

Dispatch Pipeline

How the PO, Invoice, LoadingSheet, and Ride refactor docs combine into one pipeline — plus a worked example, shop to shelf.

COMBINES

3 refactor docs

RIDE LIFECYCLE

ride_lifecycle.md

ORDER / INVOICE

order_invoice_lifecycle.md

LOADING EVENT

loading_event.md

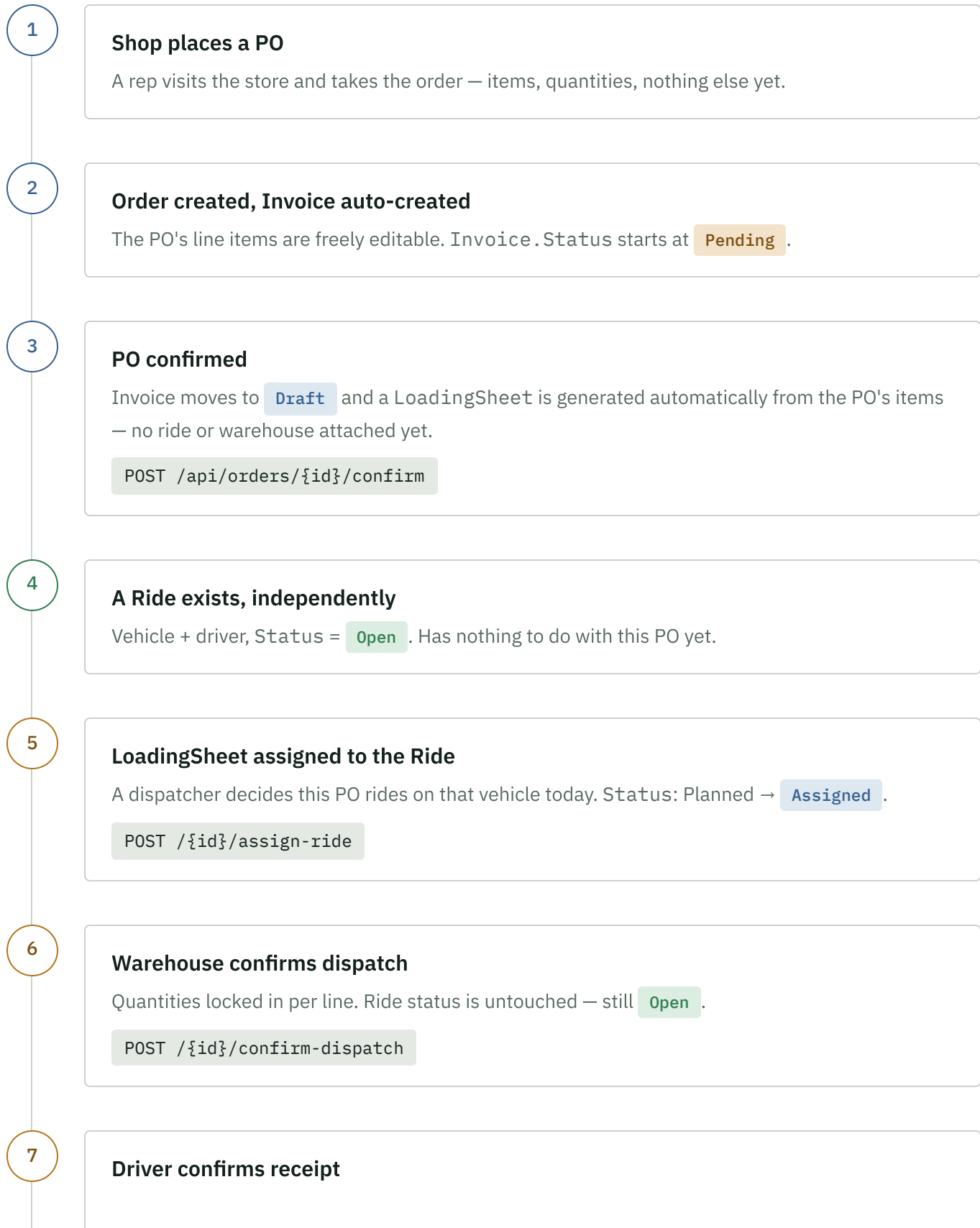
A shop's **PO** drives everything now — not a Ride, and not a Route. A PO creates its own **Invoice** and its own **LoadingSheet**; a **Ride** is just a vehicle and driver that gets attached to one or more LoadingSheets later, once someone decides which PO goes on which trip.

