SHIPSETH

Ship API Developer Guide

A plain-English walkthrough of the ShipSeth external booking API — from your first token to a delivered parcel. Warehouse setup, pricing, booking, tracking, webhooks, labels, manifests, invoices and NDR, explained as one flow with real requests and responses.

B2C Forward & Reverse | Prepaid · COD · To-Pay | Async Booking | Signed Webhooks | Bulk Labels · Manifest · Invoice | NDR Actions

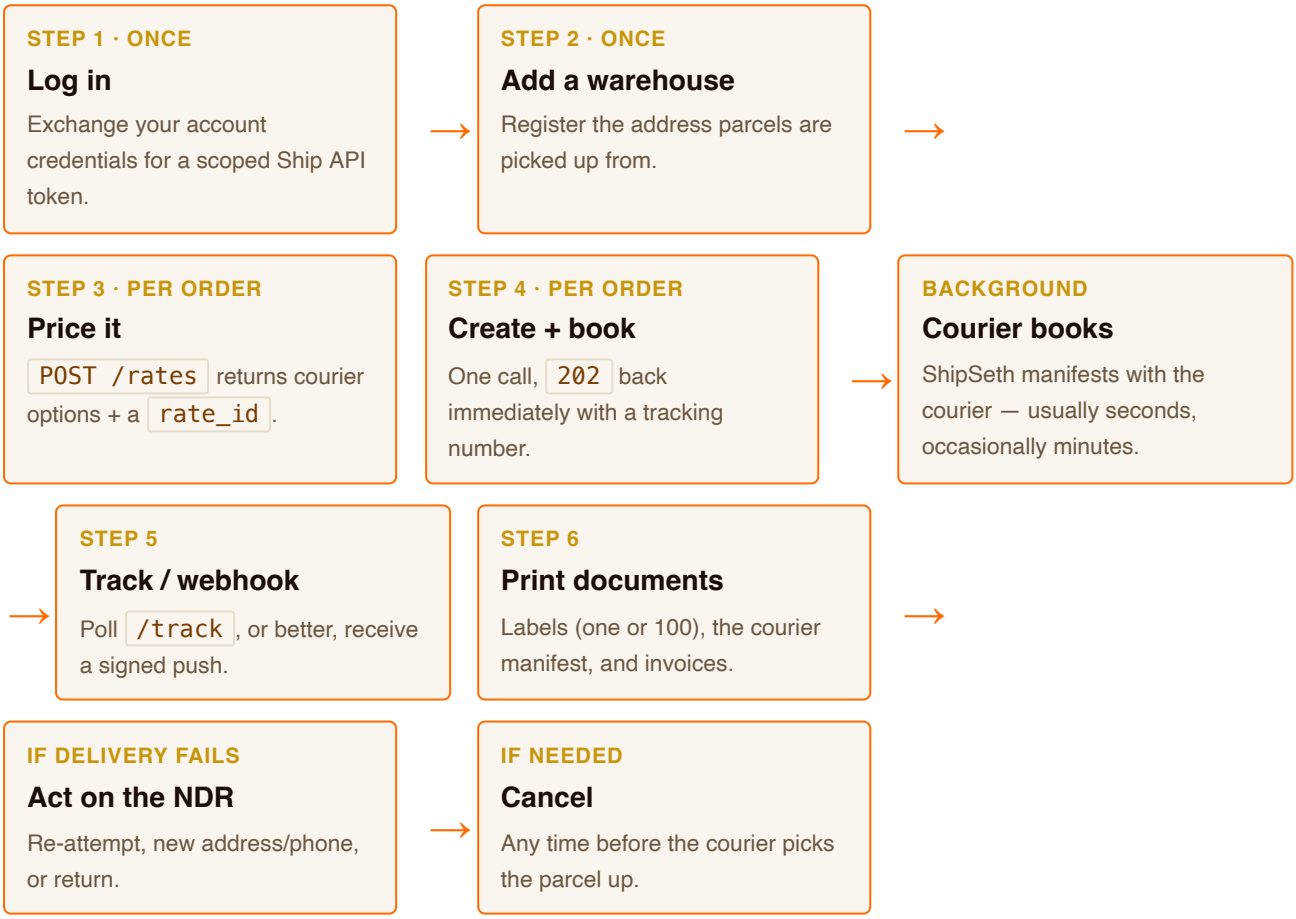
api.shipseth.com/api/ext/v1 Version 2026-09-08 · updated 29 Oct 2026 shipseth.com/developers

