日日是Oracle APEX

Oracle APEXを使った作業をしていて、気の付いたところを忘れないようにメモをとります。

2023年2月28日火曜日

PayPay Open Payment APIの呼び出しに必要なHMACオブジェクトを生成する

PayPayのOpen Payment APIのSDKとして提供されているのは、Python、Node、Java、PHPのみでPL/SQLは含まれていません。また、ドキュメントのサンプル・コードも同様です。

Oracle APEXにPayPayのWeb決済を組み込むには、API認証をできるようにする必要があります。PayPayが提供しているAPI認証の説明を読んで、HMACオブジェクトを生成するPL/SQLパッケージを作ってみました。

```
create or replace package util_paypay_api
as
/**
 * PayPayのAPI認証に必要なHMAC authヘッダーに使用する
 * HMACオブジェクトを生成する。
 *
 * 参照: https://www.paypay.ne.jp/opa/doc/jp/v1.0/webcashier#tag/API
 *
 * @param p_request_body リクエストで渡されるbody。
 * @param p_content_type リクエストヘッダーで渡されるコンテンツタイプ。
 * @param p_request_url リクエストURL
 * @param p_http_method HTTPメソッド
 * @param p_nonce ランダムに生成された文字列。8文字(多分8バイト)が推奨。null指定でランダムに生成した値を返す。
 * @param p_epoch 現在のエポックタイムスタンプ。サーバー時刻との差が2分未満。null指定で現在時刻を返す。
 * @param p_apikey  PayPayから発行されたAPI Key。
 * @param p_apikey_secret PayPayから発行されたAPI Key secret。
 *
 * @p_hash Base64でエンコーディングされたコンテンツのハッシュ値。Step1の値。
 * @p_macdata Base64でエンコーディングされたHMACオブジェクト。
 * @p_header HMAC authヘッダーに与える値。
 */
procedure generate_hmac_auth(
    p_request_body   in varchar2 default null
    ,p_content_type  in varchar2 default null
    ,p_request_url   in varchar2
    ,p_http_method   in varchar2
    ,p_nonce         in out varchar2
    ,p_epoch         in out varchar2
    ,p_apikey        in varchar2
```

```
    ,p_apikey_secret in varchar2
    ,p_hash    out varchar2
    ,p_macdata out varchar2
    ,p_header  out varchar2
);

/**
 * generate_hmac_authの簡易版。
 */
function generate_hmac_auth_header(
    p_request_body   in varchar2 default null
    ,p_content_type  in varchar2 default null
    ,p_request_url   in varchar2
    ,p_http_method   in varchar2
    ,p_apikey        in varchar2
    ,p_apikey_secret in varchar2
)
return varchar2;

end util_paypay_api;
/

create or replace package body util_paypay_api
as
C_DELIMITER constant varchar2(1) := CHR(10);

/*
 * OracleのTIMESTAMP型のデータをUNIX時間に変換する。
 */
function unixtime(p_timestamp in timestamp)
return pls_integer
is
    l_date date;
    l_epoc number;
begin
    l_date := sys_extract_utc(p_timestamp);
    l_epoc := l_date - date'1970-01-01';
    return l_epoc * 24 * 60 * 60;
end unixtime;

/**
 * RAW型をBASE64エンコードした文字列として返す。
 */
function to_base64_from_raw(t in raw)
return varchar2
as
    l_base64 varchar2(32767);
```

```
begin
    l_base64 := utl_raw.cast_to_varchar2(utl_encode.base64_encode(t));
    l_base64 := replace(l_base64, chr(13)||chr(10), '');
    /* translateは不要 */
    -- l_base64 := trim(translate(l_base64, '+/=', '-_ '));
    return l_base64;
end to_base64_from_raw;

/**
 * 一時LOBに文字列を追記する。
 *
 * @param src 一時BLOB
 * @param str 追記する文字列。デフォルトはラインフィード。
 */
procedure append_varchar2(
    src in out blob
    ,str in varchar2 default C_DELIMITER
)
as
    l_raw raw(32767);
begin
    /* strがnullの場合は何もしない。 */
    if str is null then
        return;
    end if;
    l_raw := utl_i18n.string_to_raw(str,'AL32UTF8');
    dbms_lob.writeappend(
        lob_loc => src
        ,amount => utl_raw.length(l_raw)
        ,buffer => l_raw
    );
end append_varchar2;

procedure generate_hmac_auth(
    p_request_body   in varchar2
    ,p_content_type  in varchar2
    ,p_request_url   in varchar2
