Payout to card

The Payout to card product allows you or your application user (sender) to order a transfer directly from his balance to the potential recipient's card. Sender chooses balance to be debited and provides receiver card to be top upped.

- Article
  - Payout to Cards
  - Currency Management in Payouts to Cards
  - Various forms of money transfers
  - Payouts, eCom Transactions or Card-to-Card Payments?
  - Cross-Border Card Transactions – Questions about FX Rates and Cash Flow
- Introduction
- How to integrate - API reference
- Onboarding

Article

You can find more knowledge about products on this site.

Article

Payout to Cards

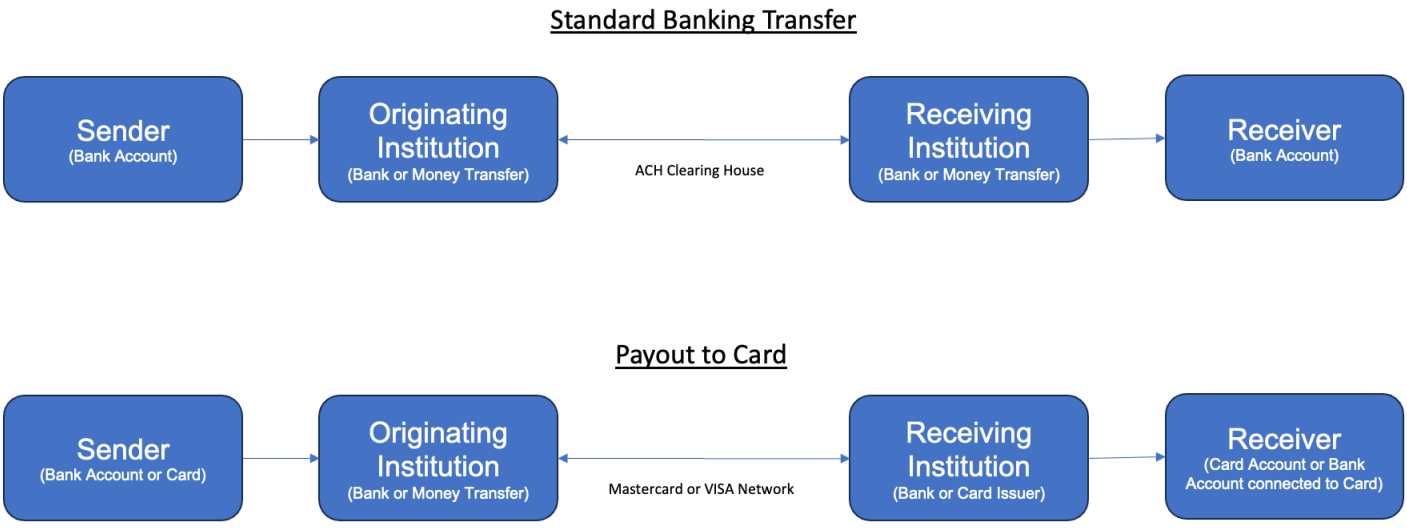
How does Payout to Cards work? Payout to Cards is a relatively new area of payment business that is not very common, so this article brings more information about it.

What is Payout to Cards? Actually, the answer is simple. It is nothing more than a normal bank transfer, but made to a card number instead of a bank account number.

Transfers to bank accounts are pretty common and I guess we understand how they work. A bank or another financial institution connects to Automated Clearing House (usually National Clearing Center or National Bank or inter-bank organization), implements the solution on both frontend (internet banking, mobile banking, internal systems) and backend (integration with core-banking system and ACH), and once Customer wants to send money and enters IBAN (bank account) of the receiver, the transfer is performed. In such a case the bank sends technical information to ACH and sends money or performs settlement either with another bank or National Bank, or any other payment organization responsible for this transfer.

