Dokumentacja techniczna integracji sklepu z bramką płatności imoje

Instrukcja integracji sklepu z bramką płatności imoje

Wersja dokumentu

Wersja	Data	Autor	Kontakt
1.0.0	14.03.2018	ING	kontakt.tech@imoje.pl
1.1.0	11.05.2018	ING	kontakt.tech@imoje.pl
1.2.0	29.06.2018	ING	kontakt.tech@imoje.pl
1.3.0	21.02.2019	ING	kontakt.tech@imoje.pl
1.3.1	19.03.2019	ING	kontakt.tech@imoje.pl
1.3.2	16.07.2019	ING	kontakt.tech@imoje.pl
1.3.3	26.07.2019	ING	kontakt.tech@imoje.pl

Spis treści

1. Wykonanie transakcji

Aby dokonać transakcję, należy wysłać żądanie metodą POST na adres:

- `produkcja`

```
https://paywall.imoje.pl/pl/payment
```

- `sandbox`

```
https://sandbox.paywall.imoje.pl/pl/payment
```

następujące parametry:

Pola wymagane:

- `serviceId` - identyfikator sklepu klienta,
- `merchantId` - identyfikator klienta,
- `amount` - kwota transakcji podana w groszach,
- `currency` - waluta,
- `orderId` - ID zamówienia, dopuszczalne znaki: A-Za-z0-9_-`#&'",./\ oraz białe znaki i znaki z zakresu UNICODE 00C0 - 02C0 (m.in. polskie znaki diakrytyczne),
- `customerFirstName` - imię osoby składającej zamówienie,
- `customerLastName` - nazwisko osoby składającej zamówienie,
- `customerEmail` - adres e-mail osoby składającej zamówienie,
- `signature` - wyliczona sygnatura.

Pola opcjonalne:

- `customerPhone` - numer telefonu osoby składającej zamówienie,
- `urlSuccess` - adres URL, na który użytkownik będzie przekierowany po pomyślnie zakończonej transakcji,
- `urlFailure` - adres URL, na który użytkownik będzie przekierowany po błędnie dokonanej transakcji,
- `urlReturn` - adres URL, na który użytkownik będzie przekierowany po dokonaniu transakcji,
- `simp` - pełny numer rachunku wirtualnego którego dotyczy wpłata. Dotyczy tylko sklepów, które obsługują płatności SIMP,
- `orderDescription` - tytuł transakcji, dopuszczalne znaki: A-Za-z0-9_-`#&'",./\ oraz białe znaki i znaki z zakresu UNICODE 00C0 - 02C0 (m.in. polskie znaki diakrytyczne),
- `visibleMethod` - metody płatności widoczne na bramce płatności. Dostępne wartości: card, pbl, blik, twisto. Jeśli tego pola nie ma lub jest puste to są wyświetlane wszystkie włączone w sklepie klienta metody płatności. Istnieje wiele konfiguracji wyświetlanych metod płatności. Należy je zawsze rozdzielać przecinkiem (np. „card,pbl”),
- `twistoData` - pole wymagane przy płatnościach Twisto. Wartością tego parametru powinna być kompletna odpowiedź z api Twisto. Opis sposobu pobrania wymaganych parametrów dla wartości twistoData jest opisana w punkcie 4,

- `validTo` - data ważności linku płatności jako timestamp w sekundach, jeżeli nie jest przekazane to link ważny jest zawsze,

Przykładowa zawartość formularza wysyłana metodą POST:

```
<input type="hidden" value="63f574ed-d90d-4abe-9cs1-39117584a7b7" name="serviceId">
<input type="hidden" value="6yt3gjtm9p1odfgx8491" name="merchantId">
<input type="hidden" value="300" name="amount">
<input type="hidden" value="PLN" name="currency">
<input type="hidden" value="123" name="orderId">
<input type="hidden" value="Example transaction" name="orderDescription">
<input type="hidden" value="John" name="customerFirstName">
<input type="hidden" value="Doe" name="customerLastName">
<input type="hidden" value="johndoe@domain.com" name="customerEmail">
<input type="hidden" value="501501501" name="customerPhone">
<input type="hidden" value="https://your-shop.com/success" name="urlSuccess">
<input type="hidden" value="https://your-shop.com/failure" name="urlFailure">
<input type="hidden" value="https://your-shop.com/return" name="urlReturn">
<input type="hidden" value="05105000995129330000123456" name="simp">
<input type="hidden" value="card,pbl" name="visibleMethod">
<input type="hidden" value='{"transaction_id":"s2kjtgl123d5261e5151s426","status":"accepted"}'
name="twistoData">
<input type="hidden"
value="cd5024f5ce5e6ff47990fe60572758fbcbcd6e3c04895d6815932b2d14e04ffd;sha256"
name="signature">
```

Wartości parametrów z adresami powinny być pełnymi adresami (absolute URL).

1.1. Wyliczanie sygnatury

Sygnaturę wyliczamy w następujący sposób:

1. Sortujemy alfabetycznie, rosnąco, po kluczach parametry zamówienia.
2. Łączymy parametry w następujący sposób: `parametr1=wartosc1&parametr2=wartosc2...parametrN=wartoscN`. Wynik zapisujemy do zmiennej (zwanej dalej `body`).
3. Wyliczamy sygnaturę metodą hashowania `sha256`: `sha256(body + private_key`. Wynik hashowania zapisujemy do zmiennej (zwanej dalej `signature`).
4. Do wyliczonej sygnatury dopisujemy po średniku użytą metodę hashowania: `signature + ';sha256'`.

