MPU National Payment Gateway

Merchant Integration Guide

Version 2.8

KBZ Copy

Copyright © MPU. All Rights Reserved. The information contained in this document is proprietary and confidential information of MPU and her affiliates. The material cannot be disclosed, duplicated, or published, in whole or in part, without MPU's prior written consent.

Document Control			
Reference			
File Name	MPU National Payment Gateway Merchant Integration Guide V2.8.docx		
Author	2C2P		
Revision History			
Version	Date	Revised By	Comments/Reasons
1.0	2 Dec 2013	Lynn Htaik Aung	First Version
1.1	11 Feb 2014	Lynn Htaik Aung	Add Inquiry/Void
1.2	24 Apr 2014	Lynn Htaik Aung	Add Logo
1.2	29 Dec 2014	Lynn Htaik Aung	Change Urls
1.4	19 Jan 2015	Paing Kyaw Zin	Change void/Inquiry request(InvoiceNo)
1.5	30 June 2015	Paing Kyaw Zin	Add FrontendURL,BackendURL and CardInfo in request
2.0	01 May 2016	Paing Kyaw Zin	Add Non-UI Features
2.1	01 Sep 2020	Tin May Linn	Change with https for UAT payment url
2.2	08 Dec 2020	Tin May Linn	Change domain name for UAT payment url
2.3	26 May 2021	Tin May Linn	Add status “RS”, “RR” and “RF” to Appendix.
2.4	12 Nov 2021	Tin May Linn	Add reminder for Tls and Tls 1.1 disable.
2.5	17 Dec 2021	Tin May Linn	Remove 60145 port in UAT environment url.
2.6	18 Mar 2022	Myat Thit Lwin	Replace with latest Response code table from NPS Guide Add merchant awareness for TLS 1.2
2.7	31 May 2022	Tin May Linn	Add sample code for hashing
2.8	06 Jul 2022	Tin May Linn	Notice for duplicate checking flow. Add inquiryDate field in the inquiry request and response.

KBZ Copy

Document Approvals				
Reference				
File Name	MPU National Payment Gateway Merchant Integration Guide V2.8.docx			
Author	2C2P			
Revision History				
Version	Approved By	Project Role	Approval Date	Signature/Electronic Approval
1.0	Thandar Aung	Project Manager		
1.1	Thandar Aung	Project Manager		
1.2	Thandar Aung	Project Manager		
1.3	Thandar Aung	Project Manager		

1.4	Thandar Aung	Project Manager		
1.5	Thandar Aung	Project Manager		
2.0	Thandar Aung	Project Manager		
2.1	Soe Lwin Htoo	Project Manager		
2.2	Soe Lwin Htoo	Project Manager		
2.3	Soe Lwin Htoo	Project Manager		
2.4	Soe Lwin Htoo	Project Manager		
2.5	Soe Lwin Htoo	Project Manager		
2.6	Soe Lwin Htoo	Project Manager		
2.7	Thandar Aung	Project Manager		
2.8	Thandar Aung	Project Manager		

