

# Merchant Implementation Guide

## For

### Unified Merchant API (UMAPI)

	Name	Signature	
Author			
Date of Submission			
Reviewer 1	Chng Tiak Yee		
Date of Review 1			
Reviewer 2 (if applicable)			
Date of Review 2 (if applicable)			
Approved by			
Date of Approval			
Accepted by			
Date of Acceptance			

## Change History

### Java Version

Version No.	Revision No.	Description	Approval Date
1	0	Initial Document	23 Dec 2005
1	1	Section 2.3.2, Added "customAttribute" field to the Payment Response Message Section 4.1.11, password encryption in "NETSConfig.xml" Section 4.1.13, support for proxy server Section 4.1.14, importing of CA root certificate Section 4.1.9, installation and static registration of the "cryptix jce" and "openpgp" providers Section 2.3.1, Added fields param1, param2, param3, param4 and param5 to Payment Request Message Section 5, list of response codes	13 Apr 2006
1	2	Added new Section on using UMAPI for "RPP" transaction.	4 Oct 2006
1	3	Add sample codes for debit transactions & server-to-server response	14 Dec 2006
1	4	Add Debit Stage Code	18 Dec 2006
1	5	Amended samples codes for credit and debit txns.	20 Dec 2006
1	6	Amended section 2.3	04 Jan 2007
1	7	Amended section 2.3	11 Jan 2007
1	8	Added sample code for querytxnstatus	21 Mar 2007
1	9	Amended the direction of flow 2 & 3 in Section 2.2.2 Amended email from <a href="mailto:helpdesk@nets.com.sg">helpdesk@nets.com.sg</a> to <a href="mailto:enetsmerchantsupport@nets.com.sg">enetsmerchantsupport@nets.com.sg</a> in Section 1.5	30 Mar 2007
1	10	Added Staging URL in Section 4.1.2	02 Apr 2007
1	11	Amended section 2.3.4 and 4.2.1.2 for 3D Secure handling.	10 Apr 2007
1	12	Added in Appendix A & Reference document	20 Apr 2007

Version No.	Revision No.	Description	Approval Date
1	13	<p>Updated Section 2.2.2.2 – Updated descriptions for step 3 and 4.</p> <p>Updated Section 2.3.4 – Added 5 as a new status for netsTxnStatus field.</p> <p>Updated Section 2.3.1 – Updated description for field TxnAmount</p> <p>Updated Section 2.3.2 – Updated description for field TxnAmount</p> <p>Updated Section 2.3.4 – Updated description for field netsAmountDeducted</p> <p>Updated Section 2.3.5 – Updated description for field TxnAmount</p> <p>Updated Section 2.3.6 – Updated description for field TxnAmount</p> <p>Added in Section 6 – Added response code 1234</p>	06 Jul 2007
1	14	<p>Amended error codes in Section 6.</p> <p>Added in Section 6 – Added response code 1302</p>	
1	15	<p>Add the merchant-date-time, merchant code for debit pay request</p> <p>Add Note to identify notify and TxnEnd URL to be implemented in Debit Flow.</p> <p>Update 3-D transaction flow.</p> <p>Add the cancel status for the cancelling of transaction.</p>	May 2008
1	16	Added the .NET version of UMAPI	
1	17	<p>Update the NetsTxnRespCode field for credit response to length of 5</p> <p>Update the NetsTxnstatus field for credit response to length of 4</p> <p>To add the TxnAmount field in the debit response message</p> <p>Remove Debit success/failure/cancel/notify URLs in debitReq XML message in Appendix A</p>	25 August 2008
1	18	<p>To add to the sample code to read the TxnAmount for debit response. (Section 4.2.1.3.2)</p> <p>To remove comment of MerchantTimeZone, merchant_txn_date, and merchant_txn_time fields in the debit payment request message.(Section 4.2.1.3)</p> <p>To add two fields to submit paymentMode and submissionMode in debit payment request message format. (Section 2.3.5)</p>	31 August 2008
1	19	To add on the sample code on debit notification/acknowledge process to read the the notification message paramter from eNETS. (Section 4.2.1.3.3)	12 September 2008

Version No.	Revision No.	Description	Approval Date
1	20	<p>Section 2.3.8 Update the length of netsresponse code.</p> <p>Section 2.3.8 Update the nets-txn-msg not include in the during transaction end server-to-server</p> <p>Section 4.2.1.3.3 Update sample code to get txnAmount using getTxnSrcAmt();</p>	23 September
1	21	<p>Section 2.3.2 Update comment of the netstxnstatus and the maximum length to 5.</p> <p>Section 2.3.4 Update the comment on netsTxnRespCode.</p>	26 September
1	22	<p>Section 2.3.1 To add the field length for CAVV,purchaseXID,status,ECI</p> <p>Section 2.3.4 Update the comment on netsTxnRespCode. Added a new column to show the inclusion for browser-to-server and server-to-server transactions.</p> <p>Section 4.2.1.1.1 To add the sample code to set MerchantTxnDtm for Credit Payment Request</p> <p>Section 4.2.1.1.2 To add the sample code to get PaymentMode for Credit Payment Response</p> <p>Section 4.2.1.1.3 To add sample code for to get payment mode, submission-mode, txn-Amount, merchantTxnDtm,netsTxnDtm and netsTxnStatus.</p> <p>Section 6.1 Grouped the table per category of response codes.</p>	14 October
1	23	<p>Section 2.3.6 Debit Payment Notification Remove PaymentMode, SubmissionMode, netsTxnDtm. Add Dest_exchg_Rate,GW_Txn_Date,GW_Txn_Time, Bank_ID, Bank_Ref_Code,Bank_Status,Bank_Time_Zone,Bank_Txn_Date,Bank_Txn_Time,Bank_Remarks,Retry ,version and reserved paramters.</p> <p>Section 2.3.8 Debit Payment Response Rename netsTxnDate to GW_Txn_Date. Rename netsTxnTime to GW_Txn_time. Add Bank_time Zone, GW_Time_zone, Dest_Exchg_Rate, Dest_Curr_Code,Src_Txn_Amt,Src_Exchg_Rate, Src_Curr_Code, Retry , version and reserved parameters</p> <p>Section 4.2.1.3.2 To add sample code</p> <p>Section 4.2.1.3.3 To add sample code</p>	23 Nov 2008
1	24	<p>Section 2.3.7 To remove the netsCertID from acknowledge message</p>	

Version No.	Revision No.	Description	Approval Date
		Section 4.2.1.3.3 To remove the setting of netsCertID from sample code To add the code to set the merchantCertID	
1	25	Updated and changed manuals style and formatting Updated information on different areas Update credit acquirer response code "79" Updated date and time format	
1	26	Update the format of merchant transaction date in section 2.3.5	
1	27	Updated sections 2.3.6 (Field 21) and 2.3.8 (Fields 7 and 10) remarks for informational fields.	21 Dec 2010
1	28	Amended contact details in section 1.5. <ul style="list-style-type: none"> <li>Address from Motorway Building to Central Plaza.</li> <li>Email address from <a href="mailto:enetsmerchantsupport@nets.com.sg">enetsmerchantsupport@nets.com.sg</a> to <a href="mailto:info@nets.com.sg">info@nets.com.sg</a></li> <li>Added hotline number 65) 6274 1212 and operating hours.</li> </ul>	04 Mar 2011
1	29	<ul style="list-style-type: none"> <li>Updated section 2.3.6, remark column of field 12: Notification.netTxnStatus is an intermediate status and SHALL NOT be used by merchant for processing.</li> <li>Updated section 2.3.8, remark column of field 5: Field contains the final transaction status.</li> </ul>	11 Apr 2011
1	30	<ul style="list-style-type: none"> <li>Updated section 2.3.4, "Max Length" of "Pareq" field set to "-"</li> <li>Updated section 4.1.8 Installation of policy JARs for JDK 1.5 &amp; 1.6</li> </ul>	20 May 2011 08 June 2011
1	31	<ul style="list-style-type: none"> <li>Updated Table of content, section 8 is included.</li> <li>Updated section 2.3.1, added new fields to table related to RPP &amp; RECURRING</li> <li>Updated section 6, new eNETS credit response code 1117 is added.</li> <li>Added new section 8, Using UMAPI for Recurring transaction, note that section numbers of following this section are changed.</li> </ul>	07 Sept 2011
1	32	<ul style="list-style-type: none"> <li>Updated section 4.1.17, added 3 new fields TimeOut, logLevel, logFile, protocol which are to be added to configuration file "NetsConfig.xml"</li> </ul>	27 Oct 2011
1	33	<ul style="list-style-type: none"> <li>UMID changes (updates to Section 1.4, Section 4.1.12, Section 4.2.2, Section 5, Section 6, Section 6.1, Section 8 &amp; New Section 2.3.9)</li> </ul>	30 Nov 2011

---

**Documents and Reference Materials**

Reference Number	Document Name	Remarks
1	VXMLIS10	VISA Level III Specs

## TABLE OF CONTENTS

<b>1</b>	<b>INTRODUCTION .....</b>	<b>1</b>
1.1	Purpose and Objective .....	1
1.2	Audience.....	1
1.3	Assumptions / Exclusions.....	1
1.4	Terminology / Conventions.....	1
1.5	Comments / Feedback.....	1
1.6	Pre-requisites for UMAPI Integration .....	2
1.6.1	Java Version .....	2
1.6.2	.NET Version .....	2
<b>2</b>	<b>TECHNICAL OVERVIEW.....</b>	<b>3</b>
2.1	High level Features and Functionality.....	3
2.2	Transaction Flows .....	4
2.2.1	Credit Transaction using UMAPI Browser Submission.....	4
2.2.1.1	Exception / Reversal Scenarios .....	6
2.2.2	Credit Transaction using UMAPI Server Submission.....	7
2.2.2.1	Non 3D Flow .....	7
2.2.2.1.1	Merchants without website.....	9
2.2.2.2	3D Flow .....	10
2.2.3	UMAPI Debit Transaction Engine .....	13
2.3	Message Format.....	15
2.3.1	Credit Payment Request (From Merchant to eNETS) .....	15
2.3.2	Credit Payment Notification (From eNETS to Merchant).....	19
2.3.3	Credit Payment Acknowledgement (From Merchant to eNETS).....	20
2.3.4	Credit Payment Response (From eNETS to merchant) .....	22
2.3.5	Debit Payment Request (From Merchant to eNETS) .....	23
2.3.6	Debit Payment Notification (From eNETS to merchant) .....	23
2.3.7	Debit Payment Acknowledgement (From Merchant to eNETS).....	25
2.3.8	Debit Payment Response (From eNETS to merchant) .....	25
2.3.9	UMID Payment Request (From Merchant to eNETS) .....	27
<b>3</b>	<b>ARCHITECTURE .....</b>	<b>32</b>
3.1	UMAPI Components.....	32
3.2	Package Structure .....	33
3.2.1	Java Version .....	33
3.2.2	.NET Version .....	34
3.2.3	UMAPI Components.....	35
3.3	Supporting files.....	35
3.3.1	Java Version .....	35
3.3.2	.NET Version .....	35

<b>4</b>	<b>SETUP / DEPLOYMENT</b>	<b>36</b>
<b>4.1</b>	<b>API installation and setup</b>	<b>36</b>
4.1.1	Pre-requisites for deploying UMAPI	36
4.1.2	Information Provided to Merchant	36
4.1.2.1	Java Version	36
4.1.2.2	.NET Version	36
4.1.3	Hosting of Domain Name / Internet Connectivity	36
4.1.4	Server Setup	36
4.1.4.1	Java Version	36
4.1.4.2	.NET Version	36
4.1.5	Installation of JRE / JSSE	37
4.1.6	Setup Java Classpath	37
4.1.7	Installation of UMAPI	37
4.1.7.1	Java Version	37
4.1.7.2	.NET Version	38
4.1.8	Installation of policy files for Cryptix OpenPGP	39
4.1.9	Installation of JCE and OpenPGP providers for Cryptix	39
4.1.10	Generation of Public/Private key pair	40
4.1.10.1	Java Version	40
4.1.10.2	.NET Version	41
4.1.11	Encryption of Private key password	42
4.1.11.1	Java Version	42
4.1.11.2	.NET Version	42
4.1.12	Uploading of Public certificate to eNETS gateway	43
4.1.13	Configuration of proxy settings	43
4.1.14	Loading of CA root certificate for eNETS	43
4.1.15	Review of UMAPI Sample scripts	44
4.1.16	Customization of scripts by merchant	44
4.1.17	Configuration of UMAPI Config files	44
<b>4.2</b>	<b>Sample Scripts</b>	<b>45</b>
4.2.1	Credit Transactions	45
4.2.1.1	Java Version	45
4.2.1.1.1	Credit Payment Request	45
4.2.1.1.2	Credit Payment Response	47
4.2.1.1.3	Credit Payment Notification	48
4.2.1.2	.NET Version	48
4.2.1.2.1	Credit Card Server Request	48
4.2.1.2.2	Credit Card Server Post	52
4.2.1.2.3	Credit Card Server Notification/Acknowledgement	54
4.2.1.2.4	Credit Card Browser Request	56
4.2.1.2.5	Credit Card Browser Redirection	57
4.2.1.2.6	Credit Card Browser Post	60
4.2.1.2.7	Credit Card Browser Notification/Acknowledgement	61
4.2.1	Debit Transactions	63
4.2.1.3	Java Version	63
4.2.1.3.1	Debit Payment Request	63
4.2.1.3.2	Debit Payment Response	64
4.2.1.3.3	Debit Payment Notification/Acknowledgement	65
4.2.1.4	.NET Version	66
4.2.1.4.1	Debit Browser Request	66
4.2.1.4.2	Debit Browser Notification/Acknowledgement	68
4.2.1.4.3	Debit Browser Redirection	70
4.2.2	UMID Transactions	75
4.2.1.5	Java Version	75
4.2.1.5.1	UMID Payment Request	75
<b>5</b>	<b>RESPONSE CODES</b>	<b>79</b>
	<b>Credit Transaction Stages</b>	<b>79</b>



<b>5.1</b>	<b>Debit Transaction Stages .....</b>	<b>80</b>
<b>5.2</b>	<b>Credit Acquirer Response Codes .....</b>	<b>80</b>
<b>6</b>	<b>ENETS CREDIT RESPONSE CODES .....</b>	<b>81</b>
<b>6.1</b>	<b>Debit Acquirer Response Codes.....</b>	<b>83</b>
<b>7</b>	<b>USING UMAPI FOR RPP TRANSACTION .....</b>	<b>87</b>
<b>7.1</b>	<b>Integration Details.....</b>	<b>87</b>
7.1.1	Java Version sample code .....	87
7.1.2	.NET Version sample code.....	88
<b>7.2</b>	<b>RPI Action Message Format .....</b>	<b>88</b>
<b>8</b>	<b>USING UMAPI FOR RECURRING TRANSACTION .....</b>	<b>89</b>
<b>8.1</b>	<b>Integration Details.....</b>	<b>89</b>
8.1.1	Java Version sample code .....	89
<b>9</b>	<b>QUERY TXN STATUS .....</b>	<b>90</b>
<b>9.1</b>	<b>Sample Codes.....</b>	<b>90</b>
9.1.1	Java Version .....	90
9.1.2	.NET Version .....	92
9.1.2.1	eNETS Transaction Request/Response .....	92
9.1.2.2	Merchant Transaction Request/Response.....	93
<b>9.2</b>	<b>Output Format .....</b>	<b>94</b>
	<b>APPENDIX A MESSAGE XML SCHEMA.....</b>	<b>94</b>
<b>A.1</b>	<b>ROOT ELEMENT .....</b>	<b>94</b>
<b>A.2</b>	<b>CREDIT MESSAGE .....</b>	<b>94</b>
<b>A.3</b>	<b>DEBIT MESSAGE: .....</b>	<b>96</b>
<b>A.4</b>	<b>MESSAGE TYPE .....</b>	<b>98</b>
<b>A.5</b>	<b>VISA INVOICE (TAKEN FROM REFERENCE DOCUMENT 1): .....</b>	<b>108</b>

## 1 Introduction

### 1.1 Purpose and Objective

The main purpose of the UMAPI Merchant Implementation guide is to provide merchants with the necessary technical information when integrating their applications with eNETS II, using the **Unified Merchant API (UMAPI)**.

This manual contains a technical overview of the UMAPI, the payment transaction flows as well as the message formats required between eNETS II and the merchant system. This manual also contains a step-by-step setup instructions and sample scripts or java programs which merchants will find useful in their integration efforts with eNETS II.

### 1.2 Audience

The UMAPI Merchant Implementation Guide is intended as a technical guide for merchant developers and system integrators who are responsible for designing or programming the necessary online applications to integrate with eNETS II.

### 1.3 Assumptions / Exclusions

Merchant developers and system integrators are expected to have working knowledge of the Java 1.4 SDK platform, and Web / Application server technology. In addition, a basic understanding of the purpose / types of digital certificates is also essential.

### 1.4 Terminology / Conventions

Term	Description
Consumer	The buyer using the internet to make a payment.
Merchant	An entity that offer goods or services for sale on the internet.
UMAPI	Unified Merchant Application Programming Interface
URL	Uniform Resource Locator. Specifies the "address" of a resource in the internet
DSA	Secure Hash Algorithm.
Keystore	A special file that stores public and private keys used in <b>Public Key Infrastructure (PKI)</b> . A keystore file can be generated using a Java tool called "keytool".
Signing cert	This certificate can be obtained from any Certificate Authority. It is used to digitally sign a message.
SSL	Secure Socket Layer
OpenPGP	Non-proprietary protocol for encrypting data using public key cryptography. The OpenPGP protocol defines the standard formats for encrypted messages, signatures, and certificates for exchanging public keys
UMID	Universal Merchant ID

### 1.5 Comments / Feedback

Please feel free to address any comments or feedback that you may have on the UMAPI Integration manual by:

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

Email: [info@nets.com.sg](mailto:info@nets.com.sg)

Hotline:(65) 6274 1212  
Operating Hours:  
8:30am to 6:00pm Mondays - Fridays.  
Closed on Saturdays, Sundays and Public Holidays.

## 1.6 Pre-requisites for UMAPI Integration

### 1.6.1 Java Version

The UMAPI can run on any system that satisfies the following requirements:

**1. Hardware**

The Merchant API can run on any operating system that supports JRE 1.4 and above. A hard disk space of about 75MB for JRE and about 50MB for the UMAPI is required. Any machine with a 64MB RAM and above (memory) is capable of running the API.

**2. Operating System**

Merchants must have an operating system that supports the JAVA Virtual Machine. The UMAPI has been developed and tested based on Java version 1.4 (**Note:** *Linux has some known problems implementing some of the API provided by Sun. Until the problems are fixed, it is not recommended that merchants use Linux*).

**3. Web Server** (Applicable for internet transaction payments)

Any web server that supports SSL can be used. The web server needs to have a fixed domain name so that eNets can contact it any time.

**4. Application Server** (Optional for internet payments – e.g. if security requirements mandate separate web and application tiers)

Any application server that supports the Java Runtime Environment can be used. The application server needs to have a fixed domain name so that eNETS can contact it any time.

**5. Browsers**

The recommended browsers that consumers should use when making payment with eNETS over the internet is Microsoft Internet Explorer 5.0 and above, Netscape Communicator 4.7 and above with support for javascript and cookies.

### 1.6.2 .NET Version

The UMAPI.net can run on any system that satisfies the following requirements:

**1. Hardware**

The Merchant API can run on any operating system that supports Microsoft .NET Framework 1.1 and above.

**2. Operating System**

Merchant must have an operating system that supports Windows 2000 or Windows 2003.

**3. Browsers**

The recommended browsers that consumers should use when making payment with eNETS over the internet is Microsoft Internet Explorer 5.0 and above, Mozilla Firefox 2.0 with support for javascript and cookies.

## 2 Technical Overview

### 2.1 High level Features and Functionality

The eNETS Payment Gateway offers multiple online payment services to merchants, such as Credit Card Payment, and Direct Debit Payment. All these services can be accessed through a single interface, the Unified Merchant Interface, which is listening to https requests at [www.enets.sg/enets2/PaymentListener.do](https://www.enets.sg/enets2/PaymentListener.do).

Depending on the merchant type, merchants can send transaction requests to eNETS in one of the following ways:

1. **Direct HTTPS requests to <https://www.enets.sg/enets2/PaymentListener.do>**

Merchants without a website can submit payment requests to eNETS directly, without sending any parameters to the URL above. In this case, eNETS will display the Master Collection page to the consumer to select which merchant to pay to. The transaction flow is illustrated in section 2.2.3.

2. **Using the UMAPI**

The UMAPI is a component that is deployed at the merchant's storefront to pass transactional details to eNETS. It exposes interfaces for the merchant storefront application to call upon, helping to form/parse the various messages exchanged between the merchant and eNETS.

UMAPI gives merchant the flexibility of implementing their payment modules using one of these submission modes:

- a. **Browser Submission.**

The merchant supplies UMAPI with payment information, and gets a ready-to-send message from UMAPI. They perform their own http post, usually through a consumer browser redirection to eNETS.

- b. **Server Submission.**

Merchant supplies UMAPI with payment information, and UMAPI will form the message and establish a server-to-server connection to eNETS to send the message and receive the payment response.

All information exchange between the merchant storefront and the eNETS Payment gateway is in XML format. Please refer to Section 2.3 for more details.

The link from the merchant to eNETS is secured through the use of the OpenPGP protocol. The data is encrypted using eNETS Gateway public key, and merchants have to digitally sign the messages with their private certificate for non-repudiation.

## 2.2 Transaction Flows

### 2.2.1 Credit Transaction using UMAPI Browser Submission

If a consumer uses the internet to make payment using the Credit Card gateway, the flow described below will apply.

**Note:** In browser submission mode, the 3D Secure's processes happen "transparently" between eNETS2 and the consumer's browser.

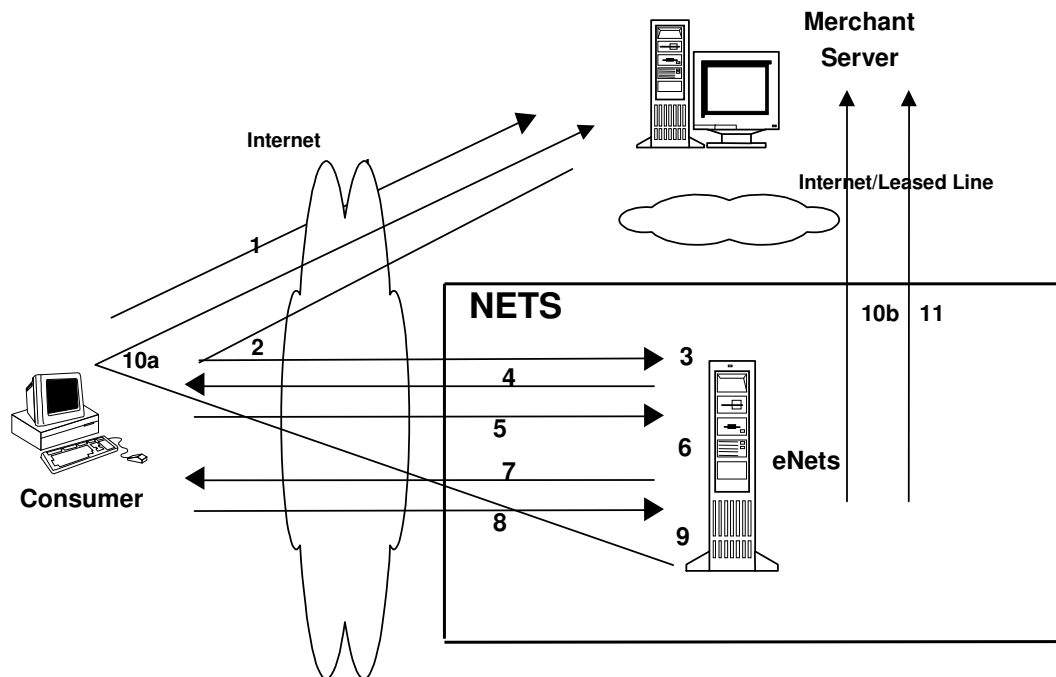
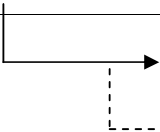


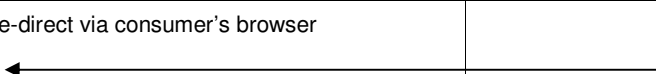

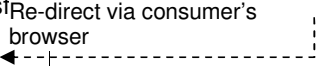


Figure 1. Transaction Flow for eNETS Credit Transaction (Browser Submission)

No.	Consumer	Merchant	eNETS Credit
1	The consumer browses to the merchant web site and makes a purchase and chooses to pay using eNETS Credit.		
2		The Merchant Server uses the UMAPI to form the Payment Request, and redirects the consumer to eNETS via customer's browser. <b>Note:</b> The merchant may pass the success, failure and cancel URL in the Payment Request message. If passed in, these will be used else the success, failure and cancel URL in the eNETS Credit merchant profile.	
3			eNETS decrypts the Payment Request message and verifies the merchant signature.
4			<ul style="list-style-type: none"> <li>eNETS verifies the parameters in the Payment Request and redirects the consumer to the Merchant Collection in the following cases:</li> <li>Merchant did not specify the Payment Mode.</li> <li>Merchant did not send the Transaction amount.</li> <li>If the merchant profile indicates that the merchant wants to publish his merchant collection page to gather additional information from the consumer.</li> </ul> <p>Else the flow continues with step 7</p>
5	Consumer fills in the required information at the merchant collection page and clicks on submit to proceed with payment.		
6			eNETS stores the gathered information in database.

No.	Consumer	Merchant	eNETS Credit
7			If the request does not contain the credit card details of the consumer, eNETS will re-direct the consumer to the Credit Payment Page. Else the flow continues with step 9.
8	The consumer enters the eNETS userID and password. They may login or enter their credit card information if they do not have an eNETS user ID and password. Consumer can choose to cancel at this point. Refer to step 12 for cancellation flow.	Re-direct via consumer's browser 	
9			eNETS Credit processes the transaction. 
10a			If the merchant POST feature is enabled in the merchant profile, eNETS will post the payment response to the POST URL provided by the merchant to eNETS. (This is transparent to consumer.)
10b		Merchant 's POST page receives the payment response	
11a			eNETS Credit sends the successful message to customer in a new pop-up window and redirects consumer to merchant URL.
11b		The merchant receives the Payment Successful message 	

### 2.2.1.1 Exception / Reversal Scenarios

12	If consumer chooses to cancel the transaction at Step 8, he will be redirected by eNETS Credit to the merchant's cancel URL. There is no post or notify message send to merchant for cancelling of transaction.
----	---

## 2.2.2 Credit Transaction using UMAPI Server Submission

### 2.2.2.1 Non 3D Flow

If a consumer uses the internet to make payment using the Credit Card gateway, the flow described below will apply:

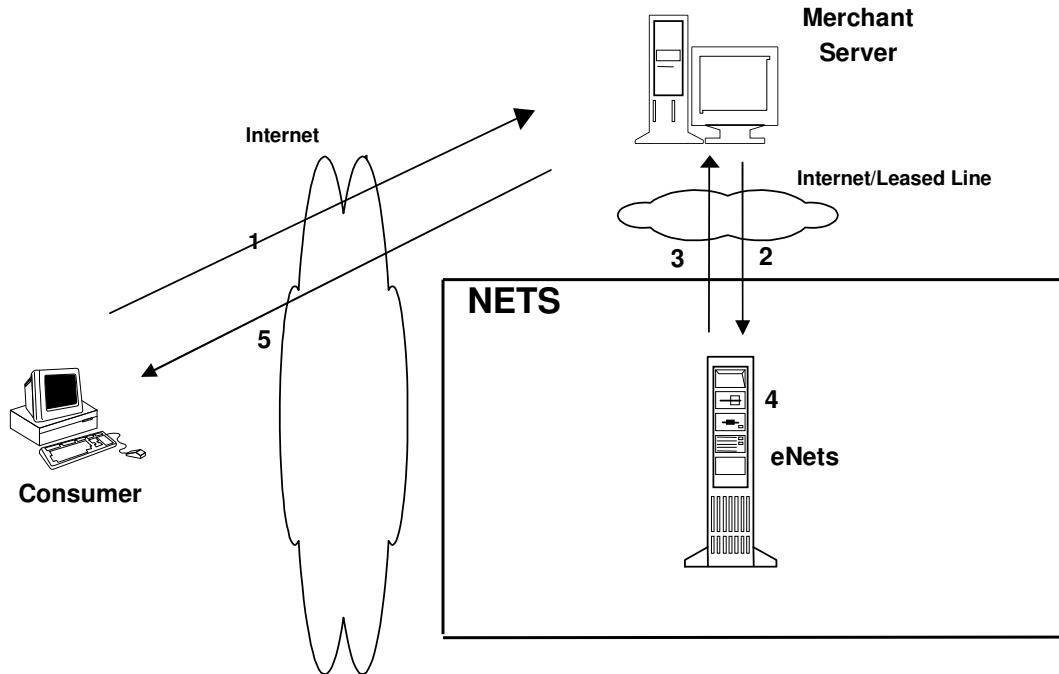



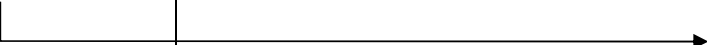


Figure 2. Transaction Flow for Credit Transaction (Server Submission)

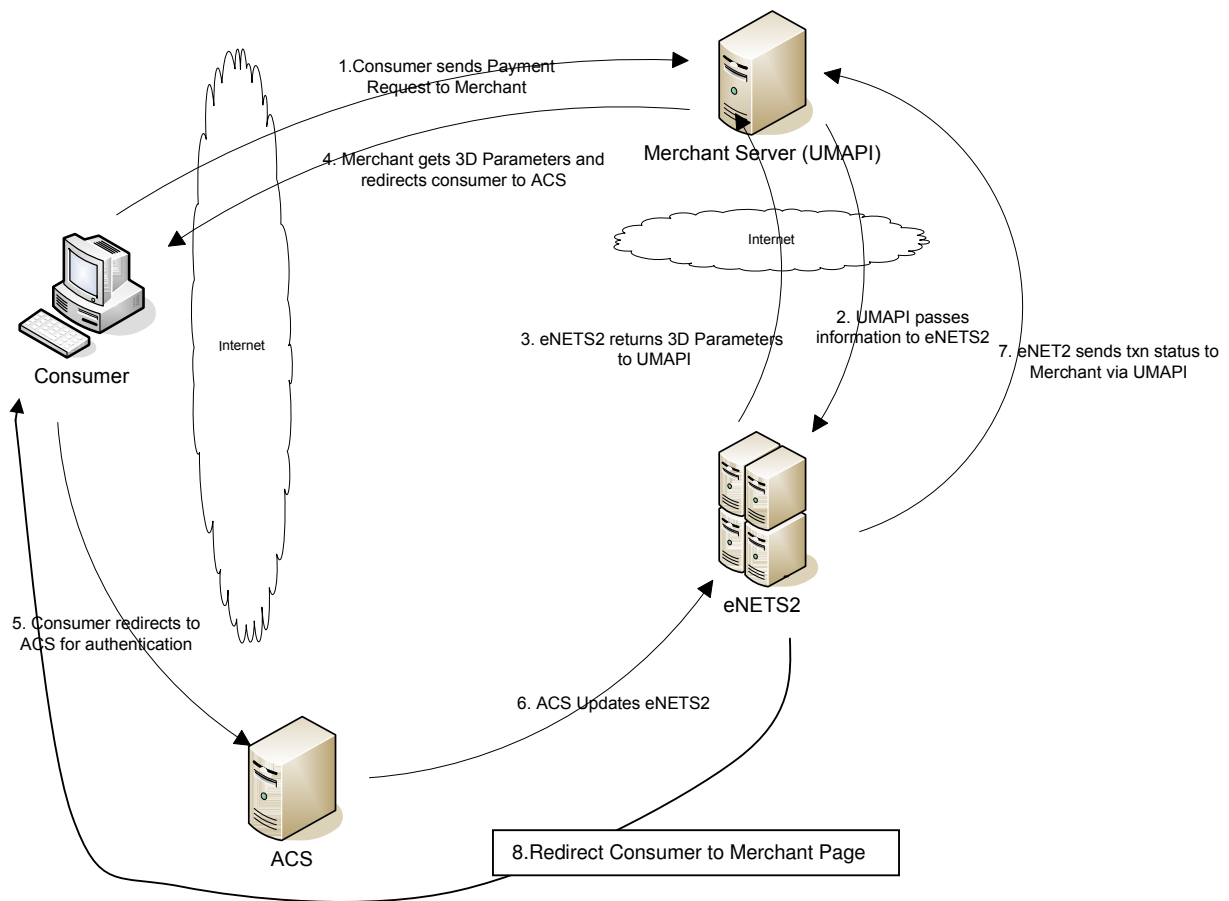


No.	Consumer	Merchant	eNETS Credit
1	The consumer browses the merchant web site and makes a purchase and chooses to pay using eNETS Credit.		
2		The Merchant Server uses UMAPI to form the Payment Request and send the request through a direct server-to-server connection to eNETS.	
3			eNETS decrypts the Payment Request message and verifies the merchant signature. eNETS Credit processes the transaction and returns the Payment Response message.
4		UMAPI decrypts the Payment Response, and verifies the eNETS signature. Based on the Payment response returned, the merchant updates its database and inventory information, and displays a receipt page to the consumer.	

