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.

Tip: API to API integration path documentation is also available here in SwaggerUI format.

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: Detailed information about what is included in the transaction metadata is shown in the `transactionInitialization` method.

Transaction initialization

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.

POST /external-api/transactions

Headers	
Key	**Value**
`Content-Type`	`application/json`
`Basic-Authorization`	`Basic dXNlcm5hbWU6cGFzc3dvcmQ=`

Example request body in JSON format

```
{
  "currencyCode": "PLN",
  "amount": 1000,
  "receiverAmount": 1000,
  "description": "Preinit for trx",
  "merchantUrl": "https://merchanturl.verestro.com",
  "notificationUrl": "https://custom-merchant-notificationurl.com",
  "orderNumber": "fgugbb4gd665564",
  "formLanguage": "pl",
  "redirectUrl": {
    "successUrl": "https://successurl.verestro.com/paytool/redirection/success",
    "failureUrl": "https://failureurl.verestro.com/paytool/redirection/success"
  },
  "sender": {
    "firstName": "John",
    "lastName": "Doe",
    "address": {
      "countryCode": "PL",
      "city": "Warszawa",
      "postalCode": "03-672",
      "street": "Promienna",
      "houseNumber": "10"
    }
  },
  "transactionConfigurationId": "dc387e4e-885e-455f-9dd9-c7cad002b476",
  "autoDeposit": true,
  "validityTime": 2
}
```

Parameter	Type	Description
`currencyCode`	string **required**	Currency for transaction (in accordance with 3-digit ISO-4217), example: USD
`amount`	number **required**	Field determining amount of cash transferred in one hundredth of the currency. [1,00 USD = 100]
`orderNumber`	string **required**	Declarative number of order that is just purchased by cardholder, set by merchant, should be unique.
`receiverAmount`	number	Information field only. Field determine receiving amount of cash transferred in one hundredth of the currency. [1 USD = 100]
`description`	string	Description of the transaction.
`formLanguage`	string	Language of transaction process in web browser, use only lowercase.

`notificationUrl`	string	URL address of merchant web system to which the notification will be sent after the transaction is completed. When notificationUrl is null, the value from the merchant configuration will be used.
`redirectUrl`	object	Object containing URL addresses where payer should be redirected after transaction finalization depending on transaction status. Adress may be the same in both fields: `successUrl` and `failureUrl`
`successUrl`	string	URL of merchant web service to forward after successful transaction flow.
`failureUrl`	string	URL of merchant web service to forward after failed transaction flow
`sender`	object	Object containing datailed payer's data.
`sender.firstName`	string	Payers's first name.
`sender.lastName`	string	Payers's last name.
`sender.address`	object	Payer's address.
`sender.address.countryCode`	string	Two character ISO 3166-1 alpha-2 code of country. For example: PL
`sender.address.city`	string	Payer's city.
`sender.address.postalCode`	string	Payer's postal code.
`sender.address.street`	string	Payer's street.
`sender.address.houseNumber`	string	Payer's house number.
`transactionConfigurationId`	string	A parameter pointing to a given terminal in the merchant's configuration. If it is not passed, the transaction will be executed in the context of the default merchant terminal.

`autoDeposit`	boolean	Flag specifying whether the funds will be immediately withdrawn from the payer's account after the transaction. If the value is `false`, the funds will be frozen on the payer's account and will wait for the `deposit` to be made. If the `deposit` is not made, the funds will be returned to the payer.
`validityTime`	number	Through this field, we can specify expiration time for transaction in minutes. If this parameter has not been provided, it will be taken from Merchant configuration.

Example response body in JSON format - 200 - OK

```
{
  "transactionId": "3fa85f64-5717-4562-b3fc-2c963f66afa6"
}
```

Parameter	Type	Description
`transactionId`	string($uuid)	The identifier of transaction assigned by Paytool system after transaction preinitialization occurrence.

Possible errors

Errors that may occur when attempting to initialize transaction:

404 - Not found

```
{
  "uuid": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "traceId": "8fc17b6c-6995-4d33-a4f5-93e99b8af2b2",
  "errorType": "MERCHANT_NOT_EXISTS",
  "merchantStatus": "Merchant entity with typed UUID: 3fa85f64-5717-4562-b3fc-
2c963f66afa6 not exists in database"
}
```

Parameter	Type	Description
`uuid`	string	The identifier of transaction assigned by Paytool system after transaction preinitialization occurrence.
`traceId`	string	Identifier of the occured error.
`errorType`	string	Type of the occured error: • MERCHANT_NOT_EXISTS
`message`	string	Description of the occured error.

409 - Conflict

```
{
  "uuid": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "traceId": "8fc17b6c-6995-4d33-a4f5-93e99b8af2b2",
  "errorType": "ENTITY_EXISTS",
  "merchantStatus": "Pre initialization with typed UUID: 3fa85f64-5717-4562-b3fc-
2c963f66afa6 exists in database"
}
```

Parameter	Type	Description
`uuid`	string	The identifier of transaction assigned by Paytool system after transaction preinitialization occurrence.
`traceId`	string	Identifier of the occured error.
`errorType`	string	Type of the occured error: • ENTITY_EXISTS
`message`	string	Description of the occured error.

500 - Internal server error

```
{
  "date": "2023-09-26 13:51:40.869246",
  "traceId": "8fc17b6c-6995-4d33-a4f5-93e99b8af2b2",
  "errorType": "INTERNAL_SERVER_ERROR",
  "message": "Internal server error occured."
}
```

Parameter	Type	Description
`date`	string($date-time)	Timestamp of the occured error.
`traceId`	string	Identifier of the occured error.
`errorType`	string	Type of the occured error: • INTERNAL_SERVER_ERROR
`message`	string	Description of the occured error.

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.

POST /external-api/transactions/google-pay

Headers	
Key	**Value**
`Content-Type`	`application/json`
`Basic-Authorization`	`Basic dXNlcm5hbWU6cGFzc3dvcmQ=`

Example request body in JSON format

```
{
  "transactionId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "token": {
        "signature":
"MEYCIQDypF11YBH1coN5ZUadvPzkqvlJzV2TgtCZZcK9jpY14QIhAKnNp4iPAfUNMgMyew/RCjIpFhph7He8kw
gilSmQ+iCL",
        "intermediateSigningKey": {
            "signedKey":
"{\"keyValue\":\"MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEFTC7wcoKdNZaC8nMT+okwTe3X+0BdS2PHQ
g8wMBj+hFyXCZZ03D6Q0WC6Hj2Opc77pZidc1dLQpb4CyvYiB7rQ\\u003d\\u003d\",\"keyExpiration\":
\"1692395417939\"}",
            "signatures": [

"MEUCIElbkp4lt6lQ6JFT9aZygQX9K2nYb4pEdqQtPjFTde5MAiEAqcQ0OQZ+8xNYY2AWXz+RgycyAtAr3ZiRkm
E3NYCRGJ4="
            ]
        },
        "protocolVersion": "ECv2",
        "signedMessage":
"{\"encryptedMessage\":\"Kbe6TjKavqbk9lsjaJB20hufV+l6Vpm3j+8djXUO9gNalFHUDFF3CS/f1KvzPV
irRHZGFft+XlzrlabKhtbvfKuN9Hotp5h5csPngGUMabYAx+m5MU8X+Y98v7kueozLVuxMjPmUonWGmQxO0l7pq
dVfOJPO2GckAyxRd9EBHoaC2CJjo1IAJzFdjuEdjMS4GoHU15n8iM8fJVrmeagmaZ4H8zcdo9WnuQbQ00HIHncz
dyGTLeglTq8dY3lSE+WjrirqvwRfXbRSsfXmp+QQtvLAR9ZW6ula2xwp5Nz14atYGKJFk9OOcF+cdoGrzlyU0ru
WEVhUeBEJzRiiSdL2xk9SK3hL+itpmxEFKL27Cibo2VHmo8MpgTxgftGTDwCnENgq6ZRf6deazHe8I/Un8QTBKO
5JxLyvvUKVLWqaz1nqaCfDcRK4SM3yRejg0aJPV/vy/joygyHS+UNjw71dy9/UcdVxJ8oQlG/34NUl9tCMYn5N9
aSd8qPJcnf5PzJQpgUXZwoGgBT5Wi7zdSrTF4RH6wMODH5IjLkSbD86s6RtjP2z+WZb9n2YwTdtlojgyD/BNBv3
GA\\u003d\\u003d\",\"ephemeralPublicKey\":\"BLX9MA6SWxduIZyxoNHVaj0L0H4WtJfX10MTQe3bii/
vAIjrB3Kfa6OiWw+BX+QwFRyfwEENJjOiN0EDqIxLQk8\\u003d\",\"tag\":\"Jxdb9OuKwqn6baKiAnpJpB+
FumFXHbHho3xnacK5o6A\\u003d\"}"
    },
  "sender": {
      "firstName" : "John",
      "lastName" : "Doe",
      "email" : "test@verestro.com"

  "language":{
      "language" : "PL"
  },
    "threeDsData": {
    "cavv": "sdf",
    "eci": "01",
    "authenticationStatus": "Y",
    "transactionXId": "9734301f-4025-4e91-81b2-f74e64f40037"
  }
}
```

