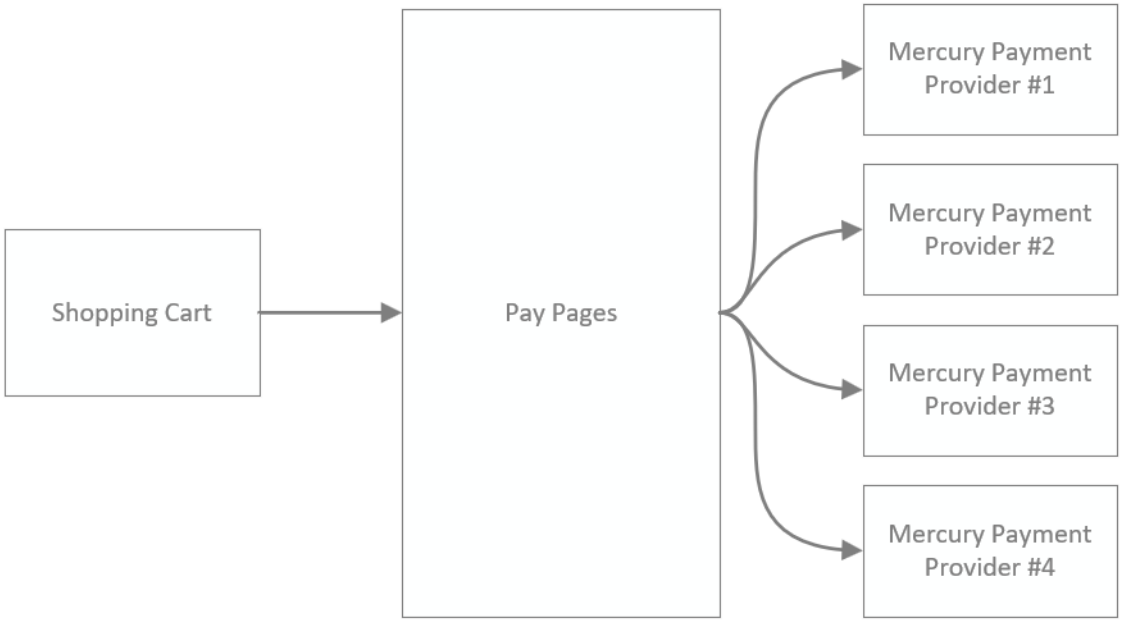
takepayments

Pay Pages Integration guide

Introduction

The paypages solution acts as a intermediary between a merchants shopping cart (or other system where they would like to accept payments) and any one of a number of payment providers.



This provides two main benefits to the integrator :

- A common interface for taking a payment regardless of provider being used.
- Real time tracking of transactions via the Mercury Merchant Portal and API.

The service is secured by requiring the body of all requests and responses to be signed with an HMAC.

This allows us to ensure that the request has come from the merchant and that the contents of the message has not been tampered with *after* the signature was generated.

The same HMAC signing is used on responses we send back to the merchant via the clients browser and via webhooks.