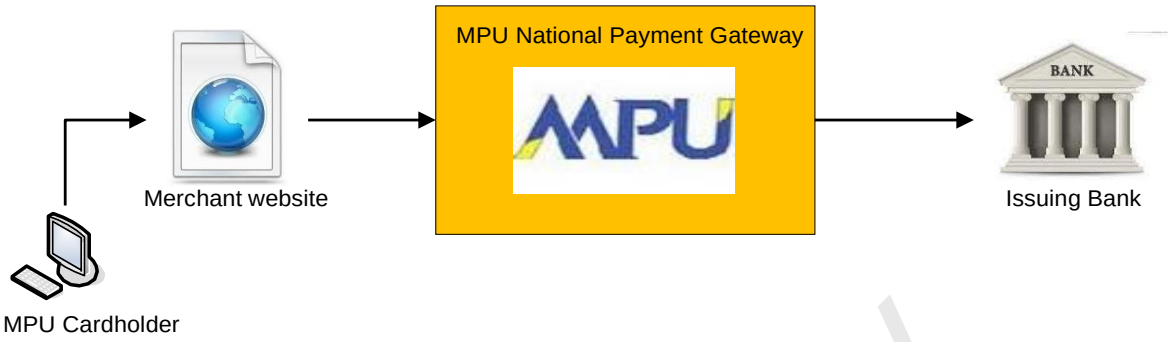
KBZ Copy

Table of Contents

Contents

KBZ Copy

1. Message Flow (Cardholder)



1. Cardholder visits merchant Web site and clicks check out button.
2. Merchant assembles transaction data and sends to MPU National Payment Gateway's payment screen.
3. MPU Payment Gateway checks authentication status of the cardholder.
4. If cardholder's authentication status is valid, OTP (one time password) is sent via SMS or email to the cardholder.
5. Upon successful authentication and OTP verification, MPU Payment Gateway sends authorization message to the issuing bank.
6. MPU Payment Gateway interprets response from the issuing bank and send back transaction results to Merchant Website.

KBZ Copy

2. Payment Request/Response

MPU payment provides merchants to do sale transactions as below.

i. Field description of Payment Request

No	Variable	Data Type	Length	Mandatory	Description	Remark
1	Version	Character	5	Y	Version	Current version value 2.0
2	merchantID	Character	15	Y	Merchant ID	Merchant ID generated by MPU's acquiring Bank.
3	invoiceNo	Character	20	Y	Unique Invoice No	Provided by Merchant. The invoice number needs to be unique in the current day to trace the transaction. Please pad '0' to the left in the case of generated invoice number length is shorter than 20.
4	productDesc	Character	50	Y	Product Description	The product description to be provided by the merchant.

5	amount	Numeric	12	Y	Transaction Amount	The amount needs to be padded with '0' from the left and include no decimal point. Example: 1.00 = 000000000100, 1.5 = 000000000150 Currency exponent follows standard ISO4217 currency codes.
6	currencyCode	Numeric	3	Y	Standard ISO4217 Currency Codes.	Refer to Appendix D
7	categoryCode	Character &Numeric	20	N	Category Code	Merchant can distinct the transaction by adding category code.
8	userDefined1	Character	150	N	Merchant Defined Information	(Optional) MPU system will response back to merchant whatever information include in request message of this field.
9	userDefined2	Character	150	N	Merchant Defined Information	(Optional) MPU system will response back to merchant whatever information include in request message of this field.
10	userDefined3	Character	150	N	Merchant Defined Information	(Optional) MPU system will response back to merchant whatever information include in request message of this field.
11	cardInfo	Character	150	N	CardInfo value computed by AES 256 With merchant secret key provided by MPU System	Encrypt card number and expiry with AES 256 Card Number and card expiry(MMYY) format (eg.9999999999999999;1115) **Note : This features can only available in version 2.0 and above**
12	FrontendURL	Character	255	N	To receive result from MPU	Cardholder will be redirected to this URL. Example: thankyou.php to display result to cardholder with Merchant logo, etc. **Note : This features can only available in version 2.0 and above**
13	BackendURL	Character	255	N	To receive backend result from MPU	Cardholder will not be redirected to this URL. The purpose of this URL is to send results for back-office use.

KBZ Copy

						Note : This features can only available in version 2.0 and above
14	hashValue	Character	150	N	Hash value computed by HMACSHA1 with secret key provided by MPU System.	hashValue will be generated by the following methods: 1. All the input parameters, which are filled in by customer, are sorted by ASCII to be Signature String e.g. **0000000500001508233 4455555764Invoice0034 5MerchantID01Product0 1UserDefined1UserDefin ed2UserDefined3950330 00000005551216**. Please see sample code for details. 2. Signature String will be encrypted by HMACSHA1 with secret key provided by MPU System. For details, please reference How to prepare signature string and do hashing

ii. Field descriptions of Payment Response

No	Variable	Data Type	Length	Mandatory	Description	Remark
1	merchantID	Character	15	Y	Merchant ID	Merchant ID generated by MPU's acquiring bank. It can be optional. If it have the value in request, it will be responded back to merchant.
2	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 result code.
3	pan	Character	16	Y	Masked Card Number	First 4 and last 4 digits of card number Example: 954400xxxxxx1111
4	amount	Numeric	12	Y	Transaction Amount	The amount will be padded with '0' from the left and include no decimal point. Example: 1.00 = 000000000100, 1.5 = 000000000150 Currency exponent follows standard ISO4217 currency codes.
5	invoiceNo	Character	20	Y	Unique Invoice No	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.

