Rivlet user manual

Getting started

On the floor

Running a station

Shipping

Planning

Buying

Quality

Engineering

Managers and the CFO

Administration

Selling through Shopify

Security in plain words

Troubleshooting

Glossary

Getting started

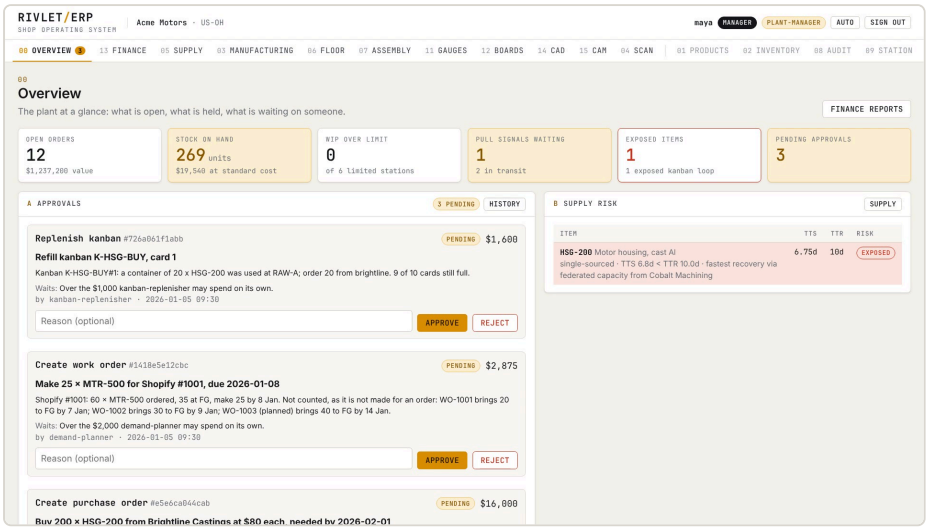
This chapter is for everyone. It says what Rivlet is, how its parts fit together, and how to sign in and find your way.

What Rivlet is

Rivlet is a shop operating system for a manufacturer. It keeps one record of the plant: items, stock, orders, work, people, machines and partners. Everyone works from that one record, each with the tools for their job.

- **The office** uses the web shell in a browser: planning, buying, quality, approvals and administration.
- **The floor** uses a Station tablet at each work station. You sign in with your badge and PIN.
- **Each station** can have a TV board that shows the day's work to everyone on the line.
- **Engineers** draw parts in CAD and turn them into machine programs in CAM.
- **Machines and instruments** connect through an edge bridge.
- **Partner factories** exchange signed quotes and orders with you.
- **A web shop** on Shopify records its orders in Rivlet, and Rivlet ships them.

Agents (small programs that watch the plant) can suggest or do routine work, such as refilling a kanban card. They never act alone on anything big: the plant's rules decide what they may do, and a person approves the rest.



Movie: Rivlet in 60 seconds (51 s). Maya's overview, the live floor, supply risk and partner capacity, a partner's signed quote checked against the ceiling kept at home, then a purchase approved with a reason. `shell-promo-60s`

Chapters: Live floor 0:05 · Supply risk 0:15 · Approvals 0:28

The plant in this manual

The pictures and movies come from a demo plant, Acme Motors in Ohio. It makes electric motors (MTR-500) and brackets. Its clock stands at Monday 5 January 2026, 09:30 plant time, three and a half hours into the Day shift.

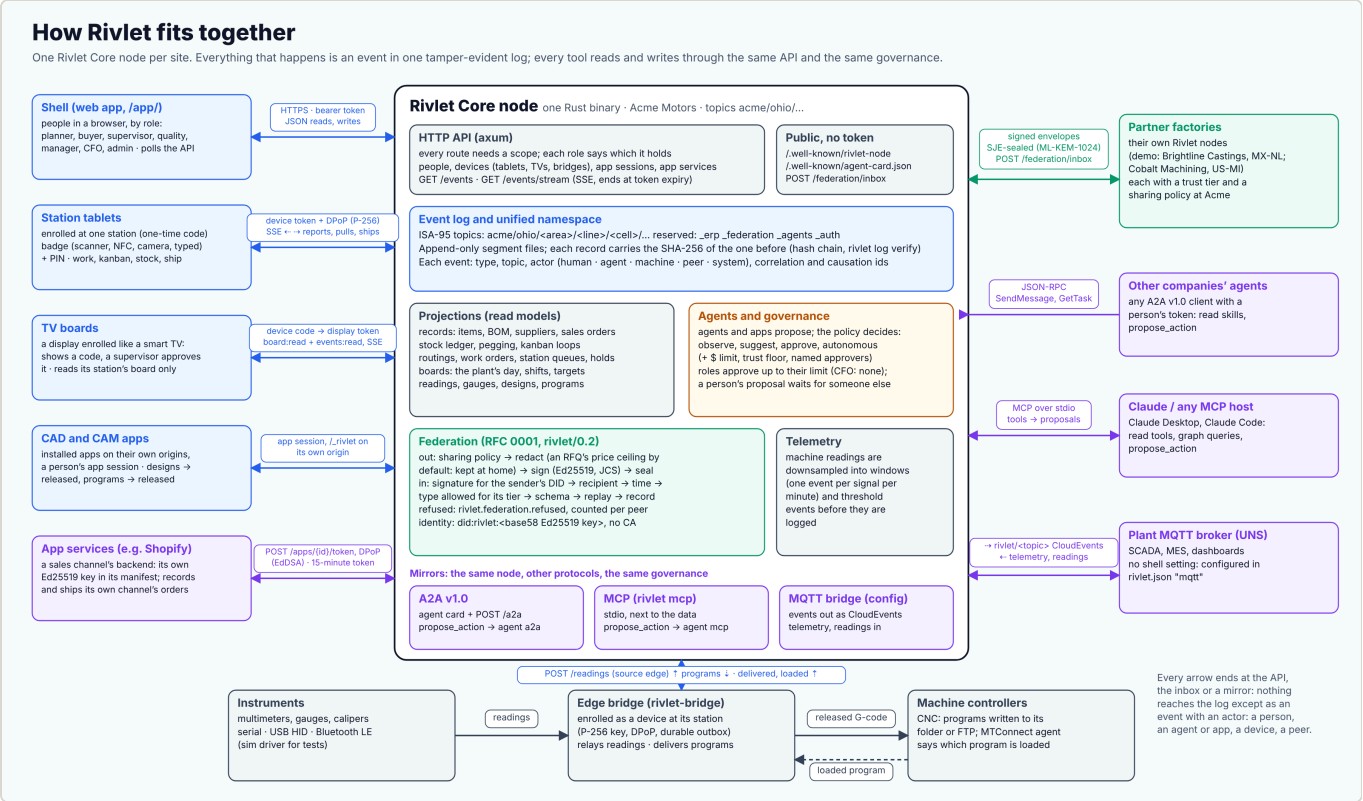
You will meet these people:

Person	Role	Approves up to	Starts on
Olivia	Floor operator, at ASM-1 only	nothing	09 Station
Sam	Line supervisor, at every station	$5,000	06 Floor
Pat	Planner	$10,000	03 Manufacturing
Bea	Buyer	$25,000	05 Supply
Quinn	Quality	nothing (signs off held units)	07 Assembly
Eli	Industrial engineer	nothing	06 Floor, in the designer
Maya	Plant manager	$250,000	00 Overview
Finn	CFO / finance	no ceiling	00 Overview
Ada	Admin	nothing (administers only)	10 Admin

The stations are ASM-1 (press), ASM-2 (wiring), TEST (end-of-line test), WIND-1 (winder), CNC-1 and CNC-2 (machining), SM-1 (the supermarket, where line stock is kept) and FG (finished goods, where orders ship from).

How the tools fit together

One Rivlet Core node runs at each site. Everything that happens there becomes an event in one log: an operator's report, a kanban pull, an agent's proposal, a partner's quote, a web order, a meter reading. Every tool reads and writes through the same node, under the same rules.

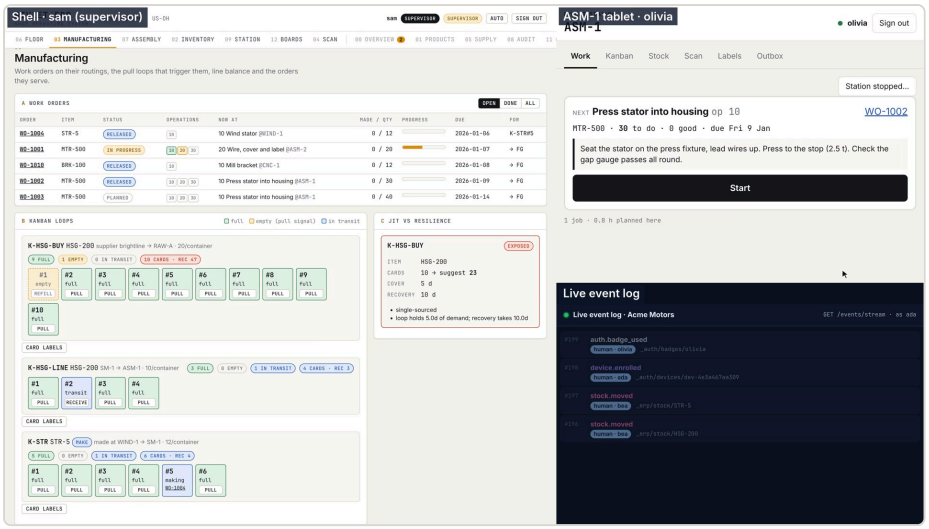


How Rivlet fits together. The Rivlet Core node at the centre: the shell, Station tablets, TV boards, CAD and CAM, app services such as a sales channel, and the edge bridge reach it through the API; partner factories through signed envelopes; other agents through A2A, MCP and MQTT mirrors. `network-architecture-overview-all-1800`

Part	Who uses it	What it is for
Shell (web app)	Office staff, managers, admins	Your sections by role: planning, buying, quality, approvals, admin. Open the node's address and add `/app/`.
Station tablet	Operators, station managers, supervisors	One tablet per station. Anyone can read it; you sign in with a badge and PIN to work.
TV board	Everyone on the line	One station's day: good today against the target, what is running, what is next.
CAD and CAM	Engineers, plant managers	Draw a part, make a machine program, release it to the machine.
Edge bridge	Set up by an admin	A small computer next to an instrument or machine. It sends readings and delivers programs.
Partner factories	Buyers, through agents	Other companies' Rivlet nodes. Quotes and orders travel as signed messages.
Sales channel apps	The web shop	The Shopify app records web orders and is told when they ship.

Every event names who did it: a person, an agent, an app, a device such as a bridge, or a partner. So you can always see who did what, and when.

To see one action reach every tool, watch this movie. Olivia reports a unit and pulls a kanban card at her tablet. The log shows her events, then the kanban agent's answer. Sam's shell shows the card in transit.



Movie: One report, every tool (37 s). Olivia reports a unit and pulls a kanban card on the ASM-1 tablet; the event log shows each event with its actor (a person, then an agent answering), and the supervisor's shell shows the card in transit and the work done. `network-one-event-everywhere`

Chapters: A report on the tablet 0:00 · A kanban pull, answered by an agent 0:13 · The shell catches up 0:26

Roles and personas

Two things decide what you see in the shell:

- **Your role** decides what you may do: which records you may change, whether you may approve, and up to what value. Your administrator sets it.
- **Your persona** decides where you start and which sections come first. It does not give you any rights.

The plant starts with these roles. Your administrator may change them or add others (see Administration).

Role	For	Approves up to
Floor operator	Work at your own stations: report, kanban, stock moves, readings, shipping, loading programs	nothing
Station manager	A floor operator's work, plus approvals, TV boards and daily targets at your stations	$1,000
Line supervisor	The same across the line, plus more reading	$5,000
Planner	Work orders, kanban loops, routings, targets, receiving goods	$10,000
Buyer	Purchase orders, suppliers, partners, receiving goods	$25,000
Quality	Holds and sign-off, readings, gauges	nothing
Industrial engineer	Items, routings, floor layouts, designs, programs	nothing
Plant manager	Releases designs, programs and floor layouts; approves	$250,000
CFO / finance	Reads everything, prices included; approves	no ceiling
Admin	People, roles, devices, apps, partners and rules	nothing: does no work

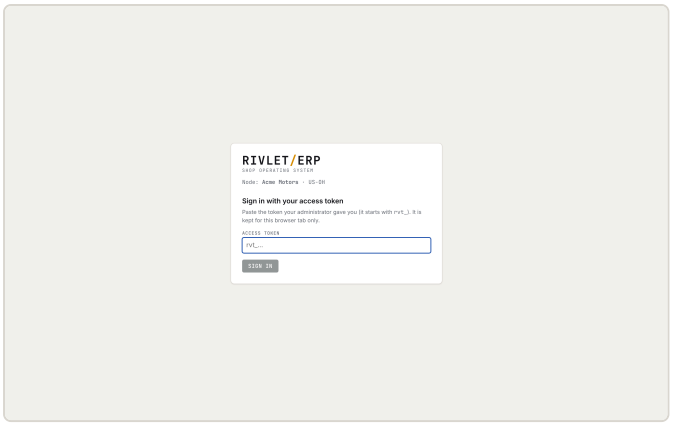
Prices, costs and values are shown only to roles that read finance (buyer, plant manager, CFO and admin). Everyone else sees "hidden" where a price would be.

