Technical Documentation

Overview

This chapter provides the instruction of the integration with the solution and with it's methods.

Prior to using this solution the Merchant have to proceed onboarding process. To create account please contact with support. The PayTool solution provides three integrations models: **Web SDK, API** and **One Time Payment API.** Each of the integration models is independent of each other. This means that if the Customer chooses the Web SDK integration model, he no longer needs the API model.

1. **Web SDK** - integration with the PayTool’s payment process through the JavaScript library
2. **API** - integration with the PayTool’s payment process using server-server connection
3. **One Time Payment API** - this is alternative API integration model - recommended for Merchant which are collecting sensitive transaction data by their own.

> Verestro recommends to use SDK integration model.

> One time payment API is a separate service and its integration is intended for Merchant who themselves collect/store sensitive data such as card number.

Integration

SDK

This section describes how to integrate the solution using the SDK provided by Verestro.

@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F

```
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
participant "Merchant App (Browser)" as browser
participant "Paytool Frontend" as pfront
participant "Paytool SDK" as psdk
participant "Paytool Backend" as pback
participant "Merchant Server" as custback
participant "Acquirer" as acq
note right of browser: User chooses "Pay with Merchant Paytool" (1)
browser->pfront
pfront->psdk: Begin transaction (2)
psdk->pback: Transaction preinitialize /preinit (3)
pback->pback: Store transaction session data (4)
pback->psdk: OK - returns transactionId (5)
psdk->pfront: Redirect with transactionId (6)
pfront->pback: Get transaction (from point 3) (7)
pback->pfront: Returns transaction details (8)
pfront->pfront: User filling data (CN, CVC, EXP) (9)
pfront->pfront: Encrypt data (from point 9) (10)
pfront->pback: Perform transaction /transaction/TRX_ID + enc body (11)
pback->acq: Transaction with 3DS (12)
note left of acq: At this point 3D Secure process takes place
acq->pback: Response (13)
pback->pfront: Transaction Result (14)
pback->custback: Send postback to provided URL (15) (Optional - configurable)
pfront->browser: redirect User (success/failure) (16)
custback->pback: Get transaction status (17) (optional but hardly recommended in case when postbacks are disabled)
pback->custback: Return transaction status (18)
custback->browser: Provide transaction result (19)
@enduml
```

> The merchant-paytool.js is a JavaScript-based client-side SDK for Merchant PayTool.

The SDK adds MerchantPayTool class to the global JavaScript scope which can be further instantiated to start PayTool's payment process. Alternatively, it also defines a custom element called merchant-paytool for a more straightforward, plug-and-play solution.

An optional step is to send a transaction status notification to the Merchant Server. Details of the notification are described in the section Postbacks.

In addition to the web SDK itself, the solution provides a GET API method Get transaction details that allows the Merchant to get the transaction status using "*transactionId*" parameter. This method is optional but we hardly recommend to use it.

API Reference

Initialization Add the following script to your website:

- Test environment

```
<script type="module" src="https://merchant-paytool.verestro.dev/merchant-paytool.js"></script>
```

- Production environment

```
<script type="module" src="https://merchant-paytool.verestro.com/merchant-paytool.js"></script>
```

> The type="module" attribute is currently required, because the SDK utilizes modern JavaScript code splitting syntax.

SDK Methods

init
(class approach only) Starts PayTool's payment process using the provided data. After successful initialization, resolves a promise and redirects to PayTool's website. Rejects a promise if any error occurs.
Parameters
data initData
Returns
Promise<void>

Interfaces

Data object passed to PayTool's backend API.

initData		
Name	**Type**	**Description**
apiKey	string	Merchant identifier, given during onboarding.
amount	number	Transaction amount in the lowest unit of money, fe. cents for USD.
currency	string	Transaction currency code.
description	string	Short description of transaction.
redirectUrls	object	Optional return url object. If not provided, urls from merchant’s config will be used instead.PayTool might append additional query parameters to the urls, fe. a transaction identifier.
redirectUrls.successUrl	string	The url where users will be redirected after a successful payment.
redirectUrls.failureUrl	string	The url where users will be redirected after a failed payment.

> **Optional:** Merchant can get transaction details using Get transaction details method. This method is an API method. More details can be found below in API chapter.

> **Optional:** Merchant can create HTTP POST endpoint which will accept requests in format JSON. If this option is enabled, the Paytool API will send information about the transaction made to the endpoint/domain provided by the Merchant. More details can be found below in API chapter.

Examples

The SDK offers different ways to initialize a payment. It can do most of the heavy lifting by itself, including UI, but it also exposes a lower-end API to let you customize your UX.

