Your APIs for us - External Balance

External Balance

External Balance API is used, when client wishes to keep end users’ balances on their side. Thanks to this API, a client who maintains his clients accounts or has his own business logic affecting transaction authorizations has the opportunity to expand his offer with various payment instruments offered by Verestro, including payment cards, bank transfers, transfers to a phone number and others. Workflow is reversed when using External Balance API - Antaca is sending request to server on client’s side.

Features

• linking (connecting) balance with customer in Antaca,
• getting list of balances,
• deleting balance link (connection),
• handling transactions,
• updating transaction status.

Purpose and scope

This guide provides an instruction and case study for using Extrnal Balance API. Document covers following topics:
• how to use External Balance API,
• transaction flow,
• how clearings are handled,
• use cases study.

Terminology

User - The end user for whom a balance is maintained along with the associated payment instruments.
Server - API exposed by Antaca’s client.
Client - company using Antaca services.

> To use external balance you must have Banking License or Payment Institution License. Additionally if you don't directly own BIN range, total sum of transactions of your users will be limited by deposit called "Master Balance".

Security

To set secured server-server connection between our services Verestro requires a similar connection as in the case of client to Verestro communication based on the x509 certificate.
In the first step, Verestro will send to the client a CSR for the dev and production environments. The next step is for the client to sign the CSR and send the certificate back to Verestro along with the base URL for the methods listed below. Verestro will authorize itself with each request with a certificate, which should be checked on the client side.

Idempotency Key

With some requests additional header X-Idempotency-Key could be send. This header contain unique random id allowing to identify single request.
If client send this header, operation should be triggered only once and for any further request with this key, response should be identical - in most cases, returned from cache.

example headers:

```
X-Idempotency-Key: 20e87975-dbfb-4c95-b239-169516c0b707
```

External Balance API

Below you will find a list of endpoints that you should implement on your server side. Please pay special attention to the appropriate security of our connection, the syntax of requests that you can expect from the Verestro side, idempotency and the exact way in which you should respond to each request.

Process of linking balances

After establishing secure connection, the client should create balance aliases for their users on the Antaca side. For identifiers created in this way, Antaca will be able to generate payment cards and process transactions. When the client creates a user on the Verestro side with confirmed KYC status and then orders the creation of a balance for him, this API will forward the request to link the balance to the user. The linking process is presented in the diagram below.

@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 "Client server" as cs
participant Antaca as a
participant Lifecycle as lc
cs->lc: 1. create user by POST /wallet (firstName, lastName, phone, email)
lc-->cs: 2. userId
alt With own KYC process
cs->lc: 3. update KYC status by PUT /user (KYC)
else With Verestro KYC process
cs->a: 4. send KYC data by /register (user data with documents and selfie)
a->a: 5. process KYC
a->lc: 6. update KYC status
end
alt With automatic balance creation
lc-->a: 7. event about the new user with KYC
a->a: 8. create a balance in the default currency
a->cs: 10. link balance (userId, balanceId, currency)
cs-->a: 11. 204 (OK)
else Every time with create a balance
cs->a: 9. create balance by POST /balance (userId, currency)
a->cs: 10. link balance (userId, balanceId, currency)
cs-->a: 11. 204 (OK)
a-->cs: 12. 201 balance created
end

@enduml
```

Once balances are linked, Antaca can:

- send a GET requests to retrieve information about specific balance. Using user ID and/or balance ID, Antaca can obtain information about balance currency and money amount.
- send a GET requests to retrieve information about all user's balances. Using only user ID Antaca can retrieve list of users balances.
- delete link between balances

When user's balance is equal to 0, it can be unlinked and deleted on Antaca side.

> **Remember to avoid billing problems**
> Deleting a balance is only possible when its status is 0. This also applies to situations in which a user with at least one balance is deleted.

Transaction processing

When a balance is created and linked for a given user with verified KYC, from that moment the client-side API should be ready to accept transactions related to it. Depending on the payment instrument used, the data transferred in the transaction object may be different, but will always refer to a specific balance.

> **Remember to avoid communication errors**
> Verestro servers attach an X-Idempotency-Key to each request in the header. This header contains a unique ID for each request to ensure idempotence. Each request with a unique identifier in this header should be processed only once on the client side and the response to it should be identical - in most cases, returned from cache

example:

