td>cvv</td>
<td>N</td>
<td>4</td>
<td>For Transaction Flow 4 only. Credit Card Validation Value. For Server submission, this field is mandatory for SALE and for AUTH transactions only.</td>
<td>C</td>
</tr>
<tr>
<td>expiryDate</td>
<td>N</td>
<td>4</td>
<td>For Transaction Flow 4 only. Credit Card Expiry Date (YYMM) For Server submission, this field is mandatory for SALE and for AUTH transactions only.</td>
<td>C</td>
</tr>
<tr>
<td>cardHolder Name</td>
<td>AN</td>
<td>255</td>
<td>For Transaction Flow 4 only. Name of Cardholder For Server submission, this field is mandatory for SALE and for AUTH transactions only.</td>
<td>C</td>
</tr>
<tr>
<td>cavv</td>
<td>ANS</td>
<td>255</td>
<td>For Transaction Flow 4 only and merchant using own MPI. Cardholder authentication Verification Value. Required when the value of Transaction Status is &quot;Y&quot; or “A”.</td>
<td>C</td>
</tr>
<tr>
<td>purchaseXiId</td>
<td>ANS</td>
<td>255</td>
<td>For <strong>Transaction Flow 4</strong> only and merchant using own MPI.</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>------</td>
<td>-----</td>
<td>----------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Transaction Identifier. Applicable for VBV transaction.</td>
<td></td>
</tr>
<tr>
<td>status</td>
<td>A</td>
<td>1</td>
<td>For <strong>Transaction Flow 4</strong> only and merchant using own MPI.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Indicates whether a transaction qualifies as an authenticated transaction.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Y = Authentication Successful. Customer was successfully authenticated. All data needed for clearing, including the Cardholder Authentication Verification Value, is included in the message.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>U = Authentication Could Not Be Performed. Authentication could not be completed, due to technical or other problems.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A = Attempts Processing Performed. Authentication could not be completed, but a proof of authentication attempt (CAVV) was generated.</td>
<td></td>
</tr>
<tr>
<td>eci</td>
<td>N</td>
<td>2</td>
<td>For <strong>Transaction Flow 4</strong> only and merchant using own MPI.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Electronic Commerce Indicator</td>
<td></td>
</tr>
<tr>
<td>param1</td>
<td>ANS</td>
<td>255</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>param2</td>
<td>ANS</td>
<td>255</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>param3</td>
<td>ANS</td>
<td>255</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>param4</td>
<td>ANS</td>
<td>255</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>
### GENERAL

<table>
<thead>
<tr>
<th></th>
<th>Ans</th>
<th>255</th>
<th>Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>param5</strong></td>
<td>ANS</td>
<td>255</td>
<td>Reserved</td>
</tr>
<tr>
<td><strong>supMsg</strong></td>
<td>ANS</td>
<td>255</td>
<td>Reserved</td>
</tr>
<tr>
<td><strong>language</strong></td>
<td>ANS</td>
<td>8</td>
<td><strong>en</strong> (Default – English), <strong>zh_cn</strong> (simplified Chinese)</td>
</tr>
<tr>
<td><strong>ipAddress</strong></td>
<td>NS</td>
<td>32</td>
<td><strong>Merchant server IP Address</strong></td>
</tr>
</tbody>
</table>
## Field Name | Field Type | Max Length | Description | MOC
--- | --- | --- | --- | ---
`netsMid` | AN | 20 | Echo from request | M
`tid` | NS | 20 | Echo from request. | O
`netsAmountDeducted` | N | 15 | The amount, in cents, deducted from payer. Set to zero for failed/incomplete transactions. Echo from `TxnReq.txnAmount` for successful transactions. | M
`currencyCode` | A | 3 | Echo from request. | M
`paymentType` | A | 5 | Echo from request. | M
`paymentMode` | A | 2 | If payer transaction is successful, the selected `paymentMode` will be returned. CC – Credit, DD – Debit, QR – QR Code. | C
`netsMidIndicator` | A | 1 | echo from request | M
`b2sTxnEndURLParam` | ANS | 255 | echo from request | C
`s2sTxnEndURLParam` | ANS | 255 | echo from request | C
`merchantTxnRef` | ANS | 20 | echo from request | M
`merchantTxnDtm` | NS | 21 | echo from request | M
`merchantTimeZone` | NS | 6 | echo from request | M
`netsTxnRef` | N | 17 | Unique reference code assigned by eNETS for this transaction. | M
`netsTxnDtm` | NS | 21 | Transaction date time as recorded by eNETS | M
<table>
<thead>
<tr>
<th>Field</th>
<th>Required</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>netsTxnStatus</td>
<td>NS</td>
<td>5</td>
<td>Status of the Transaction. 0: Transaction is successful 1: Transaction failed 5: Merchant to redirect payer browser to acsURL 9: Payer cancelled the transaction.</td>
</tr>
<tr>
<td>actionCode</td>
<td>N</td>
<td>1</td>
<td>The action code advises the next step to be taken. Please refer to Appendix E action codes.</td>
</tr>
<tr>
<td>stageRespCode</td>
<td>NS</td>
<td>13</td>
<td>The response code or error code returned by individual payment instruments. Please refer to Appendix E for the response codes.</td>
</tr>
<tr>
<td>netsTxnMsg</td>
<td>ANS</td>
<td>255</td>
<td>Transaction Message. E.g.: Approval for success transaction.</td>
</tr>
<tr>
<td>bankAuthId</td>
<td>N</td>
<td>6</td>
<td>Bank Authorisation code. For Credit Card transactions only (ie paymentMode = 'CC'). May be empty for failure cases.</td>
</tr>
<tr>
<td>maskPan</td>
<td>AN</td>
<td>19</td>
<td>For Credit Card transactions only (ie paymentMode = 'CC'). Masked PAN number,</td>
</tr>
<tr>
<td>txnRand</td>
<td>N</td>
<td>17</td>
<td>Unique reference code assigned by eNETS for this transaction.</td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Length</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>------</td>
<td>--------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>netsValueDate</td>
<td>NS</td>
<td>10</td>
<td>For Debit transactions only (ie paymentMode = ‘DD’) \nFormat: DD/MM/YYYY \nThe value (settlement) date for the transaction from an eNETS perspective. \nIf the transaction time (gw_date_time) is after the configured cutover time for the UMPlus endpoint (typically 23:00), then this field is set to be the next calendar date. \nPopulated from transaction-end message.</td>
</tr>
<tr>
<td>bankId</td>
<td>N</td>
<td>3</td>
<td>For Debit transactions only (ie paymentMode = ‘DD’) \nUnique identifier for the bank. \nPopulated from transaction-end message.</td>
</tr>
<tr>
<td>bankRefCode</td>
<td>N</td>
<td>17</td>
<td>For Debit transactions only (ie paymentMode = ‘DD’). \nUnique reference code assigned by eNETS for a transaction. Populated from transaction-end message.</td>
</tr>
<tr>
<td>eci</td>
<td>N</td>
<td>2</td>
<td>Provided when netsTxnStatus = 5. For Credit Card transactions only (ie paymentMode = ‘CC’) and <strong>Transaction Flow 4 (3D)</strong>. Electronic Commerce Indicator</td>
</tr>
<tr>
<td>pareq</td>
<td>ANS</td>
<td>-</td>
<td>Provided when netsTxnStatus = 5. For Credit Card transactions only (ie paymentMode = ‘CC’) and <strong>Transaction Flow 4 (3D)</strong>. PAreq value which is forwarded to the Access Control Server as part of the customer redirection.</td>
</tr>
<tr>
<td>termUrl</td>
<td>ANS</td>
<td>20-</td>
<td>Provided when netsTxnStatus = 5. For Credit Card transactions only (ie paymentMode = ‘CC’) and <strong>Transaction Flow 4 (3D)</strong>. termurl value (contains eNETS url which the customer gets directed to by the Access Control Server after 3D authentication).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Provided when netsTxnStatus = 5. For Credit Card transactions only (ie paymentMode = ‘CC’) and Transaction Flow 4 (3D). md value which is used by the MPI to track / reference a particular transaction.</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>md</td>
<td>AN</td>
<td>20-</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>acsUrl field value (contains customer’s Access Control)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>80-</td>
<td>Provided when netsTxnStatus = 5. For Credit Card transactions only (ie paymentMode = ‘CC’) and Transaction Flow 4 (3D). acsUrl field value (contains customer’s Access Control)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>param1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>param2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>param3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>param4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>param5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>param1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>param2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>param3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>param4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>param5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>param1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>param2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>param3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>param4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>param5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>param1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>param2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>param3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>param4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>param5</td>
</tr>
</tbody>
</table>
Appendix E – Response & Action Codes

