Token Payment Service

Token Payment Service is a functionality that allows Customer to order the one-time payment transactions with card payment token obtained from Google Pay™ or another tokenization platform incl. Card on File

- Introduction
- Overview
- Onboarding
- Integration with Token Payment Service

Introduction

Token Payment Service is a functionality that allows Customer to order the one-time payment transactions with card payment token obtained from Google Pay™. This technology provides a backend to backend oriented solution to which Customer should be integrated. By using this solution, Customer's users can easily pay for their purchases using their cards stored in Google Pay™. After selecting the Google Pay as the payment option, the Customer gets encrypted card payment token from Google Pay™. Card payment token is encrypted on the Google Pay™ side with Verestro's public Key. The Customer has to provide this encrypted token to Verestro by using Token Payment Service soution and Verestro handle entire transaction process by his own. The Customer must be registered in Google Pay™ as a merchant to be able to get a card payment token and have an account with some Acquirer with which Verestro will connect when ordering a transaction.

How to connect with us?

Verestro provides Token Payment Service API which is implemented according to the REST model. This API offers methods that allow to order transaction using payment token obtained from Google Pay™ and authenticate the cardholder using 3D Secure protocol. Verestro team actively supports Customer with integration. More informations about Token Payment Service API can be found here.

Overview

This document provides high level description of functionalities offered by Token Payment Service. Token Payment Service supports e-commerce transactions by card payment token received from Google Pay™ thus eliminating the need to use real card details during transactions. As a registered PSP in Google Pay™, Verestro will decrypt the card payment token and perform the transaction on behalf of the Customer. The solution is very easy to integrate - Customer must integrate two API methods: `tokenPayment` and `deposit`. There is also a third method `getTransacionDetails` which is optional to integrate. The solution can be supported by various Acquirers.

> Verestro recommends using the `getTransacionDetails` method. For example in situations when there were any problems with the connection between the Customer and Verestro. This method allow Customer to get current status of the ordered transaction.

> If the Customer requires the settlement of the transaction by a new Acquirer – to which Verestro is not integrated – there will be required new integration between Verestro and the new Acquirer. The specification of the new Acquirer should be provided by the Customer.

Abbreviation

This section shortly describes abbreviations and acronyms used in the document.

Abbreviation	Description
ACQ	Acquiring Institution / Acquirer
ACS	Access Control Server
OS	Operative System
Mid	Merchant Identifier identifying the Customer in the Acquirer system
PCI DSS	Payment Card Industry Data Security Standard

PAN	Permanent Account Number
CVC	Card Verification Code
3DS	3-D Secure
PSP	Payment Service Provider

Terminology

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

Name	Description
Customer/Merchant	Institution which uses Verestro products. This institution decides which solution should be used depending on the business requirements and how transaction should be processed.
User	End-User which uses Customer application and pays for Customer's goods using Google Pay™ solution. This is the root of the entity tree. User is an owner of the card stored in Google Pay™ system.
Card Payment Token	Card Payment Token is an entity created by Google Pay™ and returned to the Customer. This token is created when the Customera application user selects the card he wants to pay with Google Pay. Card Token Payment is encrypted and does not contain valid card details. This token is decrypted on the Verestro side and then Verestro orders the payment to the Customer's Acquirer.
Authorization Method	The way of the authentication of the card transaction. Verestro supports followed authorization methods: `PAN_ONLY` and `CRYPTOGRAM_3DS` if Customer's country belongs to the European Union. Authorization method is always provided in the Google Pay™ encrypted payload as `authMethod` 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 `verestro`

Gateway Merchant Id	Unique Customer identifier assigned by Verestro during the onboarding process. This identifier is in the form of a `UUID`. 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.
Payment Service Provider	Payment Service Provider is an entity that helps Merchants transfer sensitive data to Acquirer during the transaction. Payment Service Provider should be PCI DSS compilent. In the Token Payment Service solution, Verestro has the role of PSP.
Acquirer	External Institution resposible for processing transaction and 3ds requests ordered by the by Verestro Token Payment Service solution in Customer context. Acquirer connects with banks / card issuers and returns an information whether the ordered action on a given card payment token is possible.
MID	Merchant identifier. This entity is represending Customer / Merchant in Acquirer's system. Customer has to provide the mid information to enable mid configuration in the Verestro system. Required to process transactions and 3DS via Verestro system.
Card Network	This is the type of card that allows you to make payments using a card payment token. Verestro allows to use `MASTERCARD`, `VISA` and `MAESTRO` cards.
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 User wants to pay with a card for purchases on the internet.
CVC	It is a type of security code protecting against fraud in remote payments. Card Verification Code is necessary to provide it when User wants to pay with a card for purchases on the internet.
Expiration Date	It is a date of the card validity ending and contains two values – month/year. Card will be valid to the last day of the month of the year showed on it. It is necessary to provide it when User wants to pay with a card for purchases on the internet.
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 Merchant Paytool solution is performed internally in the Verestro system.

