Overview

The PayAmigo payment platform validates API requests with a HMAC signature. Only requests with the correct signature are allowed to access the system resources.
The signature together with other important security information is expected to be sent in the request headers. You will receive an error message with HTTP status 401 if the signature is missing or invalid.
The document is to explain how to generate the signature.

API Caller Credentials

You will receive these credentials when your merchant account is created. If you don't have these credentials, please contact your account manager.

Name	Description
Merchant Account Name	Assigned by PayAmigo. The unique merchant account name
Caller Name	Assigned by PayAmigo. The api caller ID
Caller Password	Assigned by PayAmigo. The api caller password, it is also the encrypted secret for HMAC signature.

Request Security Headers

These HTTP request header values are required for each API call to the CS platform. Without any of these headers, the request is treated as an unauthorised request.

Header Name	Description
X-MerchantAccount	The Merchant Account name assigned to you.
X-CallerName	The name of the API Caller making the request
X-HMAC-Timestamp	Unix timestamp (seconds from epoch). Signatures generated with timestamps more than 30 minutes older than the request (as well as future timestamps) are rejected. e.g
X-HMAC-Signature	The generated signature

Generate the Signature

Parameter definitions:

- X-CallerName - the name of the caller
- X-MerchantAccount - the name of the merchant account
- X-HMAC-Timestamp - current Unix time in seconds. This MUST be in UTC
- request_path - relative path of API, include query params if present
- request_content - The HTTP method body if present. Put it as an empty string (“”) if it is not present.

The signature can be generated as follows (pseudo-code):

```
var timestamp = seconds_from_epoch_utc
var message = string_concat(callerName, merchantAccountName,
timestamp, request_path, request_body)
var signature = hmac_sha256_as_hexadecimal(api_secret, message)
```

As a concrete example, suppose we have and caller calling the healthcheck endpoint:

- X-CallerName = “$apicaller”
- X-MerchantAccount = “Demo_Merchant”
- X-HMAC-Timestamp = “1633767872”
- Caller’s password = “aP%eUmGp$FYernKtUdq3”
- Request url = “https://sandbox.payamigo.com/api/v3/healthcheck”

It goes:
message = “$apicallerDemo_Merchant1633767872/api/v3/healthcheck”
signature =
“B6693ABCCB887DD65B8DD05FAC5AC19653154C63006896ED4912EAAEBF10FEB1”

Finally the request goes as below (in CURL):

```
curl --location --request GET 'https://sandbox.payamigo.com/api/v3/healthcheck' \
--header 'X-MerchantAccount: Demo_Merchant' \
--header 'X-CallerName: $apicaller' \
--header 'X-HMAC-Timestamp: 1633767872' \
--header 'X-HMAC-Signature:
B6693ABCCB887DD65B8DD05FAC5AC19653154C63006896ED4912EAAEBF10FEB1' \
--header 'Content-Type: application/json'
```

You will receive a HTTP 200 status with an empty response if your signature is correct.
You will see this error message when the signature is wrong:

```
{"requestId":"44d123b4-5cdc-4522-908b-3d5765addbf3","errorCode":"authent
ication_error","message":"HMAC Authentication failed. Invalid name or
password"}
```

HMAC Signature Code Examples

Java

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.Locale;


/**
*
* @param requestPath, the request context path including query string, e.g
"/api/v3/charges"
* @param requestBody: the request body content
* @return
* @throws NoSuchAlgorithmException
* @throws InvalidKeyException
*/
public String calculateHMAC(String requestPath, String requestBody) throws
NoSuchAlgorithmException, InvalidKeyException {
   //use your configured values
   String apiSecret = "aP%eUmGp$FYernKtUdq3";
   String merchantAccount = "Demo_Merchant";
   String callerName = "$apicaller";

   long timestamp = System.currentTimeMillis()/1000;

   Mac mac = Mac.getInstance("HmacSHA256");
   SecretKeySpec keySpec = new SecretKeySpec(apiSecret.getBytes(),
"HmacSHA256");
   mac.init(keySpec);

   StringBuilder message = new
StringBuilder().append(callerName).append(merchantAccount);
   message.append(timestamp);

   if (requestPath != null && requestPath.trim().length() > 0) {
       message.append(requestPath);
   }
   if (requestBody != null && requestBody.trim().length() > 0) {
       message.append(requestBody);
   }
   byte[] signatureBytes = mac.doFinal(message.toString().getBytes());
   return
```

```
org.apache.commons.codec.binary.Hex.encodeHexString(signatureBytes).toUpperCase(
Locale.ROOT);
}
```

JavaScript

```
//message = X-CallerName + X-MerchantAccount + X-HMAC-Timestamp + path + body
var timestamp = Math.floor(new Date().getTime()/1000);
var path = pm.request.url.getPathWithQuery();
var message = environment['$apicaller'] + environment['Demo_Merchant'] + timestamp + path
+ (pm.request.body.raw || "");
var secret = environment['aP%eUmGp$FYernKtUdq3'];
var signature = CryptoJS.HmacSHA256(message, secret).toString().toUpperCase();
```