HookFrame – Webhook Processing Framework

4. webhook.php

The generic entrypoint that accepts **any** HTTP POST (or other) requests. It does *not* validate tokens or assume JSON bodies. It simply:

1. Loads environment variables via `safeLoad()` (falls back to real env vars).
2. Connects to RabbitMQ and declares the target queue.
3. Reads the raw HTTP request body as a string (no parsing).
4. Wraps it in a JSON envelope with fields:
    - `source` : from `WEBHOOK_SOURCE` env or `hookframe`
    - `event` : from `WEBHOOK_EVENT` env or default `event`
    - `timestamp` : current UTC in ISO 8601
    - `payload` : the raw request body string
5. Publishes the envelope (persistent) to RabbitMQ.
6. Echoes `OK` back to the client.

```
 <?php
// webhook.php

require_once __DIR__ . '/vendor/autoload.php';

use Dotenv\Dotenv;
use PhpAmqpLib\Connection\AMQPStreamConnection;
use PhpAmqpLib\Message\AMQPMessage;

// Load .env (if exists), else use system env
$dotenv = Dotenv::create(__DIR__);
$dotenv->safeLoad();

// Connect to RabbitMQ
$conn    = new AMQPStreamConnection(
    getenv('RABBITMQ_HOST'),
    getenv('RABBITMQ_PORT'),
    getenv('RABBITMQ_USER'),
    getenv('RABBITMQ_PASSWORD')
);
$channel = $conn->channel();
```

```
$channel->queue_declare(getenv('RABBITMQ_QUEUE'), false, true, false, false);

// Read raw body
$rawBody = file_get_contents('php://input');

// Build envelope
$envelope = [
  'source'    => getenv('WEBHOOK_SOURCE') ?: 'hookframe',
  'event'     => getenv('WEBHOOK_EVENT')  ?: 'event',
  'timestamp' => gmdate('c'),
  'payload'   => $rawBody
];

// Publish envelope
$msg = new AMQPMessage(
  json_encode($envelope),
  ['delivery_mode' => AMQPMessage::DELIVERY_MODE_PERSISTENT]
);
$channel->basic_publish($msg, '', getenv('RABBITMQ_QUEUE'));

echo "OK";
?>
```

5. Envelope Details

Field	Description
`source`	Identifier of the sender (env `WEBHOOK_SOURCE` or `hookframe`).
`event`	Event name (env `WEBHOOK_EVENT` or default `event`).
`timestamp`	UTC time when envelope was created.
`payload`	Raw request body as a string; handlers decide how to parse it.

6. Handler Responsibility

Individual handlers (e.g. `ExampleHandler`) receive the envelope and can decide:

- Whether to parse `payload` as JSON, XML, form-data, etc.
- How to filter by `source` or `event` .
- Business logic and error handling.

7. Consumer & Retry Logic

See `consumer.php` for manual ACK, retry count `_retry`, and re-publish logic.

8. Configuration

```
 # .env.example
RABBITMQ_HOST=localhost
RABBITMQ_PORT=5672
RABBITMQ_USER=guest
RABBITMQ_PASSWORD=guest
RABBITMQ_QUEUE=queue_webhooks

# optional defaults for source & event
WEBHOOK_SOURCE=hookframe
WEBHOOK_EVENT=event

RETRY_LIMIT=3
```

9. License

MIT — see the `LICENSE` file.