2C2P Integration Guide

(2C2P Payment Gateway Payment- Non-UI Payment)

Credit Card Storage Payment, Recurring and IPP

Document version 1.1.0

Document Control	
Reference	
File Name	2C2P INTEGRATION GUIDE (NON-UI PAYMENT)V1.1.0
Author	Paing

Revision History			
Revision	Date	Changed By	Comments/Reasons
1.0.0	12 Aug 2013	Paing	Initial
1.0.1	14 Aug 2013	Paing	Datetime Standardization
1.0.2	19 Aug 2013	Thu Min	Added IPP
1.0.3	19 Sep 2013	Paing	Modified on IPP changes, added more details on hashing
1.0.4	25 Sep 2013	Paing	Appendix F
1.0.5	25 Sep 2013	Paing	Cancellation of Recurring and recurring amount.
1.0.6	30 Sep 2013	Paing	Added Inquiry API
1.0.7	09 Oct 2013	Paing	Added recurringAmount
1.0.8	13 Nov 2013	Paing	Adjust the demo sample
1.0.9	13 Nov 2013	Paing	Adjust the demo sample
1.1.0	19 Dec 2013	Paing	Added Promotion

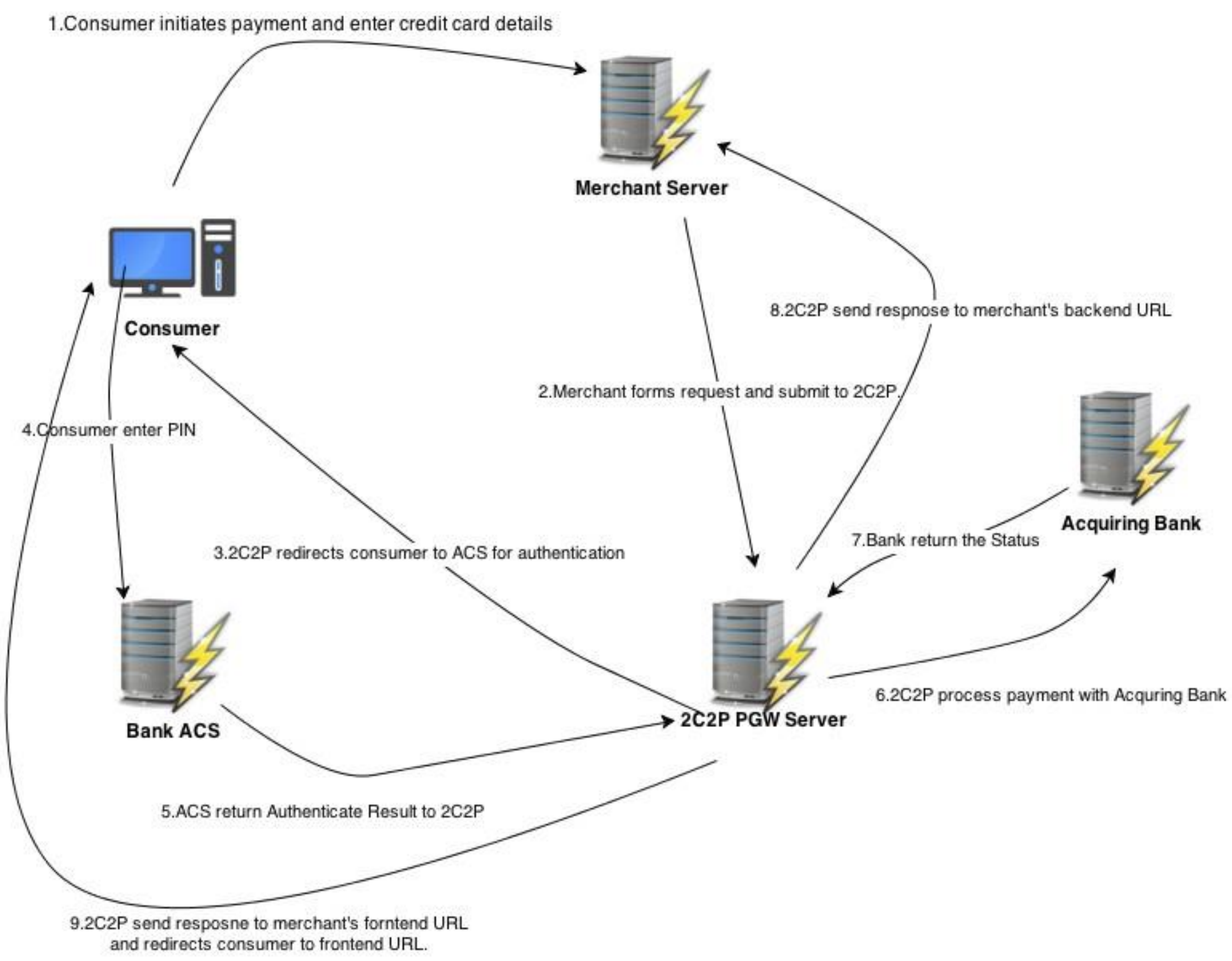
Contents

1. Introduction

1.1 Objectives

The main purpose of this payment integration guide is to provide merchants with the technical details of how to integrate their applications with 2C2P Payment Gateway for credit card payment.
The document contains overview of the system integration, the payment request message format, response message format, encryption and decryption of messages and sample codes of how to send the request message and receive the response.

1.2 System Flow (3DS)


1.Consumer initiates payment and enter credit card details
Merchant Server
Consumer
8.2C2P send respnose to merchant's backend URL
2.Merchant forms request and submit to 2C2P.
4.Consumer enter PIN
Acquiring Bank
7.Bank return the Status
3.2C2P redirects consumer to ACS for authentication
Bank ACS
2C2P PGW Server
6.2C2P process payment with Acquring Bank
5.ACS return Authenticate Result to 2C2P
9.2C2P send resposne to merchant's forntend URL and redirects consumer to frontend URL.

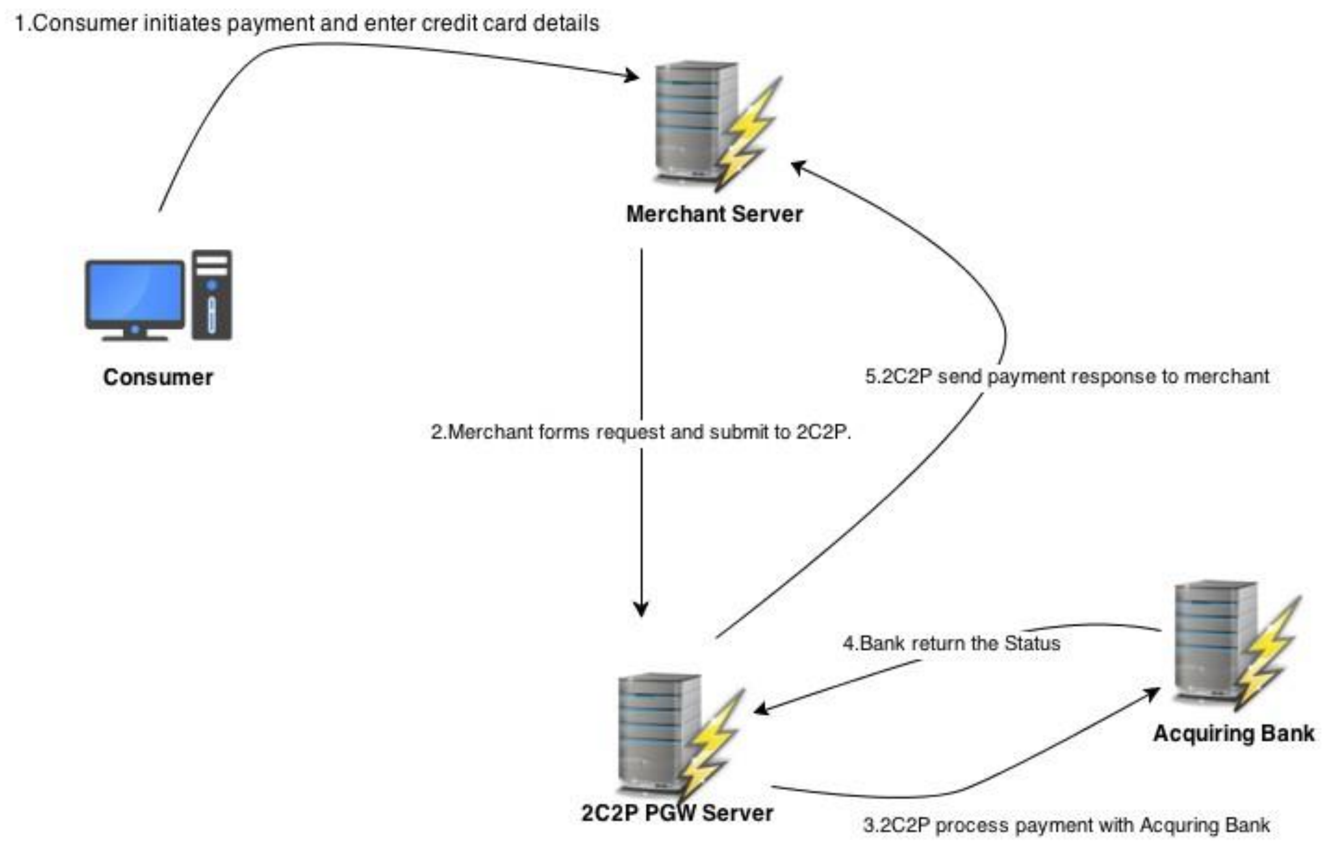
3DS Flow Description

1) Consumer initiate payment transaction and enter credit card details at merchant's system.
2) Merchant forms payment request, submit to 2C2P with form post and redirects consumer's browser to 2C2P Payment Gateway.
3) 2C2P verify the payment request and redirect consumer to ACS page for 3DS authentication.
4) Consumer enters PIN at ACS page.
5) ACS returns authentication result to 2C2P Payment Gateway.
6) 2C2P process payment transaction with Acquiring Bank.
7) Acquiring Bank returns the authorization status to 2C2P.
8) 2C2P update the status and return the payment response to merchant's backend URL via form post.
9) 2C2P returns the payment response to merchant's frontend return URL via form post and redirect consumer's browser to merchant's frontend return URL.
There are 2 types of return URL:

1. Store Card 3DS Return URL (Mandatory): merchant's frontend return URL where the customer will be redirected after the transaction is completed.
2. Store Card 3DS Backend URL (Mandatory): merchant's backend return URL where 2c2p will send the result message (exactly the same message as response message sent to the frontend return URL) before redirecting customer to 'Store Card 3DS Return URL'.
   If timeout occur, 2c2p will retry up to 3 times before proceeding to redirect to customer to 'Store Card 3DS Return URL'.