Payouts to cards work completely in the same way, but the money transfer is done to Mastercard or VISA cards. At Mastercard, this solution is called "MoneySend" (sometimes Mastercard Send or Cross-Border Send), while at VISA it is called "VISA Direct". In case of such a transaction Customer of the bank or any other money transfer organization initiates payment via the Internet or mobile application and sends money to the Primary Account Number (card number) of the receiver. The settlement of money happens via the Mastercard and VISA networks - actually through settlement bank accounts registered at Mastercard and VISA to perform a card transaction. Money is taken from the settlement account of Originating Institution (sending institution) to the settlement account of Receiving Institution.

We present this on the chart below.



In fact, there are not many differences between a standard bank transfer and Payout to Cards. Real differences are a natural result of using payment cards to process transactions. The main differences are:

- **Pricing** - obviously pricing of such a Payout to Cards is different than a standard banking transfer - usually more expensive. This is the outcome of the pricing policy of VISA and Mastercard. Nothing else. On average, Payout to Card costs around 0,5-1% + 0,1-0,8 EUR per transaction.
- **Speed of the transfer delivery** - Receiver of a Payout to Cards transaction usually receives money (globally) within 30 minutes. It is a big game changer compared to SWIFT or SEPA transfers. It really works globally. Imagine that you can send money from Brazil to Germany in 30 minutes! From Singapore to Pakistan in 30 minutes!
- **Using a card number** - Receiver needs to share his/her card number (only 16 digits) with Sender. This is a significant problem because we do not like sharing card numbers with other people. Actually we are taught that it is risky. This can impact a user conversion in many use cases.
- **Issues with a receiving network** - Sometimes it is difficult or impossible to send transactions to particular countries. For example Germany or the USA are countries where such transactions are blocked - banks usually do not accept receiving Payouts to Cards. This may be a problem for some use cases and some transaction corridors.
- **Maximum transaction value** - VISA and Mastercard decided that there are some maximum transaction values. Usually it is around 5-10k EUR or USD per transaction. There are also some monthly limits per user. It does affect the user experience but this value is growing over time.

In general, it is a great functionality that works well for banks around the world as competitive to SWIFT and ACH. It gives added value to the user who wants to transfer money quickly, especially internationally. Worth considering for all money transfer organizations and banks. The implementation of Payout to Cards can be greatly simplified by Verestro and our partner payment organization Fenige. Please check us out!

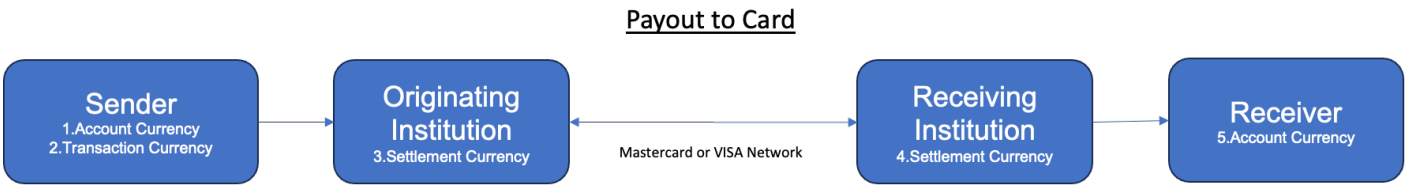
Article

Currency Management in Payouts to Cards

There are many questions about how to manage currencies in payout products. Let me briefly describe several possible scenarios.

Let's start with dependencies that have an impact on choosing various scenarios.

1. **Sender's card account currency** - first you have a user with a payment account in a particular currency, for example USD, EUR, CHF, RON etc.
2. **Transaction currency** - transaction that sending user can choose
3. **Acquirer settlement currency** - there are settlement currencies that an acquiring institution (Originating Institution) cooperating with VISA or Mastercard uses to settle money with them, for example USD, EUR, PLN. Of course, it can differ from the user account currency.
4. **Receiver's card issuer settlement currency** - a bank, which issues a card for the receiver, can have various settlement currencies with Mastercard or VISA.
5. **Receiving card settlement currency** - additionally, there is a settlement currency of the receiver's card, issued by another bank. It can be any currency, for example UAH, CZK.



That's why it is complex. At various levels of transactions there are various currencies and of course in case of currency conversion at any step various additional FX fees apply. That's why the choice of currency management strategy is not an easy one.