Onboarding

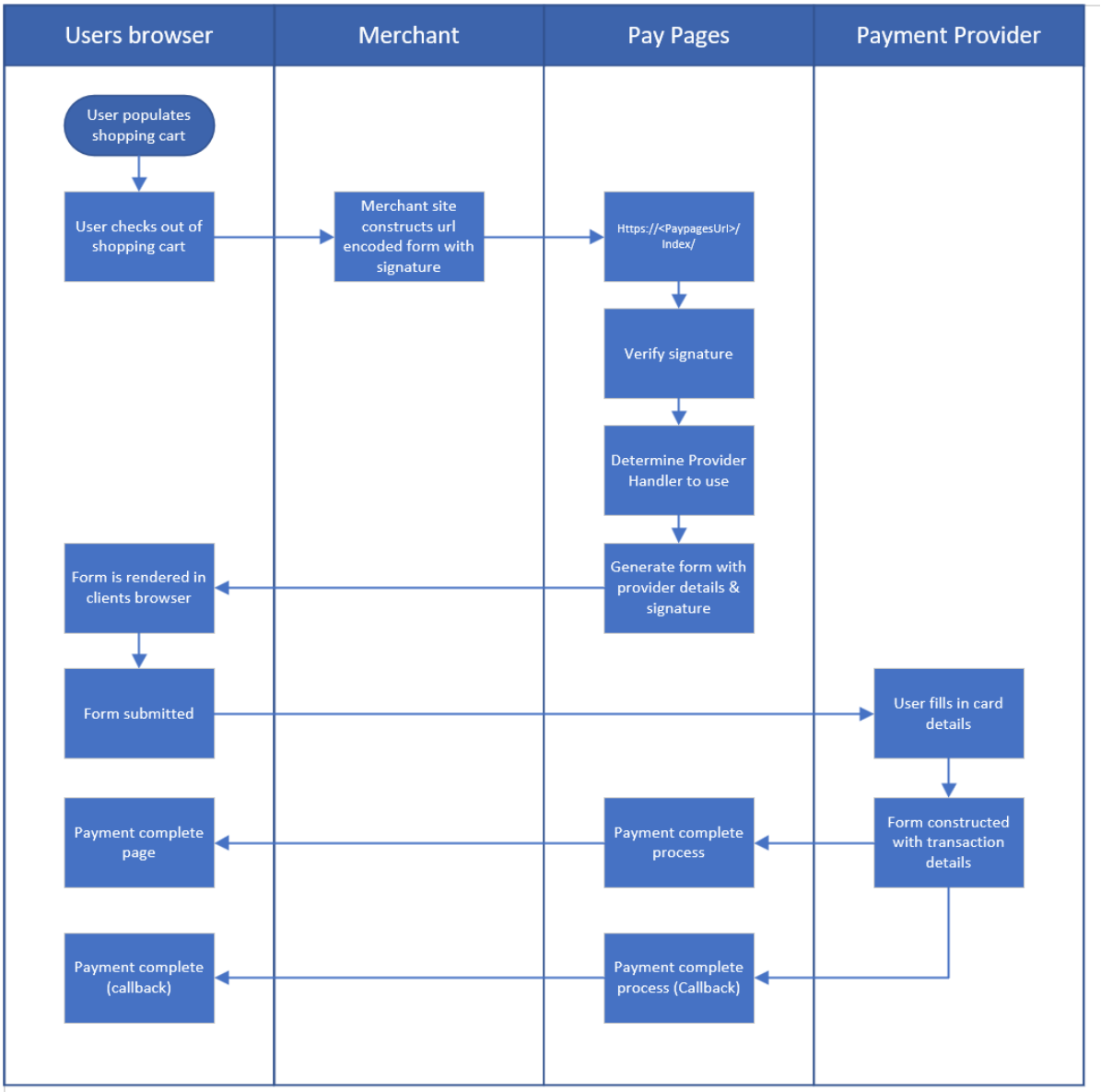
[TBD]

Process

The overall pay pages process is that the merchants system will generate a form post with a signature and send it to pay pages. Pay pages will then look up the relevant configuration for the merchant and route the customer to the appropriate payment provider mapping the payment request into the required format at the same time.

The customer will then enter their card information and make the payment before being redirected back to the merchants system (to a URL specified in the initial request).

Finally we will send an HTTP Post to the merchants system with the transaction information including its final status.



Starting a Payment

URL : `https://<Paypages Url>/`

Input Parameters

Initiating a payment is done by sending a post to `https://<Paypages Url>` with the body containing a url encoded form with the following fields.

Field Name	Form Key	Description
Amount	amount	The Amount in pence
Merchant Order Ref	merchantTransactionReference	Merchant Defined Order Reference
Merchant Category Code	merchantCategoryCode	Merchant Category Code (For test 5411 - Grocery Stores - can be used)
Customer Name	customerName	Customers Name
Customer Email	customerEmail	Customers Email Address
Customer Phone	customerPhone	Customers Phone Number
Customer Address	customerAddress	Customers Address
Customer Postcode	customerPostcode	Customers Post Code
Country Code	countryCode	Customers Country Code
Return URL	returnUrl	Url that we will redirect the customer to
Webhook Url	webhookUrl	Url that we will send the transaction complete message to
Terminal Id	terminalId	The Guid of the virtual terminal that you would like to use for this transaction - if not supplied the default terminal will be used.
TP Merchant Reference	TakePaymentsMerchantReference	The Merchants TP Reference
Signature	signature	A hash of the above fields using the merchants client secret as a salt

The final payload posted to PayPages should look like this

```
amount=1.00&MerchantTransactionReference=&merchantCategoryCode=5411&customerName=Test+Customer&customerEmail=webdevelopers%40takepayments.com&customerPhone=%2B44%280%298450099575&customerAddress=25+The+Larches+Narborough+Leicester+LE10+2RT&customerPostcode=LE10+2RT&countryCode=826&returnUrl=http%3A%2F%2Flocalhost%3A5102%2FStartTransaction%2FPaymentComplete&webhookUrl=http%
```

```
3A%2F%2Flocalhost%3A5102%2FStartTransaction%2FCallBack&terminalId=b5814ecf-
c689-64cd-3795-1f0d3f2fcb2e&TakePaymentsMerchantReference=TPM-
00001&signature=67B0531352B97E0054A3474945BF1DD42C8E9D419CC22577BF39738C3B2
03518
```

Expected output

Once being sent through the Paypages process the customer will be redirected back to the merchant (via an HTTP POST to the `Return URL` specified in the input request), the post will contain the following information (Url encoded key value pairs - the same as the input defined above).

