Technical documentation

This chapter provides the instruction of the integration with the solution and with it's methods. By using below API you will be able to order quick money transfers to debit or credit cards in 150 major currencies. Lower the costs, save time and increase the end-user satisfaction. Functionality consists of five methods allowing to order payout, check commission and follow transfer status. All methods are secured with `Basic-Authorization` of your merchant account. The `Basic-Authorization` will be provided to you during the onboarding process. Prior using this solution is to open account in acquiring institution. To complete this steps, please contact our Sales Department. We will guide you through entire process.

Environment Test API base URL

```
http://payouts.verestro.dev/
```

Environment Production API base URL

```
NOT YET IMPLEMENTED
```

Sequence diagram presenting payout process

@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F

```
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
actor "Payer" as p
participant "Customer" as c
participant "Verestro" as v
p->c: Make payout (provide transaction data)
c->v: Calculate commission POST /client/calculate-commission/payout
c<-v: Return commission
p<-c: Show commission
p->c: Confirm
c->v: Order payout with transaction data POST /api/v2/client/send-money
v->v: Contact with acquiring institution for transaction to be processed
v->c: Return transaction order-Id with status
p<-c: Payout ordered
v-->c: (optional) Send webhook
c->v: (optional) Check transaction status GET /client/send-money/{order-Id}
v->c: Return http status and proper message
@enduml
```

Important: The success of the multicurrency send-money transaction depends primarily on the correctness of your currency configuration which is done by the Acquirer. To make transactions in a currency other than the currency of the card, contact the Verestro employee.

Important: From January 2021, there is an internal functionality to restrict access for the Customer to specific method. The Acquirer employee can disable access to a given endpoint, then the HTTP status 403 FORBIDDEN will be returned. You will be informed about each access restriction action.

Note: When performing authorization, remember that there are currencies with different number of decimal places. For example: VND has no pennies and KWD has three decimal places. Please take this into account in the Amout field. For more information on other currencies, see ISO 4217.

Order payout

The method allows you to order a payout transfer. The request can be made in four forms depending on the type of reference indicating the receiver of the funds. Customer by selecting `amount` = `X` defines amount of payment in given currency. This amount is transferred to receiver payment instrument (receiver reference) in selected currency. In case there's need revaluation from one currency to another, system uses `higherRate` for this situation. For more details about specific rates please refer to `currencyRate` method.

Receiver reference	Description
`CASH-PLAIN`	Sender provides receiver's card number in plain text.
`CASH-PLAIN-WITH-CALCULATE-COMMISSION-RESULT`	Sender provides receiver's card number in plain text along with earlier calculated commission with `calculateCommissionPayout`.

POST /api/v2/client/send-money

CASH-PLAIN

Headers	
Key	**Value**
`Content-Type`	`application/vnd.sendmoney.v2+json`
`Basic-Authorization`	`Basic dXNlcm5hbWU6cGFzc3dvcmQ=`

> **Note:** The `Basic-Authorization` will be provided to you during the onboarding process.

Example request body in JSON format

```
{
  "amount" : 1000,
  "type" : "RECEIVER",
  "requestId" : "9d7cead6-3532-4028-94ef-666f426f7f74",
  "transactionId" : "TRX220132AM",
  "sender" : {
    "type" : "CASH",
```

```
    "firstName" : "Mark",
    "lastName" : "Smith",
    "street" : "Olszewskiego",
    "houseNumber" : "17A",
    "city" : "Lublin",
    "postalCode" : "20-400",
    "flatNumber" : "2",
    "email" : "senderEmail@verestro.com",
    "personalId" : "AGC688910",
    "country" : "PL"
  },
  "receiver" : {
    "type" : "PLAIN",
    "firstName" : "Rob",
    "lastName" : "Wring",
    "birthDate" : "2024-03-19",
    "cardNumber" : "5117964247989169",
    "currency" : "PLN",
    "countryOfResidence" : "PL"
  }
}
```

Parameter	Type	Description
`amount`	number **required**	The total transfer amount (in pennies).
`type`	string **required**	Transaction in `SENDER` or `RECEIVER` currency, for specific transaction type. `CARD_CARD`: above, `CASH_CARD` : `RECEIVER` , `CASH_CARD`: `SENDER`.
`requestId`	string **required**	UUID generated by the the client, used to identify single transaction. Ensures that the transaction with the given parameter is processed only once.