The Action Code table below shows a list possible action merchant should take upon receiving the response codes from the Response Code table below. The action code can be found in Appendix D – Message Format, TtxnEnd section under the field name called **actionCode**. The actionCode advises the merchant on the next step to take after receiving a non-successful response code. The response code can be found in Appendix D – Message Format, TtxnEnd section under the field name called **stageRespCode**.

Here is how an error message for payer viewing is constructed. If netsTxnStatus = 1, concatenate (stageRespCode + netsTxnMsg) and display to payer.

- netsTxnStatus, stageRespCode, netsTxnMsg can be found in the field description of Appendix D under TtxnEnd Message Format section.
- stageRespCode contains the 4 digits (xxxx) stage code (eNETS Gateway internal code) and the 5 digits (yyyyy) response code in the form of xxxx-yyyyy
- netsTxnMsg contains the message to be displayed to payer

**Sample Error Messages**

If netsTxnStatus = 1, stageResponseCode="2100-69002", netsTxnMsg = “Time out, try again”, the error message to be shown to payer will be “2100-69002 Time out, try again.”

If netsTxnStatus = 1, stageResponseCode="2100-69003, netsTxnMsg = “System Error – Call Merchant”, the error message to be shown to payer will be “2100-69003 System Error - Contact Merchant.”

The first 4 digits 2100 is NETS internal error code meant to be used during troubleshooting.

Please refer to Appendix H FAQ for more detailed handling of Transaction Query and Timeout.

There are only 4 possible action codes as described below. Action code can be found in the field description of Appendix D under TtxnEnd Message Format section.

<table>
<thead>
<tr>
<th>Action Code</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No action required</td>
<td>There is no action required in respect to eNETS Gateway because the response is successful.</td>
</tr>
<tr>
<td>1</td>
<td>Retry</td>
<td>Merchant should decline the transaction and immediately request Payer to retry. Merchant needs to use a NEW merchant reference number.</td>
</tr>
</tbody>
</table>
2. **Perform Transaction Query**

Merchant should call Transaction Query when there is no response from eNETS Gateway to the transaction request or the error code requires merchant to perform query to check the status from eNETS Gateway. Please refer to Section 6 for the Transaction Query.

3. **Setup Error**

Merchant should decline the transaction. This error should not occur after merchant has tested successfully and has gone live. However, if some critical setup value is being wrongly modified, this error will appear. For example if merchant modified a production server MID or keyId by mistake, this error will appear.

