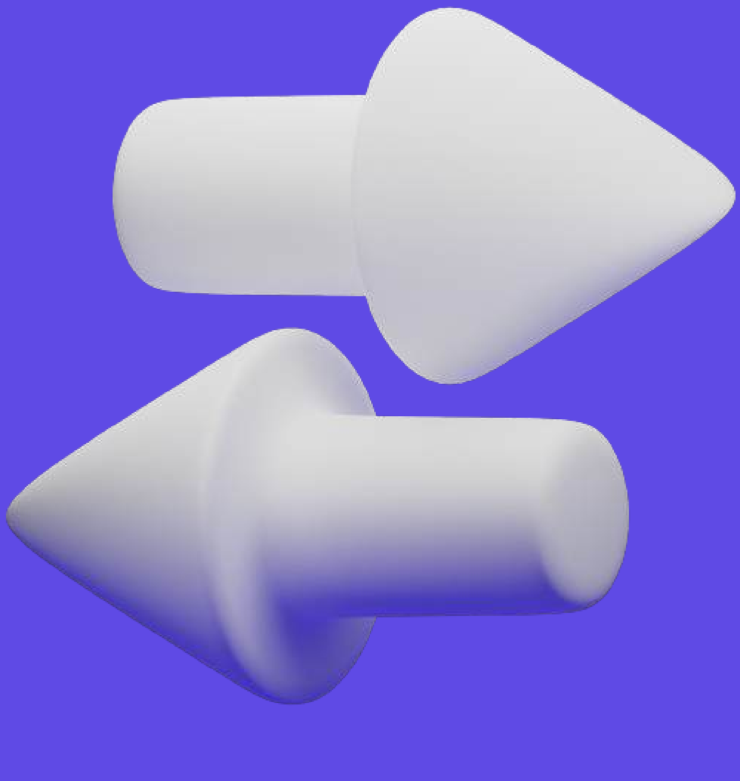
genome

How Genome Sets Up Incoming Payment Notification Service Via Webhooks

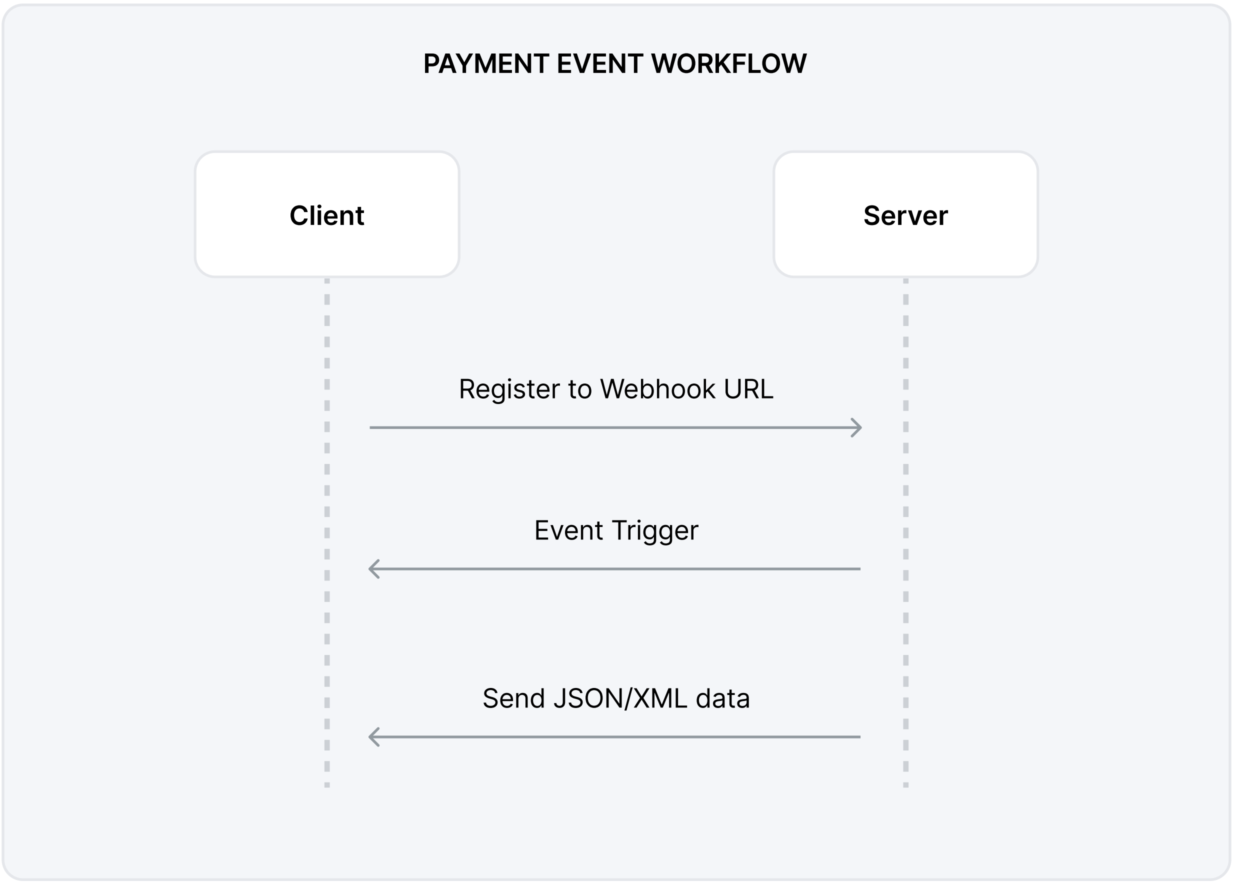


Introduction

Genome provides real-time notifications for incoming payments via API using webhooks. This service helps companies to efficiently track incoming payments and enables them to automate internal or customer communication.

This payment notification service comes in the form of push notifications sent as webhook messages to URLs a business defines.

How it works: a company subscribes to a payment event. Once the event is triggered, Genome sends an HTTP POST request to a URL of the company's choice in a JSON format. It also includes data about the payment in addition to the event. This data is called **payload**.



How to set up webhooks inside Genome

1. Open a business wallet inside Genome. If necessary, use our detailed tutorial on how to do so.
2. Contact Genome's account managers at accounts@genome.eu and request the webhook setup.

After registration for the webhook subscription, Genome will provide you with `secretData` (variable-length binary data), which you will use to receive data.

To set up a webhook, you'll need to provide Genome with the following information:

Information	Format	Description
`URL`	`VARCHAR`	This is a URL of your choice to receive data
`attemptMax`	`UInt16`	The number of attempts available to deliver data
`Direction`	`enum(incoming_only)`	`Incoming_only` - Incoming transactions that increase balance, credit transactions. You send the enum parameter that indicates the direction of funds.
`Outcome`	`enum(success_only)`	`Success_only` - information about the successful transactions. You send the enum that indicates the transaction status.

How Genome sets up incoming payment notification service via webhooks

Notes for company's developers' team

1. The creation of the payload (data about the payment)

Callback to a transaction consists of the following parts:

- Headers
- Callback data

Callback headers

There are four mandatory headers sent by Genome in every callback:

Header	Description
`User-Agent`	Custom user agent data (configurable in external systems config), so the receiver's infrastructure may perform whitelisting. Proposal is Genome Callback System
`Content-Type`	application/json
`X-Version`	Callback version. Current value is 1.
`X-Signature`	HMAC signature for callback