```
curl POST "https://server-domain.com/transactions/debit"
--header "X-Idempotency-Key: 21aa0c2a-5554-4071-bd48-b9c64a0b6270"
```

Transaction object

Each transaction request contains following data:

```
{
    "id": "b4f534ef-77c2-4f16-ab4d-496806a76fb6",
    "balanceId": "b334b384-328c-11ed-a261-0242ac120002",
    "resourceId": "9d673932-3291-11ed-a261-0242ac120002",
    "resource": "card",
    "transactionId": "ab3d89e4-3291-11ed-a261-0242ac120002",
    "referenceTransactionId": "b759931c-3291-11ed-a261-0242ac120002",
    "type": "POS",
```

```
    "amount": 10000,
    "currency": "PLN",
    "originalAmount": 10000,
    "originalCurrency": "PLN",
    "status": "AUTHORIZED",
    "description": "transaction description",
    "date": "2020-08-17T18:43:42+00:00",
    "transactionData": {
        "mcc": "5942",
        "merchantIdentifier": "003060300000005",
        "merchantName": "Book store",
        "captureMode": "NFC",
        "lastFourDigits": "4560",
        "acquirerCountry": "POL",
        "mdesDigitizedWalletId": "Google Pay",
        "cashbackPosCurrencyCode": "PLN",
        "cashbackPosAmount": 10000,
        "lastFourDpan": "7890",
        "adjustmentReasonDescription": "REFUND",
        "retrievalReferenceNumber": "749248185012",
        "cardId": "6876783"
    }
}
```


Parameters:

Parameter	Required	Description	Allowed values
id	TRUE	Unique identifier of the transaction in UUID format	any value in UUID v4 format, eg. ddb55ff9-11ca-4621-9129-81f939e66011
balanceId	TRUE	Unique user balance identifier	any string value (recommended uuid v4)
resourceId	TRUE	Unique resource identifier	any string value (recommended uuid v4)
resource	TRUE	Name of a resource.	balance, card

transactionId	TRUE	Transaction identifier obtained from card network or generated on client side using the method to generate transaction in Antaca. **IMPORTANT: this id may not be unique - it is generated by different systems**	any string value
referenceTransactionId	FALSE	Id of previous transaction to with current request relates	any string value (recommended uuid v4)
type	TRUE	Type of transaction	cashback, loan, payment, topup, commission, fee, funding, interest, withdrawal, pos, atm, cashback_at_pos, adjustment
amount	TRUE	Transaction value in gross (minor value) For example: 12.34 EUR will be sent as 1234	integer value
currency	TRUE	Currency 3-letters code in ISO 4217	ISO 4217 3-letter code
originalAmount	FALSE	Original transaction value in gross (minor value) For example: 12.34 EUR will be sent as 1234	integer value
originalCurrency	FALSE	Original currency 3-letters code in ISO 4217	ISO 4217 3-letter code
status	TRUE	Transaction status	AUTHORIZED, CLEARED, REVERSED
description	TRUE	Transaction description	any string value
date	TRUE	Date of transaction in UTC	date in UTC
transactionData	FALSE	Additional transaction data. This object presents detailed data depending on the transaction type	This object is described below

TransactionData data object:

Name	Required	Description	Allowed values

mcc	FALSE	Merchant category code	any mcc value, eg. can be found here: https://global.alipay.com/docs/ac/files/mcclist
merchantIdentifier	FALSE	The merchant identifier for the transaction	
merchantName	FALSE	Name of merchant	
captureMode	FALSE	Capture mode	magstripe, manual, emv, on behalf (EMV), nfc, ecommerce, adj
lastFourDigits	FALSE	Last 4 digits of card	
acquirerCountry	FALSE	Country of acquirer	alpha-3
mdesDigitizedWalletId	FALSE	The Wallet ID (Wallet Reference) used to digitize the card.	m4m, google pay, samsung pay, apple pay
cashbackPosCurrencyCode	FALSE	Represents the currency code of the cashback amount	ISO 4217 3-letter code
cashbackPosAmount	FALSE	Displays the actual cashback amount	integer value in gross
lastFourDpan	FALSE	Last 4 digits of Device Primary Account Number (tokenized PAN)	
adjustmentReasonDescription	FALSE	Reason for adjustment	eg. REFUND, MONEY_SEND, CHARGEBACK
retrievalReferenceNumber	FALSE	12-digit number generated to record each transaction	
cardId	FALSE	The card identifier in string format. This value could be used to communicate with the Antaca services.	any string value. Mostly it should be eg. "1234" but it can change in the future and become UUID format.

Integration with External Balance

By using External Balance API, you take an active part in processes affecting settlements, therefore the API issued by you will be subject to approval before production starts. To make this process easier, we can share with you a Postman collection with test cases.

API External balance

> For the process to function correctly, the Client **must** implement all endpoints detailed in this chapter.

Below you will find a list of endpoints that you should implement on your server side. Please pay special attention to the appropriate security of our connection, the syntax of requests that you can expect from the Verestro side, idempotency and the exact way in which you should respond to each request. If you decide to implement external balance to be able to keep the balance on your side and authorize transactions, remember that the implementation of all the methods below is required to ensure the API works.

Link balance

This method is used to link customer balance between client and server. Requested **balanceId** will be used for communication between client and Verestro side.

> If you create balance entity at your end you should create it after receiving this call. Do not create balance entity on your side as result of POST /secure/balances nor POST /secure/customers/{id}/balances, because link balance will be called before response to these methods.