KBZ Copy

6	tranRef	Character	28	Y	Transaction Reference for MPU.	Issued by MPU PGW System.
7	approvalCode	Character	6	Y	Transaction Approval Code from Card Issuer Host	
8	dateTime	Numeric	14	Y	Process Date Time value with yyyyMMddhhmmss format	
9	status	Character	2	Y	Last status of transaction	AP= Approved; Refer to Appendix B
10	failReason	Character	100	N	Reason of failure	Refer to Appendix B
11	userDefined1	Character	150	N	Merchant Defined Information	(Optional) MPU system will response back to merchant whatever information include in request message of this field.
12	userDefined2	Character	150	N	Merchant Defined Information	(Optional) MPU system will response back to merchant whatever information include in request message of this field.
13	userDefined3	Character	150	N	Merchant Defined Information	(Optional) MPU system will response back to merchant whatever information include in request message of this field.
14	hashValue	Character	150	Y	Hash value computed by HMACSHA1 with secret key provided by MPU System.	hashValue will be generated by the following methods: 1. All the input parameters, which are filled in by customer, are sorted by ASCII to be Signature String e.g. **0000000500001508233445555576 4Invoice00345MerchantID01Product01UserDefined1UserDefined2UserDefined3**. 2. Signature String will be encrypted by HMACSHA1 with secret key provided by MPU System. For details, please reference How to prepare signature string and do hashing

KBZ Copy

3. Preparing and sending payment request message using HTTP form post

E-Commerce merchant needs to prepare the following Payment Request Form Post message. To prepare this one, merchant must identify URL on `method="post" action=` to the environment that merchant would like to locate (Test or Production);

Test: https://www.mpuecomuat.com/UAT/Payment/Payment/pay
Production: https://www.mpu-ecommerce.com/Payment/Payment/pay

```
<Form method="post" action="https://www.mpuecomuat.com/UAT/Payment/Payment/pay">
<input type="text" id="merchantID" name="merchantID" value="400123456712345"/>
<input type="text" id="invoiceNo" name=" invoiceNo" value="1234567890333"/>
<input type="text" id="productDesc" name="productDesc" value="Test Product"/>
<input type="text" id="amount" name="amount" value="000000010000"/>
<input type="text" id="currencyCode" name="currencyCode" value="840"/>
<input type="text" id="userDefined1" name="userDefined1" value="userDefined1"/>
<input type="text" id="userDefined2" name="userDefined2" value="userDefined2"/>
<input type="text" id="userDefined3" name="userDefined3" value="userDefined3"/>
<input type="text" id="hashValue" name="hashValue"
value="94E8E91C29E73B9648011FADBAE19849B520B24B"/>
</Form>
```

After sending the above HTTP Form Post request , the merchant will have the following information regarding to the transaction, at merchant's "POSTURL" page. **POSTURL** will have both : **FrontEndUrl** - for browser level and **BackEndUrl** - for sever to server. These urls will be configured in the merchant setting by MPU Admin portal.

```
merchantID=400123456712345&
respCode=00&pan=954433XXXXXX1111&amount=000000001000&invoiceNo=1234567890333
&tranRef=123&approvalCode=234567& dateTime= 20130430111211& status=AP&
userDefined1=userDefined1& userDefined2=userDefined2& userDefined3=userDefined3
&hashValue=34E8E91C29E73B9648011FADBAE19849B520B24A
```

Merchant can process the result information by using the following form request methods in various programming languages.