1.1.1. Przykład wyliczenia sygnatury

```
/**
 * @param array  $orderData
 * @param string $serviceKey
 * @param string $hashMethod
 *
 * @return string
 */
function createSignature($orderData, $serviceKey, $hashMethod)
```

```
{
    $data = prepareData($orderData);

    return hash($hashMethod, $data . $serviceKey);
}

/**
 * @param array  $data
 * @param string $prefix
 *
 * @return string
 */
function prepareData(
    $data,
    $prefix = ''
) {
    ksort($data);
    $hashData = [];
    foreach($data as $key => $value) {
        if($prefix) {
            $key = $prefix . '[' . $key . ']';
        }
        if(is_array($value)) {
            $hashData[] = prepareData($value, $key);
        } else {
            $hashData[] = $key . '=' . $value;
        }
    }

    return implode('&', $hashData);
}

$hashMethod = 'sha256';

$serviceKey = 'eAyhFLuHgwl5hu-32GM8QVlCVMWRU0dGjH1c';

$fields = [
 'merchantId' => '6yt3gjtm9p1odfgx8491',
  'serviceId' => '63f574ed-d90d-4abe-9cs1-39117584a7b',
  'amount' => '300',
  'currency' => 'PLN',
  'orderId' => '123',
  'orderDescription' => 'Example transaction',
  'customerFirstName' => 'John',
  'customerLastName' => 'Doe',
  'customerEmail' => 'johndoe@domain.com',
  'customerPhone' => '501501501',
  'urlSuccess' => 'https://your-shop.com/success',
  'urlFailure' => 'https://your-shop.com/failure',
  'urlReturn' => 'https://your-shop.com/return',
  'twistoData' => '{"transaction_id":"s2kjtgl123d5261e5151s426","status":"accepted"}',
];
$result = createSignature($fields, $serviceKey, $hashMethod) . ';' . $hashMethod;
```

W tym przypadku wartość zmiennej `$result` będzie miała następującą wartość:

```
cd5024f5ce5e6ff47990fe60572758fbcbcd6e3c04895d6815932b2d14e04ffd;sha256
```

2. Obsługa notyfikacji

2.1. Weryfikacja dostarczonych notyfikacji transakcji

> Adres notyfikacyjny można ustawić panelu administracyjnym imoje. W szczegółach sklepu.

Aby upewnić się, że notyfikacja została przysłana z zaufanego źródła, należy dokonać jej weryfikacji. Każda dostarczona notyfikacja, posiada nagłówek `Content-Type` (który rozpoczyna się od: `application/json`) oraz sygnaturę.

Należy zweryfikować czy sygnatura została poprawnie podpisana. Całą zawartość notyfikacji łączymy z kluczem private_key i porównujemy wynik z przysłaną sygnaturą w nagłówku `X-Imoje-Signature`. Jeżeli wynik zgadza się - możemy być pewni, że notyfikacja została wysłana z poprawnego źródła.

```
$mySignature = hash($hashMethod, $payload . $serviceKey);
if ($mySignature === $headerSignature) {
// Notyfikacja zweryfikowana poprawnie. Przetwarzaj dalej.
} else {
// Notyfikacja zweryfikowana negatywnie. Ignoruj notyfikację.
}
```

Zmienna `$headerSignature` ma wartość `signature` w nagłówku notyfikacji: `X-Imoje-Signature`, czyli na przykład:

```
X-Imoje-Signature: merchantid=6yt3gjt1234f8h9xsdqz;serviceid=53f574ed-d4ad-aabe-9981-
39ed7584a7b7;signature=cd5024f5ce5e6ff47990fe60572758fbcbcd6e3c04895d6815932b2d14e04ffd;
sha256
```

> Notyfikacje są wysyłane z adresów IP z zakresów 54.37.185.64/28 i 54.37.185.80/28

2.2. Opis budowy notyfikacji

Kluczowymi nagłówkami są:

```
X-Imoje-Signature: merchantid=6yt3gjt1234f8h9xsdqz;serviceid=53f574ed-d4ad-aabe-9981-
39ed7584a7b7;signature=5e2ac79f4f02d368cdd6eaee17d2089dec13577c1d6e3364d6fc123c85029e8
2;alg=sha256
Content-Type: application/json; charset=UTF-8
```

gdzie:

- `merchantid` - identyfikator klienta w imoje,
- `serviceid` - identyfikator sklepu w imoje,
- `signature` - podpis notyfikacji,
- `alg` - algorytm funkcji skrótu (możliwe wartości: sha256).

Zawartość notyfikacji (body):


```
{
 "transaction": {
   "id": "51e958a8-c0e6-4537-b388-3dda226774c2",
   "type": "sale",
   "status": "rejected",
   "source": "web",
   "created": 1516360217,
   "modified": 1516360221,
   "notificationUrl": "https://notification-url.pl/file.php",
   "serviceId": "63a554ed-d4ad-407e-9981-39ed7548a7b7",
   "amount": 100,
   "currency": "PLN",
   "title": "124",
   "orderId": "124",
   "paymentMethod": "ing",
   "paymentMethodCode": "ing"
 }
}
```


- `id` - identyfikator zamówienia przypisany przez system imoje,
- `type` - w zależności od transakcji, wartość `sale` występuje przy sprzedaży, `refund` dla zwrotu,
- `status` - dla płatności zaakceptowanej status brzmi settled, dla odrzuconej rejected,
- `source` - dla tego parametru wartością może być `api` lub `web`.
- `serviceId` - unikalny identyfikator sklepu w którym została wykonana transakcja,
- `amount` - wartość zamówienia podana w najmniejszej jednostce pieniężnej waluty,
- `title` - tytuł płatności (przysłany ze sklepu),
- `orderId` - numer zamówienia (przysłany ze sklepu), dopuszczalne znaki: A-Za-z0-9_-`#&'",./\ oraz białe znaki i znaki z zakresu UNICODE 00C0 - 02C0 (m.in. polskie znaki diakrytyczne).

3. Wykonanie zwrotu

Poprawne wykonanie zwrotu polega na wysłaniu żądania metodą POST na adres:

```
https://api.imoje.pl/v1/merchant/{merchantId}/transaction/{transactionId}/refund
```

gdzie:

- `merchantId` - identyfikator klienta,
- `transactionId` - unikalny identyfikator dla każdej transakcji której dotyczy zwrot.

