

Paytool

A solution based on the Web Checkout product but extended by various payment methods such as Google Pay™, Apple Pay or Blik. The whole solution is based on the two integration models: payment using the Paytool frontend application or via a direct server to server connection. Our solution frees you from the need to integrate with acquiring institutions and banks. The payment process and threeDs authentication are performed on the Verestro side.

- [Introduction](#)
- [Overview](#)
- [Use cases](#)
- [Onboarding](#)
- [How to integrate](#)
- [Test Guide](#)

Introduction

Verestro Paytool is a solution that provides a payment gateway functionalities that supports various methods of payments. By using our solution your users can easily pay for their purchases selecting the Verestro Paytool as the payment option. This technology is delivered in the form of payment widget or API method which is opening payment session in Verestro server. After opening a payment session, the transaction can be carried out in two ways - either via widget or via API.

Tip: Check it now. Verestro Paytool demo application is available [here](#).

How to connect with us?

The Paytool solution provides two independent integration paths.

Web integration

The first way is that after opening a payment session, your browser should redirect the payer to the Paytool application or open it in iframe. Verestro will be fully responsible for the execution of transactions and handling 3D Secure authentication. In the application the payer can see the transaction metadata and chooses payment method he wish to pay. All required payment data is transferred to the Verestro Paytool backend over the Verestro internal network.

API integration

The second way is that after opening a payment session, you can carry out the payment process using the API methods provided by us. Your server should then integrate with our API, which allows you to make payments and carry out the 3D Secure process. Verestro does not provide any frontend application in this way of integration.

Tip: More information about Paytool API can be found in [API integration guide](#).

Note: It is required that you have an account in Acquirer's system which will settle your transactions. For more informations please contact our Sale Department. We are suggesting to use [Fenige](#) as this is our partner acquiring institution and we are fully integrated with this Acquirer

Overview

This document provides a description of functionalities offered by Verestro Paytool. Our solution supports various payment methods such as Google Pay™, Apple Pay, Blik and many others in the form of the payment gateway. In addition, you can decide which payment methods should be enabled. Simply put, you may decide that, for example, you want Verestro Paytool to provide payment via Google Pay but payment via Apple Pay should be disabled. In such a situation, the end user will see the Google Pay as the available payment method in the Verestro Paytool payment form, but the Apple Pay payment method will not appear at all. Transaction process mainly takes place on Verestro's side. This means that you are completely relieved of responsibility for processing the transaction and/or performing 3D Secure authentication. The only action that the you must do is to provide metadata of the transaction, which includes order number, description, amount, currency and some optional parameters.

Important! Note that if you require the settlement of the transaction by an Acquirer to which Verestro is not integrated there will be required new integration between Verestro and the new Acquirer. You should provide the specification of the new Acquirer which will allow us to perform integration.

Abbreviations

In this chapter there are abbreviations and acronyms used in the document listed in below table.

Abbreviation	Description
ACQ	Acquiring Institution / Acquirer
ACS	Access Control Server
SDK	Software Development Kit
PSP	Payment Service Provider
OS	Operative System
Mid	Merchant Identifier
PCI DSS	Payment Card Industry Data Security Standard
PAN	Permanent Account Number
CVC	Card Verification Code
3DS	3-D Secure

Terminology

This section explains a meaning of key terms and concepts used in the document.

Name	Description
Customer/Merchant	Institution which uses Verestro products. This institution decides which payment method should be available in the solution and how transaction should be processed.
End user/payer	The entity which uses Paytool solution to pay for ordered good from Customer. It is root of entity tree. End user is an owner of the wallet/card and he decides to pay for the purchase using Paytool solution, selecting it from the list of payment methods available in the Customer application.
Payment service provider	The entity which provides a payment services for external Customers who do not have direct integration with acquirers or are not PSI DSS compilent. From the perspective of he Paytool application, Verestro is the PSP.
Card	Card belongs to the user. If user intends to pay with the Paytool solution using plain card payment method, then has to insert required card's data to the appropriate fields shared by the Paytool solution payment form. Card data will not be stored in the Verestro system. They will be provided to Acquirer.
Card payment token	It is a numerical value in the form of a PAN number. It shows a given card from Google Pay or Apple Pay wallet. The card payment token replaces the card number and is delivered by Google Pay/Apple Pay to Verestro if the end user selects one of the two above mentioned payment options. Verestro passes this value to Acquirer for the payment to be made.
Authorization Method	The way of the authentication of the Google Pay™ card transaction. Verestro supports followed authorization methods: <code>PAN_ONLY</code> and <code>CRYPTOGRAM_3DS</code> if Customer's country belongs to the European Union. Authorization method is always provided in the Google Pay™ encrypted payload as <code>authMethod</code> parameter.
Gateway Id	Phrase/value that identifies a given Payment Service Provider in the Google Pay™ system. The Merchant provides gateway Id to Google Pay™ to obtain a card payment token. By provided gateway Id, Google Pay™ encrypts the card payment token with the appropriate public key. Verestro is defined by a gateway Id with the value <code>verestro</code> in Google Pay™ server.

Gateway Merchant Id	Unique Customer identifier assigned by Verestro during the onboarding process. This identifier is in the form of a <code>UUID</code> . Verestro understands and uses this to verify that the message was for the Customer that made the request. Customer passes it to Google Pay™. More information about the Gateway Merchant Id can be found in Google Pay™ documentation .
MID	Merchant identifier. This entity is representing Customer in the Acquirer's system. Customer has to provide the mid information to enable mid configuration in the Verestro system. Required to process transactions and 3DS process via Verestro system.
Bank/Issuer	Card issuing institution. In the case of an e-commerce transaction, this entity is responsible for checking whether the cardholder's balance has the appropriate amount of funds to perform a given transaction, determining whether 3D secure authentication is necessary or simply checking whether the card is active.
Cardholder	This is the end user who pays for his purchases using one of the available payment options in Verestro Paytool.
PAN	It is 7-16 digits of the credit/debit card number. These digits contain the Permanent Account Number assigned by the bank to uniquely identify the account holder. It is necessary to provide it when end user wants to pay with a card for purchases via Verestro Paytool solution.
CVC	Card Verification Code. It is a type of security code protecting against fraud in remote payments. CVC is necessary to provide it when end user wants to pay with a card for purchases via Paytool solution.
Expiration date	It is a date of the card validity ending and contains two values – month/year - for example 01/28. Card will be valid to the last day of the month of the year showed on it. It is necessary to provide it when end user wants to pay with a card for purchases via Verestro Paytool solution.
3DS	3-D Secure is a method of authorization of transaction made without the physical use of a card, used by payment organization. The 3DS process in the Verestro Paytool solution is performed internally in the Verestro system which means the Customer is not responsible for end user authentication.
PCI DSS	It is a security standard used in environments where the data of payment cardholders is processed. The standard covers meticulous data processing control and protection of users against violations.