`transactionId`	string **required**	UUID generated by the the client to assign transaction identifier.
`sender`	object **required**	Object containing datailed payer's data.
`sender.type`	string **required**	For this configuration the value of this field must be `CASH`, otherwise request will be declined.
`sender.firstName`	string **required**	Payers's first name.
`sender.lastName`	string **required**	Payers's last name.
`sender.street`	string **required**	Payer's address.
`sender.houseNumber`	string **required**	Payer's house number.
`sender.city`	string **required**	Payer's city.
`sender.postalCode`	string **required**	Payer's postal code.
`sender.flatNumber`	string **required**	Payer's flat number.
`sender.personalId`	string	Payer's personal id.
`sender.country`	string **required**	Country code in accordance with ISO 3166-1 Alpha-2. Is required for terminal crypto
`receiver`	object **required**	Object containing datailed receiver's data.
`receiver.type`	string **required**	For this configuration the value of this field must be `PLAIN`, otherwise request will be declined.

`receiver.firstName`	string **required**	Receiver's first name.
`receiver.lastName`	string **required**	Receiver's last name.
`receiver.birthDate`	string **required**	Receiver's birth day.
`receiver.cardNumber`	string **required**	Receiver's card number PAN.
`receiver.currency`	string **required**	Currency for transaction. For example: PLN.
`receiver.countryOfResidence`	string	Country code in accordance with ISO 3166-1 Alpha-2. Is required for terminal crypto

CASH-PLAIN-WITH-CALCULATE-COMMISSION-RESULT

Headers	
Key	**Value**
`Content-Type`	`application/vnd.sendmoney.v2+json`
`Basic-Authorization`	`Basic dXNlcm5hbWU6cGFzc3dvcmQ=`

> **Note:** The `Basic-Authorization` will be provided to you during the onboarding process.

Example request body in JSON format

```
{
  "calculateCommissionUuid" : "58e1fc52-dab0-46a2-9198-45eb34024c83",
  "amount" : 1000,
```

```
  "type" : "RECEIVER",
  "requestId" : "2b76bfd8-cfe5-4858-a145-eaed73b8cd9c",
  "transactionId" : "TRX220132AM",
  "sender" : {
    "type" : "CASH",
    "firstName" : "Mark",
    "lastName" : "Smith",
    "street" : "Olszewskiego",
    "houseNumber" : "17A",
    "city" : "Lublin",
    "postalCode" : "20-400",
    "flatNumber" : "2",
    "email" : "senderEmail@verestro.com",
    "personalId" : "AGC688910",
    "country" : "PL"
  },
  "receiver" : {
    "type" : "PLAIN",
    "firstName" : "Rob",
    "lastName" : "Wring",
    "birthDate" : "2024-03-19",
    "cardNumber" : "5117964247989169",
    "currency" : "PLN",
    "countryOfResidence" : "PL"
  }
}
```

Parameter	Type	Description
`amount`	number **required**	The total transfer amount (in pennies).
`type`	string **required**	Transaction in `SENDER` or `RECEIVER` currency, for specific transaction type. `CARD_CARD`: above, `CASH_CARD` : `RECEIVER` , `CASH_CARD`: `SENDER`.

`requestId`	string **required**	UUID generated by the the client, used to identify single transaction. Ensures that the transaction with the given parameter is processed only once.
`transactionId`	string **required**	UUID generated by the the client to assign transaction identifier.
`calculateCommissionUuid`	string	Unique `calculateCommission` result identifier that allows to use calculated commission in transaction.
`sender`	object **required**	Object containing datailed payer's data.
`sender.type`	string **required**	For this configuration the value of this field must be `CASH`, otherwise request will be declined.
`sender.firstName`	string **required**	Payers's first name.
`sender.lastName`	string **required**	Payers's last name.
`sender.street`	string **required**	Payer's address.
`sender.houseNumber`	string **required**	Payer's house number.
`sender.city`	string **required**	Payer's city.
`sender.postalCode`	string **required**	Payer's postal code.
`sender.flatNumber`	string **required**	Payer's flat number.
`sender.personalId`	string **required**	Payer's personal id.
`sender.country`	string **required**	Payer's country.
`receiver`	object **required**	Object containing datailed receiver's data.
`receiver.type`	string **required**	For this configuration the value of this field must be `PLAIN`, otherwise request will be declined.