Field Name	Description
Status	Status of the transaction (Approved/Declined etc)
TransactionId	ID of the transaction (from the integrated database)
MerchantTransactionId	Merchant defined transaction ID
Signature	A hash of the above fields using the merchants app secret as a salt

The signature in the body should be validated to ensure that the body has not been tampered with in transit.

Signing the request

The rules for creating and verifying signatures are as follows:

1. ALL form elements are included (Unless value is null/empty in which case both the key and value are excluded)
2. All form elements are ordered alphabetically (By the key)
3. ALL key value pairs are concatenated into a string and comma separated
4. Client secret appended to the end of the string
5. Entire string is uppercased

Given the form to be posted :

```
amount=1.00&MerchantTransactionReference=&merchantCategoryCode=5411&custome
rName=Test+Customer&customerEmail=webdevelopers%40takepayments.com&customer
Phone=%2B44%280%298450099575&customerAddress=25+The+Larches+Narborough+Leic
ester+LE10+2RT&customerPostcode=LE10+2RT&countryCode=826&returnUrl=http%3A%
2F%2Flocalhost%3A5102%2FStartTransaction%2FPaymentComplete&webhookUrl=http%
3A%2F%2Flocalhost%3A5102%2FStartTransaction%2FCallBack&terminalId=b5814ecf-
c689-64cd-3795-1f0d3f2fcb2e&TakePaymentsMerchantReference=TPM-00001
```

We would expect the string to be hashed to be the below :

```
AMOUNT=1.00,COUNTRYCODE=826,CUSTOMERADDRESS=25 THE LARCHES NARBOROUGH
LEICESTER LE10
2RT,CUSTOMEREMAIL=WEBDEVELOPERS@TAKEPAYMENTS.COM,CUSTOMERNAME=TEST
```

```
CUSTOMER,CUSTOMERPHONE=+44(0)8450099575,CUSTOMERPOSTCODE=LE10
2RT,MERCHANTCATEGORYCODE=5411,RETURNURL=HTTP://LOCALHOST:5102/STARTTRANSACT
ION/PAYMENTCOMPLETE,TAKEPAYMENTSMERCHANTREFERENCE=TPM-
00001,TERMINALID=B5814ECF-C689-64CD-3795-
1F0D3F2FCB2E,WEBHOOKURL=HTTP://LOCALHOST:5102/STARTTRANSACTION/CALLBACK,SEC
RET=1234567890
```

Hashing examples

The hash can then be generated as follows

C#

```
using System.Text;
using System.Security.Cryptography;

public class TakePaymentsHashingExample
{
    public static string GenerateSignature(string stringToHash)
    {
        using (HashAlgorithm algorithm = SHA256.Create())
        {
            var hash =
algorithm.ComputeHash(Encoding.UTF8.GetBytes(stringToHash.ToUpper()));

            var sb = new StringBuilder();
            foreach (byte b in hash)
            {
                sb.Append(b.ToString("X2"));
            }

            return sb.ToString();
        }
  }
}

// Call the signature generation function and write output to the console

var result =
TakePaymentsHashingExample.GenerateSignature("AMOUNT=1.00,COUNTRYCODE=826,CUSTOMER
ADDRESS=25 THE LARCHES NARBOROUGH LEICESTER LE10
2RT,CUSTOMEREMAIL=WEBDEVELOPERS@TAKEPAYMENTS.COM,CUSTOMERNAME=TEST
CUSTOMER,CUSTOMERPHONE=+44(0)8450099575,CUSTOMERPOSTCODE=LE10
2RT,MERCHANTCATEGORYCODE=5411,RETURNURL=HTTP://LOCALHOST:5102/STARTTRANSACTION/PAY
MENTCOMPLETE,TAKEPAYMENTSMERCHANTREFERENCE=TPM-00001,TERMINALID=B5814ECF-C689-
64CD-3795-
1F0D3F2FCB2E,WEBHOOKURL=HTTP://LOCALHOST:5102/STARTTRANSACTION/CALLBACK,SECRET=123
4567890");

Console.WriteLine(result);
```