Implementation models

Integration with Paytool should be performed by API call. To initiate a payment, you must request an [transaction initialization](#) method, which in response opens a payment session with a unique identifier. Once you have an active payment session and its identifier, you can choose one of two available payment processing methods described in [Redirect your payer](#) and [Payment process via API](#) chapters.

Tip: It is also important to mention that you should create a server method which we will be used to send you [postback after transaction](#). This step is not required but we highly recommend it as this is the way we will inform you about transaction status. We can also send e-mail post-transaction notification to your payer. More information about transaction postbacks are described in the [Use cases](#) and [How to integrate](#) chapters.

Redirect your payer

The first way is to redirect your payer to the payment webview or open this webview in iframe. This implementation model is more comprehensive because when redirecting the payer, you only need to provide us with transaction metadata and the payment session identifier. We are responsible for the rest of the payment process.

You need to authenticate your merchant account providing `Basic-Authorization` data in the [transaction initialization](#) method header.

Tip: We highly recommend using this integration model because it is much simpler and faster to implement. Additionally, most of the responsibility for the process is on our side.

Note: The `Basic-Authorization` data will be issued to you after completing the [onboarding process](#).

Example request body of the transaction metadata provided by Customer

```
{
  "transactionId": "42be3d06-4577-4a9f-b525-2cfaba244557",
  "currencyCode": "PLN",
  "amount": 100,
```

```
"description": "Test transaction",
"formLanguage": "en",
"redirectUrl": {
  "successUrl": "https://paytool.verestro.com/demo/?success=1",
  "failureUrl": "https://paytool.verestro.com/demo/?success=0"
},
"sender": {
  "firstName": "Yoshimoto",
  "lastName": "Imagawa",
  "address": {
    "countryCode": "PL",
    "city": "Kyoto",
    "postalCode": "12-345",
    "street": "Ichijo",
    "houseNumber": "1"
  }
},
"merchantUrl": "https://paytool.verestro.com/demo/",
"orderNumber": "1"
}
```

Sequence diagram of the payer redirection process

```
@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
}
actor payer
participant "Customer front" as cfront
participant "Customer backend" as cback
participant "Paytool front" as pfront
participant "Paytool backend" as pback
payer->cfront: Pay with Paytool
cfront->cback: Payer choosed Paytool
cback->pback: Customer authorization + transaction metadata
pback->pback: Store transaction metadata + open payment session
cback<-pback: OK + transaction id + redirect url
cfront<-cback: OK + transaction id + redirect url
cfront->pfront: Redirect + transaction id
pfront->pback: Get transaction data + merchant payment methods
pfront<-pback: OK response
payer<-pfront: Display transaction + payment methods
@enduml
```

Payment process via API

The second way to continue the payment process is to integrate your application along with payment session identifier with the rest of the API methods provided by our solution. By continuing the process along this path, you need to call the individual payment method you want to use and handle the threeDs authentication process if required. We are responsible for carrying out the above-mentioned processes, but you are the entity that must request individual methods responsible for every step of a given transaction. We are responsible for the rest of the payment process. In this case, you authenticate with the `Basic-Authorization` data passed in the header.

Tip: We highly recommend using [Redirect your payer](#) integration model because it is much simpler and faster to implement. Additionally, in the [Redirect your payer](#) integration model most of the responsibility for the process is on our side.

Important: In this integration model we do not provide any frontend view.

Note: The `Basic-Authorization` data will be issued to you after completing the [onboarding process](#).

Example request body of the transaction metadata provided by Customer

```
{
  "transactionId": "42be3d06-4577-4a9f-b525-2cfaba244557",
  "currencyCode": "PLN",
  "amount": 100,
  "description": "Test transaction",
  "formLanguage": "en",
  "redirectUrl": {
    "successUrl": "https://paytool.verestro.com/demo/?success=1",
    "failureUrl": "https://paytool.verestro.com/demo/?success=0"
  },
  "sender": {
    "firstName": "Yoshimoto",
    "lastName": "Imagawa",
    "address": {
      "countryCode": "PL",
      "city": "Kyoto",
      "postalCode": "12-345",
      "street": "Ichijo",
      "houseNumber": "1"
    }
  },
  "merchantUrl": "https://paytool.verestro.com/demo/",
  "orderNumber": "1"
}
```

Sequence diagram of the payment via API process

```
@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
}
actor payer
participant "Customer front" as cfront
participant "Customer backend" as cback
participant "Paytool front" as pfront
participant "Paytool backend" as pback
payer->cfront: Pay with Paytool
cfront->cback: Payer choosed Paytool
cback->pback: Customer authorization + transaction metadata
pback->pback: Store transaction metadata + open payment session
cback<-pback: OK + transaction id
note right of cfront: At this point, the Customer decides how to present the transaction to the
payer
cback->pback: Request proper payment method / executing 3D Secure
cback<-pback: Transaction result
note right of cfront: At this point, the Customer decides how to present the transaction result to
the payer
@enduml

```

Application components

Application components are all Verestro's internal services which are taking part in the Paytool processes. This chapter describes every component of the Verestro Paytool solution along with it's responsibility.

Verestro Paytool Server

Verestro Paytool Server is the backend component which consists of few internal services responsible for managing cards, card payment tokens and user data, processing transactions and 3D Secure, send notification to the Customer and enduser and storing transaction history. This component is also responsible for connection with Acquirers. Services included in the Verestro Paytool backend can be divided into two groups:

Services that are the part of the Verestro Paytool Solution.

Services supporting the functionalities offered by Verestro Paytool Solution.

Services that are the part of the Verestro Paytool Solution

Component	Description
Paytool API	A service with all methods required to complete the entire transaction process. The methods are called by Paytool Frontend App or by your API in the right order to make the entire payment and 3D Secure process. This service also communicates with the Verestro Acquirer Connector, which orders the execution of the transaction. The last and probably the most important element for which the Paytool API is responsible is opening a payment session and saving the transaction entities in the Verestro system.