`receiver.firstName`	string **required**	Receiver's first name.
`receiver.lastName`	string **required**	Receiver's last name.
`receiver.birthDate`	string **required**	Receiver's birth day.
`receiver.cardNumber`	string **required**	Receiver's card number PAN.
`receiver.currency`	string **required**	Currency for transaction. For example: PLN.
`receiver.countryOfResidence`	string	Country code in accordance with ISO 3166-1 Alpha-2. Is required for terminal crypto

Example response body in JSON format - 202 - Accepted

```
HTTP/1.1 202 Accepted
Content-Type: application/json
Content-Length: 56

{
  "orderId" : "0621091f-a35a-4e91-a6bf-1f753304ae83"
}
```

Parameter	Type	Description
`orderId`	string($uuid)	The unique identifier of transaction.

Possible errors

Errors that may occur when attempting to transfer performing:

400 - Bad request

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Content-Length: 104

{
  "error" : {
    "message" : "Another transaction with the same id has already been processed."
  }
}
```

401 - Unauthorized

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json

{
      "timestamp": "2021-12-22T12:39:53.168+0000",
      "status": 401,
      "error": "Unauthorized",
      "message": "ERROR_USER_NOTFOUND",
      "path": "/api/v2/client/send-money"
}
```

200 OK - Error validation

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=ISO-8859-1

{
  "status": "ERROR_VALIDATION",
  "error": {
    "message": "Some information is missing or incorrect.",
```

```
  "errors": [{
    "field": "requestId",
    "message": [
      "may not be null"
    ]
  },{
    "field": "type",
    "message": [
      "may not be null"
    ]
  },{
    "field": "amount",
    "message": [
      "may not be null"
    ]
  }]
}
```


403 - Forbidden


```
HTTP/1.1 403 Forbidden
Content-Type: application/json

{
    "timestamp": 1610464313387,
    "status": 403,
    "error": "Forbidden",
    "message": "No message available",
    "path": "/client/send-money-3ds"
}
```

Calculate commission payout

This method is used to receive information about the commission that will be charged for the transaction. You have to specify in the field: type two values (`SENDER` or `RECEIVER`). For Payouts the value must be `RECEIVER`. The method allows you to calculate commissions for the currencies that have been entered. Result of this method can be used in transaction by passing `calculateCommissionUuid` from the response.

POST /client/calculate-commission/payout

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

Example request body in JSON format

```
{
  "amount" : 100,
  "type" : "RECEIVER",
  "sender" : {
    "type" : "CASH"
  },
  "receiver" : {
    "type" : "PLAIN",
    "cardNumber" : "5575167825713507",
    "currency" : "PLN"
  }
}
```

Parameter	Type	Description
`amount`	number **required**	The total transfer amount (in pennies)

`type`	string **required**	Value for specific transaction type. Must be `RECEIVER`.
`sender`	object **required**	Object containing detailed payer's data.
`sender.type`	string **required**	Required configuration per request. Must be `CASH` type.
`receiver`	object **required**	Object containing detailed receiver's data.
`receiver.type`	string **required**	For this configuration the value of this field must be `PLAIN`, otherwise request will be declined.
`receiver.cardNumber`	string **required**	Receiver's card number PAN.
`receiver.currency`	string **required**	Currency for transaction. For example: PLN.

Example response body in JSON format - 200 - OK

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 186

{
  "calculateCommissionUuid" : "6d43d706-570e-47bd-be48-976c0c9b23b8",
  "depositChargeAmount" : 200,
  "depositChargeCurrency" : "PLN",
  "calculateCommissionExpiration" : 1710893068
}
```

Parameter	Type	Description
`calculateCommissionUuid`	string	Unique identifier that can be used in authorization to use calculate commission result.