<table>
<thead>
<tr>
<th>Response Code</th>
<th>Description (For Payer)</th>
<th>Remarks (For Merchant)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>Payment successful</td>
<td>Transaction is approved.</td>
</tr>
<tr>
<td>01000</td>
<td>Payment declined. Please contact card issuer.</td>
<td>Bank has declined the transaction with unknown reason.</td>
</tr>
<tr>
<td>01001</td>
<td>Payment declined. Please try again.</td>
<td>Bank has declined the transaction with unknown reason and indicated customer to try again.</td>
</tr>
<tr>
<td>01002</td>
<td>Payment declined. Please contact card issuer</td>
<td>Customer’s account has Insufficient Funds.</td>
</tr>
<tr>
<td>01003</td>
<td>Payment declined. Please contact Merchant.</td>
<td>Bank declined with Security Violation.</td>
</tr>
<tr>
<td>01004</td>
<td>Payment declined. Invalid amount.</td>
<td>Bank has declined the transaction because transaction amount is invalid. Check backend message.</td>
</tr>
<tr>
<td>01005</td>
<td>Payment declined. Invalid Account.</td>
<td>Bank has declined the transaction because customer does not have a Cheque Account.</td>
</tr>
<tr>
<td>01006</td>
<td>Payment declined. Invalid Account.</td>
<td>Bank has declined the transaction because customer does not have a Cheque Account.</td>
</tr>
<tr>
<td>01007</td>
<td>Payment declined. Invalid Account.</td>
<td>Bank has declined the transaction because customer does not have a Savings Account.</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
<td>Message</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>01010</td>
<td>Payment declined. Exceeded activity/pin-retry Limit</td>
<td>Bank has declined the transaction with unknown reason and indicated customer to try again.</td>
</tr>
<tr>
<td>01011</td>
<td>Payment declined. Exceeded Account Limit.</td>
<td>Bank has declined the transaction because customer exceeded their account limit.</td>
</tr>
<tr>
<td>01098</td>
<td>Payment declined. Please contact card issuer.</td>
<td>Bank has declined the transaction with unknown reason.</td>
</tr>
<tr>
<td>01099</td>
<td>Payment declined. Please contact card issuer.</td>
<td>Bank has declined the transaction with unknown reason.</td>
</tr>
<tr>
<td>01100</td>
<td>Payment declined. Please contact card issuer.</td>
<td>Customer has either entered wrong data or the account used does not exist.</td>
</tr>
<tr>
<td>01101</td>
<td>Payment declined. Invalid Credit Card Number.</td>
<td>Customer has either entered wrong data or the card used does not exist.</td>
</tr>
<tr>
<td>01102</td>
<td>Payment declined. Invalid Expiry Date.</td>
<td>Customer has either entered wrong data or the card used has expired.</td>
</tr>
<tr>
<td>01103</td>
<td>Payment declined. Expired card.</td>
<td>Customer has either entered wrong data or the card used has expired.</td>
</tr>
<tr>
<td>01104</td>
<td>Payment declined. Invalid CVV.</td>
<td>Customer has entered wrong CVV.</td>
</tr>
<tr>
<td>01105</td>
<td>Payment declined. Invalid data.</td>
<td>Customer has entered wrong credentials / pin.</td>
</tr>
<tr>
<td>01200</td>
<td>Payment declined. Please contact card issuer.</td>
<td>Bank has declined the transaction with unknown reason.</td>
</tr>
<tr>
<td>01201</td>
<td>Payment declined. Please contact card issuer.</td>
<td>Bank suspects Fraud Transaction.</td>
</tr>
<tr>
<td>01202</td>
<td>Payment declined. Please contact card issuer.</td>
<td>Card used is likely a Lost Card or Stolen Card.</td>
</tr>
<tr>
<td>01203</td>
<td>Payment declined. Please contact card issuer.</td>
<td>Card used is a Lost Card.</td>
</tr>
<tr>
<td>01204</td>
<td>Payment declined. Please contact card issuer.</td>
<td>Card used is likely a Stolen Card.</td>
</tr>
<tr>
<td>01301</td>
<td>Payment declined. Please contact card issuer.</td>
<td>Bank has declined the transaction because customer does not subscribe to 3D secure.</td>
</tr>
<tr>
<td>01302</td>
<td>Payment declined. Please contact card issuer.</td>
<td>Bank has declined the transaction because customer has failed 3D secure authentication.</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
<td>Reason</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>02000</td>
<td>Payment declined. Please contact Merchant.</td>
<td>Gateway has declined the transaction with unknown reason.</td>
</tr>
<tr>
<td>02001</td>
<td>Payment declined. Time out.</td>
<td>Gateway has declined the transaction because timeout on backend has occurred.</td>
</tr>
<tr>
<td>02002</td>
<td>Payment declined. User Session Expired.</td>
<td>Gateway has declined the transaction because user session has expired.</td>
</tr>
<tr>
<td>02003</td>
<td>Payment declined. User Cancelled Txn.</td>
<td>Gateway has declined the transaction because user cancelled the transaction.</td>
</tr>
<tr>
<td>02010</td>
<td>Payment declined. User Cancelled Txn.</td>
<td>User initiated cancellation and transaction was cancelled.</td>
</tr>
<tr>
<td>02200</td>
<td>Payment declined. Please contact card issuer.</td>
<td>Gateway has declined the transaction because issuer/bank related problems.</td>
</tr>
<tr>
<td>02201</td>
<td>Payment declined. Please contact card issuer.</td>
<td>Gateway has declined the transaction because card/account used is in blacklist or transaction has been flagged risky by fraud detection.</td>
</tr>
<tr>
<td>02800</td>
<td>Service unavailable. Please try again later.</td>
<td>Gateway service is unavailable.</td>
</tr>
<tr>
<td>02801</td>
<td>Service unavailable. Please try again later.</td>
<td>Bank indicated that the services are not available.</td>
</tr>
<tr>
<td>02850</td>
<td>Service unavailable. Please try again later.</td>
<td>Bank indicated that Recon Error. Try again later.</td>
</tr>
<tr>
<td>02851</td>
<td>System Error. Please contact Merchant.</td>
<td>Bank indicated that the Batch ID is not found.</td>
</tr>
<tr>
<td>02852</td>
<td>Service unavailable. Please try again later.</td>
<td>Bank indicated that Batch settlement is in progress. Try again later.</td>
</tr>
<tr>
<td>02853</td>