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.

Token Payment Service key components

Token Payment Service is a solution that has been created to provide the functionality that allows Customer to process payments using Google Pay™. An additional assumption was that the payment process should be performed outside the Customer's system, which frees him from the need to handle with sensitive data or the transaction itself. The Customer only receives information that the transaction was successful or not. Customer can also follow the most actual transaction status. This section provides introduction to technologies which are supported by Token Payment Service. High level architecture diagram is presented to show the place and usage of the each entity in the solution.

Component	Description
Token Payment Service API	Component stores the configuration data of a given Customer such Merchant Name or Merchant Id and also communicates with various Acquirers, collect transaction data and statuses. This component also triggers notifications to the Customer and the end user (depending on the Customer requirements) about successful or unsuccessful transaction.
Notification Service	Component responsible for sending information to the Customer about the transaction status. It is also responsible for sending email to the end user about the transaction. Notification Service is triggered by Token Payment Service API.

> The diagram below shows each step of the card payment token transaction 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
}
participant "User" as user
participant "Customer Application" as app
participant "Google Pay" as gp
participant "Verestro Token Payment Service" as tps
participant "Acquirer" as acq
note right of user: User wants to pay with Google Pay
user->app: 1. Pay with Google Pay and choose card
app->gp: 2. Requests for card token
gp->gp: 3. Encrypts card token with Verestro pub key
app<-gp: 4. Returns encrypted card token
app->tps: 5. Requests token payment "/payment/token.google-pay"
tps->tps: 6. Decrypts card token
tps->acq: 7. Orders transaction
tps<-acq: 8. Transaction status
note left of acq: 3DSecure authentication may be required
app<-tps: 9. Transaction status
user<-app: 10. Transaction status
user<--tps: 11. Sends email notification - optional
@enduml
```

Allowed card networks

Listed below are the types of cards supported in transactions using the Token Payment Service and Google Pay™ solution:

Card type
MASTERCARD

VISA
MAESTRO

Implementation models

Verestro provides REST API implementation model in Token Payment Service Solution. In this model Customer has his own application which should be integrated with Token Payment Service API. Verestro provides all necessary backend methods. Customer is responsible for integrate provided methods with his own application. Technical information about the integration can be found here. Below diagram shows high level architecture of the solution:

image-1670238277191.drawio.png
Image not found or type unknown

Onboarding

Register in Verestro

The onboarding process takes place mainly on Verestro side. However, in order to perform onboarding, the Customer must provide some information needed to correctly configure account in Token Payment Service. Configuration includes following information:

Customer name	Basically, it's the name of the Customer's company, his online shop and so on.
Postback URL	This is the address to which information will be send to the Customer about the transaction made by a given user. This parameter is not required if the Customer does not want to receive notifications regarding the transaction. For more information about Postback URL please check Use cases chapter.
Notification to the user	It is a flag that defines whether e-mail notifications was sent to the User. Such e-mail contains the transaction status, transaction execution date, transaction identifier, date and amount. The Customer decides whether Verestro will send such e-mails or not.
Merchant Id (MID)	This ID is represending Customer in Acquirer's system. Customer has to provide the mid information to enable mid configuration in the Verestro system. Required to process transactions and 3DS via Verestro system. Verestro offers support in creating such an account in Acquirer system if it has integration with a given Acquirer. If the client requires transactions to be processed with the participation of a new Acquirer, then Verestro must perform a new integration. The client is then responsible for providing the documentation that Verestro will use during the integration.

After creating an account for the Customer, Verestro provides necessary data to the Customer. This data is required to use the Token Payment Service API. Such data includes:

Basic Authorization	Authorization data for Customer which allow to use the solution. Authorization data are the login and password of the Customer account in the Verestro system. Basic authorization is needed if the Customer wants to `deposit` the `AUTHORIZED` transaction or to `getTransacionDetails`. Basic authorization should be provided as Authorization header with Base64 encoded value of login and password. An example of the Authorization header can be found here .

