

# Roles & permissions

Who can do what in the console, and what data each employee sees.

- [Roles and scope: the two ideas](#)
- [Scope: what data an employee can see](#)
- [The role catalogue](#)
- [Access by section](#)
- [Common role combinations](#)
- [How scope works in practice](#)
- [Managing roles from the console](#)
- [Setting an employee's scope](#)
- [Individual permission exceptions](#)
- [Requesting a new or changed role](#)

# Roles and scope: the two ideas

“ **Audience:** Platform administrators who decide who can do what in the console.  
**You need:** the Assignments and Scope sections, available to `admin` and `employee_manager` .

Access in the console is built on two independent ideas. Keeping them apart is the key to the whole model.

| Concept      | What it defines                                                | Where it's configured                 |
|--------------|----------------------------------------------------------------|---------------------------------------|
| <b>Roles</b> | What an employee <b>can do</b> — actions and modules           | Assigned to the employee; one or more |
| <b>Scope</b> | What data an employee <b>can see</b> — brands, stores, sources | Configured once, per employee         |

An employee can hold several roles at once. The permissions from all of them are combined. The scope, by contrast, is always **shared** — it applies equally to everything that employee can do.

## Why this separation matters

It means you never have to choose between "can operate the hub" and "can only see two stores". Those are answered separately:

- Give them `hub_operator` → they can work the hub.
- Set their scope to stores 1 and 2 → they only ever see those two stores' data, in the hub and everywhere else.

Adding a second role later — reports, for example — does not widen what data they see. The scope still holds.

## The three things you'll configure

| You want to...                                     | Go to                                            |
|----------------------------------------------------|--------------------------------------------------|
| Let someone do something new                       | <a href="#">Managing roles from the console</a>  |
| Change what data they can see                      | <a href="#">Setting an employee's scope</a>      |
| Grant or block one specific ability, just for them | <a href="#">Individual permission exceptions</a> |

Roles decide what someone can do; their staff record is separate — see [Employees: what this area covers](#).

# Where to read next

- [Scope: what data an employee can see](#) — the two modes and the three dimensions
- [The role catalogue](#) — every role and who it's for
- [Access by section](#) — the full grid of role against console section

# Scope: what data an employee can see

Every employee has a single scope, which controls what data is visible to them everywhere in the console.

## The two modes

| Mode     | What it means                                               |
|----------|-------------------------------------------------------------|
| Customer | Full access to all data across the entire chain             |
| Explicit | Access restricted to specific brands, stores and/or sources |

## The three dimensions of an explicit scope

When scope is explicit, you can combine these filters:

| Dimension | What it filters                   | Example                  |
|-----------|-----------------------------------|--------------------------|
| Brands    | Which brands are visible          | Brand "X" only           |
| Stores    | Which specific stores are visible | Stores 1, 2 and 5        |
| Sources   | Which order sources are visible   | Only Glovo and Uber Eats |

### How they combine:

- Each dimension narrows the view further — they stack, they don't widen each other.
- Leaving a dimension empty means "no restriction" for that dimension.
- Store groupings — predefined sets of stores — are supported, and combine with individually chosen stores.

# Worked examples

| Configuration                           | What the employee sees                                                 |
|-----------------------------------------|------------------------------------------------------------------------|
| Brand = "Example Brand"                 | All stores and sources for that brand                                  |
| Stores = 1, 2, 3                        | All brands operating in those stores                                   |
| Brand = "Example Brand" + Stores = 1, 2 | Only that brand's data, in those two stores                            |
| Stores = 1 + Sources = Glovo, Uber      | Only Glovo and Uber data in store 1, wherever source filtering applies |

## A note on Sources

The source filter is a complementary dimension with **limited activation**. Not every module uses it — it only takes effect in sections where filtering by order origin is meaningful. Those include:

- **Reports** — sales data for specific aggregators only
- **Metrics** — order timelines and logs filtered to certain channels
- **CMS** — content configuration scoped to specific sources
- **Hub** — special cases where an operator only handles orders from certain origins

In modules where source filtering is not active, the employee sees all orders within their brand and store scope, regardless of this setting.

“ This makes Sources best suited to specialised roles that need a narrower view inside one particular section — not as a general-purpose restriction.

## What scope does not do

Scope never grants anything. It only narrows. An employee with a store-limited scope and no reporting role still cannot open reports — scope decides *what data*, roles decide *what actions*. See

[Roles and scope: the two ideas](#).