How the API fits together

Every integration follows the same shape: authenticate once, tell ShipSeth where parcels are picked up, price a route, create the shipment, and then either poll or listen for status. Everything below happens over plain HTTPS JSON — no SDK required.

> **Base URL:** `https://api.shipseth.com/api/ext/v1` (local dev: `http://localhost:8000/api/ext/v1`). Every request needs `Authorization: Bearer <token>` except `/auth/login` and the public docs endpoints.

The end-to-end flow



Setup vs. per-order calls

Do once per account	Do for every shipment
• Log in → save the token • Create your warehouse(s) • Set a default warehouse + webhook URL via `PUT /config`	• `POST /rates` — price it • `POST /shipments` — create + book • Webhook → label → manifest → NDR if needed

Step 1 — Authenticate

A Ship API token is a long-lived Bearer token scoped to your account. You can hold several at once (one per server/environment) and rotate them without downtime.

1 **POST** `/auth/login`

```
curl -X POST "$BASE/auth/login" \
  -H "Content-Type: application/json" \
  -d '{
    "email_or_username": "you@example.com",
    "password": "••••••",
    "label": "warehouse-server-1"
  }'
```


```
{
  "success": true, "status": 200,
  "message": "Ship API token issued.",
  "data": {
    "token": "42|AbCdEf...",
    "token_type": "Bearer",
    "abilities": ["ship:booking", "ship:tracking"],
    "environment": "live"
  }
}
```


Save `data.token` — it is shown once. Send it back as `Authorization: Bearer <token>` on every call from here on.

ALSO USEFUL

CALL	WHAT IT DOES
POST `/auth/rotate`	Mint a new token; the old one keeps working for a 72-hour grace window so you can redeploy without a gap.
POST `/auth/revoke`	Kill the current token immediately.
GET `/auth/tokens`	List every active token on the account (never the secret itself).

> **Test mode:** add `"test": true` to `/auth/login` to get a token that books through a simulator (no real courier call, no wallet charge) — perfect for staging.

Step 2 — Set up a pickup warehouse

Every shipment needs an origin. Register it once; after that you can either pass `warehouse_id` on each shipment or set a default so you never have to.

1 **POST** `/warehouses` — create one

```
curl -X POST "$BASE/warehouses" \
  -H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" \
  -d '{
    "name": "Delhi Fulfilment Centre",
    "contact_person": "Ops Desk",
    "phone": "9000011122",
    "address": "1 Ring Road, Karol Bagh",
    "city": "New Delhi", "state": "Delhi", "pincode": "110001",
    "is_primary": true
  }'
```

```
{ "success": true, "status": 201,
  "data": { "warehouse_id": "8b1c...-uuid", "name": "Delhi Fulfilment Centre",
"is_default": true } }
```

2 **GET** `/warehouses` — list them

```
curl "$BASE/warehouses" -H "Authorization: Bearer $TOKEN"
```

3 **PUT** `/config` — make one the default (skip `warehouse_id` on every future create)

```
curl -X PUT "$BASE/config" \
  -H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" \
  -d '{ "default_warehouse_id": "8b1c...-uuid" }'
```

> This same `PUT /config` call is also where you register your **webhook URL** (Step 7) and a courier **priority list** so ShipSeth can auto-pick a courier when you don't send one.

Step 3 — Price the route

`/rates` returns every courier that can serve the route, priced, with an estimated delivery window. Pick one — or let ShipSeth pick for you at create time by skipping `rate_id`.