```
POST https://server-domain.com/users/:id/balances
```

path parameters:

id - user identifier

Name	Required	Description	Allowed values
balanceId	TRUE	Unique identifier of balance. This ID will be used in communication between client and server.	UUID v4
currency	TRUE	Currency code	should be 3 letters code in ISO 4217 https://www.iban.com/currency-codes

Headers:

```
Content-Type: application/json
Accept: application/json
```

request body:

```
{
    "balanceId":"2e520dc2-329d-11ed-a261-0242ac120002",
    "currency": "PLN"
}
```

parameters:

Name	Required	Description	Allowed values
balanceId	TRUE	Unique identifier of balance. This ID will be used in communication between client and server.	UUID v4
currency	TRUE	Currency code	should be 3 letters code in ISO 4217 https://www.iban.com/currency-codes

response:

```
204 NoContent
```

error codes:
404 - should be returned if no user has been matched by requested path parameter.

```
Code 404
{
    "title": "USER_NOT_FOUND",
    "detail": "some specific details provided by server"
}
```

Get single user balance

Method used to obtain single user balance information.

```
GET https://server-domain.com/users/:id/balances/:balanceId
```

path parameters:

Name	Required	Description	Allowed values
id	TRUE	User identifier.	Integer
balanceId	TRUE	Unique identifier of balance. This ID will be used in communication between client and server.	UUID v4

```
id - user identifier
balanceId - unique balance identifier
```

headers:

```
Accept: application/json
```

response:

```
200 OK
{
    "currency": "PLN",
    "amount": 250
}
```

response parameters:

Name	Required	Description	Allowed values
amount	TRUE	Actual balance amount in minor (penny)	integer value. For example: 12.34 EUR will be sent as 1234
currency	TRUE	Currency code	should be 3 letters code in ISO 4217 https://www.iban.com/currency-codes

```
currency  - three letter iso 4217 code
amount - actual balance amount in minor (penny), integer value. For example: 12.34 EUR will be
```

```
sent as 1234
```

error codes:
404 - should be returned if no balance found by requested balanceId.

```
Code 404
{
    "title": "BALANCE_NOT_FOUND",
    "detail": "some specific details provided by server"
}
```

403 - if requested balance does not belong to user.

```
Code 403
{
    "title": "FORBIDDEN",
    "detail": "some specific details provided by server"
}
```

Get balance collection

This method should return collection of customer balances.