Class approach

Angular

```
{
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: '<button (click)="onClick()">Pay</button>',
})
export class AppComponent {
  payTool = new MerchantPayTool();

  onClick() {
    this.payTool
      .init({
        apiKey: 'YOUR_API_KEY',
        amount: 9999,
        currency: 'CURRENCY_CODE',
        description: 'TRANSACTION_DESCRIPTION',
        redirectUrls: {
          successUrl: 'YOUR_SUCCESS_URL',
          failureUrl: 'YOUR_FAILURE_URL'
        }
      })
      .catch(console.log);
  }
}
}
```

React

```
export const App = () => {
  const payTool = new MerchantPayTool();
  return (
    <button
      onClick={() => {
        payTool
          .init({
            apiKey: 'YOUR_API_KEY',
            amount: 9999,
            currency: 'CURRENCY_CODE',
            description: 'TRANSACTION_DESCRIPTION',
```

```
          redirectUrls: {
            successUrl: 'YOUR_SUCCESS_URL',
            failureUrl: 'YOUR_FAILURE_URL'
          }
        })
        .catch(console.log);
    }}>
    Pay
  </button>
 );
};
```

Plain JavaScript

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <script
   type="module"
   src="https://merchant-paytool.verestro.com/merchant-paytool.js"
  ></script>
 </head>
 <body>
  <button id="pay-btn">Pay</button>
  <script>
   var payButton = document.getElementById('pay-btn');
   payButton.addEventListener('click', function () {
    var payTool = new MerchantPayTool();
    payTool
     .init({
      apiKey: 'YOUR_API_KEY',
      amount: 9999,
      currency: 'CURRENCY_CODE',
      description: 'TRANSACTION_DESCRIPTION',
      redirectUrls: {
       successUrl: 'YOUR_SUCCESS_URL',
       failureUrl: 'YOUR_FAILURE_URL'
      }
     })
```

```
        .catch(console.log);
    });
  </script>
 </body>
</html>
```

Web component approach

This approach focuses on modern solutions to help your integrate with PayTool as fast as possible and keep your code clean, providing a pre-built "Click to Pay" button. The component accepts data as Init Data, similarly to the init method.

Angular

```
import { Component } from '@angular/core';

@Component({
 selector: 'app-root',
 template: '<merchant-paytool [data]="data"></merchant-paytool>',
})
export class AppComponent {
 data = {
  apiKey: 'YOUR_API_KEY',
  amount: 9999,
  currency: 'CURRENCY_CODE',
  description: 'TRANSACTION_DESCRIPTION',
  redirectUrls: {
   successUrl: 'YOUR_SUCCESS_URL',
   failureUrl: 'YOUR_FAILURE_URL',
  },
 };
}
```

React@^18

```
export const App = () => {
 return (
  <merchant-paytool
   ref={el => {
    if (el) {
     el.data = {
```

```
          apiKey: 'YOUR_API_KEY',
          amount: 9999,
          currency: 'CURRENCY_CODE',
          description: 'TRANSACTION_DESCRIPTION',
          redirectUrls: {
            successUrl: 'YOUR_SUCCESS_URL',
            failureUrl: 'YOUR_FAILURE_URL'
          }
        };
      }
    }}
  />
 );
};
```

React@experimental

```
export const App = () => {
 return (
  <merchant-paytool
   data={{
    apiKey: 'YOUR_API_KEY',
    amount: 9999,
    currency: 'CURRENCY_CODE',
    description: 'TRANSACTION_DESCRIPTION',
    redirectUrls: {
     successUrl: 'YOUR_SUCCESS_URL',
     failureUrl: 'YOUR_FAILURE_URL'
    }
   }}
  />
 );
};
```

Plain JavaScript

```
export const App = () => {
 return (
  <merchant-paytool
   data={{
```

```
        apiKey: 'YOUR_API_KEY',
        amount: 9999,
        currency: 'CURRENCY_CODE',
        description: 'TRANSACTION_DESCRIPTION',
        redirectUrls: {
          successUrl: 'YOUR_SUCCESS_URL',
          failureUrl: 'YOUR_FAILURE_URL'
        }
      }}
    />
  );
};
```

API

This section describes how to integrate the solution using the REST API provided by Verestro.

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
participant "Merchant App (Browser)" as browser
participant "Paytool Frontend" as pfront
participant "Paytool Backend" as pback
participant "Merchant Server" as custback
participant "Acquirer" as acq
note right of browser: User chooses "Pay with Merchant Paytool" (1)
```

browser->pback: Transaction preinitialize /preinit (2)
pback->pback: Store transaction session data (3)
pback->browser: OK - returns transactionId (4)
browser->pfront: Redirect with transactionId (5)
pfront->pback: Get transaction (from point 3) (6)
pback->pfront: Returns transaction details (7)
pfront->pfront: User filling data (CN, CVC, EXP) (8)
pfront->pfront: Encrypt data (9)
pfront->pback: Perform transaction /transaction/TRX_ID + enc body (10)
pback->acq: Transaction with 3DS (11)
note left of acq: At this point 3D Secure process takes place
acq->pback: Response (12)
pback->pfront: Transaction Result (13)
pback->custback: Send postback to provided URL (14) (Optional - configurable)
pfront->browser: redirect User (success/failure) (15)
custback->pback: Get transaction status (16) (optional but hardly recommended in case when postbacks are disabled)
pback->custback: Return transaction status (17)
@enduml

The PayTool API allows Merchant to perform the Paytool's payment process. The first step is to initialize the payment using method "PreInitialize". Then, the Merchant Website should redirect the user to the Paytool Website with the transactionId obtained in the first step. Redirect url should build according to schema described in section "Endpoints for PayTool Website". Next step, on the PayTool side the payment process will be executed. When it is finished, the user will be redirected to the address specified by the Merchant in the method "PreInitialize".

An optional step is to send a transaction status notification to the Merchant Server. Details of the notification are described in the section "Postbacks".

Additionally, the PayTool API provides a method that allows the Merchant to "Get transaction details" using "*transactionId*" parameter.

@swagger="https://files.verestro.dev/merchant-public/paytool-api-integration.yaml"

One Time Payment API

This chapter provides information about the technical requirements of the One Time Payment API and the scructure of the objects which are need to be integrated with.

