Courier Partner Integrations

Supported Courier Partners

The system integrates with multiple courier partners in Indonesia:

1. JNE (Jalur Nugraha Ekakurir)

Features:

- Status webhook integration
- Status code mapping to internal codes
- Real-time tracking updates

Controller: `app/controllers/3p/jne.controller.js`

2. Lion Parcel

Features:

- Scheduled pickup requests via cron job
- Bearer token authentication
- Automatic pickup scheduling

Configuration:

```
// Environment variable
LIONPARCEL_PICKUP_CRON="*/30 * * * *"  // Every 30 minutes
LIONPARCEL_BEARER_TOKEN="<token>"
```

Controller: `app/controllers/3p/lion-parcel.controller.js`

3. JT Cargo (J&T Cargo)

Features:

- Full API integration
- Special pricing support
- Database-driven pricing lookup
- Fallback to JSON mapping

Special Pricing:

- Primary: `jnt_cargo_special_prices` table
- Fallback: `app/maping/jntcargo-special-pricing.json`

Controller: `app/controllers/3p/jt-cargo.controller.js`

4. AnterAja

Features:

- Webhook integration
- Access key authentication
- Status synchronization

Authentication:

- Access Key ID: `ANTERAJA_ACCESS_KEY_ID`
- Secret Access Key: `ANTERAJA_SECRET_ACCESS_KEY`

Controller: `app/controllers/3p/anteraja.controller.js`

5. Sicepat

Features:

- Status webhook support
- Tracking integration

Controller: `app/controllers/3p/sicepat.controller.js`

6. Sap Express (SAPX)

Features:

- API integration
- Shipment creation

Controller: `app/controllers/3p/sapx.controller.js`

7. POS Indonesia

Features:

- Indonesian postal service integration
- Manifest generation
- Token-based authentication

Routes: `app/routes/pos_indo.routes.js`

8. PCP Express

Features:

- PCP carrier network
- Manifest management

Routes: `app/routes/pcp_express.routes.js`

9. Laris Cargo

Features:

- Regional courier integration

Routes: `app/routes/laris_cargo.routes.js`

Integration Pattern

All courier integrations follow a similar pattern:

```
// 1. Receive webhook from partner
POST /api/3p/{partner}/webhook

// 2. Validate authentication (Basic Auth or API Key)
passport.authenticate('{partner}Strategy')

// 3. Parse partner status code
parsePartnerStatus(partnerStatusCode)

// 4. Map to internal status code
const internalStatus = statusMapping[partnerStatusCode]

// 5. Update internal tracking
await updateScan(awb, internalStatus)

// 6. Trigger internal webhooks (if configured)
await triggerWebhook(waybillData)

// 7. Return acknowledgment
```

```
return { success: true }
```

Status Mapping

Each courier has status mappings in `app/status/` directory:

- Maps partner-specific status codes to internal OnDelivery codes
- Ensures consistent tracking across all partners
- Stored as JSON configuration files

Example Status Flow:

```
Partner: "DELIVERED"
  → Internal: "DLV"
  → Display: "Delivered to Customer"
```

Resi 3P Mapping

Purpose: Link internal AWB numbers to partner AWB numbers

Table: `resi_3p`

Fields:

- `internal_awb` - OnDelivery waybill number
- `partner_awb` - Partner's tracking number
- `partner_id` - Courier partner ID
- `status` - Current status
- `created_at` - Mapping creation time

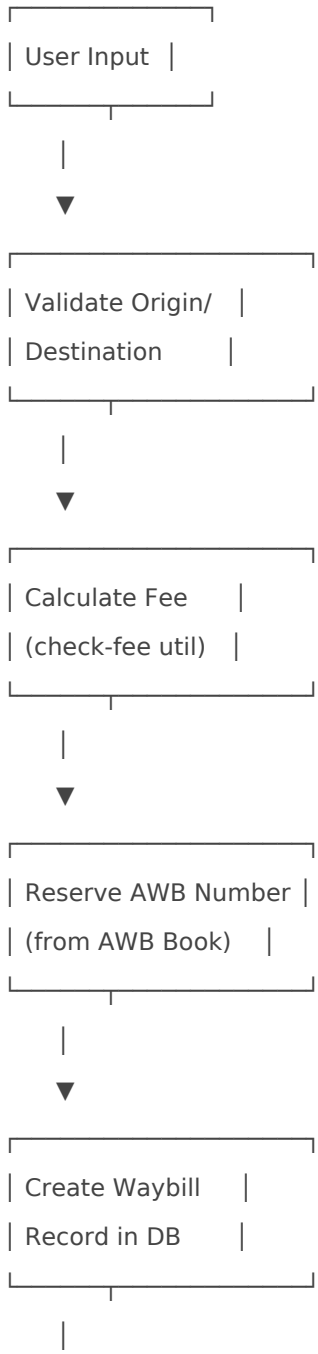
Usage:

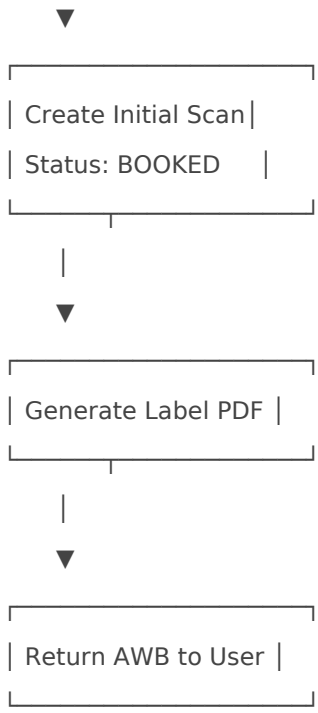
```
// When creating shipment with partner
const mapping = await Resi3P.create({
  internal_awb: "ONX1234567890",
  partner_awb: "JNE0987654321",
  partner_id: 1  // JNE
});

// When receiving webhook
const waybill = await findWaybillByPartnerAwb(
```

```
  webhookData.awb,
  partnerConfig.id
);
```

Business Workflows

Workflow 1: Waybill Creation & Booking





Workflow 2: Package Tracking Lifecycle

Outgoing Phase:

```
BOOKED → PACKED → HANDOVER → DEPARTURE
```

Transit Phase:

```
DEPARTURE → [TRANSIT] → ARRIVAL
```

Delivery Phase:

```
ARRIVAL → RECEIVING → OUT_FOR_DELIVERY → DELIVERED
                                       → PROBLEM
```

Exception Handling:

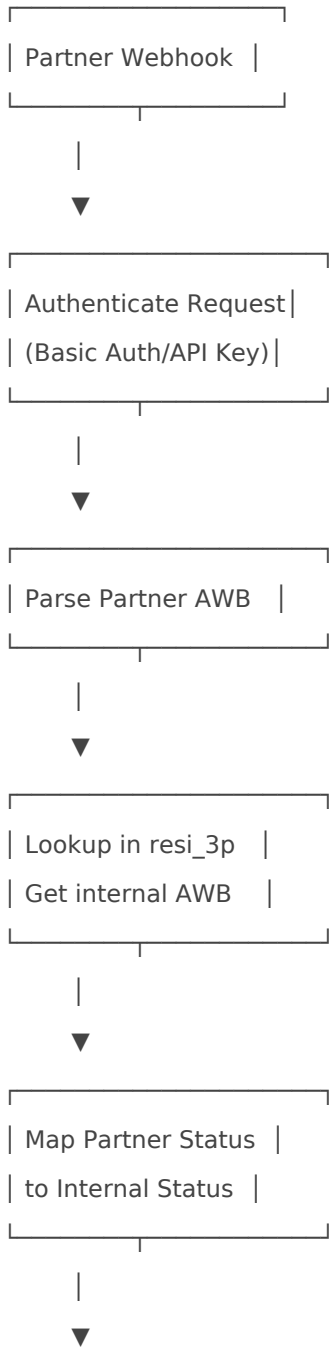
```
ANY_STATUS → PROBLEM → [PROBLEM_RESOLVED]
           → RETURN → [RETURN_COMPLETE]
           → CANCEL
```

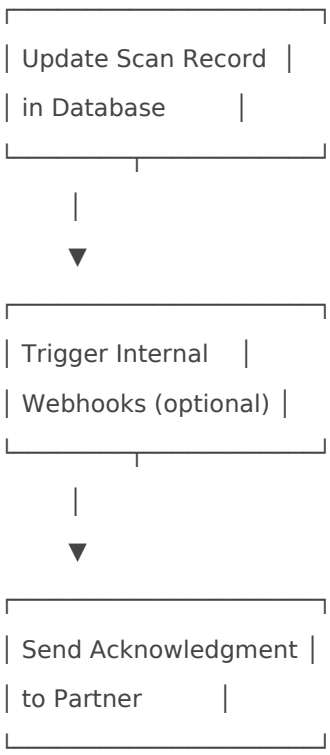
Status Codes:

- `BKD` - Booked
- `PKD` - Packed

- `HND` - Handover
- `DPT` - Departure
- `ARV` - Arrival
- `RCV` - Receiving
- `OFD` - Out for Delivery
- `DLV` - Delivered
- `PRB` - Problem
- `RTN` - Return
- `CNL` - Cancel