Gateway Id	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 `verestro` value.
Gateway Merchant Id	This is a unique Customer identifier assigned by Verestro during the onboarding process. This identifier is in the form of a `UUID`. 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.

> Authorization data on BETA and PROD environments differ in password. The login and the gateway Id is the same on both environments.

Register in Google Pay™

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.

Register as Web Merchant

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

Register as Mobile Merchant

> Mobile integration model is work in progress... The solution will be available for mobile merchants soon. This will allow Customers with mobile applications to integrate to the Token Payment Service Solution.

Google Pay Android integration checklist	A checklist presenting the Google Pay integration requirements that must be met by the Customer integrating mobile application
Google Pay Android developer documentation	Technical documentation describing mobile integration with the Google Pay solution
Google Pay Android brand guidelines	Branding requirements that must be met by the Customer's mobile application to be able to use the Google Pay solution

Integration with Token Payment Service

Overview

This chapter provides the instruction of the integration with the solution of the Token Payment Service using Google Pay™ card payment token. Prior to using this solution the Customer have to proceed onboarding process in Verestro and to have an registered merchant account in Google Pay. To register a merchant in Google Pay, please contact Google Pay™ Support.

> Google Pay™ provides a Google Pay Web integration checklist that will help the Customer with 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.

> In addition, Google Pay™ provides Google Pay Web Brand Guidelines that presents branding requirements for web merchants registered in Google Pay™. These requirements must be met by the Customer so that he can allow his payers to pay via the Google Pay™ solution.

> To create an account in Verestro system follow the instruction in the Onboarding chapter.

Integration

Verestro Token Payment Service provides a method for e-commerce payments using a card payment token. The card payment token is generated by Google Pay™ and returned to the Customer in the form of an encrypted payload. Google Pay™ encrypts the card payment token with Verestro's public key. The Customer transfers the received payload to Verestro, which in turn is decrypted on the Verestro side and then the payment is ordered. To facilitate merchant integration with the solution provided by Google Pay™ and to understand the process of making requests for a card payment token, Google Pay™ provides Google Pay Web developer documentation.

>

> Google Pay Web developer documentation is only available after whitelisting. The whitelisting process is performed by Google Pay™ during the merchant account registration process.

The Token Payment Service provides a backend-to-backend oriented payment solution. The solution was created in accordance with the assumptions of the REST API model. The functionality allows Customer to perform `DEPOSITED` or `AUTHORIZED` transactions. If it is an `AUTHORIZED` transaction, it is necessary to call the `deposit` method otherwise, the entire transaction amount will be reversed to the payer's account. To help you understand the difference between `DEPOSITED` and `AUTHORIZED` transaction, please see the table below:

`DEPOSITED`	`tokenPayment` method returned `DEPOSITED` transaction status. There is no further action required.
`AUTHORIZED`	`tokenPayment` method returned `AUTHORIZED` transaction status. This status means that the funds for the purchase have been frozen on the payer's account. The `deposit` method should be called. The amount provided in the `deposit` method may be equal to or less than the amount provided in the `tokenPayment` method. `AUTHORIZED` status appears only when the `deposit` parameter is `false`. This is the parameter accepted in the `tokenPayment` method.

Endpoints

Endpoints chapter contains description of endpoints for Token Payment Service methods. Verestro provides two implementation environments: test - `BETA` and production - `PROD`. Below table presents the addresses of each of the domains:

Environemt	Base url
`BETA`	https://merchant-beta.upaidtest.pl/champion/
`PROD`	https://merchant.upaid.pl/champion/

Methods in API

Token Payment

`POST` `[base-url]/payment/token/google-pay`

The method allows e-commerce payment using a token obtained from Google Pay™. The method also accepts additional parameters such as transaction item ID or payer details. A more detailed

description of the tokenPayment method parameters can be found in the further part of this chapter.

Request body:

```
POST /champion/payment/token/google-pay HTTP/1.1
Content-Length: 1720
Content-Type: application/json
Host: merchant.upaid.pl