1 **POST** `/rates`

```
curl -X POST "$BASE/rates" \
  -H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" \
  -d '{
    "origin_pincode": "110001",
    "destination_pincode": "400001",
    "payment_mode": "prepaid",
    "parcel": {
      "weight": 1.2, "weight_unit": "kg",
      "dimensions": { "length": 20, "width": 15, "height": 10, "unit": "cm" }
    }
  }'
```

```
{ "success": true, "data": {
  "quote_id": "q-uuid", "expires_at": "2026-09-08T13:00:00+05:30",
  "options": [
    { "rate_id": "a1f9c2e7d80b34c1", "courier_code": "DELHIVERY",
      "courier_name": "Delhivery", "service": "Surface", "total": 78.40,
      "edd_min": 2, "edd_max": 4, "edd_source": "delhivery_api" }
  ] } }
```

`options[].rate_id` is an opaque key, stable for the life of the quote (a few minutes) — pass it straight into `POST /shipments` as-is.

> **No couriers returned?** The response includes `excluded[]` with a reason per courier (out of weight range, pincode not serviceable, direction not supported, …) — check that before assuming it's a bug.

Step 4 — Create the shipment (one call, async)

This is the one call that matters most. It validates everything, places a wallet hold, and hands the booking to a background worker — you get a `202` with a tracking number in well under a second, without waiting on the courier's own API.

1 **POST** `/shipments`

```
curl -X POST "$BASE/shipments" \
  -H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" \
  -H "Idempotency-Key: $(uuidgen)" \
  -d '{
    "order_reference": "ORD-1001",
    "payment_mode": "prepaid",
    "rate_id": "a1f9c2e7d80b34c1",
    "parcel": { "weight": 1.2,
      "dimensions": { "length": 20, "width": 15, "height": 10 } },
    "consignee": {
      "name": "Asha Rao", "address": "44 Linking Road",
      "pincode": "400001", "city": "Mumbai", "state": "Maharashtra",
      "phone": "9876500011"
    }
  }'
```

```
{ "success": true, "status": 202,
  "message": "Shipment accepted — booking in progress.",
  "data": {
    "shipment_id": "3d2e...-uuid",
    "tracking_number": "976120254",
    "order_reference": "ORD-1001",
    "direction": "forward", "payment_mode": "prepaid",
    "canonical_status": "PROCESSING", "environment": "live"
  } }
```

> **Always send a fresh** `Idempotency-Key` (any UUID) on every create call. If a request times out on your side and you retry with the *same* key, you get back the original result — never a duplicate shipment or a double charge.

Payment modes

PAYMENT_MODE	FORWARD SHIPMENT	REVERSE PICKUP
`prepaid`	✅ you already collected payment	✅ only mode allowed
`cod`	✅ courier collects `cod_amount` at delivery	❌ not allowed
`to_pay`	✅ courier collects **freight** at delivery (not goods value)	❌ not allowed

A shipment always places a wallet hold for the freight amount at create time. For `prepaid` / `cod` it settles the moment the courier confirms the booking. For `to_pay` it stays held until the parcel is delivered (released — you owe nothing) or fails delivery and goes to RTO (settled — you're billed the freight, same as if it were a normal prepaid shipment).

Reverse shipments (return pickups)

Set `"direction": "reverse"` and a `return_reason` to collect a parcel *from* a consumer and bring it back to your warehouse. The field meanings flip: `warehouse_id` is where it's **returned to**, and `consignee` is who it's **collected from**. Only prepaid is allowed, and only couriers proven to support reverse pickups (Delhivery, Ekart today) are offered.

Step 5 — Track it

Your shipment starts at `PROCESSING` the instant you create it. Once the courier confirms, it moves to `MANIFESTED` and a real waybill appears. From there it follows the normal delivery lifecycle.