W zawartości żądania należy wprowadzić:

```
{
  "type": "refund",
  "serviceId": "12341234-3efa-4ea2-b193-59401da40f18",
  "amount": 100
}
```

gdzie:

- `serviceId` - identyfikator sklepu klienta,
- `amount` - kwota do zwrotu.

W żądaniu powinny być również zawarte nagłówki:

```
Content-Type: application/json
Authorization: {method} {token}
```

Dla różnych metod zawartość nagłówka `Authorization` będzie inna.

Metoda	Zawartość nagłówka
Token autoryzacyjny	Bearer {token}

Parametr `token` - to ciąg znaków dostępny w panelu merchanta dla odpowiedniej metody.

4. Twisto

> Twisto dla sklepów w trybie testowym nie jest dostępne. Symulację można dokonać jedynie w integracji z bramką płatności - do formularza należy dodać parametr `twistoData` z wartością `{"status":"accepted-verification-required","transaction_id":"sandbox"}`

Aby rozpocząć płatność Twisto należy przygotować **zebrane dane (punkt 4.4.)** z koszyka.

Podczas płatności, należy wykonać następujące kroki:

1. zaimportować bibliotekę JavaScript i ustawić klucz publiczny, którego pobranie zostało opisane w punkcie 4.2.:

```
<script type="text/javascript">
  var _twisto_config = {
    public_key: 'live_pk_gs8fcj9p2mbasxhlyte236hrupm5lubqaj7qzwh7kfuihoekud',
    script: 'https://api.twisto.pl/v2/lib/twisto.js'
  };

  (function (e, g, a) {
    function h(a) {
      return function () {
        b._.push([a, arguments])
      }
    }

    var f = ["check"], b = e || {}, c = document.createElement(a);
    a = document.getElementsByTagName(a)[0];
    b._ = [];
    for (var d = 0; d < f.length; d++) {
      b[f[d]] = h(f[d]);
    }
    this[g] = b;
    c.type = "text/javascript";
    c.async = !0;
    c.src = e.script;
    a.parentNode.insertBefore(c, a);
    delete e.script
  }).call(window, _twisto_config, "Twisto", "script");
</script>
```

2. wywołać funkcję Twisto.check z argumentami:
    - data (string) - zaszyfrowane dane płatności (4.4.) w formie ciągu znaków zakodowane w base64,
    - success (function) - przeprowadzenie dalszych czynności po poprawnej lub niepoprwanej odpowiedzi aplikacji Twisto,
    - error (function) - przeprowadzenie dalszych czynności w momencie gdy pojawił się problem podczas przesyłania danych
    - options (object)[opcjonalne] - zawartość tego obiektu musi być funkcją, a dostępna funkcja to:
        - processingStarted (function) - wywołane po poprawnym ropoczęciu procesowania.

Przykładowe wywołanie funkcji:

```
Twisto.check("{data}", function (response) {
        //success
    }, function (response) {
        //error
    }, {
        "processingStarted": function () {
            //processingStarted
        }
    }
);
```

Przykładowa poprawna odpowiedź aplikacji Twisto:

```
{
  "transaction_id": "1ad5mzbc9hsynjbkp8dlsqt7",
  "status": "accepted"
}
```

Opis statusów:

Status	Opis
accepted	zamówienie zostało pozytywnie ocenione przez system, można kontynuować płatność
accepted-verification-required	wymagana jest dodatkowa weryfikacja płatnika
rejected	zamówienie zostało odrzucone przez system

gdzie:

- `accepted` - należy wysłać standardowe zapytanie do api imoje (punkt 1) z zamówieniem i dodać parametr `twistoTransactionId` z wartością otrzymanego parametru `transaction_id`,
- `accepted-verification-required` - należy wysłać standardowe zapytanie do api imoje (punkt 1.) z zamówieniem i dodać parametr `twistoTransactionId` z wartością otrzymanego parametru `transaction_id`,
- `rejected` - status ten oznacza że Twisto odrzuciło zapytanie i należy pozwolić płatnikowi wybrać inną metodę płatności.

> W przypadku integracji z bramką płatności otrzymaną odpowiedź należy dołączyć do formularza z parametrem `twistoData`.

4.1. Obsługa komunikatów

Proces płatności wygląda w następujący sposób:

1. Wybranie metody "Kup teraz zapłać później"
2. Przesłanie danych zamówienia do Twisto
3. Odebranie odpowiedzi od Twisto:
    - odpowiedź pozytywna ("status": "accepted"), ("status": "accepted-verification-required") - przekierowanie na stronę paywall,
    - odpowiedź negatywna ("status": "rejected") - powoduje wyświetlenie komunikatu modalnego o braku możliwości zrealizowania transakcji poprzez Twisto. Wykazuje powody odmowy oraz przekierowuje do checkoutu. Możliwa jest opcja wygaszenia opcji "Kup teraz, zapłać później" tak, aby wyeliminować możliwość wporowadzenia jakichkolwiek zmian oraz modyfikacji produktów znajdujących się w koszyku.

4.1.1. Komunikaty Twisto

Reason ID	Krótki opis	Odpowiedź
0	Domyślny powód odrzucenia	Bardzo nam przykro, ale ten zakup z Twisto jest niemożliwy. Prosimy wybrać inną formę płatności.
1	Błędny adres	Mimo naszych starań nie udało nam się odnaleźć podanego przez Ciebie adresu. Sprawdź czy na pewno jest poprawny, a jeśli to nie pomoże - spróbuj skorzystać z innego adresu, np. miejsca pracy.
2	Nieopłacona faktura	Limit Twisto na ten zakup został przekroczony. Nie martw się, jeśli zapłacisz za swoje poprzednie zamówienia, Twój zakup na pewno się uda!
3	Przekroczony limit transakcji	Limit Twisto na ten zakup został przekroczony. Nie martw się, jeśli kwota Twojego zamówienia wyniesie poniżej 450PLN, Twój zakup na pewno się uda!
5	Błędna nazwa	Prosimy wpisać swoje pełne imię i nazwisko w adresie zamówienia.
6	Brak konta w Twisto	Ten zakup jest dostępny tylko dla zarejestrowanych klientów Twisto. Załóż konto na www.twisto.pl
100	Przekroczenie limitu konta w Twisto	Ten zakup przekracza poziom dostępnego limitu na Twoim Koncie Twisto. Nie martw się, jeśli kwota Twojego zamówienia wyniesie poniżej 450 PLN, Twój zakup na pewno się uda!
101	Nieopłacona faktura	Ten zakup przekracza poziom dostępnego limitu na Twoim Koncie Twisto. Nie martw się, jeśli zapłacisz za swoje poprzednie zamówienia, Twój zakup na pewno się uda!
102	Wymagane logowanie do konta	Okno logowania do konta Twisto zostało zamknięte. Z powodów bezpieczeństwa prosimy o ponownie zalogowanie się przed przejściem dalej.

