

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.