    ,p_http_method   in varchar2
    ,p_nonce         in out varchar2
    ,p_epoch         in out varchar2
    ,p_apikey        in varchar2
    ,p_apikey_secret in varchar2
    ,p_hash    out varchar2
    ,p_macdata out varchar2
    ,p_header  out varchar2
)
as
```

```
    l_content_type varchar2(32767);
    l_source_blob blob;
    l_hash raw(32767);
    l_key  raw(32767);
    l_mac  raw(32767);
begin
    /* epochがnullであれば、現在時刻のepochを設定する。 */
    if p_epoch is null then
        p_epoch := to_char(unixtime(current_timestamp));
    end if;
    /* nonceがnullであれば、8文字程度の文字列を設定する。 */
    if p_nonce is null then
        p_nonce := lower(rawtohex(dbms_crypto.randombytes(4)));
    end if;

    /* Step 1 */
    if p_http_method <> 'GET' and p_request_body is not null then
        dbms_lob.createTemporary(
            lob_loc => l_source_blob
            ,cache => true
            ,dur => dbms_lob.call
        );
        append_varchar2(l_source_blob, p_content_type);
        append_varchar2(l_source_blob, p_request_body);
        l_hash := dbms_crypto.hash(
            src => l_source_blob
            ,typ => dbms_crypto.hash_md5
        );
        dbms_lob.freeTemporary(l_source_blob);
        p_hash := to_base64_from_raw(l_hash);
        l_content_type := p_content_type;
    else
        /* 本体がないときはemptyとする。 */
        p_hash := 'empty';
        l_content_type := 'empty';
    end if;

    /* Step 2 */
    dbms_lob.createTemporary(
        lob_loc => l_source_blob
        ,cache => true
        ,dur => dbms_lob.call
    );
    append_varchar2(l_source_blob, p_request_url);
    append_varchar2(l_source_blob);
    append_varchar2(l_source_blob, p_http_method);
    append_varchar2(l_source_blob);
```

```
    append_varchar2(l_source_blob, p_nonce);
    append_varchar2(l_source_blob);
    append_varchar2(l_source_blob, p_epoch);
    append_varchar2(l_source_blob);
    append_varchar2(l_source_blob, l_content_type);
    append_varchar2(l_source_blob);
    append_varchar2(l_source_blob, p_hash);

    /* Step 3 */
    l_key := utl_i18n.string_to_raw(p_apikey_secret,'AL32UTF8');
    l_mac := dbms_crypto.mac(
        src => l_source_blob
        ,typ => dbms_crypto.HMAC_SH256
        ,key => l_key
    );
    dbms_lob.freeTemporary(l_source_blob);
    p_macdata := to_base64_from_raw(l_mac);

    /* ヘッダーの生成 */
    p_header := p_apikey || ':' || p_macdata || ':' || p_nonce || ':' || p_epoch || ':' || p_ha
end generate_hmac_auth;

function generate_hmac_auth_header(
    p_request_body   in varchar2
    ,p_content_type  in varchar2
    ,p_request_url   in varchar2
    ,p_http_method   in varchar2
    ,p_apikey        in varchar2
    ,p_apikey_secret in varchar2
)
return varchar2
as
    l_nonce   varchar2(32767) := null;
    l_epoch   varchar2(32767) := null;
    l_hash    varchar2(32767);
    l_macdata varchar2(32767);
    l_header  varchar2(32767);
begin
    generate_hmac_auth(
        p_request_body  => p_request_body
        ,p_content_type => p_content_type
        ,p_request_url  => p_request_url
        ,p_http_method  => p_http_method
        ,p_nonce        => l_nonce
        ,p_epoch        => l_epoch
        ,p_apikey       => p_apikey
        ,p_apikey_secret => p_apikey_secret
```