2C2P PGW always send back to backend URL after payment is processed regarding user action or possible browser disconnect. To prevent any failure on the payment response, it's highly recommended that merchant implement both frontend URL and Backend URL.

1.3 System Flow (non-3DS)


1.Consumer initiates payment and enter credit card details
Merchant Server
Consumer
5.2C2P send payment response to merchant
2.Merchant forms request and submit to 2C2P.
4.Bank return the Status
Acquiring Bank
2C2P PGW Server
3.2C2P process payment with Acquring Bank

Non-3DS Flow Description

1) Consumer initiate payment transaction and enter credit card details at merchant's system.
2) Merchant forms payment request, submit to 2C2P with form post by server-to-server synchronous connection.
3) 2C2P verify the payment request and process payment transaction with Acquiring Bank.
4) Acquiring Bank returns the authorization status to 2C2P.
5) 2C2P update the status and return the payment response to merchant.

2. Payment Request

This section describes the payment request xml string format and the detail explanation of individual fields in the request.

2.1 Payment Request XML

```
<PaymentRequest>
      <version>8.0</version>
      <timeStamp></timeStamp>
      <merchantID></merchantID>
      <uniqueTransactionCode>
      </uniqueTransactionCode>
      <desc></desc>
      <amt></amt>
      <currencyCode></currencyCode>
      <pan></pan>
      <expiry>
            <month></month>
            <year></year>
      </expiry>
      <storeCardUniqueID></storeCardUniqueID>
      <securityCode></securityCode>
      <clientIP></clientIP>
      <panCountry></panCountry> <panBank></panBank>
      <cardholderName></cardholderName>
      <cardholderEmail></cardholderEmail>
      <payCategoryID></payCategoryID>
      <userDefined1></userDefined1>
      <userDefined2></userDefined2>
      <userDefined3></userDefined3>
      <userDefined4></userDefined4>
      <userDefined5></userDefined5>
      <storeCard>Y</storeCard>
      <ippTransaction>Y</ippTransaction>
      <installmentPeriod>3</installmentPeriod>
      <interestType>C</interestType>
      <recurring>Y</recurring>
      <invoicePrefix></invoicePrefix>
      <recurringAmount></recurringAmount>
      <allowAccumulate></allowAccumulate>
      <maxAccumulateAmt></maxAccmulateAmt>
      <recurringInterval></recurringInterval>
      <recurringCount></recurringCount>
      <chargeNextDate></chargeNextDate>
      <promotion></promotion>
      <hashValue></hashValue>
</PaymentRequest>
```

2.2 Payment Request Parameters

Detailed descriptions of the above parameters are explained in the following table.

No	Variable	Data Type	Length	Mandatory	Description	Remark
1	version	Character	5	Y	Version of the Payment Request	8.0 Normal payment Request 8.2 For installment payment plan request.
2	timeStamp	Character	22	N	Date and time of request Message. (if empty, system will take the server received date time.)	Requst DateTime in format of "ddMMyyHHmmss"
3	merchantID	Character	15	Y	Merchant ID	Payment request merchant ID. Provided by 2C2P to merchant.
4	uniqueTransaction Code	Character	20	Y	Invoice No	Transaction unique invoice number. Provided by merchant.
5	desc	Character	50	Y	Product Description	The following symbols are not allowed: !@#$%^&*()<>
6	amt	Numeric	12	Y	Payment amount.	The amount will be padded with '0' from the left and include no decimal point. Example: 1 = 000000000100, 1.5 = 000000000150 Currency exponent is based on MasterCard GCMS.
7	currencyCode	Numeric	3	Y	Standard ISO4217 currency codes.	Refer to Appendix D
8	pan	Numeric	16	Y	Credit Card No	Keyed in by Customer at the merchant website.
9	expiry.month	Numeric	2	Y	Credit Card Expiry	Keyed in by Customer at the merchant
10	expiry.year	Numeric	4	Y	Credit Card Expiry month information.	
11	storeCardUniqueID	Character	20	N	Unique ID of store card info (generated and returned by 2c2p payment gateway if payment send with storeCard option value	

					is "Y")	
12	securityCode	Numeric	4	Y	Three/Four Digits CVV2/CVC2/CID value at the back of the card.	
13	clientIP	Character	15	Y	Client IP Address.	
14	panBank	Character	50	Y	Credit Card Issuer Bank name.	
15	panCountry	Character	2	Y	Credit Card Issuer Bank Country Code	
16	cardholderName	Character	50	Y	Cardholder Name.	
17	cardholderEmail	Character	50	Y	Cardholder Email address.	
18	payCategoryID	Character	10	N	Merchant predefined payment category code for reporting	
19	userDefined1	Character	150	N	Merchant Defined information	(Optional) 2C2P system will response back to merchant whatever information include in request message of this field
20	userDefined2	Character	150	N	Merchant Defined information	(Optional) 2C2P system will response back to merchant whatever information include in request message of this field
21	userDefined3	Character	150	N	Merchant Defined information	(Optional) 2C2P system will response back to merchant whatever information include in request message of this field
22	userDefined4	Character	150	N	Merchant Defined information	(Optional) 2C2P system will response back to merchant whatever information include in request message of this field
23	userDefined5	Character	150	N	Merchant Defined information	(Optional) 2C2P system will response back to merchant whatever information include in request message of this field

24	storeCard	Character	1	N	Command to store cardholder data at processor.	(Optional) if storecard value is 'Y', processor will response unique ID of the card data upon successful authorization. Next payment can be made by sending unique ID instead of full card information. **Note**: If UniqueID is present in the same request, this option will be ignored.
25	ippTransaction	Character	1	N	Command to do installment payment	if ippTransaction value is 'Y', processor will process the installment payment.
26	installmentPeriod	Numeric	2	N	Installment Period	Valid installment period provided by 2c2p. It is mandatory if ippTransaction flag is set to "Y".
27	interestType	Character	1	N	C / M (C - Customer Pay Interest / M - Merchant Pay Interest)	It is mandatory if ippTransaction flag is set to "Y".
28	recurring	Character	1	N	Command to do recurring payment	(Optional) if recurring value is 'Y', processor will response recurring unique ID of the transaction data upon successful authorization. Next payment will be recurred by according to recurring setting.
29	invoicePrefix	Character	15	N	Invoice number prefix	(Optional) Invoice Prefix will be used to generate invoice no of recurring payment. System will generate with invoice prefix followed by serial no in 5 digits format (e.g 123456789012345**00001**). **Note** : if it is recurring payment, it must be mandatory.

30	RecurringAmount	Numeric	12	N	Recurring Amount	(optional) the amount charged in recurring payment. If this value is not set, payment amount will be used for recurring payment.
31	allowAccumulate	Character	1	N	Allow accumulation if authorization failed.	(Optional) Merchant can set 'Y' for yes and 'N' for no to allow accumulation in next recurring cycle. Note : if it is recurring payment, it must be mandatory.
32	maxAccumulateAmt	Numeric	12	N	Limit for the accumulate amount before terminate.	(Optional) If the current accumulate amount exceeded the limitation, the recurring cycle will be terminated. Note : if it is recurring payment, it must be mandatory.
33	recurringInterval	Numeric	5	N	Recurring interval in days.	(Optional) Charge card every x days. Max value 365(1 year). Note : if it is recurring payment, it must be mandatory.
34	recurringCount	Numeric	5	N	Recurring total count allowed	(Optional) Repeat this payment x times. Value '0' for endless loop until terminated manually. Note : if it is recurring payment, it must be mandatory.
35	chargeNextDate	Character	8	N	The next date of recurring payment.	(Optional) Date in DDMMYYYY format. No value will be **current date + recurring Interval**. Note : if it is recurring payment, it must be mandatory.
36	Promotion	Character	20	N	Promotion Code	Promotion Code if merchant wants to perform promotion payment. (e.g Promotion Code "V001" is for Visa Card holder, payment gateway will accept only Visa Card for this payment.) Merchant can set promotion by Card Type OR BIN.

37	hashValue	Character	150	Y	Hash value computed by HMACSHA1 with secret key provided by 2C2P system.	Refer to Section 8 for more details and example.

3. Payment Response

This section describes the payment response xml string format and the detail explanation of individual fields in the response.

3.1 Payment Response XML

```
<PaymentResponse>
      <version>8.0</version>
      <timeStamp></timeStamp>
      <merchantID></merchantID>
      <respCode></respCode>
      <pan></pan>
      <amt></amt>
      <uniqueTransactionCode></uniqueTransactionCode>
      <tranRef></tranRef>
      <approvalCode></approvalCode>
      <refNumber></refNumber>
      <eci></eci>
      <dateTime></dateTime>
      <status></status>
      <failReason></failReason>
      <userDefined1></userDefined1>
      <userDefined2></userDefined2>
      <userDefined3></userDefined3>
      <userDefined4></userDefined4>
      <userDefined5></userDefined5>
      <storeCardUniqueID></storeCardUniqueID>
      <recurringUniqueID></recurringUniqueID>
      <hashValue></hashValue>
</PaymentResponse>
```

3.2 Payment Response Parameters

Detailed descriptions of the above parameters are explained in the following table.

No.	Variable	Data Type	Length	Mandatory	Description	Remark
1	version	Character	5	Y	Version of the Payment Request	Example 8.0
2	timeStamp	Character	22	Y	Date and time of request Message	Requst DateTime in format of "ddMMyyHHmmss"
3	merchantID	Character	15	Y	Merchant ID	Payment request merchant ID. Provided by 2C2P.
4	respCode	Character	2	Y	The code whether the transaction is successful or not.	00 = Approved; 05 = Do Not Honor etc. Refer to Appendix A for all the result code.
5	pan	Character	16	Y	Masked Credit Card Number	First 4 and last 4 digits of credit card number Example: 4444xxxxxxxx1111
6	amt	Numeric	12	Y	The amount that the customer want to convert. The amount from the Pay Request XML.	The amount will be padded with '0' from the left and include no decimal point. Example: 1 = 000000000100, 1.5 = 000000000150
7	uniqueTransactionCode	Character	20	Y	Invoice No. The invoice no from the Pay Request XML.	Provided by Merchant. The invoice number needs to be unique to trace the transaction. Please pad '0' to the left in the case of generated invoice number length is shorter than 20.
8	tranRef	Character	28	Y	Transaction Reference for the Payment Routing System.	Issued by System. This is to trace the transactions in the Routing System.
9	approvalCode	Character	6	Y	Transaction Approval Code from Credit Card Host	
10	refNumber	Character	13 - 15	Y	Transaction Reference Number from Credit Card Host.	
11	eci	Numeric	2	Y	ECI value.	

