CASE STUDY #3

Building Reliable Order Tracking When the Data Source Is Unreliable

Lights Canada , E-Commerce Platform

Project	Lights Canada , E-Commerce Platform
Area	Order Lifecycle Management & Customer Transparency
Difficulty	High , Distributed state, external dependencies, real-time UX
Outcome	Accurate, real-time order tracking with graceful degradation on data gaps

Background

One of the core customer-facing promises of the Lights Canada platform was real-time order tracking , the ability for a customer to log in and see the live status of their order from placement through delivery. For a retailer selling large, high-value items (chandeliers, fans), this was not a nice-to-have: customers spending CAD $300–$2,000 on a single item needed to know where it was.

The Problem

Order status in the real world depends on a chain of external events: payment confirmation from the gateway, fulfillment confirmation from warehouse staff, shipment data from the carrier. Each of these arrives asynchronously, via webhook or manual admin update, and none of them is guaranteed to arrive in order , or arrive at all.

Failure Scenarios Identified

- Payment webhook delayed by 30–90 seconds after the customer sees 'Order Placed' , order appears stuck.
- Warehouse staff marking an order as 'Shipped' without entering a tracking number , status advances but tracking link is broken.
- Carrier webhook (shipment status) arriving out of sequence , a 'Delivered' event arriving before 'In Transit'.
- Network failure causes a webhook to be dropped entirely , order status never advances past a given state.

Customer Impact

A customer who orders a CAD $800 chandelier and sees 'Order Placed' for 48 hours with no update will contact support, question the charge, and potentially initiate a chargeback. Order status ambiguity directly translates to customer service cost and trust erosion.

The Solution

1. Event-Sourced Order State Machine

Order status was modeled as an event log, not a mutable status field. Every state transition , payment received, order confirmed, picking started, shipped, delivered , was stored as an immutable event record with a timestamp and source (gateway webhook, admin action, carrier webhook). The customer-facing status was derived by replaying the event log, always reflecting the latest known state.

This meant out-of-order events were handled gracefully: a 'Delivered' event arriving before 'In Transit' would be stored chronologically by event timestamp, not arrival time, ensuring the displayed sequence was always logically correct.

2. Explicit State Validity Rules

A state machine definition was implemented that codified which transitions were legal. An order could not move from 'Placed' to 'Delivered' without passing through 'Confirmed' and 'Shipped'. If a webhook tried to trigger an illegal transition, it was queued as a pending event and resolved once the preceding states arrived , or flagged for admin review after a timeout.

3. Graceful Degradation Messaging

The customer-facing tracking UI was designed to communicate uncertainty honestly rather than display stale or incorrect data. If the system knew an order was confirmed but had not yet received a shipment event, it displayed: 'Your order is confirmed and being prepared , tracking will appear once dispatched.' This reduced inbound support queries because customers had accurate, contextual information rather than a frozen status.

4. Admin Reconciliation Tools

The admin panel included an order health view that surfaced orders stuck in a state for longer than expected. Admins could manually advance an order, add a tracking number retroactively, or trigger a re-fetch of the carrier status. This gave the operations team a safety net for the cases that automated logic could not resolve.

Scenario	Handling

Delayed payment webhook	Status shows 'Processing payment' with estimated resolution time
Out-of-order carrier events	Events stored by carrier timestamp; display derived from sorted log
Dropped webhook	Admin reconciliation dashboard flags stalled orders after timeout
Shipped without tracking number	Status advances; tracking section shows 'Tracking number pending'
Illegal state transition	Event queued; admin alerted if not resolved within SLA window

Outcome

Post-launch, order-related customer support tickets dropped significantly compared to the client's previous manual process. Customers reported confidence in the platform because the tracking page communicated what was known , and was honest about what wasn't. The admin reconciliation tool caught and resolved an average of 3–5 stuck orders per week in the first month, all without customer complaints.

Key Takeaway

Order tracking is not a display problem , it is a distributed systems problem. When your data arrives from multiple external sources in unpredictable order, the only reliable model is event sourcing with explicit state machine rules and graceful degradation. Honest, contextual messaging beats false precision every time.

, Ehsan