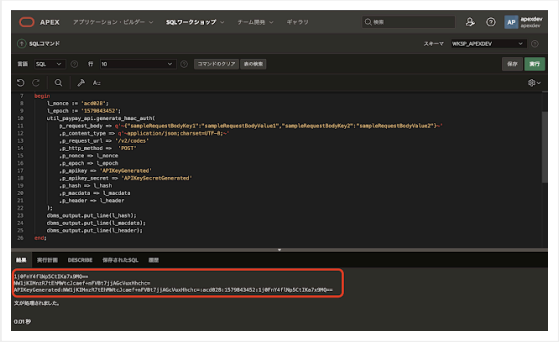
```
        ,p_hash         => l_hash
        ,p_macdata      => l_macdata
        ,p_header       => l_header
    );
    return 'hmac OPA-Auth:' || l_header;
end generate_hmac_auth_header;

end util_paypay_api;
/
```

util_paypay_api.sql hosted with ❤ by GitHub view raw

API認証のドキュメントにある**認証オブジェクトに必要なパラメータ**に記載されている**Example**の値を引数として、作成したプロシージャgenerate_hmac_authを呼び出してみます。

```
declare
    l_hash    varchar2(32767);
    l_macdata varchar2(32767);
    l_header  varchar2(32767);
    l_nonce varchar2(80);
    l_epoch varchar2(80);
begin
    l_nonce := 'acd028';
    l_epoch := '1579843452';
    util_paypay_api.generate_hmac_auth(
        p_request_body => q'~{"sampleRequestBodyKey1":"sampleRequestBodyValue1","sampleRequestB
        ,p_content_type => q'~application/json;charset=UTF-8;~'
        ,p_request_url => '/v2/codes'
        ,p_http_method =>  'POST'
        ,p_nonce => l_nonce
        ,p_epoch => l_epoch
        ,p_apikey => 'APIKeyGenerated'
        ,p_apikey_secret => 'APIKeySecretGenerated'
        ,p_hash => l_hash
        ,p_macdata => l_macdata
        ,p_header => l_header
    );
    dbms_output.put_line(l_hash);
    dbms_output.put_line(l_macdata);
    dbms_output.put_line(l_header);
end;
```

generate_hmac_auth_test.sql hosted with ❤ by GitHub view raw



結果として以下が印刷されました。

1j0FnY4flNp5CtlKa7x9MQ==
NW1jKlMnzR7tEhMWtcJcaef+nFVBt7jjAGcVuxHhchc=
APIKeyGenerated:NW1jKlMnzR7tEhMWtcJcaef+nFVBt7jjAGcVuxHhchc=:acd028:1579843452:
1j0FnY4flNp5CtlKa7x9MQ==

実装を検証するために、PayPayのAPI認証のドキュメントに記載されているJavaのサンプルを、同じ引数を与えて実行してみます。

```
import java.security.MessageDigest;
import java.nio.charset.StandardCharsets;
import java.util.Base64;
import javax.crypto.spec.SecretKeySpec;
import javax.crypto.Mac;

public class PayPayAuth {
    public static void main(String[] args)
    {
        try {
            /* Step 1 */
            String requestBody = "{\"sampleRequestBodyKey1\":\"sampleRequestBodyValue1\",\"samp
            String contentType = "application/json;charset=UTF-8;";
            MessageDigest md = MessageDigest.getInstance("MD5");
            md.update(contentType.getBytes(StandardCharsets.UTF_8));
            md.update(requestBody.getBytes(StandardCharsets.UTF_8));
            String hash = new String(
                Base64.getEncoder().encode(md.digest()),
                StandardCharsets.UTF_8);
            System.out.println(hash);

            /* Step 2 */
            String requestUrl = "/v2/codes";
            String httpMethod = "POST";
            String nonce = "acd028";
            String epoch = "1579843452";
            String DELIMITER = "\n";
            byte[] hmacData = new StringBuffer()
                .append(requestUrl)
```

```
                .append(DELIMITER)
                .append(httpMethod)
                .append(DELIMITER)
                .append(nonce)
                .append(DELIMITER)
                .append(epoch)
                .append(DELIMITER)
                .append(contentType)
                .append(DELIMITER)
                .append(hash != null ? hash : "")
                .toString()
                .getBytes(StandardCharsets.UTF_8);

            /* Step 3 */
            String apiKeySecret = "APIKeySecretGenerated";
            byte[] dataToSign = hmacData;
            SecretKeySpec signingKey = new SecretKeySpec(apiKeySecret.getBytes(StandardCharsets
                "HmacSHA256");
            Mac sha256HMAC = Mac.getInstance("HmacSHA256");
            sha256HMAC.init(signingKey);
            byte[] rawHmac = sha256HMAC.doFinal(dataToSign);
            System.out.println(Base64.getEncoder().encodeToString(rawHmac));
        } catch (Exception e) {
            e.printStackTrace(System.err);
        }
    }
}
```

PayPayHmacAuth.java hosted with ❤️ by GitHub view raw

```
% javac PayPayAuth.java
% java PayPayAuth
1j0FnY4flNp5CtIKa7x9MQ==
NW1jKIMnzR7tEhMWtcJcaef+nFVBt7jjAGcVuxHhchc=
%
```

以下の結果が得られます。

1j0FnY4flNp5CtIKa7x9MQ==
NW1jKIMnzR7tEhMWtcJcaef+nFVBt7jjAGcVuxHhchc=

Base64でエンコーディングされたHMACオブジェクトの結果が同じなので、PL/SQLのコードも概ね正しい実装になっていそうです。

API認証に使用するHMACオブジェクトを生成できたので、APEXからPayPay Open Payment APIを呼び出せるでしょう。

完

Yuji N. 時刻: 15:37

共有

‹ ホーム ›

ウェブ バージョンを表示

自己紹介

Yuji N.

日本オラクル株式会社に勤務していて、Oracle APEXのGroundbreaker Advocateを拝命しました。
こちらの記事につきましては、免責事項の参照をお願いいたします。

詳細プロフィールを表示