In ASP.Net in C#

```
string merchantID = "",respCode = "",pan = "",amount = "",invoiceNo = "",tranRef =
"",approvalCode = "", dateTime = "",status = "";

merchantID = Request.Form["merchantID"];
respCode =  Request.Form["respCode"];
pan = Request.Form["pan"];
```

In Java J2EE

```
String merchantID = "",respCode = "",pan = "",amount = "",invoiceNo = "",tranRef =
"",approvalCode = "", eci = "",dateTime = "",status = "";

merchantID = request.getParameter("merchantID");
respCode =  request.getParameter("respCode");
pan = request.getParameter("pan");
```

In PHP

```
$merchantID = "", $respCode = "", $pan = "", $amount = "", $invoiceNo = "",
$tranRef = "", $ approvalCode = "", $eci = "", $dateTime = "", $status = "";
$merchantID = _REQUEST["merchantID"];
$respCode = _REQUEST["respCode"];
$pan = _REQUEST["pan""];
```

The following is a sample code snippet of HMACSHA1 in **.Net** programming language.

```
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); //Convert Byte to String.
    }
    private string ByteArrayToHexString(byte[] Bytes)
    {
       string HexAlphabet = "0123456789ABCDEF";
        foreach (byte B in Bytes)
        {
            Result.Append(HexAlphabet[(int)(B >> 4)]);
            Result.Append(HexAlphabet[(int)(B & 0xF)]);
        }
        return Result.ToString();
    }
```

KBZ Copy

The following is a sample code snippet of HMACSHA1 in **J2EE** programming language.

```
private static char[] hexits="0123456789ABCDEF".toCharArray();
public static String getHMAC (String data, String key) {
String HMAC_SHA1_ALGORITHM = "HmacSHA1";
String result = null;

try {
SecretKeySpec signingKey = new SecretKeySpec(key.getBytes(),
HMAC_SHA1_ALGORITHM);
Mac mac = Mac.getInstance(HMAC_SHA1_ALGORITHM);
mac.init(signingKey);
byte[] rawHmac = mac.doFinal(data.getBytes());
Log.d("bytes [] : "+ rawHmac);
StringBuilder sb = new StringBuilder(rawHmac.length*2);
for (int i = 0; i < rawHmac.length; i++) {
sb.append(hexits[(((int)rawHmac[i] & 0xFF) / 16) & 0x0F]);
sb.append(hexits[((int)rawHmac[i] & 0xFF) % 16]);
}
Log.d("in med : "+ sb.toString());
//result = Base64.encodeToString(rawHmac, Base64.DEFAULT);
}
catch (Exception e) {
e.printStackTrace();
}

return sb.toString();
}
```

The following is a sample code snippet of HMACSHA1 in **PHP** programming language.

```
<?php

$signData = hash_hmac('sha1',
"0000000500001508233445555576 4Invoice00345MerchantID01Product01UserDefined
1UserDefined2UserDefined3",'746D7SCHAIQ0QUZ0MRJWU0PQ3AD7PJ8B', false);
$signData = strtoupper($signData);
echo urlencode($signData);
 ?>
```

4. Inquiry Request/Response

MPU provides API for merchant to inquiry sale and void transactions. The following describes the details of request and response parameters. Time limit for inquiry API is 180 days which is same as the dispute time frame.

i. Field description of Inquiry Request

No	Variable	Data Type	Length	Mandatory	Description	Remark
1	merchantID	Character	15	Y	Merchant ID	Merchant ID generated by MPU's acquiring Bank.
2	invoiceNo	Character	20	Y	Unique Invoice No	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.
3	inquiryDate	Character	8	Y	Transaction Date value with yyyyMMdd format	
4	actionType	Character	1	Y	Transaction action type	I = Inquiry
5	hashValue	Character	150	N	Hash value computed by HMACSHA1 with secret key provided by MPU System.	hashValue will be generated by the following methods: 1. All the input parameters, which are filled in by customer, are sorted by ASCII to be Signature String e.g. **150823344555555764I** Please see sample code for details. 2. Signature String will be encrypted by HMACSHA1 with secret key provided by MPU System. For details, please reference How to prepare signature string and do hashing

ii. Field descriptions of Inquiry Response

No	Variable	Data Type	Length	Mandatory	Description	Remark
1	merchantID	Character	15	Y	Merchant ID	Merchant ID generated by MPU's acquiring bank. It can be optional. If it have the value in request, it will be responded back to merchant.
2	respCode	Character	2	Y	The code whether the transaction is successful or not.	00 = Approved; 04= Not Found Refer to Appendix A for all result code.
3	pan	Character	16	Y	Masked Card Number	First 4 and last 4 digits of card number Example: 954400xxxxxx1111

