KwikPaisa Signature Generation Logic

KwikPaisa APIs use HMAC SHA256 based request signing.

The signature is generated using:

HMAC_SHA256(payload + timestamp, secret_key)

The payload must:

- Be recursively sorted
- Be JSON encoded
- Use unescaped slashes

PHP

```
$data=[
   'order_id'=>'123456',
   'amount'=>'100'
];

function sortPayload($data)
{
   if(!is_array($data)) return $data;

   ksort($data);

   foreach($data as $key=>$value){
      $data[$key]=sortPayload($value);
   }

   return $data;
}

$payload=json_encode(
   sortPayload($data),
   JSON_UNESCAPED_SLASHES
);

$timestamp=time();

$signature=hash_hmac(
   'sha256',
   $payload.$timestamp,
   $secretKey
);
```

Java

```
import com.fasterxml.jackson.databind.ObjectMapper;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.util.*;

public class SignatureGenerator {

    public static String generateSignature(
        Map<String, Object> data,
        String secretKey,
        long timestamp
    ) throws Exception {

        TreeMap<String, Object> sortedData = new TreeMap<>(data);

        ObjectMapper mapper = new ObjectMapper();

        String payload = mapper.writeValueAsString(sortedData);

        String signString = payload + timestamp;

        Mac sha256Hmac = Mac.getInstance("HmacSHA256");

        SecretKeySpec secretKeySpec = new SecretKeySpec(
            secretKey.getBytes(),
            "HmacSHA256"
        );

        sha256Hmac.init(secretKeySpec);

        byte[] hash = sha256Hmac.doFinal(
            signString.getBytes()
        );

        StringBuilder hexString = new StringBuilder();

        for (byte b : hash) {
            hexString.append(String.format("%02x", b));
        }

        return hexString.toString();
    }
}
```

.NET (C#)

```
using System;
using System.Collections.Generic;
using System.Security.Cryptography;
using System.Text;
using Newtonsoft.Json;

class Program
{
   static void Main()
   {
      var data = new SortedDictionary<string, object>
      {
         { "order_id", "123456" },
         { "amount", "100" }
      };

      string payload = JsonConvert.SerializeObject(data);

      long timestamp = DateTimeOffset.UtcNow.ToUnixTimeSeconds();

      string signString = payload + timestamp;

      string secretKey = "your_secret_key";

      using(var hmac = new HMACSHA256(
         Encoding.UTF8.GetBytes(secretKey)))
      {
         byte[] hash = hmac.ComputeHash(
            Encoding.UTF8.GetBytes(signString)
         );

         string signature = BitConverter
            .ToString(hash)
            .Replace("-", "")
            .ToLower();

         Console.WriteLine(signature);
      }
   }
}
```

Flutter / Dart

```
import 'dart:convert';
import 'package:crypto/crypto.dart';

void main() {

  Map<String, dynamic> data = {
    "order_id": "123456",
    "amount": "100"
  };

  final sortedKeys = data.keys.toList()..sort();

  final sortedData = {
    for (var key in sortedKeys) key: data[key]
  };

  String payload = jsonEncode(sortedData);

  int timestamp = DateTime.now()
      .millisecondsSinceEpoch ~/ 1000;

  String secretKey = "your_secret_key";

  String signString = payload + timestamp.toString();

  var hmacSha256 = Hmac(
    sha256,
    utf8.encode(secretKey)
  );

  var digest = hmacSha256.convert(
    utf8.encode(signString)
  );

  String signature = digest.toString();

  print(signature);
}
```

Node.js

```
const crypto = require('crypto');

const data = {
   order_id: '123456',
   amount: '100'
};

const sortedData = Object.keys(data)
   .sort()
   .reduce((obj, key) => {
      obj[key] = data[key];
      return obj;
   }, {});

const payload = JSON.stringify(sortedData);

const timestamp = Math.floor(Date.now() / 1000);

const secretKey = 'your_secret_key';

const signature = crypto
   .createHmac('sha256', secretKey)
   .update(payload + timestamp)
   .digest('hex');

console.log(signature);
```

Python

```
import json
import hmac
import hashlib
import time

data = {
   "order_id": "123456",
   "amount": "100"
}

sorted_data = dict(sorted(data.items()))

payload = json.dumps(
   sorted_data,
   separators=(',', ':')
)

timestamp = str(int(time.time()))

secret_key = "your_secret_key"

signature = hmac.new(
   secret_key.encode(),
   (payload + timestamp).encode(),
   hashlib.sha256
).hexdigest()

print(signature)
```