Team Mng Into

Author: Josh S. Sakweli, Backend Lead Team
Last Updated: 2026-11-17
Version: v1.0 (replaces the single `docs/team/team_api_doc.md` v0.7)

Base URL: `https://api.nexgate.co/api/v1` (production) — `https://dev.api.nexgate.co/api/v1` (staging) — `http://localhost:8765/api/v1` (local)

Short Description: Shops and delivery businesses can have a team. The owner invites people, gives them a role, links shared computers (terminals) and decides what each computer may do. Riders work from their own phone in the shipper app; shop staff and dispatchers work only on terminals. This page explains the ideas once and tells each app team which document is theirs.

Hints:

- **Read only your app's document.** Each one lists every call that app makes, written for that app.
- When the team feature is switched off (production for now), every team endpoint answers **404**.
- A person is identified **only by** `accountId`. Never send or store a username to point at a person.
- Coded refusals are HTTP **409** with `data.code`: switch on the code, never on the message.
- Times without a zone are East Africa Time (`2026-11-17T10:30:45`). Times with a zone are ISO 8601.

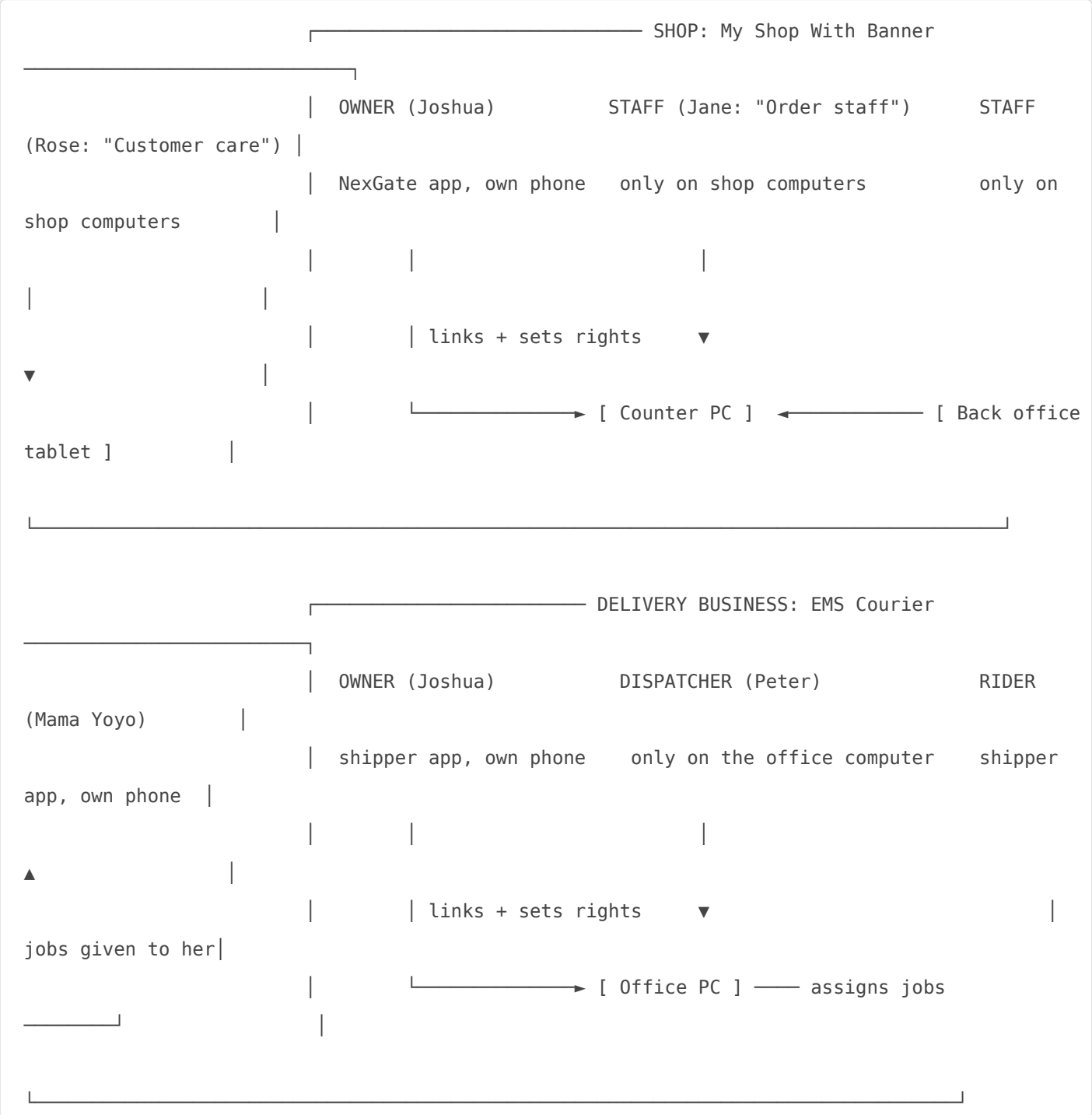
Which document is mine?