4.1.1.1. Przykład negatywnej odpowiedzi od Twisto:

```
{
      "transaction_id": "3z9h19uqrpd4jtmfj4fdod2t",
      "status": "rejected",
      "reason_id":3,
      "reason": "Niestety, Twoje zakupy przekraczają limit Twisto. Z przyjemnością zapłacimy za nie,
jeżeli ich kwota nie przekroczy 450 zł.",
      "reason_params": {
            "limit": "450 zł"
      }
}
```

Decyzja negatywna jest spowodowana przekroczeniem limutu transakcji który wynosi 450zł

4.2. Pobranie kluczy

> Pobrane klucze Twisto są stałe. Należy używać tych samych kluczy do każdej transakcji.

Klucze niezbędne do przeprowadzenia poprawnej transakcji za pomocą Twisto należy pobrać wysyłając żądanie metodą GET na adres:

```
https://paywall.imoje.pl/check/methods
```

nagłówki które muszą się znaleźć w zapytaniu:

```
Content-Type: application/json
X-Imoje-Signature: merchantid={merchantid};serviceid={serviceid};signature={signature};alg=sha256
```

gdzie:

- `merchantid` - identyfikator klienta,
- `serviceid` - identyfikator sklepu,
- `signature` - zahashowany metodą `sha256` klucz sklepu.

Przykład wygenerowanej sygnatury:

```
hash('sha256',{servicekey});
```

gdzie:

- `servicekey` - klucz sklepu.

Zawartość przykładowej odpowiedzi:

```
{
   "status": "ok",
   "data": {
      "twisto": {
         "enable": true,
         "pk": "live_pk_ayog9acnonq70j1xcn63n5ttmvsfur54sd9lcelez9feycuub3",
         "sk": "live_sk_7fd1dbbfc0c63995f71960ec6cxzhf14e6be526637aad581ff82b16170b049a5",
         "timeout": 10000
      }
   }
}
```

gdzie:

- `enable` to true jeżeli metoda płatności Twisto jest włączona dla tego sklepu, false jeżeli nie,
- `pk` to klucz publiczny Twisto,
- `sk` to klucz prywatny Twisto,
- `timeout` to przykładowy czas oczekiwania na odpowiedź Twisto. Po jego upłynięciu należy przeprowadzić płatność z wartością parametru `twistoData`: `{'details': 'timeout'}`.

4.3. Biblioteka PHP

Biblioteka PHP wspomagająca proces przygotowania danych płatności znajdują się na stronie internetowej:

```
https://github.com/TwistoPayments/Twisto.php
```

4.4. Struktura danych, które są przekazywane do systemu Twisto

4.4.1. Przygotowanie danych z użyciem biblioteki PHP

> Ważne jest, aby ustawić poprawny adres api `$twisto->setApiUrl('https://api.twisto.pl/v2/')`

```
define('TWISTO_PUBLIC_KEY', 'live_pk_v22d67vm1fdec1xp3loxsban99gnpa022xes7u56kq6ahohz1t');
define('TWISTO_SECRET_KEY',
'live_sk_8d4442659363x7s28c1f5d58da43c38c7b6ed886663cc87c41046cd729fce7c7');

$twisto = new Twisto\Twisto();

$twisto->setApiUrl('https://api.twisto.pl/v2/');
$twisto->setPublicKey(TWISTO_PUBLIC_KEY);
$twisto->setSecretKey(TWISTO_SECRET_KEY);

$customer = new Twisto\Customer('johndoe@example.com', 'John Doe');

$order_items = array(
```

```
    new Twisto\Item(
        Twisto\Item::TYPE_DEFAULT, // typ
        'Coca Cola 1 litr',       // nazwa
        530,                  // product_id (product ID - musi być unikalny dla zamówienia)
        6,                    // ilość
        156,                  // price_vat (cena brutto przedmiotu pomnożona przez ilość)
        23                    // procent vat
    ),
    new Twisto\Item(
        Twisto\Item::TYPE_DEFAULT, // typ
        'Rowerek dziecięcy',      // nazwa
        942,                  // product_id (product ID - musi być unikalny dla zamówienia)
        1,                    // ilosc
        285.31,               // price_vat (cena brutto przedmiotu pomnożona przez ilość)
        23                    // procent vat
    ),
    new Twisto\Item(
        Twisto\Item::TYPE_SHIPMENT, // typ
        'Kurier',             // nazwa
        'shipment',           // product_id (product ID - musi być unikalny dla zamówienia)
        1,                    // ilość
        119,                  // price_vat (cena brutto przedmiotu pomnożona przez ilość)
        23                    // procent vat
    ),
    /*
     * Przedmiot jako płatność (Twisto) jest również wymagana
    */
    new Twisto\Item(
        Twisto\Item::TYPE_PAYMENT, // typ
        'Twisto',             // nazwa
        'payment',            // product_id (product ID - musi być unikalny dla zamówienia)
        1,                    // ilość
        0,                    // price_vat (cena brutto przedmiotu pomnożona przez ilość)
        23                    // procent vat
    ),
    new Twisto\Item(
        Twisto\Item::TYPE_ROUND, // typ
        'Zaokraglenie',       // nazwa
        'round',              // product_id (product ID - musi być unikalny dla zamówienia)
        1,                    // ilość
        -0.31,                // price_vat (cena brutto przedmiotu pomnożona przez ilość)
        0                     // procent vat
    ),
);

$order = new Twisto\Order(
    new DateTime(),    // date_created
    new Twisto\Address( // billing_address
        'John Doe',
        'Polna 5',
        'Warszawa',
        '00001',
        'PL',
        '+48500500500'),
    new Twisto\Address( // delivery_address
```

```
        'John Doe',
        'Polna 2',
        'Warszawa',
        '00001',
        'PL',
        '+48500500500'),
    560,            // total_price_vat
    $order_items      // items
);

$previous_orders = array(
    new Twisto\Order(
        new DateTime("2012-07-08 11:14:15.638276"),
        $order->billing_address,
        $order->delivery_address,
        $order->total_price_vat,
        $order_items
    )
);

// utworzenie danych które zostaną wysłane do Twisto przez javascript
$payload = $twisto->getCheckPayload($customer, $order, $previous_orders);
```