1 **GET** `/shipments/track/<tracking_number or courier AWB>`

```
curl "$BASE/shipments/track/976120254" -H "Authorization: Bearer $TOKEN"
```


```
{ "success": true, "data": {
  "tracking_number": "976120254", "waybill": "38290100221",
  "canonical_status": "IN_TRANSIT", "current_status": "In transit to Mumbai hub",
  "eta": "2026-09-11",
  "milestones": [
    { "canonical_status": "MANIFESTED", "scanned_at": "2026-09-08T09:10:00+05:30" },
    { "canonical_status": "PICKED_UP",  "scanned_at": "2026-09-08T14:22:00+05:30" },
    { "canonical_status": "IN_TRANSIT", "scanned_at": "2026-09-09T06:00:00+05:30" }
  ] } }
```


Pass our `tracking_number` or the courier's AWB — whichever your system stores. Checking many at once? **POST** `/shipments/track` with `{"numbers": [...]}` takes up to 50; unknown ones come back in `not_found`.

Status vocabulary

STATUS	MEANING
`PROCESSING`	Accepted; the courier hasn't confirmed yet
`MANIFESTED`	Courier confirmed — waybill exists, label is ready
`PICKUP_SCHEDULED` → `PICKED_UP`	On its way to the courier network
`IN_TRANSIT` → `OUT_FOR_DELIVERY`	Moving toward the consignee
`DELIVERED`	Terminal — done
`NDR`	A delivery attempt failed — **action needed** (see “When delivery fails”)
`RTO_INITIATED` → `RTO_DELIVERED`	Returning to you
`CANCELLED`	Terminal — cancelled before pickup, charge reversed
`FAILED`	Terminal — booking failed; `failure.reason` says why
`LOST` / `DAMAGED`	The courier reports the parcel lost or damaged — raise a claim with your account manager

Branch on `canonical_status` only. `current_status` is the courier's own wording — show it, don't parse it. Treat any status you don't recognise as “in progress”.

Prefer a push? Set up a webhook once.

Instead of polling, register a URL via `PUT /config` and ShipSeth POSTs a signed event on every status change above — you'll never miss one.

```
curl -X PUT "$BASE/config" \
  -H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" \
  -d '{
    "webhook_url": "https://your-server.example/hooks/shipseth",
    "webhook_secret": "a-random-string-at-least-16-chars",
    "webhook_active": true
  }'
```

```
// POSTed to your webhook_url on every status change
{ "event": "shipment.status", "id": "delivery-uuid", "created": 1712345678,
  "data": {
    "tracking_number": "976120254", "waybill": "38290100221",
    "canonical_status": "IN_TRANSIT", "location": "Mumbai Hub",
    "scanned_at": "2026-09-09T06:00:00+05:30",
    "failure": null } }   // FAILED only: { "category": "...", "reason": "..." }
```

Deduplicate on `id` and order by `data.scanned_at` — a retried delivery can arrive late, so never move an order backwards.

VERIFY IT'S REALLY US (PHP)

```
function verify(string $sigHeader, string $rawBody, string $secret): bool {
    parse_str(str_replace(',', '&', $sigHeader), $parts);
    $expected = hash_hmac('sha256', ($parts['t'] ?? '') . '.' . $rawBody, $secret);
    return hash_equals($expected, (string) ($parts['v1'] ?? ''))
        && abs(time() - (int) ($parts['t'] ?? 0)) <= 300;
}
```

The signature arrives as `X-ShipSeth-Signature: t=<unix>,v1=<hmac>`. Reject anything older than 5 minutes to guard against replay — Node.js and Python versions of this exact check are in `webhooks.md`.

Step 6 — Print labels, manifest & invoices

Once a shipment reaches `MANIFESTED` (a real waybill exists), the label is ready.

1 **POST** `/shipments/<shipment_id>/label`

```
curl -X POST "$BASE/shipments/3d2e...-uuid/label" \
  -H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" \
  -d '{ "size": "4x6", "format": "base64" }'
```

```
{ "success": true, "data": {
  "size": "4x6", "format": "base64", "content": "JVBERi0xLjQK..." } }
```

Decode `data.content` as base64 to get the PDF bytes. Prefer a URL you can hand to a printer driver instead? Pass `"format": "url"` — you'll get back a short-lived signed link instead of the inline bytes.