Parameter	Type	Description
`transactionId`	string($uuid) **required**	The identifier of transaction assigned by Paytool system after transaction preinitialization occurrence.
`token`	object **required**	Object containing all data of the encrypted card payment token obtained by Customer from Google Pay.
`token.signature`	string	Google Pay token signature.
`token.protocolVersion`	string	Version of the protocol.
`token.signedMessage`	string	Message encrypted by Verestro public key.
`token.intermediateSigningKey`	object	Object containing key and signature.

`token.intermediateSigningKey.signedKey`	string	Key used to verification of the signature.
`token.intermediateSigningKey.signatures`	list<string>	Verification signature.
`sender`	object **required**	Object containing payer's data.
`sender.firstName`	string	Payer's first name.
`sender.lastName`	string	Payer's last name.
`sender.email`	string	Email address of the payer.
`threeDsData`	object	Outside 3ds data authorization that have been send to our API.
`threeDsData.cavv`	string	This property is determined by the Access Control Server. This property will be valid if the TransactionStatus is "Y" or "A". The value may be used to provide proof of authentication.
`threeDsData.eci`	string	This property is determined by the Access Control Server. This property contains the two digit Electronic Commerce Indicator (ECI) value, which is to be submitted in a credit card authorization message. This value indicates to the processor that the customer data in the authorization message has been authenticated. The data contained within this property is only valid if the TransactionStatus is "Y" or "A".
`threeDsData.authenticationStatus`	string	Status of the 3D Secure authentication.
`threeDsData.transactionXId`	string	External transaction Id assigned by the Acquirer.

Example response body in JSON format - 200 - OK

```
{
  "transactionId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "transactionState": "COMPLETED"
}
```

Parameter	Type	Description
`transactionId`	string($uuid)	The identifier of transaction assigned by Paytool system after transaction preinitialization occurrence.
`transactionState`	string	TransactionState is a main parameter that indicates state of transaction: • 'COMPLETED' - we are sure that transaction passed correctly. • 'FAILURE' - transaction surely not passed with paytool internal error or with processing error from acquirer. • 'IN_PROGRESS' - when transaction is in this state we are need to continue process - wait for challenge etc.

Possible errors

Errors that may occur when attempting to order Google Pay transaction:

404 - Not found

```
{
  "uuid": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "traceId": "8fc17b6c-6995-4d33-a4f5-93e99b8af2b2",
  "errorType": "MERCHANT_NOT_EXISTS",
  "merchantStatus": "Merchant entity with typed UUID: 3fa85f64-5717-4562-b3fc-
2c963f66afa6 not exists in database"
}
```

Parameter	Type	Description
`uuid`	string	The identifier of transaction assigned by Paytool system after transaction preinitialization occurrence.
`traceId`	string	Identifier of the occured error.
`errorType`	string	Type of the occured error: • MERCHANT_NOT_EXISTS
`message`	string	Description of the occured error.

500 - Internal server error

```
{
  "date": "2023-09-26 13:51:40.869246",
  "traceId": "8fc17b6c-6995-4d33-a4f5-93e99b8af2b2",
  "errorType": "THREE_DS_REQUIRED",
  "message": "Google Pay token payment with PAN_ONLY authMethod parameter - 3DS is required"
}
```

Parameter	Type	Description
`date`	string($date-time)	Timestamp of the occured error.
`traceId`	string	Identifier of the occured error.
`errorType`	string	Type of the occured error: • INTERNAL_SERVER_ERROR • THREE_DS_REQUIRED • EXTERNAL_PAYMENT_SERVICE_ERROR
`message`	string	Description of the occured error.

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

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.

GET /transactions/details/{transactionId}

Headers	
Key	**Value**
`Content-Type`	`application/json`
`Basic-Authorization`	`Basic dXNlcm5hbWU6cGFzc3dvcmQ=`

Query parameter	**Value**

`transactionId`	`<UUID>`

Example response body in JSON format - 200 - OK

```
{
  "transactionId": "5755a7a4-490f-4a83-ad25-535fbe9d7443",
  "orderNumber": "fgugbb4gd665564",
  "transactionState": "SUCCESS",
  "externalId": "5755a7a4-490f-4a83-ad25-535fbe9d7443",
  "merchantName": "Merchant",
  "amount": 2599,
  "currencyCode": "PLN",
  "description": "Transaction description",
  "transactionType": "ONE_TIME_PAYMENT",
  "cardDetails": {
    "cardProvider": "MASTERCARD",
    "cardNumber": "512485******4875"
  },
  "failedDetails": {
    "code": "SYSTEM_ERROR",
    "message": "E0200: Transaction rejected, card is blocked"
  },
  "merchantUuid": "1361d514-943b-11ee-b9d1-0242ac120002"
}
```

Parameter	Type	Description
`transactionId`	string($uuid)	The identifier of transaction assigned by Paytool system after transaction initialization occurrence.
`orderNumber`	string	The identifier of transaction assigned by Customer. Should be unique.

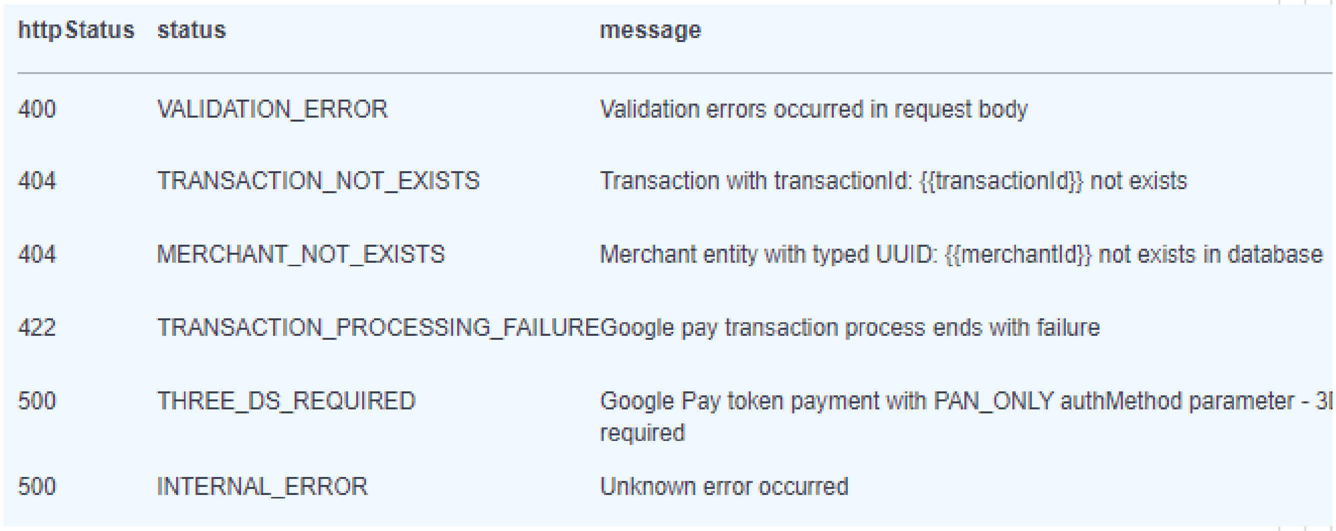
`transactionState`	string	Transaction result. There are three statuses: • COMPLETED - transaction finished with success • FAILED - transaction finished with failure • IN_PROGRESS - transaction not yet finished
`externalId`	string	The identifier of the transaction processed by Acquirer.
`merchantName`	string	The name of the Customer's merchant account in Paytool system.
`amount`	number	The amount of transaction in pennies, for instance: 1000 = 10.00$
`currencyCode`	string	ISO 4217 Alpha-2 currency code for example: PLN, EUR, USD
`description`	string	The description of the transaction provided by Customer.
`transactionType`	string	The type of method used for make transaction. Possible transaction types: • ONE_TIME_PAYMENT - payment with bare card number • GOOGLE_PAY - payment using Google Pay Wallet • APPLE_PAY - payment using Apple Pay Wallet • BLIK - payment using Blik code
`cardDetails`	object	The object containing card data.
`cardDetails.cardProvider`	string	The type of the card used to transaction. Possible card providers: • MASTERCARD • VISA • MAESTRO