**2.2.2.1.1 Merchants without website**

No.	Consumer	Merchant	eNETS
1	The consumer browses the eNETS web site and clicks the Master Collection page hyper link.		
2			eNETS brings out the master collection page to let consumer select the merchant to pay to.
3	The consumer selects the Merchant Category from the drop down list. The list of merchants under this category refreshes dynamically. The consumer then selects the merchant and clicks on submit.		
4			eNETS serves the Merchant Collection Page to the consumer to enter the transaction amount and select the Payment Mode.
5	The consumer enters the transaction amount and selects the Payment Mode (Credit)		
6			eNETS Credit processes the transaction and displays receipt page or transaction failed page to consumer upon completion of the transaction.

### 2.2.2.2 3D Flow



No.	Consumer	Merchant	eNETS	ACS
1	The consumer shops at the merchant site and submits payment details to the Merchant for processing.			
2		The merchant formulates payment details with UMAPI for submission to eNETS		
3			eNETS returns the 3D Parameters to UMAPI (see section 2.3.4 item 12 to 16). The 3D parameters are only returned if NETSTXNSTATUS = 5 and also normal non-3D flow will be applicable if NETSTXNSTATUS is not equal to 5.	
4		The merchant processes the 3D Parameters and redirects the consumer to the ACS server for authentication. The ACS' URL is contained in the field named "acsUrl" (item 15 in section 2.3.4). The merchant must generate a form that contains the variables "PaReq", "TermUrl" and "MD", and post them to the "acsURL". These values ("PaReq", "TermUrl" and "MD") are to be obtained from the fields named "PAreq", "termUrl", and "md" of section 2.3.4.		
5	Consumer gets redirect to ACS Server.			ACS Authenticates Consumer.
6			ACS updates eNETS on the authentication process.	
7		eNETS processes the transaction with the acquiring bank.		

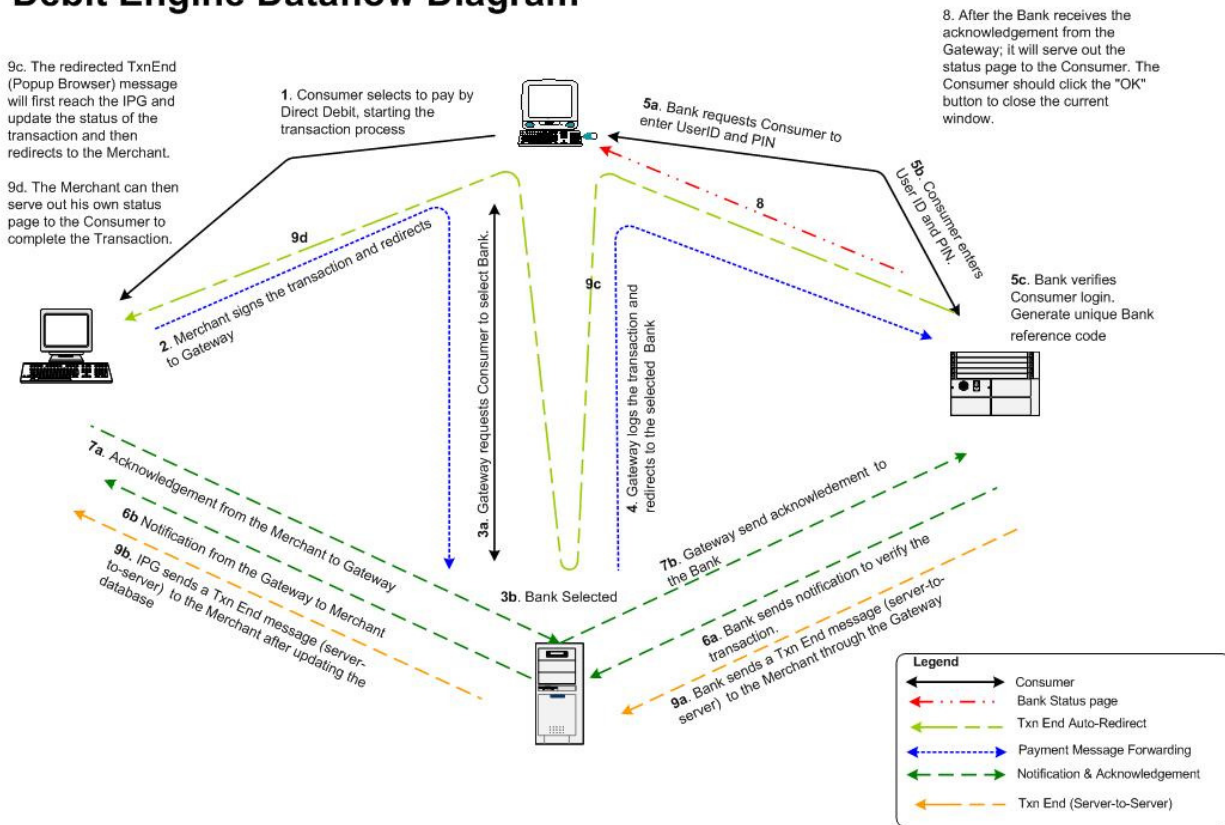
No.	Consumer	Merchant	eNETS	ACS
7a	The merchant received the transaction status ←	eNETS notifies the merchant of the transaction result. ←		
8	Merchant informs the consumer accordingly of the txn status through the success, failure URL. ←	eNETS redirect consumer to the merchant success or failure URL. ←		

## 2.2.3 UMAPI Debit Transaction Engine

The UMAPI for debit System is an internet-based direct-debit payment system that allows users to make payment over the World Wide Web. The Merchant server is probably a Web Server machine that communicates with the UMAPI Servers that in turn communicates with the Bank.

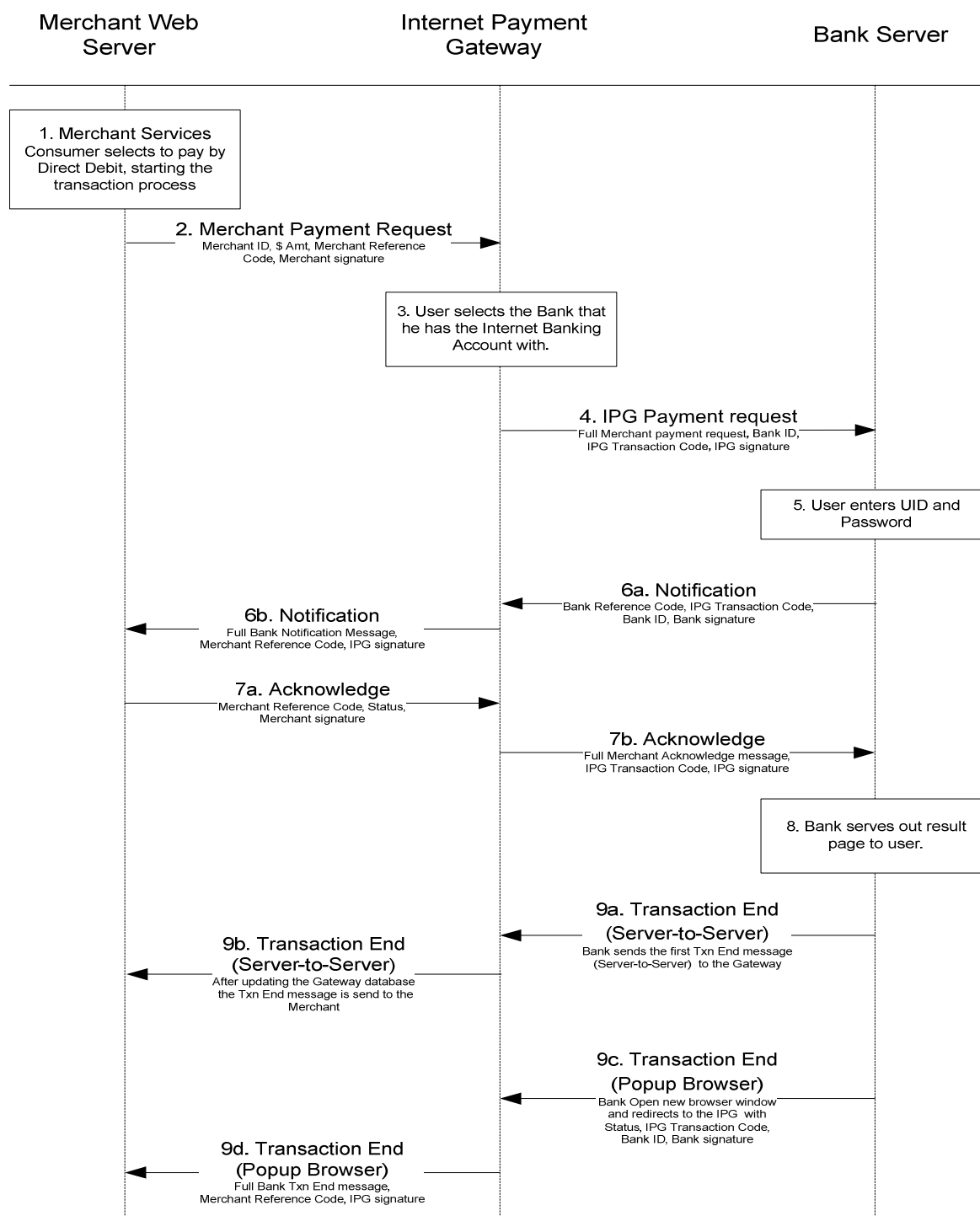
The UMAPI Debit Transaction Data Flow

### Debit Engine Dataflow Diagram



## The UMAPI Debit Transaction Web Flow

The web flow (and the message flow) of this application is shown in the following diagram. Each of the steps is thereafter described in greater detail.



**Note:** i) Merchant's notification URL (6b) to receive full bank notification message on step 6b.  
 ii) Merchant's TxnEnd2 (9b) URL to receive updated transaction status. [server-to-server]  
 iii) Merchant's TxnEnd (9d) URL to receive bank TxnEnd message.

## 2.3 Message Format

All messages exchanged between the merchant and eNETS are XML messages. They are signed and encrypted using openPGP, then posted to the eNETS II payment gateway as a parameter 'message'. Message formats are specified in XML schema. Refer to Appendix A for message schema and examples.

In the tables that follow, the Payment Mode and Inclusion columns include the values in the following table:

Field	Meaning
Payment Mode	Indicates which payment mode the field is applicable. C – Common to all payment modes CC – Credit Card Payment DD – Direct Debit Payment
Inclusion	R – Required C – Conditional O – Optional N – Not applicable

### 2.3.1 Credit Payment Request (From Merchant to eNETS)

The Payment Request message is sent by the merchant to eNETS to initiate internet credit card payment transactions.

**Note:**

- For Credit Card Authorisation transactions, there is a need to perform capture of the transactions to complete the transaction. Authorisation reserves the credit limit for a particular credit card while captures deducts the actual amount from the credit card. Authorised transactions that are not capturing are automatically released after a period of 1 week. (The issuing bank of the credit card decides the period.) Partial capturing is allowed. (i.e. Authorisation amount is \$800 but only captured \$600) Only a single capture is allowed for a single authorisation. (i.e. when authorisation amount is \$800. If \$600 had been captured, another capturing of \$200 is not allowed.) Capture Reversal is not allowed in eNETS gateway.

S/N	Field Name	Remark	Max Length	Payment Mode	Inclusion	
					Credit Browser	Credit Server
1	NetsMid	Merchant ID. Uniquely identifies merchant. Value assigned by eNETS.	20	C	R	R
2	Tid	Terminal ID. Identifies the merchant terminal, if applicable, from where transaction originates. If not present in the request, eNETS will set the Terminal ID to the IP address from where the request originates.	20	C	O	O
3	PaymentMode	Payment method of the transaction CC – Credit Card This field is required if the merchant does not publish his collection page. If not sent in the request, eNETS will redirect the consumer to the Merchant Collection Page to select the payment mode.	2	C	C	R
4	SubmissionMode	For Credit Payment S – Server Submission B – Browser Submission  For Direct Debit Payment B – Browser Submission	1	C	R	R



S/N	Field Name	Remark	Max Length	Payment Mode	Inclusion	
					Credit Browser	Credit Server
5	TxnAmount	Transaction amount in cents for all currencies. E.g. for \$10.00 in SGD, the TXN_AMT will be equal to 1000. For ¥10,000 in JPY, the TXN_AMT will be equal to 1000000.	10	C	R	R
6	CurrencyCode	Identifies the type of currency used. Three-character currency code following the ISO 4217 standard. If not set, defaults to SGD.	3	C	O	R
7	MerchantTxnRef	<p>Unique reference code assigned by the merchant for this transaction.</p> <p>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.</p> <p>eNETS will reject duplicate MERCH_REF. This is to prevent multiple submissions.</p> <p>For CAPTURE transactions, it must contain the same Merchant Reference corresponding to the prior AUTH transaction.</p> <p>For CREDIT transactions, it must contain the same Merchant Reference corresponding to the prior SALE or CAPTURE transaction.</p>	20	C	R	R
8	MerchantTxnDtm	Format: yyyyMMdd HH:mm:ss.SSS Transaction date as recorded by merchant server.	21	C	O	O
9	MerchantCertId	ID of the OpenPGP Certificate used for this transaction. It is assigned by eNETS after merchant generates his key pair and uploads the public certificate to the gateway.	5	C	R	R

S/N	Field Name	Remark	Max Length	Payment Mode	Inclusion	
					Credit Browser	Credit Server
10	PaymentType	Payment mode of credit card txn AUTH – Authorization CAPT – Capture AUTH and CAPT is where the money is first reserved from the cardholder's credit account upon successful Authorization. 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. SALE – Sale This is where the money is deducted from the cardholder's credit account immediately. CRED – Credit Is where the deducted money will be refunded back to the cardholder's credit account. RAUTH – Reverse Authorization R SALE – Reverse Sale RCREDIT – Reverse Credit	5	CC	R	R
11	Pan	Credit Card Number  For Server submission, this field is mandatory for SALE and AUTH transactions only.	19	CC	O	R
12	Cvv	Credit Card Validation Value	4	CC	O	O
13	ExpiryDate	Credit Card Expiry Date (YYMM)	4	CC	O	R
14	CardHolderName	Name of Cardholder	255	CC	O	O
15	SuccessUrl	eNETS will redirect to the merchant Success URL upon successful purchase  * Only applicable for browser submission, optional. If not sent in the request, SuccessUrl that was stored in the Merchant Profile at registration time will be used.	80	C	O	N
16	SuccessUrlParams	Query String to be appended to the Merchant Success URL, which displays the success page to shopper.  E.g. session_id=1111&member_id=23	255	C	O	N
17	FailureUrl	eNETS will redirect to this page when errors are encountered during payment.  *Only applicable for browser submission, optional. If not sent in the request, FailureUrl that was stored in the Merchant Profile at registration time will be used.	80	C	O	N

S/N	Field Name	Remark	Max Length	Payment Mode	Inclusion	
					Credit Browser	Credit Server
18	FailureUrlparams	Query String to be appended to the Merchant Failure URL, which displays error diagnostics information to shopper.	255	C	O	N
19	NotifyUrl	eNETS will send the Notify Message to this URL  *Only applicable for browser submission, optional. If not sent in the request, NotifyUrl that was stored in the Merchant Profile at registration time will be used.	80	C	O	N
20	NotifyUrlParams	Query String to be appended to the Merchant Notify URL.	255	C	O	N
21	PostUrl	eNETS will post the transaction information to the Merchant POST URL for logging and inventory update. This URL is not visible to the shopper	80	CC	O	O
22	PostUrlParams	Query String to be appended to the Merchant Post URL.	255	CC	O	O
23	CancelUrl	URL to redirect to when shopper clicks cancel during the payment process	80	CC	O	O
24	CancelUrlParams	Query String to be appended to the Merchant Cancel URL.	255	CC	O	O
25	Stan	Credit Note Number (must be unique for the same batch)  # Mandatory for CAPTURE and CREDIT transactions only.	6	CC	O	O
Fields 26 – 29 are applicable for 3D Secure transactions. Merchants who connect to a third-party MPI for credit card number authentication may send the authentication results to eNETS.						
26	Cavv	Cardholder authentication Verification Value. Required when the value of Transaction Status is “Y” or “A”	28	CC	C	C
27	purchaseXid	Transaction Identifier	20	CC	C	C
28	Status	Indicates whether a transaction qualifies as an authenticated transaction  Y = Authentication Successful. Customer was successfully authenticated. All data needed for clearing, including the Cardholder Authentication Verification Value, is included in the message.  N = Authentication Failed. Customer failed authentication. Transaction denied.  U = Authentication Could Not Be	1	CC	R	R

		Performed. Authentication could not be completed, due to technical or other problems.  A = Attempts Processing Performed. Authentication could not be completed, but a proof of authentication attempt (CAVV) was generated.				
29	ECI	Electronic Commerce Indicator	2	CC	C	C
30	Param1	Reserved Valid values: - "RPP" (see section 7) - "RECURRING" (see section 8)  Note: - Strings in uppercase - "RPP" for specific merchants only - "RECURRING" for merchants subscribed to generic recurring payment service.	100	CC	N	O
31	Param2	Reserved (for RPP, see section 7 below)	550	CC	N	O
32	Param3	Reserved	100	CC	N	O
33	Param4	Reserved	100	CC	N	O
34	Param5	Reserved	100	CC	N	O

### 2.3.2 Credit Payment Notification (From eNETS to Merchant)

S/N	Field Name	Remark	Max Length	Payment Mode	Inclusion	
					Credit Browser	Credit Server
1	NetsMid	Merchant ID. Uniquely identifies merchant. Value assigned by eNETS.	20	C	R	R
2	Tid	Terminal ID. Identifies the merchant terminal, if applicable, from where transaction originates. If not present in the request, eNETS will set the Terminal ID to the IP address from where the request originates.	20	C	O	O
3	PaymentMode	Payment method of the transaction CC – Credit Card This field is required if the merchant does not publish his collection page. If not sent in the request, eNETS will redirect the consumer to the Merchant Collection Page to select the payment mode.	2	C	C	R
4	SubmissionMode	S – Server Submission B – Browser Submission For Direct Debit payment, set to B.	1	C	R	R
5	TxnAmount	Transaction amount in cents for all currencies. E.g. for \$10.00 in SGD, the TXN_AMT will be equal to 1000. For ¥10,000 in JPY, the TXN_AMT will be equal to 1000000.	10	C	R	R
6	CurrencyCode	Identifies the type of currency used. Three-character currency code following the ISO 4217 standard.	3	C	O	R

		If not set, defaults to SGD.				
7	MerchantTxnRef	<p>Unique reference code assigned by merchant for this transaction.</p> <p>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.</p> <p>eNETS will reject duplicate MERCH_REF. This is to prevent multiple submissions.</p> <p>For CAPTURE transactions, it must contain the same Merchant Reference corresponding to the prior AUTH transaction.</p> <p>For CREDIT transactions, it must contain the same Merchant Reference corresponding to the prior SALE or CAPTURE transaction.</p>	20	C	R	R
8	MerchantTxnDtm	<p>Format: yyyyMMdd HH:mm:ss.SSS</p> <p>Transaction date as recorded by merchant server.</p>	21	C	O	O
9	MerchantCertId	ID of the OpenPGP Certificate used for this transaction. It is assigned by eNETS after merchant generates his key pair and uploads the public certificate to the gateway.	5	C	R	R
10	netsTxnRef	Unique reference code assigned by eNETS this transaction	17	C	R	3
11	netsTxnDtm	Transaction date time as recorded by eNETS	21	C	R	4
12	netsTxnStatus	<p>Status of the Transaction.</p> <p>0000: Transaction Successful</p> <p>05 : Do not honor (2 characters mean transactions rejected at bank end)</p> <p>-3093: Failed transaction within enets system.</p>	5	C	R	6

### 2.3.3 Credit Payment Acknowledgement (From Merchant to eNETS)

S/N	Field Name	Remark	Max Length	Payment Mode	Inclusion	
					Credit Browser	Credit Server
1	NetsMid	Merchant ID. Uniquely identifies merchant. Value assigned by eNETS.	20	C	R	R
2	MerchantTxnRef	Unique reference code assigned by merchant for this transaction.	20	C	R	2
3	MerchantTxnStatus	<p>Status of the Transaction:</p> <p>0000: Transaction Successful</p> <p>0001: Transaction Rejected</p> <p>0002: Use for Merchant Internal Error, Invalid message etc</p>	4	C	R	2
4	MerchantCertId	ID of the OpenPGP Certificate used for this transaction. It is assigned by eNETS after merchant generates his	5	C	R	R

		key pair and uploads the public certificate to the gateway.				
5	NetsTxnRef	Unique reference code assigned by eNETS this transaction	17	C	R	3

Note : MerchantTxnStatus of 0000 or any other value mean the merchant accepted the result of the transaction.  
MerchantTxnStatus of 0001 or 0002 will result in 'Approved' transaction being reversed.

### 2.3.4 Credit Payment Response (From eNETS to merchant)

eNETS redirects the consumer to the merchant's success or failure URL depending on the transaction status. As part of this redirection, the Payment Response message will be posted. In a server-to-server credit card transaction, the Payment Response is returned in the server response.

S/N	Field Name	Remark	Max Length	Payment Mode	Inclusion	
					Credit Browser	Credit Server
1	NetsMid	Merchant ID. Uniquely identifies merchant. Value assigned by eNETS.	20	C	R	R
2	merchantTxnRef	Unique reference code assigned by merchant for this transaction.	20	C	R	R
3	netsTxnRef	Unique reference code assigned by eNETS this transaction	17	C	R	R
4	netsTxnDtm	Transaction date time as recorded by eNETS	21	C	R	R
5	paymentMode	Payment method of the transaction CC – Credit Card	2	C	R	R
6	netsTxnStatus	Status of the Transaction. 0: Success, 2: Duplicate Success 1: Failure, 3: Duplicate Failure 5: Both merchant and card are 3D enrolled. For server-to-server flow, redirect consumer browser to acsURL. 9: Cancel	1	C	R	R
7	netsTxnRespCode	Response Code returned by individual payment instruments, or error codes. Refer to Section 5 for the list of Response Codes.  00: Success [Approved – 2 characters response code means the response is from bank] -1230 : Transaction has been cancelled. (Transaction terminated at eNETS for negative sign response code)	5	C	R	R
8	netsTxnMsg	Transaction Message. Eg :Approval for success transaction.	255	C	R	N
9	customAttribute	Only for merchants that publish their collection page. This field returns the data collected from the consumer.	-	C	O	O
10	bankAuthId	Bank Authorization code. For Credit Card transactions only. May be empty for failure cases.	6	CC	O	O
11	netsAmountDeducted	Amount Deducted in cents.	10	CC	R	R
12	Pareq	PAreq value which is forwarded to the Access Control Server as part of the consumer redirection	-	CC	R	R
13	TermUrl	termurl value (contains eNETS url which the consumer gets redirected to by the Access Control Server after 3D authentication)	20	CC	R	R
14	md	md value which is used by the MPI to track / reference a particular transaction	20	CC	R	R
15	acsUrl	acsurl field value (contains consumer's Access Control Server url)	20	CC	R	R
16	errorCode	errorcode value thrown by the MPI (If any) which will be forwarded to the Access Control Server	20	CC	R	R

**Note:** The values for S/N 12 – 16 are returned to and to be passed by the merchant if the credit card is 3D secure enrolled.

### 2.3.5 Debit Payment Request (From Merchant to eNETS)

S/N	Field Name	Remark	Max Length
1	NetsMid	Merchant ID. Uniquely identifies merchant. Value assigned by eNETS.	20
2	Tid	Terminal ID. Identifies the merchant terminal, if applicable, from where transaction originates. If not present in the request, eNETS will set the Terminal ID to the IP address from where the request originates.	20
3	TxnAmount	Transaction amount in cents for all currencies. E.g. for \$10.00 in SGD, the TXN_AMT will be equal to 1000. For ¥10,000 in JPY, the TXN_AMT will be equal to 1000000.	10
4	CurrencyCode	Identifies the type of currency used. Three-character currency code following the ISO 4217 standard. If not set, defaults to SGD.	3
5	MerchantTxnRef	Unique reference code assigned by merchant for this transaction.	20
6	MerchantCertId	ID of the Certificate used for this transaction. It is assigned by eNETS after merchant generates his key pair and uploads the public certificate to the gateway.	5
7	MerchantTimeZone	A <b>time zone</b> is a region of the Earth that has adopted the same standard time, usually referred to as the <b>local time</b> . MerchantTimeZone means the time zone which Merchant locates (e.g. "+11:30")	6
8	merchant_txn_date	Transaction Date (dd/mm/yyyy format)	10
9	merchant_txn_time	Transaction Time (hh:mm:ss format)	8
10	PaymentMode	Payment method of the transaction DD – Debit Card This field is required if the merchant does not publish his collection page. If not sent in the request, eNETS will redirect the consumer to the Merchant Collection Page to select the payment mode.	2
11	SubmissionMode	B – Browser Submission For Direct Debit payment, set to B.	1
12	Param1	Reserved	Defined by user
13	Param2	Reserved	Defined by user
14	Param3	Reserved	Defined by user
15	Param4	Reserved	Defined by user
16	Param5	Reserved	Defined by user

### 2.3.6 Debit Payment Notification (From eNETS to merchant)

S/N	Field Name	Remark	Max Length	Payment Mode	Inclusion
1	NetsMid	Merchant ID. Uniquely identifies merchant. Value assigned by eNETS.	20	C	R
2	Tid	Terminal ID. Identifies the merchant terminal, if applicable, from where transaction originates. If not present in the request, eNETS will set the Terminal ID to the IP address from where the request originates.	20	C	R



S/N	Field Name	Remark	Max Length	Payment Mode	Inclusion
3	TxnAmount	Transaction amount in cents for all currencies. E.g. for \$10.00 in SGD, the TXN_AMT will be equal to 1000. For ¥10,000 in JPY, the TXN_AMT will be equal to 1000000.	10	C	R
4	CurrencyCode	Identifies the type of currency used. Three-character currency code following the ISO 4217 standard. If not set, defaults to SGD.	3	C	R
5	Dest_Exchg_Rate	The exchange rate to convert the Txn Amount to Txn Base Amount.	12	C	R
6	MerchantTxnRef	Unique reference code assigned by merchant for this transaction.	20	C	R
7	MerchantCertId	ID of the Certificate used for this transaction. It is assigned by eNETS after merchant generates his key pair and uploads the public certificate to the gateway.	5	C	R
8	NetsTxnRef	Unique reference code assigned by eNETS this transaction	17	C	R
9	Gw_Txn_Date	Date on which transaction will be valued by Gateway.	10	C	R
10	GW_Txn_Time	Time of transaction as recorded by Gateway	8	C	R
11	GW_Time_Zone	Time Zone (relative to GMT) of the transaction date.	6	C	R
12	netsTxnStatus	Status of the Transaction. 0: Success 1 : Failure  Note: For NetsMid created after 11 <sup>th</sup> April 2011, this field contains an intermediate transaction status and <b>SHALL NOT</b> be used by merchants for processing. The final transaction status is obtainable from the 'netsTxnStatus' field in section 2.3.8 Debit Payment Response (From eNETS to merchant).	1	C	R
13	GW_Value_date	Date on which transaction will be valued by Gateway. Format is DD/MM/YYYY	10	C	R
14	Bank_ID	Uniquely identifies each Bank	4	C	R
16	Bank_Ref_Code	BANK reference code for this transaction	20	C	R
17	Bank_Status	Status of the transaction.	5	C	R
18	Bank_Time_Zone	Time Zone (relative to GMT) of the transaction	6	C	R
19	Bank_Txn_Date	Date at which total amount is debited from the Consumer account. Format :DD/MM/YYYY	10	C	R
20	Bank_Txn_Time	Time at which total amount is debited from the Consumer account.	8	C	R
21	Bank_Remarks	Free form text by BANK to be used for display or logging purpose.  Informational field and not for processing purposes.	30	C	R
22	Retry	The number of retries for current MerchantTxnRef.	4	C	R

23	Version	Software Version	10	C	O
24	Param1	Reserved	-	C	O
25	Param2	Reserved	-	C	O
26	Param3	Reserved	-	C	O
27	Param4	Reserved	-	C	O
28	Param5	Reserved	-	C	O
29	Param11	Reserved	-	C	O
30	Param12	Reserved	-	C	O
31	Param13	Reserved	-	C	O
32	Param14	Reserved	-	C	O
33	Param15	Reserved	-	C	O

**Note :** Merchant may choose to read only the necessary parameters to reference back to the transaction.

### 2.3.7 Debit Payment Acknowledgement (From Merchant to eNETS)

S/N	Field Name	Remark	Max Length
1	NetsMid	Merchant ID. Uniquely identifies merchant. Value assigned by eNETS.	20
2	MerchantTxnRef	Unique reference code assigned by merchant for this transaction.	20
3	MerchantTxnStatus	Status of the Transaction: 00000: success	5
4	MerchantCertId	ID of the OpenPGP Certificate used for this transaction. It is assigned by eNETS after merchant generates his key pair and uploads the public certificate to the gateway.	5
5	netsTxnRef	Unique reference code assigned by eNETS this transaction	17
6	Param1	Reserved	Defined by user
7	Param2	Reserved	Defined by user
8	Param3	Reserved	Defined by user
9	Param4	Reserved	Defined by user
10	Param5	Reserved	Defined by user
11	Retry	Retry Times	4
12	Version	Software Version	10

### 2.3.8 Debit Payment Response (From eNETS to merchant)

S/N	Field Name	Remark	Max Length
1	netsMid	Merchant ID. Uniquely identifies merchant. Value assigned by eNETS.	20
2	merchantTxnRef	Unique reference code assigned by merchant for this transaction.	20
3	netsTxnRef	Unique reference code assigned by eNETS this transaction	17
4	Tid	Terminal ID. Identifies the merchant terminal, if applicable, from where transaction originates. If not present in the request, eNETS will set the Terminal ID to the IP address from where the request originates.	20
5	netsTxnStatus	Status of the Transaction. 0: Success 1: Failure 9: cancel  Note: This field contains the final transaction status and <b>SHALL BE</b> used by the merchants for processing.	1
6	netsTxnRespCode	Response Code returned by individual payment instruments, or error codes. Refer to Section 6.1 for	13

		the list of Response Codes.  This is composed of 2 codes separated by “_0”, the first part is Transaction Stage Code and the second part is the Debit Acquirers Response Code. E.g.: 000008_000000 (successful transaction)	
7	netsTxnMsg	Transaction Message. (Refer to section 6.1)  This parameter is not applicable during transaction end server to server message.  Informational field and not for processing purposes.	255
8	BANK_ID	Assigned Bank Code	4
9	BANK_Ref_Code	Bank Reference Code for transaction.	20
10	BANK_Remarks	Remarks from bank.  Informational field and not for processing purposes.	30

S/N	Field Name	Remark	Max Length
11	BANK_Txn_Date	Date where transaction amount is debited. Format: dd/mm/yyyy	10
12	BANK_Txn_Time	Time where transaction amount is debited .Format: hh:mm:ss	8
13	BANK_Time_Zone	Time Zone (relative to GMT) of the transaction	6
14	BANK_Status	Bank status of transaction	5
15	GW_Txn_Date	Date on which transaction will be valued by Gateway. Format: dd/mm/yyyy	10
16	GW_Txn_Time	Time of transaction as recorded by Gateway. Format: hh:mm:ss	8
17	GW_Time_Zone	Time Zone (relative to GMT) of the transaction date.	6
18	GW_Value_date	Date on which transaction will be valued by Gateway. Format is DD/MM/YYYY	10
19	TxnAmount	Transaction amount (destination amount)	10
20	Dest_Exchg_Rate	Destination exchange rate	12
21	Dest_Curr_Code	Destination currency code	3
22	Src_Txn_Amt	Transaction amount from source	10
23	Src_Exchg_Rate	Source exchange rate	12
24	Src_Curr_Code	Source currency code	3
25	Retry	The number of retries for current MerchantTxnRef.	4
26	version	Software version	10
28	Param1	Reserved	-
29	Param2	Reserved	-
30	Param3	Reserved	-
31	Param4	Reserved	-
32	Param5	Reserved	-
33	Param11	Reserved	-
34	Param12	Reserved	-
35	Param13	Reserved	-
36	Param14	Reserved	-
37	Param15	Reserved	-

**Note :** Merchant may choose to read only the necessary parameters to reference back to the transaction.

### 2.3.9 UMID Payment Request (From Merchant to eNETS)

The UMID Payment Request should contain both Credit Payment Request and Debit Payment Request.