4.4.2. Przygotowanie danych bez biblioteki PHP

Struktura zamówienia:

```
{
    "random_nonce": {random_nonce},
    "customer": {customer},
    "order": {order},
    "previous_orders": {previous_orders}
}
```

gdzie:

- `random_nonce` - unikalny, złożony z losowych alfanumerycznych znaków identyfikator,
- `customer` - sekcja zgodna z opisem w punkcie 4.4.3.,
- `order` - sekcja zgodna z opisem w punkcie 4.4.3.,
- `previous_orders` - tablica wcześniejszych zamówień, gdzie każde pojedyńcze zamówienie ma strukturę opisaną w sekcji 4.4.3. (Order), nie jest wymagane.

> Dane te powinny być w formacie `json` oraz należy je zaszyfrować zgodnie z opisem w punkcie 4.5. Przykład przygotowania danych wraz z szyfrowaniem został przedstawiony w punkcie 4.6.

4.4.3. Struktura parametrów potrzebnych do złożenia zamówienia w Twisto

4.4.3.1. Customer

Nazwa	Wymagane	Typ	Maksymalna długość	Opis	Przykład
email	**Tak**	String	254	Adres email płatnika	"johndoe@example.com"
name	Nie	String	255	Nazwa	"Jan Kowalski"
facebook_id	Nie	String	50	Identyfikator Facebook	"123456789"
company_id	Nie	String	15	NIP	"1234567890"
vat_id	Nie	String	15	REGON	"123456789"

4.4.3.2. Order

Nazwa	Wymagane	Typ	Maksymalna długość	Opis	Przykład
date_created	**Tak**	String		Data i czas generowania zamówienia - ISO 8601	"2013-07-22T14:57:18+00:00"
billing_address	**Tak**	*Address*		Adres rozliczeniowy	
delivery_address	**Tak**	*Address*		Adres dostawy	
total_price_vat	**Tak**	Number	8.2	Koszt przedmiotu z uwzględnieniem podatku VAT	1432.23
items	**Tak**	*Item[]*		Zawartość koszyka	

4.4.3.3. Address

Nazwa	Wymagane	Typ	Maksymalna długość	Opis	Przykład
name	**Tak**	String	100	Imię i nazwisko	"Jan Kowalski"
street	**Tak**	String	100	Ulica	"Warszawska 1/5"
city	**Tak**	String	100	Miasto	"Warszawa"
zipcode	**Tak**	String	5	Kod pocztowy	"00001"
phone_number	**Tak**	String	20	Numer telefonu	
country	**Tak**	String	2	Kod kraju ISO 3166-1 alpha-2	"PL"
type	**Tak**	Number			1

4.4.3.4. Item

Nazwa	Wymagane	Typ	Maksymalna długość	Opis	Przykład
type	**Tak**	Number		Typ produktu: 0 - produkt, 1 - wysyłka, 2 - płatność, 4 - zniżka, 32 - zaokrąglenie	0
name	**Tak**	String	255	Nazwa przedmiotu	"Koszulka"
product_id	**Tak**	String	255	ID produktu (musi być unikalny dla zamówienia)	"1000"
quantity	**Tak**	Number		Ilość	2
price_vat	**Tak**	Number		Koszt przedmiotu z uwzględnieniem podatku VAT	406.07
vat	**Tak**	Number		Procent podatku VAT	23
ean_code	Nie	String	13	Kod EAN	
isbn_code	Nie	String	13	Kod ISBN	
issn_code	Nie	String	8	Kod ISSN	
heureka_category	Nie	Number		Kod kategorii w systemie Heureka	

4.5. Szyfrowanie danych

4.5.1. AES + HMAC

Zawartość zapytania należy zaszyfrować za pomocą AES-128-CBC i podpisać metodą HMAC-SHA256. Szyfrowanie powinno przebiegać w następujący sposób:

1. wygenerować kryptograficznie losowy wektor `iv`,
2. usunąć pierwsze 8 znaków z prywatnego klucza. Następnie przekonwertować ciąg danych z kodu szesnastkowego na kod binarny. Z uzyskanego ciągu znaków, należy użyć pierwszych szesnaście znaków jako `klucz` dla szyfru. Resztę danych użyć jako `salt`,
3. użyć klucza oraz wektora `iv` aby uzyskać `cipher`,
4. przekonwertować dane do UTF-8 i skompresować otrzymany ciąg znaków używając biblioteki `zlib`,
5. dodać długość łańcucha znaków jako `unsigned long int` w kolejności bajtów sieciowych i zastosować dopełnienie do wynikowego łańcucha,
6. zaszyfrować tekst i razem z wektorami `iv` i `digest` przekonwertować do Base64.