```
GET https://server-domain.com/users/:id/balances
```

path parameters:

Name	Required	Description	Allowed values
id	TRUE	User identifier	Integer
balanceId	TRUE	Unique identifier of balance. This ID will be used in communication between client and server.	UUID v4

```
id - user identifier
```

headers:

```
Accept: application/json
```

response:

```
200 OK
[
    {
        "id": "a072bd0e-328c-11ed-a261-0242ac120001",
        "currency": "PLN",
        "amount": 250
    },
    {
        "id": "b334a5e2-328c-11ed-a261-0242ac120002",
        "currency": "USD",
        "amount": 460
    }
]
```

If user has not created any balance yet, there should be returned empty collection.

```
200 OK
[]
```

response parameters:

Name	Required	Description	Allowed values
id	TRUE	Unique identifier of balance. This ID will be used in communication between client and server.	UUID v4
balanceId	TRUE	Unique identifier of balance. This ID will be used in communication between client and server.	UUID v4
currency	TRUE	Currency code	should be 3 letters code in ISO 4217 https://www.iban.com/currency-codes

```
id - unique identifier of user balance
currency  - three letter iso 4217 code
amount - actual balance amount in minor (penny), numeric value
```

Delete balance

This method is used to unattached balance from user. From legal point of view, balance should be deleted only if there is no money on it.

```
DELETE https://server-domain.com/users/:id/balances/:balanceId
```

path parameters:

Name	Required	Description	Allowed values
id	TRUE	User identifier	Integer
balanceId	TRUE	Unique identifier of balance. This ID will be used in communication between client and server.	UUID v4

response:

```
204 No Content
```

error responses:

404 - if requested balance has not been found.

```
Code 404
{
    "title": "BALANCE_NOT_FOUND",
    "detail": "some specific details provided by server"
}
```

403 - if requested balance does not belong to user.

```
Code 403
{
```

```
    "title": "FORBIDDEN",
    "detail": "some specific details provided by server"
}
```

Debit transaction

This kind of transaction is used to authorize transaction. In debit transactions Antaca asks 'if user has money?'.

```
POST https://server-domain.com/transactions/debit
```

headers:

```
Content-Type: application/json
X-Idempotency-Key: uuidV4
```

request body:

Description of the contents of the transaction object can be found above.

success response:

```
204 No Content
```

error responses:

```
422
{
    "title": "INSUFFICIENT_FUNDS",
    "detail": "some specific details provided by server"
}
```

```
422
{
    "title": "LIMITS_EXCEEDED",
    "detail": "some specific details provided by server"
}
```

```
422
{
    "title": "FRAUDS_DETECTED",
```

```
    "detail": "some specific details provided by server"
}
```

```
404
{
    "title": "BALANCE_NOT_FOUND",
    "detail": "some specific details provided by server"
}
```

```
409
{
    "title": "CLIENT_ERROR",
    "detail": "some specific details provided by server"
}
```

Force debit transaction

This kind of transaction is used to inform server side that transaction has occurred.
For this request, actual transaction already happen so server can not reject this request. This behavior can occur for offline transactions fe: in plane, subway, for refunds and referring to previously authorized transactions.

```
POST https://server-domain.com/transactions/force-debit
```

headers:

```
Content-Type: application/json
X-Idempotency-Key: uuidV4
```

request body:

Description of the contents of the transaction object can be found above.

response:

```
204 No Content
```

error response:

As mention in description section, **we do not accept transaction rejection**.

Credit transaction

Method is used to credit user balance. In credit transactions Antaca asks 'can user get money?'.

```
POST https://server-domain.com/transactions/credit
```

headers:

```
Content-Type: application/json
X-Idempotency-Key: uuidV4
```

request body:

Description of the contents of the transaction object can be found above.

success response:

```
204 No Content
```

error responses:

```
404
{
    "title": "BALANCE_NOT_FOUND",
    "detail": "cannot find requested balance"
}
```

```
422
{
    "title": "FRAUDS_DETECTED",
    "detail": "some specific details provided by server"
}
```

Force credit

Method is used to credit user balance. This kind of transaction is used to inform server side that transaction has occurred. For this request, actual transaction already happen so server can not reject this request.