Signing in to the shell

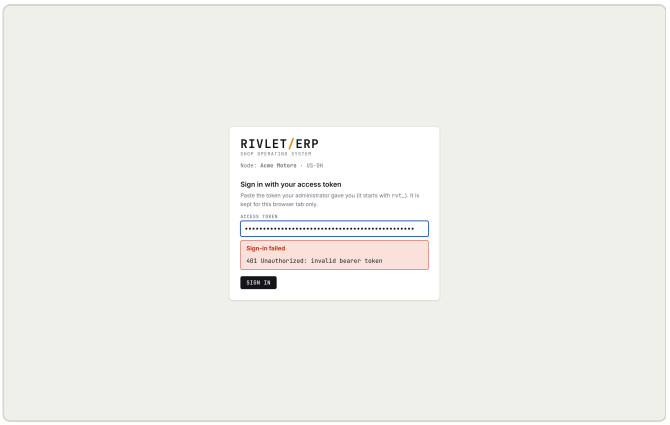
You need an access token. Your administrator gives it to you once. It starts with `rvt_`. Keep it private: anyone with it can act as you.

To sign in:

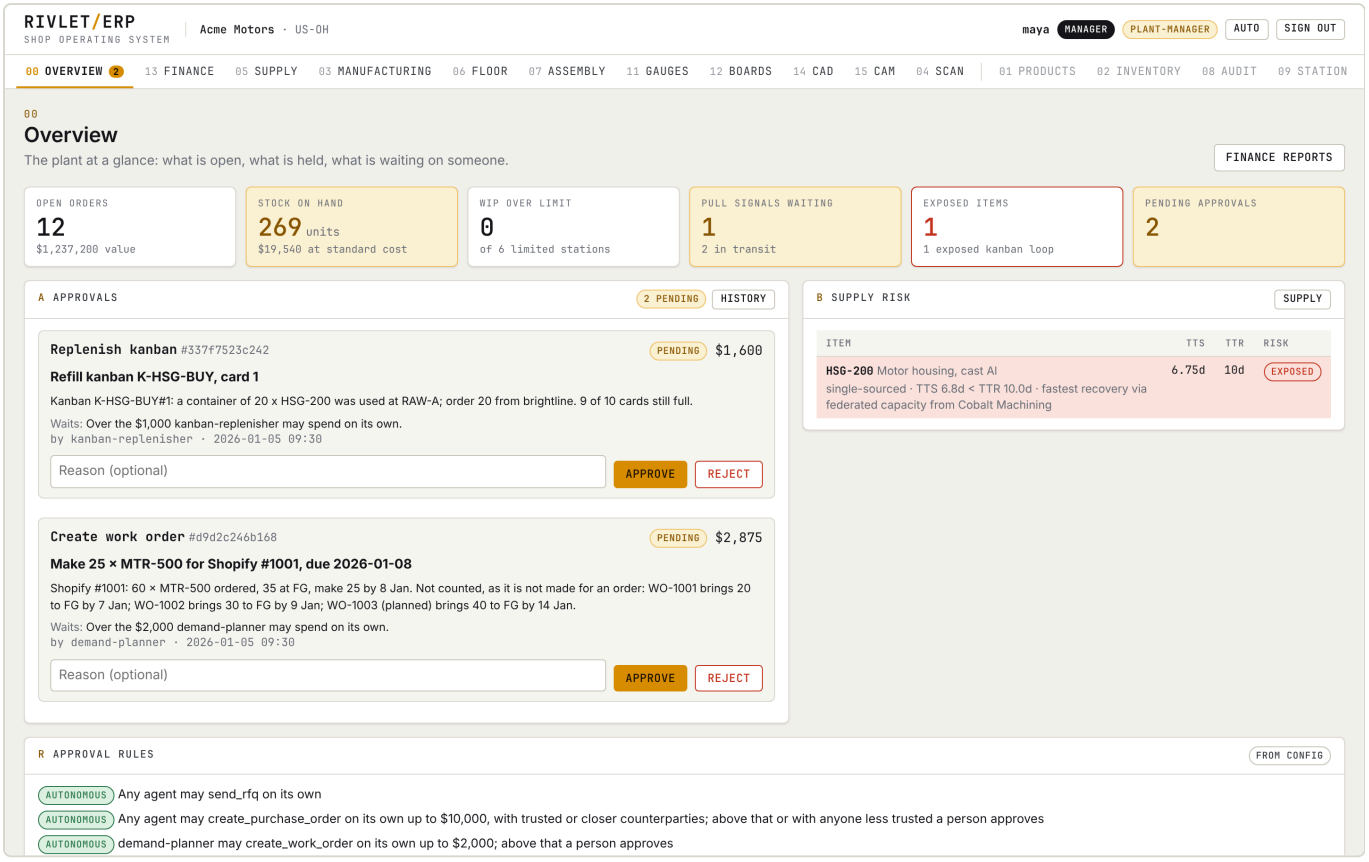
1. Open the node's address in your browser. The sign-in card shows the node's name, for example Acme Motors · US-OH.
2. Paste your access token into the field. It stays hidden as you paste it.
3. Press **Sign in**. You land on your first section.



Sign in. The sign-in screen: paste the access token your administrator gave you. `shell-signin-screen-none-desktop`

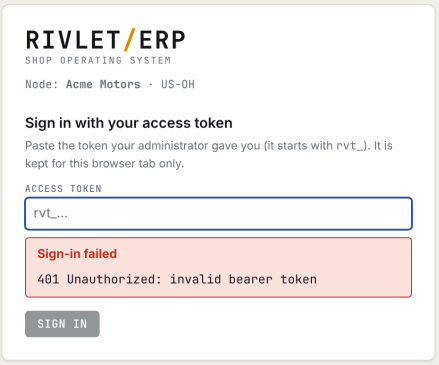


Token pasted. The token is hidden as you paste it; press Sign in. `shell-signin-filled-maya-desktop`

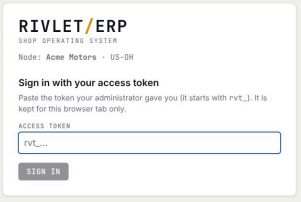


Signed in. Signed in, Maya lands on her first section; her name, persona and role are at the top right. `shell-signin-landed-maya-desktop`

If the node does not know your token, it is refused under the field. Check that you copied all of it, or ask your administrator for a new one.



Sign-in refused. A token the node does not know is refused with an error under the field. `shell-signin-invalid-token-none-desktop`



Movie: Getting started: sign in and find your way (1 min 37 s). Sign in with a token, read the top bar, use your own sections, switch theme and sign out; then the same shell as the plant manager, and a wrong token. `shell-getting-started`

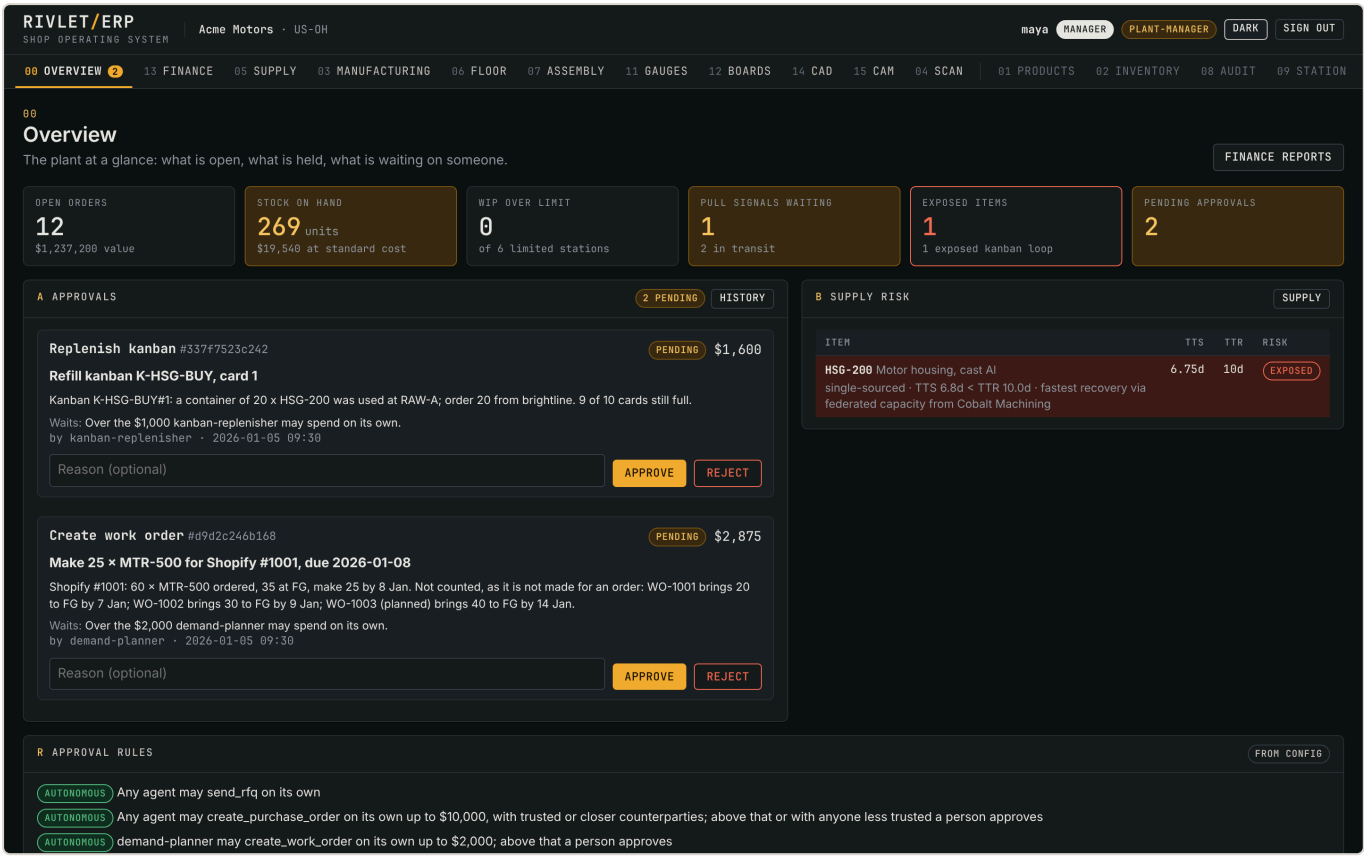
Chapters: Sign in 0:00 · The top bar 0:13 · Your sections 0:22 · Theme and sign out 0:48 · Another persona 1:00 · A wrong token 1:26

Finding your way

The top bar

On the right of the top bar you see your name, your persona (dark pill) and your role (outlined pill), then two buttons:

- **Theme** cycles through Auto, Dark and Light.
- **Sign out** forgets your token in this browser tab.



Dark theme. The theme button cycles Auto, Dark and Light; this is the Overview in dark. `shell-overview-dark-maya-desktop`

The sections

Under the top bar is the section strip. The sections are numbered:

Section	What it is for
00 Overview	The plant at a glance: open orders, stock, what waits for a decision, supply risk, the live floor
01 Products	Items the plant makes, buys and sells, with their bills of material and routings
02 Inventory	Stock by location, moving stock, and the stock ledger
03 Manufacturing	Work orders, kanban loops, line balance and open orders
04 Scan	Scan or type any code (serial, item, kanban card, work order or location) to open it
05 Supply	Purchase orders, approvals, supply risk, suppliers, partner factories, RFQs and quotes
06 Floor	The live floor, the layout designer and its revisions
07 Assembly	Builds, each unit's history (genealogy) and quality holds
08 Audit	The event log: who did what, when and why
09 Station	The Station app, inside the shell
10 Admin	People, roles, partners, rules, apps, devices, badges and boards (admins only)
11 Gauges	Instruments and their calibration
12 Boards	TV boards and each station's daily target
13 Finance	Inventory valuation, work in progress, purchase commitments and what shipped this month, exported as CSV (roles that read finance)
14 CAD, 15 CAM	The design apps, when they are installed

Your own sections come first, in dark text. After a thin divider come the other sections your role may read, in grey. A section your role cannot use is not shown.



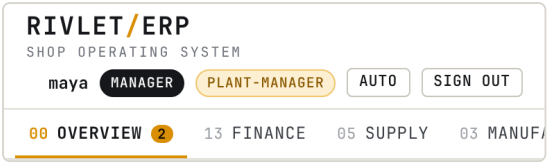
Olivia's navigation. Olivia (floor operator) sees their own sections first, then the rest of the shell after a divider; signed in, they land on /station. `shell-nav-bar-olivia-desktop`



Maya's navigation. Maya (plant manager) sees their own sections first, then the rest of the shell after a divider; signed in, they land on /overview. `shell-nav-bar-maya-desktop`

A number on a section counts what waits there for your decision. Only people who may decide see it.