Additional decision factors are related to particular use cases I want to present. There are a few possible ways of offering Payouts to the user. Let's have a look at 3 scenarios:

1. **User chooses how much money in their currency they want to transfer** - example: User has an account in USD and wants to send 100 USD to a friend. User does not know if the friend has an account in USD, EUR or PLN. He/she does not care.
   - A. In such a case there is no problem if Sender and Receiver, Acquirer and Issuer have an account in the same currency as available settlement accounts of Acquirer.

Transactions will be processed and settled in the same currency through the chain. This almost always applies for USD, EUR transactions.
- B. If Sender has an account in USD, Acquirer has a settlement currency in USD, Issuer has a settlement currency in EUR, Receiver has a card account in EUR, there will be currency conversion that will happen on Receiver's side. His/her bank (card issuer) will convert the incoming USD to EUR and charge currency conversion fees.
- C. If Sender has an account in CZK, but Acquirer does not have a settlement currency in CZK, but only USD and Receiver has an account in USD, there will be conversion happening on Sender's (acquirer) side. The sending institution will convert 1000 CZK of User to USD, will charge currency conversion fees and Receiver will receive USD after conversion. Receiver's bank will not get any currency conversion fees.

2. **User chooses currency of Receiver** - Example: User has an account in USD but needs to pay 100 EUR to Receiver because he/she knows that Receiver wants to get 100 EUR.
   - D. It is possible to recognise the settlement currency of Receiver thanks to BIN tables shared through payment schemes. Thanks to it Sender will know that Receiver's card is issued in USD, so only USD will be allowed for this transaction. In such a case currency conversion will always happen on Sender's side. In case User has an account with EUR, their Acquirer (Originating Institution) will convert 100 EUR to USD and will initiate a transaction in USD. In case User account is in a different currency than the settlement account of Acquirer, additional currency conversion fees will apply and will be charged by Acquirer.
3. **User does not have a choice** - in such a case we offer only a payment in currency defined by the payment provider, for example always the same currency as the User account.
   - E. In such a case User can send only one currency. Usually the same as his/her account currency. If User's account currency is the same as the settlement account of Acquirer, the transaction will be processed as in point 1B, which means that currency conversion can happen on Receiver's side if Receiver's card currency is different from the settlement currency.
   - F. In case User can send money in the currency which is not the settlement account of Acquirer in, some additional conversion fees will apply on Acquirer's side (like in scenario 1C).

It may look complicated, but if you look at it from the point of view of currency conversion points (5 places where conversion can happen) it is easier to understand.

Our recommendation is to use Scenario 1 and focus on implementing Scenario 1A (we can enable currencies which will be the most popular for your payment corridors). In some cases our partners use Scenario 2. It is important that calculation of commissions and spread is always dynamic, so Sender knows in advance the cost of these transactions.

I hope this article can help you understand currency conversion details. Thank you for reading.

Article

Various forms of money transfers

There are multiple forms of **money transfers**. In this article we would like to summarize the most important pros and cons of every solution:

1. **SWIFT** (Society for Worldwide Interbank Financial Telecommunication) - inter-banking payment scheme enabling global transfer, International standard
    1. Pros - almost any currency; global network, unlimited amount of transfer
    2. Cons - time of transaction (sometimes a week); cost of transaction (example: 0,3%+10 usd or more); available to banks only

2. **Payouts to Cards** - using Mastercard and VISA network, global transfers, international standard
    1. Pros - almost any currency; global network; speed (even 30 minutes to transfer money between continents
    2. Cons - Cost of transaction (example: 1% + 0,5 usd), limited amount of transfer (10.000 USD)

3. **Crypto** - using cryptography to transfer value, global transfer, international standard but sometimes forbidden by law
    1. Pros - multiple but virtual currencies; global network; speed (even 5 minutes)
    2. Cons - high costs (example: 1-2%), very often forbidden by regulators, risk of losing money, needs crypto exchange involvement

4. **SEPA** (Single Euro Payment Area) - European standard or euro currency standard
    1. Pros - speed (immediately or 1 day), price (below 1 EUR)
    2. Cons - works only from EUR to EUR, works only in the European Union