Javascript

```
async function generateSignature(stringToHash) {
  const encoder = new TextEncoder();
  const data = encoder.encode(stringToHash.toUpperCase());
  const hashBuffer = await crypto.subtle.digest('SHA-256', data);
  const hashArray = Array.from(new Uint8Array(hashBuffer));
  const hashHex = hashArray.map(byte => byte.toString(16).padStart(2,
'0')).join('');
  return hashHex;
}

(async () => {
 const result = await
generateSignature('AMOUNT=1.00,COUNTRYCODE=826,CUSTOMERADDRESS=25 THE LARCHES
NARBOROUGH LEICESTER LE10
2RT,CUSTOMEREMAIL=WEBDEVELOPERS@TAKEPAYMENTS.COM,CUSTOMERNAME=TEST
CUSTOMER,CUSTOMERPHONE=+44(0)8450099575,CUSTOMERPOSTCODE=LE10
2RT,MERCHANTCATEGORYCODE=5411,RETURNURL=HTTP://LOCALHOST:5102/STARTTRANSACTION/PAY
MENTCOMPLETE,TAKEPAYMENTSMERCHANTREFERENCE=TPM-00001,TERMINALID=B5814ECF-C689-
64CD-3795-
1F0D3F2FCB2E,WEBHOOKURL=HTTP://LOCALHOST:5102/STARTTRANSACTION/CALLBACK,SECRET=123
4567890');

 console.log(result);
 })();
```

Python

```
import hashlib

def generate_signature(string_to_hash):
    hash_algorithm = hashlib.sha256()
    hash_algorithm.update(string_to_hash.upper().encode('utf-8'))
    return hash_algorithm.hexdigest()

# Call the signature generation function and print the output

result = generate_signature("AMOUNT=1.00,COUNTRYCODE=826,CUSTOMERADDRESS=25 THE
LARCHES NARBOROUGH LEICESTER LE10
2RT,CUSTOMEREMAIL=WEBDEVELOPERS@TAKEPAYMENTS.COM,CUSTOMERNAME=TEST
CUSTOMER,CUSTOMERPHONE=+44(0)8450099575,CUSTOMERPOSTCODE=LE10
2RT,MERCHANTCATEGORYCODE=5411,RETURNURL=HTTP://LOCALHOST:5102/STARTTRANSACTION/PAY
MENTCOMPLETE,TAKEPAYMENTSMERCHANTREFERENCE=TPM-00001,TERMINALID=B5814ECF-C689-
64CD-3795-
1F0D3F2FCB2E,WEBHOOKURL=HTTP://LOCALHOST:5102/STARTTRANSACTION/CALLBACK,SECRET=123
4567890")

print(result)
```

All of the examples above will output a hash of `67B0531352B97E0054A3474945BF1DD42C8E9D419CC22577BF39738C3B203518` The whole form (including the signature) can now be sent to the pay pages system, an example of the raw request body is below.

```
amount=1.00&MerchantTransactionReference=&merchantCategoryCode=5411&customerName=Test+Customer&customerEmail=webdevelopers%40takepayments.com&customerPhone=%2B44%280%298450099575&customerAddress=25+The+Larches+Narborough+Leicester+LE10+2RT&customerPostcode=LE10+2RT&countryCode=826&returnUrl=http%3A%2F%2Flocalhost%3A5102%2FStartTransaction%2FPaymentComplete&webhookUrl=http%3A%2F%2Flocalhost%3A5102%2FStartTransaction%2FCallBack&terminalId=b5814ecf-c689-64cd-3795-1f0d3f2fcb2e&TakePaymentsMerchantReference=TPM-00001&signature=67B0531352B97E0054A3474945BF1DD42C8E9D419CC22577BF39738C3B203518
```