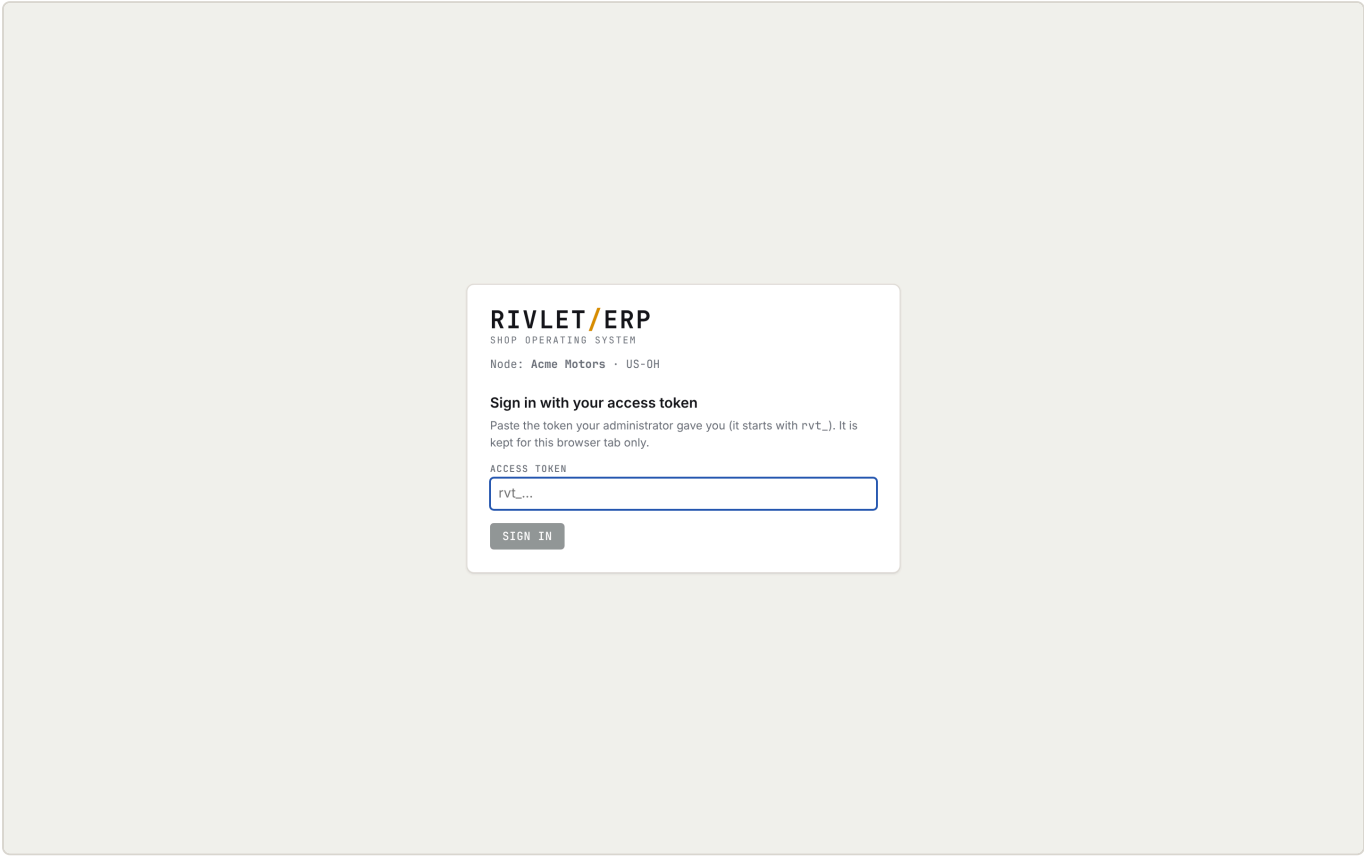
On a phone the strip scrolls sideways.



Maya's navigation (phone). The top bar and section strip at phone width for Maya (plant manager). `shell-nav-bar-maya-phone`

Signing out

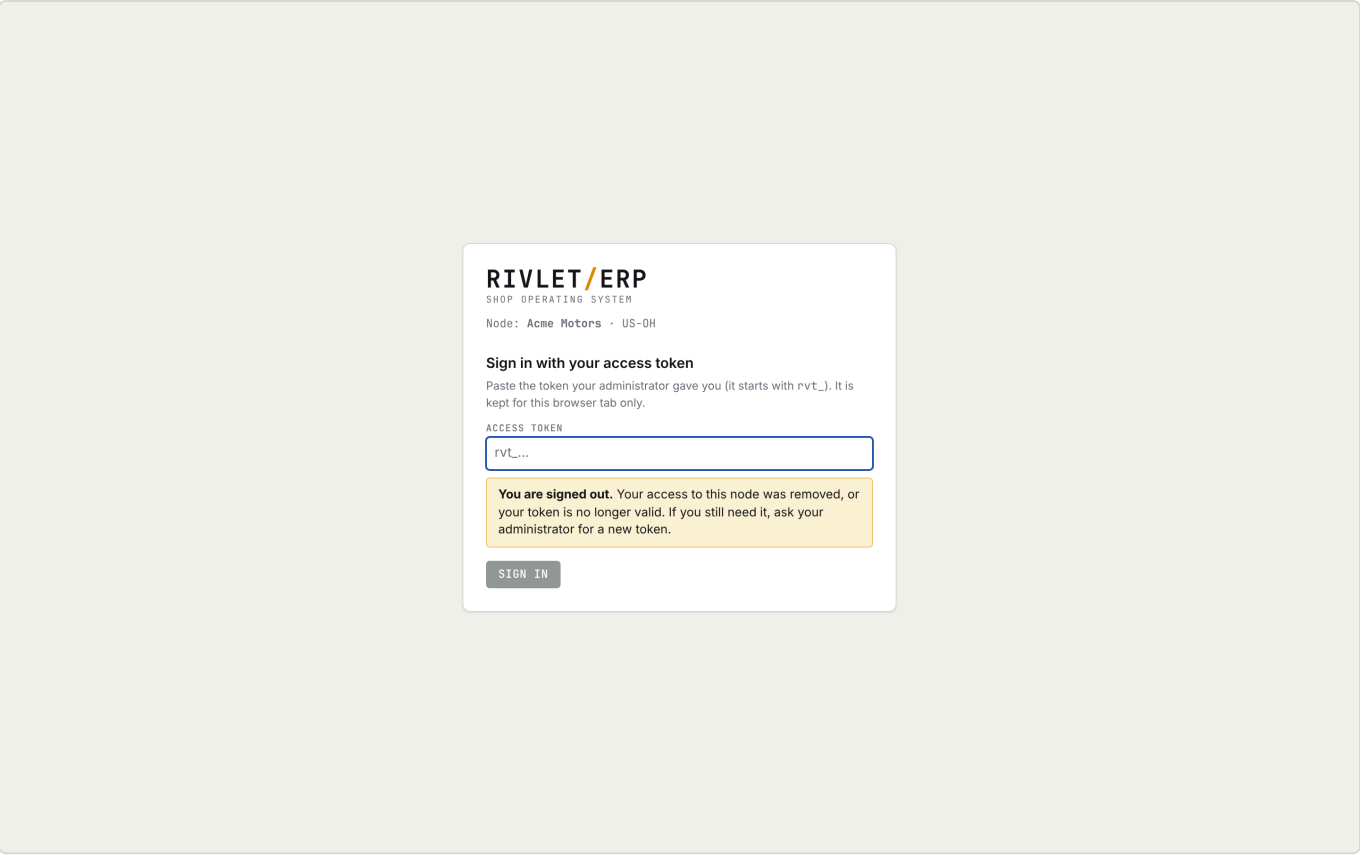
To sign out, press **Sign out** in the top bar. The shell goes back to the sign-in card and forgets your token in this tab.



Signed out. After Sign out the shell is back at the sign-in screen; the token is forgotten. `shell-signin-signed-out-maya-desktop`

Sign out also returns the shell to its start, so on a shared computer the next person to sign in lands on their own first section, not on the page the last person left.

If an administrator removes your access, your open shell stops at your next step and says: "You are signed out. Your access to this node was removed, or your token is no longer valid. If you still need it, ask your administrator for a new token."



Access removed. When an administrator removes a person, their open shell stops at the next step and says their access was removed. `shell-signin-revoked-token-operator-desktop`

Times on the screen

The plant keeps its own time (Acme keeps Ohio time, America/New_York). The day, the shifts and "today" are the plant's.

Every screen tells times in the plant's time, 24-hour, whatever zone your computer, tablet or phone is set to: the TV boards (09:30), the Station tablet ("stopped since 09:30"), the shell (approvals, the audit log and record histories read "2026-01-05 09:30", and the live floor's "updated" time is the node's clock), and CAD and CAM (when a revision was released, a draft saved, a program loaded).

If two times on one screen look hours apart, they are probably the same moment in different zones.

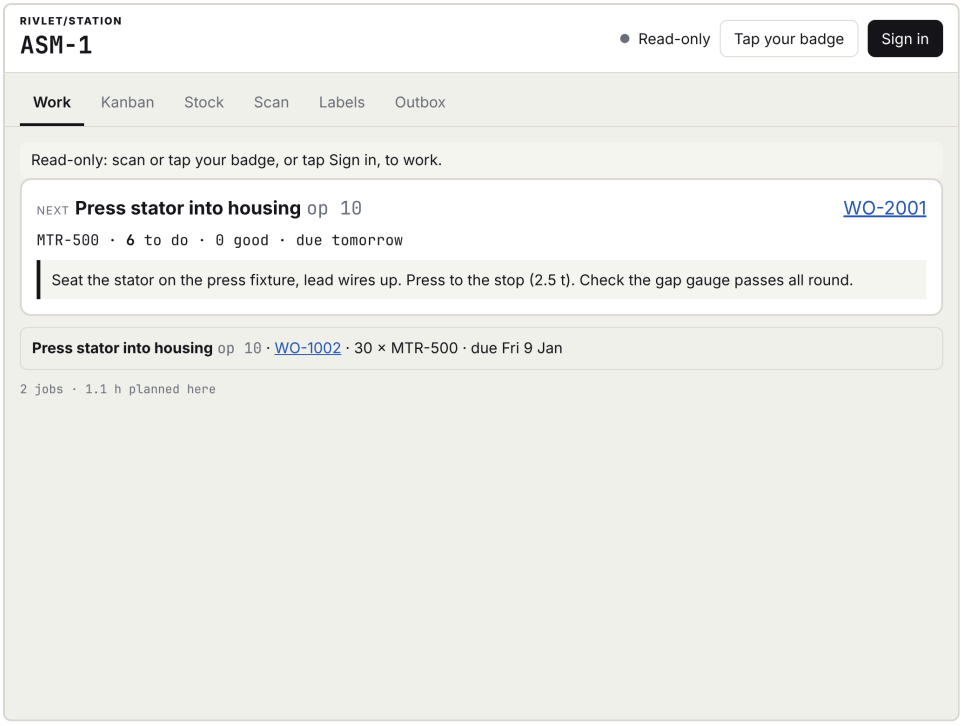
On the floor

This chapter is for floor operators. It covers the Station tablet at your work station: signing in, your queue, reporting work, scrap and stoppages, measuring, kanban, moving stock, scanning, and what happens when the network drops.

The Station tablet

Each work station has its own tablet. Once it is set up, the tablet *is* that station: it shows only that station's work.

- Anyone can read the tablet. Nobody can change anything until a person signs in with a badge and PIN.
- Everything you do is recorded as yours.
- Across the top are the tabs: **Work**, **Kanban**, **Stock**, **Scan**, **Labels** and **Outbox**. At FG (finished goods) a **Ship** tab comes first; see Shipping.
- When the tablet cannot do something, it says why in plain words, and what to do instead.



Read-only station. With nobody signed in, the tablet shows the station's queue read-only; a badge reader, an NFC tap or Sign in starts a sign-in. `floor-sign-readonly-nobody-tablet`

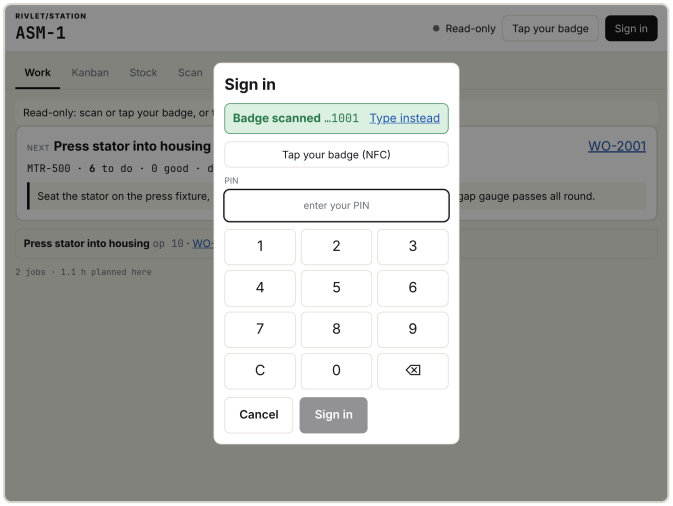
Signing in at a tablet

You need a badge and your PIN. A badge can be a printed card or an NFC tag. You may have more than one; any of them works with your one PIN. Your administrator gives you badges and sets your PIN.