KBZ COPY

4	amount	Numeric	12	Y	Transaction Amount	The amount will be padded with '0' from the left and include no decimal point. Example: 1.00 = 000000000100, 1.5 = 000000000150 Currency exponent follows standard ISO4217 currency codes.
5	invoiceNo	Character	20	Y	Unique Invoice No	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.
6	tranRef	Character	28	Y	Transaction Reference for MPU.	Issued by MPU PGW System.
7	approvalCode	Character	6	Y	Transaction Approval Code from Card Issuer Host	
8	dateTime	Numeric	14	Y	Process Date Time value with yyyyMMddhhmmss format	
9	status	Character	2	Y	status of transaction	AP= Approved; Refer to Appendix B
10	failReason	Character	100	N	son of failure	Refer to Appendix B
11	categoryCode	Character &Numeric	20	N	egory Code	Merchant can distinct the transaction by adding category code.
12	userDefined1	Character	150	N	chant Defined Information	(Optional) MPU system will response back to merchant whatever information include in request message his field.
13	userDefined2	Character	150	N	chant Defined Information	(Optional) MPU system will response back to merchant whatever information ude in request message of this field.
14	userDefined3	Character	150	N	chant Defined Information	(Optional) MPU system will response back to merchant whatever information include in request message his field.
15	hashValue	Character	150	Y	h value computed by HMACSHA1 with secret key provided by MPU System.	hashValue will be generated by the following methods: 1. All the input parameters, which are filled in by customer, are sorted by ASCII to be Signature String e.g. **000000050000150823344555555764Invoice00345**

KBZ COPY

						MerchantID01Product01 UserDefined1UserDefine d2UserDefined3. 2. Signature String will be encrypted by HMACSHA1 with secret key provided by U System. For details, please reference How to prepare signature string and do hashing

5. Void Request/Response

i. Field description of Void Request

No	Variable	Data Type	Length	Mandatory	Description	Remark
1	merchantID	Character	15	Y	Merchant ID	Merchant ID generated by MPU's acquiring Bank.
2	invoiceNo	Character	20	Y	Unique Invoice No	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.
3	actionType	Character	1	Y	Transaction action type	V = Void
4	amount	Numeric	12	Y	nsaction Amount	The amount needs to be padded with '0' from the left and include no decimal point. Example: 1.00 = 000000000100, 1.5 = 000000000150 Currency exponent follows standard ISO4217 currency codes.
5	currencyCode	Numeric	3	Y	ndard ISO4217 Currency Codes.	er to Appendix D
6	hashValue	Character	150	N	h value computed by HMACSHA1 with secret key provided by MPU System.	hashValue will be generated by the following methods: 1. All the input parameters, which are filled in by customer, are sorted by ASCII to be Signature String e.g. **000000050000150823344555557641 04V** . Please see sample code for details. 2. Signature String will be

KBZ COPY

						encrypted by HMACSHA1 secret key provided by MPU System. For details, please reference How to prepare signature string and do hashing

ii. Field descriptions of Void Response

No	Variable	Data Type	Length	Mandatory	Description	Remark
1	merchantID	Character	15	Y	Merchant ID	Merchant ID generated by MPU's acquiring bank. It can be optional. If it have the value in request, it will be responded back to merchant.
2	respCode	Character	2	Y	The code whether the transaction is successful or not.	00 = Voided 01 = Not Approved 02 = Voided Already 03 = Amount Mismatched 04 = Not Found
3	pan	Character	16	Y	Masked Card Number	First 4 and last 4 digits of card number Example: 954400xxxxxx1111
4	amount	Numeric	12	Y	Transaction Amount	The amount will be padded with '0' from the left and include no decimal point. Example: 1.00 = 000000000100, 1.5 = 000000000150 Currency exponent follows standard ISO4217 currency codes.
5	invoiceNo	Character	20	Y	Unique Invoice No	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 th is shorter than 20.
6	tranRef	Character	28	Y	Transaction Reference for MPU.	ed by MPU PGW System.
7	approvalCode	Character	6	Y	Transaction Approval Code from Card Issuer Host	
8	dateTime	Numeric	14	Y	Process Date Time value with yyyyMMddhhmmss format	
9	status	Character	2	Y	status of transaction	Approved; Refer to Appendix B