12	dateTime	Numeric	14	Y	Process Date Time value with ddMMyyHHmmss format	
13	status	Character	1 - 3	Y	Last status of transaction	Refer to Appendix C
14	failReason	Character	100	N	Reason of failure	Refer to Appendix A
15	userDefined1	Character	150	N	User defined info from request message.	
16	userDefined2	Character	150	N	User defined info from request message	
17	userDefined3	Character	150	N	User defined info from request message	
18	userDefined4	Character	150	N	User defined info from request message	
19	userDefined5	Character	150	N	User defined info from request message	
20	storeCardUniqueID	Character	20	N	Unique ID of store card info.	Unique ID of store card info will be generated by process if 'storeCard' parameter is set to 'Y' at payment request XML.
21	recurringUniqueID	Numeric	20	N	The unique id of recurrent payment cycle.	The id will be generated by 2C2P if merchant set the recurring parameters.
22	hashValue	Character	150	Y	Hash value computed by HMACSHA1 with secret key provided by 2C2P System.	Refer to Section 8 for more details and example.

4. Stored Card Details Inquiry and Maintenance

2C2P provides API for merchant to retrieve, update and delete stored credit card detail information. The following describes the details of request and response xml parameters.

4.1 Maintenance Request XML and Parameters

```
<MaintenanceRequest>
      <version>8.0</version>
      <timeStamp></timeStamp>
      <merchantID></merchantID>
      <storeCardUniqueID></storeCardUniqueID>
      <panBank></panBank>
      <panCountry></panCountry>
      <cardholderName></cardholderName>
      <cardholderEmail></cardholderEmail>
      <panExpiry></panExpiry>
      <action></action>
      <hashValue></hashValue>
</MaintenanceRequest>
```

Details of the above parameters are explained in the following table.

No.	Variable	DataType	Length	Mandatory	Description	Remark
1	Version	Character	5	Y	Version of the payment request	8.0
2	timeStamp	Character	22	N	Date and time of the maintenance request. (if empty, system will take server received date time.)	Requst DateTime in format of "ddMMyyHHmmss"
3	merchantID	Character	15	Y	Merchant ID	Payment request merchant ID. Provided by 2C2P to merchant.
4	storeCardUniqueID	Character	20	Y	Unique ID of store card info.	storeCardUniqueID as provided by processor previously.
5	panBank	Character	50	N	Credit Card Issuer Bank name.	Keyed in by Customer on the merchant website. Example: Siam Commercial Bank

6	panCountry	Character	2	N	Credit Card Issuer Bank Country Code	Selected country by Customer on the merchant website. Example: TH, SG (following ISO 3166-1 alpha 2)
7	cardholderName	Character	50	N	Cardholder Name	Keyed in by customer at merchant site. Only the following characters are allowed. `-_,'.A-Za-z&`
8	cardholderEmail	Character	50	N	Cardholder Email address	Keyed in by customer at merchant site.
9	panExpiry	Character	4	N	Credit Card Expiry	Credit Card Expiry in MMYY format (e.g 0215)
10	Action	Character	1	Y	Maintenance action command	I = Inquiry D = Delete U = Update
11	hashValue	Character	150	Y	Hash value computed by HMACSHA1 with secret key provided by 2C2P System.	Refer to Section 8 for more details and example.

4.2 Maintenance Response XML and Parameters

```
<MaintenanceResponse>
      <version>8.0</version>
      <timeStamp></timeStamp>
      <merchantID></merchantID>
      <respCode></respCode>
      <respReason></respReason>
      <panBank></panBank>
      <panCountry></panCountry>
      <cardholderName></cardholderName>
      <cardholderEmail></cardholderEmail>
      <hashValue></hashValue>
</MaintenanceResponse>
```

Details of the above parameters are explained in the following table.

No.	Variable	DataType	Length	Mandatory	Description	Remark
1	Version	Character	5	Y	Version of the payment request	Example: 8.0
2	timeStamp	Character	22	Y	Date and time of the maintenance request.	Requst DateTime in format of "ddMMyyHHmmss"
3	merchantID	Character	15	Y	Merchant ID	Payment request merchant ID. Provided by 2C2P to merchant.
4	respCode	Character	2	Y	The code whether The transaction is successful or not.	00 = Approved; 05 = Do Not Honor; Refer to Appendix A for all result code.
5	respReason	Character	100	Y	Reason of failure	Refer to Appendix C
6	panBank	Character	50	N	Credit Card Issuer Bank name.	Keyed in by Customer on the merchant website. Example: Siam Commercial Bank
7	panCountry	Character	2	N	Credit Card Issuer Bank Country Code	Example: TH, SG (following ISO 3166-1 alpha 2)
8	cardholderName	Character	50	N	Cardholder Name	Keyed in by customer at merchant site.
9	cardholderEmail	Character	50	N	Cardholder Email address	Keyed in by customer at merchant site.
10	hashValue	Character	150	Y	Hash value computed by HMACSHA1 with secret key provided by 2C2P System.	Refer to Section 8 for more details and example.

5. Installment Payment Plan Data Inquiry

5.1 Query Request XML and Parameters

```
<IppQueryRequest>
      <version>1.0</version>
      <timeStamp></timeStamp>
      <merchantID></merchantID>
      <BIN></BIN>
      <hashValue></hashValue>
</IppQueryRequest>
```

Details of the above parameters are explained in the following table.

No	Variable	Data Type	Length	Mandatory	Description	Remark
1	Version	Character	5	Y	Version of the payment request	1.0
2	timeStamp	Character	22	N	Date and time of the maintenance request. (if empty, system will take server received date time.)	Requst DateTime in format of "ddMMyyHHmmss"
3	merchantID	Character	15	Y	Merchant ID	
4	BIN	Character	6	Y	First 6 digit card number	
5	hashValue	Character	150	Y	Hash value computed by HMACSHA1 with secret key provided by 2C2P System.	Refer to Section 8 for more details and example

5.2 Query Response XML and Parameters

```
<IppQueryResponse>
      <version>1.0</version>
      <timeStamp>190913105009</timeStamp>
      <responseCode>00</responseCode>
      <respReason>Success</respReason>
      <bankName>IPP Bank Name</bankName>
      <bankShortName>IPPB</bankShortName>
      <logoURL></logoURL>
      <InstallmentOptions>
      <IPPXML>
            <IPP ID="1" InstallmentPeriod="3" MerInterestRate="0.80"
            CusInterestRate="0.99" MinAmount="1.00" CurrencyCode="THB"
            ValidFrom="2013-08-01" ValidUntil="2013-12-31" PromotionCode=
            "PC001" />
            <IPP ID="2" InstallmentPeriod="6" MerInterestRate="0.80"
            CusInterestRate="0.79" MinAmount="999.00" CurrencyCode="THB"
            ValidFrom="2013-08-01" ValidUntil="2013-12-31" PromotionCode=
            "PC002"/>
      </IPPXML>
      </InstallmentOptions>
      <hashValue>DF819608266F82A502EE08CF49B909E39D661AC0</hashValue>
</IppQueryResponse>
```

Details of the above parameters are explained in the following table.

No	Variable	Data Type	Length	Mandatory	Description	Remark
1	Version	Character	5	Y	Version of the payment request	Example: 1.0
2	timeStamp	Character	22	Y	Date and time of the maintenance request.	Requst DateTime in format of "ddMMyyHHmmss"
4	responseCode	Character	2	Y	The code whether the transaction is successful or not.	00 = Success
5	respReason	Character	100	Y	Reason of failure	
6	bankName	Character	50	N	Bank name	
7	bankShortName	Character	50	N	Bank Short Name	
8	logoURL	Character	255	N	Bank Logo	
9	InstallmentOptions	Character	-	N	Allowable installment payment options	

10	ID (installment option details)	Numeric	5	N	Unique identifier of installment option	
11	InstallmentPeriod (installment option details)	Numeric	2	N	Installment period in months.	Available installment period
12	MerInterestRate (installment option details)	Decimal	18,2	N	Interest rate merchant pay to bank	
13	CusInterestRate (installment option details)	Decimal	18,2	N	Interest rate customer pay to bank.	
14	MinAmount (installment option details)	Decimal	18,2	N	Minimum payment amount	
15	CurrencyCode (installment Option details)	Character	3	N	Allowable Payment currency	Standard ISO4217 currency codes (e.g THB)
16	ValidFrom (installment Option details)	date	10	N	Option validity from date	Option validity from date in the format of "yyyy-MM-
17	ValidUntil (installment Option details)	date	10	N	Option validity to date	Option validity to date in the format of "yyyy-MM-dd"
18	PromotionCode (installment Option details)	Character	50	N	Installment Plan promotion code	
19	hashValue	Character	150		Hash value computed by	Refer to Section 8 for more

6. Transaction Status Inquiry and Recurring Payment Cancel Request

6.1 Request XML and Parameters

```
<PaymentProcessRequest>
      <version>1.0</version>
      <timeStamp>300913171149</timeStamp>
      <merchantID>JT</merchantID>
      <invoiceNo>Test300913110447703</invoiceNo>
      <recurringUniqueID></recurringUniqueID>
      <processType>I</processType>
      <hashValue>03B9E5D29E28CF9F71188AEA2C1B8625FEDD2B8F</hashValue>
</PaymentProcessRequest>
```

Details of the above parameters are explained in the following table.

No	Variable	Data Type	Length	Mandatory	Description	Remark
1	version	Character	5	Y	Version of the request	1.0
2	timeStamp	Character	22	N	Date and time of the request. (if empty, system will take server received date time.)	Requst DateTime in format of "ddMMyyHHmmss"
3	merchantID	Character	15	Y	Merchant ID	merchantID provided by 2C2P to merchant.
4	invoiceNo	Character	20	N	Invoice Number	Note: it is mandatory for processType is "I: Inquiry"
5	recurringUniqueID	Numeric	20	N	The unique id of recurrent payment cycle.	This id is returned to merchant by 2C2P for recurring payment request. Note: it is mandatory if processType is "CR:Cancel Recurring".
6	processType	Character	2	Y	Process Type Code	I (Inquiry) CR (Cancel Recurring)

7	hashValue	Character	150	Y	Hash value computed by HMACSHA1 with secret key provided by 2C2P System.	Refer to Section 8 for more details and example

6.2 Response XML and Parameters

```
<PaymentProcessResponse>
      <version>1.0</version>
      <timeStamp>300913171149</timeStamp> <respCode>00</respCode>
      <pan>411111xxxxxx1111</pan>
      <amt>000000000100</amt>
      <invoiceNo>Test300913110447703</invoiceNo>
      <tranRef>0930110000008</tranRef>
      <approvalCode>161819</approvalCode>
      <eci>07</eci>
      <dateTime>20130930110625</dateTime> <status>A</status>
      <failReason>Inquiry Successful</failReason>
      <recurringActive>N</recurringActive>
      <hashValue>B827885A03C76BC872ED73B2790A6643A809EEDB</hashValue>
</PaymentProcessResponse>
```

Details of the above parameters are explained in the following table.