Przykład:

```
// inicjalizacja zmiennej
$iv = openssl_random_pseudo_bytes(16);

// klucz prywatny Twisto
$secret_key = 'live_sk_a3daex196gc40c4f8oa8a8as9ccx218415aa13fcsdso2ad9h0jaeed6490lld20';

// pobranie klucza i soli
$bin_key = pack("H*", substr($secret_key, 8));
$aes_key = substr($bin_key, 0, 16);
$salt = substr($bin_key, 16, 16);

// kompresja danych
$gz_data = gzcompress($data, 9);
$data = pack("N", strlen($gz_data)) . $gz_data;

// sprawdzenie sumy
$digest = hash_hmac('sha256', $data . $iv, $salt, true);

// szyfrowanie AES
$encrypted = openssl_encrypt($data, 'aes-128-cbc', $aes_key, true, $iv);
$result = base64_encode($iv . $digest . $encrypted);
```

gdzie:

- `$data` - dane zamówienia w formacie `json`.

4.5.2. SecretBox

Istnieje drugi sposób na zaszyfrowanie danych z wykorzystaniem biblioteki `Sodium`:

1. usunąć pierwsze 8 znaków z prywatnego klucza. Następnie przekonwertować ciąg danych z kodu szesnastkowego na kod binarny. Z uzyskanego ciągu znaków, należy użyć pierwszych szesnaście znaków jako `klucz` dla szyfru,
2. stworzyć losowy, alfanumerczyny ciąg znaków dalej nazywany identyfikatorem używając bezpieczenego generatora. Identyfikator dla każdego zamówienia musi być inny. Generator musi być bezpieczny do użytku kryptograficznego, więc zdecydowanie zalecamy użycie generatora dostarczonego przez bibliotekę `Sodium`,
3. wykonać szyfrowanie za pomocą biblioteki `Sodium`.

4.5.3. Przykład z użyciem biblioteki PHP

W przypadku wykorzystania biblioteki PHP szyfrowanie danych odbywa się w ramach funkcji `getCheckPayload` w klasie `Twisto`.

4.6. Kompletny przykład przygotowania danych wraz z szyfrowaniem bez biblioteki PHP

```
$random_nonce = uniqid('', true);
$customer = [
      "email" => "johndoe@example.com",
      "name"  => "John Doe",
];
$order = [
      "date_created" => "2018-08-21T16:40:18+00:00",
      "billing_address" => [
            "name" => "John Doe",
            "street" => "Polna 5",
            "city" => "Warszawa",
            "zipcode" => "00001",
            "country" => "PL",
            "phone_number" => "500500500",
      ],
      "delivery_address" => [
            "name"  => "John Doe",
            "street" => "Polna 2",
            "city" => "Warszawa",
            "zipcode" => "00001",
            "country" => "PL",
            "phone_number" => "500500500",
      ],
      "total_price_vat" => "100",
      "items" => [
            [
                  "type" => "0",
                  "name" => "Koszulka",
                  "product_id" => "123",
                  "quantity" => "1",
```

```
                "price_vat" => "100",
                "vat" => "23",
            ],
        ],
];

$data = json_encode([
    "random_nonce" => $random_nonce,
    "customer" => $customer,
    "order" => $order,
]);

$gz_data = gzcompress($data, 9);
$data = pack("N", strlen($gz_data)) . $gz_data;

$secret_key = 'live_sk_a3daex196gc40c4f8oa8a8as9ccx218415aa13fcsdso2ad9h0jaeed6490lld20';

$bin_key = pack("H*", substr($secret_key, 8));
$aes_key = substr($bin_key, 0, 16);
$salt = substr($bin_key, 16, 16);
$iv = openssl_random_pseudo_bytes(16);
$encrypted = openssl_encrypt($data, 'aes-128-cbc', $aes_key, true, $iv);
$digest = hash_hmac('sha256', $data . $iv, $salt, true);

$data = base64_encode($iv . $digest . $encrypted);
```

4.7. Dodatkowe informacje wdrożeniowe

Jak tylko zakończysz wdrożenie Twisto skontaktuj się z nami - przetestujemy implementację od strony kupującego aby mieć pewność, że wszystko działa bez zarzutu.

Z naszego doświadczenia wynika, że kilka punktów może przysporzyć problemy nawet doświadczonym developerom, poniżej lista przez którą warto przejść przed zakończeniem implementacji:

1. Sprawdź czy na pewno wysyłane są do nas wszystkie obligatoryjne pola, a ich format jest prawidłowy - jeśli tak nie jest, warto wymusić podanie odpowiednich danych walidacją poszczególnych pól formularzy na stronie.
2. Numery telefonu powinny zawsze być prawidłowe, odnosić się do telefonów komórkowych i być przesyłane w formacie międzynarodowym (np. +48 60760706)
3. Jeśli zamówienie zawiera kilka identycznych przedmiotów, parametr "`total price vat`" powinien przyjąć wartość stanowiącą iloczyn ceny jednostkowej i liczby sztuk danego produktu.
4. Parametr "`Product ID`" musi być unikalny - zarówno w aktualnym zamówieniu, jak i w historii zamówień. Często różne rozmiary czy warianty powodują zdublowanie tego parametru - w takim wypadku warto połączyć identyfikator produktu z bazy danych z polem określającym jego atrybut, tak aby mieć pewność, że w efekcie pole będzie unikalne.
5. Jeśli próba utworzenia zamówienia zakończy się niepowodzeniem (zwrócimy komunikat błędu) - nie zapomnij anulować zamówienia.
6. Elementy koszyka obniżające kwotę zamówienia (rabaty czy kupony rabatowe) powinny mieć w parametrze "`type`" wartość 4, a kwota powinna być poprzedzona minusem.

5. Płatność oneclick

> Metoda ta jest dostępna tylko i wyłącznie dla akceptantów którzy posiadają certyfikat PCI-DSS. Obciążanie karty może odbywać się jedynie na podstawie wyraźnych zgód posiadacza instrumentu płatniczego. Więcej szczegółów można znaleźć u opiekuna handlowego.