`cardDetails.cardNumber`	string	Masked card number. BIN and last four digits of the card number has been shown.
`failedDetails`	object	Object containing more detailed information about transaction's fail reason. This object remains empty if transaction was successfully completed.
`failedDetails.code`	string	Transaction's fail code/status.
`failedDetails.message`	string	Transaction's fail message.
`merchantUuid`	string($uuid)	UUID of the Customer's merchant account in Paytool system. Required for passing in next request.

Possible errors

Errors that may occur when attempting to retrieve transaction details:

The below table defines the status codes that this method may return



httpStatus	status	message
400	VALIDATION_ERROR	Validation errors occurred in request body
404	TRANSACTION_NOT_EXISTS	Transaction with transactionId: {{transactionId}} not exists
404	MERCHANT_NOT_EXISTS	Merchant entity with typed UUID: {{merchantId}} not exists in database
422	TRANSACTION_PROCESSING_FAILURE	Google pay transaction process ends with failure
500	THREE_DS_REQUIRED	Google Pay token payment with PAN_ONLY authMethod parameter - 3 required
500	INTERNAL_ERROR	Unknown error occurred

401 - Unauthorized

```
{
  "timestamp": "2023-03-29T19:16:01.288+00:00",
  "status": 401,
  "error": "Unauthorized",
  "path": "/api/v1/transactions/9609a08e-cd80-4e6e-8664-f1e6b2f2dc50"
}
```

Parameter	Type	Description
`timestamp`	string($date-time)	Timestamp of the occured error.
`status`	numeric	Numeric HTTP code of the occured error.
`error`	string	Description of HTTP status of the occured error.
`path`	string	Requested endpoint's path

Warning: The structure of this error is different from other errors

404 - Not found

```
{
  "date": "2023-09-26 13:51:40.869246",
  "traceId": "9f4c5101-6d78-4c80-9e79-3b2060c227ad",
  "errorType": "TRANSACTION_NOT_EXISTS",
  "message": "Transaction with transactionId: a39cc08a-5458-4b05-868d-3c484c9f601e
not exists"
}
```

Parameter	Type	Description
`date`	string($date-time)	Timestamp of the occured error.
`traceId`	string	Identifier of the occured error.

`errorType`	string	Enum type of the occured error.
`message`	string	Description of the occured error.

500 - Internal server error

```
{
  "date": "2023-09-26 13:51:40.869246",
  "traceId": "8fc17b6c-6995-4d33-a4f5-93e99b8af2b2",
  "errorType": "EXTERNAL_PAYMENT_SERVICE_ERROR",
  "message": "Initialize transaction process ends with some critical failure caused
by error in request to external service"
}
```

Parameter	Type	Description
`date`	string($date-time)	Timestamp of the occured error.
`traceId`	string	Identifier of the occured error.
`errorType`	string	Enum type of the occured error.
`message`	string	Description of the occured error.

Transaction 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.

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.

Note: You should provide postback endpoint during the onboarding process.

POST https://customer-notification-endpoint.com/postback

Headers	
Key	**Value**
`Content-Type`	`application/json`

Example request body in JSON format

```
{
  "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"
  }
}
```

Parameter	Type	Description
`transactionId`	string	Customer's external, custom transaction identifier. We are recommending to use random UUID.
`orderNumber`	string	The identifier of transaction assigned by Customer. Should be unique.
`transactionState`	string	Transaction result. There are three statuses: • COMPLETED - transaction finished with success • FAILED - transaction finished with failure • IN_PROGRESS - transaction not yet finished
`amount`	number	The amount of transaction in pennies, for instance: 1000 = 10.00$
`message`	string	The message about the transaction.
`currency`	string	ISO 4217 Alpha-2 currency code for example: PLN, EUR, USD
`externalTransactionId`	string	ID of the transaction received and forwarded from Acquirer. It is optional parameter.
`transactionType`	string	Payment option chosen by the end user. There are four types of the transaction: • ONE_TIME_PAYMENT • GOOGLE_PAY • APPLE_PAY • BLIK
`cardDetails`	object	Object containing card data
`cardDetails.cardProvider`	string	Type of the card. There are two types of the card: MASTERCARD or VISA.
`cardDetails.cardNumber`	string	Masked card number with visible BIN and last four digits.

`failedDetails`	object	Object containing more detailed information about transaction's fail reason. This object remains empty if transaction was successfully completed.
`failedDetails.code`	string	Transaction's fail code/status.
`failedDetails.message`	string	Transaction's fail message.

Tip: You need to response with the status `HTTP 200 - OK` after the positively received postback request. Otherwise Paytool API will retry postback sending.

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

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.

POST /external-api/transactions/3ds

Headers	
Key	**Value**
`Content-Type`	`application/json`
`Basic-Authorization`	`Basic dXNlcm5hbWU6cGFzc3dvcmQ=`

Example request body in JSON format

```
{
  "transactionId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "senderDataEncrypted":
"eyJ0eXAiOiJKT1NFIiwiZW5jIjoiQTI1NkdDTSIsImlhdCI6MiwiYWxnIjoiUlNBLU9BRVAtMjU2Iiwia2lkIj
oiNjdlNTVlM2QyZDc1YTNiNzg2NzI1NjkxOTBiZTdlMWY5Njc0ZDZlMiJ9.TkwHHD8w7rm8c7A5i2D1lUfl65LH
ESBmEJldZ93u5UwWPLankMTGcEiMEeoOFK4hCpFy83HwZy4SyeY4hcyBXNtHq0NCslv9rwToIty0FubriBE1MBy
5Ed1j2NUyrnTFyrX9s0qhZVRc_COs8CGVvArLM7_WDBRAqvN8Td7xxsG6Ms3vEzesDRDBm85kzOTvJW6UzRLCTw
6CAY9UT83DlWTWbpLXet17BPRvOtUtXgfhbWh5FbZHxet00BVGGLTVTdRYV2-_WQvzsQa-
eIqaNO_agjGqOnpZ_SpP6TRbY1rd9GFraSjd4uKt4C3LZdhkUOkI_fJMwJgbFfzEgziPYg.khjzevA2hqjQ1OYh
.iZT6PgvHiwxzFAw-OwIUyBIuoKz_Dhf88ZEBT5c1bAYEFenpXB1Z_aaKez7Z3C-X_yPfYTCGFhFN9ImtH8TX-
3qbMQ7C1CEbZIJoV_PyJ6b7LEw0CFo7P9guQ5tv3foUJI6Sz1rYC-48GJGYybs0DyiD5FrEifF79cfXC8yzbf-
O6MFt-
5ppSSXcl5jsjX4Rk996nkBT4YO2hTR80LY5NoK6_akvgqJqgO1TXqfOv6D1qJkmxYaWiLmIQlJtSXmQtbEgVkIR
g3hvpVFAY0EOl_uevRZSIBPNxjXP4GjjfE5yunSReOvdBMMktgifxoLYfKYu8cxXpoGzcHTTiMkLjYArEM2CAGQ
RA_ftf2CQ5jhCcs6Z-jhW.CEMzAbRS1FC8bhQxlPPOBg",
  "typeOfTransaction": "ONE_TIME_PAYMENT",
  "challengeNotificationUrl": "https://challengeNotificationUrl.com/198271c2",
  "browserData": {
    "javaEnabledVal": "ENABLED",
    "javaScriptEnabled": true,
    "windowSize": "S_250X400",
    "language": "PL",
    "screenHeight": 200,
    "screenWidth": 200,
    "timeZone": "60",
    "screenColorDepth": 24
    }
  }
```