<td>System Error. Please contact Merchant.</td>
<td>Bank indicated that the Batch is not found.</td>
</tr>
<tr>
<td>03000</td>
<td>System Error. Please contact Merchant.</td>
<td>Gateway detected merchant setup is incorrect.</td>
</tr>
<tr>
<td>03001</td>
<td>System Error. Please contact merchant.</td>
<td>Gateway detected merchant setup is incorrect. Merchant config may be incomplete.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Description</td>
<td>Reason</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>03100</td>
<td>System Error. Please contact merchant.</td>
<td>Gateway failed transaction because merchant message is unpassable, malformed or incomplete.</td>
</tr>
<tr>
<td>03101</td>
<td>System Error. Please contact merchant.</td>
<td>Bank or other backend has rejected the transaction indicating Format Error. Check backend message used.</td>
</tr>
<tr>
<td>03102</td>
<td>System Error. Please contact merchant.</td>
<td>Gateway detected merchant encrypted fields in message cannot be decrypted. Merchant config may be incomplete. Check certificates and keys.</td>
</tr>
<tr>
<td>03103</td>
<td>System Error. Please contact merchant.</td>
<td>Gateway detected merchant message signature cannot be verified. Merchant config may be incomplete. Check certificates.</td>
</tr>
<tr>
<td>03110</td>
<td>System Error. Please contact Merchant.</td>
<td>Gateway detected merchant setup is incorrect. Merchant config may be incomplete, or merchant is not ‘Active’.</td>
</tr>
<tr>
<td>03111</td>
<td>System Error. Please contact Merchant.</td>
<td>Gateway detected merchant NETSMID is incorrect. Merchant config may be incomplete.</td>
</tr>
<tr>
<td>03112</td>
<td>System Error. Please contact Merchant.</td>
<td>Gateway detected merchant NETSUMID is incorrect. Merchant config may be incomplete.</td>
</tr>
<tr>
<td>03113</td>
<td>System Error. Please contact Merchant.</td>
<td>Bank or issuer detected invalid merchant ID. Merchant config may be incomplete.</td>
</tr>
<tr>
<td>03114</td>
<td>System Error. Please contact Merchant.</td>
<td>Bank or issuer detected invalid terminal ID. Merchant config may be incomplete.</td>
</tr>
<tr>
<td>03115</td>
<td>System Error. Please contact Merchant.</td>
<td>Gateway has detected that merchant reference is invalid.</td>
</tr>
<tr>
<td>03116</td>
<td>System Error. Please contact Merchant.</td>
<td>Gateway has detected that eNETS Txn reference is invalid.</td>
</tr>
<tr>
<td>03117</td>
<td>System Error. Invalid Amount.</td>
<td>Gateway has detected that transaction amount is invalid.</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
<td>Message Description</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>03118</td>
<td>System Error. Invalid Amount. Below transaction threshold amount.</td>
<td>Gateway has detected that transaction amount is below minimum allowed.</td>
</tr>
<tr>
<td>03119</td>
<td>System Error. Invalid Amount. Exceed transaction threshold amount.</td>
<td>Gateway has detected that transaction amount is exceeding maximum allowed.</td>
</tr>
<tr>
<td>03120</td>
<td>System Error. Please contact merchant.</td>
<td>Gateway has detected that transaction currency is invalid or currency does not match the merchant config.</td>
</tr>
<tr>
<td>03121</td>
<td>System Error. Please contact merchant.</td>
<td>Gateway has detected that transaction authorization code provided is invalid.</td>
</tr>
<tr>
<td>03200</td>
<td>System Error. Please contact merchant.</td>
<td>Bank or Gateway has rejected the transaction because merchant or cardholder is not allowed to perform the transaction. Check merchant profile.</td>
</tr>
<tr>
<td>03201</td>
<td>System Error. Please contact merchant.</td>
<td>Bank or Gateway has rejected the transaction indicating Invalid Transaction Data. This can be caused by improper format data.</td>
</tr>
<tr>
<td>03202</td>
<td>System Error. Please contact merchant.</td>
<td>Bank or Gateway has rejected the transaction indicating Inconsistent Data. This can be caused by invalid transaction state, or data from current message does not match data from previous states.</td>
</tr>
<tr>
<td>03203</td>
<td>System Error. Please contact merchant.</td>
<td>Bank or Gateway has rejected the transaction indicating Transaction Not Found.</td>
</tr>
<tr>
<td>03204</td>
<td>System Error. Please contact merchant.</td>
<td>Bank or Gateway has rejected the transaction indicating Transaction Not Allowed.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Description</td>
<td>Reason</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>03205</td>
<td>System Error. Please contact merchant.</td>
<td>Bank or Gateway has rejected the transaction indicating Duplicate Transaction has been observed. This can be caused by merchant re-using merchant transaction ref.</td>
</tr>
<tr>
<td>03240</td>
<td>System Error. Please contact merchant.</td>
<td>Transaction reversal has failed because of specified reasons.</td>
</tr>
<tr>
<td>03241</td>
<td>System Error. Please contact merchant.</td>
<td>Transaction reversal has failed because the original authorization transaction has not been found.</td>
</tr>
<tr>
<td>03242</td>
<td>System Error. Please contact merchant.</td>
<td>Bank has rejected the transaction indicating Reversal amount not matching Authorization amount.</td>
</tr>
<tr>
<td>03243</td>
<td>System Error. Please contact merchant.</td>
<td>Transaction has failed because it has already been reversed.</td>
</tr>
<tr>
<td>03244</td>
<td>System Error. Please contact merchant.</td>
<td>Transaction has failed because of an attempt to reverse a transaction that is in a different batch.</td>
</tr>
<tr>
<td>03245</td>
<td>System Error. Please contact merchant.</td>
<td>Transaction has failed because of an attempt to execute a transaction that is currently in progress.</td>
</tr>
<tr>
<td>03246</td>
<td>System Error. Please contact merchant.</td>
<td>Transaction has failed because of an attempt to complete the transaction using an amount greater than the original authorized amount.</td>