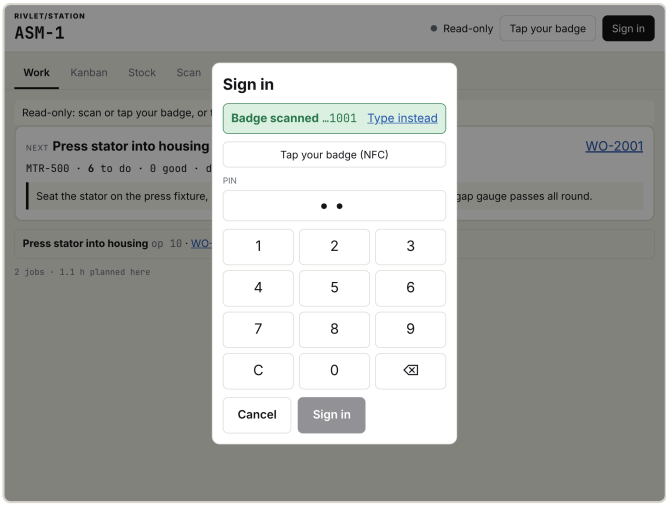
There are four ways to read your badge. Use whichever your tablet has.

With a badge scanner

1. Scan your badge with the reader. You do not need to tap anything first: it works on any tab.
2. The sign-in opens with "Badge scanned" and the last four characters of your badge.
3. Enter your PIN on the keypad. **C** clears it; ⌫ removes one digit.
4. Tap **Sign in**.



Sign in: badge scanned. A badge scanned on the read-only screen opens the sign-in with the badge read (only its last four characters shown); the PIN is next. `floor-sign-scanned-olivia-tablet`

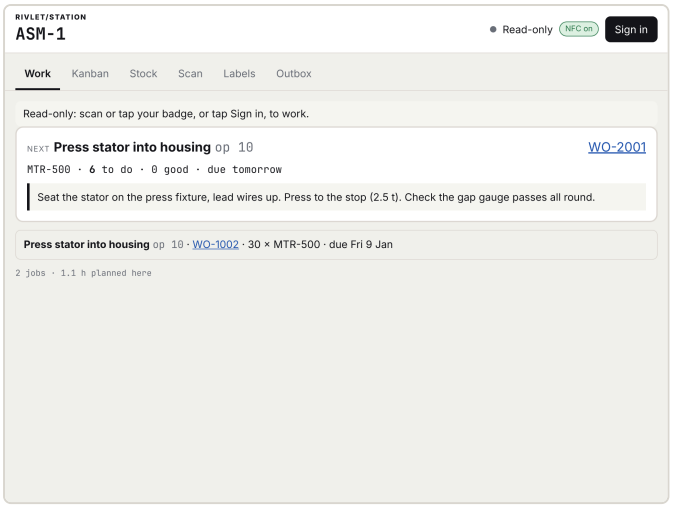


Sign in: PIN keypad. Enter your PIN on the keypad; C clears it and ⌫ removes one digit. `floor-sign-keypad-olivia-tablet`

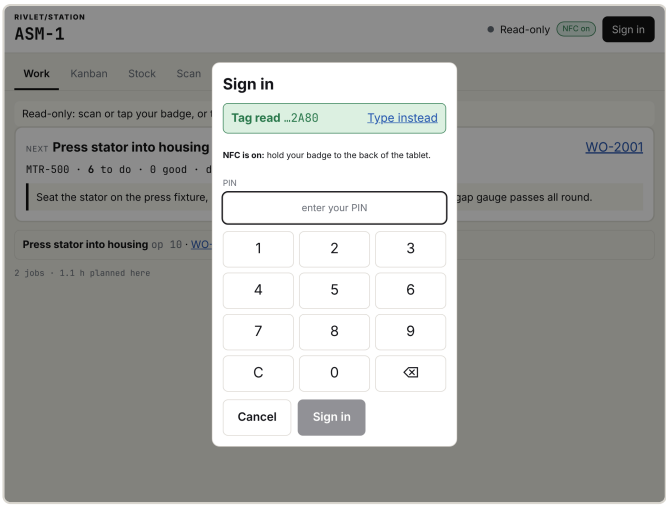
With an NFC tag

This works on Chrome on Android, or with a USB NFC reader.

1. Tap **Tap your badge** once. The reader stays on until someone signs in.
2. Hold your tag to the tablet. The sign-in says "Tag read".
3. Enter your PIN and tap **Sign in**.



NFC on. On a tablet with NFC (Chrome on Android), one tap on "Tap your badge" turns the reader on until someone signs in. `floor-sign-nfc-on-nobody-tablet`



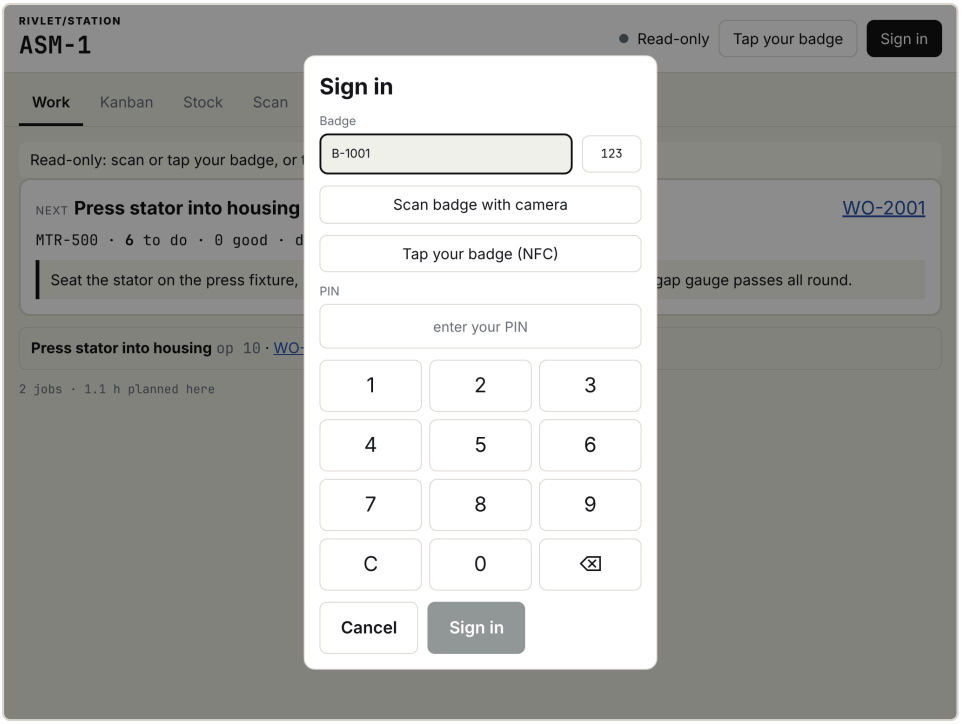
Sign in: tag read. Olivia holds her NFC tag to the tablet: "Tag read", then her PIN. `floor-sign-nfc-read-olivia-tablet`

With the camera

Some tablets and kiosks watch for a badge with their camera. A small preview in the corner says **CAMERA ON** and "Hold your badge up to the camera". Hold your badge up to it; the PIN pad opens. The camera reads the badge on the tablet itself: nothing is recorded or sent. A supervisor turns this on (see Running a station).

By typing

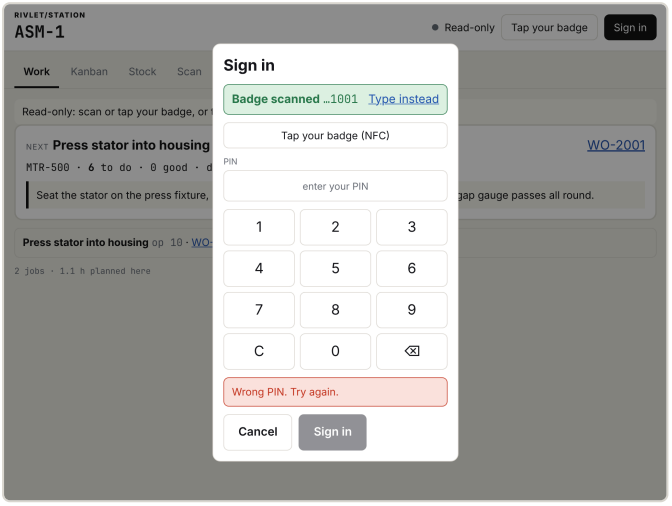
No reader? Tap **Sign in**, type your badge number, then your PIN.



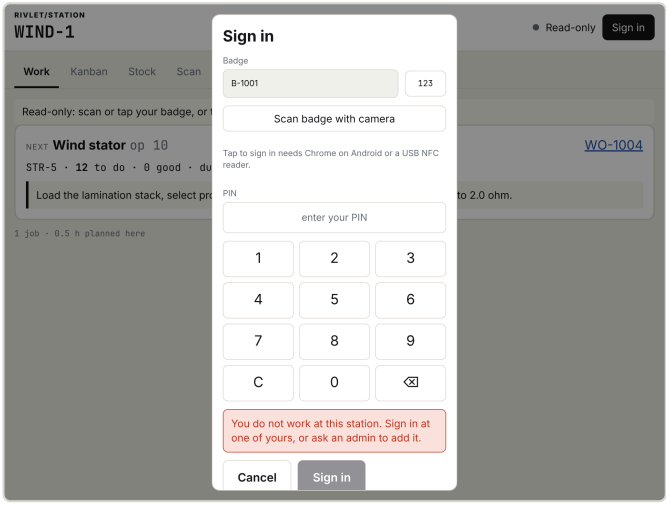
Sign in: badge typed. No reader? Tap Sign in and type the badge number; the cursor then moves to the PIN. `floor-sign-badge-olivia-tablet`

When sign-in is refused

- **"Wrong PIN. Try again."** The PIN is cleared; your badge stays read. Enter it again.
- **"Too many wrong PINs: your badges are locked for a few minutes."** Five wrong PINs lock all your badges for five minutes. Wait, or ask your supervisor.
- **"You do not work at this station. Sign in at one of yours, or ask an admin to add it."** You may only sign in at your own stations.
- **"This badge isn't registered. Ask your supervisor."**



Wrong PIN. A wrong PIN is refused and the PIN is cleared; the badge stays read. `floor-sign-wrong-pin-olivia-tablet`

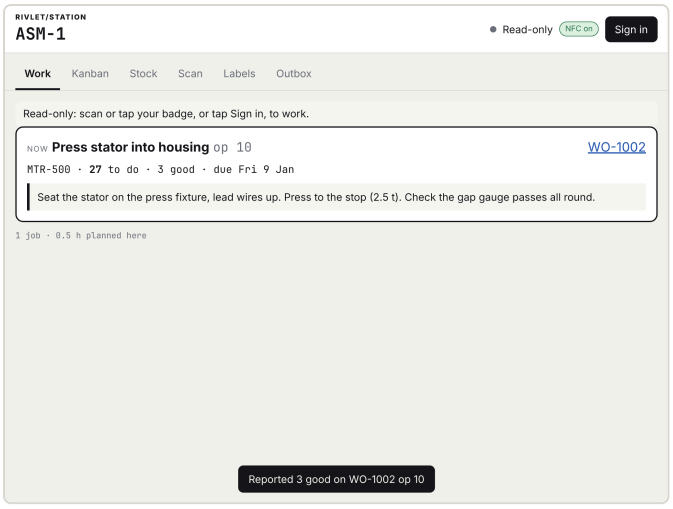


Badge refused at another station. Olivia works at ASM-1 only, so her badge is refused at the WIND-1 tablet. `floor-sign-refused-station-olivia-tablet`

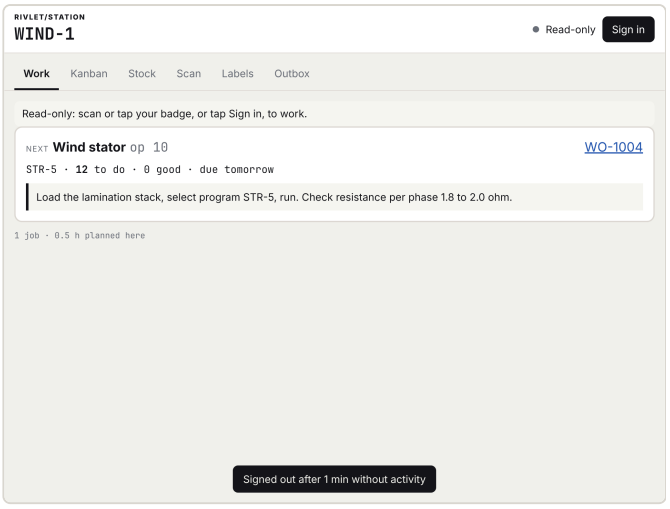
Signing out

Once signed in, your name shows at the top with a green dot. Reloading the page keeps you signed in; your PIN is never kept.