`depositChargeAmount`	number	Amount that will be charged from deposit in pennies
`depositChargeCurrency`	string	Deposit currency
`calculateCommissionExpiration`	number	Expiration date of calculate commission result in unix time

Possible errors

422 - Unprocessable entity

```
HTTP/1.1 422 Unprocessable Entity
Content-Type: application/json
Content-Length: 184

{
  "status" : "E0152",
  "message" : "Transaction rejected, issuer card not supported",
  "httpStatus" : "UNPROCESSABLE_ENTITY",
  "traceId" : "483ba538-ff94-41eb-b54d-34d1a6336ddb"
}
```

500 - Internal server error

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json
Content-Length: 150

{
  "status" : "E9000",
  "message" : "Domain error",
  "httpStatus" : "INTERNAL_SERVER_ERROR",
  "traceId" : "edeb0c72-2b63-4be4-81d3-a5a878609726"
}
```

Currency rate by provider

This method is used for determine currency rate for revaluation from funding to payment (`lowerRate`) and payment to funding (`higherRate`). Notice that `lowerRate` is used to transaction processing.

> **Tip:** Payout API allows users to select the direction of revaluation by providing specify type value in `orderPayout` request. User by selecting `type` = `SENDER` defines amount of funding in given currency. This amount is collected from sender card in selected currency. In case there's need revaluation from one currency to another, system uses `lowerRate` .

POST /client/currency-rate

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

Example request body in JSON format

```
{
  "provider" : "MASTERCARD",
  "from" : "USD",
  "to" : "PLN",
  "effectiveDate" : "2017-06-05 12:00:00"
}
```

Parameter	Type	Description

`provider`	string **required**	VISA or MASTERCARD or MAESTRO.
`from`	string **required**	Source revaluation currency.
`to`	string **required**	Destination revaluation currency.
`effectiveDate`	string	Date from which the currency rate is needed. This is optional field. When there is no effectiveDate field, then currency rate is getting from request date. (Format "yyyy-MM-ddHH:mm:ss")

Example response body in JSON format - 200 - OK

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 104

{
  "status" : "SUCCESS",
  "success" : {
    "lowerRate" : 3.735908,
    "higherRate" : 3.8522295
  }
}
```

Parameter	Type	Description
`status`	string	Status of the revaluation.
`success`	object	Rate for revaluation.
`success.lowerRate`	decimal	Rate for revaluation from funding to payment
`success.higherRate`	decimal	Rate for revaluation from payment to funding

Possible errors

200 - OK

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

{
    "status": "CURRENCY_INVALID",
    "error": {
        "message": "Invalid currency."
    }
}
```

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

{
    "status": "CURRENCY_RATES_INVALID",
    "error": {
        "message": "Invalid currency rates."
    }
}
```

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

{
    "status": "ERROR_VALIDATION",
    "error": {
        "message": "Some information is missing or incorrect.",
        "errors": [
            {
                "field": "sender.currency",
                "message": [
```

```
                    "Currency is not supported"
                ]
            },
            {
                "field": "receiver.currency",
                "message": [
                    "Currency is not supported"
                ]
            }
        ]
    }
}
```


Check status

The method allows to get a status of multi-currency transfer providing transfer order id in the method's URL address. Parameter order id was returned in the response of the `orderPayout` method.

GET /client/send-money/details/{orderId}

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

Query parameter	Value
`orderId`	`<UUID of the ordered transfer>`

Example response body in JSON format - 200 - OK

```
{
  "transactionId" : "TRX220132AM",
  "amount" : 1000,
  "amountInUsDollar" : 268,
  "bigDecimalAmount" : 10.0,
  "commission" : 200,
  "bigDecimalCommission" : 2.0,
  "orderId" : "00549d98-08cb-45d2-8673-4dcafa81f498",
  "createdDate" : "03-04-2018, 14:01",
  "fundingRrn" : "014011103023",
  "paymentRrn" : "014011103024",
  "arn" : "05411640143500000019325",
  "3DS" : true,
  "revaluationResult" : {
    "revaluationFundingAmount" : 1000,
    "bigDecimalRevaluationFundingAmount" : 10.0,
    "fundingCurrency" : "PLN",
    "revaluationPaymentAmount" : 1000,
    "bigDecimalRevaluationPaymentAmount" : 10.0,
    "paymentCurrency" : "PLN",
    "determineCurrencyRate" : {
      "from" : "PLN",
      "to" : "PLN",
      "currencyRate" : "1"
    }
  },
  "receiver" : {
    "firstName" : "John",
    "lastName" : "Novak",
    "provider" : "MASTERCARD",
    "hiddenCardNumber" : "557455******1623",
    "bankName" : "Alior Bank SA"
  },
  "sender" : {
    "firstName" : "Caroline",
    "lastName" : "Novak",
    "provider" : "MASTERCARD",
    "hiddenCardNumber" : "511796******9169",
```