</tr>
<tr>
<td>03247</td>
<td>System Error. Please contact merchant.</td>
<td>Transaction has failed because of an attempt to refund using an amount greater than the original authorized amount.</td>
</tr>
<tr>
<td>03248</td>
<td>System Error. Please contact merchant.</td>
<td>Transaction has failed because of an attempt to refund on a previously refunded transaction.</td>
</tr>
<tr>
<td>03249</td>
<td>System Error. Please contact merchant.</td>
<td>Transaction has failed because of an attempt to refund on a previously captured transaction.</td>
</tr>
<tr>
<td>03250</td>
<td>System Error. Please contact merchant.</td>
<td>Transaction has failed because of an attempt to refund on a previously charged-back transaction.</td>
</tr>
<tr>
<td>Code</td>
<td>Error Description</td>
<td>Reason</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>03251</td>
<td>System Error. Please contact merchant.</td>
<td>Transaction has failed because of an attempt to act a locked transaction.</td>
</tr>
<tr>
<td>03252</td>
<td>System Error. Please contact merchant.</td>
<td>Transaction has failed because of an attempt to act a locked transaction.</td>
</tr>
<tr>
<td>03253</td>
<td>System Error. Please contact merchant.</td>
<td>Transaction has failed because of an attempt to cancel a transaction in an invalid state.</td>
</tr>
<tr>
<td>03600</td>
<td>System Error. Please contact merchant.</td>
<td>Gateway failed transaction because merchant is not subscribed to the service. Check merchant profile.</td>
</tr>
<tr>
<td>03601</td>
<td>System Error. Please contact merchant.</td>
<td>Gateway failed transaction because merchant is not subscribed to the service. Unsupported - Card Type.</td>
</tr>
<tr>
<td>03602</td>
<td>System Error. Please contact merchant.</td>
<td>Gateway failed transaction because it is unable to route to the bank or the bank is not supported. Unsupported - Issuer or Bank.</td>
</tr>
<tr>
<td>03603</td>
<td>System Error. Please contact merchant.</td>
<td>Gateway failed transaction because merchant is not subscribed to the service. Unsupported - 3D Secure.</td>
</tr>
<tr>
<td>03604</td>
<td>System Error. Please contact merchant.</td>
<td>Gateway failed transaction because merchant is not subscribed to the service. Unsupported - External 3D Secure.</td>
</tr>
<tr>
<td>09000 - 09999</td>
<td>System Error. Please contact merchant.</td>
<td>eNETS internal error has occurred.</td>
</tr>
<tr>
<td>50001 - 50499</td>
<td>System Error. Please contact merchant.</td>
<td>Gateway Internal Error</td>
</tr>
<tr>
<td>50185</td>
<td>System Error. Please contact merchant.</td>
<td>Duplicate Merchant Transaction Reference Number</td>
</tr>
<tr>
<td>50104</td>
<td>Payment declined. Transaction cancelled by user.</td>
<td>Transaction cancelled by user</td>
</tr>
<tr>
<td>50151</td>
<td>Payment declined. Time out, please try again.</td>
<td>Gateway Timeout</td>
</tr>
<tr>
<td>50152</td>
<td>Payment declined. Time out, please try again.</td>
<td>Gateway Timeout</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
<td>Error Type</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>68001</td>
<td>System Error. Please contact merchant.</td>
<td>Mobile SDK Setup error</td>
</tr>
<tr>
<td>69001</td>
<td>System Error. Please try again.</td>
<td>Network not available</td>
</tr>
<tr>
<td>69002</td>
<td>Payment declined. Time out, please try again.</td>
<td>Server Time out</td>
</tr>
<tr>
<td>69003</td>
<td>System Error. Please contact merchant.</td>
<td>Server Error</td>
</tr>
<tr>
<td>69004</td>
<td>System Error. Please contact merchant.</td>
<td>Invalid data request</td>
</tr>
<tr>
<td>69005</td>
<td>System Error. Please contact merchant.</td>
<td>Invalid data response</td>
</tr>
<tr>
<td>69021</td>
<td>System Error. Please contact merchant.</td>
<td>Service List empty</td>
</tr>
<tr>
<td>69022</td>
<td>System Error. Please contact merchant.</td>
<td>Invalid credential</td>
</tr>
<tr>
<td>69031</td>
<td>System Error. Please contact merchant.</td>
<td>Bank List loading error</td>
</tr>
<tr>
<td>69032</td>
<td>System Error. Please contact merchant.</td>
<td>Bank List redirect fail</td>
</tr>
<tr>
<td>69034</td>
<td>System Error. Please contact merchant.</td>
<td>Bank Login loading error</td>
</tr>
<tr>
<td>69035</td>
<td>System Error. Please contact merchant.</td>
<td>Bank login redirect fail</td>
</tr>
<tr>
<td>69036</td>
<td>System Error. Please contact merchant.</td>
<td>Debit flow response invalid</td>
</tr>
<tr>
<td>69041</td>
<td>System Error. Please contact merchant.</td>
<td>Fail to encrypt credit card</td>
</tr>
<tr>
<td>69042</td>
<td>System Error. Please contact merchant.</td>
<td>Invalid ACS URL</td>
</tr>
<tr>
<td>69043</td>
<td>System Error. Please contact merchant.</td>
<td>ACS Page loading fail</td>
</tr>
<tr>
<td>69044</td>
<td>System Error. Please contact merchant.</td>
<td>ACS Redirect fail</td>
</tr>
<tr>
<td>69045</td>
<td>System Error. Please contact merchant.</td>
<td>ACS response invalid</td>
</tr>
<tr>
<td>69046</td>
<td>System Error. Please contact merchant.</td>
<td>Non 3DS data response invalid</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
<td>Error Reason</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>69061</td>
<td>System Error. Please contact merchant.</td>
<td>QR data response invalid</td>
</tr>
<tr>
<td>69071</td>
<td>Payment declined. Please install NETSPay Application.</td>
<td>NETSPay not installed</td>
</tr>
<tr>
<td>69072</td>
<td>Payment declined. Please try again.</td>
<td>Invalid QR Code</td>
</tr>
<tr>
<td>69073</td>
<td>Payment declined. Please try again.</td>
<td>NETSPay not authenticate</td>
</tr>
<tr>
<td>69074</td>
<td>Payment declined. Please try again.</td>
<td>NETSPay transaction not success</td>
</tr>
<tr>
<td>69075</td>
<td>System Error. Please contact merchant.</td>
<td>NETSPay service not available</td>
</tr>
<tr>
<td>69076</td>
<td>System Error. Please contact merchant.</td>
<td>NETSPay internal error</td>
</tr>
</tbody>
</table>
Appendix F – Android Setup