Services supporting the functionalities offered by the Verestro Paytool Solution

Component	Description
Midas API	A connector between the Verestro system and the Acquirer's system. This service transfers transaction requests to the Acquirers and also informs if the 3D Secure authentication process is required.
Notification Service API	A service responsible for sending notifications to end users and Customers. Notifications to end user can be sent via e-mail. The Customer can receive transaction postback via a specific URL he provided.
Admin Panel API	<p>A service which is communicating with Paytool API along with other listed services supporting Verestro Paytool solution. Admin Panel API provide all obtained data to the Admin Panel Frontend allowing the Customer to perform many actions such as displaying transaction history, downloading transaction reports or ordering refunds.</p> <div>Warning: Implementation is work in progress...</div>

Verestro Paytool Frontend

Verestro Paytool Frontend is the frontend component consists of two internal services which are responsible for displaying all necessary data coming from Paytool API. Verestro Paytool Frontend can be divided into two services:

Component	Description
Paytool Frontend App	<p>This is a frontend application hosted by Verestro. This is where you redirect the user when you are using the Redirect your payer integration path. This service is intended to display transaction data to the end user, enable him to select a payment method and confirm payment. To perform the above actions, the Paytool Frontend App communicates directly with the Paytool API. This service does not participate in the payment process at all if you use the Payment process via API integration path. Alternatively you can open Paytool in iframe.</p>
Admin Panel Frontend	<p>Independent frontend application hosted on the Verestro side. This website allows you to manage your account in the Verestro system. From the Admin Panel Frontend, you are able to:</p> <ul style="list-style-type: none">• view transaction history• generate transaction reports• configure mid/terminals• perform refunds• manage the Paytool frontend form appearance <div>Warning: Implementation is work in progress...</div> <div>Info: This service is not obligatory. It is intended for Customers who want to have more control over their account in the Verestro system and want to be able to order a reversal manually.</div>

Allowed card networks

Listed below are the types of cards supported in transactions using Paytool application:

Card type
MASTERCARD

VISA

MAESTRO

Security

Due to the need to process card data and perform money operations, we had to create security measures that would not allow violations of the transaction process and prevent unauthorized entities from using the solution. In this chapter, we described the main security elements for customers and their transactions.

The sequence diagram below illustrates the application workflow

```
@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 "Customer Frontend" as browser
participant "Customer Backend" as psdk
participant "Paytool Frontend" as pfront
participant "Paytool Backend" as pback
participant "Acquirer" as acq
note right of browser: User chooses "Pay with Verestro Paytool"
browser->>psdk: Transaction initialization + metadata
```

```
psdk-->pback: Transaction initialization + metadata
pback->pback: Validate transaction metadata
pback->pback: Store transaction session data
pback->psdk: OK + transactionId
psdk->pfront: Redirect end user to Paytool Frontend + transactionId
pfront-->pback: Get transaction metadata + list of the supported payment options
pback-->pfront: Return transaction metadata + list of the supported payment options
pfront->pfront: Display transaction metadata + list of the supported payment options
pfront->pback: Process transaction by chosen payment option
pback->pback: Consistency validation between current transaction data and provided when
opening the session
pback->acq: Order transaction
@enduml
```

Payment session

In order to start a payment in Paytool, a payment session must first be opened. Opening a payment session involves authorizing your merchant account and sending transaction metadata to Paytool API. Transaction metadata should be encrypted using JWE encryption standard. Based on the obtained transaction data, validation of each data is performed and then a transaction object is created and saved in the internal Verestro database. Thanks to authorization, the context of your merchant account is assigned to the created transaction. Once you have created a payment session, you will receive an identifier for this transaction. Using this identifier you will be able to continue the payment process and we will be able to check whether there has been any interference in the transaction data you provided and interrupt the process if necessary.

Note: We does not store any sensitive data such as PAN or CVC in our system. The obtained data are only required to be transferred to the Acquirer to perform transaction.

Authorization

Authorization in the Paytool application is intended to check whether the entity trying to execute the request is authorized to do so. If you have a merchant account in the Paytool system, each of your requests should be signed with the `Basic-Authorization` data for both [Redirect your payer](#) and [Payment process via API](#) integration paths. Using one of these data, we will check whether the action you have taken can be implemented. We will also check whether your merchant account is associated with a given transaction, and therefore whether it can perform any actions in the context of this transaction, and whether such a merchant account even exists in our system.

Note: The Basic-Authorization Basic-Authorization data will be issued to you after completing the [onboarding process](#).

Use cases

This chapter is mainly dedicated to the appearance of the application and a description of the individual paths that the end user will follow depending on the payment method he choosed. Each of the payment method available in Paytool application is described in a separate section. Additionally, every chapter contains diagrams depicting the entire transaction process per payment method.

Tip: Check it now. Verestro Paytool demo application is available [here](#).

Transaction initialization

The first step of the each transaction always takes place in your application. Your application integrated with the Verestro Paytool solution should have exposed option allowing to pay using Paytool available in your application. Your backend should deliver the initial transaction data to the Paytool API to open new transaction session when your payer selects the Paytool as payment option. Our API will return you the transaction identifier and the payer's redirection address. The view of transaction initialization from the end user's perspective has been shown below. This view refers to the [Redirect your payer](#) integration path:

Image-1687851731553.png

Image-1713956541734.png

Tip: We are fully responsible for the execution of the transaction and [threeDs authentication](#) process. You only initialize transaction by redirecting the end user to the Paytool payment form and providing basic transaction metadata via the your backend.

The sequence diagram below illustrates the transaction initialization process step by step

```
@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
}
actor payer
participant "Customer front" as cfront
participant "Customer backend" as cback
participant "Paytool front" as pfront
participant "Paytool backend" as pback
payer->cfront: Pay with Paytool
cfront->cback: Payer choosed Paytool
cback->pback: Customer authorization + transaction metadata
pback->pback: Store transaction metadata + open payment session
cback<-pback: OK + transaction id + redirect url
cfront<-cback: OK + transaction id + redirect url
cfront->pfront: Redirect + transaction id
pfront->pback: Get transaction data + merchant payment methods
pfront<-pback: OK response
payer<-pfront: Display transaction + payment methods
@enduml

```

Server to server

An alternative path to carry out a transaction is a [Payment process via API](#) integration path. Paytool Frontend does not appear in this path. You make all requests directly from your server to the Paytool API methods we described in [technical integration guide](#).

The sequence diagram below illustrates the transaction initialization process step by step

```

@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
}
actor payer
participant "Customer front" as cfront
participant "Customer backend" as cback
participant "Paytool backend" as pback
payer->cfront: Pay with Paytool
cfront->cback: Payer choosed Paytool
cback->pback: Customer authorization + transaction metadata
pback->pback: Store transaction metadata + open payment session
cback<-pback: OK + transaction id
note right of cfront: At this point, the Customer decides how to present the transaction to the payer
cback->pback: Request proper payment method / executing 3D Secure
cback<-pback: Transaction result
note right of cfront: At this point, the Customer decides how to present the transaction result to the payer
@enduml

```

Payment methods

Verestro as a PSP provides several different payment methods in Paytool solution. You decide which payment method should be available in Paytool payment form. You decide this during the [onboarding](#) process but it is editable whenever necessary. The end user, seeing the available

payment methods, can choose the option that is most convenient for him. This chapter describes all payment methods supported by our application.

Payment methods in Paytool
Google Pay™
Apple Pay
Blik
Plain card number

Note: Each of the payment methods available from the Paytool frontend is also available with an API to API connection.

Note: It is required that you have created account in the Acquirer's system which will settle your transactions. Verestro Paytool solution has been implemented so that it is possible to process transactions with the participation of various Acquirers. If you require the settlement of the transaction by a new Acquirer – to which we are not yet integrated – there will be required new integration between Verestro and the new Acquirer. You should provide a specification of the Acquirer API.