@swagger="https://files.verestro.dev/merchant-public/one_time_payment_1_0_4.yaml"

JWE

OneTime Payment API supports encryption of requests as standard JSON Web Encryption (JWE) per RFC 7516. Recommended to read the JWE standard: RFC 7516.

JWE represents encrypted content using JSON data structures and Base64 encoding. The representation consists of three parts: a JWE Header, a encrypted payload, and a signature. The three parts are serialized to UTF-8 bytes, then encoded using base64url encoding. The JWE's header, payload, and signature are concatenated with periods (.).

JWE typically takes the following form:

```
{Base64 encoded header}.{Base64 encoded payload}.{Base64 encoded signature}
```

JWE header

Type	Value	Constraints	Description
alg	RSA-OAEP-256	Required	Identifies the cryptographic algorithm used to secure the JWE Encrypted Key. Supported algorithms: RSA-OAEP-256, RSA-OAEP-384, RSA-OAEP-512. Recommend value: **RSA-OAEP-256**.
enc	A256GCM	Required	Identifies the cryptographic algorithm used to secure the payload. Supported algorithms:A128GCM, A192GCM, A256GCM, A128CBC-HS256, A192CBC-HS384, A256CBC-HS512. Recommend value: **A256GCM**.
typ	JOSE	Optional	Identifies the type of encrypted payload. Recommend value: **JOSE**.
iat	1637929226	Optional	Identifies the time of generation of the JWT token. Supported date format: unix time in UTC. In the case of iat send, the validity of JWE is validated. Recommend send the header due to the increase in the security level.

Type	Value	Constraints	Description
kid	5638742a57	Optional	Identifies the public key of use to encrypt payload. Supported format: SHA-1 value of the public key. In the case of kid send, the validity of public key is validated, so we can inform the client that the public key has changed.

Payload encryption

Every encrypted request should include JWE token. The jwe token should be passed in the field indicated in the method. To prepare the encrypted payload.The steps may differ depending on the libraries used:

1. Get the public key using the method: Get Public Key. The public key is encoded with Base64.
2. Decode the public key.
3. Then create a correct object to be encrypted.
4. Encrypt the created object with the public key.
5. Create JWE header compatible with: JWE Header.
6. Make a request on the method that supports JWE. Set the JWE token in the field indicated in the method.

Encrypted Data

The encrypted payload consists of the fields presented below in JSON format:

```
{
   "cardNumber": "5325943450735905",
   "cvc": "123",
   "expDate": "12/24",
   "firstName": "firstName",
   "lastName": "lastName",
   "phone": "48123123123",
   "email": "test@verestro.com"
}
```

Enpoints for One Time Payment API

One Time Payment API is available on two environments: Beta (for tests) and Production.

Environemt	Base url
Beta	https://merchant-beta.upaidtest.pl/champion/
Production	https://merchant.upaid.pl/champion/

Methods in One Time Payment API

Methods created in One Time Payment API are created in accordance with the principles of the REST methodology. Through these methods Merchant is able to make a payment. The data transferred during the execution of `oneTimePaymentWithout3ds` and `initOneTimePayment3ds` requests accept encrypted card data and user data. The data that should be encrypted is available here. The merchant retrieves the Verestro public key using the `getPublicKey` method and then encrypts the data. Encrypted sensitive data is transferred to Verestro. The methods are secured by Basic Authorization - Merchant's account login details in the Verestro system. The merchant receives this data during the onboarding process.

One time payment without 3ds

POST [base-url]/one-time-payment

This method allows to authorize and process the entire transaction without 3DS authentication. To perform MOTO transaction, pass `transactionType=MOTO`.

Request headers:			
Type	**Value**	**Constraints**	**Description**
Content-Type	application/json	Required	Content type of the request
Authorization	Basic bG9naW46cGFzc3dvcmQ=	Required	Basic Authorization token

Request body:


```
{
  "transactionType": "MOTO",
  "amount": 100,
  "currency": "EUR",
  "description": "Description of transaction: f34e8330-99fe-4ca4-8ee7-3628c989a6e2",
  "encryptedData": "eyJ0eXAiOiJKT1NFIiwiZW5jIjoiQTI1NkdDTSIsImlhdCI6MTY0NDkxMjk5MSwiY...",
  "itemId": "f34e8330-99fe-4ca4-8ee7-3628c989a6e2",
```

```
  "mid": "83e6c996-3a4d-470a-89c3-8952ac3648b7"
}
```

Request fields:			
Type	**Value**	**Constraints**	**Description**
transactionType	String	Not null	Indicates whether the transaction is MOTO or AUTH. Default value AUTH. If MOTO then one of parameters: cvc or addressDetails is required.
amount	Number	Not null	Transaction amount (in pennies)
currency	String	Not empty	Transaction currency
description	String	Not empty	Simple description of transaction
encryptedData	String	Not empty	The field contains encrypted JSON using the JWE standard. Object encryptedData is described in section of schemas.
itemId	String	Not empty	Merchant’s unique id of transaction. Ensures the idempotency of the transaction. The same in the entire 3ds process.

Request fields:			
Type	**Value**	**Constraints**	**Description**
mid	String	Optional	This parameter indicates which account will be used in the process. If mid won’t be passed, then payment will be processed using the default account.This value will be generated in the onboarding process.