**Note:**

- For UMID transaction, the consumer can opt for Credit or Debit service for payment. Thus it is required for merchant to send credit & debit payment requests in order to process UMID transaction.

S/N	Field Name	Remark	Max Length	Payment Mode	Inclusion	
					Credit Browser	Credit Server
1	NetsMid	UMID (Universal Merchant ID). Uniquely identifies merchant. Value assigned by eNETS.	20	C	R	R
2	Tid	Terminal ID. Identifies the merchant terminal, if applicable, from where	20	C	O	O

		transaction originates. If not present in the request, eNETS will set the Terminal ID to the IP address from where the request originates.				
3	PaymentMode	Payment method of the transaction This field can be set to "Empty" since the payment mode is not determined during the payment request.	2	C	C	R
4	SubmissionMode	For UMID Payment S – Server Submission B – Browser Submission	1	C	R	R
5	TxnAmount	Transaction amount in cents for all currencies. E.g. for \$10.00 in SGD, the TXN_AMT will be equal to 1000. For ¥10,000 in JPY, the TXN_AMT will be equal to 1000000.	10	C	R	R
6	CurrencyCode	Identifies the type of currency used. Three-character currency code following the ISO 4217 standard. If not set, defaults to SGD.	3	C	O	R
7	MerchantTxnRef	Unique reference code assigned by the merchant for this transaction.  For UMID Transactions: - For SALE 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.  eNETS will reject duplicate MERCH_REF. This is to prevent multiple submissions.	20	C	R	R
8	MerchantTxnDtm	Format: yyyyMMdd HH:mm:ss.SSS Transaction date as recorded by merchant server.	21	C	O	O
9	MerchantCertId	ID of the OpenPGP Certificate used for this transaction. It is assigned by eNETS after merchant generates his key pair and uploads the public certificate to the gateway.	5	C	R	R

S/N	Field Name	Remark	Max Length	Payment Mode	Inclusion	
					Credit Browser	Credit Server
10	PaymentType	Payment mode of UMID txn SALE – Sale This is where the money is deducted from the cardholder's credit account immediately.  Note: Sale is only supported for UMID.	5	CC	R	R
11	Pan	Credit Card Number  For Server submission, this field is mandatory for SALE and AUTH transactions only.	19	CC	O	R
12	Cvv	Credit Card Validation Value	4	CC	O	O
13	ExpiryDate	Credit Card Expiry Date (YYMM)	4	CC	O	R
14	CardHolderName	Name of Cardholder	255	CC	O	O
15	SuccessUrl	eNETS will redirect to the merchant Success URL upon successful purchase  * Only applicable for browser submission, optional. If not sent in the request, SuccessUrl that was stored in the Merchant Profile at registration time will be used.	80	C	O	N
16	SuccessUrlParams	Query String to be appended to the Merchant Success URL, which displays the success page to shopper.  E.g. session_id=1111&member_id=23	255	C	O	N
17	FailureUrl	eNETS will redirect to this page when errors are encountered during payment.  *Only applicable for browser submission, optional. If not sent in the request, FailureUrl that was stored in the Merchant Profile at registration time will be used.	80	C	O	N
18	FailureUrlparams	Query String to be appended to the Merchant Failure URL, which displays error diagnostics information to shopper.	255	C	O	N
19	NotifyUrl	eNETS will send the Notify Message to this URL  *Only applicable for browser submission, optional. If not sent in the request, NotifyUrl that was stored in the Merchant Profile at registration time will be used.	80	C	O	N
20	NotifyUrlParams	Query String to be appended to the Merchant Notify URL.	255	C	O	N

21	PostUrl	eNETS will post the transaction information to the Merchant POST URL for logging and inventory update. This URL is not visible to the shopper	80	CC	O	O
22	PostUrlParams	Query String to be appended to the Merchant Post URL.	255	CC	O	O
23	CancelUrl	URL to redirect to when shopper clicks cancel during the payment process	80	CC	O	O
24	CancelUrlParams	Query String to be appended to the Merchant Cancel URL.	255	CC	O	O
25	Stan	Credit Note Number (must be unique for the same batch)  # Mandatory for CAPTURE and CREDIT transactions only.	6	CC	O	O
Fields 26 – 29 are applicable for 3D Secure transactions. Merchants who connect to a third-party MPI for credit card number authentication may send the authentication results to eNETS.						
26	Cavv	Cardholder authentication Verification Value. Required when the value of Transaction Status is “Y” or “A”	28	CC	C	C
27	purchaseXid	Transaction Identifier	20	CC	C	C
28	Status	Indicates whether a transaction qualifies as an authenticated transaction  Y = Authentication Successful. Customer was successfully authenticated. All data needed for clearing, including the Cardholder Authentication Verification Value, is included in the message.  N = Authentication Failed. Customer failed authentication. Transaction denied.  U = Authentication Could Not Be Performed. Authentication could not be completed, due to technical or other problems.  A = Attempts Processing Performed. Authentication could not be completed, but a proof of authentication attempt (CAVV) was generated.	1	CC	R	R
29	ECI	Electronic Commerce Indicator	2	CC	C	C
30	Param1	Reserved Valid values: - “RPP” (see section 7) - “RECURRING” (see section 8)  Note: - Strings in uppercase - “RPP” for specific merchants only	100	CC	N	O

		- "RECURRING" for merchants subscribed to generic recurring payment service.				
31	Param2	Reserved (for RPP, see section 7 below)	550	CC	N	O
32	Param3	Reserved	100	CC	N	O
33	Param4	Reserved	100	CC	N	O
34	Param5	Reserved	100	CC	N	O

S/N	Field Name	Remark	Max Length
1	NetsMid	UMID (Universal Merchant ID). Uniquely identifies merchant. Value assigned by eNETS.	20
2	Tid	Terminal ID. Identifies the merchant terminal, if applicable, from where transaction originates. If not present in the request, eNETS will set the Terminal ID to the IP address from where the request originates.	20
3	TxnAmount	Transaction amount in cents for all currencies. E.g. for \$10.00 in SGD, the TXN_AMT will be equal to 1000. For ¥10,000 in JPY, the TXN_AMT will be equal to 1000000.	10
4	CurrencyCode	Identifies the type of currency used. Three-character currency code following the ISO 4217 standard. If not set, defaults to SGD.	3
5	MerchantTxnRef	Unique reference code assigned by merchant for this transaction.	20
6	MerchantCertId	ID of the Certificate used for this transaction. It is assigned by eNETS after merchant generates his key pair and uploads the public certificate to the gateway.	5
7	MerchantTimeZone	A <b>time zone</b> is a region of the Earth that has adopted the same standard time, usually referred to as the <b>local time</b> . MerchantTimeZone means the time zone which Merchant locates (e.g. "+11:30")	6
8	merchant_txn_date	Transaction Date (dd/mm/yyyy format)	10
9	merchant_txn_time	Transaction Time (hh:mm:ss format)	8
10	PaymentMode	Payment method of the transaction DD – Debit Card This field is required if the merchant does not publish his collection page. If not sent in the request, eNETS will redirect the consumer to the Merchant Collection Page to select the payment mode.	2
11	SubmissionMode	B – Browser Submission For Direct Debit payment, set to B.	1
12	Param1	Reserved	Defined by user
13	Param2	Reserved	Defined by user
14	Param3	Reserved	Defined by user
15	Param4	Reserved	Defined by user
16	Param5	Reserved	Defined by user



### 3 Architecture

#### 3.1 UMAPI Components

The unified merchant API enables merchants to perform transaction via credit card or debit card. It facilitates merchants with the functions described in Table 1.

Function	Description
formPayReq	Form a payment request message based on eNETS II message specifications. The result is an encrypted string ready to be sent to eNETS II payment gateway.
formAck	Form an acknowledgement message based on eNETS II message specifications. The result is an encrypted string ready to be sent to eNETS II payment gateway.
doPayment	Send out a payment request message to eNETS II payment gateway.

Function	Description
unpackNotification	Unpack notification message from eNETS II payment gateway. It also decrypts and verifies the message.
unpackResponse	Unpack payment response message from eNETS II payment gateway. It also decrypts and verifies the message.
queryTxnStatus	Query transaction status at eNETS II payment gateway.

Table 1 Unified Merchant API Functions

The eNETS payment gateway listens to HTTP request from the merchant. The Unified Merchant API gives merchant the flexibility of posting the message via API, or forming the message via API but posting it in their preferred way. The two methods are further elaborated as follows:

- Use the 'doPayment' functions. Merchants only need to supply UMAPI with payment information; UMAPI will form the message and send out to eNETS II payment gateway. Merchant can also specify proxy server if necessary. This method takes care of the HTTP post for merchant. It is simple and easy to implement.
- Use the 'formPayReq' / 'formAck' functions. Merchant supplies UMAPI with payment information and gets a ready-to-send message from UMAPI; they then do their own HTTP post to eNETS II. This method gives merchant the freedom of implementing their own business logic, e.g. re-try mechanism after time-out, and using more advanced technology, e.g. proxy authentication. It is more complicated and suitable for more advanced merchants. Internally, 'doPayment' actually makes use of the 'formPayReq' function.

## 3.2 Package Structure

### 3.2.1 Java Version

The UMAPI is provided in a jar file, umapi.jar. The package structure is as follows:

```
com.wiz.enets2.transaction.umapi
├── Merchant
├── CreditMerchant
├── DebitMerchant
├── NetsConfig
├── data
│   ├── TxnReq
│   ├── CreditTxnReq
│   ├── DebitTxnReq
│   ├── TxnNotify
│   ├── CreditNotify
│   ├── DebitNotify
│   ├── VAcctNotify
│   ├── TxnAck
│   ├── CreditAck
│   ├── DebitAck
│   ├── VAcctAck
│   ├── TxnRes
│   ├── CreditTxnRes
│   ├── DebitTxnRes
│   └── exceptions
├── TransactionError
├── TransactionException
├── logging
│   └── logger
├── util
│   └── PGPHelper
└── util
```

```

|
|__ Encryptor
|__ HttpClient
|__ PGPGeneratorApp
|__ PGPImp
|__ PGPKeyGenerator
|__ XMLUtil

```

### 3.2.2 .NET Version

The UMAPI is provided as a dynamic link library (dll), umapi.dll. The package structure is as follows:

com.wiz.enets2.transaction.umapi

```

|
|__ Merchant
|__ CreditMerchant
|__ DebitMerchant
|__ NetsConfig
|__ data
|   |
|   |__ TxnReq
|   |__ CreditTxnReq
|   |__ DebitTxnReq
|   |__ TxnNotify
|   |__ CreditNotify
|   |__ DebitNotify
|   |__ TxnAck
|   |__ CreditAck
|   |__ DebitAck
|   |__ TxnRes
|   |__ CreditTxnRes
|   |__ DebitTxnRes
|__ exceptions
|   |
|   |__ TransactionError
|   |__ TransactionException
|__ logging
|   |
|   |__ logger
|__ util
|   |
|   |__ PGPHelper
|__ util
|   |
|   |__ Encryptor
|   |__ HttpClient
|   |__ PGPGeneratorApp
|   |__ PGPImp
|   |__ PGPKeyGenerator
|   |__ XMLUtil

```

### 3.2.3 UMAPI Components

This section is common for both the Java version and the .NET version. Below are the descriptions for the main UMAPI components:

File Name	File Description
Merchant	<p>Merchant is the base class where the specialised CreditMerchant, DebitMerchant, and VAcctMerchant are subclassed. It provides merchants with the methods to perform transactions, as described in Table 1 above.</p> <p>For Credit Card transactions, CreditMerchant should be instantiated as follows:  CreditMerchant m = (CreditMerchant) Merchant.getInstance  (Merchant.MERCHANT_TYPE_CREDIT);</p> <p>For Direct Debit transactions:  DebitMerchant m = (DebitMerchant) Merchant.getInstance  (Merchant.MERCHANT_TYPE_DEBIT);</p>
TxnReq	<p>TxnReq is the data class used to encapsulate the Payment Request message, and provides getter/setter methods for setting the various fields in the message.</p> <p>Merchants need to pass the TxnReq object to the formPayReq/doPayment methods.</p>
TxnNotify	TxnNotify is the data class used to encapsulate the Payment Notification message.
TxnAck	TxnAck is the data class used to encapsulate the Payment Acknowledgement message.
TxnRes	TxnRes is the data class used to encapsulate the Payment Response message.
PGPHelper	PGPHelper is the helper class used for performing PGP operations. It is used internally for encrypting/decrypting the messages, and for signing/verification of the message signatures.
HttpClient	This is the class that acts as the Http Client, and is used internally to establish the connection to eNETS server, for server-side submission mode.
Encryptor	Utility class for encryption/decryption of sensitive data in NETSConfig.xml.

## 3.3 Supporting files

### 3.3.1 Java Version

1. Cryptix OpenPGP
2. Castor-0.9.7
3. Log4j-1.2.9
4. Commons-httpclient-3.0-rc3

### 3.3.2 .NET Version

1. log4net.dll-1.2.10
2. SecureBlackbox.dll – 6.0.0.137
3. SecureBlackbox.PGP.dll – 6.0.0.137

## 4 Setup / Deployment

### 4.1 API installation and setup

#### 4.1.1 Pre-requisites for deploying UMAPI.

The merchant should meet, or have the necessary software / hardware as well as technical support to meet all the pre-requisites stated in section 1.6 (see page 2 for more details), before progressing to further steps.

#### 4.1.2 Information Provided to Merchant

##### 4.1.2.1 Java Version

The NETS Merchant Technical Support team will provide the following information to the merchant upon completion of registration formalities:

1. Merchant ID (unique field that identifies merchant).
2. UMAPI.zip, contains the UMAPI library, supporting jar files and UMAPI configuration files.
3. OpenPGP certificate containing the Public Key of eNETS.
4. A certificate containing the Public Key of Root CA certificate for eNETS SSL Certificate. (for verification of eNETS's SSL Certificate when performing connection using httpclient of the UMAPI.)
5. URL for merchants to submit their payment request to eNETS.  
The testing URL is <https://test.enets.sg/enets2/PaymentListener.do>  
The production URL is <https://www.enets.sg/enets2/PaymentListener.do>

##### 4.1.2.2 .NET Version

The NETS Merchant Technical Support team will provide the following information to the merchant upon completion of registration formalities:

1. Merchant ID (unique field that identifies merchant).
2. Umapl.zip, containing the umapi library, supporting dlls and umapi configuration files.
3. OpenPGP certificate containing the Public Key of eNETS.
4. URL for merchants to submit their payment request to eNETS.  
The testing URL is <https://test.enets.sg/enets2/PaymentListener.do>  
The production URL is <https://www.enets.sg/enets2/PaymentListener.do>

#### 4.1.3 Hosting of Domain Name / Internet Connectivity

Merchants need to have a valid domain name (e.g. [www.abc.com](http://www.abc.com)) and public IP address, as well as connectivity with the internet (typical implementation is subscription via a leased line with an Internet Service Provider) for eNETS to communicate with them.

#### 4.1.4 Server Setup

For internet transaction payments, merchants need to install a web server, develop a web site with a payment checkout page (contains payment details, e.g. payment amount, goods / services customer is paying for etc.), and ensure the web site has internet connectivity.

The use of an application server for internet transaction payments is optional. Security requirements may require merchants to separate the web and application tiers into two physical servers.

Merchants can perform the mandated actions above either by following the installation instructions available in the accompanying technical guides or by engaging a vendor (system integrator) to perform the job.

##### 4.1.4.1 Java Version

The merchant needs to install an operating system (OS) capable of supporting the Java Runtime Environment (Ver 1.4 and above).

##### 4.1.4.2 .NET Version

The merchant needs to install an operation system (OS) capable of supporting .NET Framework 1.1 or above.

#### 4.1.5 Installation of JRE / JSSE

This section is only required for Java version. For .NET version, it is not necessary to be installed.

The merchant needs to install the JRE 1.4 and above (Java Runtime Environment version 1.4; available for download from <http://www.java.sun.com> or packaged together with commercial web / application servers).

#### 4.1.6 Setup Java Classpath

This section is only required for Java version. For .NET version, it is not necessary to be installed.

The Java Classpath is an environment variable (typically set via the operating system), containing one or more directories that are used as roots to search for .class Java files.

The CLASSPATH statement thus needs to be set in order to use the UMAPI Java class files.

Assuming a Windows based environment where the JRE was installed in the directory 'C:\JRE1\_4\', the following line should be added to the OS environment variable :

```
PATH=%PATH%; C:\JRE1_4\;
```

#### 4.1.7 Installation of UMAPI

The UMAPI comes in the form of a Winzip file (UMAPI.zip). Merchants need to extract the files to a temporary directory, and copy the contents to the appropriate directories.

The UMAPI contains the following files:

##### 4.1.7.1 Java Version

Directory	File Name	File Description
cert	uat/nets-full-pub.pgp.asc	OpenPGP certificate containing the Public Key of eNETS. It is used to encrypt the XML message during the forming of the Payment Request/ Acknowledgment message. It is also used for verifying eNETS signature during the unpacking of the Payment Response / Notification message  Merchants have a choice of which directories they wish to place the certificate. However, the location information of the certificate must be updated accordingly in the UMAPI configuration file.
	prod/nets-full-pub.pgp.asc	Same as above, but for PRODUCTION environment. For production roll-in, merchant must replace the uat eNETS public cert with the corresponding production certificate.
	entrust_ssl_ca.cer	CA root certificate of NETS SSL Server certificate.
config	log4j.properties	Configurations file for log4j. All configuration files need to be placed in the application classpath.
	NETSConfig.xml	Configurations file for UMAPI. Refer to section 4.1.16 for more details.
lib	umapi.jar	UMAPI classes, as described in section 3.2. All the libraries(with the exception of the cryptix libraries) are to be placed in your application lib directory, e.g. {APPLICATION_ROOT}/WEB-
	log4j-1.2.9.jar	Log4j library
	dom4j.jar	Java library for processing XML data.
	xerces.jar	XML java parser library
	castor-0.9.7.jar	This library is used for java-XML binding of the UMAPI data objects.

Directory	File Name	File Description
	cryptix-jce-api.jar cryptix-jce-provider.jar cryptix-message-api.jar cryptix-openpgp-provider.jar cryptix-pki-api.jar	Cryptix OpenPGP is a java implementation of the OpenPGP Standard and consists of these five jar files. Refer to section 4.1.9 for installation.
	commons-httpclient-3.0-rc3.jar commons-codec-1.3.jar commons-logging.jar	HttpClient is the underlying module for communicating with eNETS gateway.
tools	encrypt.bat	Batch file for invoking the password encryption utility
	run.bat	Batch file for invoking the PGP Certificate generator application
	encrypt.sh	Batch file for invoking the password encryption utility (Unix OS)
	run.sh	Batch file for invoking the PGP Certificate generator application (Unix OS)
samples	Credit/Test.jsp	
	Credit/Process.jsp	
	Credit/Success.jsp	

#### 4.1.7.2 .NET Version

Directory	File Name	File Description
etc	nets-uat-full-pub.pgp.asc	OpenPGP certificate containing the Public Key of eNETS. It is used to encrypt the xml message during the forming of the Payment Request/ Acknowledgment message. It is also used for verifying eNETS signature during the unpacking of the Payment Response / Notification message.  Merchants have a choice of which directories they wish to place in the certificate. However, the location information of the certificate must be updated accordingly in the UMAPI configuration file.
	nets-prod-full-pub.pgp.asc	Same as above, but for PRODUCTION environment. For production roll-in, merchant must replace the uat eNETS public cert with the corresponding production certificate.
	NETSConfig.xml	Configurations file for UMAPI.NET. Refer to section 4.1.17 for more details.  <b>Note:</b> The "etc" is defined in the following sample files: a. Credit Browser: cc_browser_request02.aspx b. Credit Server: cc_server_response.aspx c. Debit: dd_browser_request02.aspx  Look for the following line and change the pathname (eg "C:\\umapi.net\\webtest\\etc"): myCreditMerchant = (CreditMerchant)Merchant.getInstance(Merchant.MERCHANT_TYPE_CREDIT, "C:\\umapi.net\\webtest\\etc")

Directory	File Name	File Description
bin	Log4net.dll	Configurations file for log4net.
	umapi.dll	UMAPI dll, as described in section 3.2. All the libraries are to be placed in your application bin directory.
	SecureBlackbox.dll	Provides cryptographic protection for your data kept in storage or during transfer across networks.
	SecureBlackbox.PGP.dll	Cryptographic protection for PGP key
	PGPGeneratorApp.exe	PGPGenerator will generate PGP keys
	Encryptor.exe	Password encryption.
samples	cc_server_request.aspx	
	cc_server_response.aspx	

#### 4.1.8 Installation of policy files for Cryptix OpenPGP

**This section is only required for Java version. For .NET version, it is not necessary to be installed.**

This section applies if you are running JDK 1.4+ that comes with the JCE provider by default.

The default distribution of the JCE allows as Sun calls it 'strong, but limited strength cryptography'. This means that you cannot use RSA keys bigger than 2048 bits, and no symmetric ciphers that use more than 128 bits. ElGamal is not allowed at all, thus DH/DSS cannot be used for encryption.

The solution is to install the Unlimited Strength Jurisdiction Policy files, which can be downloaded from the following URL:

<http://java.sun.com/j2se/1.4/download.html>

For applications using JDK 1.5, users can refer and download from the following URL:

<http://java.sun.com/j2se/1.5.0/download.jsp>

Note: Go to Other Downloads -> Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 5.0 & click Download

For applications using JDK 1.6, users can refer and download from the following URL:

<http://java.sun.com/javase/downloads/index.jsp>

Note: Go to Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 6 & click download

These files have to be installed in \$JAVA\_HOME\$/jre/lib/security

If you are using a JRE packaged together with your commercial web/application server, please go to your provider's website to download the unlimited strength policy files.

#### 4.1.9 Installation of JCE and OpenPGP providers for Cryptix

**This section is only required for Java version. For .NET version, it is not necessary to be installed.**

1. The Cryptix JCE and OpenPGP providers need to be installed as extension libraries. Copy the five cryptix libraries to \$JAVA\_HOME\$/jre/lib/ext
2. Register the providers statically by editing the security properties file, java.security, found in \$JAVA\_HOME\$/jre/lib/security

Add the following two lines:



security.provider.n=cryptix.jce.provider.CryptixCrypto  
security.provider.n=cryptix.openpgp.provider.CryptixOpenPGP

where n is the priority you would like to assign to the providers.

#### 4.1.10 Generation of Public/Private key pair.

##### 4.1.10.1 Java Version

A toolkit is provided to the merchant to generate the PGP Keys. Launch the PGP KeyGenerator application by executing the batch file run.bat in the tools directory.

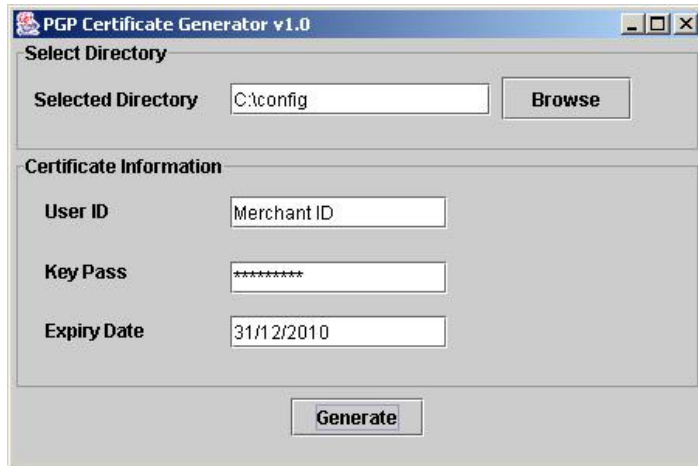


Figure 5. PGP Certificate Generator

The following inputs are required (all fields are compulsory):-

- Selected directory: output directory for the generated key pair. The public key will be saved as ddMMyyyyHHmmss-pub.pgp.asc and the private key as ddMMyyyyHHmmss-priv.pgp.asc. It is advisable to rename the newly generated files.
- User ID: Primary user id bound to the certificate.
- Key Pass: Passphrase for the private key.
- Expiry Date: Format is dd/MM/yyyy.

**Note:** User ID is only a reference value to aid cert generation. You can input the merchant name or mid or even the company name.

After successfully generating the key pair, the following message will be displayed:

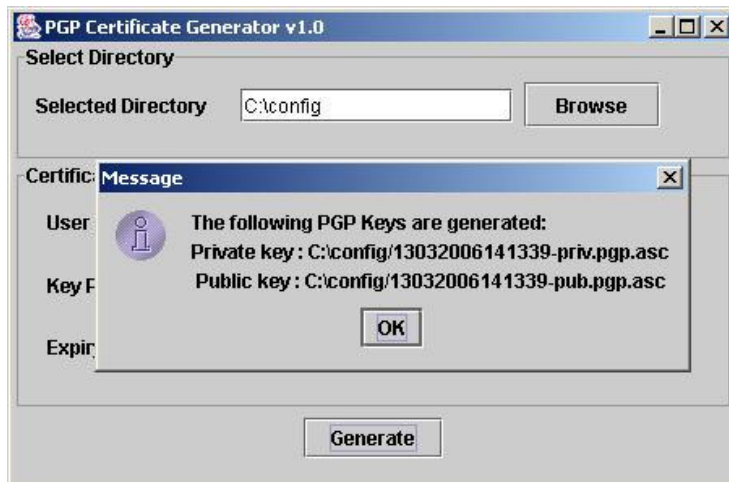


Figure 6. Certificate generation successful

#### 4.1.10.2 .NET Version

A toolkit is provided to the merchant to generate the PGP Keys. Launch the PGP KeyGenerator application by executing the executable file PGPGeneratorApp.exe in the bin directory.

Figure 7. PGP Certificate Generator

The following inputs are required (all fields are compulsory):

Field	Description
Selected directory	Output directory for the generated key pair. The public key will be saved as ddMMyyyyHHmmss-pub.pgp.asc and the private key as ddMMyyyyHHmmss-priv.pgp.asc. It is advisable to rename the newly generated files.
User Id	Primary user id bound to the certificate.
Key Pass	Passphrase for the private key.
Retype Key Pass	Retype passphrase for the private key.
Expiry Date	Format is dd/mm/yyyy.

After successfully generating the key pair, the following message will be displayed:

Figure 8. Certificate generation successful

#### 4.1.11 Encryption of Private key password.

##### 4.1.11.1 Java Version

The private PGP key generated in the previous step is protected by a key password. The key password is stored in the merchKeyPass property in umapi configuration file NETSConfig.xml. Merchants have the option of storing the password as clear text or in encrypted format.

A utility class is provided to the merchant encrypt their key password using Triple DES symmetric encryption. To invoke the utility, follow the steps below:

Assuming the umapi files were extracted to the folder UMAPI\_HOME, navigate to the folder \$UMAPI\_HOME\$/tools/. Run the batch file encrypt.bat, passing in the text to be encrypted.

> Encrypt clear text

The following output will be generated:

```
Generating new secret key to umapikey.dat
Plain text : [clear text]
Encrypted : [372dd5893e2919b96aa644cbac3293c6]
```

When this utility is run for the first time, it will generate a new secret key and save it to a file called umapikey.dat.

The secret key can be copied to a directory of your choice. Subsequent usage of the utility for encryption/decryption will make use of the generated secret key. The location of the key file needs to be specified in the configPath property in NETSConfig.

Copy the encrypted text to the merchKeyPass property in NETSConfig, and set the encryption flag to true.

##### **Note:**

Please make sure that the JCE security provider is registered in your JRE's java.security file, located in \$JAVA\_HOME\$/jre/lib/security/, where \$JAVA\_HOME\$ is the installation path of merchant's JRE.

##### 4.1.11.2 .NET Version

```
c:\ Command Prompt
Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\Documents and Settings\Administrator>cd C:\umapi.net\Encryptor\Encryptor\bin\Debug

C:\umapi.net\Encryptor\Encryptor\bin\Debug>Encryptor.exe encrypt this is a test
Error! Usage : Encrypt MODE TEXT. Example Encryptor encrypt "this is a test"

C:\umapi.net\Encryptor\Encryptor\bin\Debug>Encryptor.exe encrypt "this is a test"
Plain text: [this is a test]
Encrypted: [A16AD743D4B83F7964BDA316F1CD7155]

C:\umapi.net\Encryptor\Encryptor\bin\Debug>
```

The private PGP key generated in the previous step is protected by a key password. The key password is stored in the merchKeyPass property in UMAPI configuration file NETSConfig.xml. Merchants have the option of storing the password as clear text or in encrypted format.

A utility class is provided to the merchant encrypt to their key password using Triple DES symmetric encryption.

