Paynow – 3rd Party Shopping Cart or Link Integration Guide

A guide outlining merchant integration into Paynow for externally hosted shopping carts or applications. For details on other integration methods including Paynow hosted shopping carts, bill payments and button generation please see the relevant documents in the Developer Documentation section on Paynow

Version 1.0.5
15 August 2014

Contents

Setup

Signup and Key Generation

Each integration needs its own ID and Integration Key to identify and secure all server to server messaging. In order to get your Paynow ID and Key follow these steps:

- Register on Paynow and follow the email validation steps. https://www.paynow.co.zw/Customer/Register
- Login with your newly created account and setup the bank account details you would like to receive payment into.
- Go to the [Other Ways To Get Paid] page. https://www.paynow.co.zw/Home/Receive
- Click [Create/Manage Shopping Carts]
- Click [Create Advanced Integration]
- Enter a name you will use to identify the integration.
- Opt whether you with to absorb fees on this integration.
- Enter the email address you wish to receive transaction updates to.
- Notification URL should be left blank as we will be specifying individual URLs per transaction.
- Enter any note you wish to keep about the integration, this will not be shown to the client.
- Choose which Payment Methods you want this integration to use.
- Click Save

The integration will be created and you will be returned to the same page. In the Integration Keys section you will see Integration ID, this is the id you will use below when initiating a transaction, note this id is unique to the integration not your account, if you have more than one integration you will get multiple ids.

For security reasons we do not display your integration key on this page, you need to click [Email Key To Company Address]. It is vital you keep your Integration Key a secret.

You can now begin integration in test mode. It is recommended you [Generate New Key] when moving your site from a development environment to live, this will stop you generating test transactions on the live account and that any other developers will no longer know the Integration Key.

Test Mode

When you first create an integration it is in test mode. Test mode allows you to create, cancel and pay a transaction to test all possible scenarios with your system. However no actual money is moved and you do not need access to Visa/MasterCard/Vpayments/EcoCash/TeleCash to do a test transaction.

After creating a transaction ONLY THE MERCHANT ACCOUNT USED TO CREATE THE INTEGRATION can login and Fake a Payment. Any other users will get a message saying the merchant is in testing and they cannot proceed with payment.

When making payment in test mode select [TESTING: Faked Success] and click [Make Payment], Paynow will reply to your site as if payment has been made.

When you are happy that you have completed integration go back to the Integration Keys section and click [Request to be Set Live]. Paynow support will check you have performed at least one successful test transaction and set you to live. Once you are set live you will begin receive payment from your select Payment Methods.

System Layout

The Paynow interface relies on HTTP POSTs to exchange data between the merchant site and the Paynow server and HTTP GETs to redirect the customer.

Security information such as Hashes and Integration Keys should never be included in any client visible pages or URLs. It is recommended Integration Keys are stored in a secure manner either encrypted or separate from the main site's database.

Initiate a Transaction

When the customer is ready to make payment the merchant site must perform an Initiate Transaction request. This request is in the form of an HTTP POST to the URL:

https://www.paynow.co.zw/interface/initiatetransaction

The HTTP POST should include the following fields:

id	Integer	Integration ID shown to the merchant in the "3rd Party Site or Link Profile" area of "Receive Payment Links" section of "Sell or Receive" on Paynow.
reference	String	The transaction's reference on the merchant site, this should be unique to the transaction.
amount	Decimal	Final amount of the transaction, in USD to two decimal places (do not include a currency symbol).
additionalinfo	String	*(optional)* Additional info to be displayed on Paynow to the Customer. This should not include any confidential information.
returnurl	String	The URL on the merchant website the customer will be returned to after the transaction has been processed. It is recommended this URL contains enough information for the merchant site to identify the transaction.
resulturl	String	The URL on the merchant website Paynow will post transaction results to. It is recommended this URL contains enough information for the merchant site to identify the transaction.
authemail	String	*(optional)* If the field is present and set to an email address Paynow will attempt to auto login the customers email address as an anonymous user. If the email address has a registered account the user will be prompted to login to that account.
status	String	Should be set to "Message" at this stage of the transaction.
hash	String	Details of Hash generation are provided in a later section of this document.

Success

If the initiate transaction request is successful Paynow will reply with a message as a string and formatted as an HTTP Post, i.e. each field separated by a & and Key Value pairs separated by an = and all Values URL Encoded.

The message will contain the following fields:

browserurl	String	The URL on Paynow that the merchant site will redirect the Customer's browser to. Upon redirect the Customer will perform their transaction.
pollurl	String	The URL on Paynow the merchant site can poll to confirm the transaction's current status.
status	String	Set to "Ok" at this stage of the transaction.
hash	String	Details of Hash generation are provided in a later section of this document.

It is vital that the merchant site verify the hash value contained in the message before redirecting the Customer to the browserurl.

Below is an example of a successful initiate transaction request:

Status=Ok&BrowserUrl=http%3a%2f%2fwww.paynow.co.zw%3a7106%2fPayment%2fConfirmPayment%2f1169&PollUrl=http%3a%2f%2fwww.paynow.co.zw%3a7106%2fInterface%2fCheckPayment%2f%3fguid%3d3cb27f4b-b3ef-4d1f-9178-5e5e62a43995&Hash=8614C21DD93749339906DB35C51B06006B33DC8C192F40DFE2DB6549942C837C4452E1D1333DE9DB7814B278C8B9E3C34D1A76D2F937DEE57502336E0A071412

Error

If the initiate transaction request failed Paynow will reply with a message as a string and formatted as an HTTP Post, i.e. each field separated by a & and Key Value pairs separated by an = and all Values URL Encoded.