Parameter	Type	Description
`transactionId`	string($uuid) **required**	The identifier of transaction assigned by Paytool system after transaction preinitialization occurrence.

Sender's encrypted data. Use `getPublicKey` method to obtain Paytool public key

Example ONE_TIME_PAYMENT decrypted sender's data in JSON format

```
{
    "email":
"okonomiyaki@verestro.
com",
    "homePhone": "1-
1234567899",
    "mobilePhone": "1-
1234567899",
    "workPhone": "48-
1234567899",
    "firstName":
"Madara",
    "lastName":
"Uchicha",
    "card": {
      "cardNumber" :
"5453131785534417",

"cardExpirationDate" :
"12/24",
      "cvc2": "123"
    }
}
```

Example GOOGLE_PAY decrypted sender's data in JSON format

```
{
    "email" :
"testmail@verestro.com
",
    "homePhone" :
"123456789",
    "mobilePhone" :
"123456789",
    "workPhone" :
```

`transactionType`	string **required**	Type of the transaction. This field defines which payment method was chosen by sender to pay for his goods. Examples: • ONE_TIME_PAYMENT - payment using plain card data • GOOGLE_PAY - payment using card payment token obtained from Google Pay • APPLE_PAY - payment using card payment token obtained from Apple Pay • BLIK - payment using 6 digits Blik code
`challengeNotificationUrl`	string **required**	This property specifies the URL to which the final challenge response will be send.
`browserData`	object **required**	Sender's browser details. **Example browser details in JSON format** `"browserData": { "javaEnabledVal": "ENABLED", "javaScriptEnabled": true, "windowSize": "S_250X400", "language": "PL", "screenHeight": 200, "screenWidth": 200, "timeZone": "60", "screenColorDepth": 24 }`

`browserData.challengeWindowSize`	string	This field indicates the dimensions of the challenge window that has been displayed to the cardholder. The ACS shall reply with content that is formatted to appropriately render in this window to provide the best possible user experience. Preconfigured sizes are width x height in pixels of the window displayed in the cardholder browser. Possible values are: S_250X400 S_390X400 S_500X600 S_600X400 FULL_SCREEN. This value is included in the Challenge Request Message (CReq).
`browserData.language`	string	This field contains the cardholder's browser language alphanumeric with accordance to ISO 3166-1 alpha-2. For example polish language will be presented as "PL".
`browserData.screenHeight`	number	This field contains the total height of the cardholder's screen in pixels.
`browserData.screenWidth`	number	This field contains the total width of the cardholder's screen in pixels.
`browserData.screenColorDepth`	number	This field contains a value representing the bit depth of the color palette, in bits per pixel, for displaying images. Obtained from Cardholder browser using the screen.colorDepth property. Values accepted: 1 = 1 bit, 4 = 4 bits, 8 = 8 bits, 15 = 15 bits, 16 = 16 bits, 24 = 24 bits, 32 = 32 bits, 48 = 48 bits.
`browserData.timeZone`	string	This field contains the difference between UTC time and the cardholder's browser local time in minutes.
`browserData.javaScriptEnabled`	boolean	This field contains a value representing the ability of the cardholder's browser to execute JavaScript. Enumerated values: ENABLED, DISABLED.

`browserData.javaEnabledVal`	string	This field contains a value representing the ability of the cardholder’s browser to execute Java. Enumerated values: NOT_PRESENT, ENABLED, DISABLED. Required if `browserData.javaScriptEnabled` is set to `ENABLED`.

Example response body in JSON format - 200 - OK

```
{
  "threeDsFlow": "THREE_DS_METHOD",
  "threeDsMethodData":
"eyJ0aHJlZURTTWV0aG9kTm90aWZpY2F0aW9uVVJMIjoiaHR0cHM6Ly9wYXl0b29sLnBsL2FwaS92MS8zZHMvbm
90aWZpY2F0aW9uLzAyYzcyNWYzLTM4NWUtNGM2OC1iY2NiLTczM2I1NGYxODliNCIsInRocmVlRFNTZXJ2ZXJUc
mFuc0lEIjoiY2U4OGQ3MTEtNWQ3MS00Yzk3LWIyYzgtMmIyOWE3OWFjNjU2In0",
  "threeDsMethodUrl": "https://mpi-
staging.verestro.pl/v2/authentication/notification/acs-mock"
}
```

Parameter	Type	Description
`threeDsFlow`	string	Indicates whether a transaction qualifies as an authenticated transaction or account verification. Possible values are: • THREE_DS_METHOD - Possible additional request to ACS, in which we should include the threeDsMethodData field • FRICTIONLESS - The transaction was released for processing without 3ds challenge because the issuer allowed it. Payment process is finnished • CHALLENGE - Challenge required - login to the bank on the user's side for authentication

`threeDsMethodData`	string	Encoded data used for request to ACS.
`threeDsMethodUrl`	string	ACS endpoint for hidden request. If endpoint is not present then request is not required.

Possible errors

Errors that may occur when attempting to initialize transaction:

401 - Unauthorized

```
{
  "timestamp": "2023-03-29T19:16:01.288+00:00",
  "status": 401,
  "error": "Unauthorized",
  "path": "/api/v1/transactions/9609a08e-cd80-4e6e-8664-f1e6b2f2dc50"
}
```

Parameter	Type	Description
`timestamp`	string($date-time)	Timestamp of the occured error.
`status`	numeric	Numeric HTTP code of the occured error.
`error`	string	Description of HTTP status of the occured error.
`path`	string	Requested endpoint's path

Warning: The structure of this error is different from other errors

404 - Not found

```
{
  "uuid": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "traceId": "8fc17b6c-6995-4d33-a4f5-93e99b8af2b2",
  "errorType": "MERCHANT_NOT_EXISTS",
  "merchantStatus": "Merchant entity with typed UUID: 3fa85f64-5717-4562-b3fc-
2c963f66afa6 not exists in database"
}
```

Parameter	Type	Description
`uuid`	string	The identifier of transaction assigned by Paytool system after transaction preinitialization occurrence.
`traceId`	string	Identifier of the occured error.
`errorType`	string	Type of the occured error: • MERCHANT_NOT_EXISTS
`message`	string	Description of the occured error.

422 - Unprocessable Entity

```
{
  "uuid": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "traceId": "8fc17b6c-6995-4d33-a4f5-93e99b8af2b2",
  "errorType": "TRANSACTION_PROCESSED",
  "merchantStatus": "Transaction with transactionId: q732ecw1-dcg2-4614-94e6-
b931gb46r772 just exists"
}
```

Parameter	Type	Description
`date`	string	The identifier of transaction assigned by Paytool system after transaction preinitialization occurrence.
`traceId`	string	Identifier of the occured error.