{
  "amount": 1000,
  "itemId": "{{itemId}}",
  "description": "SUCCESS",
  "browserIp": "41.11.22.1",
  "currency": "PLN",
  "mid": "testMid",
  "deposit": false,
  "sender": {
    "firstName": "firstName",
    "lastName": "lastName",
    "email": "emaiL@email.pl"
  },
  "token": {
    "signature":
"MEUCIQDx0PjhU6041nIbz6mBagbDUGE9DF8NtLAq1hKyQih9sQIgd5V3ROT5uVXZuYt0i/1RREc1mNnkg0VlnstseI4
oN4w\u003d",
    "intermediateSigningKey": {
      "signedKey":
"{\"keyValue\":\"MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEUj6saq5iwo1JIkLti6dvNFdNJygVoFZUhiKzGwsC2ebD
5v58RutdePd2GxMvx8nGuF3YjVKjwk28R5r2hyTqJQ\\u003d\\u003d\",\"keyExpiration\":\"1667554292000\"}",
      "signatures": [

"MEYCIQDRc5NBd/GC5loetxwL3idIMMsR2vpXicoqlvsPEFlirwIhAMOHvdrHt/sMt5PxS4o0SYO3sCOb6hT9a2t+PMBc
M/u7"
      ]
    },
    "protocolVersion": "ECv2",
    "signedMessage":
"{\"encryptedMessage\":\"Tlkj3N2bApMJ2x2IUgsoTvqYLDuUNpzWmSQbgAIjG2kr+1buPgh70qeKkZkfZRCvJMV0py
1R0RzUyqJujZIqX9tIxPyBX8yQ/bZNOR5AJXDzSLgR2+WuBIa0KRySCwNmzV4U2RVkUdZiIVr8/JL/o7NH768A6rl/Gvh
```

qYfpsFtprATFgsFulDS/dnAhJTS4tykk34wZXnyCb94YHmI5rbN8FFkC/BygPVIKgGt5YWngxRZO1yBSSbcyWb3w+WXI
OaPT6XZdOqDOtOo5GgzNmYKGIdep+b0+hJsreZCG1yPw3o/QS3nDdem40jrUv/apyY1xPSjib3mjXW0e/hnkL3K43n
79po8qmswdPkyOdRh5D10ZElxXRZvI25/WqpsN/jyaeKitDlsOnIGHjiM36S1a4FrTCwmmiV8XDyVzsamW0asemTeJ9
nBbEOylF4dz0symXEWJ45l0SSvPEJqc1HMuPljw1EVAA/MF1O8HBVv7iJndyXS8c2wH5eLLClP3CkT3qb4TjfhhGbhU5J
KclYOxHzvIDUaFYljjzlCyvR+PGwaGq0bzmjtki17t2MKNLtXioQBv9rb2WHZF+tPcA3TJLfo8vijZRzFJQb9JPzDQzCJ7N2
ORfD/LAJWMkeCcvLwuBWPZd3keI\",\"ephemeralPublicKey\":\"BH6BiZF1XFldmO+8EH13KABs4ttulS68cYHg9HRgY
CbV6mdGIa4E2YQuDtsn98MtSQoJ6wA8LtIpa5L6FF9WyCM\\u003d\",\"tag\":\"1BcFY5B7BuSa3cVwRQx58fdHvwW
O8zd0BDrOzva6O14\\u003d\"}"
}
}

Request headers:			
Type	**Value**	**Constraints**	**Description**
Accept-Language	PL	Not required	Response message language
Content-Type	application/json	Required	Content type of the request

Request fields			
Name	**Name**	**Type**	**Description**
`deposit`	`Boolean`	Optional	By setting this flag, the Customer decides whether the funds will be immediately taken from the payer's account (`DEPOSITED`) or frozen (`AUTHORIZED`). Verestro enables configuration in which each successful transaction ends with the `DEPOSITED` status. Default value for this flag is `false`
`browserIp`	`String`	Optional	Browser ip
`amount`	`Number`	Not null	Transaction amount (in pennies)
`currency`	`String`	Not empty	Transaction currency
`description`	`String`	Not empty	Simple description of transaction
`sender`	`Object`	Not null	Sender details
`sender.firstName`	`String`	Not empty	Sender first name

Request fields			
Name	**Name**	**Type**	**Description**
sender.lastName	String	Not empty	Sender last name
sender.email	String	Optional	Sender email
token	Object	Not null	Token obtained from Google
token.signature	String	Not null	Token signature
token.intermediateSigningKey	Object	Not null	Token intermediate signing key
token.intermediateSigningKey.signedKey	String	Not null	Signed key
token.intermediateSigningKey.signatures[]	Array	Not null	Token signatures
token.protocolVersion	String	Not null	Protocol version
token.signedMessage	String	Not null	Signed message
itemId	String	Not empty	Merchant’s unique id of transaction. Ensures the idempotency of the transaction.
mid	String	Optional	This parameter indicates which merchant terminal will be used in the process. If mid won’t be passed, then payment will be processed using the default terminal. This value will be generated in the onboarding process.

Response body:

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

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