```
POST https://server-domain.com/transactions/force-credit
```

headers:

```
Content-Type: application/json
X-Idempotency-Key: uuidV4
```

request body:

Description of the contents of the transaction object can be found above.

success response:

```
204 No Content
```

error response:

As mention in description section, **we do not accept transaction rejection**.

Reversal

Method is used to revert any changes for previous transaction. Request body will be identical to transaction with client try to revert. If server cannot find referenced transaction then no action is required.

```
POST https://server-domain.com/transactions/reversal
```

headers:

```
Content-Type: application/json
X-Idempotency-Key: uuidV4
```

request body:

Description of the contents of the transaction object can be found above.

success response:

```
204 No Content
```

error responses:

IMPORTANT: for this method, we do not accept any error. Only satisfying behavior is to revert referenced transaction and no action if cannot find transaction.
Therefore, if a transaction is not found on the partner's side, we expect not an HTTP 404 ERROR but an HTTP 2xx SUCCESS in this case.

It tells the Partner directly to reverse the transaction as if it never happened (it was rejected by Antaca, idle timeout, etc.). Usually, there will be no referenceTransactionId in it, but there are cases where there will be. Reversal does not refund the money. It only confirms that such a transaction did not take place. The money should be refunded by force-credit.

Update transaction status

From time to time, client will inform about clearings triggered by acquirer side. If client mark transaction as cleared it means that transaction will not be corrected by any other transaction request and requested amount is final.

> This endpoint is used to inform about the change of the transaction status to CLEARED - the movement of funds should not occur here.

```
PUT https://server-domain.com/transactions/:transactionId
```

path parameters:

Name	Required	Description	Allowed values
transactionId	TRUE	Transaction identifier obtained from card network or generated on client side using the method to generate transaction in Antaca. **IMPORTANT: this id may not be unique - it is generated by different systems**	any string value

request body:

Description of the contents of the transaction object can be found above.

success response:

```
204 No Content
```

error responses:

```
404
{
    "title": "TRANSACTION_NOT_FOUND",
    "detail": "cannot find requested transaction"
}
```

Transaction Types Description

Debit transactions list:

Type	Description
POS	POS transaction (A point-of-sale) applies to the situation when a customer makes a purchase and the payment is processed through the POS system.
ATM	ATM Transaction is when the cardholder uses a physical card at an ATM to withdraw cash.
Balance Inquiry	Check the available balance of funds.
Commission	internal transaction for a partner who wants to debit user balance as a commission referenced to the other transaction.
Fee	internal transaction for a partner who wants to debit user balance as a fee. Antaca automatically **credits** company balance with the funds that were debit the user's balance
Funding	internal transaction type used to debit the user's balance. This type indicates that the funds still remain in the Antaca system, usually in conjunction with a payment type a credit transaction on the user's balance. Antaca automatically **credit** the **credit partner balance** with this transaction
Interest	internal transaction for a partner who wants debit the user's balance as part of the interest connected with credit agreement.
Withdrawal	internal transaction type used to debit the user's balance. This type indicates that the funds go outside the Antaca system, fe: withdrawal from an account at a bank branch.

Credit transactions list:

TopUp	internal transaction type used to top up the user's balance. This type indicates that the funds come from outside the Antaca system, fe: payment to an account at a bank branch. Antaca automatically **debit** the **credit partner balance** with this transaction
Payment	internal transaction type used to top up the user's balance. This type indicates that the funds come from the Antaca system, usually in conjunction with a funding type a debit transaction on the user's balance Antaca automatically **debit** the **credit partner balance** with this transaction
Loan	internal transaction for a partner who wants to top up the user's balance as part of the credit agreement. Antaca automatically **debit** the **credit partner balance** with this transaction
CreditIbanTransfer	internal transaction dedicated only for IMS API (via specific CN). IMS API uses this balance to credit funds on the user's balance.
Cashback	internal transaction for a partner who wants to top up the user's balance as part of the loyalty program Antaca automatically **debit** the **credit partner balance** with this transaction

Revision #15
Created 2024-12-03 18:02:56 UTC
Updated 2025-11-13 12:04:00 UTC