# The role catalogue

Each role covers one area of responsibility. If someone needs two areas, give them two roles — see [Common role combinations](#).

| Role                               | For whom                      | What it covers                                                                                                                                   |
|------------------------------------|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>admin</code>                 | Customer administrator        | Everything except managing role definitions. The highest role assigned to customer employees                                                     |
| <code>hub_manager</code>           | Operations manager            | Full hub — assign, force delivery, cancel, modify ETA, incidents, alarms — plus store configuration, pause delivery, metrics and archived alarms |
| <code>hub_operator</code>          | Hub shift operator            | Basic hub operations, without the critical actions. View stores and metrics                                                                      |
| <code>branch_manager</code>        | Store manager                 | Their own store: view orders, modify ETA, incidents, alarms, driver alerts, pause delivery, store configuration                                  |
| <code>branch_viewer</code>         | Read-only store access        | Viewing store configuration only. A support role, designed to be combined with others                                                            |
| <code>dispatcher_manager</code>    | Logistics technician          | Dispatcher execution, scheduler, store configuration, pause delivery, delivery zones, third-party carriers, metrics                              |
| <code>report_viewer</code>         | Management and analysts       | Reports, metrics, view stores                                                                                                                    |
| <code>polygon_manager</code>       | Delivery zone configuration   | Delivery zones, view stores                                                                                                                      |
| <code>cms_editor</code>            | Content editor                | Complaints tree, resources, images                                                                                                               |
| <code>employee_manager</code>      | HR and staff configuration    | Employees and own drivers, plus role assignments, scope and exceptions                                                                           |
| <code>third_party_manager</code>   | External fleet configuration  | External carrier configuration, at brand level                                                                                                   |
| <code>alarms_config_manager</code> | Alarm threshold configuration | Alarm thresholds for orders, drivers and stores, across the whole chain                                                                          |

## Two roles you won't assign

`super_admin` is a PizzaGest internal technical role. It has full access and bypasses every check, and is never assigned to customer employees.

`admin` is the customer-side equivalent for day-to-day purposes: the same operational reach, except that individual denies still apply to it and it cannot change role definitions.

## How permissions add up

Roles are built from a catalogue of atomic permissions — currently more than fifty, each covering one specific action such as "assign a driver" or "export metrics".

A role is a named subset of that catalogue. When an employee holds several roles, their permissions are combined, without duplicates. Nothing is subtracted by adding a role.

“ Permission counts per role change as new modules ship. Rather than memorising a number, check what a role actually grants in [Access by section](#), or open the role in the Role catalogue screen.

## Recently added modules

Chatbot configuration, External API management, the Voice agent, Promotional codes and Voucher codes are currently reachable by `admin` only. Whether they get a dedicated role, or are folded into existing ones, will be decided based on operational need.

If one of these needs to reach a non-admin employee before then, an individual exception is the intended route — see [Individual permission exceptions](#).

# Access by section

The full grid of what each role reaches. **Legend:** □ read-only · ⇐ read and write · — no access.

All access shown here is then filtered by the employee's scope — see [Scope: what data an employee can see](#).

| Section                       | admin | hub_manager | hub_operator | dispatcher_manager | branch_manager | report_viewer | polygon_manager | cms_editor | employee_manager | third_party_manager | alarms_config_manager |
|-------------------------------|-------|-------------|--------------|--------------------|----------------|---------------|-----------------|------------|------------------|---------------------|-----------------------|
| Hub — orders and status       | ⇐     | ⇐           | ⇐ basic      | —                  | □              | —             | —               | —          | —                | —                   | —                     |
| Hub — force delivery / cancel | ⇐     | ⇐           | —            | —                  | —              | —             | —               | —          | —                | —                   | —                     |
| Hub — modify ETA              | ⇐     | ⇐           | —            | —                  | ⇐              | —             | —               | —          | —                | —                   | —                     |
| Branch alarms                 | ⇐     | ⇐           | □            | —                  | □              | —             | —               | —          | —                | —                   | —                     |
| Driver alerts                 | ⇐     | □           | □            | —                  | □              | —             | —               | —          | —                | —                   | —                     |
| Alarm threshold config        | ⇐     | —           | —            | —                  | —              | —             | —               | —          | —                | —                   | ⇐                     |
| Archived alarms               | ⇐     | □           | —            | —                  | —              | —             | —               | —          | —                | —                   | —                     |
| Dispatcher                    | ⇐     | —           | —            | ⇐                  | —              | —             | —               | —          | —                | —                   | —                     |
| Scheduler                     | ⇐     | —           | —            | ⇐                  | —              | —             | —               | —          | —                | —                   | —                     |
| Store config                  | ⇐     | ⇐           | □            | ⇐                  | ⇐              | □             | □               | —          | —                | —                   | —                     |