KBZ Copy

10	hashValue	Character	150	Y	h value computed by HMACSHA1 with secret key provided by MPU System.	hashValue will be generated by the following methods: 1. All the input parameters, which are filled in by customer, are sorted by ASCII to be Signature String e.g. **000000050000150823344555555764Invoice00345MerchantID01Product01**. 2. Signature String will be encrypted by HMACSHA1 with secret key provided by U System. For details, please reference How to prepare signature string and do hashing

6. Preparing and sending inquiry/void request message using HTTP form post

Test: https://www.mpuecomuat.com/UAT/Payment/Action/api
Production: https://www.mpu-ecommerce.com/Payment/Action/api

Request

```
https://www.mpuecomuat.com/UAT/Payment/Action/api?merchantID=400123456712345
& tranRef=9549&actionType=I&hashValue=34E8E91C29E73B9648011FADBAE19849B520B24A
```

Response

```
merchantID=400123456712345&
respCode=00&pan=954433XXXXXX1111&amount=000000001000&invoiceNo=1234567890333
&tranRef=123&approvalCode=234567& dateTime= 20130430111211& status=AP&
userDefined1=userDefined1& userDefined2=userDefined2& userDefined3=userDefined3
&hashValue=34E8E91C29E73B9648011FADBAE19849B520B24A
```

KBZ Copy

7. How to prepare signature string and do hashing

Prepare signature string with the merchant request data. (Note: The merchant request data can be difference according to message type)

Message Type	Request/Response	Signature String Data
Sale	Request	Merchant Id + invoice number + product description + amount + currency code + category code + user defined 1 + user defined 2 + user defined 3 + frontend url + backend url
	Response	Merchant Id + response code + PAN + amount + invoice number + transRef + approval code + transaction datetime + status + failed reason + user defined 1 + user defined 2 + user defined 3 + category code
Void	Request	Merchant Id + invoice number + action type + amount + currency code
	Response	Merchant Id + response code + PAN + amount + invoice number + transRef + approval code + transaction datetime + status
Inquiry	Request	Merchant Id + invoice number + action type
	Response	Merchant Id + response code + PAN + amount + invoice number + transRef + approval code + transaction datetime + status + fail reason + user defined 1 + user defined 2 + user defined 3

```
ArrayList arrMerchantRequest=new ArrayList();
arrMerchantRequest.Add(merchantId);
arrMerchantRequest.Add(invoicenumber);
arrMerchantRequest.Add(productdescription);
arrMerchantRequest.Add(amount);
arrMerchantRequest.Add(currencycode);
arrMerchantRequest.Add(categoryCode);
arrMerchantRequest.Add(userdefined1);
arrMerchantRequest.Add(userdefined2);
arrMerchantRequest.Add(userdefined3);
arrMerchantRequest.Add(frontendurl);
arrMerchantRequest.Add(backendurl);

//Array list is sorted with case-sensitive ordinal string comparison.
arrMerchantRequest.Sort(StringComparer.Ordinal);

Convert sorted array into string. (i.e., signature string)
foreach(string reqData in arrMerchantRequest)
{
    signatureString+=reqData;
}


//Generate signature string to Hash
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);
```

KBZ Copy

```
        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();
      }
```

8. How to encrypt card number and expiry with AES 256 in C#

```
String strToEnc = "9999999999999999;1115"
String secretkey = <<merchant's secret key>>

private string getAESEnc(string strToEnc, string secretkey)
{

       string strEnc = string.Empty;
       AES32 sAES = new AES32(secretkey, "", 256);
       strEnc = sAES.Encrypt(strToEnc);
       return strEnc;
 }

public class AES32
{
 private string EncryptionKey = "";
 private string EncryptionVI = "";
 private int EncryptionKeySize = 0;

 public AES32(string keyValue, string vi, int keySize)
 {
     EncryptionKey = keyValue;
     EncryptionVI = vi;
     EncryptionKeySize = keySize;
 }

 public string Encrypt(string clearText)
 {

     byte[] clearBytes = Encoding.Unicode.GetBytes(clearText);
     using (Aes encryptor = Aes.Create())
     {
         Rfc2898DeriveBytes pdb = new
                        Rfc2898DeriveBytes(EncryptionKey, new byte[] { 0x49,
       0x76, 0x61, 0x6e, 0x20, 0x4d, 0x65, 0x64, 0x76, 0x65, 0x64, 0x65, 0x76
       });
         encryptor.Key = pdb.GetBytes(32);
```