Example cURL:

```
$ curl 'https://merchant.upaid.pl/champion/one-time-payment' -i -X POST \
    -H 'Content-Type: application/json' \
    -d '{
  "transactionType": "MOTO",
  "amount": 100,
  "currency": "EUR",
  "description": "Description of transaction: f34e8330-99fe-4ca4-8ee7-3628c989a6e2",
  "encryptedData": "eyJ0eXAiOiJKT1NFIiwiZW5jIjoiQTI1NkdDTSIsImlhdCI6MTY0NDkxMjk5MSwiY...",
  "itemId": "f34e8330-99fe-4ca4-8ee7-3628c989a6e2",
  "mid": "83e6c996-3a4d-470a-89c3-8952ac3648b7"
}'
```

Response body:

HTTP Response OK:

```
HTTP/1.1 200 OK
Content-Length: 209
Content-Type: application/json;charset=UTF-8

{
  "transactionId" : "08ea8e28-0aad-45eb-8368-f15bdadd5eba",
  "itemId" : "f34e8330-99fe-4ca4-8ee7-3628c989a6e2",
  "status" : "DEPOSITED",
  "externalTransactionId" : "49a91f00-26b4-49a2-9c77-ed37646ddf64"
```

```
}
```

Response fields:		
Type	**Value**	**Constraints**
transactionId	String	Identifier of transaction
itemId	String	Merchant’s unique id of transaction. Ensures the idempotency of the transaction. The same in the entire 3ds process.
status	String	Transaction status
externalTransactionId	String	External transaction id

Initialize one time payment with 3ds

POST [base-url]/one-time-payment/3ds/init

The method allows doing initialize payment using ThreeDs 2.0 protocol. As a response there are 3 possible paths:

- **Frictionless flow** - In the response: **threeDsMode = FRICTIONLESS** is present. This response denotes that payment was finished.
- **ThreeDsMethod flow** - In the response: **threeDsMode = THREE_DS_METHOD** is present. This response denotes that ThreeDsMethod flow is required. Browser should make hidden request to ACS URL (parameter in response: threeDsMethodUrl) with passing threeDSMethodData (parameter in response: threeDSMethodData). After that ACS send the notification for endpoint provided in the threeDsMethodNotificationUrl. Based on this notification methodCompletionIndicator parameter should be set in the request for method: /one-time-payment/3ds/continue.
  After executing ThreeDs method flow, make a request for the method: /one-time-payment/3ds/continue.
  Important: This path could be disabled during the onboarding process - Frictnionless and Challenge flow left. Avoiding the ThreeDsMethod flow simplifies the integration - the continue method will be performed internally in the OneTimePayment API.
- **Challenge flow** - In response: **threeDsMode = CHALLENGE** is present. This response denotes that the challenge process is required. In this flow **pageContent** is present. The parameter should be decoded and presented to the user for interaction (challange). After the challenge end, html will redirect the user to URL passed in the termUrl parameter.

The **finalize** request should be performed then.

Request headers:			
Type	**Value**	**Constraints**	**Description**
Content-Type	application/json	Required	Content type of the request
Authorization	Basic bG9naW46cGFzc3dvcmQ=	Required	Basic Authorization token

Request body:


```
{
  "amount": 100,
  "currency": "EUR",
  "description": "Description of transaction: f9814a71-ec77-4998-b354-1fec8bf73a59",
  "encryptedData": "eyJ0eXAiOiJKT1NFIiwiZW5jIjoiQTI1NkdDTSIsImlhdCI6MTY0NDkxMjk5MSwiY...",
  "termUrl": "https://termurl.verestro.com",
  "threeDsMethodNotificationUrl": "https://threedsmethodnotificationurl.verestro.com",
  "requestChallengeIndicator": "NO_PREFERENCE",
  "itemId": "string",
  "mid": "83e6c996-3a4d-470a-89c3-8952ac3648b7",
  "browserDetails": {
    "ipAddress": "77.55.135.220",
    "javaEnabled": true,
    "jsEnabled": true,
    "screenColorDepth": 1,
    "screenHeight": 1500,
    "screenWidth": 1500,
    "timezoneOffset": 60,
    "language": "EN"
  }
}
```


Request fields:			
Type	**Value**	**Constraints**	**Description**

amount	Number	Not null	Transaction amount (in pennies)
currency	String	Not empty	Transaction currency
description	String	Not empty	Simple description of transaction
encryptedData	String	Not empty	The field contains encrypted JSON using the JWE standard. Object encryptedData is described in section of schemas.
termUrl	String	Not empty	Merchant Callback URL - User will be redirected on this URL after a successful pareq/cReq request
threeDsMethodNotificationUrl	String	Optional	URL where ACS will notify about finish 3ds method.

requestChallengeIndicator	String	Optional	Indicates whether a challenge was requested for transaction. Possible values: **NO_PREFERENCE** - no preference for challenge. **CHALLENGE_NOT_REQUESTED** - challenge is not requested. **CHALLENGE_PREFER_BY_REQUESTOR_3DS** - challenge is requested: 3DS Requestor preference. **CHALLENGE_REQUESTED_MANDATE** - challenge is requested: mandate. **RISK_ANALYSIS_ALREADY_PERFORMED** - challenge is not requested: transactional risk analysis already performed. **ONLY_DATA_SHARE** - challenge is not requested: only data is shared. **STRONG_VERIFY_ALREADY_PERFORMED** - challenge is not requested: strong consumer authentication is already performed. **WHITELIST_EXEMPTION** - challenge is not requested: utilise whitelist exemption if no challenge required. **WHITELIST_PROMPT_REQUESTED** - challenge is requested: whitelist prompt requested if challenge required.
itemId	String	Not empty	Merchant’s unique id of transaction. Ensures the idempotency of the transaction. The same in the entire 3ds process.

