



Rapido Integration: Edge Case Analysis

This document outlines potential edge cases, race conditions, and failure scenarios for the Rapido Fleet Integration.

✓ Handled Scenarios (Logic Exists)

1. The "Busy Conflict" (Queue Logic)

- **Scenario:** Driver is on an active Rapido Trip (Status: `STARTED`) but an internal scheduler assigns them a trip starting in 30 mins.
- **Risk:** Driver receives internal duty but remains ONLINE on Rapido, potentially getting another booking.
- **Handling:** `validateAndSyncRapidoStatus` detects the "Active Rapido Trip". It calculates that the driver *should* be OFFLINE. Instead of calling the API (which would fail), it saves `pendingStatus: 'offline'` to `driver.providerMetadata`.
- **Resolution:** When Rapido webhook sends `COMPLETED`, the system processes the queue and marks the driver OFFLINE immediately.

2. The "Ghost" Ride (Missing Start Webhook)

- **Scenario:** We miss the `started` webhook (network glitch), but receive the `completed` webhook later.
- **Risk:** The ride never exists in our DB, so we can't track it or link it to the driver.
- **Handling:** `handleOrderStatusUpdate` checks for the ride by `providerBookingId`. If not found, it **creates a new ride** record on the fly, ensuring we capture the fare and distance even if we missed the start.

3. Missing Vehicle Number

- **Scenario:** We try to mark a driver ONLINE, but they have no vehicle assigned in our system (or assignment ended).
- **Risk:** Rapido API fails (Vehicle Number is mandatory for ONLINE).
- **Handling:** Use `Assignment` relations. If no active assignment is found, the system throws an error or defaults to OFFLINE, preventing invalid API calls.

4. Duplicate Webhooks

- **Scenario:** Rapido sends the same `dropped` webhook twice.
- **Risk:** Double counting fare or distance.
- **Handling:** Our logic updates the *existing* ride by `providerBookingId`. It is idempotent (setting status to `COMPLETED` twice does no harm).

5. Multi-Provider Conflict (Fixed)

- **Scenario:** Driver is assigned a trip on another provider (e.g., MMT).
- **Previous Risk:** `RapidoService` only checked `INTERNAL` and `RAPIDO` trips.
- **Handling:** `validateAndSyncRapidoStatus` now checks for **ANY** active trip where `provider != 'RAPIDO'`. If found, it forces the driver OFFLINE on Rapido.

6. The "Silent Fail" (Network Outage) (Fixed)

- **Scenario:** System tries to mark driver OFFLINE, but API fails.
- **Previous Risk:** Driver remains ONLINE incorrectly.

- **Handling:** Implemented a **Retry Queue**. Failed API calls are saved to `providerMetadata` with a `retryCount`. A background worker retries these every 5 minutes (up to 5 times).

7. The "Manual Override" Race (Fixed)

- **Scenario:** Driver manually goes ONLINE in the Rapido App, bypassing internal blocks.
- **Previous Risk:** Drivers could game the system.
- **Handling:** The `captain_status` webhook listener now detects `status: online` events. If received, it immediately triggers an internal eligibility check. If the driver *should* be busy, the system forces them back OFFLINE instantly.

Potential Risks & Unhandled Edge Cases

1. The "Infinite Loop" (Webhook Echo)

- **Scenario:** We call `changeCaptainStatusRequest`. Rapido changes status -> Sends us a `captain_status` webhook -> We process it -> Logic triggers another sync -> We call API again.
- **Impact:** Loops.
- **Mitigation:** We disabled the auto-update in `handleCaptainStatusCallback` (commented out code) to prevent this. We trust our internal state as the source of truth.

2. Late Webhook Arrival

- **Scenario:** Driver finishes Rapido trip at 10:00. We receive the webhook at 10:15 due to internet lag.
- **Impact:** Driver remains "Busy" in our system for 15 extra minutes. If they had an internal trip at 10:10, they might be marked late or the system might have blocked them.
- **Mitigation:** Use the `eventTime` timestamp impact from the payload (if available) to backdate the `completedAt`, though real-time operations are still delayed.

Recommended Test Scenarios

| Scenario | Trigger | Expected Outcome |
|---------------------------------|--|--|
| Break during Rapido Trip | Driver starts break while on Rapido Trip | Status Queue created (<code>pendingStatus:</code> <code>'offline'</code>). |
| Connection Timeout | API Fail during Sync | Error Logged (System should ideally retry). |
| Zero Distance Trip | Webhook with 0 distance | Ride created with 0 km (Monitor for fraud). |
| Driver Not Found | Webhook for unknown phone number | 404 Error Logged (Safe ignore). |