To invoke the utility, follow the steps below:

Run the executable file Encryptor.exe, passing in the text to be encrypted.

```
> Encryptor.exe encrypt "this is a test"
```

The following output will be generated:

```
Generating new secret key to umapikey.dat
Plain text: [this is a test]
Encrypted: [A16AD743D4B83F7964BDA316F1CD7155]
```

When this utility is run for the first time, it will generate a new secret key and save it to a file called umapikey.dat.

The secret key can be copied to a directory of your choice. Subsequent usage of the utility for encryption/decryption will make use of the generated secret key hence the location of the key file needs to be specified in the configPath property in NETSConfig.

Copy the encrypted text to the merchKeyPass property in NETSConfig, and set the encryption flag to true.

#### **4.1.12 Uploading of Public certificate to eNETS gateway.**

Merchants need to log in to the administration module at [https://test.enets.sg/enets2\\_admin/](https://test.enets.sg/enets2_admin/) to upload their public certificate generated in step 4.1.9 above.

For UMID, merchants should login with the login of individual MID (grouped under UMID) and follow the same procedure as they perform for Credit/Debit MIDs. Also note, merchants need to send a request to the eNETS Support Team to grant them access to the UMID Cert Upload functions for UMID.

##### **Configuration of proxy settings**

If the merchant is using server-side submission and the merchant's server is behind a proxy server/firewall, the merchant needs to specify the proxy settings in NETSConfig.xml. Please refer to section 4.1.16 for the properties to set.

If the proxy server requires authentication, and merchant wants to store the proxyPassword in encrypted format, refer to section 4.1.10 for instructions on the generating the encrypted password.

#### **4.1.13 Loading of CA root certificate for eNETS**

**This section is only required for Java version. For .NET version, it is not necessary to be installed.**

This section only applies to merchants using the server submission integration approach. eNETS gateway uses an X509 SSL server certificate to identify itself to client applications. For merchants using a JRE other than SUN JDK1.4.2 and above, you need to load the CA root certificate into the Java trusted keystore (default filename: cacerts in "<JRE Install path>/lib/security" directory). Perform the following steps in DOS environment for the loading of the CA root certificate into the Java truststore:

1. Execute the following command to import the CA root certificate into the Java trusted keystore.  
Assuming the CA root certificate file name is entrust\_ssl\_ca.cer

```
keytool -import -alias EntrustCA -file entrust_ssl_ca.cer -keystore <JRE Install path>/lib/security/cacerts
```

Please replace <JRE Install path> with appropriate JRE path.

2. Enter keystore password: changeit (by default if the password has not been changed.)

If the certificate already exists in the truststore under a different alias, the system will display the following message:

```
Certificate already exists in keystore under alias <alias>
Do you still want to add it? [no]: n
```

Else if the certificate is not found in the truststore,

[System display the following message:

Owner: CN=Entrust.net Secure Server Certification Authority, OU=(c) 1999 Entrust.net Limited, OU=www.entrust.net/CPS incorp. by ref. (limits liab.), O=Entrust.net, C=US

Issuer: CN=Entrust.net Secure Server Certification Authority, OU=(c) 1999 Entrust.net Limited, OU=www.entrust.net/CPS incorp. by ref. (limits liab.), O=Entrust.net, C=US

Serial number: 374ad243

Valid from: Wed May 26 00:09:40 SGT 1999 until: Sun May 26 00:39:40 SGT 2019

Certificate fingerprints:

MD5: DF:F2:80:73:CC:F1:E6:61:73:FC:F5:42:E9:C5:7C:EE

SHA1: 99:A6:9B:E6:1A:FE:88:6B:4D:2B:82:00:7C:B8:54:FC:31:7E:15:39]

**Note: System message may be different if CA used is not Entrust.**

3. Enter 'Y' when prompted below.

*Trust this certificate? [no]: y*

4. System will display confirmation message.

*Certificate was added to keystore*

5. Restart your Application Server or JVM to effect the changes

#### 4.1.14 Review of UMAPI Sample scripts

The merchant should look through the UMAPI sample scripts provided by eNETS, and understand how the scripts work together to form a payment process with eNETS.

#### 4.1.15 Customization of scripts by merchant

The merchant should customize the sample UMAPI scripts provided by eNETS according to their business and technical requirements.

Additional checks and database updates prior to composing the payment request message or responding to a notification message may be added where appropriate.

#### 4.1.16 Configuration of UMAPI Config files

Configurations for UMAPI are stored in the file NetsConfig.xml, which contains the following values:

Field	Description
merchPrivKey	Location of merchant private OpenPGP certificate
merchKeyPass	Password for merchant's private OpenPGP certificate
netsPubKey	Location of eNETS public OpenPGP certificate
payURL	eNETS server URL. Only applicable for server submission.
queryURL	eNETS URL to query the transaction status
usePGP	Must be set to true.
useProxy	Applicable for server submission. If the merchant's server is behind a proxy server, set this flag to true.
proxyHost	Proxy server IP
proxyPort	Proxy server Port
proxyUsername	Username for authentication to proxy server(if required)
proxyPassword	Password for authentication to proxy server(if required)
encryption	Flag to indicate whether the merchKeyPass and proxyPassword fields are in clear text or encrypted format. Set to true to indicate that encryption is turned on.
configPath	If password encryption is turned on, this field specifies the folder containing the secret key file. E.g. c:/umapiconfig/
TimeOut	Connection expiry time.
logLevel	Type of logging level to be set E.g. debug, info, error
logFile	Path of log file (umapi.log) to be written on server.
protocol	Protocol to be used E.g: "https"

## 4.2 Sample Scripts

### 4.2.1 Credit Transactions

#### 4.2.1.1 Java Version

##### 4.2.1.1.1 Credit Payment Request

```
<%@ page import="com.wiz.enets2.transaction.util.*,
    com.wiz.enets2.transaction.umapi.data.*,
    com.wiz.enets2.transaction.umapi.*,
    java.util.*"%>

<%
    TxnReq req = new TxnReq();
    CreditTxnReq credReq = new CreditTxnReq();

    String mid = request.getParameter("mid");
    String tid = request.getParameter("tid");
    String paymentMode = request.getParameter("payment_mode");
    String amt = request.getParameter("txn_amount");
    String currency = request.getParameter("currency_code");
    String merRef = request.getParameter("merchant_txn_ref");
    String merchTxnTime=request.getParameter("merch_txn_dateTime");
    String submitMode = request.getParameter("submission_mode");
    String merCertId = request.getParameter("merchant_cert_id");
    String pan = request.getParameter("pan");
    String expiry = request.getParameter("expiry_date");
    String stan = request.getParameter("stan");
    String paymentType = request.getParameter("payment_type");
    String successURL = request.getParameter("success_url");
    String successURLParams = request.getParameter("success_url_params");
    String failureURL = request.getParameter("failure_url");
    String failureURLParams = request.getParameter("failure_url_params");

    req.setNetsMid(mid);
    req.setTid(tid);
    req.setPaymentMode(paymentMode);
    req.setTxnAmount(amt);
    req.setCurrencyCode(currency);
    req.setMerchantTxnRef(merRef);
    req.setMerchantTxnDtm(merchTxnTime);
    req.setSubmissionMode(submitMode);
    req.setMerchantCertId(merCertId);
    req.setSuccessUrl(successURL);
    req.setSuccessUrlParams(successURLParams);
    req.setFailureUrl(failureURL);
    req.setFailureUrlParams(failureURLParams);
    req.setNotifyUrl(request.getParameter("notify_url"));
    req.setNotifyUrlParams(request.getParameter("notify_url_params"));

    credReq.setPan(pan);
    credReq.setExpiryDate(expiry);
    credReq.setPaymentType(paymentType);
    credReq.setStan(stan);
    credReq.setCardHolderName(request.getParameter("name"));
    credReq.setCvv(request.getParameter("cvv"));
    credReq.setPostUrl(request.getParameter("post_url"));
    credReq.setPostUrlParams(request.getParameter("post_url_params"));
    credReq.setCancelUrl(request.getParameter("cancel_url"));
    credReq.setCancelUrlParams(request.getParameter("cancel_url_params"));

    req.setCreditTxnReq(credReq);
```

```

    CreditMerchant m = (CreditMerchant) Merchant.getInstance (Merchant.MERCHANT_TYPE_CREDIT);
%>

<html>
<head></head>
<%
    if (submitMode.equalsIgnoreCase("B"))
    {
        String sMsg = m.formPayReq(req);
        if(sMsg != null && !sMsg.equals("")){
            String url = request.getParameter("gw_url");
%>
<body onLoad="document.txnForm.submit()">
<form name="txnForm" action="<%=url%>" method="POST">
<input type="hidden" name="message" value="<%=sMsg%>">
</form>
</body>
<%    }
    else{
%>
        <h1>ERROR FORMING PAYMENT REQUEST!</h1>
<%
    }
}
else
{
    TxnRes res = m.doPayment(req);
    if(res==null){
        %>
        <h1>ERROR DURING PAYMENT!</h1>
        <%
        }
        else{
            CreditTxnRes credRes = res.getCreditTxnRes();
            if (credRes == null)
                credRes = new CreditTxnRes();
%>
<body>
<table width="100%">
<tr colspan="2"> eNETS II Merchant Simulator Server Submission Result </tr>
<tr> <td>MID: </td> <td><input type="text" name="mid" value="<%=res.getMid()%>" /></td> </tr>
<tr> <td>Merchant Txn Ref: </td> <td><input type="text" name="merchant_txn_ref"
value="<%=res.getMerchantTxnRef()%>" /></td> </tr>
<tr> <td>NETS Txn Ref: </td> <td><input type="text" name="nets_txn_ref"
value="<%=res.getNetsTxnRef()%>" /></td> </tr>
<tr> <td>NETS Txn Time: </td> <td><input type="text" name="nets_txn_dtm"
value="<%=res.getNetsTxnDtm()%>" /></td> </tr>
<tr> <td>NETS Txn Status: </td> <td><input type="text" name="nets_txn_status"
value="<%=res.getNetsTxnStatus()%>" /></td> </tr>
<tr> <td>NETS Txn Response Code: </td> <td><input type="text" name="nets_txn_resp_code"
value="<%=res.getNetsTxnRespCode()%>" /></td> </tr>
<tr> <td>NETS Txn Msg: </td> <td><input type="text" name="nets_txn_msg"
value="<%=res.getNetsTxnMsg()%>" /></td> </tr>
<tr> <td>NETS Amt Deducted: </td> <td><input type="text" name="nets_amt_deducted"
value="<%=credRes.getNetsAmountDeducted()%>" /></td> </tr>
<tr> <td>Bank Auth ID: </td> <td><input type="text" name="bank_auth_id"
value="<%=credRes.getBankAuthId()%>" /></td> </tr>
</table>
</body>
<%
    }
}

```



```
%>
</html>
```

#### 4.2.1.1.2 Credit Payment Response

```
<%@ page import="com.wiz.enets2.transaction.util.*,
com.wiz.enets2.transaction.umapi.data.*,
com.wiz.enets2.transaction.umapi.*"%>

<%
    String sMsg = request.getParameter("message");
    CreditMerchant m = (CreditMerchant) Merchant.getInstance(Merchant.MERCHANT_TYPE_CREDIT);
    TxnRes res = m.unpackResponse(sMsg);
    CreditTxnRes credRes = res.getCreditTxnRes();
    if (credRes == null)
        credRes = new CreditTxnRes();
%>

<html>
<head></head>
<body>
<table width="100%">
<tr colspan="2"> eNETS II Merchant Simulator Success URL. </tr>
<tr> <td>MID: </td> <td><input type="text" name="mid" value="<%=res.getMid()%>"/></td> </tr>
<tr> <td>Merchant Txn Ref: </td> <td><input type="text" name="merchant_txn_ref"
value="<%=res.getMerchantTxnRef()%>"/></td> </tr>
<tr><td>Paymode: </td> <td><input type="text" name="paymode" value="<%=res.
getPaymentMode()%>"/></td> </tr>
<tr> <td>NETS Txn Ref: </td> <td><input type="text" name="nets_txn_ref"
value="<%=res.getNetsTxnRef()%>"/></td> </tr>
<tr> <td>NETS Txn Time: </td> <td><input type="text" name="nets_txn_dtm"
value="<%=res.getNetsTxnDtm()%>"/></td> </tr>
<tr> <td>NETS Txn Status: </td> <td><input type="text" name="nets_txn_status"
value="<%=res.getNetsTxnStatus()%>"/></td> </tr>
<tr> <td>NETS Txn Response Code: </td> <td><input type="text" name="nets_txn_resp_code"
value="<%=res.getNetsTxnRespCode()%>"/></td> </tr>
<tr> <td>NETS Txn Msg: </td> <td><input type="text" size="40" name="nets_txn_msg"
value="<%=res.getNetsTxnMsg()%>"/></td> </tr>
<tr> <td>NETS Amt Deducted: </td> <td><input type="text" name="nets_amt_deducted"
value="<%=credRes.getNetsAmountDeducted()%>"/></td> </tr>
<tr> <td>Bank Auth ID: </td> <td><input type="text" name="bank_auth_id"
value="<%=credRes.getBankAuthId()%>"/></td> </tr>
<%
    If(credRes.getS3dTxnRes() != null){
        S3DTxnRes s3dRes = credRes.getS3dTxnRes();
%>
<tr> <td>PAreq: </td> <td><input type="text" name="PAreq " value="<%=s3dRes.getPAreq()%>"/></td>
</tr>
<tr> <td>MD: </td> <td><input type="text" name="md " value="<%=s3dRes.getMd()%>"/></td>
</tr>
<tr> <td>ACS Url: </td> <td><input type="text" name="acsUrl" value="<%=s3dRes.getAcUrl()%>"/></td>
</tr>
<tr> <td>Terminal Url: </td> <td><input type="text" name="termUrl"
value="<%=s3dRes.getTermUrl()%>"/></td>
</tr>
<tr> <td>Error Code: </td> <td><input type="text" name="errorCode"
value="<%=s3dRes.getErrorCode()%>"/></td>
</tr>
<%
    }
%>
</table>
```



</body></html>

#### 4.2.1.1.3 Credit Payment Notification

```
<%@ page import="com.wiz.enets2.transaction.util.*,
    com.wiz.enets2.transaction.umapi.data.*,
    com.wiz.enets2.transaction.umapi.*"%>

<%
    String sMsg = request.getParameter("message");
    CreditMerchant m = (CreditMerchant) Merchant.getInstance(Merchant.MERCHANT_TYPE_CREDIT);
    TxnNotify notify = m.unpackNotification(sMsg);
    CreditNotify credNotify = notify.getCreditNotify();
    if (credNotify == null)
        credNotify = new CreditNotify();

    TxnAck ack = new TxnAck();
    CreditAck credAck = new CreditAck();

    //sample code
    String paymode= notify.getPaymentMode();
    String submitMode= notify.getSubmissionMode();
    String amount = notify.getTxnAmount();
    String merchTxnDate_Time =notify.getMerchantTxnDtm();
    String nets_Date_time=notify.getNetsTxnDtm();

    ack.setNetsMid(notify.getMid());
    ack.setMerchantCertId(notify.getMerchantCertId());
    ack.setNetsTxnRef(notify.getNetsTxnRef());
    ack.setMerchantTxnRef (notify.getMerchantTxnRef());
    ack.setMerchantTxnStatus ("0000");

    ack.setCreditAck(credAck);
    sMsg = m.formAck (ack);

    out.println(sMsg);

%>
```

#### 4.2.1.2 .NET Version

##### 4.2.1.2.1 Credit Card Server Request

**Note: For server submission, the response is received at each request. There are no redirection.**

```
<%@ Page Language="C#" AutoEventWireup="true" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi.data" %>
<%
    string nets_mid = "", merchant_txn_ref = "", nets_txn_ref = "", nets_txn_dtm = "", payment_mode = "",
    nets_txn_status = "", nets_txn_resp_code = "", nets_txn_msg = "", custom_attribute = "";
    string nets_amount_deducted = "", bank_auth_id = "";
    string PAreq = "", termUrl = "", md = "", acsUrl = "", error_code = "";

    try
    {
        //region prepare umapi objects
        CreditMerchant myCreditMerchant = null;
        TxnReq myTxnReq = null;
        CreditTxnReq myCreditTxnReq = null;
        Secure3D mySecure3D = null;
        TxnRes myTxnRes = null;
        CreditTxnRes myCreditTxnRes = null;
        S3DTxnRes myS3DTxnRes = null;
```

```

myCreditMerchant = (CreditMerchant)Merchant.getInstance(Merchant.MERCHANT_TYPE_CREDIT,
"c:\\umapi\\etc");
//endregion

//region TxnReq
myTxnReq = new TxnReq();
myTxnReq.setNetsMid(Request.Form["nets_mid"]);
myTxnReq.setPaymentMode(Request.Form["payment_mode"]);
myTxnReq.setSubmissionMode(Request.Form["submission_mode"]);
myTxnReq.setTxnAmount(Request.Form["txn_amount"]);
myTxnReq.setCurrencyCode(Request.Form["currency_code"]);
myTxnReq.setMerchantTxnRef(Request.Form["merchant_txn_ref"]);
myTxnReq.setMerchantCertId(Request.Form["merchant_cert_id"]);
myTxnReq.setTid(Request.Form["tid"]);
myTxnReq.setMerchantTxnDtm(Request.Form["merchant_txn_dtm"]);
myTxnReq.setSuccessUrl(Request.Form["success_url"]);
myTxnReq.setSuccessUrlParams(Request.Form["success_url_params"]);
myTxnReq.setFailureUrl(Request.Form["failure_url"]);
myTxnReq.setFailureUrlParams(Request.Form["failure_url_params"]);
myTxnReq.setNotifyUrl(Request.Form["notify_url"]);
myTxnReq.setNotifyUrlParams(Request.Form["notify_url_params"]);
//endregion

//region CreditTxnReq
myCreditTxnReq = new CreditTxnReq();
myCreditTxnReq.setPaymentType(Request.Form["payment_type"]);
myCreditTxnReq.setPan(Request.Form["pan"]);
myCreditTxnReq.setExpiryDate(Request.Form["expiry_date"]);
myCreditTxnReq.setCvv(Request.Form["cvv"]);
myCreditTxnReq.setCardHolderName(Request.Form["card_holder_name"]);
myCreditTxnReq.setPostUrl(Request.Form["post_url"]);
myCreditTxnReq.setPostUrlParams(Request.Form["post_url_params"]);
myCreditTxnReq.setCancelUrl(Request.Form["cancel_url"]);
myCreditTxnReq.setCancelUrlParams(Request.Form["cancel_url_params"]);
myCreditTxnReq.setStan(Request.Form["stan"]);
myCreditTxnReq.setParam1(Request.Form["param1"]);
myCreditTxnReq.setParam2(Request.Form["param2"]);
myCreditTxnReq.setParam3(Request.Form[""]);
myCreditTxnReq.setParam4(Request.Form["param4"]);
myCreditTxnReq.setParam5(Request.Form["param5"]);
//endregion

mySecure3D = new Secure3D();
mySecure3D.setTxStatus(Request.Form["status"]);
mySecure3D.setTxStatus(Request.Form["cavv"]);
mySecure3D.setTxStatus(Request.Form["purchase_xid"]);
mySecure3D.setTxStatus(Request.Form["eci"]);
//endregion

//region doPayment
myCreditTxnReq.setS3d(mySecure3D);
myTxnReq.setCreditTxnReq(myCreditTxnReq);
myTxnRes = myCreditMerchant.doPayment(myTxnReq);
myCreditTxnRes = myTxnRes.getCreditTxnRes();
if (myCreditTxnRes != null)
{
    myS3DTxnRes = myCreditTxnRes.getS3dTxnRes();
}
else
{
    Response.Write("no CreditTxnRes<br/>");
}
//endregion

```

```

//region TxnRes
if (myTxnRes != null)
{
    nets_mid = myTxnRes.getNetsMid();
    merchant_txn_ref = myTxnRes.getMerchantTxnRef();
    nets_txn_ref = myTxnRes.getNetsTxnRef();
    nets_txn_dtm = myTxnRes.getNetsTxnDtm();
    payment_mode = myTxnRes.getPaymentMode();
    nets_txn_status = myTxnRes.getNetsTxnStatus();
    nets_txn_resp_code = myTxnRes.getNetsTxnRespCode();
    nets_txn_msg = myTxnRes.getNetsTxnMsg();
    ArrayList customAttributeAL = myTxnRes.getCustomAttribute();
    if (customAttributeAL != null)
    {
        for (int i = 0; i < customAttributeAL.Count; i++)
        {
            custom_attribute += ((CustomAttribute)customAttributeAL[i]).getKey() + ": " +
((CustomAttribute)customAttributeAL[i]).getValue() + "<br/>";
        }
    }
}
//endregion

//region CreditTxnRes
if (myCreditTxnRes != null)
{
    nets_amount_deducted = myCreditTxnRes.getNetsAmountDeducted();
    bank_auth_id = myCreditTxnRes.getBankAuthId();
}
//endregion

//region S3DTxnRes
if (myS3DTxnRes != null)
{
    PAreq = myS3DTxnRes.getPAreq();
    termUrl = myS3DTxnRes.getTermUrl();
    md = myS3DTxnRes.getMd();
    acsUrl = myS3DTxnRes.getAcsUrl();
    error_code = myS3DTxnRes.getErrorCode();
}
//endregion
}
catch (Exception umapiE)
{
    Response.Write(umapiE.Message + "<br/>" + umapiE.StackTrace + "<br/>");
}
%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>UMAPI .NET Test</title>
</head>

<script type="text/javascript" language="javascript">
    function show3d()
    {
        var netsTxnStatusS = "<%= nets_txn_status %>";
        if(netsTxnStatusS != null && netsTxnStatusS != "")
        {

```

```

        if(netsTxnStatusS == "5")
        {
            document.getElementById('div_3d').style.display = "";
        }
        else
        {
            document.getElementById('div_3d').style.display = "none";
        }
    }
}
</script>

```

```

<body onload="javascript:show3d();">
    <div>
        <b>CREDIT CARD SERVER SUBMISSION RESPONSE</b>
    </div>
    <form id="form1" action="">
        <div>
            <table>
                <tr><td colspan="3"><b>TxnRes parameters:</b></td></tr>
                <tr>
                    <td style="width:200px">NETS MID</td>
                    <td style="width:450px"><%= nets_mid %></td>
                    <td style="width:100px"></td>
                </tr>
                <tr>
                    <td>Merchant Transaction Reference</td>
                    <td><%= merchant_txn_ref %></td>
                    <td></td>
                </tr>
                <tr>
                    <td>NETS Transaction Reference</td>
                    <td><%= nets_txn_ref %></td>
                    <td></td>
                </tr>
                <tr>
                    <td>NETS Transaction Date Time</td>
                    <td><%= nets_txn_dtm %></td>
                    <td></td>
                </tr>
                <tr>
                    <td>Payment Mode</td>
                    <td><%= payment_mode %></td>
                    <td></td>
                </tr>
                <tr>
                    <td>NETS Transaction Status</td>
                    <td><%= nets_txn_status %></td>
                    <td></td>
                </tr>
                <tr>
                    <td>NETS Transaction Response Code</td>
                    <td><%= nets_txn_resp_code %></td>
                    <td></td>
                </tr>
                <tr>
                    <td>NETS Transaction Message</td>
                    <td><%= nets_txn_msg %></td>
                    <td></td>
                </tr>
                <tr>
                    <td>Custom Attribute</td>
                    <td><%= custom_attribute %></td>

```

```

        <td></td>
    </tr>
    <tr><td colspan="3">&nbsp;</td></tr>
    <tr><td colspan="3"><b>CreditTxnRes parameters:</b></td></tr>
    <tr>
        <td>NETS Amount Deducted</td>
        <td><%= nets_amount_deducted %></td>
        <td></td>
    </tr>
    <tr>
        <td>Bank Authorization ID</td>
        <td><%= bank_auth_id %></td>
        <td>Maybe empty for failure case.</td>
    </tr>
    <tr><td colspan="3">&nbsp;</td></tr>
    <tr><td colspan="3"><b>S3DTxnRes parameters:</b></td></tr>
    <tr><td colspan="3"><b>Note that S3DTxnRes is under the CreditTxnRes</b></td></tr>
    <tr>
        <td>PAreq</td>
        <td><%= PAreq %></td>
        <td></td>
    </tr>
    <tr>
        <td>Term URL</td>
        <td><%= termUrl %></td>
        <td></td>
    </tr>
    <tr>
        <td>MD</td>
        <td><%= md %></td>
        <td></td>
    </tr>
    <tr>
        <td>ACS URL</td>
        <td><%= acsUrl %></td>
        <td></td>
    </tr>
    <tr>
        <td>Error Code</td>
        <td><%= error_code %></td>
        <td></td>
    </tr>
</table>
</div>
</form>
<div id="div_3d" style="display:none">
    <form id="form3d" method="post" action="<%= acsUrl %>">
        <input type="hidden" id="PaReq" name="PaReq" value="<%= PAreq %>" />
        <input type="hidden" id="TermUrl" name="TermUrl" value="<%= termUrl %>" />
        <input type="hidden" id="MD" name="MD" value="<%= md %>" />
        <input type="submit" id="submit" name="submit" value="Go to ACS page" />
    </form>
</div>
</body>
</html>

```

#### 4.2.1.2.2 Credit Card Server Post

```

<%@ Page Language="C#" AutoEventWireup="true" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi.data" %>
<%

```

```

try
{
    string filenameS = ".\\log\\";
    filenameS += "cc-server-post." + DateTime.Now.ToString("yyyyMMddHHmmss") + ".txt";
    System.IO.FileStream outputFS = new System.IO.FileStream(filenameS, System.IO.FileMode.Create,
System.IO.FileAccess.Write);

    string messageS = Request.Form["message"];
    try
    {
        CreditMerchant myCreditMerchant = null;
        TxnRes myTxnRes = null;
        CreditTxnRes myCreditTxnRes = null;
        S3DTxnRes myS3DTxnRes = null;

        myCreditMerchant =
(CreditMerchant)Merchant.getInstance(Merchant.MERCHANT_TYPE_CREDIT, "c:\\umapi\\etc");
        myTxnRes = myCreditMerchant.unpackResponse(messageS);
        myCreditTxnRes = myTxnRes.getCreditTxnRes();
        string tempS = "nets mid: " + myTxnRes.getNetsMid() + "\r\n";
        tempS += "merchant transaction reference: " + myTxnRes.getMerchantTxnRef() + "\r\n";
        tempS += "nets transaction reference: " + myTxnRes.getNetsTxnRef() + "\r\n";
        tempS += "nets transaction date time: " + myTxnRes.getNetsTxnDtm() + "\r\n";
        tempS += "payment mode: " + myTxnRes.getPaymentMode() + "\r\n";
        tempS += "nets transaction status: " + myTxnRes.getNetsTxnStatus() + "\r\n";
        tempS += "nets transaction response code: " + myTxnRes.getNetsTxnRespCode() + "\r\n";
        tempS += "nets transaction message: " + myTxnRes.getNetsTxnMsg() + "\r\n";
        ArrayList customAL = myTxnRes.getCustomAttribute();
        if (customAL != null)
        {
            for (int i = 0; i < customAL.Count; i++)
            {
                tempS += "custom attribute: " + ((CustomAttribute)customAL[i]).getKey() + ": " +
((CustomAttribute)customAL[i]).getValue() + "\r\n";
            }
        }
        byte[] tempBA = System.Text.Encoding.UTF8.GetBytes(tempS);
        outputFS.Write(tempBA, 0, tempBA.Length);

        myS3DTxnRes = myCreditTxnRes.getS3dTxnRes();
        string tempS = "nets amount deducted: " + myCreditTxnRes.getNetsAmountDeducted() + "\r\n";
        tempS += "bank authorization id: " + myCreditTxnRes.getBankAuthId() + "\r\n";
        tempS += "nets risk score: " + myCreditTxnRes.getNetsRiskScore() + "\r\n";
        byte[] tempBA = System.Text.Encoding.UTF8.GetBytes(tempS);
        outputFS.Write(tempBA, 0, tempBA.Length);

        string tempS = "pareq: " + myS3DTxnRes.getPAreq() + "\r\n";
        tempS += "term url: " + myS3DTxnRes.getTermUrl() + "\r\n";
        tempS += "md: " + myS3DTxnRes.getMd() + "\r\n";
        tempS += "acs url: " + myS3DTxnRes.getAcUrl() + "\r\n";
        tempS += "error code: " + myS3DTxnRes.getErrorCode() + "\r\n";
        byte[] tempBA = System.Text.Encoding.UTF8.GetBytes(tempS);
        outputFS.Write(tempBA, 0, tempBA.Length);
    }
    catch (Exception umapiE)
    {
        string tempS = umapiE.Message + "\r\n" + umapiE.StackTrace + "\r\n";
        byte[] tempBA = System.Text.Encoding.UTF8.GetBytes(tempS);
        outputFS.Write(tempBA, 0, tempBA.Length);
    }
    finally
    {

```

```

        outputFS.Close();
    }
}
catch (Exception ex)
{
    Response.Write(ex.Message + "<br/>" + ex.StackTrace + "<br/>");
}
%>

```

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

```

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div></div>
    </form>
</body>
</html>

```

#### 4.2.1.2.3 Credit Card Server Notification/Acknowledgement

```

<%@ Page Language="C#" AutoEventWireup="true" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi.data" %>
<%
    try
    {
        string filenameS = ".\\log\\";
        filenameS += "cc-server-notify." + DateTime.Now.ToString("yyyyMMddHHmmss") + ".txt";
        System.IO.FileStream outputFS = new System.IO.FileStream(filenameS, System.IO.FileMode.Create,
        System.IO.FileAccess.Write);

        string messageS = Request.Form["message"];
        try
        {
            CreditMerchant myCreditMerchant = null;
            TxnNotify myTxnNotify = null;
            CreditNotify myCreditNotify = null;
            TxnAck myTxnAck = null;
            CreditAck myCreditAck = null;
            string urlS = null;
            myCreditMerchant =
(CreditMerchant)Merchant.GetInstance(Merchant.MERCHANT_TYPE_CREDIT, "c:\\umapi\\etc");
            try
            {
                urlS = NETSConfig.get("payURL");
                urlS += (urlS.IndexOf("?") < 0) ? "?" : "&" + "message=";
            }
            catch (Exception netsConfigE)
            { ; }

            myTxnNotify = myCreditMerchant.unpackNotification(messageS);
            myCreditNotify = myTxnNotify.getCreditNotify();
            string tempS = "nets mid: " + myTxnNotify.getNetsMid() + "\r\n";
            tempS += "terminal id: " + myTxnNotify.getTid() + "\r\n";
            tempS += "payment mode: " + myTxnNotify.getPaymentMode() + "\r\n";
            tempS += "submission mode: " + myTxnNotify.getSubmissionMode() + "\r\n";
            tempS += "transaction amount: " + myTxnNotify.getTxnAmount() + "\r\n";