No	Variable	Data Type	Length	Mandatory	Description	Remark
1	version	Character	5	Y	Version of the request	1.0
2	timeStamp	Character	22	N	Date and time of the request. (if empty, system will take server received date time.)	Requst DateTime in format of "ddMMyyHHmmss"
3	respCode	Character	2	Y	Response Code	00 = Success, 99 = Failed Note: For response code 99, Refer to fail Reason for more details.
4	pan	Character	14	N	Masked Credit Card Number	
5	amt	Numeric	12	N	Payment request amount in 12 digit format. (e.g 10 TBH is equal to 000000001000)	
6	invoiceNo	Character	20	N	Invoice No. Invoice no from	

					The Payment Request.	
7	tranRef	Character	28	N	Transaction Reference for the Payment Routing System.	Issued by System. This is to trace the transactions in the Routing System.
8	approvalCode	Character	6	N	Transaction Approval Code from Credit Card Host	
9	eci	Character	2	N	Transaction ECI Code from Credit Card Authentication System	
10	dateTime	Character	15	N	Process Date Time value with yyyyMMddhhmmss format	
11	status	Character	3	N	Status of Payment Transaction	Refer to Appendix C
12	failReason	Character	100	N	Fail Response description.	
13	recurringActive	Character	1	N	Indicate Recurring Transaction	Value "Y" : Transaction is active recurring transaction. Value "N" : Transaction is not recurring.
14	hashValue	Character	150	Y	Hash value computed by HMACSHA1 with secret key provided by 2C2P System.	Refer to Section 8 for more details and example

7. Encryption and Decryption of Request and Response

The request and response between merchant system and 2C2P PGW is encrypted using PKCS7 asymmetric encryption method.

Before sending payment request to 2C2P Payment Gateway, the request xml string has to be encrypted with public key provided by 2C2P.

The response received from 2C2P Payment Gateway is encrypted with merchant public key. Merchant needs to decrypt using its private key before parsing the response values.

2C2P's public key will be shared with merchant for transmission between Merchant and 2C2P. Merchant's public key need to share with 2C2P for transmission between 2C2P and Merchant.

7.1 Encrypting Request XML String

The following technology can be applied for encryption:

- Java Applet
- Java Bean
- .NET Framework compatible DLL for .NET Framework version 1.1 and 2.0
- PHP

The following example shows how to encrypt using 2C2P Library DLL named "SinaptIQPKCS7" with C# and PHP programming language.

In C#

```
SinaptIQPKCS7.PKCS7 pkcs7 = new SinaptIQPKCS7.PKCS7(); string
beforeEncryptXML = "<PaymentRequest>" +
"<version>8.0</version>" +
"<timeStamp>170713145720</timeStamp>" + "<merchantID>JT</merchantID>"
+
"<uniqueTransactionCode>Test170713145740829</uniqueTransactionCode>"
+ "<desc>Test Payment</desc>" +
"<amt>000000001000</amt>" +
"<currencyCode>764</currencyCode>" +
"<pan>4111111111111111</pan>" +
"<expiry>" +
"<month>02</month>" +
"<year>2016</year>" + "</expiry>" +
"<securityCode>123</securityCode>" +
"<panCountry>SG</panCountry>" +
"<panBank>Test Bank</panBank>" +
"<cardholderName>Mr. Test</cardholderName>" +
"<cardholderEmail>test@test.com</cardholderEmail>" +
"<payCategoryID></payCategoryID>" +
"<storeCard>Y</storeCard>" +
"<recurring>Y</recurring>" +
"<invoicePrefix>INV102020303333</invoicePrefix>" +
"<allowAccumulate>Y</allowAccumulate>" +
"<maxAccumulateAmt>000000100000</maxAccumulateAmt>" +
"<recurringInterval>1</recurringInterval>" +
"<recurringCount>4</recurringCount>" +
"<chargeNextDate>18072013</chargeNextDate>" +
"<hashValue>98420C9AB5922D637BC99215C4921FA76A644A4E</hashValue>" +
"<clientIP>192.168.0.100</clientIP>" +
"<userDefined1>User Defined 1</userDefined1>" +
"<userDefined2>User Defined 2</userDefined2>" +
"<userDefined3>User Defined 3</userDefined3>" +
"<userDefined4>User Defined 4</userDefined4>" +
"<userDefined5>User Defined 5</userDefined5>" +
"</PaymentRequest>";
string encryptedPkcs7Str = pkcs7.encryptMessage(inputMsg,
pkcs7.getPublicCert("D:/2C2P/Application/2C2P-
RandD/demo2.2c2p.com(public).cer"));
```

In PHP

```
<?php
include('pkcs7.php'); //to use this library create a folder for
temporary file in the same location as pkcs7.php with named ‘tmp’.

//public and private keys
$publicKey = "demo2_2c2p.crt";
//content to encrypt
$msg = "sample";

$pkcs7 = new pkcs7();
//to encrypt
$encrypt_text = $pkcs7->encrypt($msg,$publicKey);
echo $encrypt_text;
?>
```

7.2 Decrypting Response XML String

The following example shows how to decrypt the encrypted payment response from 2C2P PGW using C# and PHP programming languages.

In C#

```
SinaptIQPKCS7.PKCS7 pkcs7 = new SinaptIQPKCS7.PKCS7();

string encryptedPaymentResponse =
"MIIE2wYJKoZIhvcNAQcDoIIEzDCCBMgCAQAxggFAMIIBPAIBADAkMBYxFDASBgNVBAMTC1NpbmFwdEl
RIENBAgoeg+bBAAAAAAAMMA0GCSqGSIb3DQEBAQUABIIBAGfvpnQI07oTmL+Mmfdpf6ms+h+2w4+rs0k
q/8/U049Poy5aUl9mwwVKgx1OksFgsYj7+R7CrXR6MvK/P49eFl+wFVtHRS0VxfAQLo3ja+GpqRdMd2o
akkPVJ/AK1VWooli07pzDxdixjN5DDBuRH+mBrV+G1js8eymgpIx1PmNPL9zyGZmiXvurT3xg91amWJQ
/+bfJ2pmBBlMqs+cav5oTJ7v1eoQPzFK+8CvA+dSyZK7hSkGT1gciPVwhCQW7xAIfB/6tR0p2/hgZXlY
nZOn+PbNMozD1/pqPNQjnfW4WG9SsLPiplycLhF7fpJ3dch+RHhBPGkcw19Ow0SeS3TEwggN9Bgkqhki
G9w0BBwEwFAYIKoZIhvcNAwcECPkjkibE353SgIIDWGz3MSEQo0dBJd542O/O3R4Ro7MsK/DhKJyvHG8
aX1AJOYHcMe7XcFCH19jFZFiT0lPM11iHS+XJpz91lSZ8ZAhoaoKt0EbQhGqsQYL5tLOLKDdaJloonH9
SSTiFpTlPQlpRh/qBeBuDegCCxHGXhe4UkWlOds1vCWS+87uRr/+BpY7H5tAgJCXy22ScIDLjxgrYd4O
s4ax3h0Q7v3xkwrL4yTOwLx8dr0HcQRDdHecAjzeHqazig8kqFWKC0jrmBbCDQ2W7r6fFsIYeb6xZceH
fuCiMvPOeKOtjJgbA7rkWq/gaKpNuwE8b4PN2zazI3rANSpLwYSluN4FtN5wj+YivvplgTlhkFm8lGQo
V9hwUaSkJX2shi3yG/O1UjVZIXDtPF7aPPtVFkCjpQXwGkrFUX45XFHoWuI9ozSDd9wlXNwucc05wD9+
dHQChhlSqHkPEcoJ4fzn8PGBmJOCdLSxQ/omkGh52sH8vcOOZFH+vUTv0FwEUb7lpePKqC3xSivOXkU/
O68ttAUW40ZzQKWat9QOId9ij7LxklGXzbnt6F8fJx7TSezB5s6u2JjzoQkHS04NQ7l8fXFObzoL+j4q
LG6MSUxxuA/awL4wNGZLqZ9q2OMUHB9DQoAlSJ/cgdSq1jvmWBEV/vROcxvyaSNuZ8NsqSmsLvSEFeek
hqkCjxHDmHCl4SoQD2iLTBf8AuvJix5R+9sor4x3wMIqcsD72aPalrZOIc6DIbziz0DjZQ+UIjvs4STp
tgPhJuYmutd0Yk+CGNrXtM0zn+a9NLMooDYrOu0BpT3PlQeaFOILfMbckNbDmU8X6kx2cc4y48w2aFtQ
R9QgVcvQ1n7K+JxEKVnpZKMRBs1bBtfd7gNOg59byWNa+JcAzA+2AHPMtKlFksevfGtYO64KwEaG1EIo
4wjJYJ+Czk1ScTvKjwhY2GgsJeyVCLQb3LF51P2EZoyA7zogC7mJJhO4Nksl1dScH/yo3B8fyQ6DcD20
gVySee1Q3Esd/SlgOgxRSG9xbx6lXgf1vEdpF5hCfnvH+hzynpUe1GY4CmHojLtY0YfwUdUq7EX7NChE
n7Gh2yfXSAz7WQM3KJwODYV05jhoH5jSslIrJQt1RzVkHuzS386MUvEhumKis/r0=";

string paymentResponse = pkcs7.decryptMessage(encryptedPaymentResponse,
pkcs7.getPrivateCert("D:/2C2P/Application/2C2P-RandD/demo2.2c2p.com(2c2p).pfx",
"2c2p"));
```

In PHP

```
<?php
include('pkcs7.php'); //to use this library create a folder for temporary
      file in the same location as pkcs7.php with named ‘tmp’.
//public and private keys
$publicKey = "demo2_2c2p.crt";
$privateKey = "demo2_2c2p.pem";
$privateKeyPass = "2c2p";
$encrypt_text = "encrypted message"; //message returned by 2c2p PGW

//to decrypt
$decrypted= $pkcs7-
>decrypt($encrypt_text,$publicKey,$privateKey,$privateKeyPass);
echo $decrypted;
?>
```

8. Hashing

In order to have data integrity and to identify the correct source of the request and response, merchant needs to send hash value together in the request and 2C2P PGW returns the hash value in the response.
Hash value is computed using HMACSHA1 algorithm with merchant secret key (provided by 2C2P to merchant).

To generate the hash value,
Step 1. To get the signature string that is combination of the request/response parameters value set in order.

