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/1RREc1mNnkg0VlnstseI4oN
4w\u003d",
    "intermediateSigningKey": {
      "signedKey":
"{\"keyValue\":\"MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEUj6saq5iwo1JIkLti6dvNFdNJygVoFZUhiKzGwsC2
ebD5v58RutdePd2GxMvx8nGuF3YjVKjwk28R5r2hyTqJQ\\u003d\\u003d\",\"keyExpiration\":\"166755429200
0\"}",
      "signatures": [

"MEYCIQDRc5NBd/GC5loetxwL3idIMMsR2vpXicoqlvsPEFIirwIhAMOHvdrHt/sMt5PxS4o0SYO3sCOb6hT9a2t+PMBcM
/u7"
      ]
    },
    "protocolVersion": "ECv2",
    "signedMessage":
"{\"encryptedMessage\":\"Tlkj3N2bApMJ2x2IUgsoTvqYLDuUNpzWmSQbgAIjG2kr+1buPgh70qeKkZkfZRCvJMV0p
```

```
y1R0RzUyqJujZIqX9tIxPyBX8yQ/bZNOR5AJXDzSLgR2+WuBIa0KRySCwNmzV4U2RVkUdZiIVr8/JL/o7NH768A6rl/Gvh
qYfpsFtprATFgsFulDS/dnAhJTS4tykk34wZXnyCb94YHmI5rbN8FFkC/BygPVIKgGt5YWngxRZO1yBSSbcyWb3w+WXIOa
PT6XZdOqDOtOo5GgzNmYKGIdep+b0+hJsreZCG1yPw3o/QS3nDdem40jrUv/apyY1xPSjib3mjXW0e/hnkL3K43n79po8q
mswdPkyOdRh5D10ZElxXRZvI25/WqpsN/jyaeKitDlsOnIGHjiM36S1a4FrTCwmmiV8XDyVzsamW0asemTeJ9nBbEOylF4
dz0symXEWJ45l0SSvPEJqc1HMuPljw1EVAA/MF1O8HBVv7iJndyXS8c2wH5eLLCIP3CkT3qb4TjfhhGbhU5JKclYOxHzvI
DUaFYljjzlCyvR+PGwaGq0bzmjtki17t2MKNLtXioQBv9rb2WHZF+tPcA3TJLfo8vijZRzFJQb9JPzDQzCJ7N2ORfD/LAJ
WMkeCcvLwuBWPZd3keI\",\"ephemeralPublicKey\":\"BH6BiZF1XFldmO+8EH13KABs4ttulS68cYHg9HRgYCbV6md
GIa4E2YQuDtsn98MtSQoJ6wA8LtIpa5L6FF9WyCM\\u003d\",\"tag\":\"1BcFY5B7BuSa3cVwRQx58fdHvwWO8zd0BD
rOzva6O14\\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

Request fields			
Name	**Name**	**Type**	**Description**
sender.firstName	String	Not empty	Sender first name
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/1RREc1mNnkg0VlnstseI4oN
4w\u003d",
    "intermediateSigningKey": {
      "signedKey":
"{\"keyValue\":\"MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEUj6saq5iwo1JIkLti6dvNFdNJygVoFZUhiKzGwsC2
ebD5v58RutdePd2GxMvx8nGuF3YjVKjwk28R5r2hyTqJQ\\u003d\\u003d\",\"keyExpiration\":\"166755429200
0\"}",
      "signatures": [

"MEYCIQDRc5NBd/GC5loetxwL3idIMMsR2vpXicoqlvsPEFIirwIhAMOHvdrHt/sMt5PxS4o0SYO3sCOb6hT9a2t+PMBcM
/u7"
      ]
    },
    "protocolVersion": "ECv2",
    "signedMessage":
"{\"encryptedMessage\":\"Tlkj3N2bApMJ2x2IUgsoTvqYLDuUNpzWmSQbgAIjG2kr+1buPgh70qeKkZkfZRCvJMV0p
y1R0RzUyqJujZIqX9tIxPyBX8yQ/bZNOR5AJXDzSLgR2+WuBIa0KRySCwNmzV4U2RVkUdZiIVr8/JL/o7NH768A6rl/Gvh
qYfpsFtprATFgsFulDS/dnAhJTS4tykk34wZXnyCb94YHmI5rbN8FFkC/BygPVIKgGt5YWngxRZO1yBSSbcyWb3w+WXIOa
PT6XZdOqDOtOo5GgzNmYKGIdep+b0+hJsreZCG1yPw3o/QS3nDdem40jrUv/apyY1xPSjib3mjXW0e/hnkL3K43n79po8q
mswdPkyOdRh5D10ZElxXRZvI25/WqpsN/jyaeKitDlsOnIGHjiM36S1a4FrTCwmmiV8XDyVzsamW0asemTeJ9nBbEOylF4
dz0symXEWJ45l0SSvPEJqc1HMuPljw1EVAA/MF1O8HBVv7iJndyXS8c2wH5eLLCIP3CkT3qb4TjfhhGbhU5JKclYOxHzvI
DUaFYljjzlCyvR+PGwaGq0bzmjtki17t2MKNLtXioQBv9rb2WHZF+tPcA3TJLfo8vijZRzFJQb9JPzDQzCJ7N2ORfD/LAJ
WMkeCcvLwuBWPZd3keI\",\"ephemeralPublicKey\":\"BH6BiZF1XFldmO+8EH13KABs4ttulS68cYHg9HRgYCbV6md
GIa4E2YQuDtsn98MtSQoJ6wA8LtIpa5L6FF9WyCM\\u003d\",\"tag\":\"1BcFY5B7BuSa3cVwRQx58fdHvwWO8zd0BD
rOzva6O14\\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}}"
}
```

Revision #62
Created 25 November 2022 08:46:37 by Jagoda Mazurek
Updated 23 January 2023 12:48:45 by Jakub Kotyński