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 Customer have to proceed onboarding process. To create account please contact with support.

Integration

This section describes how to integrate the solution using the SDK provided by Verestro. The merchant-paytool.js is a JavaScript-based client-side SDK. The SDK adds MerchantPayTool class to the global JavaScript scope which can be further instantiated to start Donate Widget’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 postback section.

Additionally, the PayTool SDK provides a method that allows you to "get transaction details" using "*transactionId*" parameter.

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.

initData		
Name	**Type**	**Description**
redirectUrls.failureUrl	string	The url where users will be redirected after a failed payment.

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'
        }
      }}
    />
  );
};
```

> In case of questions regarding the integration, Verestro actively supports Customer in the implementation.

Postbacks

This section describes the method that allows Customer to receive notifications after every donation made. If Customer wants to handle notifications after transactions Customer must create HTTP POST endpoint which will accept requests in format JSON. If this option is enabled, the Donate Widget API will send information about the donation made to the endpoint provided by the

Customer.

> This feature is optional but we strongly recommend using it. Without this method, the Customer will not receive any information whether the donation was successful or not.

POST site.customer.com/notifications

Request body:

```
POST site.customer.com/notifications HTTP/1.1
Content-Type: application/json
Content-Length: 265

{
  "transactionId": "3a1e9961-75bc-4279-8791-033c180fa239",
  "status": "DEPOSITED",
  "amount": 100,
  "currency": "USD",
  "description": "description"
}
```

Request headers:			
Type	**Value**	**Constraints**	**Description**
Content-Type	application/json	Required	Content type of the request.

Request fields:			
Type	**Value**	**Constraints**	**Description**
transactionId	String	Required	Unique Id of transaction.
status	String	Required	Transaction status. **AUTHORIZED**, **DEPOSITED**, **FAILED**, **UNKNOWN**, **REFUNDED**, **REVERSED.**
amount	Number	Required	Amount for transaction (minor units of currency).
currency	String	Required	Currency of given amount.

Request fields:			
Type	**Value**	**Constraints**	**Description**
description	String	Required	Simple description of transaction.

Response status: HTTP/1.1 200 OK

Example cURL:

```
$ curl 'https://site.customer.com/notifications' -i -u 'login:password' -X POST -H 'Content-Type: application/json' -d '{

  "transactionId": "387cd038-fa39-40e1-a6ac-0d610b594787",
    "status": "DEPOSITED",
  "amount": 100,
  "currency": "USD",
  "description": "description",
}'
```

Get transaction details

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

Method allows getting transaction details using "*transactionId*". Method is protected by BasicAuth of the Customer.

Request:

```
GET /champion/transactions/cd670818-dfbe-44fc-8948-8fbf8992d8d3 HTTP/1.1
Authorization: Basic bG9naW46cGFzc3dvcmQ=
Content-Type: application/json
Host: merchant.upaid.pl
```

Request headers:			
Type	**Value**	**Constraints**	**Description**

Content-Type	application/json	Required	Content type of the request.
Authorization	Basic bG9naW46cGFzc3dvcmQ=	Required	Basic Authorization token.

Response body:

HTTP Response - SUCCESS:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 211

{
  "transactionId" : "cd670818-dfbe-44fc-8948-8fbf8992d8d3",
  "amount" : 100,
  "currency" : "PLN",
  "description" : "description",
  "bin" : "4442211",
  "lastFourDigits" : "1234",
  "status" : "DEPOSITED"
}
```

Response fields:		
Type	**Value**	**Description**
transactionId	String	Unique Id of transaction.
amount	Number	Transaction amount in pennies.
currency	String	Transaction currency.
description	String	Transaction description.
bin	String	Card bin.
lastFourDigits	String	Card last four digits.

Response fields:		
Type	**Value**	**Description**
status	String	Transaction Status. Possible values: DEPOSITED - transaction finished with success INITIALIZED_3DS - transaction during processing FAILED - transaction failed.

Example cURL:

```
$ curl 'https://merchant.upaid.pl/champion/transactions/cd670818-dfbe-44fc-8948-8fbf8992d8d3' -i -u
'login:password' -X GET \
    -H 'Content-Type: application/json'
```

Revision #21
Created 29 June 2022 12:45:02 by Jakub Kotyński
Updated 12 January 2024 12:21:18 by Jagoda Mazurek