Request/Response Type	Signature String
Payment Request	merchantID + uniqueTransactionCode + amt
Payment Response	merchantID + tranRef + amt
Maintenance Request	merchantID + storeCardUniqueID + panBank + panCountry + cardholderName + cardholderEmail + panExpiry + action
Maintenance Response	merchantID + respCode + respReason + storedCardUniqueID + panBank + panCountry + cardHolderName + cardHolderEmail
Installment Payment Plan Inquiry Request	merchantID + BIN
Installment Payment Plan Inquiry Response	responseCode + respReason + bankName + bankShortName + logoURL
Transaction Status Inquiry / Recurring Payment Cancel Request	version + merchantID + invoiceNo + recurringUniqueID + processType
Transaction Status Inquiry / Recurring Payment Cancel Response	version + respCode + pan + amt + invoiceNo + tranRef + approvalCode + eci + dateTime + status + failReason

Step 2. To generate hash value using hmacsha1 function with signature string and secret key. (Refer to hashing functions and sample code of hashing mentioned below.)
The following are the SHA1 hashing function available in four different programming languages.

Hashing function in C#

```
private string getHMAC(string signatureString, string secretKey)
{
      System.Text.UTF8Encoding encoding = new
System.Text.UTF8Encoding();
      byte[] keyByte = encoding.GetBytes(secretKey);
      HMACSHA1 hmac = new HMACSHA1(keyByte);
      byte[] messageBytes = encoding.GetBytes(signatureString);
      byte[] hashmessage = hmac.ComputeHash(messageBytes);
      return ByteArrayToHexString(hashmessage);
}
private string ByteArrayToHexString(byte[] Bytes)
{
      StringBuilder Result = new StringBuilder(); string HexAlphabet =
      "0123456789ABCDEF";
      foreach (byte B in Bytes)
      {
            Result.Append(HexAlphabet[(int)(B >> 4)]);
            Result.Append(HexAlphabet[(int)(B & 0xF)]);
      }
      return Result.ToString();
}
```

Hashing function in PHP

```
<?php

$signData = hash_hmac('sha1', "signatureString",'SecretKey', false);

$signData =  strtoupper($signData);

echo urlencode($signData);

?>
```

Hashing function in Java

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

public class hash_hmac_sha1 {
public static String hmacSha1(String strSignatureString, String key) {
try {
       // Get an hmac_sha1 key from the raw key bytes
       byte[] keyBytes = key.getBytes();
       SecretKeySpec signingKey = new SecretKeySpec(keyBytes, "HmacSHA1");

       // Get an hmac_sha1 Mac instance and initialize with the signing key
       Mac mac = Mac.getInstance("HmacSHA1");
       mac.init(signingKey);
       // Compute the hmac on input data bytes
       byte[] rawHmac = mac.doFinal(strSignatureString.getBytes());

       // Convert raw bytes to Hex
       // byte[] hexBytes = Base64Coder.encode(rawHmac);
       //  Covert array of Hex bytes to a String
       return byteArrayToHexString(rawHmac);
       } catch (Exception e) {
       throw new RuntimeException(e);
}
}
static String byteArrayToHexString(byte in[]) {

       byte ch = 0x00;
       int i = 0;
       if (in == null || in.length <= 0)
              return null;
       String pseudo[] = { "0", "1", "2", "3", "4", "5", "6", "7", "8", "9",
              "A", "B", "C", "D", "E", "F" };
       StringBuffer out = new StringBuffer(in.length * 2);

       while (i < in.length) {
              ch = (byte) (in[i] & 0xF0); // Strip offhigh nibble ch = (byte) (ch >>>
              4);
              // shift the bits down
              ch = (byte) (ch & 0x0F);
              // must do this is high order bit is on!
              out.append(pseudo[ch]); // convert thenibble to a String // Character
              ch = (byte) (in[i] & 0x0F); // Strip off low nibble
              out.append(pseudo[ch]); // convert the nibble to a String // Character
              i++;
       }
       String rslt = new String(out); return rslt;
       }
}
```

Hashing function in Perl

```
#!/usr/bin/perl -w
use Digest::HMAC_SHA1 qw(hmac_sha1_hex);
print "content-type: text/html\n\n";
my $hmac_data = "signaturestring";
my $secret_key="746D7SCHAIQ0QUZ0MRJWU0PQ3AD7PJ8B";
my $output = hmac_sha1_hex($hmac_data, $secret_key);
print $output;
```

Example of Hashing in C#

```
//Payment Request
string strSignatureString = merchantID + uniqueTransactionCode + amt;
string HashValue = computeHashValue(strSignatureString, SecretKey);

//Maintenance Request
string strSignatureString = merchantID    + storeCardUniqueID + panBank
+ panCountry + cardholderName + cardholderEmail +  panExpiry + action;
string HashValue = computeHashValue(strSignatureString, SecretKey);

//Installment Payment Plan Inquiry Request
string strSignatureString = merchantID    + BIN;
string HashValue = computeHashValue(strSignatureString, SecretKey);
```

Example of Hashing in PHP

```
//Payment Request
<?php
$strSignatureString = $merchantID . $uniqueTransactionCode . $amt;
$HashValue =  computeHashValue($strSignatureString,$SecretKey);
?>

//Maintenance Request
<?php
$strSignatureString = $merchantID . $ storeCardUniqueID . $panBank . $panCountry
. $cardholderName . $cardholderEmail . $panExpiry . $action;
$HashValue =  computeHashValue($strSignatureString,$SecretKey);
?>

//Installment Payment Plan Inquiry Request
<?php
$strSignatureString = $merchantID . $BIN;
$HashValue =  computeHashValue($strSignatureString,$SecretKey);
?>
```

9. Sending the payment request and Receiving response

9.1 Sending Payment Request to 2C2P (3DS) - Frontend Post

Step 1=> Form xml request string

Step 2 => Encrypt xml request string

Step 3 => Send encrypted value to 2C2P Payment URL with Form post. The form tag variable name for the encrypted message will be “paymentRequest”.

```
<Form method=post
action="http://demo2.2c2p.com/2C2PFrontend/storedCardPaymentV2/AuthPayment.aspx">

<input type="hidden" Name="paymentRequest"
value="MIIGEwYJKoZIhvcNAQcDoIIGBDCCBgACAQAxggFAMIIBPAIBADAkMBYxFDASBgNVBAMTC1NpbmFwdElRIE
NBAgoeg+bBAAAAAAAMMA0GCSqGSIb3DQEBAQUABIIBAAcXZrZwmGdYyFKwfskyMzqoRCvMfmRlPoq7wauu/tNHtQg
5XSjAmrWEfu2qK0D41++uPL8urU7PUzXgGmQgMWsBQcUaF64kvbhuoYK8g7Q+b2jO5in09waRfXoJxlkE1g+5Ecwj
ZamIjrJQyX1+8sOtMRhSlEwQhkVInF77/4Q9UYpJiofegZJJn74XgG88YsCPjn1JggvTA6R2YWxofB50vh05yA8Ga
U1uQxiCQaxqWL+YcwOq//8vO0eBHbSamxg2m8/RqTWx+/7g/D8HSmSyE0QtKBa8dXQr6lcUiOxmJ2L+bo44fdcITB
R4oKQkkdeU/bSJ+W1z76qdXnfaxjkwggS1BgkqhkiG9w0BBwEwFAYIKoZIhvcNAwcECJ8mAU1aPOW3gIIEkOnpZ7l
He/pqaf3GxoWPXkLtUfnPN314BWmACdQOrgyJZKDdOruBMvp6Qix+0CCOJR365avc73cHW11pNNMdS5x86ab3Hj3n
BjtMXzz16xJI/NSBxpw0grGjEAJQYiP/i0ihTka5OP95n5n+quYtVn6vRoChx46uRibm8bnjv1UuEGfgVf21HJWpC
fMJ+WRr5XPhaQgU+rAA9M5Hn+8C4gyclHnVMzgzIdneIPBSPs3002Jo6rr6Tojitly/bx/rGeij46oYYWxlEUN5sK
LUBwl21GD5VdGCeKqAnCpRraa/SNfEJIXrblWrzVgMX1TxI6su8bo6qIiKWT4ZkaIqKxIYH8aCVhDxhBwfkvKUpqm
yQmPa4/+3VFNzNiDXvmFNMQOgTFP7WQsXykSPn1lAIIX1q+lWDhQGQ4wogfwM9s9m0FLrkGtisgJXmob2UaVMdv+R
xilr3Wzv8haSY7TcXCrzDXqYPHxDrTL5tZsim+Tno/IheQT40wtCKx8kDIbHcTNhfdk1cciRVK68790FXH8wPuWdN
VTA849OXILu92nZvVU93ODJ2PNXQ2wVscFXs3leVFaAW37qfQ4T8Yx8HZ2JOTIHflIt0LVoqmC6gUjTkjdkMOxaQi
gWM3KVV/iQxsF0s6ZCftEyzOOTgWsCY9RVEbQhMzFuW2dr0Te9Ezqw11gQ0ibkPy7dV03WL3PpNC+wg8tlQ2lohzy
7EXNn2TuOaIAGBf4cOrR0n3DwiDOS2On6zQw6OF9haY61UxBirjUQP5wPAs6gzRsexAkraOgvxikjQRqrwqXpMhbh
wxcG3HcdH353kVt4OQvlKWAmSVettuO0X6t7gT9K61/qJ8TJimsNSEebJPZ90w0d4PdXb9TT2gZGv/pyMtzJkcz8f
yDz0jXzalKEvqAR8kSyNVhVQcQGlMjwbKfyS8S/dGGj1txfTCYV7Nigys4yE0Fg+n604rV1PxZLW0UY81uMasZPbw
PLRDgTTkpECELARjhiteamfmxfiqQEuL5GjOitNNWt8G7tS8+WGyxM4qdBJZZ2+QtQbhNyasPG5LnepFhg2f+mPC7
7Hl6TiUmyBMAxb8SUFRdaHZq6uMN6Ehil5dVuzf+0oaglpLSDUJReBbXjdpi2vJLcH3rM9ArR+0bHdyS3o7dSlxUV
vtQ9TEQmFvEoL2dMqAGUjAUD63E1tJzVWgGrZQtrhjMK/ASLtCCqnqRbn8N0G2I7ALPj+tWdsTLs+JIJeQqmK1H6U
K/Xv1qxNS22Zq908xb7ePcv9VCMNcxpOzApQie+faf6ANK4x6y9MYfaenrDrcBq95Md51nG/GnIQEflhIh+8NGx0v
KWpqOdZamq4lKlTuiMQ8HkgFV9jaHIYLJl5waVCf/RebX2oK2vXXR/W0u+nOOMCEbM6e7y52v/Kc+v/P3L5NFM9NQ
U6uouZOEkVqajdLyyU9rFhUaGUOsOTjgcOUTILhMSJLCNS32vpukwi5xoJkJTIeLi0D/9AI/A9jBCNTDCz8WqUN2D
9Z6EH4pBNPZbh5oN3Alpv07X4dFQcgpAMaDqR9g=">
<input type="submit" value="Send" name="submit" >
</Form>
```

9.2 Receiving Payment Response from 2C2P (3DS)

Step 1 => Receiving Payment Response : the payment processing system will send back encrypted Payment Response XML message to the *predefined merchant's return URL using HTTP Form Post method. The form tag name of the encrypted payment response will be 'paymentResponse'.

The following is the example code snippet in C# on how to receive payment response from 2C2P system.