```

```

tempS += "currency code: " + myTxnNotify.getCurrencyCode() + "\r\n";
tempS += "merchant transaction reference: " + myTxnNotify.getMerchantTxnRef() + "\r\n";
tempS += "merchant transaction date time: " + myTxnNotify.getMerchantTxnDtm() + "\r\n";
tempS += "merchant cert id: " + myTxnNotify.getMerchantCertId() + "\r\n";
tempS += "nets transaction reference: " + myTxnNotify.getNetsTxnRef() + "\r\n";
tempS += "nets transaction date time: " + myTxnNotify.getNetsTxnDtm() + "\r\n";
tempS += "nets transaction status: " + myTxnNotify.getNetsTxnStatus() + "\r\n";
byte[] tempBA = System.Text.Encoding.UTF8.GetBytes(tempS);
outputFS.Write(tempBA, 0, tempBA.Length);

string tempS = "bank authorization id: " + myCreditNotify.getBankAuthId() + "\r\n";
tempS += "nets risk score: " + myCreditNotify.getNetsRiskScore() + "\r\n";
tempS += "param 1: " + myCreditNotify.getParam1() + "\r\n";
tempS += "param 2: " + myCreditNotify.getParam2() + "\r\n";
tempS += "param 3: " + myCreditNotify.getParam3() + "\r\n";
tempS += "param 4: " + myCreditNotify.getParam4() + "\r\n";
tempS += "param 5: " + myCreditNotify.getParam5() + "\r\n";
byte[] tempBA = System.Text.Encoding.UTF8.GetBytes(tempS);
outputFS.Write(tempBA, 0, tempBA.Length);

myCreditAck = new CreditAck();
myTxnAck.setCreditAck(myCreditAck);
myTxnAck.setMerchantCertId(myTxnNotify.getMerchantCertId());
myTxnAck.setMerchantTxnRef(myTxnNotify.getMerchantTxnRef());
myTxnAck.setMerchantTxnStatus("0000");
myTxnAck.setMid(myTxnNotify.getMid());
myTxnAck.setNetsMid(myTxnNotify.getNetsMid());
myTxnAck.setNetsTxnRef(myTxnNotify.getNetsTxnRef());

messageS = null;
messageS = myCreditMerchant.formAck(myTxnAck);
byte[] tempBA = System.Text.Encoding.UTF8.GetBytes("redirecting to " + urlS + messageS +
"\r\n");
outputFS.Write(tempBA, 0, tempBA.Length);
outputFS.Close();
Response.Redirect(urlS + messageS, true);
}
catch (Exception umapiE)
{
    string tempS = umapiE.Message + "\r\n" + umapiE.StackTrace + "\r\n";
    byte[] tempBA = System.Text.Encoding.UTF8.GetBytes(tempS);
    outputFS.Write(tempBA, 0, tempBA.Length);
}
finally
{
    outputFS.Close();
}
}
catch (Exception ex)
{
    Response.Write(ex.Message + "<br/>" + ex.StackTrace + "<br/>");
}
%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">

```



```

</div></div>
</form>
</body>
</html>

```

#### 4.2.1.2.4 Credit Card Browser Request

```

<%@ Page Language="C#" AutoEventWireup="true" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi.data" %>
<%
    string messageS = "", urlS = "";
    CreditMerchant myCreditMerchant = null;
    TxnReq myTxnReq = null;
    CreditTxnReq myCreditTxnReq = null;
    myCreditMerchant = (CreditMerchant)Merchant.getInstance(Merchant.MERCHANT_TYPE_CREDIT,
"c:\\umapi\\etc");
    urlS = NETSConfig.get("payURL");
    myTxnReq = new TxnReq();
    myTxnReq.setNetsMid(Request.Form["nets_mid"]);
    myTxnReq.setSubmissionMode(Request.Form["submission_mode"]);
    myTxnReq.setTxnAmount(Request.Form["txn_amount"]);
    myTxnReq.setMerchantTxnRef(Request.Form["merchant_txn_ref"]);
    myTxnReq.setMerchantCertId(Request.Form["merchant_cert_id"]);
    myTxnReq.setPaymentMode(Request.Form["payment_mode"]);
    myTxnReq.setTid(Request.Form["tid"]);
    myTxnReq.setCurrencyCode(Request.Form["currency_code"]);
    myTxnReq.setMerchantTxnDtm(Request.Form["merchant_txn_dtm"]);
    myTxnReq.setSuccessUrl(Request.Form["success_url"]);
    myTxnReq.setSuccessUrlParams(Request.Form["success_url_params"]);
    myTxnReq.setFailureUrl(Request.Form["failure_url"]);
    myTxnReq.setFailureUrlParams(Request.Form["failure_url_params"]);
    myTxnReq.setNotifyUrl(Request.Form["notify_url"]);
    myTxnReq.setNotifyUrlParams(Request.Form["notify_url_params"]);

    myCreditTxnReq = new CreditTxnReq();
    myCreditTxnReq.setPaymentType(Request.Form["payment_type"]);
    myCreditTxnReq.setPan(Request.Form["pan"]);
    myCreditTxnReq.setCvv(Request.Form["cvv"]);
    myCreditTxnReq.setExpiryDate(Request.Form["expiry_date"]);
    myCreditTxnReq.setCardHolderName(Request.Form["card_holder_name"]);
    myCreditTxnReq.setPostUrl(Request.Form["post_url"]);
    myCreditTxnReq.setPostUrlParams(Request.Form["post_url_params"]);
    myCreditTxnReq.setCancelUrl(Request.Form["cancel_url"]);
    myCreditTxnReq.setCancelUrlParams(Request.Form["cancel_url_params"]);
    myCreditTxnReq.setStan(Request.Form["stan"]);
    myCreditTxnReq.setParam1(Request.Form["param1"]);
    myCreditTxnReq.setParam2(Request.Form["param2"]);
    myCreditTxnReq.setParam3(Request.Form[""]);
    myCreditTxnReq.setParam4(Request.Form["param4"]);
    myCreditTxnReq.setParam5(Request.Form["param5"]);

    ArrayList productAL = new ArrayList();
    string[] productSA = Request.Form["product_details"].Split('|');
    for (int i = 0; i < int.Parse(productSA[0]); i++)
    {
        ProductDetails productPD = new ProductDetails(productSA[i * 4 + 1], productSA[i * 4 + 2], productSA[i
* 4 + 3], int.Parse(productSA[i * 4 + 4]));
        productAL.Add(productPD);
    }

    myCreditTxnReq.setEzProtect(myEzProtect);

```

```

myCreditTxnReq.setS3d(mySecure3D);

myTxnReq.setCreditTxnReq(myCreditTxnReq);

messageS = myCreditMerchant.formPayReq(myTxnReq);
%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>UMAPI .NET Test</title>
</head>
<body>
    <form id="form1" method="post" action="<%= urlS %>">
        <input type="hidden" id="message" name="message" value="<%= messageS %>" />
    </form>
    <script type="text/javascript" language="javascript">
        document.forms[0].submit();
    </script>
</body>
</html>

```

#### 4.2.1.2.5 Credit Card Browser Redirection

**Success:** When the transaction had been approved by the bank, it will redirect to the merchant's success page.

**Failure:** When the bank rejects the transaction or incorrect information is entered, it will redirect to the merchant's failure page.

**Cancel:** Click the 'Cancel' button at the page, it will cancel the transaction and redirect to the merchant's cancel page.

```

<%@ Page Language="C#" AutoEventWireup="true" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi.data" %>
<%
    try
    {
        string messageS = Request.Form["message"];
        try
        {
            CreditMerchant myCreditMerchant = null;
            TxnRes myTxnRes = null;
            CreditTxnRes myCreditTxnRes = null;
            S3DTxnRes myS3DTxnRes = null;

            myCreditMerchant =
(CreditMerchant)Merchant.getInstance(Merchant.MERCHANT_TYPE_CREDIT, "c:\\umapi\\etc");
            myTxnRes = myCreditMerchant.unpackResponse(messageS);
            myCreditTxnRes = myTxnRes.getCreditTxnRes();
            nets_mid.Text = myTxnRes.getNetsMid();
            merchant_txn_ref.Text = myTxnRes.getMerchantTxnRef();
            nets_txn_ref.Text = myTxnRes.getNetsTxnRef();
            nets_txn_dtm.Text = myTxnRes.getNetsTxnDtm();
            payment_mode.Text = myTxnRes.getPaymentMode();
            nets_txn_status.Text = myTxnRes.getNetsTxnStatus();
            nets_txn_resp_code.Text = myTxnRes.getNetsTxnRespCode();
            nets_txn_msg.Text = myTxnRes.getNetsTxnMsg();
            ArrayList customAttributeAL = myTxnRes.getCustomAttribute();

```

```

        if (customAttributeAL != null)
        {
            for (int i = 0; i < customAttributeAL.Count; i++)
            {
                custom_attribute.Text += ((CustomAttribute)customAttributeAL[i]).getKey() + ": " +
                ((CustomAttribute)customAttributeAL[i]).getValue() + "<br/>";
            }
        }

        myS3DTxnRes = myCreditTxnRes.getS3dTxnRes();
        nets_amount_deducted.Text = myCreditTxnRes.getNetsAmountDeducted();
        bank_auth_id.Text = myCreditTxnRes.getBankAuthId();

        PAreq.Text = myS3DTxnRes.getPAreq();
        termUrl.Text = myS3DTxnRes.getTermUrl();
        md.Text = myS3DTxnRes.getMd();
        acsUrl.Text = myS3DTxnRes.getAcsUrl();
        error_code.Text = myS3DTxnRes.getErrorCode();
    }
    catch (Exception umapiE)
    {
        Response.Write(umapiE.Message + "<br/>" + umapiE.StackTrace + "<br/>");
    }
}
catch (Exception e)
{ ; }
%>

```

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>UMAPI .NET Test</title>
</head>
<body>
    <div>
        <b>CREDIT CARD BROWSER SUBMISSION SUCCESS URL</b>
    </div>
    <form id="form1" runat="server">
        <div>
            <table>
                <tr><td colspan="3"><b>Success URL parameters:</b></td></tr>
                <tr>
                    <td style="width:200px">Author</td>
                    <td style="width:450px"><asp:Label runat="server" ID="author" /></td>
                    <td style="width:100px"></td>
                </tr>
                <tr><td colspan="3">&nbsp;</td></tr>
                <tr><td colspan="3"><b>TxnRes parameters:</b></td></tr>
                <tr>
                    <td>NETS MID</td>
                    <td><asp:Label runat="server" ID="nets_mid" /></td>
                    <td></td>
                </tr>
                <tr>
                    <td>Merchant Transaction Reference</td>
                    <td><asp:Label runat="server" ID="merchant_txn_ref" /></td>
                    <td></td>
                </tr>
                <tr>
                    <td>NETS Transaction Reference</td>
                    <td><asp:Label runat="server" ID="nets_txn_ref" /></td>
                    <td></td>
                </tr>
            </table>
        </div>
    </form>
</body>
</html>

```

```

        <td></td>
    </tr>
    <tr>
        <td>NETS Transaction Date Time</td>
        <td><asp:Label runat="server" ID="nets_txn_dtm" /></td>
        <td></td>
    </tr>
    <tr>
        <td>Payment Mode</td>
        <td><asp:Label runat="server" ID="payment_mode" /></td>
        <td></td>
    </tr>
    <tr>
        <td>NETS Transaction Status</td>
        <td><asp:Label runat="server" ID="nets_txn_status" /></td>
        <td></td>
    </tr>
    <tr>
        <td>NETS Transaction Response Code</td>
        <td><asp:Label runat="server" ID="nets_txn_resp_code" /></td>
        <td></td>
    </tr>
    <tr>
        <td>NETS Transaction Message</td>
        <td><asp:Label runat="server" ID="nets_txn_msg" /></td>
        <td></td>
    </tr>
    <tr>
        <td>Custom Attribute</td>
        <td><asp:Label runat="server" ID="custom_attribute" /></td>
        <td></td>
    </tr>
    <tr><td colspan="3">&nbsp;</td></tr>
    <tr><td colspan="3"><b>CreditTxnRes parameters:</b></td></tr>
    <tr>
        <td>NETS Amount Deducted</td>
        <td><asp:Label runat="server" ID="nets_amount_deducted" /></td>
        <td></td>
    </tr>
    <tr>
        <td>Bank Authorization ID</td>
        <td><asp:Label runat="server" ID="bank_auth_id" /></td>
        <td>Maybe empty for failure case.</td>
    </tr>
    <tr><td colspan="3">&nbsp;</td></tr>
    <tr><td colspan="3"><b>S3DTxnRes parameters:</b></td></tr>
    <tr><td colspan="3"><b>Note that S3DTxnRes is under the CreditTxnRes</b></td></tr>
    <tr>
        <td>PAreq</td>
        <td><asp:Label runat="server" ID="PAreq" /></td>
        <td></td>
    </tr>
    <tr>
        <td>Term URL</td>
        <td><asp:Label runat="server" ID="termUrl" /></td>
        <td></td>
    </tr>
    <tr>
        <td>MD</td>
        <td><asp:Label runat="server" ID="md" /></td>
        <td></td>
    </tr>
    <tr>

```

```

        <td>ACS URL</td>
        <td><asp:Label runat="server" ID="acsUrl" /></td>
        <td></td>
    </tr>
    <tr>
        <td>Error Code</td>
        <td><asp:Label runat="server" ID="error_code" /></td>
        <td></td>
    </tr>
</table>
</div>
</form>
</body>
</html>

```

#### 4.2.1.2.6 Credit Card Browser Post

```

<%@ Page Language="C#" AutoEventWireup="true" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi.data" %>
<%
    try
    {
        string filenameS = ".\\log\\";
        filenameS += "cc-browser-post." + DateTime.Now.ToString("yyyyMMddHHmmss") + ".txt";
        System.IO.FileStream outputFS = new System.IO.FileStream(filenameS, System.IO.FileMode.Create,
        System.IO.FileAccess.Write);

        string messageS = Request.Form["message"];
        try
        {
            CreditMerchant myCreditMerchant = null;
            TxnRes myTxnRes = null;
            CreditTxnRes myCreditTxnRes = null;
            S3DTxnRes myS3DTxnRes = null;

            myCreditMerchant =
            (CreditMerchant)Merchant.getInstance(Merchant.MERCHANT_TYPE_CREDIT, "c:\\umapi\\etc");
            myTxnRes = myCreditMerchant.unpackResponse(messageS);
            myCreditTxnRes = myTxnRes.getCreditTxnRes();
            string tempS = "nets mid: " + myTxnRes.getNetsMid() + "\\r\\n";
            tempS += "merchant transaction reference: " + myTxnRes.getMerchantTxnRef() + "\\r\\n";
            tempS += "nets transaction reference: " + myTxnRes.getNetsTxnRef() + "\\r\\n";
            tempS += "nets transaction date time: " + myTxnRes.getNetsTxnDtm() + "\\r\\n";
            tempS += "payment mode: " + myTxnRes.getPaymentMode() + "\\r\\n";
            tempS += "nets transaction status: " + myTxnRes.getNetsTxnStatus() + "\\r\\n";
            tempS += "nets transaction response code: " + myTxnRes.getNetsTxnRespCode() + "\\r\\n";
            tempS += "nets transaction message: " + myTxnRes.getNetsTxnMsg() + "\\r\\n";
            ArrayList customAL = myTxnRes.getCustomAttribute();
            if (customAL != null)
            {
                for (int i = 0; i < customAL.Count; i++)
                {
                    tempS += "custom attribute: " + ((CustomAttribute)customAL[i]).getKey() + ": " +
                    ((CustomAttribute)customAL[i]).getValue() + "\\r\\n";
                }
            }
            byte[] tempBA = System.Text.Encoding.UTF8.GetBytes(tempS);
            outputFS.Write(tempBA, 0, tempBA.Length);
        }

        myS3DTxnRes = myCreditTxnRes.getS3dTxnRes();
    }
    catch { }
}

```

```

string tempS = "nets amount deducted: " + myCreditTxnRes.getNetsAmountDeducted() + "\r\n";
tempS += "bank authorization id: " + myCreditTxnRes.getBankAuthId() + "\r\n";
tempS += "nets risk score: " + myCreditTxnRes.getNetsRiskScore() + "\r\n";
byte[] tempBA = System.Text.Encoding.UTF8.GetBytes(tempS);
outputFS.Write(tempBA, 0, tempBA.Length);

string tempS = "pareq: " + myS3DTxnRes.getPareq() + "\r\n";
tempS += "term url: " + myS3DTxnRes.getTermUrl() + "\r\n";
tempS += "md: " + myS3DTxnRes.getMd() + "\r\n";
tempS += "acs url: " + myS3DTxnRes.getAcUrl() + "\r\n";
tempS += "error code: " + myS3DTxnRes.getErrorCode() + "\r\n";
byte[] tempBA = System.Text.Encoding.UTF8.GetBytes(tempS);
outputFS.Write(tempBA, 0, tempBA.Length);
}
catch (Exception umapiE)
{
    string tempS = umapiE.Message + "\r\n" + umapiE.StackTrace + "\r\n";
    byte[] tempBA = System.Text.Encoding.UTF8.GetBytes(tempS);
    outputFS.Write(tempBA, 0, tempBA.Length);
}
finally
{
    outputFS.Close();
}
}
catch (Exception ex)
{
    Response.Write(ex.Message + "<br/>" + ex.StackTrace + "<br/>");
}
}%>

```

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

```

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>

            </div>
        </form>
</body>
</html>

```

#### 4.2.1.2.7 Credit Card Browser Notification/Acknowledgement

```

<%@ Page Language="C#" AutoEventWireup="true" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi.data" %>
<%
    try
    {
        string filenameS = ".\\log\\";
        filenameS += "cc-browser-notify." + DateTime.Now.ToString("yyyyMMddHHmmss") + ".txt";
        System.IO.FileStream outputFS = new System.IO.FileStream(filenameS, System.IO.FileMode.Create,
        System.IO.FileAccess.Write);

        string messageS = Request.Form["message"];
    }
    try

```

```

{
    CreditMerchant myCreditMerchant = null;
    TxnNotify myTxnNotify = null;
    CreditNotify myCreditNotify = null;
    TxnAck myTxnAck = null;
    CreditAck myCreditAck = null;
    string urlS = null;

    myCreditMerchant =
(CreditMerchant)Merchant.getInstance(Merchant.MERCHANT_TYPE_CREDIT, "c:\\umapi\\etc");
    try
    {
        urlS = NETSConfig.get("payURL");
        urlS += (urlS.IndexOf("?") < 0) ? "?" : "&" + "message=";
    }
    catch (Exception netsConfigE)
    { ; }

    myTxnNotify = myCreditMerchant.unpackNotification(messageS);
    myCreditNotify = myTxnNotify.getCreditNotify();
    string tempS = "nets mid: " + myTxnNotify.getNetsMid() + "\r\n";
    tempS += "terminal id: " + myTxnNotify.getTid() + "\r\n";
    tempS += "payment mode: " + myTxnNotify.getPaymentMode() + "\r\n";
    tempS += "submission mode: " + myTxnNotify.getSubmissionMode() + "\r\n";
    tempS += "transaction amount: " + myTxnNotify.getTxnAmount() + "\r\n";
    tempS += "currency code: " + myTxnNotify.getCurrencyCode() + "\r\n";
    tempS += "merchant transaction reference: " + myTxnNotify.getMerchantTxnRef() + "\r\n";
    tempS += "merchant transaction date time: " + myTxnNotify.getMerchantTxnDtm() + "\r\n";
    tempS += "merchant cert id: " + myTxnNotify.getMerchantCertId() + "\r\n";
    tempS += "nets transaction reference: " + myTxnNotify.getNetsTxnRef() + "\r\n";
    tempS += "nets transaction date time: " + myTxnNotify.getNetsTxnDtm() + "\r\n";
    tempS += "nets transaction status: " + myTxnNotify.getNetsTxnStatus() + "\r\n";
    byte[] tempBA = System.Text.Encoding.UTF8.GetBytes(tempS);
    outputFS.Write(tempBA, 0, tempBA.Length);

    string tempS = "bank authorization id: " + myCreditNotify.getBankAuthId() + "\r\n";
    tempS += "nets risk score: " + myCreditNotify.getNetsRiskScore() + "\r\n";
    tempS += "param 1: " + myCreditNotify.getParam1() + "\r\n";
    tempS += "param 2: " + myCreditNotify.getParam2() + "\r\n";
    tempS += "param 3: " + myCreditNotify.getParam3() + "\r\n";
    tempS += "param 4: " + myCreditNotify.getParam4() + "\r\n";
    tempS += "param 5: " + myCreditNotify.getParam5() + "\r\n";
    byte[] tempBA = System.Text.Encoding.UTF8.GetBytes(tempS);
    outputFS.Write(tempBA, 0, tempBA.Length);
    myCreditAck = new CreditAck();
    myTxnAck = new TxnAck();
    myTxnAck.setCreditAck(myCreditAck);
    myTxnAck.setMerchantCertId(myTxnNotify.getMerchantCertId());
    myTxnAck.setMerchantTxnRef(myTxnNotify.getMerchantTxnRef());
    myTxnAck.setMerchantTxnStatus("0000");
    myTxnAck.setMid(myTxnNotify.getMid());
    myTxnAck.setNetsMid(myTxnNotify.getNetsMid());
    myTxnAck.setNetsTxnRef(myTxnNotify.getNetsTxnRef());

    messageS = null;
    if (myTxnAck != null)
        messageS = myCreditMerchant.formAck(myTxnAck);
    if (urlS != null && messageS != null)
    {
        byte[] tempBA = System.Text.Encoding.UTF8.GetBytes("redirecting to " + urlS + messageS +
"\r\n");

        outputFS.Write(tempBA, 0, tempBA.Length);
        outputFS.Close();
    }
}

```

```

        Response.Redirect(urlS + messageS, true);
    }
    catch (Exception umapiE)
    {
        string tempS = umapiE.Message + "\r\n" + umapiE.StackTrace + "\r\n";
        byte[] tempBA = System.Text.Encoding.UTF8.GetBytes(tempS);
        outputFS.Write(tempBA, 0, tempBA.Length);
    }
    finally
    {
        outputFS.Close();
    }
}
catch (Exception ex)
{
    Response.Write(ex.Message + "<br/>" + ex.StackTrace + "<br/>");
}
%>

```

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

```

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div></div>
    </form>
</body>
</html>

```

## 4.2.1 Debit Transactions

### 4.2.1.3 Java Version

#### 4.2.1.3.1 Debit Payment Request

```

<%@ page import="com.wiz.enets2.transaction.util.*,
    com.wiz.enets2.transaction.umapi.data.*,
    com.wiz.enets2.transaction.umapi.*"%>

<%
    TxnReq req = new TxnReq();
    DebitTxnReq debitReq = new DebitTxnReq();

    String mid = request.getParameter("mid");
    String tid = request.getParameter("tid");
    String amt = request.getParameter("txn_amount");
    String currency = request.getParameter("currency_code");
    String merRef = request.getParameter("merchant_txn_ref");
    String merCertId = request.getParameter("merchant_cert_id");
    String txnDate = request.getParameter("merchant_txn_date");
    String txnTime = request.getParameter("merchant_txn_time");
    String timeZone = request.getParameter("merchant_time_zone");
    String param1 = request.getParameter("param1");
    String param2 = request.getParameter("param2");
    String param3 = request.getParameter("param3");
    String param4 = request.getParameter("param4");
    String param5 = request.getParameter("param5");

    req.setNetsMid(mid);

```



```

req.setTid(tid);
req.setTxnAmount(amt);
req.setCurrencyCode(currency);
req.setMerchantTxnRef(merRef);
req.setMerchantCertId(merCertId);
req.setPaymentMode("DD");
req.setSubmissionMode("B");
debitReq.setParam1(param1);
debitReq.setParam2(param2);
debitReq.setParam3(param3);
debitReq.setParam4(param4);
debitReq.setParam5(param5);

debitReq.setMerchantTxnDate(txnDate);
debitReq.setMerchantTxnTime (txnTime);
debitReq.setMerchantTimeZone (timeZone);
req.setDebitTxnReq(debitReq);

DebitMerchant m = (DebitMerchant) Merchant.getInstance (Merchant.MERCHANT_TYPE_DEBIT);
String sMsg = m.formPayReq(req);
String url = request.getParameter("gw_url");

```

```
%>
```

```

<html>
<head></head>
<body onLoad="document.txnForm.submit()">
<form name="txnForm" action="<%=url%>" method="POST">
<input type="hidden" name="message" value="<%=sMsg%>">
</form>
</body>

</html>

```

#### 4.2.1.3.2 Debit Payment Response

```

<%@ page import="com.wiz.enets2.transaction.util.*,
com.wiz.enets2.transaction.umapi.data.*,
com.wiz.enets2.transaction.umapi.*,
java.util Enumeration"%>

<%
System.out.println("3");
Enumeration enum = request.getParameterNames();
System.out.println("4" +enum);
while (enum.hasMoreElements()){
System.out.println("5 ");
String attribute = (String) enum.nextElement();
System.out.println("=====>txnend : "+ attribute+" :"+ request.getParameter(attribute));
}
String sMsg = request.getParameter("message");
DebitMerchant m = (DebitMerchant) Merchant.getInstance(Merchant.MERCHANT_TYPE_DEBIT);
TxnRes res = m.unpackResponse(sMsg);
DebitTxnRes debitRes = res.getDebitTxnRes();
if (debitRes == null)
debitRes = new DebitTxnRes();
%>
<html>
<head><meta http-equiv="CACHE-CONTROL" content="NO-CACHE"></head>
<body>
<table width="100%">

```

```

<tr colspan="2"> eNETS II Merchant Simulator Success URL. </tr>
<tr> <td>Merchant ID: </td> <td><input type="text" name="merchant_id" value="%=res
res.getNetsMid()%>" /></td> </tr>
<tr> <td>Terminal Id: </td> <td><input type="text" name="tid" value="%=res. getTid()%>" /></td> </tr>
<tr> <td>Merchant Txn Ref: </td> <td><input type="text" name="merchant_txn_ref"
value="%=res.getMerchantTxnRef()%>" /></td> </tr>
<tr> <td>NETS Txn Ref: </td> <td><input type="text" name="nets_txn_ref"
value="%=res.getNetsTxnRef()%>" /></td> </tr>
<tr> <td>GW Txn Time: </td> <td><input type="text" name="nets_txn_time"
value="%=debitRes.getGwTxnTime()%>" /></td> </tr>
<tr> <td>GW Txn Date: </td> <td><input type="text" name="nets_txn_date"
value="%=debitRes.getGwTxnDate()%>" /></td> </tr>
<tr> <td>GW Value Date: </td> <td><input type="text" name="gw_value_date" value="%=debitRes.
getGwValueDate() %>" /></td> </tr>
<tr> <td>GW Time Zone: </td> <td><input type="text" name="gw_time_zone" value="%=debitRes.
getGwTimeZone() %>" /></td> </tr>
<tr> <td>NETS Txn Status: </td> <td><input type="text" name="nets_txn_status"
value="%=res.getNetsTxnStatus()%>" /></td> </tr>
<tr> <td>NETS Txn Response Code: </td> <td><input type="text" name="nets_txn_resp_code"
value="%=res.getNetsTxnRespCode()%>" /></td> </tr>
<tr> <td>NETS Txn Msg: </td> <td><input type="text" name="nets_txn_msg"
value="%=res.getNetsTxnMsg()%>" /></td> </tr>
<tr> <td>Bank ID: </td> <td><input type="text" name="bank_id" value="%=debitRes.getBankId()%>" /></td>
</tr>
<tr> <td>Bank Ref ID: </td> <td><input type="text" name="bank_ref_code"
value="%=debitRes.getBankRefCode()%>" /></td> </tr>
<tr> <td>Bank Remarks: </td> <td><input type="text" name="bank_remarks" value="%=debitRes.
getBankRemarks() %>" /></td> </tr>
<tr> <td>Bank Status: </td> <td><input type="text" name="bank_status" value="%=debitRes.getBankStatus()
%>" /></td> </tr>
<tr> <td>Bank Txn Date: </td> <td><input type="text" name="bank_txn_date" value="%=debitRes.
getBankTxnDate() %>" /></td> </tr>
<tr> <td>Bank Txn Time: </td> <td><input type="text" name="bank_txn_time" value="%=debitRes.
getBankTxnTime() %>" /></td> </tr>
<tr> <td>Bank Time Zone: </td> <td><input type="text" name="bank_time_zone" value="%=debitRes.
getBankTimeZone() %>" /></td> </tr>
<tr> <td>Txn Amount: </td> <td><input type="text" name="txnAmount" value="%=debitRes.
getTxnDestAmt() %>" /></td> </tr>
<tr> <td>Dest Exchg Rate: </td> <td><input type="text" name="dest_exchg_rate" value="%=debitRes.
getFuncDestExchangeRate() %>" /></td> </tr>
<tr> <td>Dest Currency Code: </td> <td><input type="text" name="dest_currency_code" value="%=debitRes.
getTxnDestCurrencyCode() %>" /></td> </tr>
<tr> <td>Src Txn Amount: </td> <td><input type="text" name="srctxnAmount" value="%=debitRes.
getTxnSrcAmt() %>" /></td> </tr>
<tr> <td>Src Exchg Rate: </td> <td><input type="text" name=" src_exchg_rate" value="%=debitRes.
getSrcFuncExchangeRate() %>" /></td> </tr>
<tr> <td>Src Currency Code: </td> <td><input type="text" name=" src_currency_code" value="%=debitRes.
getTxnSrcCurrencyCode() %>" /></td> </tr>
<tr> <td>Retry: </td> <td><input type="text" name=" retry" value="%=debitRes.getRetry() %>" /></td> </tr>
<tr> <td>Version: </td> <td><input type="text" name=" version" value="%=debitRes. getVersion()%>" /></td>
</tr>
</table>
</body>
</html>

```

#### 4.2.1.3.3 Debit Payment Notification/Acknowledgement

```

<%@ page import="com.wiz.enets2.transaction.util.*,
com.wiz.enets2.transaction.umapi.data.*,
com.wiz.enets2.transaction.umapi.*"%>

```

```

<%

```

```
String sMsg = request.getParameter("message");
DebitMerchant m = (DebitMerchant) Merchant.getInstance(Merchant.MERCHANT_TYPE_DEBIT);
TxnNotify notify = m.unpackNotification(sMsg);
DebitNotify debitNotify = notify.getDebitNotify();
if (debitNotify == null)
    debitNotify = new DebitNotify();

TxnAck ack = new TxnAck();
DebitAck debitAck = new DebitAck();

ack.setNetsMid(notify.getNetsMid());
ack.setMerchantTxnRef(notify.getMerchantTxnRef());
ack.setMerchantTxnStatus("00000");
ack.setNetsTxnRef(notify.getNetsTxnRef());
ack.setMerchantCertId(notify.getMerchantCertId());

debitAck.setParam1(debitNotify.getParam1());
debitAck.setParam2(debitNotify.getParam2());
debitAck.setParam3(debitNotify.getParam3());
debitAck.setParam4(debitNotify.getParam4());
debitAck.setParam5(debitNotify.getParam5());
debitAck.setVersion(debitNotify.getVersion());
debitAck.setRetry(retry);                                     //assume retry value is set

//shown as example to retrieve the value of notification parameters
String nets_tid=debitNotify.getTid();
String merchantCertID=notify.getMerchantCertId();
String retry= debitNotify.getRetry();
String netsTxnStatus=notify.getNetsTxnStatus();
String txnAmt= debitNotify.getTxnSrcAmt();
String currency=debitNotify.getTxnSrcCurrencyCode();
String exchange_rate=debitNotify.get ();
String gwTxnDate=debitNotify getGwTxnDate();
String gwTxnTime=debitNotify getGwTxnTime();
String gwTimeZone=debitNotify getGwTimeZone();
String gwValDate=debitNotify.getGwValueDate();

String bankID=debitNotify getBankId();
String bankTxnTime=debitNotify getBankTxnTime();
String bankTxnDate=debitNotify getBankTxnDate();
String bankRefCode=debitNotify getBankRefCode();
String bankTimeZone=debitNotify getBankTimeZone();
String bankStatus=debitNotify getBankStatus();
StringbankRemarks=debitNotify getBankRemarks();

//Example on getting the value of reserved parameters 11 to 15 not shown .