SIZE	USE CASE
`a4`	Office printer, default
`4x6`	Thermal label printer — most common for courier labels
`4x2.5` / `3x2` / `2x3`	Smaller thermal formats

A whole dispatch batch at once

All three bulk calls take `{"shipment_ids": [...]}` (up to 100) and `"format": "base64" | "url"`, return **one merged PDF**, and list anything they couldn't include in `skipped` instead of failing.

CALL	WHAT YOU GET
POST `/labels`	Every label in the order you sent, plus `size`. Unbooked shipments are skipped with `not_booked`.
POST `/manifests`	The courier hand-over sheet — one page per warehouse × courier, AWB barcodes, COD to collect, signature boxes. Only booked shipments not yet picked up. The same shipments always print the same `manifest_number`.
POST `/invoices`	Your invoice for each parcel, from its `items[]`. It's a *Tax Invoice* when your GSTIN is known, with CGST + SGST or IGST worked out the same way as on the label. Works before the AWB exists.

```
curl -X POST "$BASE/manifests" \
  -H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" \
  -d '{ "shipment_ids": ["3d2e...", "7a91..."], "format": "url" }'
```

The response lists each `manifest_number` with its `shipment_ids`, plus `skipped` and the PDF (`content`).

When delivery fails — NDR

A failed delivery attempt arrives as a `NDR` webhook with the courier's reason in `remarks`. You have 48 hours to tell us what to do. After that, ShipSeth asks the courier to try again for you.

1 **GET** `/shipments/<id>/ndr`

```
{ "success": true, "data": { "attempts": [ {
  "attempt_number": 1, "reason": "Consignee not available",
  "status": "open", "actionable": true,
  "action_deadline": "2026-10-31T10:00:00+05:30" } ] } }
```

2 **POST** `/shipments/<id>/ndr/action`

```
curl -X POST "$BASE/shipments/3d2e...-uuid/ndr/action" \
  -H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" \
  -d '{ "action": "change_phone", "new_phone": "9876500011" }'
```

ACTION	NEEDS	THEN
`re_attempt`	optional `re_attempt_date`	Sent to the courier; status stays `NDR` until its next scan
`change_address`	`new_address`	Same as above
`change_phone`	`new_phone`	Same as above
`rto`	—	Parcel returns to you → `RTO_INITIATED` (RTO freight applies)

You never need an NDR id: the action always applies to the latest open attempt. If nothing is open you get `409 no_open_ndr`.

Testing end to end

A `"test": true` token books through a simulator: no courier, no wallet charge. Every other part of the API behaves the same. No courier scans a test parcel, though, so move it yourself to see each webhook:

```
curl -X POST "$BASE/test/shipments/3d2e...-uuid/scan" \
  -H "Authorization: Bearer $TEST_TOKEN" -H "Content-Type: application/json" \
  -d '{ "status": "NDR", "remarks": "Consignee not available" }'
```

`status` can be any of `PICKED_UP`, `IN_TRANSIT`, `OUT_FOR_DELIVERY`, `DELIVERED`, `NDR`, `RTO_INITIATED`, `RTO_DELIVERED`, `LOST` or `DAMAGED`. Skipped steps are walked for you, each with its own webhook. An `NDR` scan opens a real NDR you can act on.

> **Moving from another aggregator?** The `integration-playbook.md` next to this guide maps Shiprocket statuses and calls one-to-one, and lists the go-live checklist we sign off with you.

Optional — pickup & cancellation

Schedule pickup

Every shipment auto-schedules pickup with the courier for you by default. If you'd rather control the date yourself, pass `"auto_pickup": false` when creating, then call this once the shipment is `MANIFESTED`:

```
curl -X POST "$BASE/shipments/3d2e...-uuid/pickup" \
  -H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" \
  -d '{ "pickup_date": "2026-09-09" }'
```