browserDetails	Object	Required	Browser Details Object
browserDetails.ipAddress	String	Required by some Acquirers	This field contains the IP address of the cardholder’s browser as returned by the HTTP headers.
browserDetails.javaEnabled	Boolean	Required by some Acquirers	This field contains a value representing the ability of the cardholder’s browser to execute Java. Required if jsEnabled=true
browserDetails.jsEnabled	Boolean	Required by some Acquirers	This field contains a value representing the ability of the cardholder’s browser to execute JavaScript
browserDetails.screenColorDepth	Number	Required by some Acquirers	This field contains a value representing the bit depth of the color palette, in bits per pixel, for displaying images. Values accepted: **1** = 1 bit, **4** = 4 bits, **8** = 8 bits, **15** = 15 bits, **16** = 16 bits, **24** = 24 bits, **32** = 32 bits, **48** = 48 bits . Required when jsEnabled=true.
browserDetails.screenHeight	Number	Required by some Acquirers	This field contains the total height of the cardholder’s screen in pixels. Required when jsEnabled=true

browserDetails.screenWidth	Number	Required by some Acquirers	This field contains the total width of the cardholder’s screen in pixels. Required when jsEnabled=true.
browserDetails.timezoneOffset	String	Required by some Acquirers	This field contains the difference between UTC time and the cardholder’s browser local time in minutes. Required if jsEnabled=true
browserDetails.language	String	Required by some Acquirers	This field contains the cardholder’s browser language as defined in IETF BCP 47
mid	String	Optional	This parameter indicates which account will be used in the process. If mid won’t be passed, then payment will be processed using the default account. This value will be generated in the onboarding process.

Example cURL:

```
curl -X 'POST' \
  'https://merchant-beta.upaidtest.pl/champion/one-time-payment/3ds/init' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "amount": 100,
  "currency": "EUR",
  "description": "Description of transaction: f9814a71-ec77-4998-b354-1fec8bf73a59",
  "encryptedData": "eyJ0eXAiOiJKT1NFIiwiZW5jIjoiQTI1NkdDTSIsImlhdCI6MTY0NDkxMjk5MSwiY...",
  "termUrl": "https://termurl.verestro.com",
  "threeDsMethodNotificationUrl": "https://threedsmethodnotificationurl.verestro.com",
  "requestChallengeIndicator": "NO_PREFERENCE",
  "itemId": "string",
  "mid": "83e6c996-3a4d-470a-89c3-8952ac3648b7",
```

```
  "browserDetails": {
    "ipAddress": "77.55.135.220",
    "javaEnabled": true,
    "jsEnabled": true,
    "screenColorDepth": 1,
    "screenHeight": 1500,
    "screenWidth": 1500,
    "timezoneOffset": 60,
    "language": "EN"
  }
}'
```

Response body:

HTTP Response OK - FRICTIONLESS:

```
HTTP/1.1 200 OK
Content-Length: 2153
Content-Type: application/json;charset=UTF-8

{
  "transactionId": "08ea8e28-0aad-45eb-8368-f15bdadd5eba",
  "status": "DEPOSITED",
  "approvalCode": "20210615100078475104",
  "pageContent": null,
  "externalTransactionId": "49a91f00-26b4-49a2-9c77-ed37646ddf64",
  "itemId": "08ea8e28-0aad-45eb-8368-f15bdadd5eba",
  "acsUrl": null,
  "threeDsMode": "FRICTIONLESS",
  "threeDsMethodUrl": null,
  "threeDSMethodData": null,
  "creq": null
}
```

HTTP Response OK - CHALLENGE:

```
HTTP/1.1 200 OK
Content-Length: 2153
Content-Type: application/json;charset=UTF-8

{
```

```
  "transactionId": "08ea8e28-0aad-45eb-8368-f15bdadd5eba",
  "status": "INITIALIZED",
  "pageContent": "PGh0bWw+PFNDUklQVCBMQU5HVUFHRT0iSmF2YXNjcmlwdCI+Zn....",
  "orderNumber": null,
  "externalTransactionId": "49a91f00-26b4-49a2-9c77-ed37646ddf64",
  "itemId": "08ea8e28-0aad-45eb-8368-f15bdadd5eba",
  "acsUrl": "http://acs-site.com",
  "threeDsMode": "CHALLENGE",
  "threeDsMethodUrl": null,
  "threeDSMethodData": null,
  "creq": "0bWw+PGhlYWQ+CiAgICAgICAg1ldGEgSF48UVRVUlWPSJQcmFnbWEiIENPTlRF"
}
```