eNETS Android SDK is provided as an AAR library “enetsLib-Release.aar”. Below are the steps that detail the setup for the SDK on android.

Followings permissions must be added in the merchant apps Android Manifest.

```xml
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission
    android:name="android.permission.INTERACT_ACROSS_USERS_FULL"
    android:protectionLevel="normal"/>
<uses-permission android:name="com.nets.netspay.QR_TRANSACTION"/>
```

Note: "android.permission.INTERACT_ACROSS_USERS_FULL" is essential to function the in app feature properly.

**Step 1:** After downloading the eNETS SDK .aar file, copy it to the project folder.

**Step 2:** Import the .aar file from the location it was saved, to the project as follows,

1. Go to the projects module settings,
2. Click plus icon as shown in the above screenshot.
3. Select “import jar/.aar” option as follows,
4. Next, select the .aar file location as shown below.

5. The imported .aar file will be shown in the list as follows. Click the plus sign on the right and select Module dependency.

6. From the list shown, choose the eNETS lib, and click ok.
7. Re-run the gradle build and then clean and re-build the project.

**Step 3:** Get the keyID and Secret Key.
The Key ID will be needed by the request header. The Secret Key is needed create the HMAC. The following shows how HMAC value is created.

HMAC must be created at the merchant server and passed it to the `sendPaymentRequest(…)` method.
The request is in the string representation of the JSON object. The string looks as follows,
String request = "\n" + 
  " \"as\": \"J\", \n" + 
  " \"msg\": \{\n" + 
  " \"netsMid\": \"UMID_878163000\", \n" + 
  " \"subscriptionMode\": \"B\", \n" + 
  " \"txnAmount\": \"900\", \n" + 
  " \"merchantTxnRef\": \"20170511 16:28:04:51\", \n" + 
  " \"merchantTxnDtm\": \"20170511 16:28:04:516\", \n" + 
  " \"paymentType\": \"SALE\", \n" + 
  " \"currencyCode\": \"SGD\", \n" + 
  " \"merchantTimeZone\": \"+8:00\", \n" + 
  " \"b2sTxnEndURL\": \"https://sit2.enets.sg/MerchantApp/rest/s2sTxnEnd\", \n" + 
  " \"s2sTxnEndURL\": \"https://sit2.enets.sg/MerchantApp/rest/s2sTxnEnd\", \n" + 
  " \"clientType\": \"S\", \n" + 
  " \"netsMidIndicator\": \"U\", \n" + 
  " \"ipAddress\": \"172.18.20.161\", \n" + 
  " \"language\": \"en\", \n" + 
  " \"ss\": \"i\", \n" + 
  " \"mobileOS\": \"ANDROID\", \n" + 
  " }\n+
\}"

String hmac = <To be created at the merchant server>

Send the payment service list request as follows,

```java
PaymentRequestManager manager = PaymentRequestManager.getSharedInstance();
try {
  manager.sendPaymentRequest(key, hmac, txn, new PaymentCallback() {
    @Override
    public void onResult(final PaymentResponse response) {
      if (response instanceof DebitCreditPaymentResponse) {
        final DebitCreditPaymentResponse debitCreditResponse = (DebitCreditPaymentResponse) response;
        Log.i(TAG, "Success: " + debitCreditResponse.txnResp + " HMAC " + debitCreditResponse.hmac + " Key ID " + debitCreditResponse.keyId);
        String txn_Information = debitCreditResponse.txnResp;
        String txn_HMAC = debitCreditResponse.hmac;
        String txn_KeyID = debitCreditResponse.keyId;
      } else if (response instanceof NonDebitCreditPaymentResponse) {
        final NonDebitCreditPaymentResponse nonDebitCreditResponse = (NonDebitCreditPaymentResponse) response;
        Log.i(TAG, "Success: " + nonDebitCreditResponse.status + " Application " + nonDebitCreditResponse.app);
        String txn_Status = nonDebitCreditResponse.status;
      }
    }
  }
  @Override
  public void onFailure(final NETSError error) {
    Log.e(TAG, "Transaction Failed: " + error.responeCode + " Action Code: " + error.actionCode);
  }
```

eNETS Merchant Integration Guide eNETS Open API
When the onResult callback method has been called, it will pass "PaymentResponse" object. If the request is related to credit or debit the PaymentResponse object must be casted to "DebitCreditPaymentResponse" and it contains key id, transaction information and HMAC. If the request is related to non-credit or debit, for example in app calling scenario, then the PaymentResponse object must be casted to "NonDebitCreditPaymentResponse". This object contains the response of the application being called. In case of a failure, it will pass a "NETSError" object which contains a response and action code.

```java
String txn_ResponseCode = error.responseCode;
String txn_ActionCode = error.actionCode;
}
}, context);
} catch (InvalidPaymentRequestException e) {
    Log.e("Test Main... ", e.getMessage());
} catch (Exception e) {
    Log.e("Test Main... ", e.getMessage());
}
```
Appendix G – iOS Setup

SDK Setup procedure

Project with Cocoa pod
Copy only eNETSLib.framework provided from SDK folder into project folder