Workflow 3: 3rd-Party Integration





Workflow 4: Fee Calculation

```
┌──────────────────────┐
│ Get Base Price       │
│ (by weight/distance) │
└──────────┬───────────┘
           │
           ▼
┌──────────────────────┐
│ Apply Special        │
│ Pricing (if VIP)     │
└──────────┬───────────┘
           │
           ▼
┌──────────────────────┐
│ Add Surcharges       │
│ (remote area, etc.)  │
└──────────┬───────────┘
           │
           ▼
┌──────────────────────┐
│ Calculate Insurance  │
│ (if requested)       │
└──────────┬───────────┘
           │
           ▼
┌──────────────────────┐
│ Return Total Fee     │
│ with Breakdown       │
└──────────────────────┘
```

Workflow 5: Scanning Operations

Packing Scan:

1. Scan AWB number
2. Verify waybill exists
3. Check current status (must be BOOKED)
4. Record packing details (weight, dimensions)
5. Update status to PACKED
6. Generate packing list

Handover Scan:

1. Create handover manifest
2. Scan multiple AWBs
3. Verify all AWBs are PACKED
4. Record handover details (from/to location)
5. Update all AWBs to HANDOVER status
6. Generate handover manifest PDF
7. Print manifest for courier

Receiving Scan:

1. Scan incoming AWBs
2. Match with expected manifest
3. Record arrival time
4. Update status to RECEIVING
5. Flag any discrepancies
6. Assign to delivery driver

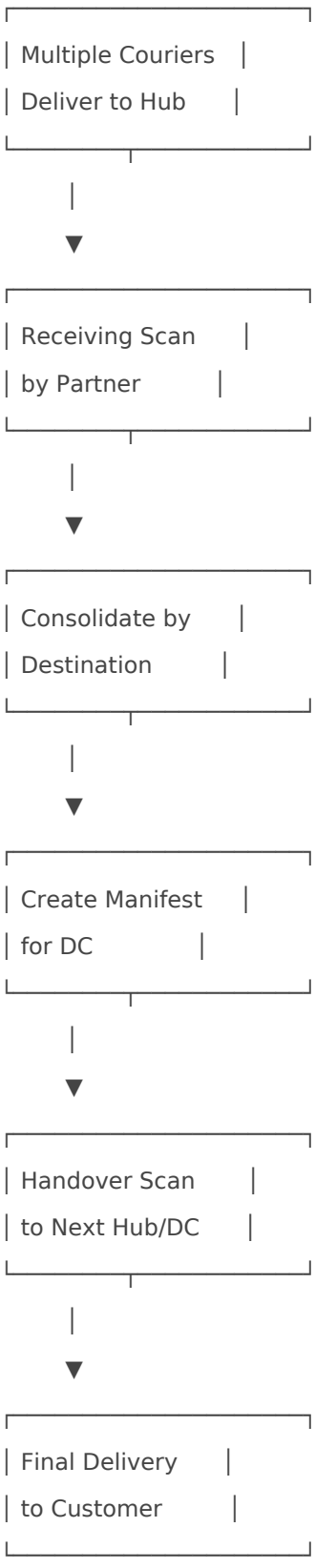
Delivery Scan:

1. Driver scans AWB
2. Capture POD (photo/signature)
3. Record delivery details
4. Update status to DELIVERED
5. Send customer notification
6. Update commission records

Problem Scan:

1. Scan AWB with issue
2. Select problem type (damaged, lost, refused, etc.)
3. Capture photos
4. Add notes
5. Update status to PROBLEM
6. Trigger problem resolution workflow
7. Notify relevant parties

Workflow 6: Multi-Courier Consolidation



Workflow 7: Quality Management

Weight Discrepancy Handling:

1. Package weighed at receiving
2. Compare with declared weight
3. If difference > threshold:
   - Capture photo evidence
   - Record actual weight
   - Create alter_weight record
   - Calculate price adjustment
   - Notify customer
4. Update billing if needed

Proof of Delivery (POD):

1. Driver completes delivery
2. Capture recipient signature (digital)
3. Take delivery photo
4. Record GPS coordinates
5. Upload to server
6. Link to waybill
7. Status → DELIVERED

Proof of Pickup (POP):

1. Driver arrives at pickup location
2. Take package photos
3. Capture sender signature
4. Record GPS coordinates
5. Upload to server
6. Link to waybill
7. Status → BOOKED

Revision #1
Created 24 February 2026 06:52:49 by ondeliveloper
Updated 24 February 2026 06:55:28 by ondeliveloper