`errorType`	string	Type of the occured error: • UNPROCESSABLE_ENTITY
`message`	string	Description of the occured error.

500 - Internal server error

```
{
  "date": "2023-09-26 13:51:40.869246",
  "traceId": "8fc17b6c-6995-4d33-a4f5-93e99b8af2b2",
  "errorType": "INTERNAL_SERVER_ERROR",
  "message": "Internal server error occured."
}
```

Parameter	Type	Description
`date`	string($date-time)	Timestamp of the occured error.
`traceId`	string	Identifier of the occured error.
`errorType`	string	Type of the occured error: • INTERNAL_SERVER_ERROR
`message`	string	Description of the occured error.

Continue threeDs process

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.

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

Headers	
Key	**Value**
`Content-Type`	`application/json`
`Basic-Authorization`	`Basic dXNlcm5hbWU6cGFzc3dvcmQ=`

Example request body in JSON format

```
{
  "transactionId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "senderDataEncrypted":
"eyJ0eXAiOiJKT1NFIiwiZW5jIjoiQTI1NkdDTSIsImlhdCI6MiwiYWxnIjoiUlNBLU9BRVAtMjU2Iiwia2lkIj
oiNjdlNTVlM2QyZDc1YTNiNzg2NzI1NjkxOTBiZTdlMWY5Njc0ZDZlMiJ9.TkwHHD8w7rm8c7A5i2D1lUfl65LH
ESBmEJldZ93u5UwWPLankMTGcEiMEeoOFK4hCpFy83HwZy4SyeY4hcyBXNtHq0NCslv9rwToIty0FubriBE1MBy
5Ed1j2NUyrnTFyrX9s0qhZVRc_COs8CGVvArLM7_WDBRAqvN8Td7xxsG6Ms3vEzesDRDBm85kzOTvJW6UzRLCTw
6CAY9UT83DlWTWbpLXet17BPRvOtUtXgfhbWh5FbZHxet00BVGGLTVTdRYV2-_WQvzsQa-
eIqaNO_agjGqOnpZ_SpP6TRbY1rd9GFraSjd4uKt4C3LZdhkUOkI_fJMwJgbFfzEgziPYg.khjzevA2hqjQ1OYh
.iZT6PgvHiwxzFAw-OwIUyBIuoKz_Dhf88ZEBT5c1bAYEFenpXB1Z_aaKez7Z3C-X_yPfYTCGFhFN9ImtH8TX-
3qbMQ7C1CEbZIJoV_PyJ6b7LEw0CFo7P9guQ5tv3foUJI6Sz1rYC-48GJGYybs0DyiD5FrEifF79cfXC8yzbf-
O6MFt-
5ppSSXcl5jsjX4Rk996nkBT4YO2hTR80LY5NoK6_akvgqJqgO1TXqfOv6D1qJkmxYaWiLmIQlJtSXmQtbEgVkIR
g3hvpVFAY0EOl_uevRZSIBPNxjXP4GjjfE5yunSReOvdBMMktgifxoLYfKYu8cxXpoGzcHTTiMkLjYArEM2CAGQ
RA_ftf2CQ5jhCcs6Z-jhW.CEMzAbRS1FC8bhQxlPPOBg",
  "typeOfTransaction": "ONE_TIME_PAYMENT"
}
```

Parameter	**Type**	**Description**
`transactionId`	string($uuid) **required**	The identifier of transaction assigned by Paytool system after transaction preinitialization occurrence.

`senderDataEncrypted`	string **required**	Sender's encrypted data. **Example decrypted sender's data in JSON format** `{ "email": "okonomiyaki@verestro.com", "homePhone": "1-1234567899", "mobilePhone": "1-1234567899", "workPhone": "48-1234567899", "firstName": "Madara", "lastName": "Uchicha", "card": { "cardNumber" : "5453131785534417", "cardExpirationDate" : "12/24", "cvc2": "123" } }`

`transactionType`	string **required**	Type of the transaction. This field defines which payment method was chosen by sender to pay for his goods. Examples: • ONE_TIME_PAYMENT - payment using plain card data • GOOGLE_PAY - payment using card payment token obtained from Google Pay • APPLE_PAY - payment using card payment token obtained from Apple Pay

Example response body in JSON format - 200 - OK

```
{
  "threeDsFlow": "CHALLENGE",
  "acsUrl": "acsurl.com",
  "cReq": "c29tZSBjcmVx",
  "challengeHtmlFormBase64":
"PGh0bWw+PFNDUklQVCBMQU5HVUFHRT0iSmF2YXNjcmlwdCI+ZnVuY3Rpb24gT25Mb2FkRXZlbnQoKXsgZG9jdW
1lbnQuZG93bmxvYWRGb3JtLnN1Ym1pdCgpOyB9PC9TQ1JJUFQ+PGJvZHkgT25Mb2FkPSJPbkxvYWRFdmVudCgpO
yI+PGZvcm0gbmFtZT0iZG93bmxvYWRGb3JtIiBhY3Rpb249Imh0dHBzOi8vamF2YS1kZXZlbC5mZW5pZ2UucGw6
ODE4MS9mZW5pZ2UtbXBpIiBtZXRob2Q9IlBPU1QiPjxJTlBVVCB0eXBlPSJoaWRkZW4iIG5hbWU9IlBhUmVxIiB
2YWx1ZT0iZUp4bGtsdHZ3akFNaGY4SzRwM20wcVJVbFluRTRHSFZ4TVF1VDN1TFdndUsybERTRnNHL244TmxiRn
FrU0Q2T2UrcDhEbnh1UGVMeUE0dkJvNEVWZHAzZDRLZ3FaMk01bFdwc1lEMS94NE9CSS9xdTJqc2pJaDVKWUhkS
lgvaGlhMTF2d0JhSHAvelZhQ1ZFb29EZEpEVG84NlhodENhMEJiQnJBcHh0MEhUb1N0VWdzSXVDWWorNDNwOU5J
aEpnZHdHRHI4MjI3OXN1WTJ4bmozWlM0aEZyRVEydHJjcW9yYk5VcVpnVmRZV3VCeGFxZ1QzYVdnOGg2c2o5Vkp
VbVg4dzMvL1l5ajFlN3R4bXdVQUdsN2RGSUxsSWVTejRTT2xNNjB5bXdTeDVzRTlveWk2K1hVYXg1eERuZDlKcU
NOdnhwZmhWMEZvNStwNEFRZTNURm1jeGp1dDVkQVo3YXZhUGVEWEg5aVlFOUdsODhCN3BGVDlqaUFESXNSU3hwU
UtsS3ROUnFtc2lVa04yS2dtTkY1Q1RuK21JWkJMQmd3MjdqSkVDWFNWUDA1d1Y4QTEvYXFXUT0iPjxpbnB1dCB0
eXBlPSJoaWRkZW4iIG5hbWU9IlRlcm1VcmwiIHZhbHVlPSJodHRwczovL3VybCI+PGlucHV0IHR5cGU9ImhpZGR
lbiIgbmFtZT0iTUQiIHZhbHVlPSJiV1E9Ij42Zvcm0+PC9ib2R5PjwvaHRtbD4=",
  "threeDsSessionData": "ZGY0NDJhODQtOGJkZS00MmQ3LWExMTUtMzUyNjNlZGY1MGFk"
}
```

Parameter	Type	Description

`threeDsFlow`	string	Indicates whether a transaction qualifies as an authenticated transaction or account verification. Possible values are: <ul><li>FRICTIONLESS - The transaction was released for processing without 3ds challenge because the issuer allowed it. Payment process is finnished</li><li>CHALLENGE - Challenge required - login to the bank on the user's side for authentication</li></ul>
`threeDsSessionData`	string	Encoded data used for request to ACS.
`transactionStatus3ds`	string	ACS endpoint for hidden request. If endpoint is not present then request is not required.
`acsUrl`	string	The url to which the 3DS parameters will be sent.
`cReq`	string	Challenge Request Message - additional information to support the 3DS authentication process and to carry the authentication data from the Cardholder.
`challengeHtmlFormBase64`	string	Base64 encoded data contains HTML form to display for user in order to login bank account.