HTTP Response OK - THREE_DS_METHOD:


```
HTTP/1.1 200 OK
Content-Length: 2153
Content-Type: application/json;charset=UTF-8

{
  "transactionId": "08ea8e28-0aad-45eb-8368-f15bdadd5eba",
  "status": "INITIALIZED",
  "pageContent": null,
  "externalTransactionId": "49a91f00-26b4-49a2-9c77-ed37646ddf64",
  "itemId": "08ea8e28-0aad-45eb-8368-f15bdadd5eba",
  "acsUrl": null,
  "threeDsMode": "THREE_DS_METHOD",
  "threeDsMethodUrl": "http://three-ds-method-site.com",
  "threeDSMethodData": "YW5zSUQiIlDogImQ4ODhjOGEwLNWJkNy04MDAwLTAwMDAwaW9uQXN5bmMiIH0",
  "creq": null
}
```


Response fields:		
Type	**Value**	**Constraints**
transactionId	String	Identifier of transaction
status	String	Transaction status
approvalCode	String	Acquirer approval code

Response fields:		
Type	**Value**	**Constraints**
pageContent	Null	This field is a BASE64 encrypted html source file containing the challenge 3-D Secure frame
externalTransactionId	String	External transaction id
acsUrl	Null	URL address of ACS. Browser should be redirect on this address with passing creq/pareq parameter.
threeDsMode	String	ThreeDS process mode which informs about. One of: [CHALLENGE, THREE_DS_METHOD, FRICTIONLESS]
threeDsMethodUrl	Null	URL for perform THREE_DS_METHOD. Present when threeDsMode=THREE_DS_METHOD
threeDSMethodData	Null	Parameter which should be pass (without changes) with redirection to ACS URL. Only in THREE_DS_METHOD flow
itemId	String	Merchant’s unique id of transaction. Ensures the idempotency of the transaction.
creq	Null	cReq message which should be transferred (with no changes) with redirection to the ACS url address. Present when threeDs with CHALLENGE

Continue one time payment with 3ds

POST [base-url]/one-time-payment/3ds/continue

The method allows continuing payment using ThreeDs 2.0 protocol. This method should be executed after perform process ThreeDsMethod flow - this step is mandatory only when ThreeDsMethod flow is present (Initialize payment with 3ds returned **threeDsMode = THREE_DS_METHOD**). The Continue one time payment with 3ds method can be performed directly by the Merchant or internally by Verestro - depending on the Merchant's decision.

> Selecting the option in which Verestro performs the Continue one time payment with 3ds method releases the customer from the need to integrate this method. In addition, in this situation, the Merchant does not need to handle the **threeDsMode=THREE_DS_METHOD**

> case, because it will never occur.

Method Continue one time payment with 3ds requires parameter **methodCompletionIndicator** to be set in the request:
Y - when notification after from acs on the threeDsMethodNotificationUrl was present
N - when notification after from acs on the threeDsMethodNotificationUrl was not present
U - no information about notification

As a response there are 2 possible paths:

- **Frictionless flow** - In response: **threeDsMode = FRICTIONLESS** are present. This response denotes that payment was finished.
- **Challenge flow** - In response: **threeDsMode = CHALLENGE** is present. This response denotes that the challenge process is required. In this flow pageContent is present. The parameter should be decoded and presented to the user for interaction (challenge). After the challenge end, html will redirect the user to URL passed in the termUrl parameter. The Finalize one time payment with 3ds request should be performed then.

Request headers:			
Type	**Value**	**Constraints**	**Description**
Content-Type	application/json	Required	Content type of the request
Authorization	Basic bG9naW46cGFzc3dvcmQ=	Required	Basic Authorization token

Request body:

```
POST /champion/one-time-payment/3ds/continue HTTP/1.1
Content-Length: 181
Content-Type: application/json
Host: merchant.upaid.pl

{
  "itemId": "08ea8e28-0aad-45eb-8368-f15bdadd5eba",
  "methodCompletionIndicator": "Y",
  "encryptedData": "eyJ0eXAiOiJKT1NFIiwiZW5jIjoiQTI1NkdDTSIsImlhdCI6MTY0NDkxMjk5MSwiY..."
}
```

Request fields:

Type	Value	Constraints	Description
itemId	String	Not empty	Transaction amount (in pennies)
methodCompletionIndicator	String	Not empty	Flag informs about invoke 3DS Method: **Y** = success, **N** = failed, **U** = no data
encryptedData	String	Not empty	The field contains encrypted JSON using the JWE standard. Object EncryptedData is described in section of schemas.

Example cURL:

```
$ curl 'https://merchant.upaid.pl/champion/one-time-payment/3ds/continue' -i -X POST \
    -H 'Content-Type: application/json' \
    -d '{
  "itemId": "08ea8e28-0aad-45eb-8368-f15bdadd5eba",
  "methodCompletionIndicator": "Y",
  "encryptedData": "eyJ0eXAiOiJKT1NFIiwiZW5jIjoiQTI1NkdDTSIsImlhdCI6MTY0NDkxMjk5MSwiY..."
}'
```

Response body*:*

HTTP Response OK - FRICTIONLESS:

```
HTTP/1.1 200 OK
Content-Length: 2153
Content-Type: application/json;charset=UTF-8

{
  "transactionId": "08ea8e28-0aad-45eb-8368-f15bdadd5eba",
  "status": "DEPOSITED",
  "approvalCode": "20210615100078475104",
  "pageContent": null,
```

```
  "externalTransactionId": "49a91f00-26b4-49a2-9c77-ed37646ddf64",
  "itemId": "08ea8e28-0aad-45eb-8368-f15bdadd5eba",
  "acsUrl": null,
  "threeDsMode": "FRICTIONLESS",
  "threeDsMethodUrl": null,
  "threeDSMethodData": null,
  "creq": null
}
```