5. **Payouts to wallets** - various providers offer various payouts mechanisms to multiply local wallet providers or cash-out networks
    1. Pros - localization
    2. Cons - no global standard in speed and price, usually more expensive

6. **Virtual cards** - you can issue a virtual card, send card data to the receiver and the receiver can use the card globally
    1. Pros - global standard, very quick and very cheap, receiver can use card for ATM withdrawal, POS and eCommerce payments

    2. Cons - non-standard way of sending money, receiver reluctance

7. **Local ACH** (Automated Clearing House or local scheme) - there are multiple local or national payment schemes globally that you can use once you integrate with them. Usually requires a bank license to integrate.
    1. Pros - quick and cheap, standard in the country
    2. Cons - no global standard, works only locally

If you are asking yourself which solution you should use for your user it is actually a wrong question. We recommend using all. Give choice to your users, apply various fees on various methods of transfer, let users choose the best way of payments for them. It is actually very important strategy because:

- for users in Poland SEPA transfer or local ACH are the most common ways of payments nowadays
- for users in Ukraine Payouts to cards are the most common mechanism they have been using for years
- for users in USA SWIFT or local payment schemes are the most common mechanism

If you are building an international service, you really need multiple ways of sending money for your users.

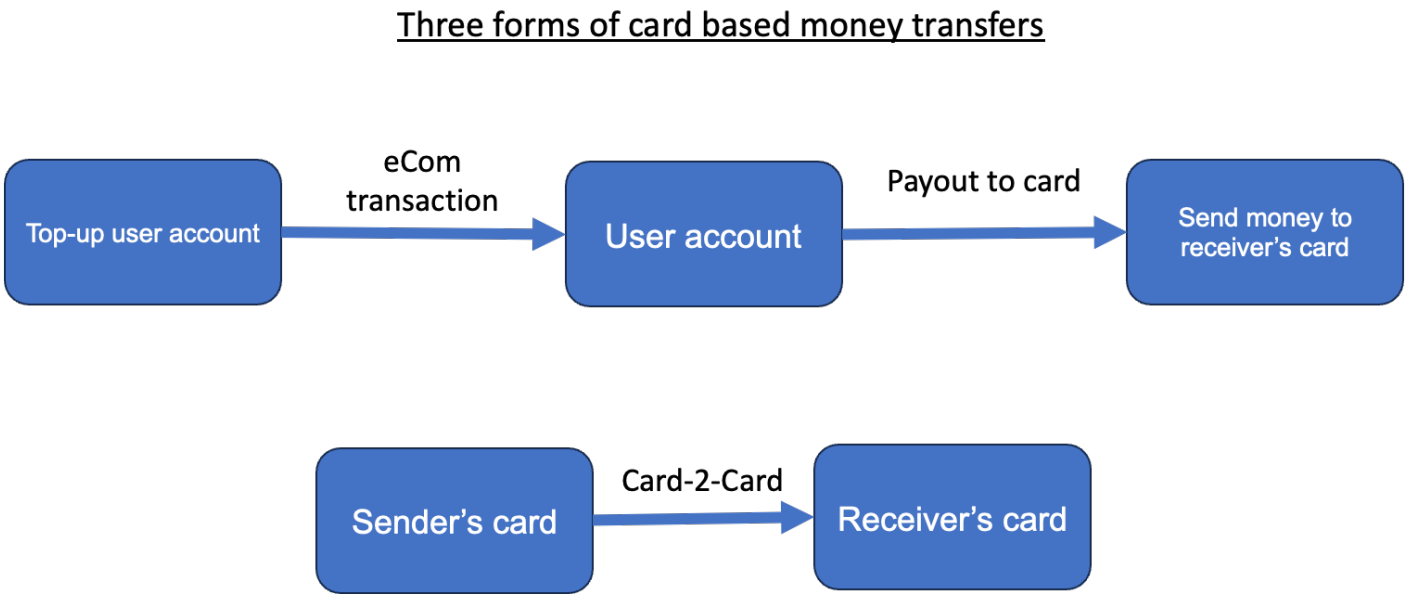
Thanks for reading.