//Send acknowledge
ack.setDebitAck(debitAck);
sMsg = m.formAck (ack);

out.println(sMsg);

%>
```

#### 4.2.1.4 .NET Version

##### 4.2.1.4.1 Debit Browser Request

```
<%@ Page Language="C#" AutoEventWireup="true" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi.data" %>
<%
```

```

string messageS = "", urlS = "";
try
{
    DebitMerchant myDebitMerchant = null;
    TxnReq myTxnReq = null;
    DebitTxnReq myDebitTxnReq = null;
    myDebitMerchant = (DebitMerchant)Merchant.getInstance(Merchant.MERCHANT_TYPE_DEBIT,
"c:\\umapi\\etc");
    try
    {
        urlS = NETSConfig.get("payURL");
    }
    catch (Exception netsConfigE)
    { ;}

    myTxnReq.setNetsMid(Request.Form["nets_mid"]);
    myTxnReq.setSubmissionMode(Request.Form["submission_mode"]);
    myTxnReq.setTxnAmount(Request.Form["txn_amount"]);
    myTxnReq.setMerchantTxnRef(Request.Form["merchant_txn_ref"]);
    myTxnReq.setMerchantCertId(Request.Form["merchant_cert_id"]);
    myTxnReq.setPaymentMode(Request.Form["payment_mode"]);
    myTxnReq.setTid(Request.Form["tid"]);
    myTxnReq.setCurrencyCode(Request.Form["currency_code"]);
    myTxnReq.setMerchantTxnDtm(Request.Form["merchant_txn_dtm"]);
    myTxnReq.setSuccessUrl(Request.Form["success_url"]);
    myTxnReq.setSuccessUrlParams(Request.Form["success_url_params"]);
    myTxnReq.setFailureUrl(Request.Form["failure_url"]);
    myTxnReq.setFailureUrlParams(Request.Form["failure_url_params"]);
    myTxnReq.setNotifyUrl(Request.Form["notify_url"]);
    myTxnReq.setNotifyUrlParams(Request.Form["notify_url_params"]);

    myDebitTxnReq.setMerchantTimeZone(Request.Form["merchant_time_zone"]);
    myDebitTxnReq.setMerchantTxnDate(Request.Form["merchant_txn_date"]);
    myDebitTxnReq.setMerchantTxnTime(Request.Form["merchant_txn_time"]);
    myDebitTxnReq.setParam1(Request.Form["param1"]);
    myDebitTxnReq.setParam2(Request.Form["param2"]);
    myDebitTxnReq.setParam3(Request.Form["param3"]);
    myDebitTxnReq.setParam4(Request.Form["param4"]);
    myDebitTxnReq.setParam5(Request.Form["param5"]);

    myTxnReq.setDebitTxnReq(myDebitTxnReq);
    messageS = myDebitMerchant.formPayReq(myTxnReq);
    byte[] tempBA = System.Text.Encoding.UTF8.GetBytes("redirecting to " + urlS + messageS +
"\r\n");

    outputFS.Write(tempBA, 0, tempBA.Length);
    outputFS.Close();
}
catch (Exception umapiE)
{
    Response.Write(ex.Message + "<br/>" + ex.StackTrace + "<br/>");
}

%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>UMAPI .NET Test</title>
</head>
<body>
    <form id="form1" method="post" action="<%= urlS %>">
        <input type="hidden" id="message" name="message" value="<%= messageS %>" />
    </form>

```

```
<script type="text/javascript" language="javascript">
    document.forms[0].submit();
</script>
</body>
</html>
```

#### 4.2.1.4.2 Debit Browser Notification/Acknowledgement

```
<%@ Page Language="C#" AutoEventWireup="true" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi.data" %>
<%
    try
    {
        string filenameS = ".\\log\\";
        filenameS += "dd-browser-notify." + DateTime.Now.ToString("yyyyMMddHHmmss") + ".txt";
        System.IO.FileStream outputFS = new System.IO.FileStream(filenameS, System.IO.FileMode.Create,
        System.IO.FileAccess.Write);

        string messageS = Request.Form["message"];
        try
        {
            DebitMerchant myDebitMerchant = null;
            TxnNotify myTxnNotify = null;
            DebitNotify myDebitNotify = null;
            TxnAck myTxnAck = null;
            DebitAck myDebitAck = null;
            string urlS = null;
            myDebitMerchant = (DebitMerchant)Merchant.getInstance(Merchant.MERCHANT_TYPE_DEBIT,
            "c:\\umapi\\etc");
            try
            {
                urlS = NETSConfig.get("payURL");
                urlS += (urlS.IndexOf("?") < 0) ? "?" : "&" + "message=";
            }
            catch (Exception netsConfigE)
            { ; }

            myTxnNotify = myDebitMerchant.unpackNotification(messageS);

            myDebitNotify = myTxnNotify.getDebitNotify();
            string tempS = "nets mid: " + myTxnNotify.getNetsMid() + "\r\n";
            tempS += "terminal id: " + myTxnNotify.getTid() + "\r\n";
            tempS += "payment mode: " + myTxnNotify.getPaymentMode() + "\r\n";
            tempS += "submission mode: " + myTxnNotify.getSubmissionMode() + "\r\n";
            tempS += "transaction amount: " + myTxnNotify.getTxnAmount() + "\r\n";
            tempS += "currency code: " + myTxnNotify.getCurrencyCode() + "\r\n";
            tempS += "merchant transaction reference: " + myTxnNotify.getMerchantTxnRef() + "\r\n";
            tempS += "merchant transaction date time: " + myTxnNotify.getMerchantTxnDtm() + "\r\n";
            tempS += "merchant cert id: " + myTxnNotify.getMerchantCertId() + "\r\n";
            tempS += "nets transaction reference: " + myTxnNotify.getNetsTxnRef() + "\r\n";
            tempS += "nets transaction date time: " + myTxnNotify.getNetsTxnDtm() + "\r\n";
            tempS += "nets transaction status: " + myTxnNotify.getNetsTxnStatus() + "\r\n";
            byte[] tempBA = System.Text.Encoding.UTF8.GetBytes(tempS);
            outputFS.Write(tempBA, 0, tempBA.Length);

            string tempS = "bank id: " + myDebitNotify.getBankId() + "\r\n";
            tempS += "bank reference code: " + myDebitNotify.getBankRefCode() + "\r\n";
            tempS += "bank remarks: " + myDebitNotify.getBankRemarks() + "\r\n";
            tempS += "bank status: " + myDebitNotify.getBankStatus() + "\r\n";
            tempS += "bank time zone: " + myDebitNotify.getBankTimeZone() + "\r\n";
```

```

tempS += "bank transaction date: " + myDebitNotify.getBankTxnDate() + "\r\n";
tempS += "bank transaction time: " + myDebitNotify.getBankTxnTime() + "\r\n";
tempS += "nets exchange rate: " + myDebitNotify.getFuncDestExchangeRate() + "\r\n";
tempS += "nets status: " + myDebitNotify.getGwStatus() + "\r\n";
tempS += "nets time zone: " + myDebitNotify.getGwTimeZone() + "\r\n";
tempS += "nets transaction code: " + myDebitNotify.getGwTxnCode() + "\r\n";
tempS += "nets transaction date: " + myDebitNotify.getGwTxnDate() + "\r\n";
tempS += "nets transaction time: " + myDebitNotify.getGwTxnTime() + "\r\n";
tempS += "nets value date: " + myDebitNotify.getGwValueDate() + "\r\n";
tempS += "retry: " + myDebitNotify.getRetry() + "\r\n";
tempS += "bank exchange rate: " + myDebitNotify.getSrcFuncExchangeRate() + "\r\n";
tempS += "tid: " + myDebitNotify.getTid() + "\r\n";
tempS += "nets transaction amount: " + myDebitNotify.getTxnDestAmt() + "\r\n";
tempS += "nets currency code: " + myDebitNotify.getTxnDestCurrencyCode() + "\r\n";
tempS += "bank transaction amount: " + myDebitNotify.getTxnFuncAmt() + "\r\n";
tempS += "bank currency code: " + myDebitNotify.getTxnFunCurrencyCode() + "\r\n";
tempS += "transaction amount: " + myDebitNotify.getTxnSrcAmt() + "\r\n";
tempS += "transaction currency code: " + myDebitNotify.getTxnSrcCurrencyCode() + "\r\n";
tempS += "version: " + myDebitNotify.getVersion() + "\r\n";
tempS += "param 1: " + myDebitNotify.getParam1() + "\r\n";
tempS += "param 2: " + myDebitNotify.getParam2() + "\r\n";
tempS += "param 3: " + myDebitNotify.getParam3() + "\r\n";
tempS += "param 4: " + myDebitNotify.getParam4() + "\r\n";
tempS += "param 5: " + myDebitNotify.getParam5() + "\r\n";
tempS += "param 11: " + myDebitNotify.getParam11() + "\r\n";
tempS += "param 12: " + myDebitNotify.getParam12() + "\r\n";
tempS += "param 13: " + myDebitNotify.getParam13() + "\r\n";
tempS += "param 14: " + myDebitNotify.getParam14() + "\r\n";
tempS += "param 15: " + myDebitNotify.getParam15() + "\r\n";
byte[] tempBA = System.Text.Encoding.UTF8.GetBytes(tempS);
outputFS.Write(tempBA, 0, tempBA.Length);

myDebitAck = new DebitAck();
myDebitAck.setCertId(myTxnNotify.getMerchantCertId());
myDebitAck.setParam1(myDebitNotify.getParam1());
myDebitAck.setParam2(myDebitNotify.getParam2());
myDebitAck.setParam3(myDebitNotify.getParam3());
myDebitAck.setParam4(myDebitNotify.getParam4());
myDebitAck.setParam5(myDebitNotify.getParam5());
myDebitAck.setRetry(myDebitNotify.getRetry());
myDebitAck.setVersion(myDebitNotify.getVersion());

myTxnAck = new TxnAck();
myTxnAck.setDebitAck(myDebitAck);
myTxnAck.setMerchantCertId(myTxnNotify.getMerchantCertId());
myTxnAck.setMerchantTxnRef(myTxnNotify.getMerchantTxnRef());
myTxnAck.setMerchantTxnStatus("0000");
myTxnAck.setMid(myTxnNotify.getMid());
myTxnAck.setNetsMid(myTxnNotify.getNetsMid());
myTxnAck.setNetsTxnRef(myTxnNotify.getNetsTxnRef());

messageS = null;
if (myTxnAck != null)
    messageS = myDebitMerchant.formAck(myTxnAck);
if (urlS != null && messageS != null)
    Response.Redirect(urlS + messageS, true);
}
catch (Exception umapiE)
{
    string tempS = umapiE.Message + "\r\n" + umapiE.StackTrace + "\r\n";
    byte[] tempBA = System.Text.Encoding.UTF8.GetBytes(tempS);
    outputFS.Write(tempBA, 0, tempBA.Length);
}

```

```

        finally
        {
            outputFS.Close();
        }
    }
    catch (Exception ex)
    {
        Response.Write(ex.Message + "<br/>" + ex.StackTrace + "<br/>");
    }
}
%>

```

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

```

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div></div>
    </form>
</body>
</html>

```

#### 4.2.1.4.3 Debit Browser Redirection

**Success:** When the transaction had been approved by the bank, it will redirect to the merchant's success page.

**Failure:** When the bank rejects the transaction or incorrect information is entered, it will redirect to the merchant's failure page.

**Cancel:** Click the 'Cancel' button at the page, it will cancel the transaction and redirect to the merchant's cancel page.

```

<%@ Page Language="C#" AutoEventWireup="true" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi.data" %>
<%
    try
    {
        string messageS = Request.Form["message"];
        try
        {
            DebitMerchant myDebitMerchant = null;
            TxnRes myTxnRes = null;
            DebitTxnRes myDebitTxnRes = null;
            myDebitMerchant = (DebitMerchant)Merchant.getInstance(Merchant.MERCHANT_TYPE_DEBIT,
"c:\\umapi\\etc");

            myTxnRes = myDebitMerchant.unpackResponse(messageS);
            myDebitTxnRes = myTxnRes.getDebitTxnRes();
            nets_mid.Text = myTxnRes.getNetsMid();
            merchant_txn_ref.Text = myTxnRes.getMerchantTxnRef();
            nets_txn_ref.Text = myTxnRes.getNetsTxnRef();
            nets_txn_dtm.Text = myTxnRes.getNetsTxnDtm();
            payment_mode.Text = myTxnRes.getPaymentMode();
            nets_txn_status.Text = myTxnRes.getNetsTxnStatus();
            nets_txn_resp_code.Text = myTxnRes.getNetsTxnRespCode();
            nets_txn_msg.Text = myTxnRes.getNetsTxnMsg();
            ArrayList customAttributeAL = myTxnRes.getCustomAttribute();

```



```

        if (customAttributeAL != null)
        {
            for (int i = 0; i < customAttributeAL.Count; i++)
            {
                custom_attribute.Text += ((CustomAttribute)customAttributeAL[i]).getKey() + ": " +
                ((CustomAttribute)customAttributeAL[i]).getValue() + "<br/>";
            }
        }

        bank_id.Text = myDebitTxnRes.getBankId();
        bank_ref_code.Text = myDebitTxnRes.getBankRefCode();
        bank_remarks.Text = myDebitTxnRes.getBankRemarks();
        bank_status.Text = myDebitTxnRes.getBankStatus();
        bank_time_zone.Text = myDebitTxnRes.getBankTimeZone();
        bank_txn_date.Text = myDebitTxnRes.getBankTxnDate();
        bank_txn_time.Text = myDebitTxnRes.getBankTxnTime();
        func_dest_exchange_rate.Text = myDebitTxnRes.getFuncDestExchangeRate();
        gw_time_zone.Text = myDebitTxnRes.getGwTimeZone();
        gw_txn_date.Text = myDebitTxnRes.getGwTxnDate();
        gw_txn_time.Text = myDebitTxnRes.getGwTxnTime();
        gw_value_date.Text = myDebitTxnRes.getGwValueDate();
        src_func_exchange_rate.Text = myDebitTxnRes.getSrcFuncExchangeRate();
        txn_dest_amt.Text = myDebitTxnRes.getTxnDestAmt();
        txn_dest_currency_code.Text = myDebitTxnRes.getTxnDestCurrencyCode();
        txn_fun_currency_code.Text = myDebitTxnRes.getTxnFunCurrencyCode();
        txn_func_amt.Text = myDebitTxnRes.getTxnFuncAmt();
        txn_src_amt.Text = myDebitTxnRes.getTxnSrcAmt();
        txn_src_currency_code.Text = myDebitTxnRes.getTxnSrcCurrencyCode();
        retry.Text = myDebitTxnRes.getRetry();
        version.Text = myDebitTxnRes.getVersion();
        param1.Text = myDebitTxnRes.getParam1();
        param2.Text = myDebitTxnRes.getParam2();
        param3.Text = myDebitTxnRes.getParam3();
        param4.Text = myDebitTxnRes.getParam4();
        param5.Text = myDebitTxnRes.getParam5();
        param11.Text = myDebitTxnRes.getParam11();
        param12.Text = myDebitTxnRes.getParam12();
        param13.Text = myDebitTxnRes.getParam13();
        param14.Text = myDebitTxnRes.getParam14();
        param15.Text = myDebitTxnRes.getParam15();
    }
    catch (Exception umapiE)
    {
        Response.Write(umapiE.Message + "<br/>" + umapiE.StackTrace + "<br/>");
    }
}
catch (Exception e)
{ ; }
%>

```

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

```

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>UMAPI .NET Test</title>
</head>
<body>
    <div>
        <b>DIRECT DEBIT BROWSER SUBMISSION SUCCESS URL</b>
    </div>
    <form id="form1" runat="server">
    <div>

```



```

<table>
  <tr><td colspan="3"><b>Success URL parameters:</b></td></tr>
  <tr>
    <td style="width:200px">Author</td>
    <td style="width:450px"><asp:Label runat="server" ID="author" /></td>
    <td style="width:100px"></td>
  </tr>
  <tr><td colspan="3">&nbsp;</td></tr>
  <tr><td colspan="3"><b>TxnRes parameters:</b></td></tr>
  <tr>
    <td>NETS MID</td>
    <td><asp:Label runat="server" ID="nets_mid" /></td>
    <td></td>
  </tr>
  <tr>
    <td>Merchant Transaction Reference</td>
    <td><asp:Label runat="server" ID="merchant_txn_ref" /></td>
    <td></td>
  </tr>
  <tr>
    <td>NETS Transaction Reference</td>
    <td><asp:Label runat="server" ID="nets_txn_ref" /></td>
    <td></td>
  </tr>
  <tr>
    <td>NETS Transaction Date Time</td>
    <td><asp:Label runat="server" ID="nets_txn_dtm" /></td>
    <td></td>
  </tr>
  <tr>
    <td>Payment Mode</td>
    <td><asp:Label runat="server" ID="payment_mode" /></td>
    <td></td>
  </tr>
  <tr>
    <td>NETS Transaction Status</td>
    <td><asp:Label runat="server" ID="nets_txn_status" /></td>
    <td></td>
  </tr>
  <tr>
    <td>NETS Transaction Response Code</td>
    <td><asp:Label runat="server" ID="nets_txn_resp_code" /></td>
    <td></td>
  </tr>
  <tr>
    <td>NETS Transaction Message</td>
    <td><asp:Label runat="server" ID="nets_txn_msg" /></td>
    <td></td>
  </tr>
  <tr>
    <td>Custom Attribute</td>
    <td><asp:Label runat="server" ID="custom_attribute" /></td>
    <td></td>
  </tr>
  <tr><td colspan="3">&nbsp;</td></tr>
  <tr><td colspan="3"><b>DebitTxnRes parameters:</b></td></tr>
  <tr>
    <td>Transaction Currency Code</td>
    <td><asp:Label runat="server" ID="txn_src_currency_code" /></td>
    <td></td>
  </tr>
  <tr>
    <td>Transaction Amount</td>

```

```

        <td><asp:Label runat="server" ID="txn_src_amt" /></td>
        <td></td>
    </tr>
    <tr>
        <td>Bank ID</td>
        <td><asp:Label runat="server" ID="bank_id" /></td>
        <td></td>
    </tr>
    <tr>
        <td>Bank Reference Code</td>
        <td><asp:Label runat="server" ID="bank_ref_code" /></td>
        <td></td>
    </tr>
    <tr>
        <td>Bank Remarks</td>
        <td><asp:Label runat="server" ID="bank_remarks" /></td>
        <td></td>
    </tr>
    <tr>
        <td>Bank Status</td>
        <td><asp:Label runat="server" ID="bank_status" /></td>
        <td></td>
    </tr>
    <tr>
        <td>Bank Currency Code</td>
        <td><asp:Label runat="server" ID="txn_fun_currency_code" /></td>
        <td></td>
    </tr>
    <tr>
        <td>Bank Exchange Rate</td>
        <td><asp:Label runat="server" ID="src_func_exchange_rate" /></td>
        <td></td>
    </tr>
    <tr>
        <td>Bank Transaction Amount</td>
        <td><asp:Label runat="server" ID="txn_func_amt" /></td>
        <td></td>
    </tr>
    <tr>
        <td>Bank Time Zone</td>
        <td><asp:Label runat="server" ID="bank_time_zone" /></td>
        <td></td>
    </tr>
    <tr>
        <td>Bank Transaction Date</td>
        <td><asp:Label runat="server" ID="bank_txn_date" /></td>
        <td></td>
    </tr>
    <tr>
        <td>Bank Transaction Time</td>
        <td><asp:Label runat="server" ID="bank_txn_time" /></td>
        <td></td>
    </tr>
    <tr>
        <td>NETS Currency Code</td>
        <td><asp:Label runat="server" ID="txn_dest_currency_code" /></td>
        <td></td>
    </tr>
    <tr>
        <td>NETS Exchange Rate</td>
        <td><asp:Label runat="server" ID="func_dest_exchange_rate" /></td>
        <td></td>
    </tr>

```

```

<tr>
  <td>NETS Transaction Amount</td>
  <td><asp:Label runat="server" ID="txn_dest_amt" /></td>
</tr>
<tr>
  <td>NETS Time Zone</td>
  <td><asp:Label runat="server" ID="gw_time_zone" /></td>
</tr>
<tr>
  <td>NETS Transaction Date</td>
  <td><asp:Label runat="server" ID="gw_txn_date" /></td>
</tr>
<tr>
  <td>NETS Transaction Time</td>
  <td><asp:Label runat="server" ID="gw_txn_time" /></td>
</tr>
<tr>
  <td>NETS Value Date</td>
  <td><asp:Label runat="server" ID="gw_value_date" /></td>
</tr>
<tr>
  <td>Retry</td>
  <td><asp:Label runat="server" ID="retry" /></td>
</tr>
<tr>
  <td>Version</td>
  <td><asp:Label runat="server" ID="version" /></td>
</tr>
<tr>
  <td>Param 1</td>
  <td><asp:Label runat="server" ID="param1" /></td>
</tr>
<tr>
  <td>Param 2</td>
  <td><asp:Label runat="server" ID="param2" /></td>
</tr>
<tr>
  <td>Param 3</td>
  <td><asp:Label runat="server" ID="param3" /></td>
</tr>
<tr>
  <td>Param 4</td>
  <td><asp:Label runat="server" ID="param4" /></td>
</tr>
<tr>
  <td>Param 5</td>
  <td><asp:Label runat="server" ID="param5" /></td>
</tr>
<tr>
  <td>Param 11</td>
  <td><asp:Label runat="server" ID="param11" /></td>

```

```

        <td></td>
    </tr>
    <tr>
        <td>Param 12</td>
        <td><asp:Label runat="server" ID="param12" /></td>
        <td></td>
    </tr>
    <tr>
        <td>Param 13</td>
        <td><asp:Label runat="server" ID="param13" /></td>
        <td></td>
    </tr>
    <tr>
        <td>Param 14</td>
        <td><asp:Label runat="server" ID="param14" /></td>
        <td></td>
    </tr>
    <tr>
        <td>Param 15</td>
        <td><asp:Label runat="server" ID="param15" /></td>
        <td></td>
    </tr>
</table>
</div>
</form>
</body>
</html>

```

## 4.2.2 UMID Transactions

### 4.2.1.5 Java Version

#### 4.2.1.5.1 UMID Payment Request

```

<%@ page import="com.wiz.enets2.transaction.util.*,
    com.wiz.enets2.transaction.umapi.data.*,
    com.wiz.enets2.transaction.umapi.*,
    org.apache.commons.lang.*,
    java.util.*"%>

<%
    TxnReq req = new TxnReq();
    CreditTxnReq credReq = new CreditTxnReq();

    String mid = request.getParameter("mid");
    String tid = request.getParameter("tid");
    String paymentMode = request.getParameter("payment_mode");
    String amt = request.getParameter("txn_amount");
    String currency = request.getParameter("currency_code");
    String merRef = request.getParameter("merchant_txn_ref");
    String submitMode = request.getParameter("submission_mode");
    String merCertId = request.getParameter("merchant_cert_id");
    String pan = request.getParameter("pan");
    String expiry = request.getParameter("expiry_date");
    String stan = request.getParameter("stan");
    String paymentType = request.getParameter("payment_type");

    String successURL = request.getParameter("success_url");
    String successURLParams = request.getParameter("success_url_params");
    String failureURL = request.getParameter("failure_url");
    String failureURLParams = request.getParameter("failure_url_params");
    String notifyURL = request.getParameter("notify_url");
    String notifyURLParams = request.getParameter("notify_url_params");

```

```
String postURL          = request.getParameter("post_url");
String postURLParams = request.getParameter("post_url_params");
String cancelURL       = request.getParameter("cancel_url");
String cancelURLParams = request.getParameter("cancel_url_params");
```

```
//Start: For RPP only
```

```
String param1 = request.getParameter("param1");
String param2 = request.getParameter("param2");
String param3 = request.getParameter("param3");
String param4 = request.getParameter("param4");
String param5 = request.getParameter("param5");
//End : For RPP only
```

```
req.setNetsMid(mid);
req.setTid(tid);
req.setPaymentMode(paymentMode);
req.setTxnAmount(amt);
req.setCurrencyCode(currency);
req.setMerchantTxnRef(merRef);
req.setSubmissionMode(submitMode);
req.setMerchantCertId(merCertId);
req.setSuccessUrl(successURL);
if (successURLParams != null && !successURLParams.equals("")) {
    req.setSuccessUrlParams(successURLParams);
}
```

```
req.setFailureUrl(failureURL);
if (failureURLParams != null && !failureURLParams.equals("")) {
    req.setFailureUrlParams(failureURLParams);
}
```

```
if (notifyURL != null)
    req.setNotifyUrl(notifyURL);
```

```
if (notifyURLParams != null)
    req.setNotifyUrlParams(notifyURLParams);
```

```
// =====
credReq.setPan(pan);
credReq.setExpiryDate(expiry);
credReq.setPaymentType(paymentType);
credReq.setStan(stan);
credReq.setCardHolderName(request.getParameter("name"));
credReq.setCvv(request.getParameter("cvv"));
if (postURL != null)
    credReq.setPostUrl(postURL);
if (postURLParams != null)
    credReq.setPostUrlParams(postURLParams);

if (cancelURL != null)
    credReq.setCancelUrl(cancelURL);
if (cancelURLParams != null)
    credReq.setCancelUrlParams(cancelURLParams);
```

```
//Start : For RPP only
```

```
credReq.setParam1(param1);
credReq.setParam2(param2);
credReq.setParam3(param3);
credReq.setParam4(param4);
credReq.setParam5(param5);
//End : For RPP only
```

```

EzProtect prot = new EzProtect();
prot.setBillFirstName(request.getParameter("bill_first_name"));
prot.setBillLastName(request.getParameter("bill_last_name"));
prot.setBillInitial(request.getParameter("bill_initial"));
prot.setBillAddr1(request.getParameter("bill_addr1"));
prot.setBillAddr2(request.getParameter("bill_addr2"));
prot.setBillCoyName(request.getParameter("bill_coy_name"));
prot.setBillCity(request.getParameter("bill_city"));
prot.setBillState(request.getParameter("bill_state"));
prot.setBillZipCode(request.getParameter("bill_zip_code"));
prot.setBillCountry(request.getParameter("bill_country"));
prot.setBillMobileNum(request.getParameter("bill_mobile_num"));
prot.setBillPhoneNum(request.getParameter("bill_phone_num"));
prot.setBillFaxNum(request.getParameter("bill_fax_num"));
prot.setBillEmail(request.getParameter("bill_email"));
prot.setShipFirstName(request.getParameter("ship_first_name"));
prot.setShipLastName(request.getParameter("ship_last_name"));
prot.setShipInitial(request.getParameter("ship_initial"));
prot.setShipAddr1(request.getParameter("ship_addr1"));
prot.setShipAddr2(request.getParameter("ship_addr2"));
prot.setShipCoyName(request.getParameter("ship_coy_name"));
prot.setShipCity(request.getParameter("ship_city"));
prot.setShipState(request.getParameter("ship_state"));
prot.setShipZipCode(request.getParameter("ship_zip_code"));
prot.setShipCountry(request.getParameter("ship_country"));
prot.setShipMobileNum(request.getParameter("ship_mobile_num"));
prot.setShipPhoneNum(request.getParameter("ship_phone_num"));
prot.setShipFaxNum(request.getParameter("ship_fax_num"));
prot.setShipEmail(request.getParameter("ship_email"));
prot.setShopperIpAddr(request.getParameter("shopper_id_addr"));
prot.setProductFormat(request.getParameter("product_format"));

String prdDetails = request.getParameter("product_details");
if (prdDetails != null && !prdDetails.equals(""))
{
    ProductDetails detail = null;
    java.util.StringTokenizer st1 = new java.util.StringTokenizer (prdDetails, "|");
    System.out.println("$$$$$$$ " + prdDetails);
    int max = Integer.parseInt(st1.nextToken());
    System.out.println("$$$$$$$ max=" + max);
    int count = 0;
    ArrayList detailLst = new ArrayList (max);
    while (count < max)
    {
        System.out.println("$$$$$$$ in while");
        detail = new ProductDetails();
        detail.setName(st1.nextToken());
        System.out.println("$$$$$$$ name = " + detail.getName());
        detail.setSku(st1.nextToken());
        System.out.println("$$$$$$$ sku = " + detail.getSku());
        detail.setPrice(st1.nextToken());
        System.out.println("$$$$$$$ price = " + detail.getPrice());
        detail.setQuantity(Integer.parseInt(st1.nextToken()));
        System.out.println("$$$$$$$ quantity = " + detail.getQuantity());
        detailLst.add(detail);
        System.out.println("$$$$$$$ after add");
        count++;
    }
    prot.setProductDetails(detailLst);
    System.out.println("$$$$$$$ after set product details");
}

```

```

credReq.setEzProtect(prot);
    System.out.println("$$$$$$$ after set product ezprotect");
req.setCreditTxnReq(credReq);

    DebitTxnReq debitReq = new DebitTxnReq();

    String mid2 = request.getParameter("mid");
    String tid2 = request.getParameter("tid");
    String amt2 = request.getParameter("txn_amount");
    String currency2 = request.getParameter("currency_code");
    String merRef2 = request.getParameter("merchant_txn_ref");
    String merCertId2 = request.getParameter("merchant_cert_id");
    String txnDate2 = request.getParameter("merchant_txn_date");
    String txnTime2 = request.getParameter("merchant_txn_time");
    String timeZone2 = request.getParameter("merchant_time_zone");
    String countryCode = request.getParameter("merchant_country_code");
    String param6 = request.getParameter("param1");
    String param7 = request.getParameter("param2");
    String param8 = request.getParameter("param3");
    String param9 = request.getParameter("param4");
    String param10 = request.getParameter("param5");

    req.setNetsMid(mid2);
    req.setTid(tid2);
    req.setTxnAmount(amt2);
    req.setCurrencyCode(currency2);
    req.setMerchantTxnRef(merRef2);
    req.setMerchantCertId(merCertId2);
    debitReq.setParam1(param6);
    debitReq.setParam2(param7);
    debitReq.setParam3(param8);
    debitReq.setParam4(param9);
    debitReq.setParam5(param10);

    debitReq.setMerchantTxnDate(txnDate2);
    debitReq.setMerchantTxnTime (txnTime2);
    debitReq.setMerchantTimeZone (timeZone2);
    debitReq.setMerchantCountryCode (countryCode);
    req.setDebitTxnReq(debitReq);

```

```

    UMIDMerchant m = UMIDMerchant.getInstance();
%>

<html>
<head></head>
<%
    if (submitMode.equalsIgnoreCase("B"))
    {
        String sMsg = m.formPayReq(req);
        String url = StringEscapeUtils.escapeHtml((String)request.getParameter("gw_url"));
%>
<body onLoad="document.txnForm.submit()">
<form name="txnForm" action="<%=url%>" method="POST">
<input type="hidden" name="message" value="<%=sMsg%>">
</form>
</body>

<%
    }
    else
    {
        String url = request.getParameter("gw_url");

```