[illegible]

| Section                        | admin | hub_manager | hub_operator | dispatcher_manager | branch_manager | report_viewer | polygon_manager | cms_editor | employee_manager | third_party_manager | alarms_config_manager |
|--------------------------------|-------|-------------|--------------|--------------------|----------------|---------------|-----------------|------------|------------------|---------------------|-----------------------|
| Promotional codes <sup>1</sup> | ⇒     | —           | —            | —                  | —              | —             | —               | —          | —                | —                   | —                     |
| Voucher codes <sup>1</sup>     | ⇒     | —           | —            | —                  | —              | —             | —               | —          | —                | —                   | —                     |
| Role and assignment management | ⇒     | —           | —            | —                  | —              | —             | —               | —          | ⇒                | —                   | —                     |

<sup>1</sup> **Recently added modules**, currently reachable by `admin` only. See [The role catalogue](#).

## Notes on reading this table

- `super_admin` has ⇒ on every row and is omitted to keep the table readable.
- `branch_viewer` is a read-only support role covering store configuration only; it is omitted for the same reason. Combine it with another role rather than using it alone.
- "⇒ basic" for `hub_operator` means the everyday hub actions, but **not** the critical ones — force delivery, cancel and modify ETA are excluded.
- **Role and assignment management** splits in two: `employee_manager` can assign existing roles, set scope and manage exceptions, but cannot create or edit role definitions. See [Managing roles from the console](#).

## If a role is nearly right

Don't look for a role that matches a person exactly. Either combine two roles ([Common role combinations](#)), or keep the closest role and add a single exception ([Individual permission exceptions](#)).

# Common role combinations

Because an employee can hold several roles and all of them share one scope, most real profiles are built by combining two or three roles rather than looking for a perfect single match.

| Real profile                                    | Roles                                                    | Scope                          |
|-------------------------------------------------|----------------------------------------------------------|--------------------------------|
| Operations manager who also views reports       | hub_manager + report_viewer                              | Customer, or explicit by store |
| Operations manager who manages alarm thresholds | hub_manager + alarms_config_manager                      | Customer, or explicit by store |
| Operations manager who also runs the dispatcher | hub_manager + dispatcher_manager                         | Customer, or explicit by store |
| Shift operator who also views reports           | hub_operator + report_viewer                             | Explicit — their stores        |
| Store manager with access to reports            | branch_manager + report_viewer                           | Explicit — their stores        |
| Brand manager who doesn't operate orders        | employee_manager + polygon_manager + third_party_manager | Explicit — brand               |
| Delivery zone technician                        | polygon_manager                                          | Explicit — brand               |
| Management, data only                           | report_viewer                                            | Customer                       |
| Full administrator                              | admin                                                    | Customer                       |

## Why hub\_manager doesn't include the dispatcher

hub\_manager deliberately focuses on hub operations. If your operations manager also needs the dispatcher, the scheduler or third-party carriers, assign hub\_manager + dispatcher\_manager.

Each role keeps its own area of responsibility, and the permissions combine automatically. This is preferred over widening hub\_manager, because it keeps the two responsibilities visible and separately removable.

## Rules of thumb

**Combine roles, don't request new ones.** Most needs are a combination that already exists. A new role definition requires a request to the development team — see [Requesting a new or changed role](#).

**Roles only ever add.** Assigning a second role never reduces what the first one granted, and never widens what data is visible. Only scope controls that.

**Use exceptions for the one-off case.** If a profile matches an existing role except for a single ability, keep the role and add an individual grant or deny — see [Individual permission exceptions](#).

# How scope works in practice

Four behaviours explain almost every "why can this person see that?" question.

## Roles are additive

An employee with `hub_manager` + `report_viewer` can operate the hub **and** view reports. Neither role restricts the other, and the order you assign them in makes no difference.

## Scope filters data automatically

Having access to a module does not mean seeing all of its data. The system filters everything to the employee's scope.

A `report_viewer` whose scope is limited to the brand "Pizza Express" opens the same reports screen as everyone else, but sees only that brand's figures. Nothing is hidden or greyed out — the data simply isn't in their results.

“ This is worth explaining to people directly. Two colleagues comparing the same report and seeing different totals is expected behaviour, not a fault.

## Scope applies equally to every role

Unlike older systems, scope is **not** configured per role. If an employee can see stores 1, 2 and 3, that applies to everything they do — hub, reports, metrics, store configuration, all of it.

This is a deliberate simplification. It means you can reason about one employee's visibility by looking at one setting, rather than reconciling several.