Callback signature sent in header is produced by HMAC SHA256 algorithm using configured (for customer) secret and full callback HTTP request body.

Example headers

```
{
USER-AGENT: ”Genome Callback System”,
X-SIGNATURE: “b8576caa04cc74984fe1ef69aa4cf086b26a69300ddf5285347629d”,
X-VERSION: “1”
}
```

Callback shared data

Shared data is a chunk of the callback body that contains transactional information of all transactions.

Body (JSON array)	Description
reference	*transaction (if transaction data being sent in callback references another transaction, all the data about that transaction is placed into callback)
bulk_id	*uint64
payment_id	uint64
transaction_id	uint64
transaction_type	uint64
transaction_status	uint64
created_at	uint64
booked_at	uint64
processed_at	uint64

Body (JSON array)	Description
`amount` — `amount` — `currency`	object decimal decimal
`description`	string
`error` — `code` — `message`	object uint32 string
`sender` — `wallet_id` — `account_id` — `iban` — `bic` — `name`	Participant: Null Null *string Filled if possible *string Filled if possible *string Filled if possible and not empty
`receiver` — `wallet_id` — `account_id` — `iban` — `bic` — `name`	Participant: *uint64 *uint64 *string Filled if possible *string Filled if possible *string Filled if possible and not empty

Example of callback shared data

```
{
  "transaction_id": ”12214”,
  "transaction_type": "SEPA_INSTANT_INCOMING",
  "transaction_status": "SUCCESS",
  "created_at": "2024-11-07T11:47:31Z",
  "booked_at": "2024-11-07T11:47:31Z",
  "processed_at": "2024-11-07T11:47:33Z",
  "amount": {
    "currency": "EUR",
    "amount": “1.0”
  },
  "sender": {
    "iban": "LT583003000000012345",
    "bic": "RECEIVERXXX",
    "name": "Sender Name 1"
  },
  "receiver": {
    "wallet_id": “1050000000029590”,
    "account_id": “1051097800000021139”,
    "iban": "LT411010058291544307",
    "bic": "RECEIVER002",
    "name": "Receiver Name 1"
  }
}
```

2. The implementation of the retry strategy

In cases where notifications cannot successfully reach the customer, Genome will retry delivery with an exponential backoff for every failed delivery. A number of attempts are provided as ***attemptMax*** value, which the business chooses to set.

Your endpoint must respond with the ***http 200 ok*** status code to indicate successful receipt of the notification. After the last unsuccessful retry, Genome will stop sending notifications.