Article

Payouts, eCom Transactions or Card-to-Card Payments?

While thinking about card-based money transfer solutions, our partners usually ask for three products - **payouts to cards**, **eCom transactions** or **card-to-card payments**. In this article we will describe differences between those 3 ways of money transfers.

Let me start with a chart.



There are three use cases that you may be interested in. The choice of product depends on a use case decision.

Use Case 1. **Top-up user account** - in this case our starting point is the user's account kept somewhere in your systems. Your users need to reload this account with money. You can use various forms of transfers to your account, but *if you want to reload an account from Mastercard or VISA card*, we should enable eCom transactions to you. You will be registered as a merchant with our partnering acquirers and we will enable payments using cards, Apple Pay, Google Pay or other means of payment.

Use Case 2. **Payout to card** - in this case we assume that your user has an account and money on this account. We need to enable payments from this account to any card in the world. **This money**

transfer will be very quick - less than 30 minutes. In this case you should be using our product called "Payouts to cards". This will enable your user to transfer money to any Mastercard or VISA card.

Use Case 3. **Card-2-Card** - in this case our assumption is that you do not have a user's account. You do not store the money of your users. You just want to **enable a money transfer service from one card to another card**. From Mastercard to VISA, VISA to Mastercard or Mastercard to Mastercard or VISA to VISA card. In such a case we recommend that you use our card-2-card product.

It is important not to mix those use cases and choose the correct product. All three products have different fees, AML requirements so please think of your use case and let's decide what to use.

Thanks for reading.

Article

Cross-Border Card Transactions – Questions about FX Rates and Cash Flow

The topic of settlements of cross-border card transactions is often touched by our customers. It is not always clear how various fees are calculated. In this article we would like to explain how it works in detail.

The standard transaction flow

In general, a **cross-border transaction** works in the same way as a **domestic transaction**:

1. The cardholder presents the card to the merchant or terminal (either online by entering card data, or in a contactless way or as a standard plastic card transaction or at an ATM).

2. The acquirer gets card and transaction data from the terminal.

3. Based on the **BIN** table, the acquirer sends transaction authorization to **Mastercard or VISA (Payment Schemes)**.

4. The payment scheme transfers authorization to the card issuer.

5. The card issuer approves or declines the authorization and blocks the amount on the cardholder account.

6. In case of approved transactions, the acquirer prepares a clearing file and sends it for settlement.

7. The payment scheme receives a clearing file and processes it.

8. The payment scheme takes money from the issuer settlement account and transfers it to the acquirer settlement account.

9. The payment scheme settles fees between itself, the issuer and the acquirer.

10. The merchant receives money for the transaction.

11. The acquirer charges merchant fees.

The added complexity of currency conversion in cross-border transactions

As mentioned above, this process is the same for domestic and cross-border transactions. However, in cross-border transactions the currency conversion process takes place, bringing an additional layer of complexity. As conversion applies at various moments of transaction, let me describe this in detail below:

1. Merchant and Acquirer

a. The merchant can agree with the acquirer that they want to **receive money in one or multiple currencies**. Depending on this contract, the acquirer will transfer money to the merchant in those currencies. It is impossible that ALL currencies in the world will be used, so usually merchants want to receive money in a few main currencies

b. The acquirer has **Settlement Services** with payment schemes in several or many currencies.

c. Depending on the Settlement Services, if a transaction is performed in currency X, the acquirer receives money in currency X. No currency conversion cost will apply at a payment scheme.

d. However, if the transaction is performed in currency X, but the acquirer does not have Settlement Service in this currency, the payment scheme will convert currency X to currency Y and send money to the acquirer in currency Y.

e. The acquirer will receive currency Y on their bank account and will either convert it to the merchant settlement currency or will transfer it directly to the merchant in currency Y.

f. Various fees charged by the acquirer can apply if the acquirer is performing currency conversion.