Plain card number

The most common payment method available in our application is payment using plain card details. This payment method consists in providing a payment form in which the end user can enter his payment card details. The option to pay with plain card details is available at the button shown below:

image-1685428055821.png

After clicking on the payment option using plain card details, the end user is redirected to the specially prepared payment form. Now he is able to provide the appropriate card details and clicks the **Pay** button at the bottom of the screen. At this point, the card details are encrypted and sent to the Paytool backend which requests Acquirer to process transaction. At this moment [threeDs authentication](#) process may be required. The payment process using plain card details from the end user's perspective has been shown below:

image-1713956617464.png

The sequence diagram below illustrates the transaction with plain card details process step by step

```
@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
}
actor "End User" as user
participant "Paytool Frontend" as pfront
participant "Paytool Backend" as pback
participant "Customer Server" as custback
participant "Acquirer" as acq
user->pfront: User chooses payment by plain card details
user<-pfront: Redirect end user to the payment form
user->pfront: User filling data (CN, CVC, EXP)
pfront->pfront: Encrypt data
pfront->pback: Perform transaction providing encrypted payment request body
pback->pback: Decrypt encrypted payment request body
pback->acq: Order transaction
note left of acq: At this point 3D Secure process may occur
acq->pback: Response
pback->pfront: Transaction Result
pback-->pback: Store transaction result
pfront->pfront: Redirect end user (success/failure)
pback->custback: Send transaction postback to provided URL (optional - configurable)
note right of pback: Verestro can also send the notification after transaction to the end user(optional)
custback->pback: Get transaction status (optional)
pback->custback: Return transaction status
@enduml
```

Google Pay

Paytool as a registered hosted checkout in the Google Pay™ allows to perform a payment with the card from the Google Pay™ Wallet. Using our application your payer can choose the Google Pay™ as the payment method from the Verestro Paytool payment methods list. The option to pay with Google Pay™ card token is available at the button shown below:

[image.1685428002241.png](#)

After clicking on the Google Pay™ payment option, the Google Pay™ Wallet popup is displayed. The payer should select the appropriate card from the Google Pay™ Wallet which is then encrypted on the Google Pay™ side and transferred to our application. We will decrypt obtained data and transfer them to the Acquirer in the transaction request. At this moment [threeDs authentication](#) process may be required but it will be handled on our side. The payment process using Google Pay™ card payment token from the end user's perspective has been shown below:

[image.1689246683126.png](#)

The sequence diagram below illustrates the payment process using Google Pay™ card payment token step by step

```
@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
}
actor "End User" as user
participant "Paytool Frontend" as pfront
```

```
participant "Paytool Backend" as pback
participant "Google Pay" as gpay
participant "Customer Server" as custback
participant "Acquirer" as acq
user->pfront: End user chooses payment by Google Pay
pfront->gpay: Open end user Google Wallet
user<--gpay: Opens end user's Google Wallet
user-->gpay: Choose card from Google Wallet
gpay->gpay: Encrypt chosen card token data
pfront<-gpay: Provide encrypted card token data
pfront->pback: Order transaction
pfront-->pback: Provide encrypted card token data
pback->pback: Decrypt encrypted card token data
pback->acq: Order transaction
note left of acq: At this point 3D Secure process may occur
acq->pback: Response
pback->pfront: Transaction Result
pback-->pback: Store transaction result
pfront->pfront: Redirect end user (success/failure)
pback->custback: Send transaction postback to provided URL (optional - configurable)
note right of pback: Verestro can also send the notification after transaction to the end user (optional)
custback->pback: Get transaction status (optional)
pback->custback: Return transaction status
@enduml
```

Google Pay payment via API

Alternatively Verestro as a registered PSP in the Google Pay™ allows to perform a payment with card token generated by Google. This payment option provides a backend to backend oriented solution. In this payment model you must be [registered merchant in Google Pay™](#) to be able to get a card payment token from Google Pay™.

To use this option your API should be integrated with the `googlePayTransaction` method provided by Paytool API. By using this solution you need to display Google Pay™ Wallet in your application by your own. After selecting the Google Pay™ as the payment option a payer chooses proper card from Google Pay™ Wallet. After that action you get encrypted card payment token from Google Pay™. Card payment token is encrypted on the Google Pay™ side with Verestro's public key. You have to provide this encrypted token to Paytool API by using `googlePayTransaction` method. If you do this, we will handle transaction process on our side. At this moment 3D Secure process may be required and you need to handle it on your side using [threeDs authentication](#) methods provided by Paytool API.

The sequence diagram below illustrates the payment process using Google Pay™ card payment token API to API step by step

```
@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 "End user" as user
participant "Customer Application" as app
participant "Google Pay" as gp
participant "Paytool Backend" as tps
participant "Acquirer" as acq
note right of user: User wants to pay with Google Pay
user->>app: Pay with Google Pay and choose card
app->>gp: Requests for card token
gp->>gp: Encrypts card token with Verestro pub key
app<-gp: Returns encrypted card token
app->>tps: Requests token payment
tps->>tps: Decrypts card token
tps->>acq: Orders transaction
tps<-acq: Transaction status
note left of acq: At this point 3D Secure process may occur
app<-tps: Transaction status
user<-app: Transaction status
user<-tps: Sends email notification - optional
@enduml
```

Important: If you use Paytool with [Payment process via API](#) integration path, make sure you are a registered merchant in the Google Pay™ system. This does not apply to Customers using the [Redirect your payer](#) route.

Note: Google Pay™ provides a [Google Pay Web integration checklist](#) that will help you with the integration step by step. The documentation is available after whitelisting in Google Pay™ system. The whitelisting process is performed by Google Pay™ during the Customer's merchant account registration process.

Note: Google Pay™ provides [Google Pay Web Brand Guidelines](#) that presents branding requirements for web merchants registered in Google Pay™. You must meet these requirements so that you can allow his payers to pay via the Google Pay™ solution.

Apple Pay

Verestro as a PSP registered in the Apple Pay™ allows to perform a payment with card token generated by Apple. Using our application your payer can choose the Apple Pay™ as the payment method from the Paytool payment methods list. The option to pay with Apple Pay™ card token is available at the button shown below:

[image-1699524383207.png](#)

Warning! Apple Pay payment method is available in Safari web browser only.