Metoda integracji oneclick pozwala na zintegrowanie płatności kartą płatniczą z poziomu sklepu. W takim przypadku nie jest wymagane przekierowanie na bramkę płatności. Aby zastosować tę metodę należy osadzić skrypt JavaScript:

```
<script
   src="https://paywall.imoje.pl/js/widget.min.js"
   id="imoje-widget__script"

   data-merchant-id="6yt3gjtm9p1odfgx8491"
   data-service-id="63f574ed-d90d-4abe-9cs1-39117584a7b"
   data-amount="100"
   data-currency="PLN"
   data-order-id="123"
   data-customer-id="123"
   data-customer-first-name="John"
   data-customer-last-name="Doe"
   data-customer-email="johndoe@domain.com"
   data-
signature="65f6b5564d4810b045bb3b49074e4f2cab673c3917ac0b6daa18f9e4762fd7b9;sha256">
</script>
```

gdzie

> Poniższe dwa parametry `src,id` należy nie uwzględniać przy wyliczaniu sygnatury.
> Widżet dla **środowiska testowego (sandbox)** znajduje się pod adresem:
> `https://sandbox.paywall.imoje.pl/js/widget.min.js`

- `src` - adres URL do skryptu `widget.min.js`,
- `id` - ID skryptu. Zawsze musi mieć wartość `imoje-widget__script`.

Parametry wymagane:

- `data-merchant-id` - identyfikator klienta,
- `data-service-id` - identyfikator sklepu klienta,
- `data-amount` - kwota transakcji podana w groszach,
- `data-currency` - waluta,
- `data-order-id` - ID zamówienia, dopuszczalne znaki: A-Za-z0-9_-`#&'",./\ oraz białe znaki i znaki z zakresu UNICODE 00C0 - 02C0 (m.in. polskie znaki diakrytyczne),
- `data-customer-id` - ID płatnika, dopuszczalne znaki: A-Za-z0-9_-
- `data-customer-first-name` - imię osoby składającej zamówienie,
- `data-customer-last-name` - nazwisko osoby składającej zamówienie,
- `data-customer-email` - adres e-mail osoby składającej zamówienie,

- `data-signature` - wyliczona sygnatura.

Parametry opcjonalne:

- `data-customer-phone` - numer telefonu osoby składającej zamówienie
- `data-order-description` - tytuł transakcji, dopuszczalne znaki: A-Za-z0-9_-`#&'",./\ oraz białe znaki i znaki z zakresu UNICODE 00C0 - 02C0 (m.in. polskie znaki diakrytyczne),
- `data-url-success` - adres URL, na który użytkownik będzie przekierowany po pomyślnie zakończonej transakcji,
- `data-url-failure` - adres URL, na który użytkownik będzie przekierowany po błędnie dokonanej transakcji,
- `data-url-return` - adres URL, na który użytkownik będzie przekierowany po dokonaniu transakcji,
- `data-url-cancel` - adres URL, na który użytkownik będzie przekierowany po naciśnięciu przycisku `Anuluj`,
- `data-recurring` - oznaczenie płatności jako płatność rekurencyjna, dopuszczalne wartości: `true` lub `false`.

Parametry konfiguracyjne - **nie należy ich uwzględniać przy wyliczaniu sygnatury**:

- `data-inline` - jeżeli jest ustawione na `true` to istnieje możliwość aby iframe był osadzony w elemencie o identyfikatorze `imoje-widget__wrapper` (element o takim identyfikatorze musi istnieć). Jeżeli tego parametru nie ma lub jest ustawiony na `false` wtedy iframe będzie wyświetlony na cały ekran,
- `data-element-id` - wskazanie identyfikatora elementu który ma być interaktywny z widżetem, domyślnie jest `false`,
- `data-element-event` - wskazanie jaki event ma rozpocząć interakcję z widżetem, domyślnie jest `click`.

5.1. Wyliczanie sygnatury

Nazwy parametrów do wyliczenia sygnatury należy zamienić zgodnie z przykładem:

- parametr `data-order-id` na `orderId`.

Sygnaturę należy wyliczyć w taki sam sposób jak jest to opisane w punkcie `1.1.`.

5.2. Otrzymanie profilu klienta

Dla każdej płatności metodą oneclick w notyfikacji jest wysyłany profil instrumentu płatniczego. Profil ten jest identyfikowany za pomocą parametru `data-customer-id`. Należy pamiętać, że każdy instrument płatniczy powinien mieć swoją unikalną wartość tego parametru. Notyfikacja z uwzględnionym profilem instrumentu płatniczego (np. karty płatniczej) ma strukturę notyfikacji przy zwykłej płatności wzbogaconą o dodatkowe pole `statusCode` oraz sekcja `profile` o strukturze:


```
{
  "transaction": {
    "id": "57a105fe-af75-41eb-9195-6d0bf9183c7e",
    "status": "rejected",
    "source": "api",
    "created": 1549621088,
    "modified": 1549621088,
    "notificationUrl": "https://notificationurl.com/ipn",
    "serviceId": "6879ff96-3efa-4ea2-b193-59401da40f18",
    "amount": 1000,
    "currency": "PLN",
    "orderId": "2171ef23-828e-47e1-a3f6-80fdd4030863",
    "paymentMethod": "card",
    "paymentMethodCode": "oneclick",
    "statusCode": "PAYMENT_ERROR",
    "paymentProfile": {
      "id": "d6a5bd6c-8e9c-496e-beb2-f0ca08215645",
      "merchantMid": "6yt3gjtm9p7b8h9xsdqz",
      "merchantCustomerId": "39ac1087-e632-41ff-acb8-8d661068a9d5",
      "firstName": "John",
      "lastName": "Doe",
      "maskedNumber": "****1791",
      "month": "10",
      "year": "2020",
      "organization": "MASTERCARD",
      "isActive": 1,
      "profile": "ONE_CLICK"
    }
  }
}
```

gdzie:

- `statusCode` - kod odpowiedzi od prowidera. Pełna lista odpowiedzi znajduje się w punkcie `5.2.1.`,
- obiekt `paymentProfile`:
    - `id` - identyfikator profilu,
    - `merchantMid` - identyfikator klienta,
    - `merchantCustomerId` - identyfikator płatnika, dopuszczalne znaki: A-Za-z0-9_-
    - `firstName` - imię posiadacza instrumentu płatniczego na który jest zarejestrowany profil,
    - `lastName` - nazwisko posiadacza instrumentu płatniczego na który jest zarejestrowany profil,
    - `maskedNumber` - cztery gwiazdki oraz ostatnie cztery cyfry instrumentu płatniczego,
    - `month` - data ważności karty: miesiąc,
    - `year` - data ważności karty: rok,
    - `organization` - organizacja płatnicza która wydała zarejestrowaną kartę,
    - `isActive` - aktywność profilu: 1 - aktywna, 0 - nieaktywna,
    - `profile` - rodzaj profilu.