Possible errors

Errors that may occur when attempting to initialize transaction:

401 - Unauthorized

```
{
  "timestamp": "2023-03-29T19:16:01.288+00:00",
  "status": 401,
  "error": "Unauthorized",
  "path": "/api/v1/transactions/9609a08e-cd80-4e6e-8664-f1e6b2f2dc50"
}
```

Parameter	Type	Description
`timestamp`	string($date-time)	Timestamp of the occured error.
`status`	numeric	Numeric HTTP code of the occured error.
`error`	string	Description of HTTP status of the occured error.
`path`	string	Requested endpoint's path

Warning: The structure of this error is different from other errors

404 - Not found

```
{
  "uuid": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "traceId": "8fc17b6c-6995-4d33-a4f5-93e99b8af2b2",
  "errorType": "MERCHANT_NOT_EXISTS",
  "merchantStatus": "Merchant entity with typed UUID: 3fa85f64-5717-4562-b3fc-
2c963f66afa6 not exists in database"
}
```

Parameter	Type	Description
`uuid`	string	The identifier of transaction assigned by Paytool system after transaction preinitialization occurrence.
`traceId`	string	Identifier of the occured error.
`errorType`	string	Type of the occured error: • MERCHANT_NOT_EXISTS

`message`	string	Description of the occured error.

422 - Unprocessable Entity

```
{
  "uuid": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "traceId": "8fc17b6c-6995-4d33-a4f5-93e99b8af2b2",
  "errorType": "TRANSACTION_PROCESSED",
  "merchantStatus": "Transaction with transactionId: q732ecw1-dcg2-4614-94e6-
b931gb46r772 just exists"
}
```

Parameter	Type	Description
`date`	string	The identifier of transaction assigned by Paytool system after transaction preinitialization occurrence.
`traceId`	string	Identifier of the occured error.
`errorType`	string	Type of the occured error: • UNPROCESSABLE_ENTITY
`message`	string	Description of the occured error.

500 - Internal server error

```
{
  "date": "2023-09-26 13:51:40.869246",
  "traceId": "8fc17b6c-6995-4d33-a4f5-93e99b8af2b2",
  "errorType": "INTERNAL_SERVER_ERROR",
  "message": "Internal server error occured."
}
```

Parameter	Type	Description

`date`	string($date-time)	Timestamp of the occured error.
`traceId`	string	Identifier of the occured error.
`errorType`	string	Type of the occured error: • INTERNAL_SERVER_ERROR
`message`	string	Description of the occured error.

Finalize threeDs process

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.

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

Headers	
Key	**Value**
`Content-Type`	`application/json`
`Basic-Authorization`	`Basic dXNlcm5hbWU6cGFzc3dvcmQ=`

Example request body in JSON format

```
{
  "transactionId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "senderDataEncrypted":
"eyJ0eXAiOiJKT1NFIiwiZW5jIjoiQTI1NkdDTSIsImlhdCI6MiwiYWxnIjoiUlNBLU9BRVAtMjU2Iiwia2lkIj
oiNjdlNTVlM2QyZDc1YTNiNzg2NzI1NjkxOTBiZTdlMWY5Njc0ZDZlMiJ9.TkwHHD8w7rm8c7A5i2D1lUfl65LH
ESBmEJldZ93u5UwWPLankMTGcEiMEeoOFK4hCpFy83HwZy4SyeY4hcyBXNtHq0NCslv9rwToIty0FubriBE1MBy
5Ed1j2NUyrnTFyrX9s0qhZVRc_COs8CGVvArLM7_WDBRAqvN8Td7xxsG6Ms3vEzesDRDBm85kzOTvJW6UzRLCTw
6CAY9UT83DlWTWbpLXet17BPRvOtUtXgfhbWh5FbZHxet00BVGGLTVTdRYV2-_WQvzsQa-
eIqaNO_agjGqOnpZ_SpP6TRbY1rd9GFraSjd4uKt4C3LZdhkUOkI_fJMwJgbFfzEgziPYg.khjzevA2hqjQ1OYh
.iZT6PgvHiwxzFAw-OwIUyBIuoKz_Dhf88ZEBT5c1bAYEFenpXB1Z_aaKez7Z3C-X_yPfYTCGFhFN9ImtH8TX-
3qbMQ7C1CEbZIJoV_PyJ6b7LEw0CFo7P9guQ5tv3foUJI6Sz1rYC-48GJGYybs0DyiD5FrEifF79cfXC8yzbf-
O6MFt-
5ppSSXcl5jsjX4Rk996nkBT4YO2hTR80LY5NoK6_akvgqJqgO1TXqfOv6D1qJkmxYaWiLmIQlJtSXmQtbEgVkIR
g3hvpVFAY0EOl_uevRZSIBPNxjXP4GjjfE5yunSReOvdBMMktgifxoLYfKYu8cxXpoGzcHTTiMkLjYArEM2CAGQ
RA_ftf2CQ5jhCcs6Z-jhW.CEMzAbRS1FC8bhQxlPPOBg",
  "typeOfTransaction": "ONE_TIME_PAYMENT",
  "cRes": "eJxlkltvwjAMhf8K4p3m0qRUlYnE4GHVxMQuT3uLWguK2lDSFsG/n8NlbFqk"
}
```

Parameter	Type	Description
`transactionId`	string($uuid) **required**	The identifier of transaction assigned by Paytool system after transaction preinitialization occurrence.

`senderDataEncrypted`	string **required**	Sender's encrypted data. **Example decrypted sender's data in JSON format**
		```
		{
		    "email": "okonomiyaki@verestro.com",
		    "homePhone": "1-1234567899",
		    "mobilePhone": "1-1234567899",
		    "workPhone": "48-1234567899",
		    "firstName": "Madara",
		    "lastName": "Uchicha",
		    "card": {
		      "cardNumber" : "5453131785534417",
		      "cardExpirationDate" : "12/24",
		      "cvc2": "123"
		    }
		}
		```

`transactionType`	string **required**	Type of the transaction. This field defines which payment method was chosen by sender to pay for his goods. Examples: • ONE_TIME_PAYMENT - payment using plain card data • GOOGLE_PAY - payment using card payment token obtained from Google Pay • APPLE_PAY - payment using card payment token obtained from Apple Pay
`cRes`	string	Payer's authentication response.

Example response body in JSON format - 200 - OK

```
{
  "transactionId": "6f4c5101-6d78-4c80-9e79-3b2060c221ad",
  "approvalCode": "CODE_00",
  "externalId": "9f4c5101-6d78-4c80-9e79-3b2060c227ad",
  "transactionStatus": "APPROVED",
  "transactionStatus3ds": "Y"
}
```

Parameter	Type	Description
`transactionId`	string	
`approvalCode`	string	
`externalId`	string	
`transactionStatus`	string	
`transactionStatus3ds`	string	

Possible errors

Errors that may occur when attempting to initialize transaction:

401 - Unauthorized

```
{
  "timestamp": "2023-03-29T19:16:01.288+00:00",
  "status": 401,
  "error": "Unauthorized",
  "path": "/api/v1/transactions/9609a08e-cd80-4e6e-8664-f1e6b2f2dc50"
}
```

Parameter	Type	Description
`timestamp`	string($date-time)	Timestamp of the occured error.
`status`	numeric	Numeric HTTP code of the occured error.
`error`	string	Description of HTTP status of the occured error.
`path`	string	Requested endpoint's path

Warning: The structure of this error is different from other errors

404 - Not found

```
{
  "uuid": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "traceId": "8fc17b6c-6995-4d33-a4f5-93e99b8af2b2",
  "errorType": "MERCHANT_NOT_EXISTS",
  "merchantStatus": "Merchant entity with typed UUID: 3fa85f64-5717-4562-b3fc-
2c963f66afa6 not exists in database"
}
```

Parameter	Type	Description

`uuid`	string	The identifier of transaction assigned by Paytool system after transaction preinitialization occurrence.
`traceId`	string	Identifier of the occured error.
`errorType`	string	Type of the occured error: • MERCHANT_NOT_EXISTS
`message`	string	Description of the occured error.