After clicking on the Apple Pay™ payment option, the Apple Pay™ Wallet popup is displayed. End user should select the appropriate card from the Apple Pay™ Wallet popup which is then encrypted on the Apple Pay™ side and transferred to our application. We decrypt obtained data and transfer them to the Acquirer in the transaction request. At this moment [threeDs authentication](#) process may be required. The payment process using Apple Pay™ card payment token from the end user's perspective has been shown below:

[image-1699526152020-18.03.png](#)

The sequence diagram below illustrates the payment process using Apple Pay™ card payment token step by step

```
@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
}
actor "End User" as user
participant "Paytool Frontend" as pfront
participant "Paytool Backend" as pback
participant "Apple Pay" as gpay
participant "Customer Server" as custback
participant "Acquirer" as acq
user->pfront: End user chooses payment by Apple Pay
pfront->gpay: Open end user's Apple Pay Wallet
user<--gpay: Opens end user's Apple Pay Wallet
user-->gpay: Choose card from Apple Pay Wallet
note left of user: At this point Apple Pay may require user authorization
gpay->gpay: Encrypt chosen card token data
pfront<-gpay: Provide encrypted card token data
pfront->pback: Order transaction
pfront-->pback: Provide encrypted card token data
pback->pback: Decrypt encrypted card token data
pback->acq: Order transaction
note left of acq: At this point 3D Secure process may occur
acq->pback: Response
pback->pfront: Transaction Result
pback-->pback: Store transaction result
pfront->pfront: Redirect end user (success/failure)
pback->custback: Send transaction postback to provided URL (optional - configurable)
note right of pback: Verestro can also send the notification after transaction to the end user (optional)
custback->pback: Get transaction status (optional)
pback->custback: Return transaction status
```

@enduml

Note: At this moment Apple Pay biometric authorization may be required.

Note: Payment using the Apple Pay Wallet is only possible if the payer uses MacOS.

Blik

Warning! Implemenation is work in progress...

Paytool supports cardless payments using the Blik code. The end user can choose the Blik as the payment method from the Verestro Paytool payment methods list. The option to pay with Blik code is available at the button shown below:

image-1685523665104.png

After selecting this payment method, the end user must provide his e-mail address in the proper field in displayed popup and then confirm the willingness to make the transaction. Blik as an external service, in response opens a view that allows end user to enter the 6 digits code. This step of the process takes place outside the Verestro. The 6 digits Blik code is generated in the end user's bank application. The payment process using Blik code from the end user's perspective has been shown below:

image-1687851945379.png

image-1687852047098.png

The sequence diagram below illustrates the payment process using Blik step by step

```
@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
}
actor "End User" as user
participant "Paytool Frontend" as pfront
participant "Paytool Backend" as pback
participant "Acquirer" as acq
participant "Blik" as blik
participant "Customer Server" as custback
user->pfront: End user chooses payment by Blik (1)
pfront->pback: Requests for payment by Blik (2)
pback->acq: Orders Blik transaction (3)
acq->blik: Internal request to Blik system (4)
acq<-blik: Response from Blik system with HTML payment Base64 encoded form (5)
pback<-acq: Provides Blik's response (5.1)
pfront<-pback: Provides Blik's response (5.2)
pfront->pfront: Decode Blik HTML form (6)
user<-pfront: Displays Blik HTML form (7)
user->user: Enters Blik's 6 digits code (8)
user-->blik: After form submitting request to Blik system is triggered (9)
pfront<--blik: Redirects to Paytool Frontend (10)
user<--pfront: Shows progress bar because the user has to wait for transaction status (11)
acq<--blik: Webhook with information about Blik's transaction status (12)
pback<--acq: Webhook with information about Blik's transaction status (12.1)
pfront->pback: Get status of the Blik transaction (13)
pfront<-pback: Provides Blik transaction status (14)
user<-pfront: Displays transaction status (15)
custback<--pback: Send transaction postback to provided URL (16) (optional - configurable)
@enduml

```

ThreeDs authentication

Transactions using payment cards usually require additional authentication of the end user (card holder). In accordance with the European Union directive PSD2, the Paytool solution uses the threeDs protocol version 2.0 for the authentication process. The threeDs process consists in asking the card issuer whether additional authentication of the card holder is necessary or not. It depends on the bank's response whether the threeDs process will be necessary to perform. This is

determined by one of two statuses that the bank returns during the transaction process: `FRICITIONLESS` or `CHALLENGE`. The meaning of each status is described in the section [ThreeDs modes](#). We take responsibility for performing the threeDs authentication process. The sequence diagrams below illustrates the threeDs authentication process step by step separately for both cases: `FRICITIONLESS` or `CHALLENGE`.

ThreeDs modes

ThreeDs mode	Description
<code>FRICITIONLESS</code>	The response containing the <code>FRICITIONLESS</code> mode denotes that payment is successfully finnished. Additional 3DS authentication is not required.
<code>CHALLENGE</code>	The response containing this mode denotes that the card holder must be additionally authenticated. Along with the <code>CHALLENGE</code> mode, the Bank also returns an encoded HTML template, which is displayed to the end user in the Verestro Paytool Frontend, thus allowing the end user to perform 3DS authentication.

`FRICITIONLESS`

In this threeDs mode there is not any additional action from the end user side required.

The sequence diagram below illustrates the threeDs process for frictionless mode step by step

@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
}
actor "End User" as user
participant "Paytool Frontend" as pfront
participant "Paytool Backend" as pback
participant "Acquirer" as acq
participant "Bank" as bank
user->pfront: End user confirmed transaction (1)
note right of user: For example the end user choosed payment with plain card number (1.1)
pfront->pback: Provide transaction request to perform (2)
pback->acq: Provide transaction request to perform (3)
acq->bank: contact with end user's bank (4)
bank->acq: Return FRICTIONLESS status - transaction succeed (5)
acq->pback: Provide bank's response (6)
pback->pfront: Provide bank's response (6.1)
pfront->user: Display transaction status (7)
@enduml

```

CHALLENGE

In this threeDs mode the end user authentication is required. Authentication takes place on the threeDs bank template displayed by the Verestro Paytool application to the end user during the transaction process.

The sequence diagram below illustrates the threeDs process for challenge mode step by step