# Individual exceptions override roles

You can grant or deny one specific permission to one specific employee, optionally with an expiry date.

**A deny always wins**, whatever their roles grant. See [Individual permission exceptions](#).

## Putting it together

When someone reports they can't do something, check in this order:

1. **Do they have a role that grants the action?** → [Access by section](#)
2. **Is there a deny exception on them?** → [Individual permission exceptions](#)
3. **Is the data simply outside their scope?** → [Setting an employee's scope](#)

Most reports resolve at step 3, and look like a missing permission until you check.

# Managing roles from the console

The console manages roles, assignments, scopes and individual exceptions through four sections.

| Section               | What it lets you do                                                                                       | Required permissions                                       |
|-----------------------|-----------------------------------------------------------------------------------------------------------|------------------------------------------------------------|
| <b>Role catalogue</b> | View, create, edit and delete role definitions — their name and the permissions they include              | <code>roles:read</code> / <code>roles:write</code>         |
| <b>Assignments</b>    | Assign and unassign roles to employees, activate and deactivate them                                      | <code>employees:read</code> / <code>employees:write</code> |
| <b>Scope</b>          | Configure what data each employee can access — brands, stores, sources                                    | <code>employees:read</code> / <code>employees:write</code> |
| <b>Exceptions</b>     | Create individual grants or denies on a specific permission for an employee, with an optional expiry date | <code>employees:read</code> / <code>employees:write</code> |

## Who can reach each section

**Role catalogue** is restricted to `super_admin` — PizzaGest's own technical role. Creating or changing a role definition goes through the development team, who review the change before applying it. See [Requesting a new or changed role](#).

**Assignments, Scope and Exceptions** are available to `admin` and `employee_manager`. These are the three you will use day to day: assigning existing roles, setting scope, and managing exceptions.

“ This split is deliberate. Changing what a role *means* affects every employee holding it, across the whole chain — so it is reviewed. Changing who *holds* a role affects one person, and is yours to do directly.

## Assigning a role to an employee

1. Open **Assignments**.
2. Find the employee.
3. Add the role or roles they need — see [The role catalogue](#) and [Common role combinations](#).
4. Confirm their scope separately, under **Scope**. A new role does not change existing scope.

An assignment can be deactivated rather than removed, which keeps the record while suspending the access.

“ Assigning a role does not give someone a way to sign in. Setting or resetting their password is a separate task, done from the Employees screen — see [Employees - Password](#).

## Safeguards

These protect you from the changes that are hardest to undo:

- A role with active assignments **cannot be deleted**.
- Permissions attached to a role are validated against the system permission catalogue, so a typo can't create a permission that does nothing.
- Every assignment and every exception records **who created it**, for auditing.
- Individual denies always override what a role grants.

## Related pages

- [Setting an employee's scope](#) — the three dimensions in practice
- [Individual permission exceptions](#) — grants and denies
- [Requesting a new or changed role](#) — when no combination fits

# Setting an employee's scope

“ **You need:** `employees:read` and `employees:write` — held by `admin` and `employee_manager` .

Scope is set once per employee and applies to everything they do. See [Scope: what data an employee can see](#) for the model; this page is the task.

## Before you start

Decide which of the two modes the person needs:

- **Customer** — they legitimately need the whole chain. Management, analysts, chain-wide operations.
- **Explicit** — everyone else. Anyone tied to particular stores or a particular brand.

When in doubt, choose explicit and add to it. Widening a scope later is a one-field change; discovering someone has been seeing another brand's figures for months is not.

## Setting an explicit scope

1. Open **Scope** and find the employee.
2. Switch the mode to **Explicit**.
3. Fill only the dimensions you actually want to restrict:
  - **Brands** — leave empty for "all brands".
  - **Stores** — individual stores, store groupings, or both.
  - **Sources** — only if the restriction genuinely matters; see the caution below.
4. Save, and confirm the result on a screen the employee uses, such as Reports.

“ Leaving a dimension empty means **no restriction on that dimension** — not "no access". An explicit scope with every dimension empty is equivalent to full access.

# Using store groupings

Groupings are predefined sets of stores. They combine with individually selected stores rather than replacing them, so you can give someone a region plus one extra store without listing the region store by store.

Prefer a grouping over a long list of store IDs: when a store is added to the region later, the employee's scope follows automatically.

## A caution on Sources

The source filter only takes effect in modules that implement it — Reports, Metrics, CMS and some Hub cases. Elsewhere the employee sees everything within their brand and store scope regardless.