```
string encryptedPaymentResponse = "";

encryptedPaymentResponse = Request.Form["paymentResponse"];
```

The following is the example code snippet in PHP on how to receive payment response from 2C2P system.

```
$encryptedPaymentResponse = "";

$encryptedPaymentResponse = _REQUEST["paymentResponse"];
```

Step 2 => Decrypt the payment response using merchant's private key and interpret the result. The following example use 2C2P .Net framework dll to decrypt the payment response.

In C#

```
SinaptIQPKCS7.PKCS7 pkcs7 = new SinaptIQPKCS7.PKCS7();

string encryptedPaymentResponse =
"MIIE2wYJKoZIhvcNAQcDoIIEzDCCBMgCAQAxggFAMIIBPAIBADAkMBYxFDASBgNVBAMTC1NpbmFwdEl
RIENBAgoeg+bBAAAAAAAMMA0GCSqGSIb3DQEBAQUABIIBAGfvpnQI07oTmL+Mmfdpf6ms+h+2w4+rs0k
q/8/U049Poy5aUl9mwwVKgx1OksFgsYj7+R7CrXR6MvK/P49eFl+wFVtHRS0VxfAQLo3ja+GpqRdMd2o
akkPVJ/AK1VWooli07pzDxdixjN5DDBuRH+mBrV+G1js8eymgpIx1PmNPL9zyGZmiXvurT3xg91amWJQ
/+bfJ2pmBBlMqs+cav5oTJ7v1eoQPzFK+8CvA+dSyZK7hSkGT1gciPVwhCQW7xAIfB/6tR0p2/hgZXlY
nZOn+PbNMozD1/pqPNQjnfW4WG9SsLPiplycLhF7fpJ3dch+RHhBPGkcw19Ow0SeS3TEwggN9Bgkqhki
G9w0BBwEwFAYIKoZIhvcNAwcECPkjkibE353SgIIDWGz3MSEQo0dBJd542O/O3R4Ro7MsK/DhKJyvHG8
aX1AJOYHcMe7XcFCH19jFZFiT0lPM11iHS+XJpz91lSZ8ZAhoaoKt0EbQhGqsQYL5tLOLKDdaJloonH9
SSTiFpTlPQlpRh/qBeBuDegCCxHGXhe4UkWlOds1vCWS+87uRr/+BpY7H5tAgJCXy22ScIDLjxgrYd4O
s4ax3h0Q7v3xkwrL4yTOwLx8dr0HcQRDdHecAjzeHqazig8kqFWKC0jrmBbCDQ2W7r6fFsIYeb6xZceH
fuCiMvPOeKOtjJgbA7rkWq/gaKpNuwE8b4PN2zazI3rANSpLwYSluN4FtN5wj+YivvplgTlhkFm8lGQo
V9hwUaSkJX2shi3yG/O1UjVZIXDtPF7aPPtVFkCjpQXwGkrFUX45XFHoWuI9ozSDd9wlXNwucc05wD9+
dHQChhlSqHkPEcoJ4fzn8PGBmJOCdLSxQ/omkGh52sH8vcOOZFH+vUTv0FwEUb7lpePKqC3xSivOXkU/
O68ttAUW40ZzQKWat9QOId9ij7LxklGXzbnt6F8fJx7TSezB5s6u2JjzoQkHS04NQ7l8fXFObzoL+j4q
LG6MSUxxuA/awL4wNGZLqZ9q2OMUHB9DQoAlSJ/cgdSq1jvmWBEV/vROcxvyaSNuZ8NsqSmsLvSEFeek
hqkCjxHDmHCl4SoQD2iLTBf8AuvJix5R+9sor4x3wMIqcsD72aPalrZOIc6DIbziz0DjZQ+UIjvs4STp
tgPhJuYmutd0Yk+CGNrXtM0zn+a9NLMooDYrOu0BpT3PlQeaFOILfMbckNbDmU8X6kx2cc4y48w2aFtQ
R9QgVcvQ1n7K+JxEKVnpZKMRBs1bBtfd7gNOg59byWNa+JcAzA+2AHPMtKlFksevfGtYO64KwEaG1EIo
4wjJYJ+Czk1ScTvKjwhY2GgsJeyVCLQb3LF51P2EZoyA7zogC7mJJhO4Nksl1dScH/yo3B8fyQ6DcD20
gVySee1Q3Esd/SlgOgxRSG9xbx6lXgf1vEdpF5hCfnvH+hzynpUe1GY4CmHojLtY0YfwUdUq7EX7NChE
n7Gh2yfXSAz7WQM3KJwODYV05jhoH5jSslIrJQt1RzVkHuzS386MUvEhumKis/r0=";

string paymentResponse = pkcs7.decryptMessage(encryptedPaymentResponse,
pkcs7.getPrivateCert("D:/2C2P/Application/2C2P-RandD/demo2.2c2p.com(2c2p).pfx",
"2c2p"));
```

In PHP

```
<?php
include('pkcs7.php'); //to use this library create a folder for temporary file
in the same location as pkcs7.php with named 'tmp'.
//public and private keys
$publicKey = "demo2_2c2p.crt";
$privateKey = "demo2_2c2p.pem";
$privateKeyPass = "2c2p";
$encrypt_text = "encrypted message"; //message returned by 2c2p PGW //to
decrypt
$decrypted= $pkcs7-
>decrypt($encrypt_text,$publicKey,$privateKey,$privateKeyPass); echo
$decrypted;
?>
```

9.3 Sending Payment Request and Receiving and Response (non-3DS) - Server-to-Server Post

Step 1=> Form xml request string

Step 2 => Encrypt xml request string

Step 3 => Send encrypted value to 2C2P Payment URL with server to server post and get the encrypted response.

Example in C#

```
public bool SendRequest(string strEncryptedRequest, string paymentURL, out
string strEncryptedResponse, out string err)
{
       strEncryptedResponse = "";
       err = "";
       try
       {
              //Create an instance of the WebRequest class
              WebRequest objRequest = WebRequest.Create(paymentURL);
              objRequest.Timeout = 120000;
              //In milliseconds - in this case 2 min
              objRequest.Method = "POST";
              objRequest.ContentLength = strEncryptedRequest.Length;
              objRequest.ContentType = "application/x-www-form-urlencoded";
              //Create an instance of the StreamWriter class and attach the
              WebRequest object to it - here's where we do the posting
              StreamWriter postWriter = new
              StreamWriter(objRequest.GetRequestStream());
              postWriter.Write(strEncryptedRequest);
              postWriter.Close();
              //Create an instance of the WebResponse class and get the output
              to the rawOutput string
              WebResponse objResponse = objRequest.GetResponse();
              StreamReader sr = new
              StreamReader(objResponse.GetResponseStream());
              strEncryptedResponse = sr.ReadToEnd();
              sr.Close();
              return true;
       }
       catch (Exception ex)
       {
              err = ex.Message.ToString(); return false;
       }
```

Example in PHP

```
<?php
require("HTTP.php");
//POST TO PGW
$HTTP = new HTTP();

$result = $HTTP-
>post("http://demo2.2c2p.com/2c2pfrontend/storedCardPaymentV2/Payment.aspx","payment
Request=".$encrypted);
echo "result: ".$result;
?>
```

HTTP.php

```
<?php
/**
* CURL POST

* @param string $url
* @param string $fields_string
* @return void
* @author 2c2p
*/

Class HTTP
{
        function post($url,$fields_string)
        {
                //open connection
                $ch = curl_init();

                curl_setopt($ch,CURLOPT_URL, $url);
                curl_setopt($ch,CURLOPT_POSTFIELDS, $fields_string);
                curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
                //execute post
                $result = curl_exec($ch); //close connection
                curl_close($ch);