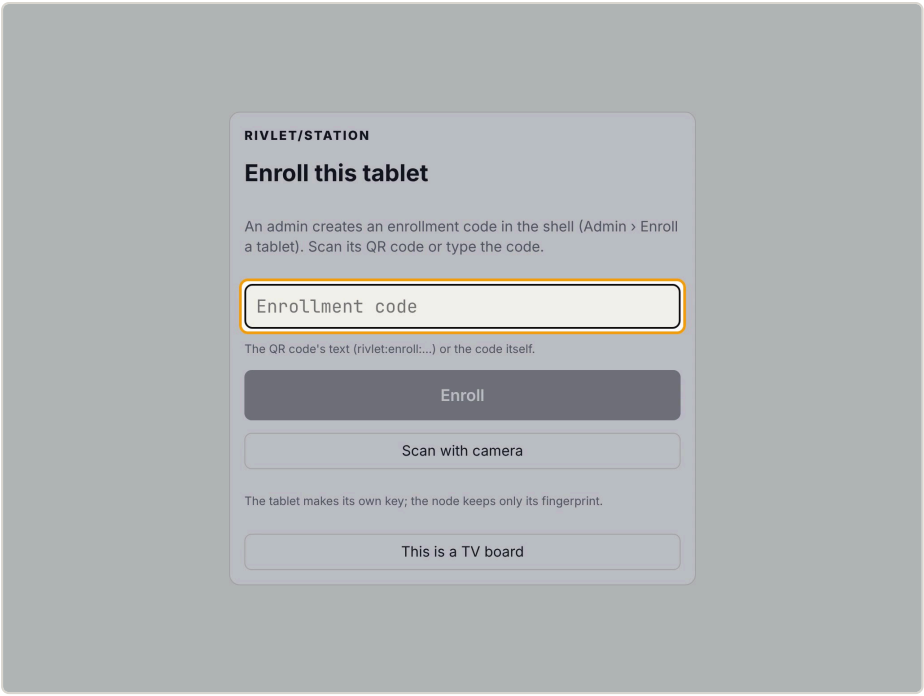
- Tap **Sign out** when you leave the station.
- After 10 minutes without a tap, the tablet signs you out by itself.
- Work you queued while offline stays yours. It is sent when the tablet is back online.



Signed out. After Sign out the tablet is read-only again until the next badge; olivia's refusal goes with her. `floor-sign-signed-out-nobody-tablet`



Signed out for inactivity. After the idle limit (10 minutes by default) without a tap, the tablet signs the person out by itself. `floor-sign-idle-signed-out-nobody-tablet`



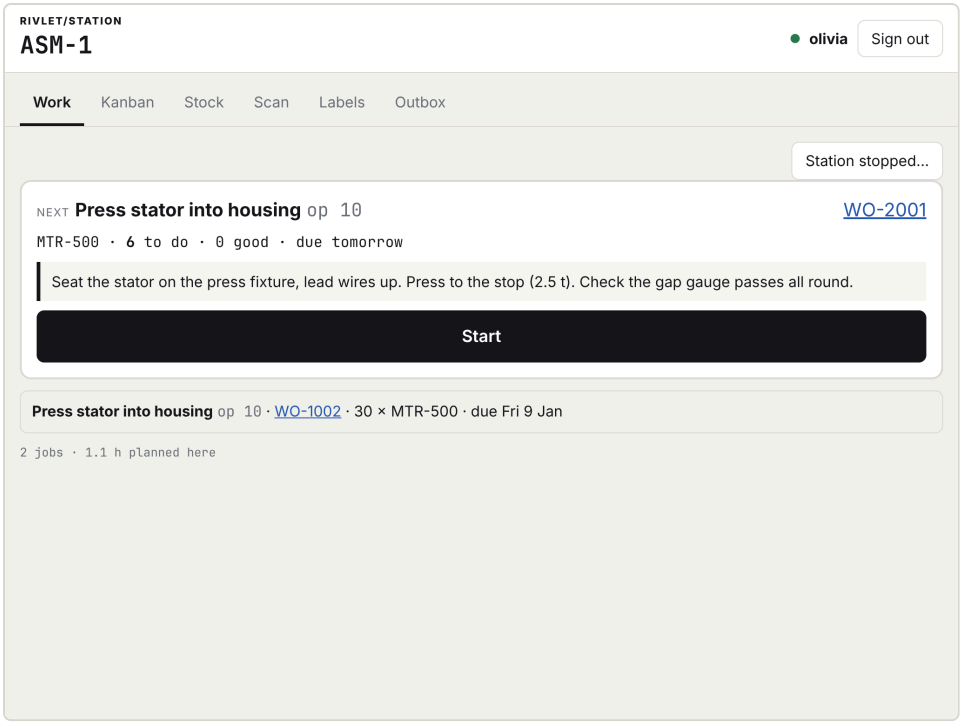
Movie: Enroll a tablet, sign in and out (1 min 35 s). Enroll a new tablet at ASM-1 with a code and open it read-only; sign in with a badge scanner (a wrong PIN first), sign out; sign in with an NFC tag; sign out; and by typing the badge. `floor-sign-in-and-out`

Chapters: Enroll the tablet 0:00 · Read-only until someone signs in 0:18 · Scan your badge 0:23 · Sign out 0:56 · Tap an NFC tag 0:59 · Or type it 1:16

Your queue

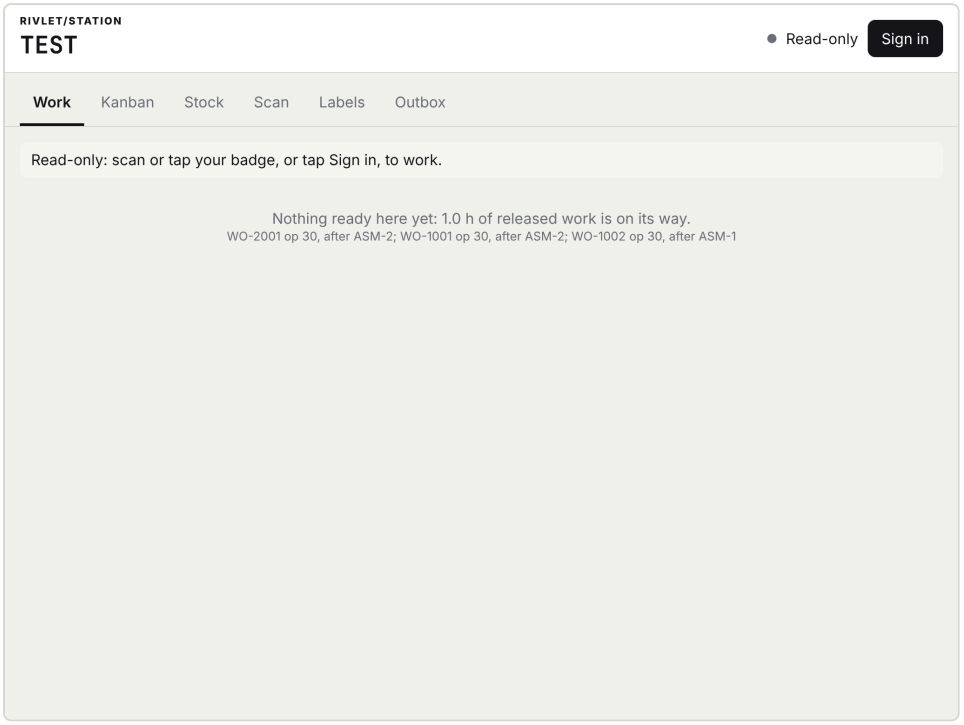
The **Work** tab is your queue. It lists the jobs whose current operation is at this station, by due date, with a running job first.

- The top card is the job to do now, with its instructions.
- Each job shows its work order, the operation, the item, how many are to do, how many are good so far and when it is due.
- A job made for a customer's order shows the order it is for, for example "Shopify #1001".
- Under the queue: how many jobs, and the hours planned here.



The work queue. The next job at the station on top, with its instructions; later jobs are listed under it. `floor-work-queue-olivia-tablet`

When nothing is ready, the tablet says so and what is on its way: "Nothing ready here yet: 1.0 h of released work is on its way."



Nothing to do here yet. A station with nothing ready says so, and what is on its way. `floor-work-empty-queue-nobody-tablet`

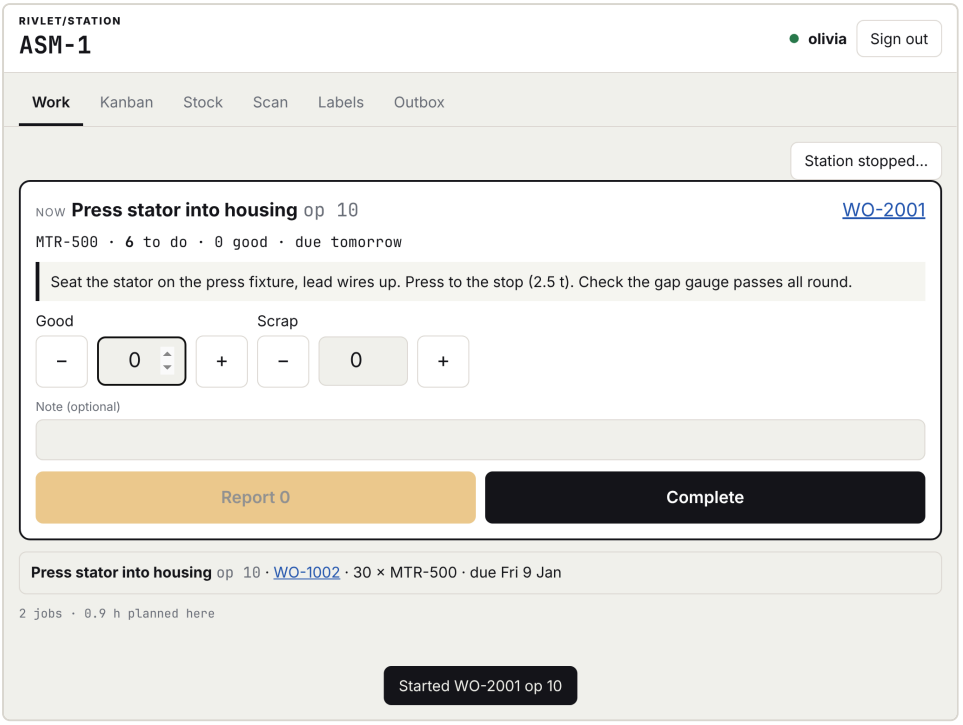
When you complete an operation, the job moves on to the next station's queue at once. Nobody carries a paper traveller. See How work travels between stations.

Start, report, scrap and stoppages

To start a job

1. Sign in.
2. On the **Work** tab, tap **Start** on the top card.

A short note confirms it once the node has recorded it, saying what happened, for example "Started WO-2001 op 10". The job now shows **Good** and **Scrap** counters, **Note (optional)**, **Report** and **Complete**.

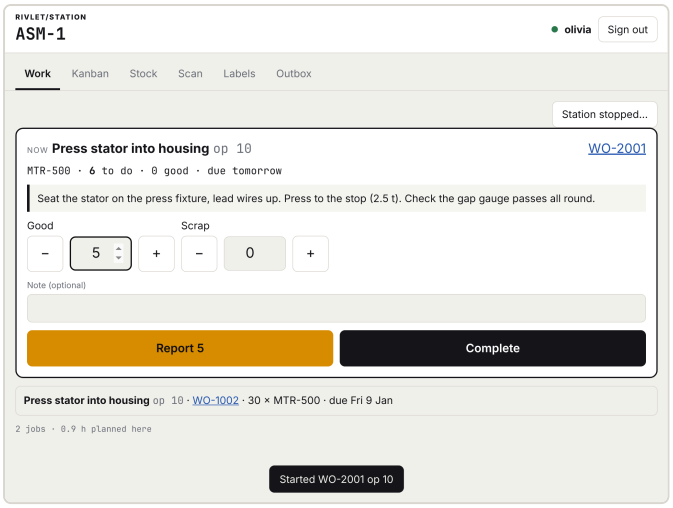


Job started. Once started, the job shows Good and Scrap counters, a note, Report, Stop and Complete. `floor-work-started-olivia-tablet`

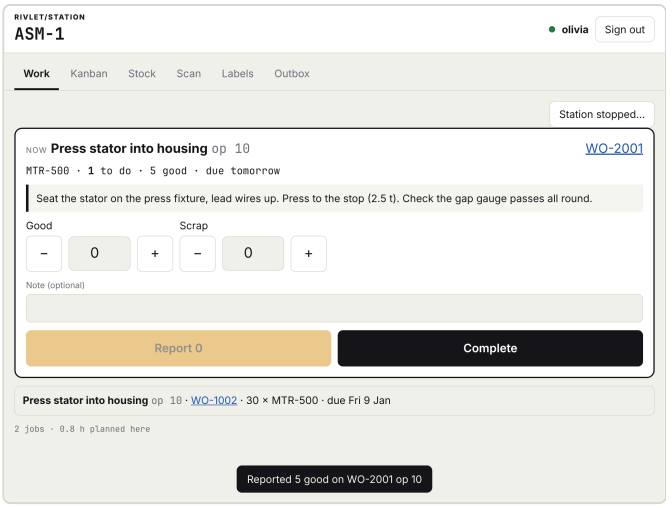
To report good units

1. Set **Good** with **+** and **−**, or type the number.
2. Add a note if it helps.
3. Tap **Report N** (for example **Report 6**).

One tap is one report. The report counts once the node has it; then the counters go back to 0.