This means a source restriction is **not** a reliable way to hide data in general. Use it to focus a specialised role, not to enforce a boundary. If the boundary matters, express it with brands or stores.

## Checking your work

The fastest verification is to look at a screen that shows data volume, such as Reports or the hub order list, and confirm the employee sees what you intended — no more, no less.

Remember that scope does not grant anything. If they see nothing at all, check their roles before adjusting the scope — see [How scope works in practice](#).

# Individual permission exceptions

“ You need: `employees:read` and `employees:write` — held by `admin` and `employee_manager` .

An exception grants or denies one specific permission to one specific employee, on top of whatever their roles say. Use it when a person matches an existing role in every respect but one.

## The two kinds

| Kind  | Effect                                                 | Typical use                                                                                  |
|-------|--------------------------------------------------------|----------------------------------------------------------------------------------------------|
| Grant | Adds one permission the employee's roles don't include | A shift operator who also needs to cancel orders, without giving them the whole manager role |
| Deny  | Removes one permission the employee's roles do include | A store manager who should not be able to pause delivery                                     |

## The one rule that decides conflicts

**A deny always wins.** It beats every grant, from every role the employee holds, and from an explicit grant exception too. There is no ordering to reason about and no priority to configure.

This is deliberate: it means a deny is a reliable way to remove an ability, without auditing which roles might be putting it back.

“ `super_admin` is the single exception — it bypasses all checks, including denies. `admin` does **not**; individual denies apply to it normally.

# Creating one

1. Open **Exceptions** and find the employee.
2. Choose the permission.
3. Choose grant or deny.
4. Optionally set an expiry date.
5. Save. The record keeps who created it, for auditing.

## Use the expiry date

Most exceptions are temporary in intent — cover for an absence, a seasonal push, a trial of wider access. Very few are temporary in practice, because nobody remembers to remove them.

Setting an expiry date when you create the exception turns that problem into a non-problem. If the access turns out to be permanent, the right fix is a role change, not an exception that never ends.

## When not to use an exception

**When two roles already cover it.** If the person needs a whole area, assign the role for that area — see [Common role combinations](#). Exceptions are for single permissions, not for assembling an area out of parts.

**When several people need the same thing.** Three identical exceptions are a sign the role is wrong. Request a role change instead — see [Requesting a new or changed role](#).

**To restrict data.** Exceptions control actions, not visibility. To limit what someone can see, set their scope — see [Setting an employee's scope](#).

## Reviewing them

Exceptions are the part of the access model most likely to drift, because each one is invisible from the role catalogue. Review the exceptions list periodically and remove the ones whose reason has passed.

# Requesting a new or changed role

Creating or editing a role definition is restricted to PizzaGest's technical team. As a customer administrator you assign existing roles, set scope and manage exceptions — but the role catalogue itself goes through a request.

This page explains why, and how to make the request land well.

## Why it's reviewed

A role definition is shared. Changing what `hub_manager` means changes it for every employee holding it, in every store, immediately. Unlike an assignment — which affects one person and is easy to reverse — a role change has chain-wide reach, so it is reviewed before it is applied.

## Try these first

Most requests turn out to be unnecessary. In order of preference:

1. **Combine existing roles.** Two or three roles together cover most real profiles — see [Common role combinations](#).
2. **Add an individual exception.** If an existing role fits except for one ability, grant or deny that one permission — see [Individual permission exceptions](#).
3. **Check it's not a scope problem.** "They can't see the data" is usually scope, not permissions — see [How scope works in practice](#).

A new role is the right answer when the same combination-plus-exception pattern keeps recurring across several people.

## What to include in the request

| Detail                                  | Why it matters                                                           |
|-----------------------------------------|--------------------------------------------------------------------------|
| The job, in plain words                 | "Regional supervisor for 12 stores" says more than a list of permissions |
| What they must be able to do            | The actions, named as they appear in the console                         |
| What they must <b>not</b> be able to do | Often the more important half, and the part that's usually left out      |
| The closest existing role               | And precisely how it falls short                                         |
| How many people will hold it            | One person is usually an exception; several is usually a role            |
| Whether it's permanent                  | A seasonal need is better served by exceptions with expiry dates         |

# What happens next

The request is analysed against the existing catalogue — partly to build the right role, and partly to check it doesn't overlap an existing one in a way that would make the model harder to reason about.

If a new role is created, it appears in the Role catalogue and you assign it like any other.

# While you wait

You don't have to leave someone blocked. Assign the closest role and cover the gap with exceptions, ideally with an expiry date matching your expected timeline. When the new role arrives, assign it and remove the exceptions.