PM Product Assignment

Pratyush Ranjan | Senior Product Manager

Product: Real-Time Order Tracking via Webhook Infrastructure

Company: Blowhorn (eCommerce Fulfilment & Logistics)

1. Who is the Customer / Partner?

Blowhorn is a B2B last-mile logistics platform serving eCommerce brands and enterprise retail clients across 70+ Indian cities. The OMS served a diverse customer base, and this product initiative was deliberately designed as a tiered solution. It is not a one-size-fits-all.

Customer segments and how each consumed the tracking capability:

- **Tier 1 : Small & Mid-size Merchants (SMB tier, ~60% of partner count).**
  - These include D2C sellers, regional retailers, and ONDC-listed brands processing under 200 daily shipments.
  - These partners lacked dedicated tech teams and were best served by Blowhorn's built-in OMS tracking UI, a no-integration-required dashboard that displayed live order statuses and auto-sent templated SMS notifications via Blowhorn's own communication layer. They got real-time tracking without writing a single line of code.
- **Tier 2 : Mid-to-Large Merchant Partners (growth tier, ~30% of partner count).**
  - These include Shopify (and other similar tools) and ONDC sellers with little to medium sized in-house tech capability, processing 200–800 daily shipments.
  - These partners wanted to own their customer communication stack but didn't need complex CRM integrations. They used our webhook feeds to trigger their own SMS/email workflows, with moderate customisation needs.
- **Tier 3 : Enterprise Clients (strategic tier, ~10% of partner count, ~50% of GMV).**
  - These include large retail and consumer brands such as Decathlon, Croma, and Apollo, processing 1,000–3,000+ daily shipments each.
  - These clients had dedicated integration teams, existing CRM and marketing automation stacks, and required real-time, richly-structured webhook events to power delivery communication, marketing triggers, and operational dashboards.

2. The Challenge: What Problem Were We Solving?

Prior to this initiative, Blowhorn's OMS relied on a fragmented, semi-manual tracking architecture. Merchants had to poll Blowhorn's internal APIs at intervals to check order status, a costly, unreliable approach that introduced latency of 30–90 minutes between a status change occurring and the merchant being notified.

This created three compounding pain points:

- **High WISMO (Where Is My Order) call volumes:** Support teams at merchant partners were fielding 18–25% of all inbound customer contacts as WISMO calls, with little to 0 self-service resolution because status data was stale.
- **Delivery failures from delayed rider + OTP communication**: When an order moved to 'Out for Delivery', the rider's details and OTP were not pushed to the end-customer in real time. This contributed to a first-attempt delivery failure rate of approximately 14–17%, with re-delivery costs estimated at ₹45–65 per failed attempt.
- **Missed marketing and CX triggers:** Enterprise clients had no reliable mechanism to fire post-purchase SMS/email campaigns tied to delivery milestones, meaning upsell, review-request, and loyalty touchpoints were either delayed or absent entirely.

At 10,000+ daily orders, even a 15% failed-delivery rate meant 1,500 re-deliveries/day, causing a direct margin erosion of ₹67,500–97,500 daily across the network.

3. Value Proposition of the Solution

We designed and shipped a two-layer Order Tracking Architecture within Blowhorn's OMS. It was a tiered solution that matched product complexity to partner capability:

- **Layer 1 : Built-in OMS Tracking (for SMB partners):** A zero-integration tracking UI within the Blowhorn merchant portal, with automated templated SMS sent via Blowhorn's own SMS-backed communication service. Partners simply toggled it on. No API keys, no endpoints, no engineering overhead.
- **Layer 2 : Webhook Notification Infrastructure (for growth + enterprise partners):** A configurable, event-driven system that pushed real-time order-status updates to registered partner endpoints the moment a status transition occurred. Thereby enabling partners to build fully custom downstream workflows.

This tiered design was a deliberate product strategy: it ensured 100% of the partner base got access to real-time tracking (not just technically capable ones), while giving enterprise clients the programmatic control they needed. The core value proposition of Layer 2, specifically:

- **Eliminate polling latency:** Replace reactive polling with proactive push, reducing status propagation time from ~60 minutes to under 90 seconds.
- **Enable downstream automation at zero marginal cost for partners:** Partners could now build their own SMS, email, and CX workflows on top of a reliable event stream, without custom engineering on our side per client.
- **Unlock data-driven delivery operations:** Real-time events enabled first-attempt delivery optimization by pushing rider details + OTPs to end-customers at the exact right moment, not hours later.