The message will contain the following fields:

status	String	Set to "Error" at this stage of the transaction.
error	String	Details of the failure on Paynow

Below is an example of a unsuccessful initiate transaction request:

Status=Error&Error=Invalid+amount+field

Complete a Transaction

When the transaction has been completed the Paynow server will send a status update message to the merchant server using the resulturl specified by the merchant when the transaction initiation occurred. The Customer's browser will then be redirected back to the returnurl on the merchant site to view the results.

Status update

Whenever the status of a transaction is changed, for example payment made, the Paynow server will send the following message to the merchant server. The message will be sent as an HTTP POST to the resulturl specified by the merchant when the transaction initiation occurred.

reference	String	The transaction's reference on the merchant site.
amount	Decimal	Final amount of the transaction, in USD to two decimal places.
paynowreference	String	Reference number for the transaction in Paynow.
pollurl	String	The URL on Paynow the merchant site can poll to confirm the transaction's current status.
status	String	After payment is complete Paynow will notify the merchant site with one of the following statuses: • "Paid" – Transaction paid successfully, the merchant will receive the funds at next settlement • "Awaiting Delivery" – Transaction paid successfully, but is sitting in suspense waiting on the merchant to confirm delivery of the goods. • "Delivered" – The user or merchant has acknowledged delivery of the goods but the funds are still sitting in suspense awaiting the 24 hour confirmation window to close. The following are other possible status settings, these will sent to the merchant site if they change in Paynow or if the merchant polls Paynow for the current status: • "Created" – Transaction has been created in Paynow, but has not yet been paid by the customer. • "Sent" – Transaction has been created in Paynow and an up stream system, the customer has been referred to that upstream system but has not yet made payment. • "Cancelled" – The transaction has been cancelled in Paynow and may not be resumed and needs to be recreated. • "Disputed" – Transaction has been disputed by the Customer and funds are being held in suspense until the dispute has been resolved. • "Refunded" – Funds were refunded back to the customer.
hash	String	Details of Hash generation are provided in a later section of this document.

The merchant server is not expected to reply to this message. If an important update is received, e.g. that the transaction has been paid it is recommended the merchant site polls Paynow for a status to confirm it matches the incoming message from Paynow.

On any status update it is vital the merchant site validates the hash in the message to confirm the message authenticity.

Polling for a Status Update

The merchant site can poll for a current transaction status to Paynow at any point, but it should only be done in the following two scenarios:

- The merchant site receives an important status update from Paynow and wants to poll Paynow to confirm the status.
- The merchant site is going to delete old or unpaid transactions, before doing this the merchant site should poll Paynow and confirm the transaction status before deleting it from their system.

To poll for a transaction status the merchant site should perform an empty HTTP POST to the pollurl sent by Paynow in transaction initiation or status update. Paynow will reply with a string formatted as an HTTP POST, i.e. each field separated by a & and Key Value pairs separated by an = and all Values URL Encoded, with the same fields as if it were posting a result to the merchant site.

An example of the result from Paynow is shown below:

```
reference=siteid123&paynowreference=1%2c082&amount=100.00&status=Created&pollurl=http%3a%2f%2flocalhost%3a7105%2fInterface%2fCheckPayment%2f%3fguid%3d811e0233-e9c6-474f-a01a-f2a4df1dde51&Hash=2D72F08C4F34B99DEC391E2A24F24C2598060B9F6D63CB0B961FEDAE7D7D69D6321931A18F8E1E0268DE5A4F72B5D76E5A8A767C810180D9D5AC921B444B51BA
```

Generating Hash

Any message to or from Paynow must include a hash and the hash must be validated to ensure the authenticity of the message. To generate a hash value for the message follow the steps:

1. Concatenate the values in the message for each element in their raw form (i.e. if it is from a result string URL decode any values first, if it is from a form post then this will already have been done).
2. Append the Integration Key which can be found by editing the appropriate receive payment link in the Receive Payment Links area of Paynow
3. UTF8 encode the string, note in PHP this should not be necessary, depending on your configuration.
4. Create a SHA512 hash of the string and output the result as upper case hexadecimal.

Code Examples

Please note these examples are not intended as examples of the best way of doing the hash, merely ones that are easy to understand.

PHP

```
private function CreateHash($values, $IntegrationKey){
            $string = "";
            foreach($values as $key=>$value) {
                if( strtoupper($key) != "HASH" ){
                    $string .= $value;
                }
            }
            $string .= $IntegrationKey;
            $hash = hash("sha512", $string);
            return strtoupper($hash);
        }
```

C#

```
private static string GenerateTwoWayHash(Dictionary<string, string> items, Guid guid)
{
            string concat = string.Join("", items.Select(c => (c.Value != null ? c.Value.Trim() : "")).ToArray());
            SHA512 check = SHA512.Create();
            byte[] resultArr = check.ComputeHash(Encoding.UTF8.GetBytes(concat + guid));
            return ByteArrayToString (resultArr);
}

public static string ByteArrayToString(byte[] ba)
{
             StringBuilder hex = new StringBuilder(ba.Length * 2);
             foreach (byte b in ba)
                          hex.AppendFormat("{0:x2}", b);
             return hex.ToString();
}
```

Change Log

1.0.2

- Added authemail to initiate transaction.

1.0.3

- Corrected capitalisation errors.
- Clarified reference to Merchant Key as Integration Key.

1.0.4

- Corrected C# example code.

1.0.5

- Added Setup section.
- Added Delivered as possible initial option if the user selects skip BuySafe.