KBZ Copy

```
            encryptor.IV = pdb.GetBytes(16);
            using (MemoryStream ms = new MemoryStream())
            {
                                    using (CryptoStream cs = new CryptoStream(ms,
                 encryptor.CreateEncryptor(), CryptoStreamMode.Write))
                {
                    cs.Write(clearBytes, 0, clearBytes.Length);
                    cs.Close();
                }
                clearText = Convert.ToBase64String(ms.ToArray());
            }
        }
        return clearText;
    }

    public string Decrypt(string cipherText)
    {

        byte[] cipherBytes = Convert.FromBase64String(cipherText);
        using (Aes encryptor = Aes.Create())
        {
            Rfc2898DeriveBytes pdb = new
                           Rfc2898DeriveBytes(EncryptionKey, new byte[] { 0x49,
          0x76, 0x61, 0x6e, 0x20, 0x4d, 0x65, 0x64, 0x76, 0x65, 0x64, 0x65, 0x76
          });
            encryptor.Key = pdb.GetBytes(32);
            encryptor.IV = pdb.GetBytes(16);
            using (MemoryStream ms = new MemoryStream())
            {
                                using (CryptoStream cs = new CryptoStream(ms,
          encryptor.CreateDecryptor(), CryptoStreamMode.Write))
                {
                    cs.Write(cipherBytes, 0, cipherBytes.Length);
                    cs.Close();
                }
                cipherText = Encoding.Unicode.GetString(ms.ToArray());
            }
        }
        return cipherText;
    }


}
```

KBZ Copy

Merchant Acceptance Awareness

MPU payment system will not support TLS 1.0 and TLS 1.1 for security purpose. TLS 1.1 dates back to 2006, is considered a deprecated protocol and is no longer secure. Merchant websites need to support higher than TLS 1.1. Currently, MPU only support TLS 1.2.

Appendix A - Result Code Table

Result Code	Result Description
00	Transaction Approved
02	Refer to card issuer
03	Invalid Merchant
04	Do not honor
05	Unable to process
06	Invalid transaction for this terminal

08	Issuer Timeout
09	No Original
10	Unable To Reverse
12	Invalid transaction for card/issuer/acquirer
13	Invalid amount (format error)
14	Invalid card number
15	Invalid Issuer BIN
17	Invalid capture date (terminal business date)
19	System error; Re-enter transaction
20	No From Account
21	No To Account
22	No Checking Account
23	No Saving Account
24	No Credit Account
25	Unable Locate Record
30	Format error
36	Restricted card (only cross transactions with GRG)
39	Transaction not allowed
41	Lost card; Pickup
43	Hot card; Pickup (if possible)
51	No funds available
54	Expired card
55	Incorrect PIN; Re-enter
56	No Card Record
57	Txn Not Permitted On Card
58	Txn Not Permitted On Term
61	Exceeds limit
62	Restricted card
63	MAC failed
65	Exceeds frequency limit
66	Exceed fee
67	Retain card; no reason specified
68	Late response
75	Exceeds PIN retries
76	Invalid account
77	No sharing arrangement between IST/Switch and network
78	Requested function not available
79	Key validation error
81	Translate PIN failed
84	Invalid life cycle on transaction
87	PIN Key error
89	Security violation
91	IST/Switch not available
92	Invalid issuer in transaction
93	Invalid acquirer in transaction
94	Invalid originator in transaction

KBZ Copy

96	System error
98	Duplicate reversal
99	Duplicate transaction

Appendix B – Transaction Status Table

Status	Result Description
AP	Approved
RS	Ready to Settle
SE	Settled
VO	Voided
DE	Declined
FA	Failed
RE	Refund Pending
RR	Refund Ready
RF	Refunded
PR	Payment Gateway receive

Appendix C – Currency Codes

Currency Code	Currency Name
104	Myanmar Kyat (MMK)

=====End of Document =====

KBZ Copy