```

TxnRes res = m.doPayment(req);
CreditTxnRes credRes = res.getCreditTxnRes();
if (credRes == null)
    credRes = new CreditTxnRes();
%>
<body>
<table width="100%">
<tr colspan="2"> eNETS II Merchant Simulator Server Submission Result </tr>
<tr> <td>MID: </td> <td><input type="text" name="mid" value="<%=res.getMid()%>"/></td> </tr>
<tr> <td>Merchant Txn Ref: </td> <td><input type="text" name="merchant_txn_ref"
value="<%=res.getMerchantTxnRef()%>"/></td> </tr>
<tr> <td>NETS Txn Ref: </td> <td><input type="text" name="nets_txn_ref"
value="<%=res.getNetsTxnRef()%>"/></td> </tr>
<tr> <td>NETS Txn Time: </td> <td><input type="text" name="nets_txn_dtm"
value="<%=res.getNetsTxnDtm()%>"/></td> </tr>
<tr> <td>NETS Txn Status: </td> <td><input type="text" name="nets_txn_status"
value="<%=res.getNetsTxnStatus()%>"/></td> </tr>
<tr> <td>NETS Txn Response Code: </td> <td><input type="text" name="nets_txn_resp_code"
value="<%=res.getNetsTxnRespCode()%>"/></td> </tr>
<tr> <td>NETS Txn Msg: </td> <td><input type="text" name="nets_txn_msg"
value="<%=res.getNetsTxnMsg()%>"/></td> </tr>
<tr> <td>NETS Amt Deducted: </td> <td><input type="text" name="nets_amt_deducted"
value="<%=credRes.getNetsAmountDeducted()%>"/></td> </tr>
<tr> <td>Bank Auth ID: </td> <td><input type="text" name="bank_auth_id"
value="<%=credRes.getBankAuthId()%>"/></td> </tr>
</table>
</body>
<%
}
%>
</html>

```

## 5 Response codes

ENETS Gateway will return the response code for the transaction in field netsTxnRespCode of the Payment Response Message.

For Credit transactions, the response code can either be a two digit number returned from the merchant's acquiring host or a four digit number which indicates an error response from ENETS Gateway.

For the online transaction status query feature, the response returned to the merchant is in the format of <stage>\_<responsecode>.

### Credit Transaction Stages

Module	Stage Code	Stage
UMI	2001	STAGE_UMI_TRANSLATOR: From merchant to ENETS
	2002	STAGE_UMI_PAYMENT_LISTENER: From merchant to ENETS, stage at which the server decrypts the incoming message and verifies the merchant signature
	2003	STAGE_UMI_MASTER_COLLECTION: Master Collection stage
	2004	STAGE_UMI_MERCHANT_COLLECTION: Merchant Collection stage
	2005	STAGE_UMI_PAYMENT: stage at which the txn req message has been successfully validated.
	2006	STAGE_UMI_VALIDATION: validation of txn req message
CREDIT	1001	STAGE_CREDIT_VALIDATION: Validation of credit related parameters
	1002	STAGE_CREDIT_TO_COMMS: From ENETS to HOST
	1003	STAGE_CREDIT_FROM_COMMS: From HOST to ENETS
UMID	3001	STAGE_UMID_MESSAGE: UMID Message received
	3002	STAGE_UMID_MESSAGE: Selected Service received



## 5.1 Debit Transaction Stages

Module	Stage Code	Stage
	000001	Payment Request from ENETS2 to Bank Host
	000002	Notification from Bank Host to eNETS2
	000003	Acknowledgement from eNETS2 to Bank Host
	000004	Txn End from Bank Host to eNETS2
	000005	Payment Request from Merchant to eNETS2
	000006	Notification from eNETS2 to Merchant
	000007	Acknowledgement from Merchant to eNETS2
	000008	Txn End from eNETS2 to Merchant

## 5.2 Credit Acquirer Response Codes

The following table shows the different values that will be returned by the Acquirer Host in field 39 of the host response message.

Response Code (P-39)	ISO8583 Definition
00	Approval
01	Refer to Card Issuer
02	Refer to Card Issuer's Special Condition
03	Invalid Merchant
04	Pick-up
05	Do not honour
08	Approved, Verify ID & Signature
12	Invalid transaction
13	Invalid amount
14	Invalid card number
19	Re-enter transaction
21	No Transaction
25	Unable to locate record on file
30	Format error
31	Bank not supported by switch
41	Lost card
43	Stolen card, pickup
51	Not sufficient funds
52	No cheque account
53	No savings account
54	Expired card
55	Incorrect PIN
56	No card record
58	Transaction not permitted to terminal
61	Exceeds withdrawal amount limit
63	Security violation
75	Allowable number of PIN tries exceeded
76	Invalid product code
77	Reconcile error
78	Trans. Number not found
79	Invalid CVV2/CVC2
80	Batch number not found
85	Batch not found
88	Approved, Have cm call AMEX
89	Bad terminal ID
91	Issuer or switch is inoperative
94	Duplicate transmission
95	Reconcile error. Batch upload started
96	System malfunction

**Note:** Response Code returned by different acquirer bank may have different meaning.

## 6 ENETS Credit Response Codes

Error Code(-ve)	Error Message
1000	Missing NetsMid
1001	Invalid NetsMid
1002	Invalid Merchant Id
1003	Merchant is not registered
1004	Merchant is not active
1005	Invalid Terminal Id
1006	Invalid Payment Mode
1007	Missing Txn Amount
1008	Invalid Txn Amount
1009	Invalid Currency Code
1010	Missing Merchant Reference
1011	Invalid Merchant Reference
1012	Invalid Merchant Txn Date Time
1013	Invalid Submission Mode
1014	Invalid Merchant Cert ID
1015	Invalid Credit Card Number
1016	Missing Credit Card Number
1017	Invalid Expiry Date
1018	Missing Expiry Date
1019	Invalid CVV/CVC2
1020	Invalid Card Holder Name
1021	Missing Payment Type
1022	Invalid Payment Type
1023	Invalid Success Url
1024	Invalid Success Url Params
1025	Invalid Failure Url
1026	Invalid Failure Url Params
1027	Invalid Notify Url
1028	Invalid Notify Url Params
1029	Invalid Cancel Url
1030	Invalid Cancel Url Params
1031	Invalid Post Url
1032	Invalid Post Url Params
1033	Invalid Stan
1034	This page is currently UNDER CONSTRUCTION. Please try again later.
1100	Malformed XML Input
1101	Invalid XML Schema
1102	Schema Validation Error
1103	PGP Error
1104	Unable to Sign With PGP
1105	Unable to Encrypt With PGP
1106	Unable to Decrypt With PGP
1107	Unable to Verify Signature With PGP
1108	Invalid PGP Signature
1109	XML Utility Error
1110	Unable to Generate Transaction Reference
1111	Merchant Not Subscribing to This Service
1112	Payment Mode Not Supported
1113	Unable to Decode
1114	Unable to Encode
1115	Unable to Format Merchant Message
1116	Invalid Certificate ID
1117	Merchant not subscribed to recurring payment service
1200	Invalid login
1201	Error retrieving consumer info
1202	Credit Card Number not allowed
1203	Credit Card Expired

1204	Transaction Amount not within allowed range
1205	Reversal Amount not equal to original amount
1206	Capture Amount more than original Authorization amount
1207	Credit Amount more than original Sale/Capture amount
1208	Transaction Not Allowed
1209	Batch Close in Progress
1210	Bank not active
1211	BITMAP ERROR
1212	Reversal not allowed in different batch
1213	No Mapping found for Txn Type
1214	Unknown Merchant Type
1215	Error getting HostMID
1216	No acquiring bank found
1217	No HostTID configured for merchant
1218	Host Mid not found in database
1219	Host Tid not found in database
1220	Error occurred during batch close
1221	Error instantiating processor
1234	Error instantiating acquirer
1222	Transaction does not exist
1223	Duplicate transaction (Applicable to UMID duplicate transactions also)
1224	Transaction being processed
1225	Transaction has already been reversed
1226	Transaction has already been credited
1227	Transaction has already been captured
1228	Transaction has already been charged back
1229	Transaction Locked
1230	Transaction has already been cancelled
1231	System Error
1232	System Error
1233	System Error
1234	Transaction does not exist
1290	Comm channel type not defined
1291	Comm channel info not defined
1300	Comms Timeout
1301	Comms Error
1302	Merchant bank is in maintenance.
1401	Stan does not tally with request
1402	Terminal ID does not tally with request
1403	MTID does not tally with request
1404	Missing Response code
1405	Unable to parse response
4035	Transaction Failed. Please try again.
1501	Cannot Notify Merchant. Contact eNETS
1502	Cannot Notify Merchant. Contact eNETS
1503	Cannot Notify Merchant. Contact eNETS
1504	Cannot Process Transaction. Contact Merchant
1505	Reversal failed
1600	EzProtect parameters are required
1700	Error in determining enrollment/3D transaction status
1701	Card not enrolled
1702	Authentication failed
1703	Unable to connect to ParesListener
1704	Card Type not recognized
1705	Acquirer not 3D secure enrolled
1706	Merchant not 3D secure enrolled
1707	Acquirer's password missing
1708	Acquirer's password is invalid
1709	Invalid currency code

1710	Invalid transaction data
1711	PARReq not received by ACS
1712	Can not find serial number
1713	Invalid 3D merchant Id
1714	Invalid credit card number
1715	Invalid credit card expiry date
1716	Invalid order number
1717	Invalid purchase amount
1718	Recurrence frequency is invalid
1719	Authorization number is invalid
1720	Card not supported
1721	Validation Error
1722	3D processing error. Kindly contact your issuing bank to investigate
1723	System Error during Secure3D authentication
2000	A System Error occurred
2001	A Database Error occurred
2002	Invalid Session. Please contact Merchant to check your transaction status.
2003	Txn cancelled at service selection page (UMID)

## 6.1 Debit Acquirer Response Codes

### a. Successful Transaction

Code	Error Description	Reasons	Action
00000	Transaction Successful		Merchant to proceed with fulfillment of order.

### b. Bank Related Response Codes

Code	Error Description	Reasons	Action
20001	Internal System Error	Internal processing errors such as Database unavailability and network problem, etc.	If Gateway returns this status for Notification Acknowledgement Message, the Bank will have to initiate a reversal of the transaction.
20002	Data Validation Error	Invalid data sent from either party	The bank will not proceed with the transaction and will initiate a reversal. It will inform customer the transaction is unsuccessful.
20003	XML Related Error	Either XML parsing error or invalid/missing DTD etc.	If Gateway returns this status for Notification Acknowledgement Message, DBS will have to initiate a reversal of the transaction.
20004	Transaction Failed	Transaction is not successful, due to reasons such as customers has entered invalid PIN or insufficient funds, etc.	Customers will be notified of the status and be redirected back to the Merchant page.
20005	Communication Errors	Open connection type of errors, broken pipe errors or time out errors.	For notification message, the Bank will retry a configurable number of times.
20006	Digital Certificate Errors	Either party's certificate is either expired or invalid.	The Bank/Gateway will not accept the message and if the notification message has already been sent, the Bank will initiate a reversal.
20007	Invalid digital Signature	Verification of either party's signature failed.	The Bank/Gateway will not accept the message and if the notification message has already been sent, the Bank will initiate a reversal.

Code	Error Description	Reasons	Action
20008	Session Errors	Time out due to customer inactivity.	Customer will not be allowed to proceed and will be informed that his/her session has expired. They will then be redirected back to the Merchant site.
20009	Bank API configuration error		SS should tell Banks to reconfigure API.
20010	Failed to receive acknowledgement from Gateway		SS should check Gateway
20011	No match for Gateway public certificate.		Double check Gateway public certificates.

### c. Gateway Related Response Codes

Code	Error Description	Reasons	Action
30001	Merchant not found in Merchant_Profile table	No merchant record in the merchant_profile table.	SS should Check Gateway database.
30002	Merchant Failure_URL not found in Merchant_Notify_URL table.		SS should Check Gateway database.
30003	Unable to retrieve Merchant's Public key from Merchant_Cert table.		SS should Check Gateway database.
30004	Signature Error	Unable to verify Merchant signature due to different keys used.	Merchant to check that the public key sent to the Gateway is correct.
30005	Duplicate Transaction	The payment request consists of a Merchant reference code that was used before.	Merchant to advice consumer to wait before trying again.
30006	Merchant Ref Code generated is not unique.		Merchant to advice consumer to wait before trying again.
30007	Unable to insert transaction to MerchantArchive table.		SS should check gateway database
30008	Unable to save transaction to ASPArchive table.		SS should check gateway database
30009	Unable to insert transaction into Merchant_Txn_Info and Merchant_DD_Txn_Info tables.		SS should check gateway database
30010	Unable to insert ASP Txn Time into ASP_Txn_Info table.		SS should check gateway database
30011	Unable to generate gateway signature		Merchant should check the public key of Gateway.
30012	Bank ID not found.	Transaction cancelled by user at Gateway's Bank selection page.	Merchant can show the same payment page again.
30013	Bank ID not enabled		Fill in the problem log and escalate ENETS

Code	Error Description	Reasons	Action
30014	Client-side HTTP error when notifying Merchant.		SS should check Gateway HTTP.
30015	Server-side HTTP error when notifying Merchant.		SS should check Gateway HTTP.
40001	Unable to retrieve Bank' URL from Bank_Profile table.		Check Gateway database
40002	Merchant Ref code gotten from Bank is null.		Check that BankMgr is running.
40003	Unable to retrieve Bank's Public key from Database		Check Gateway database
40004	Unable to verify Bank Signature.		The Gateway will not accept the message. Check with the respective bank
40005	Unable to save Txn End html to BankArchive table.		Check Gateway database
40006	Unable to save Txn End html to ASPArchive table.		Check Gateway database
40007	Unable to insert transaction into Log_Merchant_Trans_End table.		Check Gateway database
40008	Unable to update status in Merchant_Txn_Info table.		Check Gateway database
40009	Unable to retrieve Merchant_URL in Merchant_Notify_URL table.		Check Gateway database
40010	Duplicate Txn End	The payment request consists of a bank reference code that was used before.	Bank to advice consumer to wait before trying again.
40011	Transaction timeout	The Gateway has no response in specified time.	Check Gateway itself and link between Gateway and bank.
40012	Gateway private key has been revoked		
40013	Bank private key has been revoked		
40014	Merchant private key has been revoked.		
40099	Transaction Unsuccessful	Transaction cancelled by user at Gateway's Bank selection page.	Merchant can show the same payment page again.

**d. System / Application Related Response Codes**

Code	Error Description	Reasons	Action
50001	Registry Access Error		Fill in the problem log and escalate eNETS
50002	Database Access Error		Fill in the problem log and escalate eNETS
50003	Page forward Error		Fill in the problem log and escalate eNETS
60001	Unable to get the CRL		Fill in the problem log and escalate ENETS
60002	Unable to send e-mail to Administrator		Fill in the problem log and escalate eNETS
60003	Unable to validate the merchant Certificate		Fill in the problem log and escalate eNETS
60004	Unable to update revoked flag in database for Merchants		Check the database
60005	Unable to validate the Gateway Certificates		Fill in the problem log and escalate ENETS
60006	Unable to update revoked flag in database for Gateway		Check the database
60007	Unable to validate the Bank Certificates		Fill in the problem log and escalate ENETS
60008	Unable to update revoked flag in database for Banks		Check the database
60009	No certificate Authority ID specified		Fill in the problem log and escalate ENETS
70001	Unable to connect to the Password Server		Double check password.
70002	Invalid database password		Key into the correct password.
70003	Failed to check connection to the database		Fill in the problem log and escalate ENETS
80001	Error loading data into context		Fill in the problem log and escalate ENETS
90001	Invalid URL for reloading data into context		Fill in the problem log and escalate ENETS

**e. Merchant Related Response Codes**

Code	Error Description	Reasons	Action
1001	Wrong merchant or customer identification.		SS should check whether this merchant's records are in the database or not.
1002	Wrong bank PIN keyed-in by user.		User should key in again.
1003	Internet Banking transaction failed. Please check with the Bank concerned.		SS should check with the Bank.
1004	Session timeout.	Transaction has taken or user is inactive for too long	Customer will not be allowed to proceed and will be informed that his/her session has expired. They will then be redirected back to the Merchant site.
1005	Database access error.		SS should check Gateway database.
1006	Error in re-direction of web page.		SS should check Gateway connection with Banks or Merchants.
1007	Invalid digital signature.	Verification of Gateway signature failed	The Gateway will not accept the message.
1008	Network error.		SS should check Gateway network.

Code	Error Description	Reasons	Action
1009	Unable to connect to bank server.		SS call respective bank to check his bank.
1010	Error due to user PC shutdown unexpectedly.		Nil
1011	User cancelled transaction before completion.		Nil
1035	Invalid UMID		

**Note:** The following are complete successful debit response: 000003\_00000, 000004\_00000 and 000008\_00000.

## 7 Using UMAPI for RPP Transaction

### 7.1 Integration Details

To send an RPP transaction payment request, the merchant needs to embed the RPI details in the credit payment request message formatted through UMAPI before sending to eNETS payment collection page. There are two additional fields in the message that are used to carry the RPI details. Specifically, the two fields are Param1 and Param2.

#### 7.1.1 Java Version sample code

The following JSP code fragment, illustrates the additional requirements on formatting the credit payment request message.

```
<%@ page import="com.wiz.enets2.transaction.util.*,
                com.wiz.enets2.transaction.umapi.data.*,
                com.wiz.enets2.transaction.umapi.*,
                java.util.*"%>

<%
    TxnReq req = new TxnReq();
    CreditTxnReq credReq = new CreditTxnReq();

    .....

    req.setNetsMid(mid);           // The Nets MID assigned to merchant for online transaction
    .....

    req.setTxnAmount(amt);         // The First Payment Amount
    req.setCurrencyCode(currency); // The Currency Code of First and Subsequent Payment
    Amount

    .....

    credReq.setParam1("RPP");
    credReq.setParam2(rpiDetails);

    .....

%>
```

ENETS public key should be used in this case.

In the code sample above, "rpiDetails" is a String variable containing the Online RPI Action Message that carries the RPI details.



### 7.1.2 .NET Version sample code

The following ASP.NET code fragment, illustrates the additional requirements on formatting the credit payment request message.

```
<%@ Page Language="C#" AutoEventWireup="true" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi.data" %>
<%
    String rpiDetails = null;

...

    TxnReq myTxnReq = null;
    CreditTxnReq myCreditTxnReq = null;

...

    myTxnReq = new TxnReq();

...

    myCreditTxnReq = new CreditTxnReq();

...

    myCreditTxnReq.setParam1("RPP");
    myCreditTxnReq.setParam2(rpiDetails);

...

    myTxnReq.setCreditTxnReq(myCreditTxnReq);
    myTxnRes = myCreditMerchant.doPayment(myTxnReq);

...

%>
```

In the code sample above, "rpiDetails" is a String variable containing the Online RPI Action Message that carries the RPI details.

## 7.2 RPI Action Message Format

S/N	Field Description	Attributes	Required	Remarks
1.	RPI ID	20 AN	Y	ID uniquely identifies the new RPI.
2.	NETSMID	128 AN	Y	ID uniquely identifies the eNETS enrolled merchant.
3.	Subsequent Payment Amount	16 N	Y	Amount charged for the subsequent payment cycles after the first payment is made. In cents or the lowest currency unit.
4.	Frequency of Payment	02 N	Y	Number of frequency unit.
5.	Frequency Unit	01 N	Y	'0' denotes weekly '1' denotes monthly

6.	Number of Payment Terms	03 N	Y	Total number of payment cycles including the first payment. '0' for infinite
7.	Customer ID	20 ANS	N	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.

The merchant should take note of the following:

- Fields in the message are separated by a pipe character '|'. Hence, the character should not be used as part of the value of any field.
- Padding is not required.
- If a field is optional and the merchant does not specify any value for the field, it should be formatted as zero-length string.
- Currency Code specifies the currency for both the First Payment Amount and Subsequent Payment Amount. It is given as a standard field in UMAPI payment request message.
- Date of First Payment is taken from the date portion of field MerchantTxnDtm of UMAPI payment request message.
- First Payment Amount is taken from the field TxnAmount of UMAPI payment request message.

## 8 Using UMAPI for Recurring Transaction

### 8.1 Integration Details

To send an Recurring transaction payment request, the merchant needs to embed the Recurring parameter in the credit payment request message formatted through UMAPI before sending to eNETS payment collection page. The additional field in the message which represent the Recurring transaction is the field Param1.

#### 8.1.1 Java Version sample code

The following JSP code fragment, illustrates the additional requirements on formatting the credit payment request message.

```
<%@ page import="com.wiz.enets2.transaction.util.*,
                com.wiz.enets2.transaction.umapi.data.*,
                com.wiz.enets2.transaction.umapi.*,
                java.util.*"%>

<%
    TxnReq req = new TxnReq();
    CreditTxnReq credReq = new CreditTxnReq();

    .....

    req.setNetsMid(mid);           // The Nets MID assigned to merchant for online transaction

    .....

    req.setTxnAmount(amt);         // The First Payment Amount
    req.setCurrencyCode(currency); // The Currency Code of First and Subsequent Payment
    Amount

    .....

    credReq.setParam1("RECURRING");
```

.....  
%>

## 9 Query Txn Status

UMAPI has the ability to allow merchants to query the status of a transaction previous done.

### 9.1 Sample Codes

#### 9.1.1 Java Version

test.jsp is merchant simulator page to collection consumer, order details. After collection all the data, it will be redirected to process.jsp to form the transaction request to ENETS payment gateway.

Sample code:

```
<%@ page import="com.wiz.enets2.transaction.umapi.NETSConfig,
    java.util.Date,
    java.text.SimpleDateFormat" %>
<html>
<head></head>
<body>
<%
    SimpleDateFormat format = new SimpleDateFormat();
    format.applyPattern("yyyyMMdd HH:mm:ss.SSS");
%>
%>
<form name="txnForm" action="process.jsp" method="post">
<table >
<tr colspan="2"> eNETS II Merchant Simulator Page. </tr>
<tr>    <td>GW URL: </td> <td><input type="text" name="gw_url"
value="<%=NETSConfig.get("payURL")%>"/></td> </tr>
<tr>    <td>MID: </td> <td><input type="text" name="mid" value="EZGJA877IJ"/></td> </tr>
<tr>    <td>TID: </td> <td><input type="text" name="tid" value="127.0.0.1"/></td> </tr>
<tr>    <td>Payment Mode: </td> <td><select name="payment_mode"> <option
value="CC">Credit</option><option value="DD">Debit</option><option value="VA">Virtual
Account</option></select></td> </tr>
<tr>    <td>Amount: </td> <td><input type="text" name="txn_amount" value="1000"/></td> </tr>
<tr>    <td>Currency Code: </td> <td><input type="text" name="currency_code" value="SGD"/></td> </tr>
<tr>    <td>Merchant Txn Ref: </td> <td><input type="text" name="merchant_txn_ref" value="<%=
format.format(new Date()) %>"/></td> </tr>
<tr>    <td>Submission Mode: </td> <td><select name="submission_mode"><option
value="S">Server</option><option value="B" selected>Browser</option></select></td> </tr>
<tr>    <td>Merchant Cert ID: </td> <td><input type="text" name="merchant_cert_id" value="1"/></td>
</tr>
<tr>    <td>Credit Card: </td> <td><input type="text" name="pan" value="5655450000000000"/></td> </tr>
<tr>    <td>Expiry: </td> <td><input type="text" name="expiry_date" value="0712"/></td> </tr>
<tr>    <td>Payment Type: </td> <td><select name="payment_type">
<option value="SALE">SALE</option><option value="AUTH">AUTH</option>
<option value="CAPT">CAPT</option><option value="CRED">CRED</option>
<option value="RSALE">REV SALE</option><option value="RAUTH">REV AUTH</option>
<option value="RCAPT">REV CAPT</option><option value="RCRED">REV CRED</option>
</select>
</td> </tr>
<tr>    <td>Name on card: </td> <td><input type="text" name="name" value="Ah Hao"/></td> </tr>
<tr>    <td>CVV: </td> <td><input type="text" name="cvv" value="232"/></td> </tr>
<tr>    <td>Invoice: </td> <td><input type="text" name="invoice" value=""/></td> </tr>
```

```
|  |  |
| --- | --- |
| Notify URL: |  |
| Post URL: |  |
| Merchant Success URL: |  |
| Merchant Failure URL: |  |
|  | |
| <B>Enter 3DSecure Information </B> | |
| Status: |  |
| Eci: |  |
| Cavv: |  |
| Xid: |  |
|  | |
|  | |

```

The queryProcess.jsp is trying to construct the query message and send it to the eNETS query server. It gets the response from ENETS query server, and displays the status of transaction being queried.

There are two approaches to run queryProcess.jsp

- passing "GW\_txn\_Code" parameter to query with the unique eNETS transaction reference code.
- passing "mid" and "Merchant\_Txn\_Code" 2 parameters to query with both merchant id and merchant transaction code. (For UMID transaction, "mid" value should contain UMID)

Sample code:

```
<%@ page import="com.wiz.enets2.transaction.util.*,
com.wiz.enets2.transaction.umapi.data.*,
com.wiz.enets2.transaction.umapi.*,
com.wiz.enets2.transaction.util.*,
java.util.*"%>

<%
String mid = request.getParameter("mid");
String merchTxnRef = request.getParameter("Merchant_Txn_Code");
String netsRef = request.getParameter("GW_Txn_Code");

CreditMerchant m = (CreditMerchant) Merchant.getInstance (Merchant.MERCHANT_TYPE_CREDIT);
String status = null;
%>
<html>
<head></head>
<body>
<%
if(mid != null && merchTxnRef != null && netsRef == null){
status = m.queryTxnStatus(mid, merchTxnRef);
String[] statusList = status.split(";");
%>
<table>
<tr> <td>Merchant_Txn_Code: </td> <td><%=merchTxnRef%></td> </tr>
<tr> <td>MID: </td> <td><%=mid%></td> </tr>
<%
for(int i=0; i< statusList.length; i++){
String statusInd = statusList[i];
%>
<tr> <td>Status: </td> <td><%=statusInd%></td> </tr>
<%
}
%>
</table>
<%
}else if(netsRef != null && mid == null && merchTxnRef == null){
status = m.queryTxnStatus(netsRef);
%>
<table>
<tr> <td>GW_Txn_Code: </td> <td><%=netsRef%></td> </tr>
<tr> <td>Status: </td> <td><%=status%></td> </tr>
</table>
<% }else{ %>
<table>
<tr> <td>Status: </td> <td>UNKNOWN</td> </tr>
</table>
<%} %>
</body>
</html>
```

## 9.1.2 .NET Version

### 9.1.2.1 eNETS Transaction Request/Response

```
<%@ Page Language="C#" AutoEventWireup="true" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi.data" %>
<%
try
{
CreditMerchant myCreditMerchant = null;
```

```

        myCreditMerchant = (CreditMerchant)Merchant.getInstance(Merchant.MERCHANT_TYPE_CREDIT,
"c:\\umapi\\etc");
        string responseS = myCreditMerchant.queryTxnStatus(Request.Form["nets_txn_ref"]);
        response.Text = responseS;
    }
    catch (Exception umapiE)
    {
        Response.Write(umapiE.Message + "<br/>" + umapiE.StackTrace + "<br/>");
    }
}
%>

```

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

```

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>UMAPI .NET Test</title>
</head>
<body>
    <div>
        <b>QUERY TRANSACTION STATUS RESPONSE</b>
    </div>
    <form id="form1" runat="server">
        <div>
            <table>
                <tr>
                    <td style="width:200px">Transaction Status</td>
                    <td style="width:450px"><asp:Label runat="server" ID="response" Text="" /></td>
                    <td style="width:100px"></td>
                </tr>
            </table>
        </div>
    </form>
</body>
</html>

```

### 9.1.2.2 Merchant Transaction Request/Response

```

<%@ Page Language="C#" AutoEventWireup="true" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi" %>
<%@ Import Namespace="com.wiz.enets2.transaction.umapi.data" %>
<%
    try
    {
        CreditMerchant myCreditMerchant = null;
        myCreditMerchant = (CreditMerchant)Merchant.getInstance(Merchant.MERCHANT_TYPE_CREDIT,
"c:\\umapi\\etc");
        string responseS = myCreditMerchant.queryTxnStatus(Request.Form["nets_mid"],
Request.Form["merchant_txn_ref"]);
        response.Text = responseS;
    }
    catch (Exception umapiE)
    {
        Response.Write(umapiE.Message + "<br/>" + umapiE.StackTrace + "<br/>");
    }
}
%>

```

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

```

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>UMAPI .NET Test</title>

```

```

</head>
<body>
  <div>
    <b>QUERY TRANSACTION STATUS RESPONSE</b>
  </div>
  <form id="form1" runat="server">
    <div>
      <table>
        <tr>
          <td style="width:200px">Transaction Status</td>
          <td style="width:450px"><asp:Label runat="server" ID="response" Text="" /></td>
          <td style="width:100px"></td>
        </tr>
      </table>
    </div>
  </form>
</body>
</html>

```

## 9.2 Output Format

The output for the queryprocess.jsp will be as follows:

<stage code>\_<error code>.

The stage code and error code can be found in section 5.1 and beyond.

## Appendix A Message XML Schema

### A.1 Root Element

```

<xs:element name="eNETS">
  <xs:annotation>
    <xs:documentation>Root element for all transaction message</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:choice>
      <xs:element name="creditReq" maxOccurs="unbounded"/>
      <xs:element name="creditNotify" maxOccurs="unbounded"/>
      <xs:element name="creditAck" maxOccurs="unbounded"/>
      <xs:element name="creditRes" maxOccurs="unbounded"/>
      <xs:element name="debitReq" maxOccurs="unbounded"/>
      <xs:element name="debitNotify" maxOccurs="unbounded"/>
      <xs:element name="debitAck" maxOccurs="unbounded"/>
      <xs:element name="debitRes" maxOccurs="unbounded"/>
      <xs:element name="vAcctReq" maxOccurs="unbounded"/>
      <xs:element name="vAcctNotify" maxOccurs="unbounded"/>
      <xs:element name="vAcctAck" maxOccurs="unbounded"/>
      <xs:element name="vAcctRes" maxOccurs="unbounded"/>
    </xs:choice>
  </xs:complexType>
</xs:element>

```

### A.2 Credit Message

#### creditReq Message

```

<xs:element name="creditReq">
  <xs:complexType>
    <xs:all>
      <xs:element name="application_type" minOccurs="0"/>
      <xs:element name="mid"/>
      <xs:element name="tid"/>
      <xs:element name="client_type"/>
      <xs:element name="customer_id_type" minOccurs="0"/>
      <xs:element name="customer_id" minOccurs="0"/>
      <xs:element name="customer_ref" minOccurs="0"/>
      <xs:element name="payment_method"/>
    </xs:all>
  </xs:complexType>
</xs:element>

```