Report good. Set how many came out good, then tap Report: one tap, one report. `floor-work-report-filled-olivia-tablet`

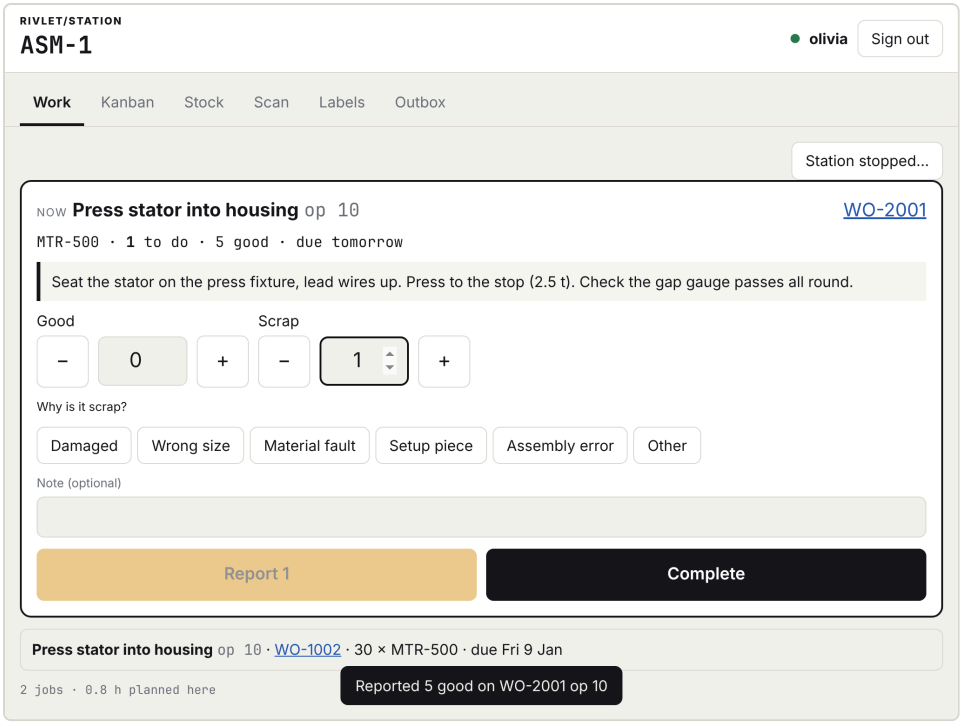


Reported. The report is confirmed once the node has it; the counts update and the counters start again from 0. `floor-work-reported-olivia-tablet`

To report scrap

1. Set **Scrap** above 0.
2. Under "Why is it scrap?", pick a reason: **Damaged**, **Wrong size**, **Material fault**, **Setup piece**, **Assembly error** or **Other**.
3. Add a note if it helps.
4. Tap **Report N**. Report waits until you pick a reason.

A unit scrapped at the first operation of an order is made again there, so the order does not end up short.



Scrap says why. A unit reported as scrap needs a reason: pick one (Damaged, Wrong part, ...) and add a note if it helps. `floor-work-scrap-reason-olivia-tablet`

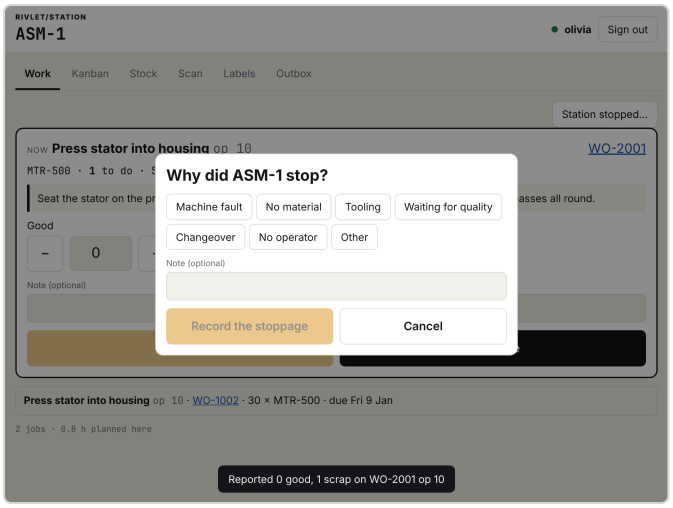
When the station stops

To record a stoppage:

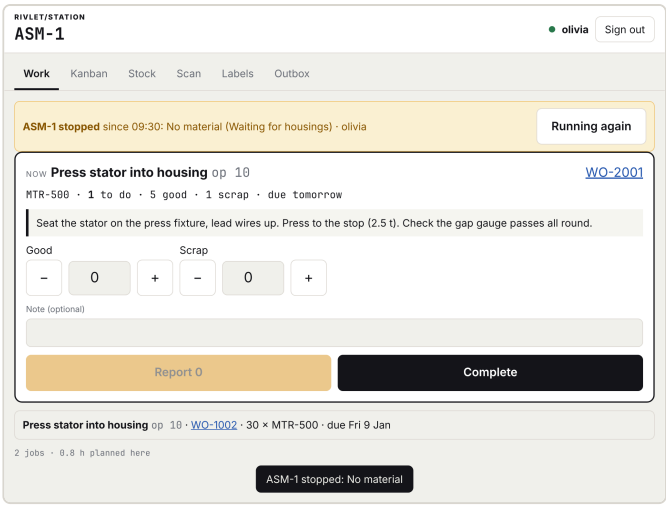
1. Tap **Station stopped...** (top right of the Work tab).
2. Pick why: **Machine fault**, **No material**, **Tooling**, **Waiting for quality**, **Changeover**, **No operator** or **Other**.
3. Add a note if it helps, for example "Waiting for housings".
4. Tap **Record the stoppage**.

The tablet shows "Station stopped", the reason, who stopped it and since when. The shell and the station's TV board turn red with the reason.

To end it, tap **Running again**. Rivlet records how long the station was down.



Station stopped: why?. Stop asks why the station is down (no material, breakdown, waiting for quality…), with an optional note; the boards and the shell show it. `floor-work-stop-dialog-olivia-tablet`



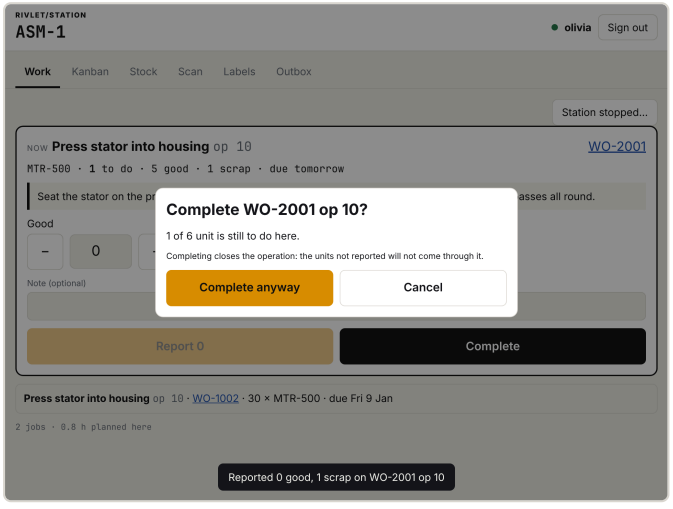
Station stopped. "Station stopped: No material", with who stopped it and since when; Running again ends it and records how long it was down. `floor-work-stopped-olivia-tablet`

The tablet tells times in the plant's time, as the TV board beside it does ("stopped since 09:30").

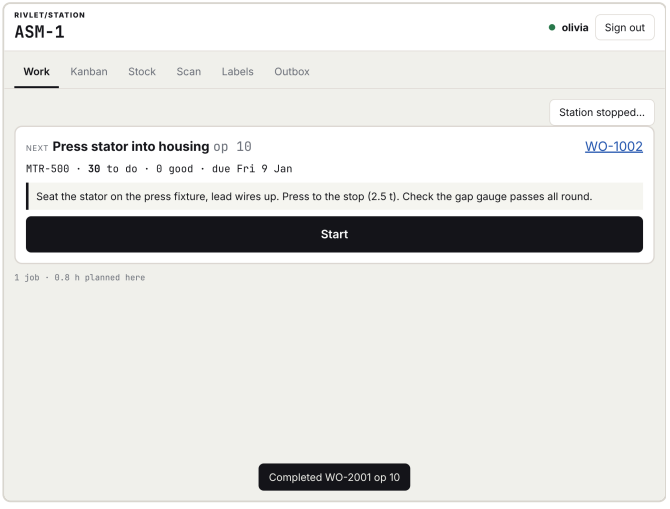
To complete an operation

1. Report the last units.
2. Tap **Complete**.

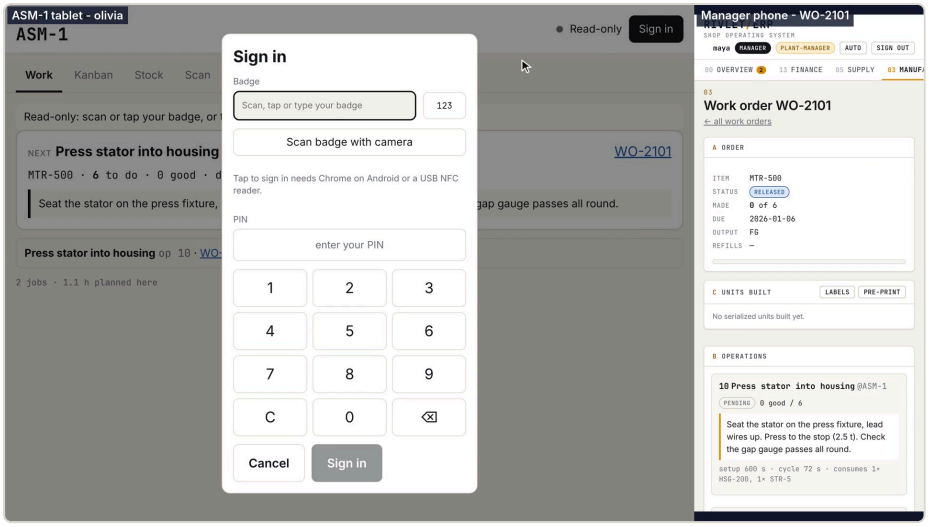
The job leaves your queue and goes to the next station. If units are still to do, Complete asks first: **Complete anyway**, or **Cancel** to keep the operation running.



Complete asks first. Complete with units still to do asks first; Cancel leaves the operation running. `floor-work-confirm-complete-olivia-tablet`



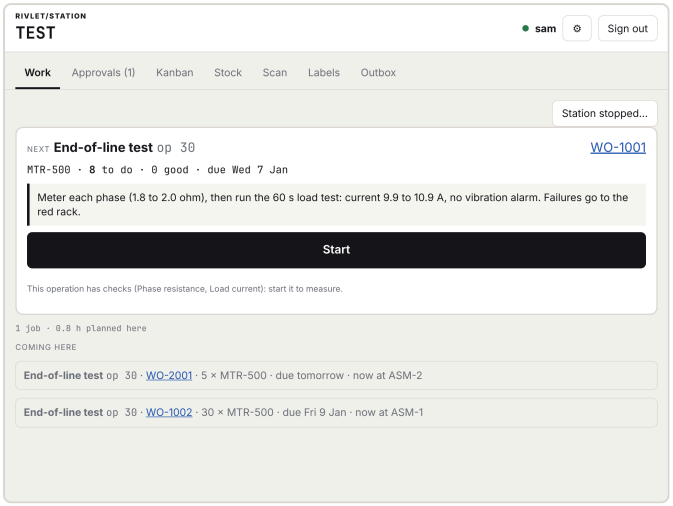
Operation complete. The completed job leaves the station; the next one moves to the top. `floor-work-completed-olivia-tablet`



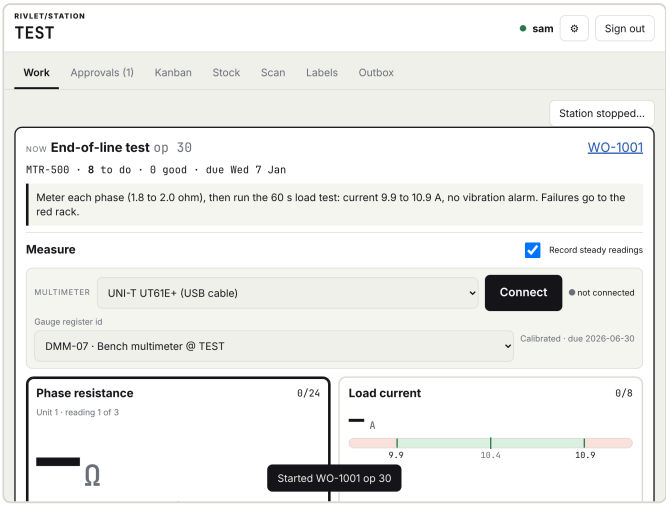
Movie: A shift at ASM-1 (38 s). Olivia badges in, starts the next job, reports six good, pulls a kanban card and completes the job, while the work order updates on the manager's phone. `floor-promo-shift`

Measuring

Some operations have checks: a measurement each unit must pass, such as a phase resistance or a torque. A job with checks lists them. When you start it, the **Measure** panel opens on the Work tab.