The business case was clear: reducing first-attempt delivery failures by even 5 percentage points across the Decathlon + Croma enterprise tier alone would save the network ₹2.5–3.2L per month in re-delivery costs, while increasing partner NPS and reducing churn risk.

4. Top Scenarios Enabled by the Solution

Scenario A: Automated End-Customer Delivery Communication

Merchant partners integrated our webhook feed to automatically trigger SMS and email notifications to end-customers at four key moments : order confirmed, out for delivery (with rider name + contact + OTP), delivered, and failed delivery. This replaced a manual ops step that was previously handled by Blowhorn's internal support team via batch jobs.

- **Feature 1 : Webhook Event Schema**: Defined a standardised JSON payload per event type (order_confirmed, out_for_delivery, delivered, delivery_failed, return_initiated) with consistent field names, timestamps, and rider metadata, enabling partners to integrate once and consume all events.
- **Feature 2 : Retry + Delivery Guarantee Logic:** Built exponential backoff retry (3 attempts over 10 minutes) with a dead-letter queue for failed deliveries, ensuring >99.5% event delivery reliability even during partner-side downtime.
- **Feature 3 : OTP & Rider Detail Injection:** For the out_for_delivery event specifically, the webhook payload was enriched with rider name, masked mobile number, and delivery OTP, all these pulled from the dispatch system in real time at trigger time.

Scenario B: Marketing & CRM Trigger Automation

Enterprise clients wired our webhook events into their CRM and marketing automation tools to fire hyper-contextual campaigns. A delivered event would trigger a review-request SMS within 2 hours; a delivery_failed event would trigger a proactive apology + rescheduling flow.

- **Feature 1 : Custom Webhook Configurations per Client:** Partners could register multiple endpoints and map specific event types to each endpoint, e.g., send out_for_delivery only to their logistics ops dashboard, but all events to their CRM.
- **Feature 2 : Webhook Dashboard & Logs:** Built a self-serve portal where partner ops teams could view event history, retry failed webhooks manually, and test endpoint connectivity, thereby, reducing dependency on Blowhorn's integration support team.

Scenario C: Return & Exception Handling Automation

Return events (return_initiated, return_picked_up) were previously communicated via manual merchant-support tickets, often with 4–6 hour lag. Webhook-driven return events allowed merchants to auto-update their OMS inventory and trigger refund flows without human intervention.

- **Feature 1 : Return Event Types**: Extended the event schema to include return_initiated and return_picked_up, with reason codes and condition flags, enabling downstream warehouse and refund automation.
- **Feature 2 : Merchant Acknowledgement API:** Partners sent back an HTTP 200 ACK upon receiving each webhook; our system tracked acknowledgement rates per partner and flagged at-risk integrations automatically.

5. Top 3 User Stories

User Story 1: Automated Delivery Communication for End-Customers

As a merchant partner managing 500–2,000 daily shipments, *I want to receive a real-time push notification the moment an order moves to 'Out for Delivery'*, complete with rider details and delivery OTP, *so that I can immediately relay this to my end-customer via SMS, without any manual intervention from my ops team.*

Why this matters: Before this capability, the OTP and rider details sat inside Blowhorn's internal dispatch system and were communicated to end-customers only when the rider called, often minutes before arrival or not at all. This meant end-customers frequently missed deliveries because they had no prior notice, no OTP ready, and no way to contact the rider proactively. Each failed delivery cost the merchant ₹45–65 in re-delivery charges plus a degraded customer experience.

Acceptance criteria / supporting context:

- The out_for_delivery webhook fires within 60 seconds of the dispatch system updating the order status.
- The payload includes: rider_name, rider_phone_masked, delivery_otp, estimated_delivery_window, and order_id.
- The event is retried up to 3 times with exponential backoff if the partner endpoint returns a non-200 response.
- Merchant is able to configure which endpoint receives this specific event type independently of other event types.

User Story 2: Self-Serve Webhook Integration & Observability

As a technical integration lead at a merchant or enterprise partner, *I want to register, test, and monitor my webhook endpoints from a self-serve dashboard so that I can go live with real-time order tracking in under 2 hours without raising a support ticket to Blowhorn's integration team.*

Why this matters: Before this, every new webhook integration required a Blowhorn solutions engineer to manually configure the endpoint, test the connection, and validate the payload, a process that took 5–10 business days per partner due to backlog and coordination overhead. This was a direct blocker to onboarding velocity and a source of frequent partner escalations. With 70+ active cities and 200+ logistics partners, this approach was not scalable.