```
    "bankName" : "Alior Bank SA"
  }
}
```

Response parameters		
Parameter	**Type**	**Description**
`amount`	number	Amount of the transferred cash of the currency in pennies [1PLN = 100].
`amountInUsDollar`	number	Amount of the transferred cash in pennies in USD currency [1PLN = 100].
`bigDecimalAmount`	number	Amount of the transferred cash with decimal precision.
`commission`	number	Amount of the commission added to the ordered transfer in pennies [1PLN = 100]
`bigDecimalCommission`	number	Amount of the commission added to the ordered transfer with decimal precision.
`orderId`	string	Unique transaction identifier.
`transactionId`	string	This parameter is used to send you your own internal transaction identifier. This field is also sent by the `webhook` method.
`createdDate`	string	Date of transaction order.
`fundingRrn`	string	Funding retrieval reference number.
`paymentRrn`	string	Payment retrieval reference number.
`arn`	string	Acquirering institution reference number.
`3DS`	boolean	The value: `true` / `false` informs whether 3DS was performed or not.

`revaluationResult`	object	Detailed information about revaluation between sender currency and receiver currency.
`revaluationResult.revaluationFundingAmount`	number	Amount of the funding transaction in `fundingCurrency` in pennies [1PLN = 100].
`revaluationResult.bigDecimalRevaluationFundingAmount`	number	Amount of the funding transaction in decimal precision.
`revaluationResult.fundingCurrency`	string	Currency code the same as sender’s card currency.
`revaluationResult.revaluationPaymentAmount`	number	Amount of the payment transaction in `paymentCurrency` in pennies [1PLN = 100].
`revaluationResult.bigDecimalRevaluationPaymentAmount`	number	Amount of the payment transaction in decimal precision.
`revaluationResult.paymentCurrency`	string	Currency code the same as receivers's card currency.
`revaluationResult.determineCurrencyRate`	object	Details about currency conversion.
`revaluationResult.determineCurrencyRate.from`	number	Currency coverted "from".
`revaluationResult.determineCurrencyRate.to`	number	Result of the conversion.
`revaluationResult.determineCurrencyRate.currencyRate`	number	Currency rate.

Possible errors

203 - Non-authoritative information

Important! After you get 203 and if you don’t get a response (200 - succeeded or 500 - declined) within 60 seconds then please contact us.

```
HTTP/1.1 203 Non-Authoritative Information
```

401 - Unauthorized


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


404 - Not found


```
HTTP/1.1 404 Not Found
Content-Type: application/json
Content-Length: 51

{
  "errorStatus" : "ERROR_TRANSACTION_NOT_FOUND"
}
```


422 - Unprocessable entity


```
HTTP/1.1 422 Unprocessable Entity
Content-Type: application/json
Content-Length: 1287

{
  "transactionId" : "TRX220132AM",
  "amount" : 1000,
  "amountInUsDollar" : 268,
  "bigDecimalAmount" : 10.0,
  "commission" : 200,
  "bigDecimalCommission" : 2.0,
  "orderId" : "00549d98-08cb-45d2-8673-4dcafa81f498",
  "createdDate" : "03-04-2018, 14:01",
```

```
  "fundingRrn" : "014011103023",
  "paymentRrn" : "014011103024",
  "arn" : "05411640143500000019325",
  "3DS" : true,
  "revaluationResult" : {
    "revaluationFundingAmount" : 1000,
    "bigDecimalRevaluationFundingAmount" : 10.0,
    "fundingCurrency" : "PLN",
    "revaluationPaymentAmount" : 1000,
    "bigDecimalRevaluationPaymentAmount" : 10.0,
    "paymentCurrency" : "PLN",
    "determineCurrencyRate" : {
      "from" : "PLN",
      "to" : "PLN",
      "currencyRate" : "1"
    }
  },
  "receiver" : {
    "firstName" : "John",
    "lastName" : "Novak",
    "provider" : "MASTERCARD",
    "hiddenCardNumber" : "557455******1623",
    "bankName" : "Alior Bank SA"
  },
  "sender" : {
    "firstName" : "Caroline",
    "lastName" : "Novak",
    "provider" : "MASTERCARD",
    "hiddenCardNumber" : "511796******9169",
    "bankName" : "Alior Bank SA"
  },
  "transactionStatus" : "DECLINED",
  "cardBlockType" : "TEMP",
  "cardBlockedUntil" : "2024-03-21T01:04:17.573",
  "errorStatus" : "ERROR_SENDER_CARD_IS_BLOCKED"
}
```

422 - Unprocessable entity CASH-CARD

