Your APIs for us - Shared authorization

Shared authorization

The Shared Autorization API is a tool that gives our Clients control over the transaction authorization process. Through this API, a Client can make the **authorization decision** based on their own business rules, even when end users' funds are held and managed on Verestro side.

In this model, the Client's system, via a dedicated endpoint, receives an authorization request from **Antaca** and responds with either an approval or a decline. A positive response from the client is our signal to debit the funds from the balance in our system and finalize the payment, provided that other transactional conditions on Verestro's side are met (e.g., sufficient funds or AML rules).

> A positive authorization decision from the Client doesn't guarantee the transaction will be successful, as it can still be DECLINED by Verestro for other reasons.

Features

• transactions processing

Purpose and scope

This guide provides an instruction and case study for using Shared Autorization API. Document covers following topics:
• how to use Shared Authotization API,
• transaction flow,
• 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.

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
```

Shared Authorization 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.

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 witch 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	

Name	Required	Description	Allowed values
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 Shared Authorization

By using Shared Autorization 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 Shared Authorization Details

> 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 Shared Authorization API, remember that the implementation of all the methods below is required to ensure the API works.

Debit transaction

This kind of transaction is used to authorize transaction.

```
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.

```
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 witch 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.

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.

Revision #7
Created 29 August 2025 13:36:31
Updated 15 October 2025 14:02:40