```
}
```

Response fields		
Name	**Name**	**Description**
transactionId	String	Identifier of transaction. This parameter should be provided in the deposit method if the tokenPayment method returned AUTHORIZED transaction status. This parameter also defines in the context of which transaction Verestro should return information when executing the getTransactionDetails method.
itemId	String	Merchant's unique id of transaction. Ensures the idempotency of the transaction.
status	String	Transaction status
externalTransactionId	String	External transaction id

tokenPayment **method cURL example:**

```
$ curl 'https://merchant.upaid.pl/champion/payment/token/google-pay' -i -X POST \
    -H 'Content-Type: application/json' \
    -d '{
  "amount": 1000,
  "itemId": "{{itemId}}",
  "description": "SUCCESS",
  "browserIp": "41.11.22.1",
  "currency": "PLN",
  "mid": "testMid",
  "deposit": false,
  "sender": {
    "firstName": "firstName",
    "lastName": "lastName",
    "email": "emaiL@email.pl"
  },
  "token": {
    "signature":
```

```
"MEUCIQDx0PjhU6041nIbz6mBagbDUGE9DF8NtLAq1hKyQih9sQIgd5V3ROT5uVXZuYt0i/1RREc1mNnkg0VlnstseI4oN4w\u003d",
   "intermediateSigningKey": {
    "signedKey":
"{\"keyValue\":\"MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEUj6saq5iwo1JIkLti6dvNFdNJygVoFZUhiKzGwsC2ebD5v58RutdePd2GxMvx8nGuF3YjVKjwk28R5r2hyTqJQ\\u003d\\u003d\",\"keyExpiration\":\"1667554292000\"}",
    "signatures": [

"MEYCIQDRc5NBd/GC5loetxwL3idIMMsR2vpXicoqlvsPEFIirwIhAMOHvdrHt/sMt5PxS4o0SYO3sCOb6hT9a2t+PMBcM/u7"
    ]
   },
   "protocolVersion": "ECv2",
   "signedMessage":
"{\"encryptedMessage\":\"Tlkj3N2bApMJ2x2IUgsoTvqYLDuUNpzWmSQbgAIjG2kr+1buPgh70qeKkZkfZRCvJMV0py1R0RzUyqJujZIqX9tIxPyBX8yQ/bZNOR5AJXDzSLgR2+WuBIa0KRySCwNmzV4U2RVkUdZiIVr8/JL/o7NH768A6rl/GvhqYfpsFtprATFgsFulDS/dnAhJTS4tykk34wZXnyCb94YHmI5rbN8FFkC/BygPVIKgGt5YWngxRZO1yBSSbcyWb3w+WXIOaPT6XZdOqDOtOo5GgzNmYKGIdep+b0+hJsreZCG1yPw3o/QS3nDdem40jrUv/apyY1xPSjib3mjXW0e/hnkL3K43n79po8qmswdPkyOdRh5D10ZElxXRZvI25/WqpsN/jyaeKitDlsOnIGHjiM36S1a4FrTCwmmiV8XDyVzsamW0asemTeJ9nBbEOylF4dz0symXEWJ45l0SSvPEJqc1HMuPljw1EVAA/MF1O8HBVv7iJndyXS8c2wH5eLLCIP3CkT3qb4TjfhhGbhU5JKclYOxHzvIDUaFYljjzlCyvR+PGwaGq0bzmjtki17t2MKNLtXioQBv9rb2WHZF+tPcA3TJLfo8vijZRzFJQb9JPzDQzCJ7N2ORfD/LAJWMkeCcvLwuBWPZd3keI\",\"ephemeralPublicKey\":\"BH6BiZF1XFldmO+8EH13KABs4ttulS68cYHg9HRgYCbV6mdGIa4E2YQuDtsn98MtSQoJ6wA8LtIpa5L6FF9WyCM\\u003d\",\"tag\":\"1BcFY5B7BuSa3cVwRQx58fdHvwWO8zd0BDrOzva6O14\\u003d\"}"
  }
}'
```


> Google Pay™ provides an optional method to retrieve the Billing Shipping Address. However, these data are not used in the `tokenPayment` method, so please do not provide them.

> Google Pay™ does not provide information such as `firstName` or `lastName`. Parameters like this, however, are required to make a payment using the `tokenPayment` method. They must be provided by the Customer together with the card payment token. Basicly the Customer must collect and provide a card payment token from Google Pay and the name of the user of his application.

Transaction statuses

Depending on the transaction status, it may be necessary for the Customer to perform an action (applies to the situation in which the transaction status is `AUTHORIZED`). If the Customer for some

reason (e.g. connection problem) did not receive information about the status of a given transaction, it is recommended to use the `getTransactionDetails` method, which returns information about the current status of a given transaction. The table below presents all possible transaction statuses:

Name	Description
`PENDING`	Transaction waiting for execution.
`FAILED`	Transaction failed.
`AUTHORIZED`	Entire amount of the order was locked.
`DEPOSITED`	Bank account was charged.

Deposit

`POST` `[base-url]/champion/deposit`

This method should be called by the Customer if `tokenPayment` method returns an `AUTHORIZED` transaction status. This action is required to deposit freezed funds. This happens if the `deposit` flag in `tokenPayment` was set to `false`. The amount provided in the `deposit` method may be equal to or less than the amount provided in the `tokenPayment` method. This method is secured by Customer's account credentials (basic authorization). Customer's account is created by Verestro during the onboarding process.

Request body:

```
POST /champion/deposit HTTP/1.1
Authorization: Basic bG9naW46cGFzc3dvcmQ=
Content-Length: 180
Content-Type: application/json
Host: merchant.upaid.pl

{
  "transactionId": "91929c94-974a-413a-8409-5101c933daa9",
  "amount": 100,
  "requestChallengeIndicator": "NO_PREFERENCE",
```

```
  "keyThreedsExemption": "TRANSACTION_RISK_ANALYSIS"
}
```

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

Request fields			
Name	**Name**	**Type**	**Description**
`transactionId`	String	Required	Unique ID of the transaction. This parameter appears in the response of the successfuly performed transaction using `tokenPayment` method
`amount`	Number	Required	Amount for transaction (minor units of currency)

Request fields			
Name	**Name**	**Type**	**Description**
requestChallengeIndicator	String	Optional	Indicates whether a challenge was requested for transaction. Possible values: **NO_PREFERENCE** - no preference for challenge. **CHALLENGE_NOT_REQUESTED** - challenge is not requested. **CHALLENGE_PREFER_BY_REQUESTOR_3DS** - challenge is requested: 3DS Requestor preference. **CHALLENGE_REQUESTED_MANDATE** - challenge is requested: mandate. **RISK_ANALYSIS_ALREADY_PERFORMED** - challenge is not requested: transactional risk analysis already performed. **ONLY_DATA_SHARE** - challenge is not requested: only data is shared. **STRONG_VERIFY_ALREADY_PERFORMED** - challenge is not requested: strong consumer authentication is already performed. **WHITELIST_EXEMPTION** - challenge is not requested: utilise whitelist exemption if no challenge required. **WHITELIST_PROMPT_REQUESTED** - challenge is requested: whitelist prompt requested if challenge required.

Request fields			
Name	**Name**	**Type**	**Description**
keyThreedsExemption	String	Optional	Reason for exemption from strong authentication (SCA). Possible values: **LOW_VALUE_PAYMENT** - low amount of payment. **TRANSACTION_RISK_ANALYSIS** - transactional risk analysis already performed. **TRUSTED_BENEFICIARY** - beneficiary is trusted. **SECURE_CORPORATE_PAYMENT** - corporate payment is secure. **RECURRING_PAYMENT** - recurring payment. **OTHER_MERCHANT_INITIATED_TRANSACTION** - other merchant initiated transaction. **SCA_DELEGATION** - delegation of strong authentication (SCA).

Response status: HTTP/1.1 200 OK

deposit **method cURL example:**