W przypadku próby obciążenia nieaktywnego profilu odpowiedź będzie wyglądać:


```
{
  "apiErrorResponse": {
    "code": "TRX-ERROR-120301",
    "message": "Payment profile inactive.",
    "instance": {
      "serviceId": "6879ff96-3efa-4ea2-b193-59401da40f18",
      "paymentProfileId": "d6a5bd6c-8e9c-496e-beb2-f0ca08215645",
      "amount": 100,
      "currency": "PLN",
      "orderId": "2171ef23-828e-47e1-a3f6-80fdd4030863"
    },
    "errors": []
  }
}
```


5.2.1. Kody odpowiedzi prowidera

Kod odpowiedzi	Opis kodu odpowiedzi
`AUTHORIZED`	Płatność została zaakceptowana
`PAYMENT_ERROR`	Błąd płatności
`CARD_EXPIRED`	Data ważności instrumentu płatniczego wygasła

6. Multiwypłaty

Opcja dostępna w przypadku włączonej funkcji multiwypłaty. Wykonujemy transakcje zgodnie z opisem punktu 1. z jednym dodatkowym parametrem:

- `multipayout` - tablic, której każdy element powinien zawierać wszystkie poniższe pola:
  - `ban` - numer konta bankowego,
  - `amount` - kwota transakcji podana w groszach,
  - `label` - nazwa odbiorcy (max 35 znaków),

Każda wypłata zawarta w formularzu poniżej powinna zawierać kolejne indexy numerowane od 0.

Przykładowa zawartość formularza wysyłana metodą POST:

```
<input type="hidden" value="63f574ed-d90d-4abe-9cs1-39117584a7b7" name="serviceId">
<input type="hidden" value="6yt3gjtm9p1odfgx8491" name="merchantId">
<input type="hidden" value="300" name="amount">
<input type="hidden" value="PLN" name="currency">
<input type="hidden" value="123" name="orderId">
<input type="hidden" value="Example transaction" name="orderDescription">
<input type="hidden" value="John" name="customerFirstName">
<input type="hidden" value="Doe" name="customerLastName">
<input type="hidden" value="johndoe@domain.com" name="customerEmail">
<input type="hidden" value="501501501" name="customerPhone">
<input type="hidden" value="https://your-shop.com/success" name="urlSuccess">
<input type="hidden" value="https://your-shop.com/failure" name="urlFailure">
<input type="hidden" value="https://your-shop.com/return" name="urlReturn">
<input type="hidden" value="05105000995129330000123456" name="simp">
<input type="hidden" value="card,pbl" name="visibleMethod">
<input type="hidden" value='{"transaction_id":"s2kjtgl123d5261e5151s426","status":"accepted"}'
name="twistoData">
<input type="hidden" value="55105000026800208114085773" name="multipayout[0][ban]">
<input type="hidden" value="100" name="multipayout[0][amount]">
<input type="hidden" value="Nazwa firmy 0" name="multipayout[0][label]">
<input type="hidden" value="81105000029779266659470337" name="multipayout[1][ban]">
<input type="hidden" value="200" name="multipayout[1][amount]">
<input type="hidden" value="Nazwa firmy 1" name="multipayout[1][label]">
<input type="hidden"
value="1a466af99a18c4691576bbbf5b935e2ac082285ae28a88f8686ac3317836a6f5;sha256"
name="signature">
```

6.1. Wyliczanie sygnatury

Sygnature obliczamy zgodnie z punktem 1.1. z uwzględnieniem dodatkowych parametrów

```
$fields = [
    'merchantId' => '6yt3gjtm9p1odfgx8491',
    'serviceId' => '63f574ed-d90d-4abe-9cs1-39117584a7b',
    'amount' => '300',
    'currency' => 'PLN',
    'orderId' => '123',
    'orderDescription' => 'Example transaction',
    'customerFirstName' => 'John',
    'customerLastName' => 'Doe',
    'customerEmail' => 'johndoe@domain.com',
    'customerPhone' => '501501501',
    'urlSuccess' => 'https://your-shop.com/success',
    'urlFailure' => 'https://your-shop.com/failure',
    'urlReturn' => 'https://your-shop.com/return',
    'twistoData' => '{"transaction_id":"s2kjtgl123d5261e5151s426","status":"accepted"}',
    'multipayout' => [
        [
        'ban' => '55105000026800208114085773',
        'amount' => '100',
        'label' => 'Nazwa firmy 0',
        ],
        [
        'ban' => '81105000029779266659470337',
        'amount' => '200',
        'label' => 'Nazwa firmy 1',
        ],
    ],
];
```

7. Minimalne wartości kwot transakcji, zwrotów

Dla każdej metody płatności obowiązują następujące limity:

Metoda płatności	Minimalna kwota płatności (PLN)
Przelewy online Pay-By-Link	1
Płatność za pomocą BLIK	0.10
Płatność kartą	0.01
Płatność oneclick	0.01

Poniżej tego progu dana metoda płatności nie jest dostępna.

8. Dane kontaktowe, wsparcie techniczne

Adres e-mail: kontakt.tech@imoje.pl
Telefon: +48 32 319 35 70
WWW: https://www.imoje.pl