

Webhooks

Receive signed HTTP notifications when invoices are reported, paid, received and more.

Source: <https://alyabooks.ai/docs/guides/webhooks>

Webhooks push events to an https endpoint you control. Create them in the app under **API access** → **Webhooks** or with the API (scope `webhooks:manage`).

What an invoice event carries

- Every `invoice.*` payload has the invoice header (id, number, type, status, dates, currency, totals, buyer, `isTest`, `metadata`, `externalId`) and two blocks that let a receiver act without calling back:
- `validation`: `validity` and the blocking `errors[]` from the last check, each with `rule`, `field`, `message` and a public `docsUrl`.
 - `rejection`: on `invoice.rejected`, the `stage` (`taxilla`, `buyer` or `fta`), the provider's `error` text, the `submissionId` and when it happened; `null` otherwise.

For the explained version of each error (API field path, English and Arabic explanation and fix) call `GET /invoices/{id}/validation`.

Create a subscription

```
curl -X POST "$BASE/webhooks" -H "X-API-KEY: $KEY" -H "Content-Type: application/json" -d '{
  "name": "ERP sync",
  "targetUrl": "https://erp.example.com/hooks/einvoicing",
  "events": ["invoice.reported", "invoice.rejected", "invoice.paid",
"incoming.validated"],
  "headers": { "Authorization": "Bearer my-inbound-token" }
}'
```

The response includes `secret` (`whsec_...`) **once**. Use `["*"]` to subscribe to every event. Targets must be public `https` hosts; private networks, localhost and cloud metadata addresses are rejected, and DNS answers pointing to private ranges are refused at delivery time.

Events

Event	When
<code>invoice.created</code>	An invoice, credit note or debit note was created (draft or validated)

Event	When
<code>invoice.sent</code>	Emailed or logged as sent
<code>invoice.submitted</code>	Handed to the ASP
<code>invoice.delivered</code>	Accepted by the buyer's access point
<code>invoice.reported</code>	Accepted by the FTA (status <code>accepted</code>)
<code>invoice.rejected</code>	Rejected by the ASP, the buyer's access point or the FTA
<code>invoice.paid</code>	Payments cover the full amount
<code>invoice.overdue</code>	The daily job flagged the invoice as overdue
<code>invoice.deleted</code>	A draft was deleted
<code>validation.completed</code>	An API-triggered validation finished (<code>validate: true</code> or <code>POST .../validate</code>)
<code>incoming.created</code> , <code>incoming.validated</code> , <code>incoming.accepted</code> , <code>incoming.rejected</code> , <code>incoming.paid</code>	Received documents
<code>quote.created</code> , <code>quote.sent</code> , <code>order_confirmation.created</code> , <code>proforma.sent</code> , <code>reminder.created</code> , <code>reminder.sent</code> , <code>document.sent</code>	Other documents
<code>customer.created</code> , <code>product.created</code>	Catalogue
<code>payment_link.paid</code>	A pay-by-link payment was confirmed by the gateway (<code>data.invoiceId</code> , <code>data.paymentId</code> , <code>data.provider</code>)
<code>bank.transaction_matched</code>	A bank transaction was matched to an invoice or supplier invoice and a payment recorded
<code>connector.synced</code>	An accounting connector finished a sync (<code>data.provider</code> , counts, errors)

GET `/webhooks/events` returns the same list with descriptions.

Delivery format

```
POST /hooks/einvoicing HTTP/1.1
Content-Type: application/json
User-Agent: Webhook-Delivery/1.0
X-Webhook-Event: invoice.paid
X-Webhook-Delivery: 7c0d...           (unique per delivery attempt series)
X-Webhook-Event-Id: 9b1e...          (unique per event; the same for every subscriber)
X-Webhook-Attempt: 1
X-Webhook-Signature: t=1700000000,v1=5f1a...

{
  "id": "9b1e...",
  "event": "invoice.paid",
  "createdAt": "2026-09-23T10:15:00Z",
  "organizationId": "...",
```

```

    "data": {
      "id": "3f2e...", "invoiceNumber": "INV-0042", "status": "accepted", "paymentStatus":
"paid",
      "total": 6100, "paidAmount": 6100, "currency": "AED",
      "buyer": { "id": "...", "name": "Buyer LLC", "trn": "100000000000011" }, "isTest": false
    }
  }
}

```

Respond with any `2xx` within 10 seconds. Redirects are not followed.

Verify the signature

`v1` is the hex HMAC-SHA256 of "`<t>.<raw body>`" with your secret. Reject timestamps older than five minutes to prevent replays.

```

import { createHmac, timingSafeEqual } from "node:crypto";

export function verify(secret: string, header: string, rawBody: string): boolean {
  const parts = Object.fromEntries(header.split(",").map((p) => p.split("=")));
  if (Math.abs(Date.now() / 1000 - Number(parts.t)) > 300) return false;
  const expected = createHmac("sha256",
secret).update(`${parts.t}.${rawBody}`).digest("hex");
  return timingSafeEqual(Buffer.from(expected), Buffer.from(parts.v1 ?? ""));
}

```

```

import hmac, hashlib, time

def verify(secret: str, header: str, raw_body: bytes) -> bool:
  parts = dict(p.split("=", 1) for p in header.split(","))
  if abs(time.time() - int(parts["t"])) > 300:
    return False
  expected = hmac.new(secret.encode(), f"{parts['t']}".encode() + raw_body,
hashlib.sha256).hexdigest()
  return hmac.compare_digest(expected, parts.get("v1", ""))

```

Use the raw request body, not a re-serialised JSON object.

Retries and monitoring

Failed deliveries (non-2xx, timeout, connection error) are retried after at least 1 min, 5 min, 30 min, 2 h and 12 h (a job sends due retries every 5 minutes); after six attempts the delivery is marked `failed`. A webhook that accumulates 25 exhausted deliveries in a row is paused with a `disabledReason`; fix the endpoint and set `isActive: true` to resume.

Route	Purpose
GET /webhooks/{id}/deliveries?status=failed	Delivery log with attempts, response status, error, duration
GET /webhooks/{id}/deliveries/{deliveryId}	One delivery including its payload

Route	Purpose
POST <code>/webhooks/{id}/deliveries/{deliveryId}/retry</code>	Retry now
POST <code>/webhooks/{id}/test</code>	Send a <code>webhook.test</code> event and return the result
POST <code>/webhooks/{id}/rotate-secret</code>	New signing secret

Deliveries are idempotent from your side when you key on `X-Webhook-Event-Id` : the same event may be delivered more than once after a retry.

How delivery runs

Events are written to an outbox by database triggers the moment the change is committed. A worker pushes them out immediately after API and app actions (submission, sending, imports, payments) and every minute from the scheduler, so a delivery normally arrives within seconds and never later than a minute.