```
$ curl 'https://merchant.upaid.pl/champion/deposit' -i -u 'login:password' -X POST \
    -H 'Content-Type: application/json' \
    -d '{
  "transactionId": "91929c94-974a-413a-8409-5101c933daa9",
  "amount": 100,
  "requestChallengeIndicator": "NO_PREFERENCE",
  "keyThreedsExemption": "TRANSACTION_RISK_ANALYSIS"
}'
```

Get transaction details

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

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

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

Response body:

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

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

Response fields		
Name	**Name**	**Description**
`transactionId`	`String`	Identifier of transaction. This parameter should be provided in the `deposit` method if the `tokenPayment` method returned `AUTHORIZED` transaction status. This parameter also defines in the context of which transaction Verestro should return information when executing the `getTransactionDetails` method.

Response fields		
Name	**Name**	**Description**
`amount`	`Number`	Transaction amount in pennies
`currency`	`String`	Transaction currency
`description`	`String`	Transaction description
`status`	`String`	One of the possible transaction statuses
`threeDsMode`	`String`	ThreeDS process mode which informs about

`getTransactionDetails` **method cURL example:**

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

Possible errors

This chapter presents all the errors that can be obtained using the Token Payment Service solution. The chapter lists each of the error statuses with a description, as well as an example JSON body with a given error.

HTTP Status	Error Status	Error Message
400 - Bad Request	`VALIDATION_ERROR`	Some fields are invalid
400 - Bad Request	`REFUSED_BY_BANK`	Transaction has been rejected by the Acquirer or Issuer
400 - Bad Request	`LACK_OF_FUNDS`	Lack of funds
400 - Bad Request	`TRANSACTION_LIMIT_EXCEEDED`	Transaction limit exceeded
400 - Bad Request	`ACQUIRER_ERROR`	An exception occured on the Acquirer side
400 - Bad Request	`TRANSACTION_NOT_FOUND`	Transaction with provided `transactionId` does not exist in Verestro system
401 - Unauthorized	`UNAUTHORIZED`	Provided merchant's credentials are invalid
422 - Unprocessable entity	`OPERATION_NOT_PERMITTED`	Operation is not permitted

HTTP Status	Error Status	Error Message
422 - Unprocessable entity	`ERROR_WRONG_MID_VALUE`	Provided merchant ID (MID) is not configured for this merchant
422 - Unprocessable entity	`ERROR_CANNOT_DEPOSIT`	Cannot deposit in case: transaction is deposited. Transaction is not `AUTHORIZED` or amount provided in `deposit` is greater than transaction origin amount (this error can occur only in `deposit` method)
422 - Unprocessable entity	`ERROR_WRONG_TRANSACTION_ID`	Incorrect `transactionId` (this error can occur only in `deposit` method)
500 - Internal Server Error	`ERROR_ACQ_EXCEPTION`	An exception occured on the Acquirer side
500 - Internal Server Error	`TRANSACTION_IDEMPOTENCY_ERROR`	Transaction with provided `itemId` already exists
500 - Internal Server Error	`INTERNAL_SERVER_ERROR`	Internal application error
504 - Gateway timeout	`ACQUIRER_RESPONSE_TIMEOUT`	The acquirer has not responded in a specified time

Error examples in JSON format

Response status: HTTP/1.1 400 Bad Request

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

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

Response status: HTTP/1.1 400 Bad Request

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

{
   "transactionId": "{{transactionId}}",
   "status": "REFUSED_BY_BANK",
   "message": "Insufficient funds",
   "traceId": "{{traceId}}"
}
```

Response status: HTTP/1.1 401 Unauthorized

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

{
   "timestamp": "2022-11-30T10:54:32.275+00:00",
   "status": 401,
   "error": "Unauthorized",
   "message": "Unauthorized",
}
```

Response status: HTTP/1.1 422 Unprocessable entity

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

{
   "status": "ERROR_CANNOT_DEPOSIT",
   "message": "ERROR_CANNOT_DEPOSIT",
   "traceId": "{{traceId}}"
}
```

Response status: HTTP/1.1 500 Internal Server Error

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

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

Response status: HTTP/1.1 504 Gateway Timeout

```
HTTP/1.1 504 Gateway Timeout
Content-Length: 32
Content-Type: application/json;charset=UTF-8=

{
   "transactionId": "{{transactionId}}",
   "status": "ACQUIRER_RESPONSE_TIMEOUT",
   "message": "ACQUIRER_RESPONSE_TIMEOUT",
   "traceId": "{{traceId}}"
}
```