```

@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
}
actor "End User" as user
participant "Paytool Frontend" as pfront
participant "Paytool Backend" as pback
participant "Acquirer" as acq
participant "Bank" as bank
user->pfront: End user confirmed transaction (1)
note right of user: For example the end user choosed payment with plain card number (1.1)
pfront->pback: Provide transaction request to perform (2)
pback->acq: Provide transaction request to perform (3)
acq->bank: contact with end user's bank (4)
bank->acq: Return CHALLENGE status along with bank HTML template (5)
acq->pback: Provide bank's response (6)
pback->pfront: Provide bank's response (6.1)
pfront->pfront: Decode bank HTML template (7)
pfront->user: Display bank content to the end user (8)
user-->bank: Perform 3DS authentication (9)
user<--bank: Authentication succeed (10)
acq<-bank: Provide success status and authentication ID (11)
pback<-acq: Provide success status and authentication ID (11.1)
pfront<-pback: Provide success status and authentication ID (11.2)
pfront->pback: Finalize transaction and provide authentication ID (12)
pback->acq: Finalize transaction and provide authentication ID (12.1)
acq->bank: Finalize transaction (12.2)
bank->acq: Transaction succeed (13)
acq->pback: Provide transaction status (14)
pback->pfront: Provide transaction status (15)
pfront->user: Display transaction status (16)
@enduml

```

Transaction result

Depending on whether the transaction was successful or not, the end user will see the appropriate status on the screen. From the status screen, he is able to return to the store's website manually - by clicking the **Return to the shop** button or wait for an automatic redirection. At the same time, we will send you post-transaction notification containing the status and details of the transaction. Depending on the configuration of your merchant account, an e-mail notification may be sent to the end user. Transaction status views from the end user's perspective are presented below:

Notification after transaction

As mentioned in the [Transaction result](#) chapter, Paytool solution supports the sending of post-transaction notifications informing you about the status of a given transaction made in your context. This functionality is optional and it is up to you to decide whether you want to receive post-transaction notifications. This decision is made during the [onboarding](#) process and always can be edited. We recommend using this functionality, as this allow you to receive information on each of the transactions made using Verestro Paytool.

Important: In order for you to receive [notification after transaction](#), you must define your own endpoint to which notification will be directed. This is necessary so that we know where to send an HTTP POST request with transaction details. During the [onboarding](#) process, the you can provide a valid and accessible URL address that will serve as the endpoint for the post-transaction notifications.

Post-transaction notification example sent by Verestro Paytool to the Customer

```
{
  "transactionId": "string",
  "orderNumber": "9afds032dad",
  "transactionState": "FAILED",
  "amount": 0,
  "message": "string",
  "currency": "string",
  "externalTransactionId": "string",
  "transactionType": "ONE_TIME_PAYMENT",
  "cardDetails": {
    "cardProvider": "MASTERCARD",
    "cardNumber": "512485*****4875"
  },
  "failedDetails": {
    "code": "SYSTEM_ERROR",
    "message": "E0200: Transaction rejected, card is blocked"
  }
}
```

End user notifications

As mentioned in the [Transaction result](#) chapter, Paytool solution supports an e-mail notification sending. Such notification contains the status and details of the transaction. This message is purely informative so that the payer knows that a transaction has been made in the context of his chosen payment instrument. This functionality is optional and it is up to you to decide whether you us to send end users notifications after the transaction. This decision is made during the [onboarding](#) process and always can be edited.

Note: The appearance of the e-mail notification is configurable.

Example e-mail notification after successful transaction	Example e-mail notification after failed transaction
	

Onboarding

This chapter is intended to present you the requirements that will allow you to use the Paytool solution in your application. We have presented here what information is necessary to provide so that you can join the Paytool program and so that we can properly create the required merchant account for you in our system.

Business onboarding

Tip: Please remember to inform us on which environment you want us to configure an account for you. Verestro Paytool offers two environments: **TEST** and **PROD**. For more information about application environments please visit [How to integrate chapter](#).

To start using Paytool payment widget you need to go through a few on-boarding steps:

1. Please contact our sales - salesteam@verestro.com
2. Please respond to some introduction question that will let us prepare proposal for you.
3. You will receive offer for eCommerce payments.
4. If you accept the offer you will be asked to provide some company documents required for the AML verification process.
5. After succesful AML process you will receive contract with our partnering acquiring institution.
6. You will be asked to provide account numbers for settlements and the data necessary to configure Paytool for you (see below)
7. And finally you will enable Paytool widget or API on your website or mobile app to enable payments.

Technical onboarding

Tip: The onboarding process takes place mainly on Verestro side. However, in order for all possibilities offered by Verestro Paytool to be available, the you must provide some information needed to correctly configure your merchant account in our system. Configuration includes below presented information:

Configuration parameter	Description
-------------------------	-------------

Customer name	Basically, it's the name of the your company. The following parameter will be displayed to the end user in Verestro Paytool form.
Customer website URL	Basically, it's the website URL of the your online application. The following parameter will be displayed to the end user in Verestro Paytool form. Such addresses are required to be added to our whitelist.
Transaction session life time	The time given in seconds after which the initialization of the transaction will expire and processing of the payment in its context will no more be possible.
Redirect URL's	Those are the addresses to which end user will be redirected after a successful or unsuccessful transaction performed in Verestro Paytool. These parameters can be static (added to the your merchant account configuration) or sent in the order initiating the transaction.
Postback URL	Using this address we will send you information about the transaction made by a given end user. This parameter is not required if you does not want to receive notifications regarding the transaction.
Autodeposit property	This parameter controls whether a transactions made by merchant will be automatically deposited. This parameters can be static (added to the your merchant account configuration) or sent during the transaction initialization.
Supported payment methods	You should determine which of available payment methods you want to enable in Verestro Paytool. Defaultly all payment methods are enabled. This parameter can be changed anytime. Available payment methods in Paytool: Google Pay, Apple Pay, Blik, plain card number.
Acquirer's Merchant Id (MID)	<p>It is an identifier of your account in the Acquirer system. This parameter is obligatory to perform transaction. If your company supports more then 1 MID, please let us know which one should be treat as default one.</p> <div> <p>Note: Default MID will be selected if you will not provide any of your MID's in the transaction initialization request.</p> <p>Tip: We are suggesting to use Fenige as this is our partner acquiring institution and we are fully integrated with this Acquirer</p> </div>
Acquirer credentials	Login and password that should be used to authorize your payment in the system of a given Acquirer in the context of a given terminal.
(Optional) Customer's employee e-mail address	This e-mail will be used as login to Admin Panel.

(Optional) Customer's logo and/or picture	These are image that could be, for example, your company logo. They will be posted as requested in the Verestro Paytool webview. This point is optional. If you do not provide such pictures, the Verestro Paytool will be shown with it's default appearance.
---	--

Important! If you require transactions to be processed with the participation of a Acquirer to which Verestro is not integrated, then there must be performed a new integration. You should provide the given Acquirer API documentation that Verestro will use during the integration.

Returned data

After creating an account for the Customer, Verestro returns all necessary data which allow to use the solution. Such data includes:

Configuration parameter	Description
Basic Authorization	This is map of the key: login and value: password. Basic Authorization is required to use API methods in Verestro Paytool. Basic Authorization is issued to the Customer after onboarding process.
Gateway Id	<p>This is a constant and unique value that defines the PSP in the Google Pay™ system. When making a call to get a card token, the Customer transfers this value to Google Pay™ in the request. Verestro is defined by Gateway Id with <code>verestro</code> value.</p> <p>Note: This only applies to Customers who make Google Pay payments by connecting directly to the Paytool API. Customers redirecting their payers to the Paytool frontend form should ignore this information.</p>
Gateway Merchant Id	<p>This is a unique Customer identifier assigned by Verestro during the onboarding process. This identifier is in the form of a <code>UUID</code>. Verestro understands and uses this to verify that the message was for the Customer that made the request. Customer passes it to Google Pay™. More information about the Gateway Merchant Id can be found in Google Pay™ documentation.</p> <p>Note: This only applies to Customers who make Google Pay payments by connecting directly to the Paytool API. Customers redirecting their payers to the Paytool frontend form should ignore this information.</p>

Register in Google Pay™

Note: This only applies to Customers who make Google Pay payments by connecting directly to the Paytool API. If you are using [Redirect your payer](#) integration path you can ignore this section.

To use the Token Payment Service solution, it is necessary for the Customer to be registered as merchant in the Google Pay™ system. An unregistered Customer will not be able to get the card payment token from Google Pay. To register merchant account in Google Pay™ visit [Google Pay for Business quick start guide](#) or contact Google Pay™ support. After completing registration as a merchant, the Customer will receive an access to Google Pay™ documentation enabling technical integration.

Google Pay Web integration checklist	A checklist presenting the Google Pay integration requirements that must be met by the Customer integrating web application
Google Pay Web developer documentation	Technical documentation describing web integration with the Google Pay solution
Google Pay Web Brand Guidelines	Branding requirements that must be met by the Customer's web application to be able to use the Google Pay solution

How to integrate

This chapter provides the instruction of the integration with the solution and with it's methods. Prior to using this solution is you have to proceed [onboarding](#) process and you require created account in the Verestro Paytool system.

Note: To create an account in the Verestro Paytool system please contact with support.

Environment Test API base URL

`https://paytool-api.verestro.dev`

Environment Production API base URL

`https://paytool-api.verestro.com`

Tip: Below are presented sequence diagrams describing both ways of using the Paytool solution. In both cases depending on your Customer account configuration in Paytool, our backend will send e-mail notification to the payer.

Integration method consisting in redirecting the payer to Paytool web view