422 - Unprocessable Entity

```
{
  "uuid": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "traceId": "8fc17b6c-6995-4d33-a4f5-93e99b8af2b2",
  "errorType": "TRANSACTION_PROCESSED",
  "merchantStatus": "Transaction with transactionId: q732ecw1-dcg2-4614-94e6-
b931gb46r772 just exists"
}
```

Parameter	Type	Description
`date`	string	The identifier of transaction assigned by Paytool system after transaction preinitialization occurrence.
`traceId`	string	Identifier of the occured error.
`errorType`	string	Type of the occured error: • UNPROCESSABLE_ENTITY
`message`	string	Description of the occured error.

500 - Internal server error

```
{
  "date": "2023-09-26 13:51:40.869246",
  "traceId": "8fc17b6c-6995-4d33-a4f5-93e99b8af2b2",
  "errorType": "INTERNAL_SERVER_ERROR",
  "message": "Internal server error occured."
}
```

Parameter	Type	Description
`date`	string($date-time)	Timestamp of the occured error.
`traceId`	string	Identifier of the occured error.
`errorType`	string	Type of the occured error: • INTERNAL_SERVER_ERROR
`message`	string	Description of the occured error.

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.

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

Headers	
Key	**Value**
`Content-Type`	`application/json`
`Basic-Authorization`	`Basic dXNlcm5hbWU6cGFzc3dvcmQ=`

Query parameter	Value
`transactionId`	`<UUID>`

Example request body in JSON format

```
{
  "amount": 100
}
```

Parameter	Type	Description
`amount`	number **required**	The amount of the transaction to be deducted from the payer's account. It cannot be higher than the amount declared when initiating the transaction.

Example response body in JSON format - 200 - OK

```
{
  "transactionId": "3fa85f64-5717-4562-b3fc-2c963f66afa6"
}
```

Parameter	Type	Description
`transactionId`	string	Identifier of the transaction assigned during the `transactionInitialization` process.

Possible errors

Errors that may occur when attempting to initialize transaction:

401 - Unauthorized

```
{
  "timestamp": "2023-03-29T19:16:01.288+00:00",
  "status": 401,
  "error": "Unauthorized",
  "path": "/api/v1/transactions/9609a08e-cd80-4e6e-8664-f1e6b2f2dc50"
}
```

Parameter	Type	Description
`timestamp`	string($date-time)	Timestamp of the occured error.
`status`	numeric	Numeric HTTP code of the occured error.
`error`	string	Description of HTTP status of the occured error.
`path`	string	Requested endpoint's path

Warning: The structure of this error is different from other errors

404 - Not found

```
{
  "uuid": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "traceId": "8fc17b6c-6995-4d33-a4f5-93e99b8af2b2",
  "errorType": "MERCHANT_NOT_EXISTS",
  "merchantStatus": "Merchant entity with typed UUID: 3fa85f64-5717-4562-b3fc-
2c963f66afa6 not exists in database"
}
```

Parameter	Type	Description
`uuid`	string	The identifier of transaction assigned by Paytool system after transaction preinitialization occurrence.
`traceId`	string	Identifier of the occured error.
`errorType`	string	Type of the occured error: • MERCHANT_NOT_EXISTS
`message`	string	Description of the occured error.

422 - Unprocessable Entity

```
{
  "uuid": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "traceId": "8fc17b6c-6995-4d33-a4f5-93e99b8af2b2",
  "errorType": "TRANSACTION_PROCESSED",
  "merchantStatus": "Transaction with transactionId: q732ecw1-dcg2-4614-94e6-
b931gb46r772 just exists"
}
```

Parameter	Type	Description
`date`	string	The identifier of transaction assigned by Paytool system after transaction preinitialization occurrence.
`traceId`	string	Identifier of the occured error.
`errorType`	string	Type of the occured error: • UNPROCESSABLE_ENTITY
`message`	string	Description of the occured error.

500 - Internal server error

```
{
  "date": "2023-09-26 13:51:40.869246",
  "traceId": "8fc17b6c-6995-4d33-a4f5-93e99b8af2b2",
  "errorType": "INTERNAL_SERVER_ERROR",
  "message": "Internal server error occured."
}
```

Parameter	Type	Description
`date`	string($date-time)	Timestamp of the occured error.
`traceId`	string	Identifier of the occured error.
`errorType`	string	Type of the occured error: • INTERNAL_SERVER_ERROR
`message`	string	Description of the occured error.

Transaction 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.

POST /external-api/transactions/refund

Headers	
Key	**Value**
`Content-Type`	`application/json`
`Basic-Authorization`	`Basic dXNlcm5hbWU6cGFzc3dvcmQ=`

Example request body in JSON format

```
{
  "transactionId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "amountToRefund": 150
}
```

Parameter	Type	Description
`transactionId`	string($uuid) **required**	The identifier of transaction assigned by Paytool system after transaction preinitialization occurrence.
`amountToRefund`	number	Field determining amount to refund of cash transferred in one hundredth of the currency. Amount must be equal to the amount declared in the PreInitialization step. [1 USD = 100]

Example response body in JSON format - 200 - OK

```
{
  "transactionId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "transactionState": "COMPLETED"
}
```

Parameter	Type	Description
`transactionId`	string($uuid)	The identifier of transaction assigned by Paytool system after transaction preinitialization occurrence.

Possible errors

Errors that may occur when attempting to initialize transaction:

401 - Unauthorized

```
{
  "timestamp": "2023-03-29T19:16:01.288+00:00",
  "status": 401,
  "error": "Unauthorized",
  "path": "/api/v1/transactions/9609a08e-cd80-4e6e-8664-f1e6b2f2dc50"
}
```

Parameter	Type	Description
`timestamp`	string($date-time)	Timestamp of the occured error.
`status`	numeric	Numeric HTTP code of the occured error.
`error`	string	Description of HTTP status of the occured error.
`path`	string	Requested endpoint's path

Warning: The structure of this error is different from other errors

404 - Not found

```
{
  "uuid": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "traceId": "8fc17b6c-6995-4d33-a4f5-93e99b8af2b2",
  "errorType": "MERCHANT_NOT_EXISTS",
  "merchantStatus": "Merchant entity with typed UUID: 3fa85f64-5717-4562-b3fc-
2c963f66afa6 not exists in database"
}
```

Parameter	Type	Description
`uuid`	string	The identifier of transaction assigned by Paytool system after transaction preinitialization occurrence.
`traceId`	string	Identifier of the occured error.
`errorType`	string	Type of the occured error: • MERCHANT_NOT_EXISTS
`message`	string	Description of the occured error.

500 - Internal server error

```
{
  "date": "2023-09-26 13:51:40.869246",
  "traceId": "8fc17b6c-6995-4d33-a4f5-93e99b8af2b2",
  "errorType": "INTERNAL_SERVER_ERROR",
  "message": "Internal server error occured."
}
```

Parameter	Type	Description
`date`	string($date-time)	Timestamp of the occured error.
`traceId`	string	Identifier of the occured error.
`errorType`	string	Type of the occured error: • INTERNAL_SERVER_ERROR
`message`	string	Description of the occured error.

Get public key

This method returns Patool public key which can be used to encrypt sensitive data which some of our other methods require, for example `initialize3ds.sendersEncryptedData`.