```
HTTP/1.1 422 Unprocessable Entity
Content-Type: application/json
Content-Length: 1358

{
  "transactionId" : "TRX220132AM",
  "amount" : 1000,
  "amountInUsDollar" : 268,
  "bigDecimalAmount" : 10.0,
  "commission" : 200,
  "bigDecimalCommission" : 2.0,
  "orderId" : "00549d98-08cb-45d2-8673-4dcafa81f498",
  "createdDate" : "03-04-2018, 14:01",
  "fundingRrn" : "014011103023",
  "paymentRrn" : "014011103024",
  "arn" : "05411640143500000019325",
  "3DS" : false,
  "revaluationResult" : {
    "revaluationFundingAmount" : 1000,
    "bigDecimalRevaluationFundingAmount" : 10.0,
    "fundingCurrency" : "PLN",
    "revaluationPaymentAmount" : 1000,
    "bigDecimalRevaluationPaymentAmount" : 10.0,
    "paymentCurrency" : "PLN",
    "determineCurrencyRate" : {
      "from" : "PLN",
      "to" : "PLN",
      "currencyRate" : "1"
    }
  },
  "receiver" : {
    "firstName" : "John",
    "lastName" : "Novak",
    "provider" : "MASTERCARD",
```

```
    "hiddenCardNumber" : "557455******1623",
    "bankName" : "Alior Bank SA"
  },
  "sender" : {
    "firstName" : "Caroline",
    "lastName" : "Novak",
    "provider" : "CASH"
  },
  "transactionStatus" : "DECLINED",
  "merchantSettlementCurrency" : "USD",
  "fenigeCommissionInMerchantSettlementCurrency" : 0.05,
  "transactionAmountInMerchantSettlementCurrency" : 2.68,
  "cardBlockType" : "TEMP",
  "cardBlockedUntil" : "2024-03-21T01:04:18.165",
  "errorStatus" : "ERROR_SENDER_CARD_IS_BLOCKED"
}
```


500 - Internal server error

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/plain;charset=ISO-8859-1

PAYMENT_TRANSACTION_DECLINED:CODE_05
```

Webhook

This method allow you to receive notification after the ordered transaction. After handling the request from Verestro system, you will be notified of the current status of the transaction. Then you can be sure that the transaction processing was finished and you can get the transaction details if you want to. This functionality is optional and it is not required to use Payout solution.

Note: To use the webhooks functionality, please notify Verestro Sales Department. After that we will configure URL address and a secret token which you will be using to communicate with webhook service. Please notice you must specify the URL - webhook will be sent to this

address. The secret token will be generated by the Verestro employee and sent to the client.

Sequence diagram presenting webhook 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 "Customer" as c
participant "Verestro" as v
c->v: Transaction request
c<-v: Response
v->v: Transaction processing...
v->c: Transaction processing finnished callback (webhook)
c->v: Response HTTP Status 200 OK
@enduml
```

You must return HTTP status 200 OK after receiving webhook. Otherwise our server will retry the request. There are 3 attemps of requesting webhook. Every repeat is executed with 5 seconds interval excluding timeout from your server.

Tip: In order to protect client API by polling or other undesirable actions, the webhook service uses headers. If you want to use get webhook notification, you need to handle required headers on your side.

Tip: To build `X-MERCHANT-SECRET` header:
1. Concatenate secret token established by you and Verestro's employee with `orderId` of transaction
2. Hash with SHA256 function result of above operation

Example of X-MERCHANT-SECRET building