A job with checks. A job with checks lists them; start it to open the Measure panel. `floor-measure-next-sam-tablet`



The Measure panel. The Measure panel: the instrument and its calibration, each check's spec band and progress, and the units measured. `floor-measure-panel-sam-tablet`

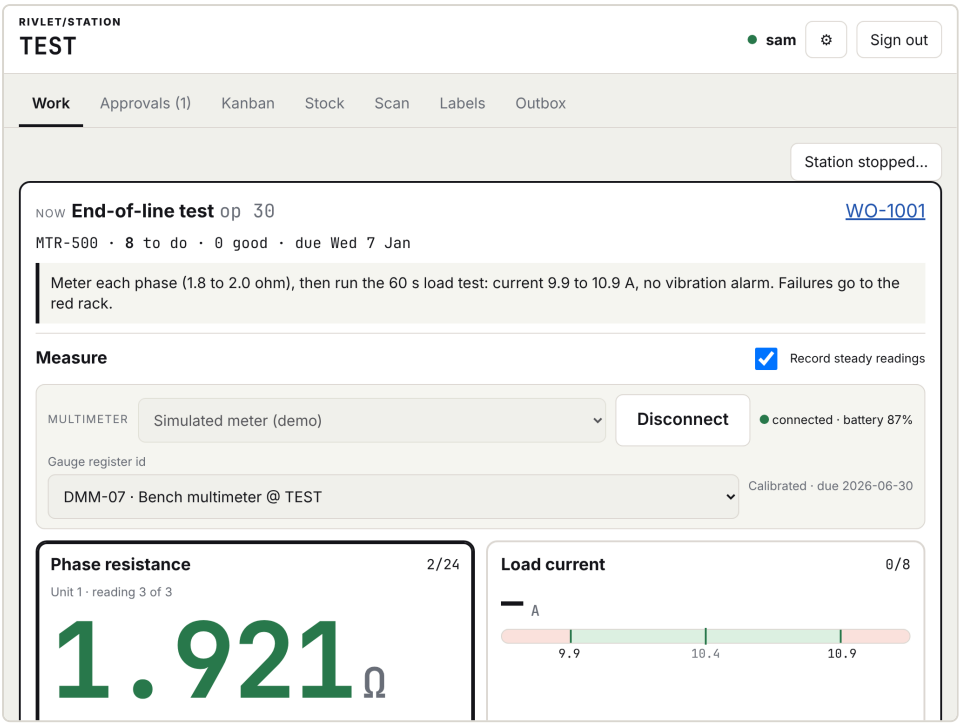
The panel shows the instrument and its calibration, each check's spec band and progress, and the units measured.

To measure a unit

1. Tap **Connect** and pick your instrument (a Bluetooth or USB meter, or one that types its value like a keyboard). The tablet remembers it for next time.

2. Measure. A value that holds steady inside the band is recorded by itself, and the panel moves to the next reading.
3. Lift the probes between readings, so the same value is not taken twice.
4. **Retake** drops the last reading of a check. **Next** skips to the next check.
5. When the unit is measured, report it as usual. Good units with in-spec readings are proposed as good.

You can also type a value. Readings from an instrument past its calibration date are refused: pick a calibrated one.

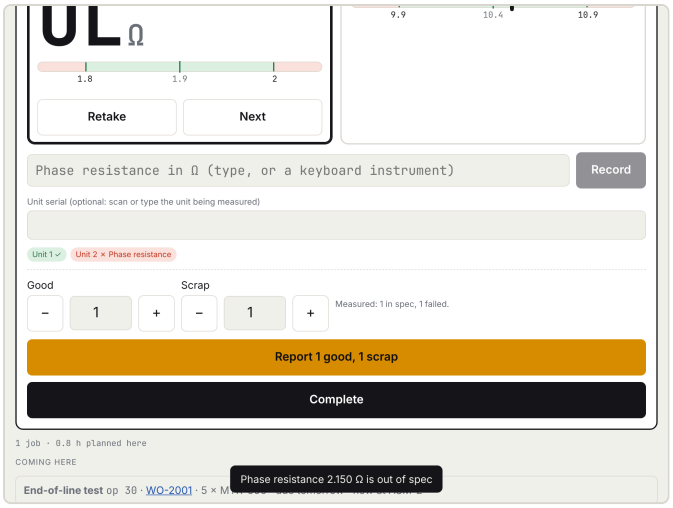


A live reading. The meter's live value against the spec band; a steady in-spec value is recorded by itself. `floor-measure-live-sam-tablet`

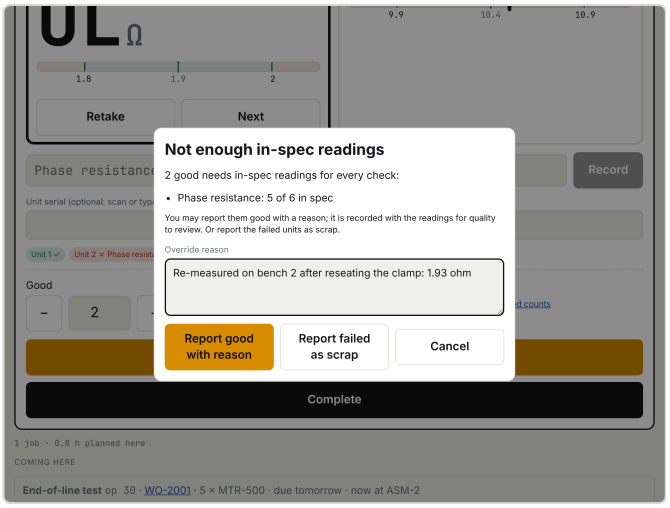
When a unit fails

A reading outside the band marks the unit failed, in red. What happens next depends on the check:

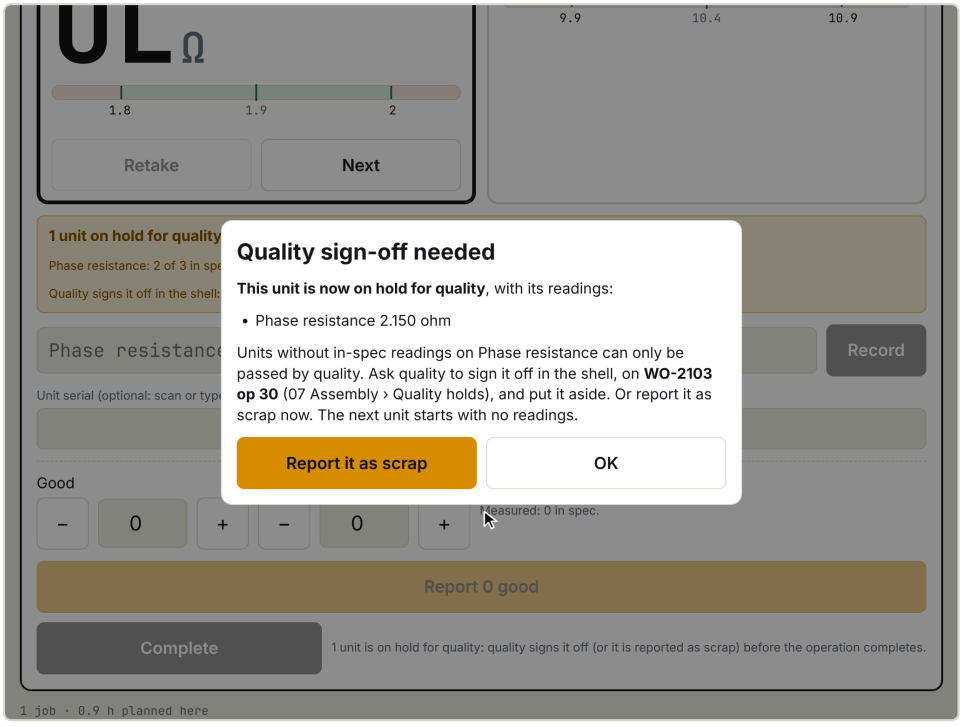
The check says	What you can do
Operator decides	Report it good with a reason (kept with its readings for quality), or report it as scrap.
Quality decides	You cannot pass it. Reporting it good puts the unit on hold for quality. Tap **OK** and put it on the red rack. Quality signs it off; see Quality.
Scrap	The unit is reported as scrap.



Out of spec. A reading outside the band marks the unit failed in red; the proposed report counts it as scrap. `floor-measure-out-of-spec-sam-tablet`



Report a failed unit good. Reporting a failed unit good asks for a reason, kept with the readings for quality; or report it as scrap. `floor-measure-override-sam-tablet`



On hold for quality. A unit failing a "Quality decides" check goes on hold for quality with its readings; the dialog says where quality signs it off. `floor-quality-signoff-dialog-olivia-tablet`

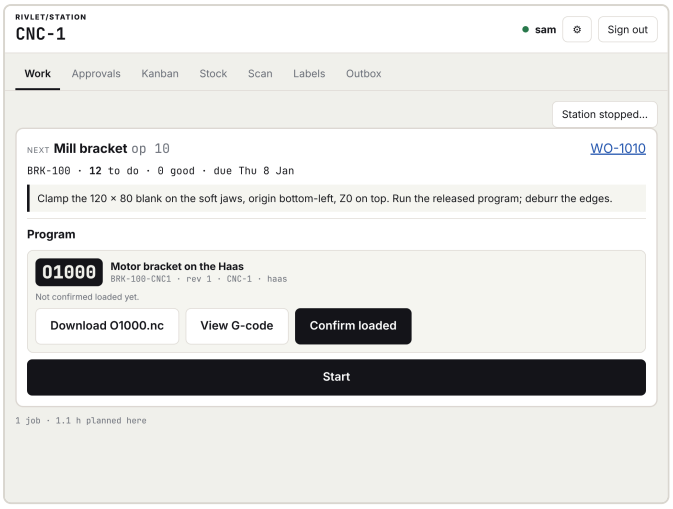
At some benches, an edge bridge sends the meter's readings for you. They appear on the tablet as they arrive. See Administration.

At a machine: loading a program

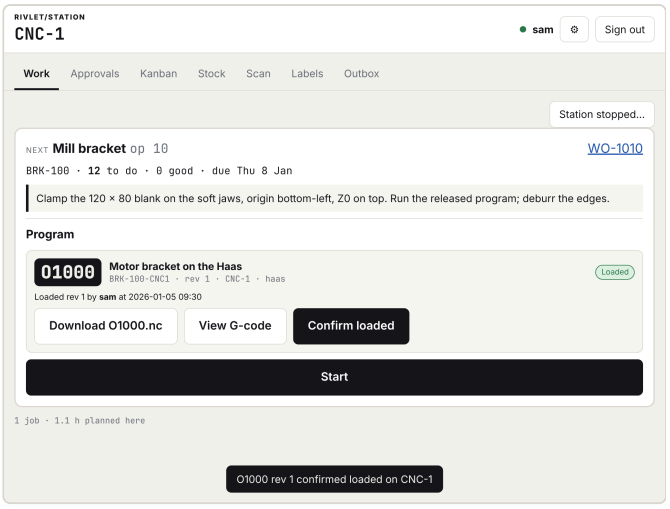
At a machine station (CNC-1, for example), a job carries its released program: the program number (such as O1001), its revision and the machine.

To load a program:

1. On the job card, tap **View G-code** to read it, or **Download** to save it for the controller.
2. Load it into the machine's controller.
3. Tap **Confirm loaded**. The card says who loaded which revision, for example "Loaded rev 1 by sam".

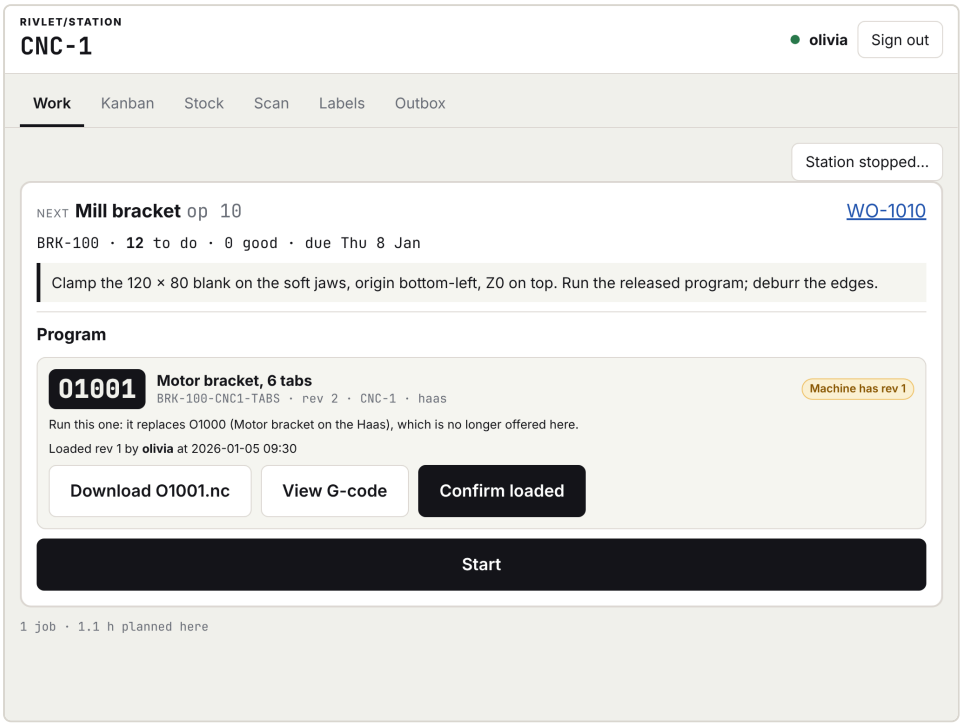


Program for the job. At a machine station the job carries its released CNC program: number, revision, machine and whether it is loaded. `floor-prog-card-sam-tablet`



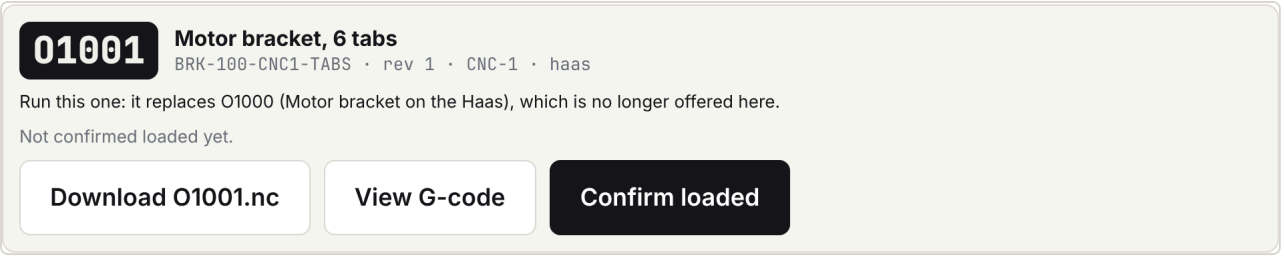
Program confirmed loaded. Confirm loaded records who loaded which revision on which machine. `floor-prog-loaded-sam-tablet`

When a newer revision is released, the card warns "Machine has rev 1" until someone loads the new one and confirms it.



