

HMAC Authentication API

There are two methods Paystation uses to securely authenticate server requests in our system: IP limiting and HMAC authentication. This document describes our HMAC authentication implementation.

In cryptography, a keyed-hash message authentication code (HMAC) is a specific construction for calculating a message authentication code (MAC) involving a cryptographic hash function in combination with a secret cryptographic key. As with any MAC, it may be used to simultaneously verify both the data integrity and the authentication of a message.

We generate the HMAC authentication key used to hash your data within the Paystation system and pass this to you to use during your requests.

We currently support HMAC authentication on the following interfaces:

Payment Processor - <https://www.paystation.co.nz/direct/paystation.dll?>

Reporting System - <https://payments.paystation.co.nz/reporting/?>

Quick Lookup - <https://payments.paystation.co.nz/lookup/?>

To use HMAC authentication you need to append two extra variables to the URL/GET query string that you are using to initiate a transaction or lookup. This assumes that you are passing all other parameters in the request body as standard POST key/value pairs.

pstn_HMACTimestamp is a unix timestamp that you will need to generate right before you calculate the HMAC and send us the transaction. This value is used during HMAC calculation

pstn_HMAC is the HMAC hash of a concatenation of 3 byte mapped values, the timestamp, the string 'paystation', and the POST request body. This is generated using the SHA512 cryptographic algorithm and the HMAC authentication key provided by Paystation.

string to hash = byte array of timestamp (integer/string) + byte array of 'paystation' (string) + byte array of request body (string)

A PHP code example is shown below:

```
$authenticationKey='12345' //You will get this from Paystation

//example POST request body - This would include your actual
Paystation ID and Gateway ID
$postBody =
'paystation=_empty&pstn_am=100&pstn_pi=paystationID&pstn_gi=gatewayID&
pstn_ct=mastercard&pstn_cn=5123456789012346&pstn_ex=1305&pstn_ms=merc
hantsession0002331&pstn_2p=t&pstn_nr=t';

//HMAC generation
$hmactimestamp = 'paystation'; //webservice identification.

$hmactimestamp = time(); //unix timestamp
```

```
//Byte mapped concatenation of timestamp, webservice name and POST
body.
$macBody =
pack('a*', $macTimestamp).pack('a*', $macWebserviceName).pack('a*', $
postBody);
//Note : Timestamp can be byte mapped as an integer or a string, we
accept both.

//HMAC hash generated using SHA512 algorithm, hashing the macBody
using the authenticationKey
$macHash = hash_hmac('sha512', $macBody, $authenticationKey);

$macGetParams='&pstn_HMACTimestamp='.$macTimestamp.'&pstn_HMAC='.$mac
Hash;
```

These two extra parameters you've generated would then be added to the URL/GET query string of your transaction/lookup request.

The following is an example URL with the HMAC GET variables and subsequent POST variables for a 2-Party transaction:

```
https://www.paystation.co.nz/direct/paystation.dll?pstn_HMACTimestamp=
1391050533&pstn_HMAC=f8066b3fe7fefcf59ed5790620833765770eea9432ca06181
7f8abaaaf4552c89234a5e35ee82f957e738053450010e96b23e284ba8a43ea05826b9c
d1f10be70
```

POST variables:

```
paystation=_empty&pstn_am=100&pstn_pi=paystationID&pstn_gi=gatewayID&
pstn_ct=mastercard&pstn_cn=5123456789012346&pstn_ex=1305&pstn_ms=merc
hantsession0002331&pstn_2p=t&pstn_nr=t
```

HMAC Failures

In addition to all standard error messages, when using HMAC authentication you can now receive an error code of 160 (HMAC validation failed).