2. Payment Scheme

a. As described above, acquirers have **Settlement Services enabled by Payment Schemes**. Issuers have the same.

b. The issuer can have Settlement Service in main currencies:

i. Local currency

ii. EUR settlement

iii. USD settlement

iv. Eventually in other currencies

c. Every new Settlement Service is a paid service, so both issuers and acquirers must decide which is the correct setup of settlement services. Depending on this decision, the Payment Scheme will perform more or less currency conversion operations and will earn fees during this process.

d. The Payment Scheme provides issuers and acquirers with currency conversion tables which can act as a **directional FX rate** for them. However, the **real FX rate** for a particular transaction is performed at the moment of transaction settlement at the Payment Scheme.

e. This means that if the cardholder performs a transaction, his/her issuer is never sure what will be the settlement amount for this transaction

3. Issuer and Cardholder

a. Finally, depending on the Settlement Service agreed with the Payment Scheme, the issuer enables cards for their users.

b. The cardholder has usually (actually always) a payment account connected with this card and the cardholder knows that he/she holds money in a particular currency enabled by his/her bank / issuer.

c. It means that all transactions on this payment account will be charged in this particular currency.

d. Depending on the currency of the transaction and the settlement service enabled by the payment scheme, the issuer will perform the currency conversion and charge the cardholder additional fees.

Conclusion

This topic is highly complex and depends on many factors. Issuers and acquirers must make a decision which approach is the best for them, which Settlement Services to enable with **Mastercard or VISA** and what should be the fee charged for this service because it is impossible to avoid all risks connected with currency conversion changes.

Introduction

Payout to card is a method that allows making a payment by providing the recipient’s card details. This type of transfer does not require a card on the sender’s side. The sender simply selects the balance from which the operation should be executed and provides the card details to which the payout should be sent. If the sender has sufficient funds in the balance and the recipient’s card is valid, the payout is executed instantly.

How to connect with us?

Our onboarding process can be divided into two stages: business and technical.
The first stage is the business onboarding, which involves signing the required documents with our Sales Department.

The second stage is the technical onboarding, which involves integrating with our API.

> **Note:** It is required that you have an account in Acquirer's system which will settle your transactions. For more informations please contact our Sale Department. We are suggesting to use Fenige - our partner acquiring institution.

How to integrate - API reference

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)
note right of p: Sender chooses his balance \nand provides receiver's card
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
note left of v: We contact with acquiring institution \nfor 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"
  },
  "additionalData" : {
    "note" : "Restaurant. Tip for Joe Doe"
  },
  "transactionReason" : "Test document transaction"
}
```

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
`additionalData`	object	Object allowing the transaction initiator to add an additional information of the transaction.
`additionalData.note`	string **required**	Note for the transaction. @Length(min = 0, max = 150), @Pattern(regexp = `^(?!\s*$)(?!.*[\r\n]).+$`)

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"
}
```

> **Note:** If the fund are transferred by a **company,** all sender fields (including name, address, and contact details) must contain the company's information, not the personal details of an individual. The data should be provided in accordance with the requirements outlined in the table below.

Business Name	firstName	lastName
Company	Company	Company
Fine Company	Fine	Company
Big Company International	Big	Company Integrnational

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"
}
```

Onboarding

This chapter is intended to present you the requirements that will allow you to use the Payout to Card solution in your Company. We have presented here what information is necessary to provide so that you can join the Payout to Card program and so that we can properly create the required corporate balance account for you thus allowing you to funds management and card issuing.

Business onboarding

To start using Money Transfers via Card you need to go through a few on-boarding steps:
1. Please contact our sales - *salesteam@verestro.com*
2. Please respond to some introduction question that will let us prepare proposal for you.
3. You will receive offer for card issuing and payouts processes.
4. If you accept the offer you will be asked to provide some company documents required for the AML verification process.
5. You need to perform KYB process to pass verification of your company.
6. And finally you will enable you your own balance in Verestro Payout to Card program.

Technical onboarding

After completing the business onboarding process, you will be granted the appropriate access. This will allow you to integrate with the API methods provided by us, as described in the "How to integrate – API reference" section.

> **Tip:** Please remember to inform us on which environment you want us to configure an account for you. Verestro Payout to Card offers two environments: `TEST` and `PROD`.

> **Important!** Implementation of the `PROD` environment is work in progress...