```

    <xs:annotation>
      <xs:documentation>for credit card payment, set to 'CC'.</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="payment_type"/>
  <xs:element name="txn_amount"/>
  <xs:element name="currency_code"/>
  <xs:element name="merchant_txn_ref"/>
  <xs:element name="merchant_txn_dateTime"/>
  <xs:element name="merchant_message" minOccurs="0"/>
  <xs:element name="merchant_cert_id"/>
  <xs:element name="successURL"/>
  <xs:element name="cancelURL"/>
  <xs:element name="failureURL"/>
  <xs:element name="service_name"/>
  <xs:element name="pan" minOccurs="0"/>
  <xs:element name="expiry_date" minOccurs="0"/>
  <xs:element name="CVC2" minOccurs="0"/>
  <xs:element name="stan" minOccurs="0"/>
  <xs:element name="post_str" minOccurs="0"/>
  <xs:element name="user_id" minOccurs="0"/>
  <xs:element name="sponser_name" minOccurs="0"/>
  <xs:element name="choose_address" minOccurs="0"/>
  <xs:element name="optional" minOccurs="0"/>
  <xs:element name="optional_str" minOccurs="0"/>
  <xs:element name="param1" minOccurs="0"/>
  <xs:element name="param2" minOccurs="0"/>
  <xs:element name="param3" minOccurs="0"/>
  <xs:element name="param4" minOccurs="0"/>
  <xs:element name="param5" minOccurs="0"/>
  <xs:element name="Invoice" minOccurs="0"/>
  <xs:element name="3d" minOccurs="0">
    <xs:annotation>
      <xs:documentation>element that contains all 3D secure information</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:all>
        <xs:element name="purchase_xid"/>
        <xs:element name="tx_cavv"/>
        <xs:element name="tx_eci"/>
        <xs:element name="tx_status"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
  <!--xs:element name="merchant_signature"/ -->
  <!--xs:element name="txn_code"/ -->
</xs:all>
</xs:complexType>
</xs:element>

```

### creditNotify Message

```

<xs:element name="creditNotify">
  <xs:complexType>
    <xs:all>
      <xs:element name="application_type" minOccurs="0"/>
      <xs:element name="mid"/>
      <xs:element name="tid"/>
      <xs:element name="client_type"/>
      <xs:element name="customer_id_type" minOccurs="0"/>
      <xs:element name="customer_id" minOccurs="0"/>
      <xs:element name="payment_method">
        <xs:annotation>
          <xs:documentation>for credit card payment, set to 'CC'.</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="payment_type"/>
      <xs:element name="txn_amount"/>
      <xs:element name="currency_code"/>
      <xs:element name="merchant_txn_ref"/>
      <xs:element name="merchant_txn_dateTime"/>
      <xs:element name="merchant_message" minOccurs="0"/>
      <xs:element name="nets_txn_ref"/>
      <xs:element name="nets_txn_dateTime"/>
      <xs:element name="nets_txn_status"/>
    </xs:all>
  </xs:complexType>
</xs:element>

```



```

<xs:element name="nets_vcard_balance" minOccurs="0"/>
<xs:element name="nets_risk_score" minOccurs="0"/>
<xs:element name="bank_auth_id" minOccurs="0"/>
<xs:element name="param1" minOccurs="0"/>
<xs:element name="param2" minOccurs="0"/>
<xs:element name="param3" minOccurs="0"/>
<xs:element name="param4" minOccurs="0"/>
<xs:element name="param5" minOccurs="0"/>
<!--xs:element name="nets_signature"/ -->
<!--xs:element name="txn_code"/ -->
</xs:all>
</xs:complexType>
</xs:element>

```

### creditAck Message

```

<xs:element name="creditAck">
  <xs:complexType>
    <xs:all>
      <xs:element name="mid"/>
      <xs:element name="application_type" minOccurs="0"/>
      <xs:element name="merchant_txn_ref"/>
      <xs:element name="merchant_txn_status"/>
      <xs:element name="merchant_cert_id"/>
      <xs:element name="nets_txn_ref"/>
      <xs:element name="param1" minOccurs="0"/>
      <xs:element name="param2" minOccurs="0"/>
      <xs:element name="param3" minOccurs="0"/>
      <xs:element name="param4" minOccurs="0"/>
      <xs:element name="param5" minOccurs="0"/>
      <!--xs:element name="merchant_signature"/-->
    </xs:all>
  </xs:complexType>
</xs:element>

```

### creditRes Message

```

<xs:element name="creditRes">
  <xs:complexType>
    <xs:all>
      <xs:element name="merchant_txn_ref"/>
      <xs:element name="nets_txn_ref"/>
      <xs:element name="nets_txn_dateTime"/>
      <xs:element name="nets_txn_status"/>
      <xs:element name="nets_txn_resp_code"/>
      <xs:element name="nets_txn_message"/>
      <xs:element name="nets_amount_deducted"/>
      <xs:element name="nets_vcard_balance"/>
      <xs:element name="nets_risk_score"/>
      <xs:element name="bank_auth_id"/>
      <!--xs:element name="nets_signature"/-->
      <!--xs:element name="txn_code"/ -->
    </xs:all>
  </xs:complexType>
</xs:element>

```

## A.3 Debit Message:

### debitReq Message

```

<xs:element name="debitReq">
  <xs:complexType>
    <xs:all>
      <xs:element name="mid"/>
      <xs:element name="tid"/>
      <xs:element name="payment_method">
        <xs:annotation>
          <xs:documentation>for credit card payment, set to 'DD'.</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="txn_amount"/>
      <xs:element name="currency_code"/>
      <xs:element name="merchant_txn_ref"/>
      <xs:element name="merchant_time_zone"/>
      <xs:element name="merchant_country_code"/>
    </xs:all>
  </xs:complexType>
</xs:element>

```

```

<xs:element name="retry"/>
<xs:element name="param1" minOccurs="0"/>
<xs:element name="param2" minOccurs="0"/>
<xs:element name="param3" minOccurs="0"/>
<xs:element name="param4" minOccurs="0"/>
<xs:element name="param5" minOccurs="0"/>
<xs:element name="merchant_cert_id"/>
<!--xs:element name="merchant_signature"/-->
<!--xs:element name="txn_code"/ -->
</xs:all>
</xs:complexType>
</xs:element>

```

### debitNotify Message

```

<xs:element name="debitNotify">
  <xs:complexType>
    <xs:all>
      <xs:element name="mid"/>
      <xs:element name="tid"/>
      <xs:element name="merchant_txn_ref"/>
      <xs:element name="currency_code"/>
      <xs:element name="txn_amount"/>
      <xs:element name="retry"/>
      <xs:element name="bank_id"/>
      <xs:element name="bank_ref_code"/>
      <xs:element name="bank_remarks"/>
      <xs:element name="bank_txn_dateTime"/>
      <xs:element name="bank_time_zone"/>
      <xs:element name="bank_status"/>
      <xs:element name="bank_cert_id"/>
      <xs:element name="nets_txn_ref"/>
      <xs:element name="nets_txn_base_currency_code"/>
      <xs:element name="nets_txn_base_amount"/>
      <xs:element name="nets_exchange_rate"/>
      <xs:element name="nets_txn_dateTime"/>
      <xs:element name="nets_value_date"/>
      <xs:element name="nets_time_zone"/>
      <xs:element name="nets_txn_status"/>
      <xs:element name="nets_cert_id"/>
      <xs:element name="param1" minOccurs="0"/>
      <xs:element name="param2" minOccurs="0"/>
      <xs:element name="param3" minOccurs="0"/>
      <xs:element name="param4" minOccurs="0"/>
      <xs:element name="param5" minOccurs="0"/>
      <!--xs:element name="bank_signature"/-->
      <!--xs:element name="nets_signature"/ -->
    </xs:all>
  </xs:complexType>
</xs:element>

```

### debitAck Message

```

<xs:element name="debitAck">
  <xs:complexType>
    <xs:all>
      <xs:element name="mid"/>
      <xs:element name="merchant_txn_ref"/>
      <xs:element name="merchant_txn_status"/>
      <xs:element name="retry"/>
      <xs:element name="merchant_cert_id"/>
      <xs:element name="nets_txn_ref"/>
      <xs:element name="param1" minOccurs="0"/>
      <xs:element name="param2" minOccurs="0"/>
      <xs:element name="param3" minOccurs="0"/>
      <xs:element name="param4" minOccurs="0"/>
      <xs:element name="param5" minOccurs="0"/>
      <!--xs:element name="merchant_signature"/-->
    </xs:all>
  </xs:complexType>
</xs:element>

```

### debitRes Message

```

<xs:element name="debitRes">
  <xs:complexType>

```

```

<xs:all>
  <xs:element name="tid"/>
  <xs:element name="merchant_txn_ref"/>
  <xs:element name="currency_code"/>
  <xs:element name="txn_amount"/>
  <xs:element name="retry"/>
  <xs:element name="bank_id"/>
  <xs:element name="bank_ref_code"/>
  <xs:element name="bank_remarks"/>
  <xs:element name="bank_txn_dateTime"/>
  <xs:element name="bank_time_zone"/>
  <xs:element name="bank_status"/>
  <xs:element name="bank_cert_id"/>
  <xs:element name="nets_txn_ref"/>
  <xs:element name="nets_txn_base_currency_code"/>
  <xs:element name="nets_txn_base_amount"/>
  <xs:element name="nets_exchange_rate"/>
  <xs:element name="nets_txn_dateTime"/>
  <xs:element name="nets_value_date"/>
  <xs:element name="nets_time_zone"/>
  <xs:element name="nets_txn_status"/>
  <xs:element name="nets_cert_id"/>
  <xs:element name="param1" minOccurs="0"/>
  <xs:element name="param2" minOccurs="0"/>
  <xs:element name="param3" minOccurs="0"/>
  <xs:element name="param4" minOccurs="0"/>
  <xs:element name="param5" minOccurs="0"/>
  <!--xs:element name="bank_signature"/-->
  <!--xs:element name="nets_signature"/-->
</xs:all>
</xs:complexType>
</xs:element>

```

## A.4 Message Type

```

<xs:element name="application_type">
  <xs:annotation>
    <xs:documentation>merchant's particular service, assigned by eNETS tech team</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="4"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="mid">
  <xs:annotation>
    <xs:documentation>merchant ID</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="12"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="tid">
  <xs:annotation>
    <xs:documentation>terminal ID</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="10"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="client_type">
  <xs:annotation>
    <xs:documentation>identify whether the consumer uses PC or WAP phone to initiate transaction. For client type "02"
(WAP), the customer id type and customer id have to be empty at the same time or the customer id type can be 'hp' and the
customer id is filled with the customer's handphone number.</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">

```

```

<xs:length value="10"/>
<xs:enumeration value="01">
  <xs:annotation>
    <xs:documentation>Internet</xs:documentation>
  </xs:annotation>
</xs:enumeration>
<xs:enumeration value="02">
  <xs:annotation>
    <xs:documentation>WAP</xs:documentation>
  </xs:annotation>
</xs:enumeration>
<xs:enumeration value="03">
  <xs:annotation>
    <xs:documentation>Internet VCard</xs:documentation>
  </xs:annotation>
</xs:enumeration>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="customer_id_type">
  <xs:annotation>
    <xs:documentation>identify the type of customer ID that is used. Value assigned by eNETS.</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="2"/>
      <xs:enumeration value="HP"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="customer_id">
  <xs:annotation>
    <xs:documentation>for credit only. uniquely identify customer to eNETS.</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="20"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="customer_ref">
  <xs:annotation>
    <xs:documentation>identify customer to merchant. Value assigned by merchant</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="20"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="payment_method">
  <xs:annotation>
    <xs:documentation>payment method of the txn, can be credit, debit, vcard or NA. when NA, customer can choose from
payment page.</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="2"/>
      <xs:enumeration value="CC">
        <xs:annotation>
          <xs:documentation>credit card txn</xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="DD">
        <xs:annotation>
          <xs:documentation>direct debit txn</xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="VC">
        <xs:annotation>
          <xs:documentation>VCard txn</xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="NA">

```

```

        <xs:annotation>
            <xs:documentation>merchant let customer to choose payment method at eNETS collection
page.</xs:documentation>
        </xs:annotation>
    </xs:enumeration>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="payment_type">
    <xs:annotation>
        <xs:documentation>payment mode of credit card txn</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:length value="3"/>
            <xs:enumeration value="SAL">
                <xs:annotation>
                    <xs:documentation>credit card sale txn</xs:documentation>
                </xs:annotation>
            </xs:enumeration>
            <xs:enumeration value="AUT">
                <xs:annotation>
                    <xs:documentation>credit card authorization txn</xs:documentation>
                </xs:annotation>
            </xs:enumeration>
            <xs:enumeration value="CAP">
                <xs:annotation>
                    <xs:documentation>credit card capture txn</xs:documentation>
                </xs:annotation>
            </xs:enumeration>
            <xs:enumeration value="REV">
                <xs:annotation>
                    <xs:documentation>reversal txn</xs:documentation>
                </xs:annotation>
            </xs:enumeration>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="txn_amount">
    <xs:annotation>
        <xs:documentation>indicate txn amount to be debited from consumer's account. last two numbers represent decimal value.
txn amount can be less than or equal to the txn amount in original Authorization txn (partial capture).</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:length value="16"/>
            <xs:minExclusive value="0"/>
            <xs:maxExclusive value="10000000000000000"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="currency_code">
    <xs:annotation>
        <xs:documentation>identify the type of currency used</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:length value="3"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="merchant_txn_ref">
    <xs:annotation>
        <xs:documentation>unique reference code assigned by merchant for this txn. For credit card capture txn, the
merchant_txn_ref must be the same as the original merchant_txn_ref in the original authorization txn.</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:length value="30"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="merchant_txn_dateTime">
    <xs:annotation>

```

```

    <xs:documentation>txn time as recorded by merchant server</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:dateTime"/>
  </xs:simpleType>
</xs:element>
<xs:element name="merchant_message">
  <xs:annotation>
    <xs:documentation>for SMS txn only. contains any message that the merchant may want eNETS to display to
user.</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="80"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="merchant_time_zone">
  <xs:annotation>
    <xs:documentation>time zone (relative to GMT) of the merchant txn date, e.g. "+08:00"</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="6"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="merchant_country_code">
  <xs:annotation>
    <xs:documentation>string value of the country code of the merchant. 3-character string</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="3"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="retry">
  <xs:annotation>
    <xs:documentation>for debit only. the number of retries for current merchant_txn_ref.</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="99"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="merchant_cert_id">
  <xs:annotation>
    <xs:documentation>global unique identifier of the certificate being used by merchant.</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="20"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="merchant_txn_status">
  <xs:annotation>
    <xs:documentation>merchant status of the txn. used in acknowledgement message</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="5"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="successURL">
  <xs:annotation>
    <xs:documentation>a specified URL for eNETS to redirect back to merchant on successful transaction</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">

```

```

        <xs:length value="80"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="cancelURL">
    <xs:annotation>
      <xs:documentation>a specified URL for eNETS to redirect back to merchant on canceled transaction</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:length value="80"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="failureURL">
    <xs:annotation>
      <xs:documentation>a specified URL for eNETS to redirect back to merchant on failed transaction</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:length value="80"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="service_name">
    <xs:annotation>
      <xs:documentation>eNETS service name</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:all>
        <xs:element name="MOSET" type="xs:boolean" minOccurs="0"/>
        <xs:element name="3DSECURE" type="xs:boolean" minOccurs="0"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
  <xs:element name="pan">
    <xs:annotation>
      <xs:documentation>credit card number</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:length value="19"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="expiry_date">
    <xs:annotation>
      <xs:documentation>credit card expiry date</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:gYearMonth"/>
    </xs:simpleType>
  </xs:element>
  <xs:element name="CVC2">
    <xs:annotation>
      <xs:documentation>credit card CVC value</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:length value="4"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="stan">
    <xs:annotation>
      <xs:documentation>credit note number. must be unique within one batch. </xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:length value="6"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="post_str">

```

```

<xs:annotation>
  <xs:documentation>query string to append to the Merchant POST URL. </xs:documentation>
</xs:annotation>
<xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:length value="80"/>
  </xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="user_id">
  <xs:annotation>
    <xs:documentation>optional eNETS II user login ID. </xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="15"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="sponser_name">
  <xs:annotation>
    <xs:documentation>sponser name of eNETS II user (from OneTouch). </xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="30"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="choose_address">
  <xs:annotation>
    <xs:documentation>Indicate whether the shopper can choose shipping address. </xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:boolean"/>
  </xs:simpleType>
</xs:element>
<xs:element name="optional">
  <xs:annotation>
    <xs:documentation>Indicate whether the merchant has optional data to send to eNETS Credit Payment Server.
  </xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:boolean"/>
  </xs:simpleType>
</xs:element>
<xs:element name="optional_str">
  <xs:annotation>
    <xs:documentation>query string to append to the Merchant Optional URL. </xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="80"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="param1">
  <xs:annotation>
    <xs:documentation>spared for eNETS internal use</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="50"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="param2">
  <xs:annotation>
    <xs:documentation>spared for eNETS internal use</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="50"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

```



```

    </xs:simpleType>
  </xs:element>
  <xs:element name="param3">
    <xs:annotation>
      <xs:documentation>spared for eNETS internal use</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:length value="50"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="param4">
    <xs:annotation>
      <xs:documentation>spared for eNETS internal use</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:length value="50"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="param5">
    <xs:annotation>
      <xs:documentation>spared for eNETS internal use</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:length value="50"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="merchant_signature">
    <xs:annotation>
      <xs:documentation>digital signature formed by signing all the fields in the message. Serves as non-repudiation from
sender.</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:length value="176"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="Invoice">
    <xs:annotation>
      <xs:documentation>refer to visa_invoice.dtd for details, extracted from VISA document VXMLIS10.pdf.</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="purchase_xid">
    <xs:annotation>
      <xs:documentation>purchase_xid returned from mpi</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:length value="20"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="tx_cavv">
    <xs:annotation>
      <xs:documentation>tx_cavv value returned from mpi</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:length value="28"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="tx_eci">
    <xs:annotation>
      <xs:documentation>tx_eci value returned from mpi</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:string">

```

```

        <xs:length value="2"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="tx_status">
    <xs:annotation>
      <xs:documentation>tx_status returned from mpi</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:length value="1"/>
        <xs:enumeration value="Y"/>
        <xs:enumeration value="N"/>
        <xs:enumeration value="U"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="nets_txn_ref">
    <xs:annotation>
      <xs:documentation>unique reference code assigned by eNETS for this txn</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:length value="14"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="nets_txn_dateTime">
    <xs:annotation>
      <xs:documentation>transaction time as recorded by eNETS system.</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:dateTime"/>
    </xs:simpleType>
  </xs:element>
  <xs:element name="nets_txn_status">
    <xs:annotation>
      <xs:documentation>transaction status, 4-digit code.</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:length value="5"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="nets_txn_resp_code">
    <xs:annotation>
      <xs:documentation>transaction response code.</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:integer">
        <xs:enumeration value="0">
          <xs:annotation>
            <xs:documentation>Success</xs:documentation>
          </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="1">
          <xs:annotation>
            <xs:documentation>Failure</xs:documentation>
          </xs:annotation>
        </xs:enumeration>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="nets_txn_message">
    <xs:annotation>
      <xs:documentation>transaction message. e.g. Approval, Declined, etc.</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:length value="150"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

```

```

<xs:element name="nets_amount_deducted">
  <xs:annotation>
    <xs:documentation>amount deducted in cents or lowest currency unit</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minExclusive value="0"/>
      <xs:maxExclusive value="10000000000"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="nets_vcard_balance">
  <xs:annotation>
    <xs:documentation>balance in VCard</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minExclusive value="0"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="nets_txn_base_currency_code">
  <xs:annotation>
    <xs:documentation>indicate the base currency for this txn, 3-character string. e.g. SGD, USD, etc.</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="3"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="nets_txn_base_amount">
  <xs:annotation>
    <xs:documentation>total amount incurred for this txn in base currency. Fully numeric in IVSA format, it assumes the lowest
    denomination for the currency. E.g. in SGD, "1234" means S$12.34 and "12" means S$0.12.</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minExclusive value="0"/>
      <xs:maxExclusive value="10000000000000"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="nets_exchange_rate">
  <xs:annotation>
    <xs:documentation>the exchange rate to convert the txn amount to txn base amount. default to "1.00" for SGD to
    SGD.</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:decimal"/>
  </xs:simpleType>
</xs:element>
<xs:element name="nets_value_date">
  <xs:annotation>
    <xs:documentation>date on which txn will be valued by eNETS II system.</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:date"/>
  </xs:simpleType>
</xs:element>
<xs:element name="nets_time_zone">
  <xs:annotation>
    <xs:documentation>time zone (relative to GMT) of nets txn date. E.g. "+08:00"</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="6"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="nets_cert_id">
  <xs:annotation>
    <xs:documentation>global unique identifier of the certificate used by eNETS II system. </xs:documentation>
  </xs:annotation>

```

```

<xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:length value="20"/>
  </xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="nets_signature">
  <xs:annotation>
    <xs:documentation>digital signature formed by signing all the fields in the message. Servers as non-repudiation from
sender.</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="176"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="bank_auth_id">
  <xs:annotation>
    <xs:documentation>bank authorization code. For credit card txn only. May be empty for failure cases.</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="12"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="bank_id">
  <xs:annotation>
    <xs:documentation>string value of the assigned bank id. For debit.</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="4"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="bank_ref_code">
  <xs:annotation>
    <xs:documentation>bank reference code for this txn.</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="20"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="bank_remarks">
  <xs:annotation>
    <xs:documentation>remarks from the bank.</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="20"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="bank_txn_dateTime">
  <xs:annotation>
    <xs:documentation>time at which total amount is deducted from the consumer account.</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:dateTime"/>
  </xs:simpleType>
</xs:element>
<xs:element name="bank_time_zone">
  <xs:annotation>
    <xs:documentation>>time zone (relative to GMT) of bank txn date. E.g. "+08:00".</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="20"/>
    </xs:restriction>
  </xs:simpleType>

```

```

</xs:element>
<xs:element name="bank_status">
  <xs:annotation>
    <xs:documentation>>bank status of the txn</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="5"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="bank_cert_id">
  <xs:annotation>
    <xs:documentation>global unique identifier of the certificate used by bank. </xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="20"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="bank_signature">
  <xs:annotation>
    <xs:documentation>digital signature of Bank formed by signing all the fields in the message. Servers as non-repudiation
from sender.</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="200"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

```

## A.5 VISA Invoice (taken from Reference Document 1):

```

<?xml version="1.0" encoding="UTF-8"?>
<ELEMENT Invoice (InvoiceHeader, InvoiceDetails+, InvoiceSummary)>
<!ATTLIST Invoice
sectorUsageVersion CDATA #IMPLIED>
<ELEMENT InvoiceHeader (InvoiceType, InvoiceStatus, TaxTreatment, DiscountTreatment?, Invoice-
Treatment, InvoiceNumber, InvoiceDate, TaxPointDate?, Currency, Party, Party, Party*, Payment?,
PONum?, DeliveryNoteNum?, Ref*, Date*, GenText*)>
<ELEMENT InvoiceType EMPTY>
<!ATTLIST InvoiceType
stdValue (380 | 381) "380"
stdName (UNTDID:1001) "UNTDID:1001">
<ELEMENT InvoiceStatus EMPTY>
<!ATTLIST InvoiceStatus
stdValue (9 | 10 | 53) "9"
stdName (UNTDID:1225) "UNTDID:1225">
<ELEMENT TaxTreatment EMPTY>
<!ATTLIST TaxTreatment
stdValue (NIL | GIL | NLL | GLL | NON) "NLL"
stdName (VISA:TAXT) "VISA:TAXT">
<ELEMENT DiscountTreatment EMPTY>
<!ATTLIST DiscountTreatment
stdValue (UN | UG | TN) "UG"
stdName (VISA:DSCT) "VISA:DSCT">
<ELEMENT InvoiceTreatment EMPTY>
<!ATTLIST InvoiceTreatment
stdValue (P | EP | E) "P"
stdName (VISA:INVT) "VISA:INVT">
<ELEMENT InvoiceNumber (#PCDATA)>
<ELEMENT InvoiceDate (#PCDATA)>
<ELEMENT TaxPointDate (#PCDATA)>
<ELEMENT Currency EMPTY>
<!ATTLIST Currency
stdValue CDATA "USD"
stdName (ISO:4217) "ISO:4217">
<ELEMENT Party (PartyID?, Name?, Street?, PostalInfo?, Contact*, Ref*)>
<!ATTLIST Party
stdValue CDATA #REQUIRED
stdName CDATA "UNTDID:3035">

```

```

<ELEMENT PartyID (#PCDATA)>
<ELEMENT Name (Name1, Name2?, Name3?)>
<ELEMENT Name1 (#PCDATA)>
<ELEMENT Name2 (#PCDATA)>
<ELEMENT Name3 (#PCDATA)>
<ELEMENT Street (Street1, Street2?, Street3?, Street4?)>
<ELEMENT Street1 (#PCDATA)>
<ELEMENT Street2 (#PCDATA)>
<ELEMENT Street3 (#PCDATA)>
<ELEMENT Street4 (#PCDATA)>
<ELEMENT PostalInfo (City?, CountrySubEntity?, PostalCode?, Country?)>
<ELEMENT City (#PCDATA)>
<ELEMENT CountrySubEntity (#PCDATA)>
<ELEMENT PostalCode (#PCDATA)>
<ELEMENT Country (#PCDATA)>
<ELEMENT Contact (Name1?, TelNum?, EMail?, Function?)>
<ELEMENT TelNum (#PCDATA)>
<ELEMENT EMail (#PCDATA)>
<ELEMENT Function (#PCDATA)>
<ELEMENT Payment (PaymentDueDate?, PaymentTerms*, PaymentMean?, CardInfo?)>
<ELEMENT PaymentDueDate (AbsoluteDate | RelativeDate)>
<ELEMENT AbsoluteDate (#PCDATA)>
<ELEMENT RelativeDate (RefDate, TimeRelation, TypeOfPeriod, NumberOfPeriods)>
<ELEMENT RefDate EMPTY>
<!ATTLIST RefDate
stdValue CDATA "5"
stdName CDATA "UNTDID:2475">
<ELEMENT TimeRelation EMPTY>
<!ATTLIST TimeRelation
stdValue CDATA "3"
stdName CDATA "UNTDID:2009">
<ELEMENT TypeOfPeriod EMPTY>
<!ATTLIST TypeOfPeriod
stdValue CDATA "CD"
stdName CDATA "UNTDID:2151">
<ELEMENT NumberOfPeriods (#PCDATA)>
<ELEMENT PaymentTerms (PaymentTermType, (AbsoluteDate | RelativeDate), DiscountPercent)>
<ELEMENT PaymentTermType (#PCDATA)>
<!ATTLIST PaymentTermType
stdValue CDATA "22"
stdName CDATA "UNTDID:4279">
<ELEMENT PaymentMean (#PCDATA)>
<!ATTLIST PaymentMean
stdValue CDATA "ZZZ"
stdName CDATA "UNTDID:4461">
<ELEMENT CardInfo (CardNum, CardAuthCode?, CardRefNum?, CardExpirationDate?, CardType?, CardholderName?,
Ref*)>
<ELEMENT CardNum (#PCDATA)>
<ELEMENT CardAuthCode (#PCDATA)>
<ELEMENT CardExpirationDate (#PCDATA)>
<ELEMENT CardType (#PCDATA)>
<!ATTLIST CardType
stdValue CDATA "VS"
stdName CDATA "VISA:CARD">
<ELEMENT CardRefNum (#PCDATA)>
<ELEMENT CardholderName (#PCDATA)>
<ELEMENT PONum (#PCDATA)>
<ELEMENT DeliveryNoteNum (#PCDATA)>
<ELEMENT Ref (#PCDATA)>
<!ATTLIST Ref
stdValue CDATA "VA"
stdName CDATA "UNTDID:1153">
<ELEMENT Date (#PCDATA)>
<!ATTLIST Date
stdValue CDATA #REQUIRED
stdName CDATA "UNTDID:2005">
<ELEMENT GenText (#PCDATA)>
<!ATTLIST GenText
stdValue CDATA #REQUIRED
stdName CDATA "UNTDID:4451">
<ELEMENT InvoiceDetails (BaseItemDetail, UnitPrice?, POLineNum?, LineItemSubtotal?, Tax*, Line-
DiscountInfo?, Date*, SpecialCond?, Ref*, GenText*)>
<ELEMENT BaseItemDetail (LineItemNum?, SubLineItemNum?, PartNumDetail+, Quantity)>
<ELEMENT LineItemNum (#PCDATA)>

```

```

<ELEMENT SubLineItemNum (#PCDATA)>
<ELEMENT PartNumDetail ((PartNum, PartDesc?) | PartDesc)>
<!ATTLIST PartNumDetail
stdValue CDATA "VP"
stdName CDATA "UNTDID:7143">
<ELEMENT PartNum (#PCDATA)>
<ELEMENT PartDesc (#PCDATA)>
<ELEMENT Quantity (Qty, UnitOfMeasure?)>
<ELEMENT Qty (#PCDATA)>
<ELEMENT UnitOfMeasure EMPTY>
<!ATTLIST UnitOfMeasure
stdValue CDATA "EA"
stdName CDATA "UNTDID:6411">
<ELEMENT UnitPrice (#PCDATA)>
<ELEMENT POLineNum (#PCDATA)>
<ELEMENT LineItemSubtotal (#PCDATA)>
<ELEMENT Tax (TaxFunction, TaxType, TaxCategory, TaxPercent, TaxableAmount?, TaxAmount?, Location?)>
<ELEMENT TaxFunction EMPTY>
<!ATTLIST TaxFunction
stdValue CDATA "7"
stdName (UNTDID:5283) "UNTDID:5283">
<ELEMENT TaxType EMPTY>
<!ATTLIST TaxType
stdValue CDATA "VAT"
stdName (UNTDID:5153) "UNTDID:5153">
<ELEMENT TaxCategory EMPTY>
<!ATTLIST TaxCategory
stdValue CDATA #REQUIRED
stdName CDATA "UNTDID:5305">
<ELEMENT TaxPercent (#PCDATA)>
<ELEMENT TaxableAmount (#PCDATA)>
<ELEMENT TaxAmount (#PCDATA)>
<ELEMENT Location (#PCDATA)>
<ELEMENT LineDiscountInfo ((DiscountValue | DiscountPercent), UnitPricePreDiscount?)>
<ELEMENT DiscountPercent (#PCDATA)>
<ELEMENT DiscountValue (#PCDATA)>
<ELEMENT UnitPricePreDiscount (#PCDATA)>
<ELEMENT SpecialCond (#PCDATA)>
<!ATTLIST SpecialCond
stdValue CDATA "OTHER"
stdName CDATA "UNTDID:4183">
<ELEMENT InvoiceSummary (TaxSummary*, InvoiceTotals, ActualPayment*)>
<ELEMENT TaxSummary (DiscountSummary?, Tax)>
<ELEMENT DiscountSummary (LineItemTotals, QtyDiscount?, ValueDiscount?, SubTotalAfterQtyValue-
Discount, SettlementDiscountAmt?, SubTotalAfterSettDiscount?)>
<ELEMENT LineItemTotals (#PCDATA)>
<ELEMENT QtyDiscount (#PCDATA)>
<ELEMENT ValueDiscount (#PCDATA)>
<ELEMENT SubTotalAfterQtyValueDiscount (#PCDATA)>
<ELEMENT SettlementDiscountAmt (#PCDATA)>
<ELEMENT SubTotalAfterSettDiscount (#PCDATA)>
<ELEMENT InvoiceTotals (DiscountSummary?, NetValue, TaxValue?, GrossValue)>
<ELEMENT NetValue (#PCDATA)>
<ELEMENT TaxValue (#PCDATA)>
<ELEMENT GrossValue (#PCDATA)>
<ELEMENT ActualPayment (PaymentAmount, PaymentMean, PaymentDate, CardInfo?, Ref*)>
<ELEMENT PaymentAmount (LocalCurrencyAmt, ForeignCurrencyPayment?)>
<ELEMENT LocalCurrencyAmt (#PCDATA)>
<ELEMENT ForeignCurrencyPayment (ForeignCurrencyAmt, Currency)>
<ELEMENT ForeignCurrencyAmt (#PCDATA)>
<ELEMENT PaymentDate (#PCDATA)

```

\*\*\* END OF DOCUMENT \*\*\*