```
@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
}
actor payer
participant "Customer front" as cfront
participant "Customer backend" as cback
participant "Paytool front" as pfront
participant "Paytool backend" as pback
payer->cfront: Pay with Paytool
cfront->cback: Payer choosed Paytool
cback->pback: Customer authorization + transaction metadata
pback->pback: Store transaction metadata + open payment session
cback<-pback: OK + transaction id
cfront<-cback: OK + transaction id
cfront->pfront: Redirect + transaction id
pfront->pback: Get transaction data + merchant payment methods
pfront<-pback: OK response
payer<-pfront: Display transaction + payment methods
@enduml

```

Integration method consisting in continuing process via API calls

```

@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
}
actor payer
participant "Customer front" as cfront
participant "Customer backend" as cback
participant "Paytool front" as pfront
participant "Paytool backend" as pback
payer->cfront: Pay with Paytool
cfront->cback: Payer choosed Paytool
cback->pback: Customer authorization + transaction metadata
pback->pback: Store transaction metadata + open payment session
cback<-pback: OK + transaction id
note right of cfront: At this point, the Customer decides how to present the transaction to the payer
cback->pback: Request proper payment method / executing 3D Secure
cback<-pback: Transaction result
note right of cfront: At this point, the Customer decides how to present the transaction result to the payer
@enduml
```

Note: If you choose to redirect the payer to our Paytool web view, the following payment methods will be available. The payment methods available in the Paytool payment form are customizable according to your requirements.

Methods in API

Our solution provides API method allowing you to order and process transaction by [creating payment session](#) and send transaction metadata to our system. In order to opening new payment session you receive `transactionId` in response which is unique identifier of the created session. Every ordered transaction has his own payment session. Using `transactionId` you can whether redirect your payer to our Paytool web view or continue the transaction process using other API methods provided by our solution. This section describes all API methods in Paytool solution.

Tip: There are description of every Paytool API method. To see example request bodies please visit complete API documentation is [available here](#) in SwaggerUI format. Please visit to see our methods.

Tip: Detailed information about what is included in the transaction metadata is shown in the [transactionInitialization](#) method.

Transaction initialization

POST /external-api/transactions

You need to call this API method to create new payment session in the Verestro Paytool system. You request this method with every new transaction order. After successful initialization, the generated `transactionId` in the form of UUID will be returned in the response. The `transactionId` parameter is required for the further parts of the payment process, regardless of whether you redirect your payer to our Verestro web view or you continue the process by requesting other of our API methods. This is important that you saves the `transactionId` parameter in cache for example - by this parameter, our system will map proper session with a given transaction.

Note: You must use this method whenever you start the Paytool payment process, regardless of whether in the next step you redirect the payer to our Paytool frontend application or continue the process by connecting to other Paytool methods from your API.

Google Pay

POST /external-api/transactions/google-pay

This method allows you to pay using the received card payment token from Google Pay. Please note that in the case of Google Pay payments using the API method, you must first obtain a Google payment token yourself. This method can only be executed after creating a payment session using [transactionInitialization](#) method.

Note: Use this method only when you continue the payment process by communicating with Paytool via API requests.

Important! 3D Secure authentication may be required when requesting a server-to-server Google Pay transaction. To perform 3D Secure authentication, please refer to the methods in the [threeDs authentication](#) chapter. Does not apply to Customers redirecting the payer to the Paytool frontend app.

Important! You need to be registered merchant in Google Pay system if you want to use this method. You will find the detailed information about registration in Google Pay in the [Onboarding chapter](#).

Apple Pay

Important! Apple Pay is available only in the Redirect your payer implementation model. Implementation of the server to server communication version of the method is work in progress...

Warning! Apple Pay payment method is available in Safari web browser only.

Get transaction details

```
GET /external-api/transactions/{transactionId}
```

This method allows you to get the current status of the transaction. Using this you can obtain information whether a given transaction has already been made or has been rejected for some reason. Based on the response from this method, you are able to determine whether you can release the goods to your payer or not. Method is basing on the `transactionId` parameter which is provided in the [transactionInitialization](#) method response.

Tip: You can call this method regardless of which integration way you perform (redirecting the payer to Paytool or continuing the payment process via server to server communication) as [getTransactionDetails](#) method will inform you about the current transaction status.

Tip: Integration with the [getTransactionDetails](#) method described is optional but we hardly recommend using this method as without it, you will not receive any information about the ordered transaction. You can also integrate [transactionPostback](#) method described below.

Transaction postback