Add frameworks in “Embedded Binaries” and “Linked Frameworks and Libraries”
Update podfile with following libraries. Run ‘pod install’ in project folder with terminal.

```ruby
# platform :ios, '9.0'

target 'eNETSMerchant' do

    use_frameworks!

    # Pods for eNETSMerchant
    pod 'Alamofire', '~> 4.4'
pod 'MBProgressHUD', '~> 1.6.8'
pod 'Cashen'
pod 'Presentr'
pod 'SwiftyJSON'
pod 'CryptoSwift'

end
```

**Project without Cocoa pod**

Copy following frameworks from SDK folder into project folder

- Alamofire framework
- Cashen.framework
- CryptoSwift.framework
- ENETSLib.framework
- MBProgressHUD.framework
- Presentr.framework
- SwiftyJSON.framework

Add frameworks in “Embedded Binaries” and “Linked Frameworks and Libraries”

<table>
<thead>
<tr>
<th>Embedded Binaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENETSLib.framework</td>
</tr>
<tr>
<td>Presentr.framework</td>
</tr>
<tr>
<td>CryptoSwift.framework</td>
</tr>
<tr>
<td>Cashen.framework</td>
</tr>
<tr>
<td>SwiftyJSON.framework</td>
</tr>
<tr>
<td>MBProgressHUD.framework</td>
</tr>
<tr>
<td>Alamofire.framework</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Linked Frameworks and Libraries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>ENETSLib.framework</td>
</tr>
<tr>
<td>Presentr.framework</td>
</tr>
<tr>
<td>CryptoSwift.framework</td>
</tr>
<tr>
<td>Cashen.framework</td>
</tr>
<tr>
<td>SwiftyJSON.framework</td>
</tr>
<tr>
<td>MBProgressHUD.framework</td>
</tr>
<tr>
<td>Alamofire.framework</td>
</tr>
</tbody>
</table>

Page 101
eNETS Merchant Integration Guide eNETS Open API

**Inter App communication Setup**

NETSPay app will callback to eNETS Mobile SDK for QR Transaction. That requires Merchant application to do setup as following.

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information Property List</td>
<td>Dictionary (18 items)</td>
<td></td>
</tr>
<tr>
<td>▼ LSApplicationQueriesSchemes</td>
<td>Array (1 item)</td>
<td>netspay</td>
</tr>
<tr>
<td>Item 0</td>
<td>String</td>
<td>netspay</td>
</tr>
<tr>
<td>▼ URL types</td>
<td>Array (1 item)</td>
<td>com.nets.enets.Demo</td>
</tr>
<tr>
<td>▼ URL Identifier</td>
<td>String</td>
<td>com.nets.enets.Demo</td>
</tr>
<tr>
<td>Item 0</td>
<td>String</td>
<td>Demo</td>
</tr>
<tr>
<td>Localization native development re...</td>
<td>String</td>
<td>en</td>
</tr>
<tr>
<td>Executable file</td>
<td>String</td>
<td>$(EXECUTABLE_NAME)</td>
</tr>
<tr>
<td>Bundle identifier</td>
<td>String</td>
<td>$(PRODUCT_BUNDLE_IDENTIFIER)</td>
</tr>
<tr>
<td>InfoDictionary version</td>
<td>String</td>
<td>6.0</td>
</tr>
<tr>
<td>Bundle name</td>
<td>String</td>
<td>$(PRODUCT_NAME)</td>
</tr>
<tr>
<td>Bundle OS Type code</td>
<td>String</td>
<td>APPL</td>
</tr>
<tr>
<td>Bundle versions string, short</td>
<td>String</td>
<td>1.2</td>
</tr>
<tr>
<td>Bundle version</td>
<td>String</td>
<td>11</td>
</tr>
</tbody>
</table>

Application need to update info.plist in project

- **URL types**
  - URL identifier = Bundle ID
  - URL scheme = ProductName

- **LSApplicationQueriesSchemes**
  - NETSPay

Note:

URL Scheme is required for eNETS to callback when transaction is completed. LSApplicationQueriesSchemes is to query whether NETSPay app is installed in the iOS device.
Objective-C project
Set to ‘Yes’ to “Always Embed Swift Standard Libraries” under Build Options.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embed Asset Packs in Product Bundle</td>
<td>No</td>
</tr>
<tr>
<td>Always Embed Swift Standard Libraries</td>
<td>Yes</td>
</tr>
<tr>
<td>CodeWarrior/MS-Style Inline Assembly</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Appendix H – FAQ & Exception Handling

1. **What timeout value should merchant set before sending the TxnReq, Transaction Request message to eNETS Gateway?**
   In consideration of the fact that a payment transaction flow has multiple trips between browser/phone app to servers for eCredit, eDebit and QR, a Payer in the eCommerce may take between 5 seconds to a few minutes depending on the payment option and how savvy the Payer is. In view of the various internal segmented timeout and timeout between the payment trips, in general we recommend the timeout value to be 5 minutes.

2. **What should merchant do when encountering time-out?**
   a) Merchant should perform 1st Transaction Query (see section 6) when Transaction Request encountered timeout.
   b) If netsTxnStatus has no status (see question 4), perform 2nd Transaction Query 5 seconds later.
   c) If after 2nd query netsTxnStatus still has no status, merchant can perform a 3rd transaction query 5 minutes later or login to eNETS Admin Portal to check the status or call NETS.

3. **Is it compulsory for merchant to process b2sTxnEnd since there is already a s2sTxnEnd response?**
   eNETS gateway sent two response messages to merchant, one directly to Merchant Server (s2sTxnEnd) and another via browser redirection (b2sTxnEnd). b2sTxnEnd is a backup response. It is advisable for merchant to process both s2sTxnEnd and b2sTxnEnd, in case merchant failed to receive s2sTxnEnd. Having said that, merchant can choose to ignore b2sTxnEnd if merchant can take a small risk to solely rely on s2sTxnEnd. However, merchant needs to wait for browser redirection back to merchant portal in order to push the payment result to the payer browser page.