Cancel a shipment

Works until the courier picks the parcel up (`MANIFESTED` or `PICKUP_SCHEDULED`). Cancel before the courier confirms and nothing is charged; cancel after booking and the freight is refunded to your wallet. Once it's `PICKED_UP` you get `422 not_cancellable`.

```
curl -X POST "$BASE/shipments/3d2e...-uuid/cancel" \
  -H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" \
  -d '{ "reason": "Customer changed their mind" }'
```

Errors, rate limits & good habits

Response envelope

Every response — success or error — has the same shape, plus a `request_id` to quote when you contact support.

```
// success
{ "success": true,
  "status": 202,
  "message": "…",
  "data": { ... },
  "request_id": "…" }
```

```
// error
{ "success": false,
  "status": 422,
  "code": "eway_bill_required",
  "message": "…",
  "errors": { "eway_bill": [...] },
   "request_id": "…" }
```

Common error codes

CODE	STATUS	WHAT IT MEANS
`insufficient_balance`	402	Top up your wallet — `data.required_paise` / `available_paise` tell you by how much
`rate_unavailable`	409	The `rate_id` you sent expired — re-run `/rates` and retry (fresh options are included)
`no_courier` / `no_reverse_courier`	422	Nobody serves this route/direction — check `excluded[]`
`eway_bill_required`	422	Invoice value is over ₹50,000 — an e-way bill number is mandatory
`idempotency_key_required`	422	You forgot the `Idempotency-Key` header on `/shipments`
`too_many_in_flight`	429	Too many shipments awaiting booking at once — wait for `Retry-After` seconds
`kyc_required`	403	Finish account KYC in the ShipSeth dashboard before booking
`duplicate_order_reference`	409	A live shipment already exists for this order — you retried with a new `Idempotency-Key`. Use `data.shipment_id`
`no_open_ndr`	409	NDR action on a shipment with no open NDR

The full, closed vocabulary of error codes lives in `errors.md` next to this guide.

Rate limits

ENDPOINT	LIMIT
`/auth/login`	5 / min
`/rates`	120 / min
`/shipments` (create), labels, manifests, invoices	60 / min
`/shipments/track/*`, bulk track	300 / min
`/shipments/*/ndr/action`	30 / min
everything else	120 / min

A `429` response includes `Retry-After` and `X-RateLimit-*` headers — back off and retry rather than hammering.

Good habits

- Always send `Idempotency-Key` on `/shipments`, one per order, and **reuse the same key** when you retry that order.
- Log our `request_id` with every call — support finds any request from it.
- Prefer webhooks over polling once you're past prototyping.
- Treat `rate_id` as short-lived — price right before you create, don't cache it for hours.
- Keep a separate token per server/environment (`label` on login) so you can revoke one without affecting the others.
- Use `"test": true` on login while building — no wallet impact, real response shapes.

Endpoint quick reference

CALL	PURPOSE
POST `/auth/login`	Get a token
POST `/auth/rotate` · `/revoke`	Rotate or kill a token
GET `/warehouses` · POST	List / create pickup warehouses
GET `/config` · PUT	Default warehouse, webhook, courier priority
POST `/rates`	Price a route
POST `/shipments`	Create + book (one call, async)
GET `/shipments` · `/shipments/<id>`	List / fetch a shipment
GET `/shipments/track/<number>`	Track by tracking number or courier AWB
POST `/shipments/track`	Bulk track (up to 50)
POST `/shipments/<id>/label`	Get the shipping label
POST `/labels`	Bulk labels — one PDF (up to 100)
POST `/manifests`	Courier pickup manifest
POST `/invoices`	Seller invoices — one PDF
GET `/shipments/<id>/ndr` · POST `…/ndr/action`	Read / act on a failed delivery
POST `/test/shipments/<id>/scan`	Sandbox: move a test shipment
POST `/shipments/<id>/pickup`	Schedule pickup
POST `/shipments/<id>/cancel`	Cancel a shipment
GET `/webhooks/deliveries`	See recent webhook delivery attempts

> **Interactive reference & more guides:** shipseth.com/developers — full OpenAPI reference, plus dedicated pages on idempotency, pagination, test mode and the deprecation policy. Raw spec: openapi.yaml.