```
POST https://merchant-url-address.com/postback
```

This method allows our Paytool backend to send a webhook notification containing information about the transaction made by the end user. You must define his own endpoint to which postbacks will be directed by our server - this is necessary so that the Verestro system knows where to send an HTTP POST request with conversion data. The URL must be a valid and accessible endpoint that can receive HTTP requests. When the conversion event occurs, the server will send an HTTP POST request to the provided URL with the conversion data in the request body.

Note: You should provide postback endpoint during the [onboarding](#) process.

Tip: You should implement this method regardless of which integration option you use (redirecting the payer to Paytool or continuing the payment process via server to server communication) as [transactionPostback](#) method will inform you about the current transaction status.

Tip: Integration with the [transactionPostback](#) method is optional but we hardly recommend using this method as without it, you will not receive any information about the ordered transaction. You can also integrate [getTransactionDetails](#) method described above.

Payment with threeDs authentication

This section describes how you should integrate with the threeDs authentication process provided by Verestro Paytool API. The threeDs authentication process consists of 3 methods that will need to be invoked or omitted depending on the status. More detailed information can be found in the description of each of the 3 methods that make up the entire threeDs authentication process. After successfully threeDs authentication process the payment is executed automatically.

Note: Use below threeDs authentication methods only when you continue the payment process by communicating with Paytool via API requests.

Initialize threeDs process

```
POST /external-api/transactions/3ds
```

This is the method you always call to start the threeDs authentication process. Depending on the returned `THREE_DS_MODE` parameter, you will need to call one of the other two methods - [continue3ds](#) for `THREE_DS_MODE=THREE_DS_METHOD` or [finalize3ds](#) for `THREE_DS_MODE=CHALLENGE`. If you receive the `FRICTIONLESS` mode in the `THREE_DS_MODE` field, no further action is required. The cardholder's bank allowed the transaction to be performed without additional authentication. The payment has been completed successfully.

Note: Use this method only when you continue the payment process by communicating with Paytool via API requests.

Continue threeDs process

POST /external-api/transactions/3ds/continue

This is a method by which threeDs authentication process can be continued. This method should be executed after receiving `THREE_DS_MODE=THREE_DS_METHOD`.

Note: Use this method only when you continue the payment process by communicating with Paytool via API requests.

Finalize threeDs process

POST /external-api/transactions/3ds/finalize

This is the method you must use if we want to end the transaction flow after the successful `CHALLENGE` process. This is the last step in the threeDs authentication process.

Note: Use this method only when you continue the payment process by communicating with Paytool via API requests.

Deposit

POST /external-api/transactions/{transactionId}/deposit

This method allows you to deposit frozen funds from the payer's account and to collect the proper amount for the user purchase. It can only be used when the transaction status is `WAITING_FOR_DEPOSIT` and thus requires a deposit to complete the payment process. The deposit amount must be equal to or less than the amount declared in the [transactionInitialization](#) process.

Note: If a transaction with the `WAITING_FOR_DEPOSIT` status will not be deposited with this method, the frozen funds will be returned to the payer's account.

Note: This method must be used regardless of whether the funds have been frozen via the redirect or API payment path.

Transaction refund

POST /external-api/transactions/refund

This method allows you to refund funds to the payer's account. After calling this method, the funds for the purchase will be refunded to the payer. The method can only be called if the transaction is cleared. This method is optional.

Tip: You can call this method regardless of which integration option you use (redirecting the payer to Paytool or continuing the payment process via server to server communication) as `transactionRefund` is one of the two ways to return funds to the payer. The second way is to make a return from the Admin Panel. If you want to access the Admin Panel, please specify this during the [onboarding](#) process.

Get public key

GET /external-api/public-key

This method returns Paytool public key which can be used to encrypt sensitive data which some of our other methods require, for example [initialize3ds.sendersEncryptedData](#).

Get resources

GET /external-api/links

This method returns url address to the most actual terms of use and RODO info. Terms can be get in two languages depending on the Accept-Language header value: `pl,en-us;q=0.7`. If header value is `null`, the default resources language will be set to english.

Get card details

This method returns detailed information about the card submitted in the request. The data returned includes the consumer type, funding type and its origin (country). We distinguish two consumer types: `CONSUMER` and `BUSSINESS`. We distinguish three funding types: `DEBIT`, `CREDIT` and `PREPAID`. If it is not possible to determine the consumer type, it is marked as `UNKNOWN`.

Opening Paytool in iframe

This section describes the steps you need to take if you want the Paytool form to be displayed as an iframe on your website instead of redirecting your payer to Verestro Paytool website.

Tip: This section applies to customers who use the Redirect your payer implementation model. If you use a different integration model with Paytool, you can ignore this chapter.

Initialization

To start working with the SDK, add the following script to your website:

```
<script src="https://paytool.verestro.com/v1/paytool.js"></script>
```

The script creates a new property named `Paytool` in the `window` object, giving a global access to it's API.

Tip: Complete Paytool Client SDK documentation is [available here](#). Please visit to see our methods.

Note: Cross-Origin Resource Sharing -> Your domain must be whitelisted for the script to load properly on your website.
See the full integration guidelines for more.

Testing: You can also check your integration on a staging environment to avoid charging payments.

To do this, simply replace the script's src with `https://paytool.verestro.dev/v1/paytool.js`.

Note: A separate enrollment is required, your production credentials won't work.
The staging environment, even though meant to be stable, often changes and temporal issues might occur.

Test Guide

This section provides guidelines for testing various payment methods supported in Paytool solution, including plain card number, google pay, apple pay, and BLIK on environment **TEST**.

Tip: Check it now. Verestro Paytool demo application is available [here](#).

Plain card number

This method involves processing payments using credit or debit cards. In order to test it you will need to generate test cards because of possibility of getting card blocked due to anti-fraud policies on the acquirer side. Payment with plain card number use case is described [here](#).

Test cards

Card type	Card BIN
VISA	511796XXXXXXXXXX
Mastercard	545313XXXXXXXXXX

There's need of generation PAN numbers from these BIN's, in order to do such so you can use [Credit Card Generator](#).

Note: CVV may be randomized, expiration date must not be expired.

Testing various transaction paths

As described in the chapter on [3DS authentication](#), there are various ThreeDs paths, In order to be able to test different flows, you need to manipulate the transaction amount to get into the selected transaction path:

ThreeDs mode	Amount
--------------	--------

FRICTIONLESS	100 - 1000
CHALLENGE	1001 - 3000

Note: Amount is a value determining amount of cash transferred in one hundredth of the currency [1,00 USD = 100].

Tip: Select an amount between 3001-8000 to test failed transaction.

Google Pay

In order to be able to test google-pay, you need to add a tokenized card to your google account. You need to register with the [google test card suite group](#) to do so. Follow these instructions from google's documentation. After registration in [google test card suite group](#) you are ready to test Google Pay token payment. After logging into your test account and selecting Google Pay, you should have a choice of several test cards added to your account which you can use to make a payment in Paytool TEST environment. Google Pay payment use case is described [here](#).

Note: To be able to test all flows of ThreeDs in Google Pay follow the description about [testing various transaction paths](#)

Apple Pay

This method involves processing payments using tokenized credit or debit cards. You need to create Apple sandbox tester account. Follow [apple documentation](#) to do so. After registration and logging in into tester account you are ready to test Apple Pay token payment. There is a list of [apple-pay valid payment cards](#). Payment with Apple Pay use case is described [here](#).

Blik

Important! Implemenation is work in progress...