

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. We recommend using **Redirect you payer** integration path.

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 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 cback: 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 cback: At this point, the Customer decides how to present the transaction result to the payer
@enduml
```

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 your 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 you want to end the transaction flow after the successful `CHALLENGE` process. This is the last step in the threeDs authentication process. By calling this method you will receive the final state of the transaction.

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.

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

POST /external-api/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.