Four entities, three documents

ENTITY	WHAT IT IS	COMES FROM	OWNING DOC
Order "PO"	What a shop ordered	Created by a rep / sales flow	order_invoice
Invoice	The billing document for that PO	Auto-created the instant the PO exists	order_invoice
LoadingSheet	What needs to be physically loaded for that PO	Auto-created the instant the PO is confirmed	loading_event
Ride	A vehicle + driver + meter reading, nothing else	Created independently, whenever a trip is planned	ride_lifecycle

One PO, ten steps, three documents

○ order_invoice_lifecycle.md ● loading_event.md ● ride_lifecycle.md



Match → stock moves, warehouse to vehicle. Mismatch → **DiscrepancyFlagged**, a supervisor resolves it.

```
POST /{id}/confirm-receipt
```

8

Delivery happens

Shop receives goods; returns, damage, or short-delivery get noted for reconciliation.

9

Invoice finalized

Adjustments applied, total locked. This is what the shop actually received and paid — the number day-end calculations run on.

```
POST /api/invoices/{id}/finalize
```

10

Ride closed or cancelled

Independent of any one PO — a Ride can carry several POs' LoadingSheets, and closes once the whole trip is done.

```
POST /api/rides/{id}/close
```

Green Mart orders soda & chips

PO #4471 · Green Mart		Soda × 50 · Chips × 20
01	Rep takes the order. Order #4471 created — FK_StoreId = GreenMart.	
02	Invoice #INV-4471 auto-created, Pending . Order still fully editable.	
03	PO confirmed: POST /api/orders/4471/confirm Invoice → Draft . LoadingSheet #LS-9001 auto-created, items copied, Planned , no ride yet.	
04	Ride #R-77 already exists — Truck-12, driver Amir, Open . Unrelated to Green Mart so far.	
05	Dispatcher assigns the load: assign-ride { rideId: R-77, sourceWarehouseId: WH-Central } LS-9001 → Assigned . Three other shops' sheets join the same ride the same way.	
06	Warehouse pulls 50 crates soda + 20 cases chips, confirms dispatch. LS-9001 → DispatchConfirmed . Ride #R-77 stays Open .	
07	Amir counts on load-in: only 48 crates of soda fit. Received: Soda 48 / Chips 20 vs. Dispatched: Soda 50 / Chips 20. Mismatch → DiscrepancyFlagged . No stock movement yet.	
08	Supervisor confirms 48 is correct, resolves: QuantityResolved = Soda 48 / Chips 20. LS-9001 → Resolved . Stock posts: -48 Soda / -20 Chips at WH-Central, +48 Soda / +20 Chips on Truck-12.	
09	Amir delivers. Green Mart takes only 45 crates of soda — 5 damaged in transit — and pays for what they received.	

10

Office finalizes: `POST /api/invoices/INV-4471/finalize` with a 5-crate deduction.

Invoice → **Finalized** . Total locked to 45 soda + 20 chips — this is the number day-end reconciliation runs on.

11

Every sheet on Ride `#R-77` reaches its own resolution; the trip ends. `POST /api/rides/R-77/close` → **Closed** .

Distribution ERP — Refactor Docs

See [ride_lifecycle.md](#) · [order_invoice_lifecycle.md](#) · [loading_event.md](#) for full field-level design