HTTP Response OK - CHALLENGE:


```
HTTP/1.1 200 OK
Content-Length: 2153
Content-Type: application/json;charset=UTF-8

{
  "transactionId": "08ea8e28-0aad-45eb-8368-f15bdadd5eba",
  "status": "INITIALIZED",
  "pageContent": "PGh0bWw+PFNDUklQVCBMQU5HVUFHRT0iSmF2YXNjcmlwdCI+Zn....",
  "orderNumber": null,
  "externalTransactionId": "49a91f00-26b4-49a2-9c77-ed37646ddf64",
  "itemId": "08ea8e28-0aad-45eb-8368-f15bdadd5eba",
  "acsUrl": "http://acs-site.com",
  "threeDsMode": "CHALLENGE",
  "threeDsMethodUrl": null,
  "threeDSMethodData": null,
  "creq": "0bWw+PGhlYWQ+CiAgICAgICAg1ldGEgSF48UVRVUlWPSJQcmFnbWEiIENPTlRF"
}
```


Response fields:		
Type	**Value**	**Constraints**
transactionId	String	Identifier of transaction
status	String	Transaction status
approvalCode	String	Acquirer approval code

Response fields:		
Type	**Value**	**Constraints**
externalTransactionId	String	External transaction id
itemId	String	Merchant’s unique id of transaction. Ensures the idempotency of the transaction.
acsUrl	Null	URL address of ACS. Browser should be redirect on this address with passing creq/pareq parameter.
threeDsMode	String	ThreeDS process mode which informs about. One of: [CHALLENGE, FRICTIONLESS]
creq	Null	cReq message which should be transferred (with no changes) with redirection to the ACS url address. Present when threeDs with CHALLENGE

Finalize one time payment with 3ds

POST [base-url]/one-time-payment/3ds/finalize

This method allows finalizing payment using ThreeDs 2.0 protocol. This method should be executed after successfully performed challenge by the user.

Request headers:			
Type	**Value**	**Constraints**	**Description**
Content-Type	application/json	Required	Content type of the request
Authorization	Basic bG9naW46cGFzc3dvcmQ=	Required	Basic Authorization token

Request body:

```
POST /champion/one-time-payment/3ds/finalize HTTP/1.1
Content-Length: 194
Content-Type: application/json
Host: merchant.upaid.pl

{
  "itemId": "08ea8e28-0aad-45eb-8368-f15bdadd5eba",
  "cRes": "IiOiI0ODY2Njc3Nzg4OCIsImludGVybmFsS",
  "encryptedData": "eyJ0eXAiOiJKT1NFIiwiZW5jIjoiQTI1NkdDTSIsImlhdCI6MTY0NDkxMjk5MSwiY..."
}
```

Request fields:			
Type	**Value**	**Constraints**	**Description**
itemId	String	Not empty	Transaction amount (in pennies)
cRes	String	Optional	Message obtained after CHALLENGE process on the endpoint passed in termUrl.
encryptedData	String	Not empty	The field contains encrypted JSON using the JWE standard. Object EncryptedData is described in section of schemas.

Example cURL:

```
$ curl 'https://merchant.upaid.pl/champion/one-time-payment/3ds/finalize' -i -X POST \
    -H 'Content-Type: application/json' \
    -d '{
  "itemId": "08ea8e28-0aad-45eb-8368-f15bdadd5eba",
  "cRes": "IiOiI0ODY2Njc3Nzg4OCIsImludGVybmFsS",
  "encryptedData": "eyJ0eXAiOiJKT1NFIiwiZW5jIjoiQTI1NkdDTSIsImlhdCI6MTY0NDkxMjk5MSwiY..."
}'
```

Response body:

HTTP Response OK:

```
HTTP/1.1 200 OK
Content-Length: 251
Content-Type: application/json;charset=UTF-8

{
  "transactionId" : "08ea8e28-0aad-45eb-8368-f15bdadd5eba",
  "itemId" : "08ea8e28-0aad-45eb-8368-f15bdadd5eba",
  "paymentAmount" : 100,
  "status" : "DEPOSITED",
  "externalTransactionId" : "49a91f00-26b4-49a2-9c77-ed37646ddf64",
  "links" : [ ]
}
```

Response fields:		
Type	**Value**	**Constraints**
transactionId	String	Identifier of transaction
itemId	String	Merchant’s unique id of transaction. Ensures the idempotency of the transaction.
status	String	Transaction status
externalTransactionId	String	External transaction id

Get public key

GET [base-url]/one-time-payment/public-key

This method allows to get publicKey. This method is auxiliary for other methods: One Time Payment. Obtained public key will allow the Merchant to encrypt sensivite data of the transaction he collected from the user. This data is required to perform transaction.