                return $result;
        }
}
```

Note: Payment Request (non-3DS), Maintenance Request, Install Payment Plan Data Inquiry Request, Transaction Status Inquiry Request, Cancellation of Recurring Request require server-to-server backend post.

Payment(3DS) requires frontend form post and redirect.

10. Appendix A - Payment Result Code Table

Result Code	Result Description
00	Approved
01	Refer to Card Issuer
02	Refer to Issuer's Special Conditions
03	Invalid Merchant ID
04	Pick Up Card
05	Do Not Honor
06	Error
07	Pick Up Card, Special Conditions
08	Honor with ID
09	Request in Progress
10	Partial Amount Approved
11	Approved VIP
12	Invalid Transaction
13	Invalid Amount
14	Invalid Card Number
15	No Sun Issuer
16	Approved, Update Track 3
17	Customer Cancellation
18	Customer Dispute
19	Re-enter Transaction
20	Invalid Response
21	No Action Taken

22	Suspected Malfunction
23	Unacceptable Transaction Fee
24	File Update not Supported by Receiver
25	Unable to Locate Record on File
26	Duplicate File Update Record
27	File Update Field Edit Error
28	File Update File Locked Out
29	File Update not Successful
30	Format Error
31	Bank not Supported by Switch
32	Completed Partially
33	Expired Card - Pick Up
34	Suspected Fraud - Pick Up
35	Contact Acquirer - Pick Up
36	Restricted Card - Pick Up
37	Call Acquirer Security - Pick Up
38	Allowable PIN Tries Exceeded
39	No Credit Account
40	Requested Function not Supported
41	Lost Card - Pick Up
42	No Universal Amount
43	Stolen Card - Pick Up
44	No Investment Account
45	Settlement Success
46	Settlement Fail
47	Cancel Success
48	Cancel Fail
49	No Transaction Reference Number

50	Host Down
51	Insufficient Funds
52	No Cheque Account
53	No Savings Account
54	Expired Card
55	Incorrect PIN
56	No Card Record
57	Trans. not Permitted to Cardholder
58	Transaction not Permitted to Terminal
59	Suspected Fraud
60	Card Acceptor Contact Acquirer
61	Exceeds Withdrawal Amount Limits
62	Restricted Card
63	Security Violation
64	Original Amount Incorrect
65	Exceeds Withdrawal Frequency Limit
66	Card Acceptor Call Acquirer Security
67	Hard Capture - Pick Up Card at ATM
68	Response Received Too Late
69	Reserved
70	Settle amount cannot more than authorized amount
71	Inquiry Record Not Exist
72	Reserved
73	Reserved
74	Reserved
75	Allowable PIN Tries Exceeded
76	Invalid Credit Card Format
77	Invalid Expiry Date Format

78	Invalid Three Digits Format
79	Reserved
80	User Cancellation by closing Internet Browser
81	Reserved
82	Reserved
83	Reserved
84	Reserved
85	Reserved
86	ATM Malfunction
87	No Envelope Inserted
88	Unable to Dispense
89	Administration Error
90	Cut-off in Progress
91	Issuer or Switch is Inoperative
92	Financial Institution not Found
93	Trans Cannot be Completed
94	Duplicate Transmission
95	Reconcile Error
96	System Malfunction
97	Reconciliation Totals Reset
98	MAC Error
99	Reserved

11. Appendix B - Maintenance Response Code Table

Result Code	Result Description
00	Action Successful.
01	Stored card ID cannot be found
02	Invalid Request
03	Invalid Merchant ID
04	Invalid Stored Card Unique ID
05	Invalid Customer Email

12. Appendix C - Transaction Status Code Table

Status	Result Description
A	Approved
PF	Payment Failed / Authorization Failed
AR	Authentication Rejected(MPI Reject)
CBR	Corporate BIN Reject
FF	Fraud Rule Rejected
ROE	Routing Failed
IP	Invalid Promotion
F	Failed to process payment
S	Settled
RF	Refunded
V	Voided
RR	Refund Rejected

13. Appendix D - Currency Code Table

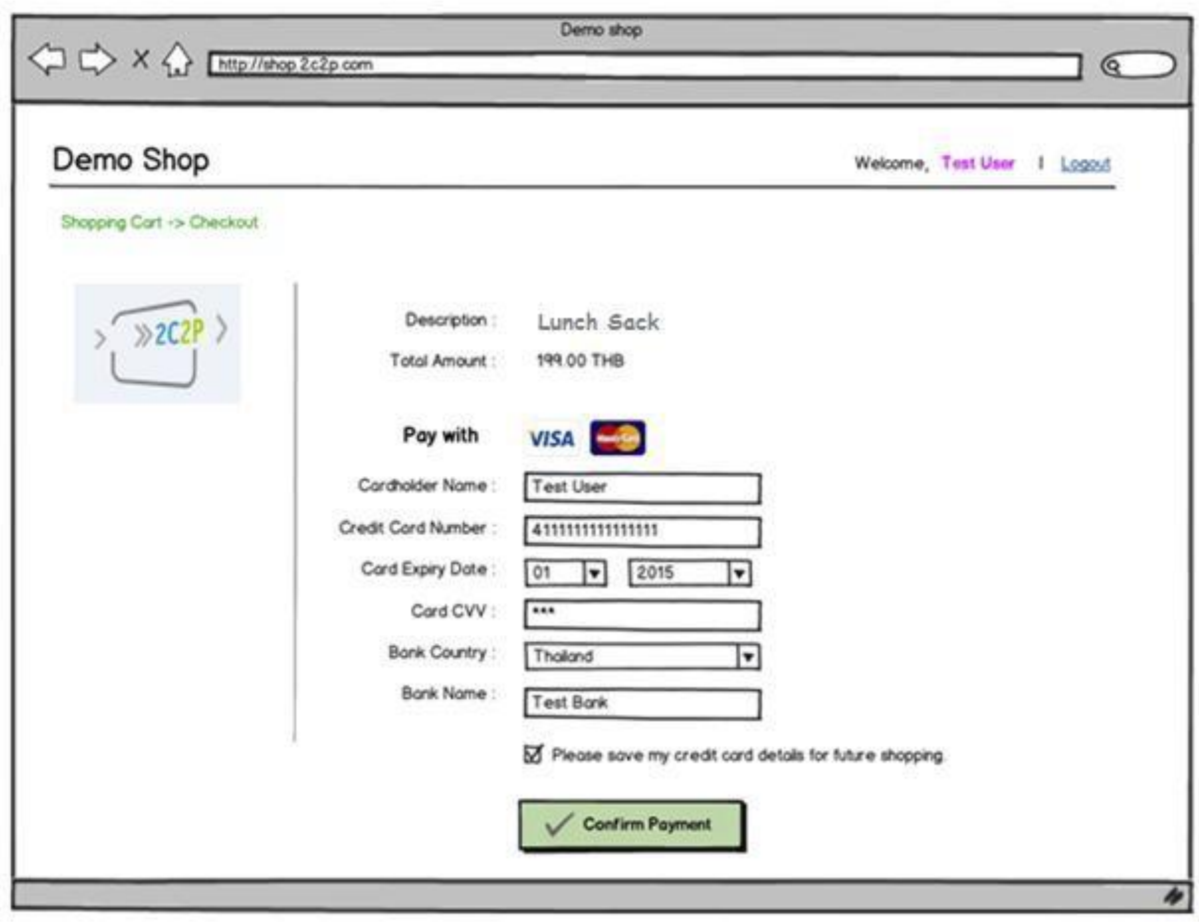
Currency Code	Currency Name
764	Thai Baht
840	US Dollar
702	Singapore Dollar
392	Japan Yen
826	Pound Sterling
458	Malaysian Ringgit
360	Indonesia Rupiah
978	Euro

14. Appendix E - Payment API URLs

Environment	Type	URL
Demo	Payment 3DS	http://demo2.2c2p.com/2C2PFrontend/storedCardPaymentv2/AuthPayment.aspx
Demo	Payment Non-3DS	http://demo2.2c2p.com/2C2PFrontend/storedCardPaymentv2/payment.aspx
Demo	Stored Card Inquiry & Maintenance	http://demo2.2c2p.com/2C2PFrontend/storedCardPaymentv2/maintenance.aspx
Demo	Installment Payment Plan Data Inquiry	http://demo2.2c2p.com/2C2PFrontend/storedCardPaymentV2/IppOption.aspx
Demo	Inquiry & Cancellation of Recurring	http://demo2.2c2p.com/2C2PFrontend/storedCardPaymentV2/PaymentAction.aspx
Production	Payment 3DS	https://s.2c2p.com/storedCardPaymentV2/AuthPayment.aspx
Production	Payment Non-3DS	https://s.2c2p.com/storedCardPaymentV2/payment.aspx
Production	Maintenance	https://s.2c2p.com/storedCardPaymentV2/maintenance.aspx
Production	Installment Payment Plan Data Inquiry	https://s.2c2p.com/storedCardPaymentV2/IppOption.aspx
Production	Inquiry & Cancellation of Recurring	https://s.2c2p.com/storedCardPaymentV2/PaymentAction.aspx

15. Appendix F - Different Scenarios of Request XML regarding with Merchant website

Case A: Registered customer is checking out and want to save card details to future use.



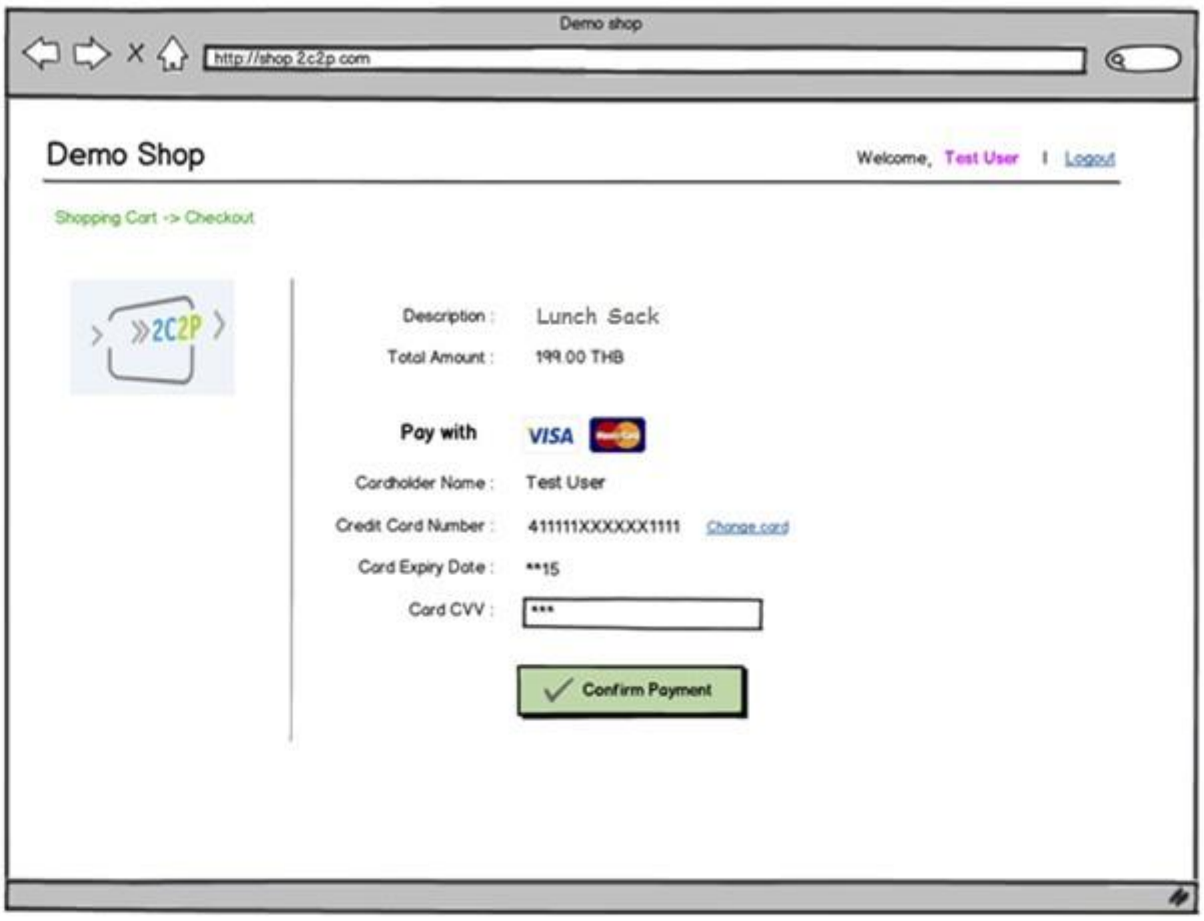
Description :	Lunch Sack
Total Amount :	199.00 THB
Cardholder Name :	Test User
Credit Card Number :	4111111111111111
Card Expiry Date :	01 / 2015
Card CVV :	***
Bank Country :	Thailand
Bank Name :	Test Bank

☑ Please save my credit card details for future shopping.

Case A Sample XML Request