Acceptance criteria / supporting context:

- Partners can register up to 5 endpoint URLs per account via the dashboard, mapping each to specific event types.
- A 'Send Test Event' function allows partners to fire a synthetic webhook payload to any registered endpoint and view the HTTP response in real time, enabling validation without live orders.
- The Event Log surface shows the last 500 events per partner, filterable by event type, status (delivered/failed/retried), and timestamp.
- Failed events can be manually retried from the dashboard, with a visible retry history per event.
- Post-launch: median partner integration time dropped from 7 business days to 1.5 business days (79% reduction). Support tickets related to webhook setup fell by 68% within 6 weeks of launch.

User Story 3: CRM & Marketing Trigger Automation on Delivery Events

As a CX or marketing manager at an enterprise client, *I want to receive a structured webhook event the moment an order is marked as 'Delivered' or 'Delivery Failed', with a consistent payload schema and delivery guarantee, so that I can automatically trigger post-purchase CRM workflows within minutes of the delivery outcome, not hours later.*

Why this matters: Enterprise clients had sophisticated CRM stacks but no reliable, real-time signal from their logistics layer. The result: review-request campaigns fired 6–12 hours after delivery (when intent had cooled), and failed-delivery recovery flows were triggered the next business day, by which point the customer had often already initiated a chargeback or social complaint. Real-time webhook events made these workflows operationally viable for the first time.

Acceptance criteria / supporting context:

- The delivered and delivery_failed events fire within 90 seconds of the rider updating the order status in the dispatch app.
- The payload includes: order_id, delivery_timestamp, failure_reason (if applicable), delivery_address_confirmed, and customer_id.
- Partners can configure separate endpoint URLs for delivered vs. delivery_failed events, enabling independent CRM workflow routing.
- Payload schema is versioned (v1, v2) with backward compatibility guaranteed for 12 months, giving enterprise clients confidence to build on top of the API without fear of breaking changes.

6. Measurable Business Impact

The following metrics were tracked over a 12-week post-launch period across the pilot cohort of 15 enterprise clients and 4 merchant partners:

Metric	Before	After
First-attempt delivery success rate	~83%	~91% (+9.6%)
WISMO contacts as % of total support volume	~22%	~9% (-59%)
Partner integration lead time (new webhook)	7 business days	1.5 days (-79%)
Webhook event delivery reliability	N/A (polling)	>99.5%
Status propagation latency	30–90 min	<90 seconds
Post-delivery review submission rate (pilot client)	Baseline	+22%
Failed-delivery support contacts (pilot client)	Baseline	-31%
Estimated daily re-delivery cost savings (network)	—	₹60,000–80,000/day

7. Technical Overview

The solution was built as an event-driven microservice layer on top of Blowhorn's existing OMS:

- **Event Source:** Order status transitions in the OMS (MySQL-backed) were captured via a CDC (Change Data Capture) pattern. This decoupled the webhook system from the core OMS write path, ensuring zero latency impact on order processing.
- **Webhook Dispatcher Service:** A Node.js microservice consumed from the topic, enriched payloads with rider and OTP data from the dispatch service (via internal gRPC call), and dispatched to registered partner endpoints over HTTPS.
- **Reliability Layer:** Implemented exponential backoff retry (3 attempts: 30s, 2min, 8min). Events undelivered after 3 retries were surfaced in the partner dashboard for manual retry.
- **Partner Configuration Store:** Webhook endpoint registrations and event-type mappings stored in PostgreSQL with a thin REST API for the self-serve dashboard (React frontend).
- **Observability:** Each dispatch attempt logged to Elasticsearch with event_id, partner_id, event_type, http_status, latency_ms, and retry_count, powering the Event Log UI and internal SLA monitoring dashboards in Kibana.
- **Comprehensive Unified JSON Schema:** A deliberate architectural decision was made to design a single, comprehensive webhook payload that contained all possible fields across all event types, rider details, OTP, delivery timestamps, failure reason codes, return condition flags, customer ID, and address confirmation, even if not all fields were populated for every event. This meant partners integrated against one schema once, and never needed to re-integrate as new event types or fields were introduced. Fields irrelevant to a specific event were returned as null rather than omitted, ensuring the contract never changed shape. This was the key design choice that enabled the self-serve integration model and the 1.5-day median onboarding time.
- **Infrastructure:** Deployed on AWS (ECS Fargate), with auto-scaling configured to handle 3x normal volume, a critical requirement given Blowhorn's 2x volume spikes during sale events. System maintained 99.9% uptime throughout.

Note: Specific client names used with permission for context. All metrics are directionally accurate and based on pilot-phase measurement. Detailed observability dashboards available upon request.