GET /external-api/public-key

Headers	
Key	**Value**
`Content-Type`	`application/json`
`Basic-Authorization`	`Basic dXNlcm5hbWU6cGFzc3dvcmQ=`

Example response body in JSON format - 200 - OK

```
{
    "value":
"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEApFt6jz/LP7ZBbfuj6v1yRl/oKz2sbmkLnzP3mXb2eD
WTKL/SyOxcCjwOxnlxNR5Q9BaHMbyFCBaCtAeOmT3mIgB+IoeH00bkibko1NdLi8q//kj/YLYO6x1qzHeSDE4vX
Wig/zE00k3ptU3qznLm1JqfFs7IRinVVqczO+p6NP/XaF5rs46mreqIOsvMzoX4wuIFeG2X+zwIws+YOyLUgJoO
fpJT26FnJIDOPw1uYQMzrG+QY9akZoy0UbDCnezxH7eb8rxSxftUt0LUUKqjssETMU+pTcDVQ6cmeCkwPlnzbge
4hozR04Q5J+y/m/JMzqDBAJ3jB16+XPjaWVleMQIDAQAB"
}
```

Parameter	Type	Description
`value`	string	Paytool public key.

Possible errors

Errors that may occur when attempting to initialize transaction:

401 - Unauthorized

```
{
  "timestamp": "2023-03-29T19:16:01.288+00:00",
  "status": 401,
  "error": "Unauthorized",
  "path": "/api/v1/transactions/9609a08e-cd80-4e6e-8664-f1e6b2f2dc50"
}
```

Parameter	Type	Description
`timestamp`	string($date-time)	Timestamp of the occured error.
`status`	numeric	Numeric HTTP code of the occured error.
`error`	string	Description of HTTP status of the occured error.
`path`	string	Requested endpoint's path

Warning: The structure of this error is different from other errors

```
{
  "uuid": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "traceId": "8fc17b6c-6995-4d33-a4f5-93e99b8af2b2",
  "errorType": "MERCHANT_NOT_EXISTS",
  "merchantStatus": "Merchant entity with typed UUID: 3fa85f64-5717-4562-b3fc-
2c963f66afa6 not exists in database"
}
```

Parameter	Type	Description
`uuid`	string	The identifier of transaction assigned by Paytool system after transaction preinitialization occurrence.
`traceId`	string	Identifier of the occured error.
`errorType`	string	Type of the occured error: • MERCHANT_NOT_EXISTS
`message`	string	Description of the occured error.

500 - Internal server error

```
{
  "date": "2023-09-26 13:51:40.869246",
  "traceId": "8fc17b6c-6995-4d33-a4f5-93e99b8af2b2",
  "errorType": "INTERNAL_SERVER_ERROR",
  "message": "Internal server error occured."
}
```

Parameter	Type	Description
`date`	string($date-time)	Timestamp of the occured error.
`traceId`	string	Identifier of the occured error.
`errorType`	string	Type of the occured error: • INTERNAL_SERVER_ERROR
`message`	string	Description of the occured error.

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.

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 it.

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.

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` .

Tip: 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.

API

`redirect`

Type signature

```
static redirect(transactionId: string): void
```

Example usage

HTML & JavaScript

`index.html`

```
<button id="pay-btn">Pay</button>
<script>
  var transactionId = '1234-1234-1234-1234'; // <- Fetched from server
  var btn = document.getElementById('pay-btn');
  btn.addEventListener('click', function () {
    Paytool.redirect(transactionId);
  });
</script>
```

React

`App.jsx`

```
export function App() {
  const transactionId = '1234-1234-1234-1234'; // <- Fetched from server
  return <button onClick={() => Paytool.redirect(transactionId)}>Pay</button>;
}
```

`embed`

Type signature

```
static async embed(transactionId: string, opts?: Partial<EmbedOptions>):
Promise<EmbedResponse>
```

Attaches a custom dialog component to your app's body and opens Paytool Web App in an iframe.

If the payment process ends successfully, resolves the promise with `EmbedResponse`.

Otherwise, if the process fails or the user closes the dialog, the promise is rejected with `EmbedError`, that's why it is essential to catch any possible errors and act accordingly.

Example usage

HTML & JavaScript

Basic

`index.html`

```
<button id="pay-btn">Pay</button>
<script>
  var transactionId = '1234-1234-1234-1234'; // <- Fetched from server
  var btn = document.getElementById('pay-btn');
  btn.addEventListener('click', function () {
    Paytool.embed(transactionId)
      .then(function (result) {
        // Handle success here
      })
      .catch(function (err) {
        // Handle error here
      });
  });
</script>
```

With options

`index.html`

```
<button id="pay-btn">Pay</button>
<div id="paytool-container"></div>
<script>
  var transactionId = '1234-1234-1234-1234'; // <- Fetched from server
  var btn = document.getElementById('pay-btn');
  btn.addEventListener('click', function () {
    Paytool.embed(transactionId, {
      size: 'compact',
      target: document.getElementById('paytool-container')
    })
      .then(function (result) {
        // Handle success here
      })
      .catch(function (err) {
        // Handle error here
      });
  });
</script>
```

React

Basic

`App.jsx`

```
export function App() {
  const transactionId = '1234-1234-1234-1234'; // <- Fetched from server

  const handleClick = async () => {
    try {
      const result = await Paytool.embed(transactionId);
      // Handle success here
    } catch (err) {
      // Handle error here
    }
  };

  return <button onClick={handleClick}>Pay</button>;
}
```

With options

`App.jsx`

```
export function App() {
  const transactionId = '1234-1234-1234-1234'; // <- Fetched from server
  const containerRef = useRef();

  const handleClick = async () => {
    try {
      const result = await Paytool.embed(transactionId, {
        size: 'compact',
        target: containerRef.current
      });
      // Handle success here
    } catch (err) {
      // Handle error here
    }
  };

  return (
    <>
      <button onClick={handleClick}>Pay</button>
      <div ref={containerRef}></div>
    </>
  );
}
```

`isEmbedError`

Type signature

```
static async embed(transactionId: string, opts?: Partial<EmbedOptions>):
Promise<EmbedResponse>
```

Accepts an unknown `err` and returns `true` if the `err` is an instance of `EmbedError`.

Useful to determine whether the `err` is a message from the SDK or a different JavaScript error.

Example usage

HTML & JavaScript

`index.html`

```
<button id="pay-btn">Pay</button>
<script>
  var transactionId = '1234-1234-1234-1234'; // <- Fetched from server
  var payButton = document.getElementById('pay-btn');
  payButton.addEventListener('click', function () {
    Paytool.embed(transactionId).catch(function (err) {
      if (!Paytool.isEmbedError(err)) return; // Ignore (not recommended) or catch it
using your monitoring tool (recommended)
```

```
      // Handle the err accordingly, for example:
      if (!err.final) return console.log('Transaction aborted.');
      console.log('Transaction failed with status:' + err.status);
    });
  });
</script>
```

React

`App.jsx`

```
export function App() {
  const transactionId = '1234-1234-1234-1234'; // <- Fetched from server

  const handleClick = async () => {
    try {
      const result = await Paytool.embed(transactionId);
    } catch (err) {
      if (!Paytool.isEmbedError(err)) return; // Ignore (not recommended) or catch it using your monitoring tool (recommended)

      // Handle the err accordingly, for example:
      if (!err.final) return console.log('Transaction aborted.');
      console.log('Transaction failed with status:' + err.status);
    }
  };

  return <button onClick={handleClick}>Pay</button>;
}
```

Interfaces

`EmbedOptions`

```
interface EmbedOptions {
  target: HTMLElement;
  size: 'full' | 'compact';
  allowCancel: boolean;
}
```

Defaults

```
{
  target: document.body,
  size: 'full',
  allowCancel: true
}
```

`EmbedResponse`

```
interface EmbedResponse {
  transactionId: string;
  status: string;
  final: true;
  success: boolean;
}
```

`EmbedError`

```
interface EmbedError {
  transactionId: string;
  status?: string;
  reason: string;
  final: boolean;
  success: false;
}
```

Revision #179
Created 1 March 2023 10:55:25 by Jakub Kotyński
Updated 29 April 2024 13:19:46 by Jakub Kotyński