```
<PaymentRequest>
       <version>8.0</version>
       <merchantID>123</merchantID>
       <uniqueTransactionCode>190913161823507</uniqueTransactionCode>
       <desc>Lunch Sack</desc>
       <amt>000000019900</amt>
       <currencyCode>764</currencyCode>
       <pan>4111111111111111</pan>
       <expiry>
              <month>01</month>
              <year>2015</year>
       </expiry>
       <securityCode>123</securityCode>
       <panCountry>TH</panCountry>
       <panBank>Test Bank</panBank>
       <cardholderName>Test User</cardholderName>
       <cardholderEmail>testuser@2c2p.com</cardholderEmail>
       <hashValue>7FD0CFD50EA522BB93F632F9156E1A7E4CD0CCF8</hashValue>
       <clientIP>202.57.142.19</clientIP>
       <storeCard>Y</storeCard>
</PaymentRequest>
```

Case B: Returning customer can use saved card to pay

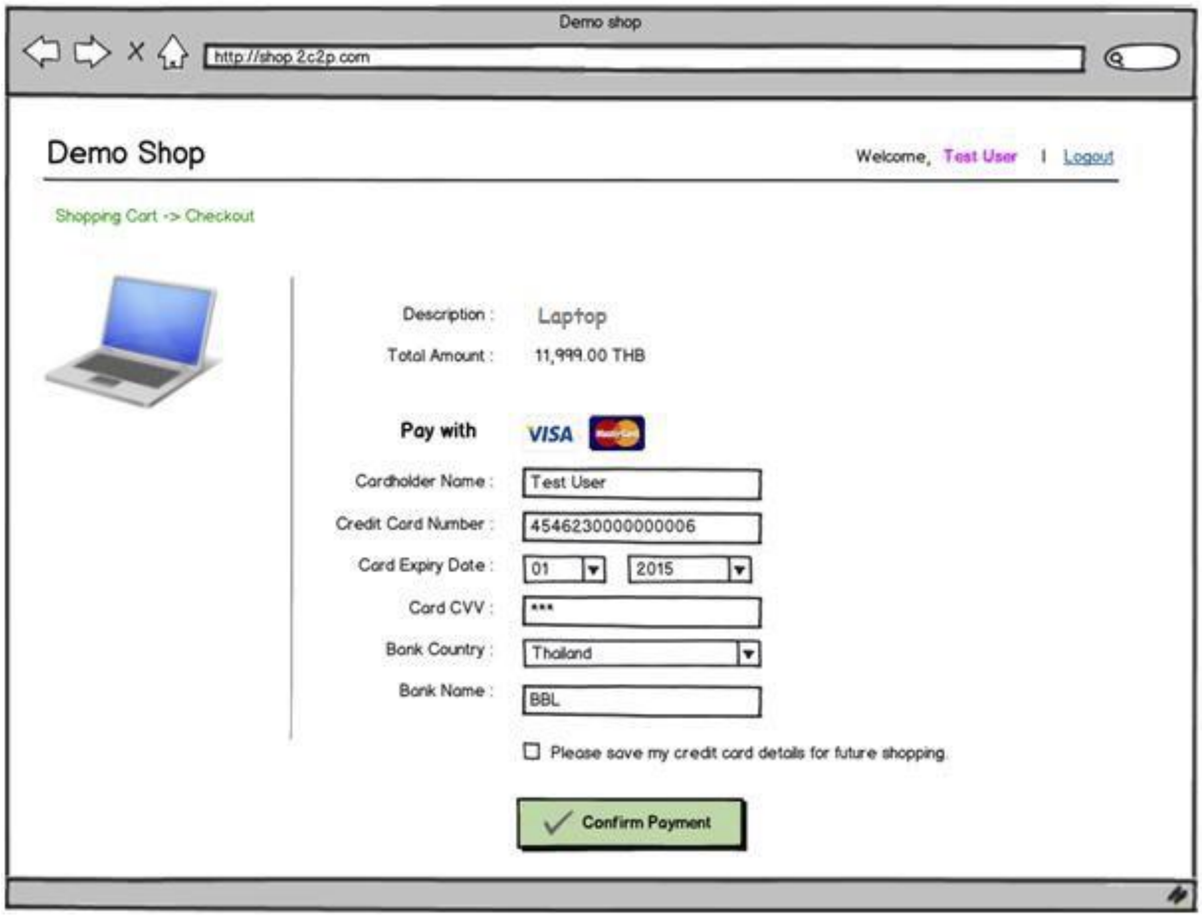


Description : Lunch Sack
Total Amount : 199.00 THB
Pay with VISA
Cardholder Name : Test User
Credit Card Number : 411111XXXXXX1111 Change card
Card Expiry Date : **15
Card CVV : ***

Case B : Payment Request XML

```
<PaymentRequest>
       <version>8.0</version>
       <merchantID>123</merchantID>
       <uniqueTransactionCode>190913161823508</uniqueTransactionCode>
       <desc>Lunch Sack</desc>
       <amt>000000019900</amt>
       <currencyCode>764</currencyCode>
       <storeCardUniqueID>22021314562061227526</storeCardUniqueID>
       <securityCode>123</securityCode>
       <hashValue>B1FFAEE9E539C5EABF979B5FBC305843B3F1D966</hashValue>
       <clientIP>202.57.142.19</clientIP>
</PaymentRequest>
```

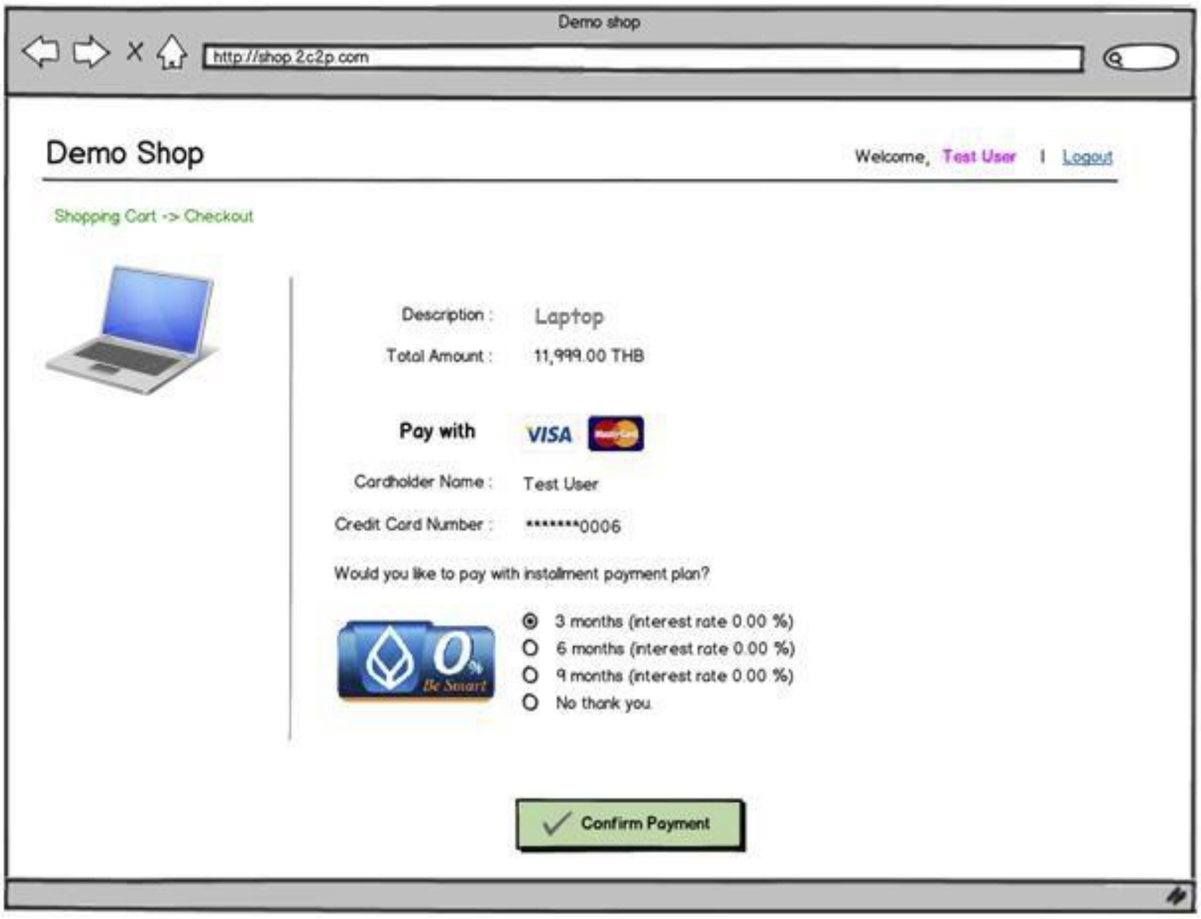
Case C (Step A) : Customer fill credit card details and confirm payment



Description :	Laptop
Total Amount :	11,999.00 THB
Cardholder Name :	Test User
Credit Card Number :	4546230000000006
Card Expiry Date :	01 / 2015
Card CVV :	***
Bank Country :	Thailand
Bank Name :	BBL

☐ Please save my credit card details for future shopping.

Case C (Step B) : merchant web shop display Installment Payment Plan (IPP) option before final confirmation



Description : Laptop
Total Amount : 11,999.00 THB

Pay with

Cardholder Name : Test User
Credit Card Number : *******0006

Would you like to pay with installment payment plan?

[✓ 3 months (interest rate 0.00 %)]
6 months (interest rate 0.00 %)
9 months (interest rate 0.00 %)
No thank you

Case C Payment Request XML

```
<PaymentRequest>
       <version>8.2</version>
       <merchantID>123</merchantID>
       <uniqueTransactionCode>190913161823507</uniqueTransactionCode>
       <desc>Laptop</desc>
       <amt>000001199900</amt>
       <currencyCode>764</currencyCode> <pan>4546230000000006</pan>
       <expiry>
              <month>01</month>
              <year>2015</year>
       </expiry>
       <securityCode>123</securityCode>
       <panCountry>TH</panCountry>
       <panBank>BBL</panBank>
       <cardholderName>Test User</cardholderName>
       <cardholderEmail>testuser@2c2p.com</cardholderEmail>
       <hashValue>7FD0CFD50EA522BB93F632F9156E1A7E4CD0CCF8</hashValue>
       <clientIP>202.57.142.19</clientIP>
       <storeCard>N</storeCard>
       <ippTransaction>Y</ippTransaction>
       <installmentPeriod>3</installmentPeriod>
       <interestType>C</interestType>
</PaymentRequest>
```