

Stripe Webhooks with Checkout & Invoices

Nicholas Brandon Yang

Summary report

Capability Commerce · Stage 3

2026-07-13 · Test mode

Why webhooks matter

When a customer pays, Stripe sends them back to a success page — but that redirect is not reliable proof of payment, because the customer can close the tab, lose connection, or open the URL without paying. A **webhook** fixes this: it is an HTTP request Stripe sends from its own servers directly to a URL on my server whenever something happens, and Stripe keeps retrying until my server confirms it. In short, the redirect is what the customer sees, while the webhook is what my system can actually trust.

Checkout

For a one-time payment through Stripe Checkout or a Payment Link, the key event is **checkout.session.completed**, which fires when the customer finishes the session. When handling it, Stripe recommends three rules: verify the event's signature using the raw request body and the endpoint's signing secret; be idempotent by storing each event id so retries never fulfil an order twice; and reply with a 2xx response within 20 seconds, doing any slow work like emails afterwards.

Invoices

An invoice has its own lifecycle — it is created as a draft, becomes open when finalized, and turns to paid once payment succeeds. Rather than polling a single flag, I listen to events along that path. The one to remember is **invoice.paid**, which Stripe recommends because it fires for both online payments and invoices marked paid manually, whereas `invoice.payment_succeeded` only fires for actual payment attempts. Invoices appear automatically for subscriptions, or can be sent on their own through the Invoicing API or dashboard.

How it connects — and what's next for SpinDry

Checkout and Invoices are two ways money arrives, and webhooks are the single reliable channel that confirms it: a one-time sale uses Checkout events, while a recurring or billed sale uses Invoice events. The security rule is constant — the secret key (`sk_`) and webhook signing secret (`whsec_`) live only in server environment variables, never in a webpage. My SpinDry page currently uses a Payment Link, so it holds no keys at all. The next step (Path B) is to add a small backend with a webhook endpoint, so a real payment automatically writes a JSONL evidence line — closing the loop from capability to product to payment to evidence.