```
import hashlib

# secret token established by client with verestro's employee
secret = 'mNaU9TaK4m9myYYFBJgKu8slNH2fCKutJyzXwI'

# orderId received from webhook's request
order_id = 'c168a885-acfa-4a91-a1ad-ed7a042b7238'

# concatenate strings in correct order
concatenated = secret + order_id

# use SHA256 hashing function
hashed = hashlib.sha256(concatenated.encode('utf-8')).hexdigest()

# then compare 'hashed' variable with content of 'X-MERCHANT-SECRET' header
```

There are three possible states of the webhook: `TRANSACTION_APPROVED`, `TRANSACTION_DECLINED` or `TRANSACTION_REVERSED`. Each of the webhooks is presented below:

Headers	
Key	**Description**
`X-MERCHANT-SECRET`	SHA256 Hash string composed from secret token and orderId placed in request body of this webhook
`X-MERCHANT-TIMESTAMP`	Timestamp of server response in UNIX format for instance: 1614023731

TRANSACTION_APPROVED

```
Content-Type: application/json
X-MERCHANT-SECRET: 3cbd17f561150a1394cabbe2b6031fd83f3f3081abe28c32b7fed16f32aebc4a
```

```
X-MERCHANT-TIMESTAMP: 1614800720
{
    "orderId": "c168a885-acfa-4a91-a1ad-ed7a042b7238",
    "transactionId": "TRX220132AM",
    "status": "APPROVED",
    "responseCode": "CODE_00",
    "amount": 900,
    "amountCurrency": "PLN",
    "amountInUsDollar": 248,
    "revaluationResult": {
      "revaluationFundingAmount": 900,
      "bigDecimalRevaluationFundingAmount": 9,
      "fundingCurrency": "PLN",
      "revaluationPaymentAmount": 900,
      "bigDecimalRevaluationPaymentAmount": 9,
      "paymentCurrency": "PLN",
      "determineCurrencyRate": {
        "from": "PLN",
        "to": "PLN",
        "currencyRate": "1"
      }
    },
    "commissionAmount": 46,
    "commissionCurrency": "PLN"
}
```

TRANSACTION_DECLINED

```
Content-Type: application/json
X-MERCHANT-SECRET: 3cbd17f561150a1394cabbe2b6031fd83f3f3081abe28c32b7fed16f32aebc4a
X-MERCHANT-TIMESTAMP: 1614800720

{
    "orderId": "42e8a03a-eb2e-4208-b99b-ac2ad6308498",
    "transactionId": "TRX220132AM",
    "status": "DECLINED",
    "responseCode": "CODE_05",
    "errorMessage": "FUNDING_TRANSACTION_DECLINED: CODE_05",
```

```
    "amount": 900,
    "amountCurrency": "PLN",
    "amountInUsDollar": 248,
    "revaluationResult": {
      "revaluationFundingAmount": 900,
      "bigDecimalRevaluationFundingAmount": 9,
      "fundingCurrency": "PLN",
      "revaluationPaymentAmount": 900,
      "bigDecimalRevaluationPaymentAmount": 9,
      "paymentCurrency": "PLN",
      "determineCurrencyRate": {
        "from": "PLN",
        "to": "PLN",
        "currencyRate": "1"
      }
    },
    "commissionAmount": 46,
    "commissionCurrency": "PLN",
    "merchantAdviceCode": "03 - Do not try again"
}
```


TRANSACTION_REVERSED


```
Content-Type: application/json
X-MERCHANT-SECRET: 3cbd17f561150a1394cabbe2b6031fd83f3f3081abe28c32b7fed16f32aebc4a
X-MERCHANT-TIMESTAMP: 1614800720

{
    "orderId": "1b498361-f8db-406e-943b-ca2b12b7aa38",
    "transactionId": "TRX220132AM",
    "status": "REVERSED",
    "responseCode": "CODE_00",
    "amount": 1000,
    "amountCurrency": "PLN",
    "amountInUsDollar": 273,
    "revaluationResult": {
      "revaluationFundingAmount": 1000,
```

```
      "bigDecimalRevaluationFundingAmount": 10,
      "fundingCurrency": "PLN",
      "revaluationPaymentAmount": 262,
      "bigDecimalRevaluationPaymentAmount": 2.62,
      "paymentCurrency": "USD",
      "determineCurrencyRate": {
        "from": "PLN",
        "to": "USD",
        "currencyRate": "0.2616157"
      }
    },
    "commissionAmount": 1,
    "commissionCurrency": "PLN"
}
```


Revision #69
Created 29 March 2024 09:24:01 by Jakub Kotyński
Updated 8 April 2024 11:48:45 by Jakub Kotyński