I build…	My document	Who uses those screens
NexGate app (Android, iOS, web)	`nexgate_app_team_api.md`	shop owners (team, roles, shop computers, calls), anyone invited to a team, buyers (who is bringing my parcel, calling the delivery)
Shipper app (Android)	`shipper_app_team_api.md`	delivery business owners (team, riders, office computers, calls), riders (my jobs, duty, calls)
Terminal app (Windows / Mac desktop, shop tablet)	`terminal_app_api.md`	the shop counter computer and the shipping office computer: shop staff, dispatchers, and owners when they sit there

Related:

- **Pushes:** `docs/notifications/push_contracts_api_doc.md` (all apps).
- **Product rules:** `docs/team/SHOP_TEAM.md`, `docs/team/SHIPPER_TEAM.md`, `docs/team/WORK_CALLS.md`.
- **What changed for existing screens:** `docs/changelog/frontend_changelog_2026-11-17.md`.

The picture



Key ideas

Places and roles

A **place** is a shop or a delivery business. In paths it appears as `{kind}/{placeId}`:

`{kind}`	`{placeId}`	Roles a person can have there
`shops`	the shop id	`OWNER`, `STAFF` (with a shop role, see below)
`businesses`	the delivery business id	`OWNER`, `DISPATCHER`, `RIDER`

- The **owner** is the person who created the place. They always hold every right and can never be removed, locked, given another role, or leave.
- **Business roles are fixed.** A dispatcher accepts offers and assigns riders on the office computer. A rider carries the jobs given to them, from their own phone, and **never sees money**.
- **Shop roles are built by each shop** from the rights NexGate defines. Three ready-made roles (Order staff, Customer care, Shop manager) appear on first use and can be edited or deleted.

Shop rights

Right	Allows
`SEE_ORDERS`	view the shop's orders
`HANDLE_ORDERS`	accept, mark Ready, hand over with the shipper's code, collect
`CUSTOMER_CHAT`	read and reply to customers as the shop
`PRICE_OFFERS`	send and cancel a special-price offer in chat
`ANSWER_CALLS`	answer and make calls as the shop
`PRODUCTS`	add products; edit names, photos, descriptions (never prices)
`STOCK`	change quantities, low-stock level, out of stock

Never a right (owner only): catalogue prices and selling terms, money, cancelling orders, shop settings, team, roles, terminals.

Business rights (fixed by role)

Right	Owner	Dispatcher	Rider
`VIEW_JOBS`	✅	✅	only jobs given to them
`ANSWER_OFFERS`	✅	✅	✅
`ASSIGN_JOBS`	✅	✅	✅
`ANSWER_CALLS`	✅	✅	only calls about their own jobs
`VIEW_EARNINGS`	✅	✅	✅

Terminals (shared computers)

A **terminal** is a computer or tablet linked to **one** place. The owner links it from her own phone and ticks what it may do (its **permissions**). People tap their face on it and type their **own PIN**.

```
What a person may do ON a terminal  =  their rights  ∩  the terminal's permissions

Jane ("Order staff": SEE_ORDERS, HANDLE_ORDERS)  on  Counter PC (HANDLE_ORDERS,
RECEIVE_CALLS)
     → SEE_ORDERS, HANDLE_ORDERS        (no ANSWER_CALLS: Jane's role doesn't have it)
```

Terminal permission	Kind	Gives (if the person also has it)
`ORDER_ALERTS`	shop	`SEE_ORDERS` + new-order pop-ups
`HANDLE_ORDERS`	shop	`SEE_ORDERS`, `HANDLE_ORDERS`
`CUSTOMER_CHAT`	shop	`CUSTOMER_CHAT`, `PRICE_OFFERS`
`RECEIVE_CALLS`	shop and business	`ANSWER_CALLS` + incoming call rings
`PRODUCTS_STOCK`	shop	`PRODUCTS`, `STOCK`
`GET_OFFERS`	business	`ANSWER_OFFERS` + offer rings
`ASSIGN_RIDERS`	business	`ASSIGN_JOBS`

The three kinds of caller

Every request is one of these. The server checks which on **every** call.

```
┌──────────────────┬──────────────────────────────────────┬──────────────────┐
│ Caller           │ Sends                                │ Reaches          │
```

```
├──────────────────────┼──────────────────────────────────────────┼──────────────────────
──────────────┤
 | A person, own phone    | Authorization: Bearer <person token>     | everything, as
today                    |
 | The terminal alone     | X-Terminal-Key: <terminal key>           | only the terminal's
own calls            |
 |                        |                                          | (faces, sign in,
settings, live stream)  |
 | A person on a terminal | Authorization: Bearer <member token>     | only that terminal's
allow-list,          |
 | (member session)       | + X-Terminal-Key: <terminal key>         | only for that place,
only with the right |

└──────────────────────┴──────────────────────────────────────────┴──────────────────────
──────────────┘
```

Anything not on a terminal's allow-list answers **403** before any feature code runs. A member token without its terminal's key answers **401**.

Activity

Every team action is recorded with who did it and on which terminal. **The owner sees everyone; a member sees only their own.** Kept one year; nobody can edit or delete a row, not even the owner.