This should be sent to the pay pages system as an HTTP Post, this can be done in many ways such as the example below :

```
const form = document.createElement('form');
form.style.display = 'none'; // Hide the form

form.method = 'POST';
form.action = '<--PAYPAGES URL-->'; // Replace with the correct URL

const requestBody = {
  amount: '1.00',
  MerchantTransactionReference: '',
  merchantCategoryCode: '5411',
  customerName: 'Test Customer',
  customerEmail: 'webdevelopers@takepayments.com',
  customerPhone: '+44(0)8450099575',
  customerAddress: '25 The Larches Narborough Leicester LE10 2RT',
  customerPostcode: 'LE10 2RT',
  countryCode: '826',
  returnUrl: 'http://localhost:5102/StartTransaction/PaymentComplete',
  webhookUrl: 'http://localhost:5102/StartTransaction/CallBack',
  terminalId: 'b5814ecf-c689-64cd-3795-1f0d3f2fcb2e',
  TakePaymentsMerchantReference: 'TPM-00001',
  signature: '67B0531352B97E0054A3474945BF1DD42C8E9D419CC22577BF39738C3B203518'
};

for (const key in requestBody) {
  if (requestBody.hasOwnProperty(key)) {
    const input = document.createElement('input');
    input.type = 'hidden';
    input.name = key;
    input.value = requestBody[key];
    form.appendChild(input);
```

```
    }
}

document.body.appendChild(form);

form.submit();
```

Receiving Transaction Status

The transaction status will be sent back to you via two routes, an HTTP Post *via the clients browser* containing the following form data

```
Status=Declined&TransactionId=26e2125b-74f6-4696-9aad-
c11df8c3499a&Signature=1F6257AB3B7A4A1D4AA31EF4B494778C827D54E197D6F20010B7
297A26B3E559
```

This same form will also be posted *outside the browser* to the webhook URL that you specified in the original request.

Validating the signature

When you receive posts from the pay pages system the data will be signed in the same way as the initial request that you sent in. In order to validate the signature :

1. Remove the signature from the posted form data.
2. Order the form data alphabetically (By Key)
3. Comma separate the key value pairs
4. Append ",SECRET="
5. Uppercase the whole string

Given the form body shown above you should have a string like this :

```
STATUS=DECLINED,TRANSACTIONID=26E2125B-74F6-4696-9AAD-
C11DF8C3499A,SECRET=1234567890
```

When run through the hashing algorithm defined above this should yield the following signature which can be compared to the signature that you have received, confirming that the payload has not been tampered with.

```
1F6257AB3B7A4A1D4AA31EF4B494778C827D54E197D6F20010B7297A26B3E559
```

You can now compare this signature to the one received in the HTTP Post and validate that the form has not been tampered with.

Note that this form may contain other fields not listed here, the same logic applies, generate the string to be signed with ALL form fields where the value is not empty / null / blank

Test Card Details

Type	Card Number	CV2	Address
Master Card Debit	5301250070000191	419	25 The Larches, Narborough, Leicester, LE10 2RT

Testing different results for the transaction is achieved by specifying different expiry months on the card - Note that the expiry must always be in the future. The Year component of the expiry date has no bearing on the outcome.

Month	Result
January	Authorised
February	Declined

Initiate a Refund

URL : `https://<Gateway URL>/Refund`

All transactions start and end the same way within pay pages, you send a signed form via HTTP Post into the system with the required fields and receive a response back via the RedirectUrl and WebHookUrl that you specified.

For a refund the required fields are :

Field Name	Form Key	Description
Transaction Id	transactionId	The ID of the transaction being refunded
Return URL	returnUrl	Url that we will redirect the customer to
Webhook Url	webhookUrl	Url that we will send the transaction complete message to
TP Merchant Reference	TakePaymentsMerchantReference	The Merchants TP Reference
Signature	signature	A hash of the above fields using the merchants client secret as a salt

For guidance on generating the signature follow the same process as detailed under the "Starting a Payment" section.

As with the "Starting a Payment" process you will receive a response from pay pages via the return URL and the WebHook Url that you have specified, you should validate this response in the same way as you would a payment response.

Initiate a Query

URL : `https://<Gateway URL>/Query`

All transactions start and end the same way within pay pages, you send a signed form via HTTP Post into the system with the required fields and receive a response back via the RedirectUrl and WebHookUrl that you specified.

For a query the required fields are :

Field Name	Form Key	Description
Transaction Id	transactionId	The ID of the transaction being refunded
Return URL	returnUrl	Url that we will redirect the customer to
TP Merchant Reference	TakePaymentsMerchantReference	The Merchants TP Reference
Signature	signature	A hash of the above fields using the merchants client secret as a salt

For guidance on generating the signature follow the same process as detailed under the "Starting a Payment" section.

As with the "Starting a Payment" process you will receive a response from pay pages via the return URL that you have specified, you should validate this response in the same way as you would a payment response - note that queries do not send data to any webhooks.

Common Issues

The most common issue encountered is with the signature generation. You should ensure that the input string for the request signature is in the alphabetical order, only include Key Value Pairs where the value has been set and ensure the secret is appended to the end of the string.

You can test your signing algorithm using the examples of forms and signatures given above.