4. **Is it possible for ‘Transaction Query’ to have no status?**
   Transaction Query will return the status that the eNETS Gateway has at that point in time depending on the state of the transaction. See Appendix D under the section TxnEnd, field netsTxnStatus for details. The status can be either
   a. Success – netsTxnStatus = 0
   b. Failure – netsTxnStatus = 1 (Failed), 9 (Payer cancelled the transaction)
   c. It is possible for netsTxnStatus to have no value because Gateway is still waiting for response from the banks or other source of funds and has no status at that
point in time. Merchant will have to login to eNETS Admin Portal to check the status or call NETS.

Below is a table specifying the supported Payment Type for Transaction Query. For unsupported Payment Type, please login to eNETS Admin Portal to check the status or call NETS.

<table>
<thead>
<tr>
<th>Payment Mode</th>
<th>Payment Type</th>
<th>Description</th>
<th>Txn Query Supported Y/N</th>
</tr>
</thead>
<tbody>
<tr>
<td>DD</td>
<td>SALE</td>
<td>Purchase</td>
<td>Y</td>
</tr>
<tr>
<td>QR</td>
<td>SALE</td>
<td>Purchase</td>
<td>Y</td>
</tr>
<tr>
<td>CC</td>
<td>SALE</td>
<td>Purchase</td>
<td>Y</td>
</tr>
<tr>
<td>CC</td>
<td>AUTH</td>
<td>Auth-Capture</td>
<td>Y</td>
</tr>
<tr>
<td>CC</td>
<td>CAPT</td>
<td>Auth-Capture</td>
<td>N</td>
</tr>
<tr>
<td>CC</td>
<td>CRED</td>
<td>Refund</td>
<td>N</td>
</tr>
<tr>
<td>CC</td>
<td>RAUTH</td>
<td>Reverse Auth</td>
<td>N</td>
</tr>
<tr>
<td>CC</td>
<td>RSALE</td>
<td>Reverse Sale (Void)</td>
<td>N</td>
</tr>
<tr>
<td>CC</td>
<td>RCRED</td>
<td>Reverse Refund</td>
<td>N</td>
</tr>
</tbody>
</table>

5. What if after a certain period of time merchant still does not get the notification from NETS Gateway and the Transaction Query returns In-Progress status?
You will only display SUCCESS and deliver your goods or services on one condition that is only when you receive a Response Code = 00000. If Transaction Query returns In-Progress, if merchant business flow allows, inform Payer that payment request has been received and a confirmation will be sent via email or sms.

6. For Method 2 Integration (Mobile App), can I depend on the transaction completion response status from mobile SDK to show SUCCESS to Payer?
The Transaction Complete status in Step 5 returned by Mobile SDK is just to acknowledge that payment request has been received but it has not gone to eNETS Gateway for processing yet. The dependable transaction response status must come from Step 6 or TxnQuery.
## Appendix I – Change History

<table>
<thead>
<tr>
<th>Version No.</th>
<th>Revision No.</th>
<th>Description</th>
<th>Approval Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00</td>
<td>Initial Release</td>
<td>17 Jul 2017</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>Section 3 - Added “Integration method”. Appendix C – Corrected the table, Banks' wallets will not be available from App-To-App in the first release.</td>
<td>20 Jul 2017</td>
</tr>
<tr>
<td>1</td>
<td>02</td>
<td>Appendix F Added support both Swift and Objective-C projects in eNETS SDK. New protocol update from NETSPay to eNETS SDK. (Support Multiple eNETS Merchant applications running on the same device)</td>
<td>29 Jul 2017</td>
</tr>
<tr>
<td>1</td>
<td>03</td>
<td>Section 5.1.1a &amp; 5.3.1a – added instruction to display eNETS payment selection option page in a separate page instead of embedded in Merchant web page. Section 5.1.1a, 5.2.1a, 5.3.1a, 5.4.1a Added &lt;Replace&gt; and &lt;default value1,2&gt; in the json message sample codes and reorganize the fields. Appendix D ipAddress is changed from Mandatory to Optional. Added permissions in Appendix F Android setup Replace the entire Appendix G iOS setup</td>
<td>13 Aug 2017</td>
</tr>
<tr>
<td>1</td>
<td>04</td>
<td>Section 5.1.1a, 5.2.1a, 5.3.1a, 5.4.1a there are 6 fields to &lt;Replace&gt; instead of 4 fields.</td>
<td>17 Aug 2017</td>
</tr>
<tr>
<td>Version No.</td>
<td>Revision No.</td>
<td>Description</td>
<td>Approval Date</td>
</tr>
<tr>
<td>------------</td>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>1</td>
<td>05</td>
<td>Section 5.1.1 Step4, 5.2.1 Step6, 5.3.1 Step 7, 5.4.1 Step 5, Step 8 – Added JSON sample string for txnRes.</td>
<td>23 Aug 2017</td>
</tr>
<tr>
<td>1</td>
<td>06</td>
<td>Section 5.2.1 Step 5 enhance the explanation of the dependable transaction response status. Added FAQ &amp; Exception Handling section in Appendix H.</td>
<td>30 Aug 2017</td>
</tr>
</tbody>
</table>
| 1          | 07           | • Section 2 – Amended Logo  
• Section 5.2.1 Step 3 – Enhance onResult of eDebit and eCredit for both Android and iOS.  
• Section 5.2.1 Step 6 – added s2sTxnEnd  
• Section 6.2 corrected the description of paymentType: Change Payment mode to Payment type.  
• Section 6.2 added netsTxnStatus  
• Appendix D – Enhance paymentMode and paymentType description.  
• Appendix E and H – Enhance the description of Action Code and FAQ.  
• Appendix F – added a new permission.  
• Replace consumer to payer for the entire doc.  
• Replace NETS 2.0 to NETSPay | 10 Sep 2017    |