Standard Response Format

Success Response Structure

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Invite sent",
  "action_time": "2026-11-17T10:30:45",
  "data": { }
}
```

Coded refusal (HTTP 409)

```
{
  "success": false,
  "httpStatus": "CONFLICT",
  "message": "This person is already invited or on the team",
  "action_time": "2026-11-17T10:30:45",
  "data": { "code": "ALREADY_INVITED", "reason": "This person is already invited or on the team" }
}
```

Some refusals carry `data.details` (for example `WRONG_PIN` → `details.triesLeft`).

Standard Response Fields

Field	Type	Description
`success`	boolean	`true` for success, `false` for errors
`httpStatus`	string	HTTP status name (`OK`, `CONFLICT`, `NOT_FOUND`, …)
`message`	string	Human-readable result; safe to show
`action_time`	string	When the response was made
`data`	object/string	The payload, or the error details

Standard Error Types

- `400 BAD_REQUEST`: invalid request data
- `401 UNAUTHORIZED`: missing, invalid or expired token; a member token without its terminal key; a session that ended
- `403 FORBIDDEN`: a terminal calling something not on its allow-list, or for another place
- `404 NOT_FOUND`: the thing doesn't exist, isn't yours, or the team feature is off
- `409 CONFLICT`: a coded refusal (`data.code`), listed per endpoint
- `422 UNPROCESSABLE_ENTITY`: validation failed (`data` has one message per field)
- `429 TOO_MANY_REQUESTS`: rate limit

HTTP Method Badge Standards

- **GET** - GET - read
- **POST** - POST - create / act

- **PUT** - `PUT` - replace
- **PATCH** - `PATCH` - change some fields
- **DELETE** - `DELETE` - remove

All refusal codes

Code	Meaning	Where
`NOT_THE_OWNER`	only the owner does this	team, roles, terminals, settings
`INVITE_WHO`	send exactly one of `accountId` / `phone`	invite
`ROLE_NOT_FOR_BUSINESSES`, `ROLE_NOT_FOR_SHOPS`, `SHOP_ROLE_REQUIRED`	wrong kind of role for this place	invite, change role
`CANNOT_INVITE_YOURSELF`	the owner invited their own account or number	invite
`ALREADY_INVITED`	already invited or on the team	invite
`TOO_MANY_INVITES`, `TOO_MANY_PHONE_INVITES`	20 invites per place per hour, 5 of them by phone	invite
`PHONE_INVALID`	the number can't be read	invite
`INVITE_NOT_OPEN`, `INVITE_EXPIRED`	withdrawn / answered, or older than 6 hours	accept, decline, withdraw
`ALREADY_ON_TEAM`	accepting a phone invite while already on that team	accept
`OWNER_ROLE_FIXED`, `OWNER_CANNOT_BE_REMOVED`, `OWNER_CANNOT_LEAVE`	the owner's own row	team
`NOT_ON_TEAM`, `MEMBER_STATE`	not on the team (any more); wrong state (e.g. unlocking someone not locked)	team
`ROLE_NAME_TAKEN`, `ROLE_EMPTY`, `ROLE_IN_USE`	shop roles	roles
`MEMBER_REQUIRED`, `OWN_ACTIVITY_ONLY`	activity	activity
`RIDER_NOT_AVAILABLE`, `RIDER_OFF_DUTY`, `JOB_NOT_ASSIGNABLE`, `ASSIGN_RACE`	giving a job to a rider	riders
`RIDER_HOLDS_PARCEL`, `PARCEL_IN_HAND`, `RIDERS_ONLY`	a rider holding a parcel; rider-only calls	riders
`PLATFORM`, `SLOW_DOWN`, `CODE_USED`, `TOO_MANY_TERMINALS`, `PERMISSION_NOT_HERE`	linking a terminal	terminals
`ROLE_NOT_FOR_TERMINAL`, `NOT_ACTIVE_HERE`	adding yourself to / signing in on a terminal	terminals

Code	Meaning	Where
`WRONG_PIN`, `PIN_LOCKED`, `PIN_TOO_MANY`, `SIGN_IN_AGAIN`	PIN and session	terminals
`ORDER_TAKEN`, `ORDER_FINISHED`	packing tags	shop orders
`NO_ORDER`, `NO_SHIPPER_YET`, `CALLS_CLOSED`, `NOT_ON_THIS_ORDER`, `NOT_YOUR_JOB`, `NOT_YOUR_ORDER`, `USE_SHOP_CALLS`, `NOT_FOR_SHIPPERS`, `SAME_PERSON`, `OWN_BUSINESS`, `OWN_SHOP`, `BUSY`, `ACT_AS_PLACE`, `NOT_YOUR_SHOP`, `NOT_YOUR_BUSINESS`, `NOT_ON_THIS_CALL`, `ANSWERED_ELSEWHERE`, `NO_PLACE`	calls about a parcel	work calls

Revision #2
Created 17 November 2026 09:22:10 by Admin Qbit
Updated 17 November 2026 09:23:20 by Admin Qbit