Request headers:			
Type	**Value**	**Constraints**	**Description**
Content-Type	application/json	Required	Content type of the request
Authorization	Basic bG9naW46cGFzc3dvcmQ=	Required	Basic Authorization token

Request body:

```
GET /champion/one-time-payment/public-key HTTP/1.1
Content-Type: application/json
Host: merchant.upaid.pl
```

Example cURL:

```
$ curl 'https://merchant.upaid.pl/champion/one-time-payment/3ds/finalize' -i -X POST \
    -H 'Content-Type: application/json' \
    -d '{
  "itemId": "08ea8e28-0aad-45eb-8368-f15bdadd5eba",
  "cRes": "IiOiI0ODY2Njc3Nzg4OCIsImludGVybmFsS",
  "encryptedData": "eyJ0eXAiOiJKT1NFIiwiZW5jIjoiQTI1NkdDTSIsImlhdCI6MTY0NDkxMjk5MSwiY..."
}'
```

Response body:

HTTP Response OK:

```
HTTP/1.1 200 OK
Content-Length: 610
Content-Type: application/json;charset=UTF-8

{
  "value" :
"LS0tLS1CRUdJTiBQVUJMSUMgS0VZLS0tLS0KTUlJQklqQU5CZ2txaGtpRzl3MEJBUUVGQUFPQ0FROEFNSUlCQ2dLQ0
FRRUE5QXdvaDlzRW1CeDFUK2llb0lVMWpLUVJuUmg3SGFBS0tHdGdPTFQ3WDIyYlVuaHNDbld0SEVSR2pOcVBsQ2
05Wmh5SHo2ajFlSi9UVmdhQ1NEYmtKZFNUMmpCVDZDKzBKb0NnYld0S2EwblIlMVI4NWNVWUgzT3dYcFBZSHhG
MW1MRlRVU2doUXFBVXZ1QzVla2FTQi9lTGV2WmRXTTcyazAybTZ0bGJUVWRhbUtQRnp1VzhxMGpZYUZ0Z01XYXl
ZUG81WXJJU1ZWWVZTVGdVZjZqR2ZCQUZ3N2d5dHVBaFlUMTVpT2g0NWtpc2pZb1BzaEd2RitqQ0FSbzlDMDhKST
dubVVGSEUrczNYQm84S0t0THNNKzhvcWNZdy9maVhlbkljSHNvcTQ2bnBWMEtTWW9MZm5pa0hnTnoxUFR0dnpB
Z29NeXdSSDVrdURhYmNEeVRoZVFJREFRQUIKLS0tLS1FTkQgUFVCTElDIEtFWS0tLS0t"
```

```
}
```

Response fields:		
Type	**Value**	**Constraints**
value	String	Public key encoded with Base64

Get transaction details

GET [base-url]/transactions/${transactionId}

This method is optional and does not affect the process of the transaction itself. Nevertheless, Verestro recommends using this method. In case of any problem with the connection, it allows the Merchant to know the current status of a given transaction. This method is secured by Merchant's account credentials (basic authorization). Merchant's account is created by Verestro during the onboarding process.

Request headers:			
Type	**Value**	**Constraints**	**Description**
Authorization:	Basic bG9naW46cGFzc3dvcmQ=	Required	Merchant's account credentials
Content-Type	application/json	Required	Content type of the request

Response body:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 245

{
  "transactionId" : "ee58ef03-d6ed-4a07-8885-d050c439ec6c",
  "amount" : 100,
  "currency" : "PLN",
  "description" : "description",
  "status" : "DEPOSITED",
```

```
  "threeDsMode" : "FRICTIONLESS"
}
```

Response fields		
Name	**Name**	**Description**
transactionId	String	Identifier of transaction. This parameter should be provided in the deposit method if the tokenPayment method returned AUTHORIZED transaction status. This parameter also defines in the context of which transaction Verestro should return information when executing the getTransactionDetails method.
amount	Number	Transaction amount in pennies
currency	String	Transaction currency
description	String	Transaction description
status	String	One of the possible transaction statuses
threeDsMode	String	ThreeDS process mode which informs about

Example cURL:

```
$ curl 'https://merchant.upaid.pl/champion/transactions/ee58ef03-d6ed-4a07-8885-d050c439ec6c' -i -u
'login:password' -X GET \
    -H 'Content-Type: application/json'
```

Error examples

This chapter lists examples of errors that may occur when using the One Time Payment API solution.

HTTP Response - VALIDATION_ERROR:

```
HTTP/1.1 400 Bad Request
Content-Length: 32
Content-Type: application/json;charset=UTF-8=

{
```

```
  "status": "VALIDATION_ERROR",
  "message": "Some fields are invalid",
  "data": [
    {
      "field": "{{field_name_from_request}}",
      "message": "{{message}}"
    }
  ],
  "traceId": "{{traceId}}"
}
```

HTTP Response - UNAUTHORIZED:

```
HTTP/1.1 401 Unauthorized
Content-Length: 32
Content-Type: application/json;charset=UTF-8=

{
  "timestamp": "{{timestamp}}",
  "status": 401,
  "error": "Unauthorized",
  "message": "Unauthorized",
  "traceId": "{{traceId}}"
  "path": "{{path}}"
}
```

HTTP Response - UNPROCESSABLE_ENTITY:

```
HTTP/1.1 422 Unprocessable Entity
Content-Length: 32
Content-Type: application/json;charset=UTF-8=

{
  "status": "ERROR_INVALID_CVC",
  "message": "Invalid cvc",
  "traceId": "{{traceId}}"
}
```

HTTP Response - INTERNAL_SERVER_ERROR:

```
HTTP/1.1 500 Internal Server Error
Content-Length: 32
Content-Type: application/json;charset=UTF-8=

{
   "status": "INTERNAL_SERVER_ERROR",
   "message": "Internal server error exception",
   "traceId": "{{traceId}}"
}
```

Revision #96
Created 6 June 2022 03:31:53 by Jakub Kotyński
Updated 2 March 2023 13:58:03 by Jagoda Mazurek