The machine has an old revision. Revision 2 is released but the machine was confirmed with revision 1, so the card warns "Machine has rev 1" until someone loads and confirms the new one. `design-station-stale-olivia-tablet`

The card offers only the program to run. When a newer program replaced an older one for the same operation and machine, the card says so, for example "Run this one: it replaces O1000 (...), which is no longer offered here." Load the program the card shows.



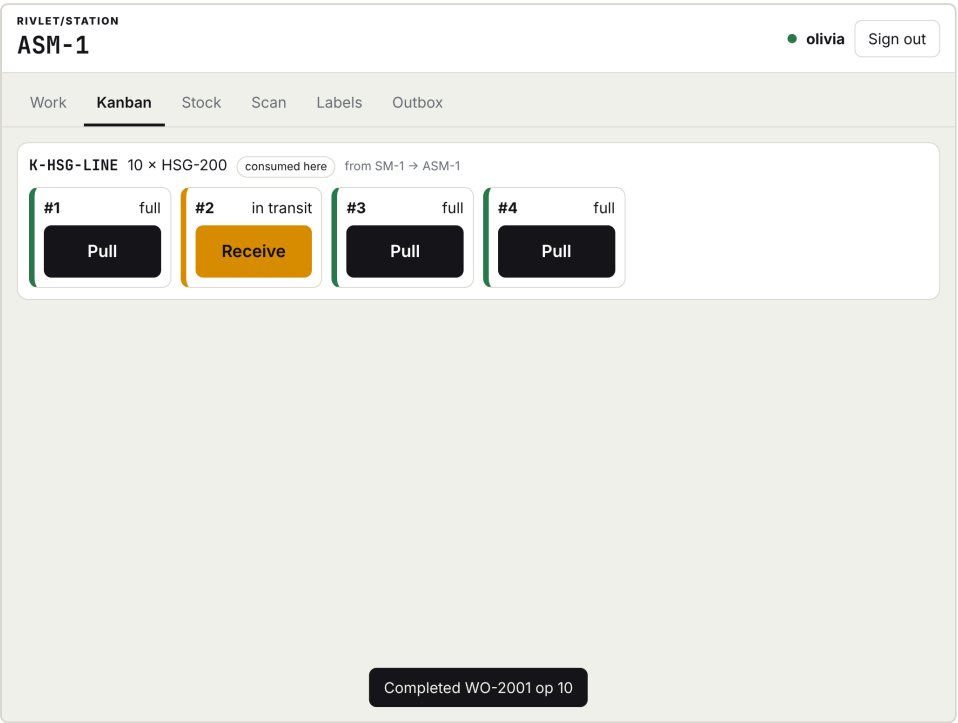
Which program to run. The job card offers one program for the operation and says so: "Run this one: it replaces O1000", the program it was released over, which is no longer offered here. `design-station-run-this-olivia-tablet`

The confirmation note after each action shows at the bottom of the screen, or just under the tabs when a button or field is at the bottom.

On a phone, when there are buttons both at the bottom and just under the tabs (at a machine station, **Confirm loaded** and **Start** below, **Station stopped...** above), the note stays at the bottom and covers them for a few seconds. Wait for it to go, or scroll.

Kanban

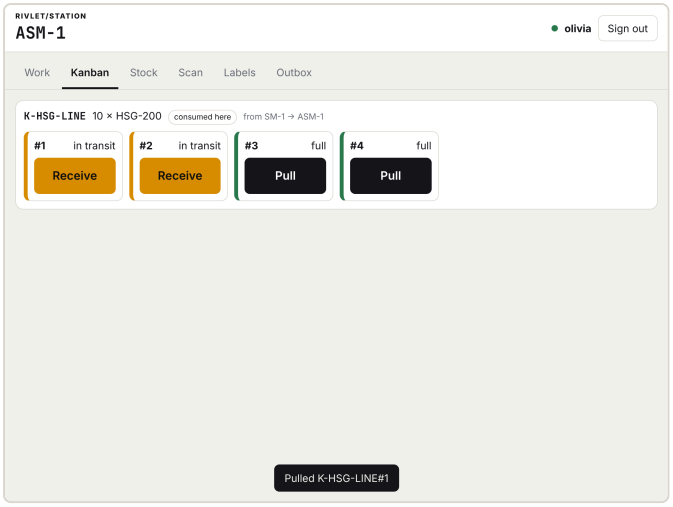
Some parts come to the line in containers, each with a kanban card. The card is the signal to refill.



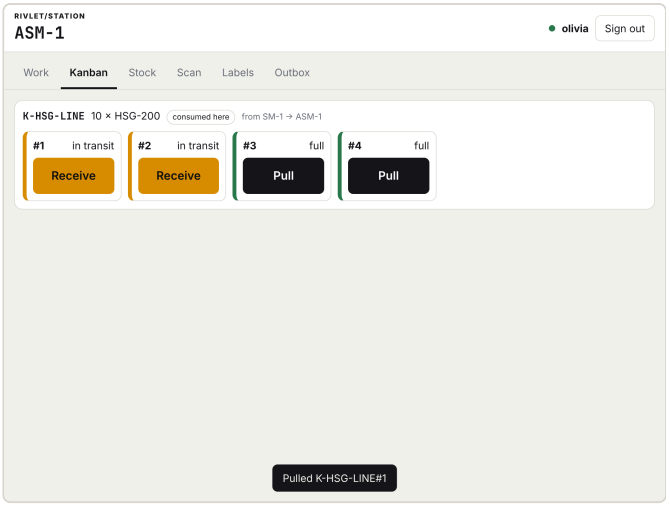
Kanban at ASM-1. Each card is one container of housings: full, empty (refill signalled) or in transit. `floor-kanban-board-olivia-tablet`

To pull a card

When you open a full container, tap **Pull** on its card. The empty card is the refill signal. A refill from the supermarket runs on its own, and seconds later the card shows *in transit*, with **Receive**.



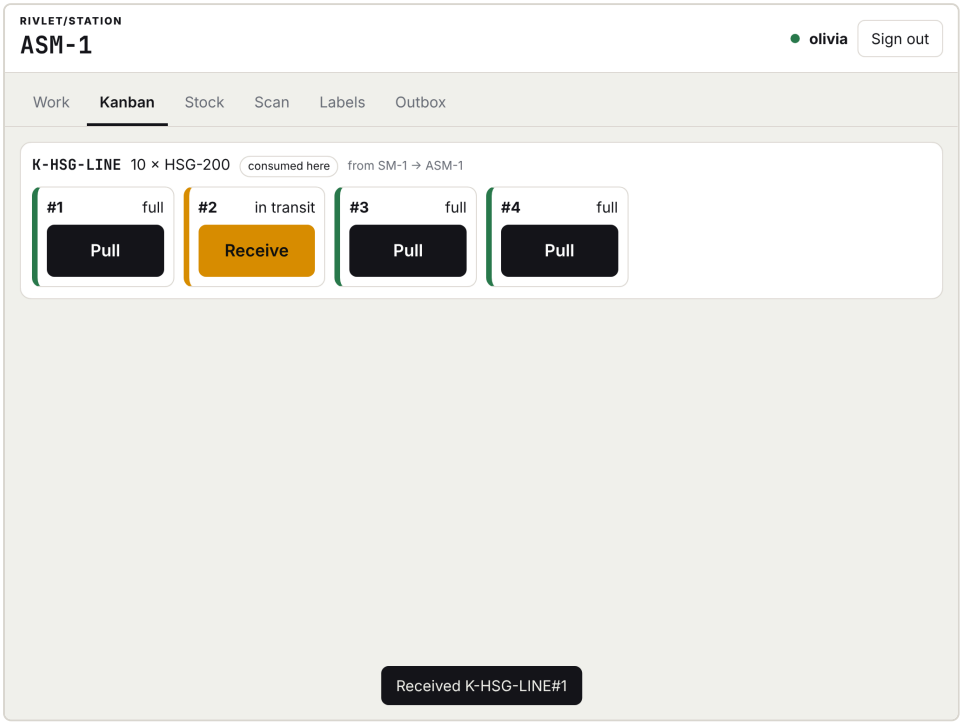
Card pulled. Pull the card when you open a container; the empty card is the refill signal. `floor-kanban-pulled-olivia-tablet`



Refill on its way. Seconds later the supermarket refill is on its way and the card shows in transit, with Receive. `floor-kanban-in-transit-olivia-tablet`

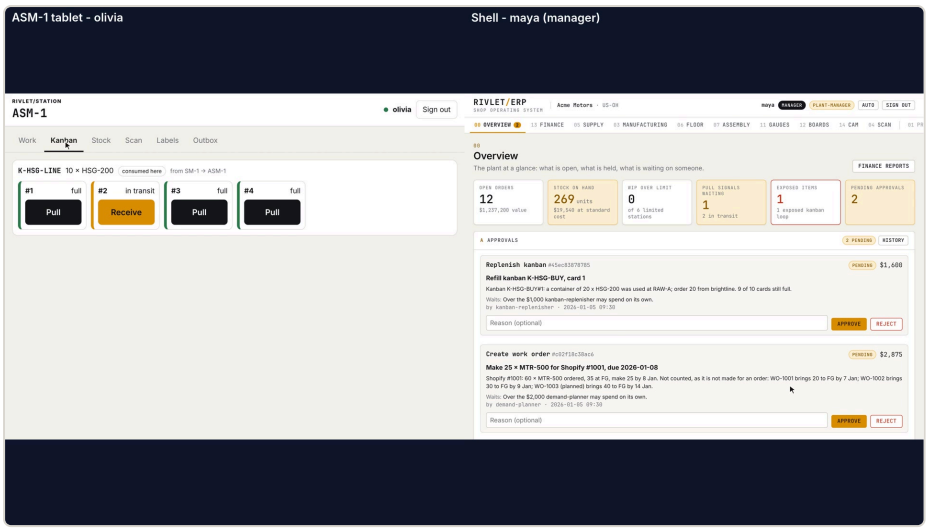
To receive a card

When the container reaches the line, tap **Receive** on its card. The card is full again, and the parts are in stock at your station.



Card received. Tap Receive when the container reaches the line; the card is full again. `floor-kanban-received-olivia-tablet`

A refill bought from a supplier may cost more than agents may spend on their own ($1,000 in the demo). It then waits for a person to approve it. The card stays empty until it arrives. If an empty card has no refill on its way, tap **Request refill**.



Movie: The kanban loop (40 s). A pull at ASM-1 empties a card; the kanban-replenisher proposes a refill from the supermarket, which runs on its own (under $1,000); a bought refill of housings from Brightline ($1,600) waits for maya to approve; the line card is received back at ASM-1. `floor-kanban-loop`

Chapters: Pull a card 0:00 · The refill runs on its own 0:09 · A bought refill waits for approval 0:14 · Receive the card 0:31

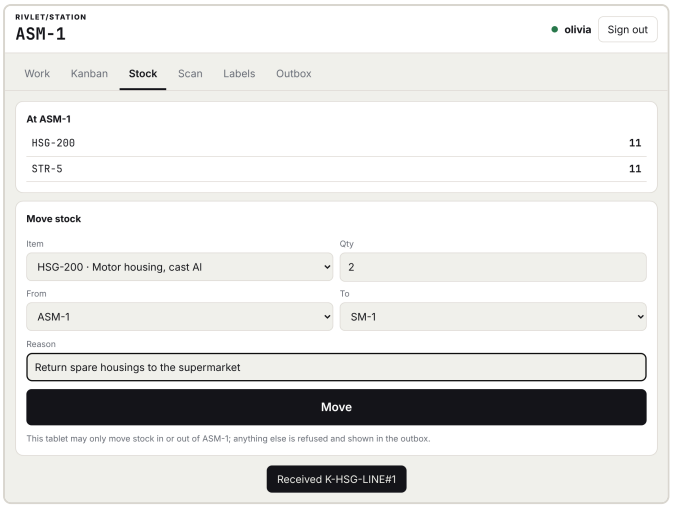
Moving stock

The **Stock** tab shows what is at your station, and a **Move stock** form.

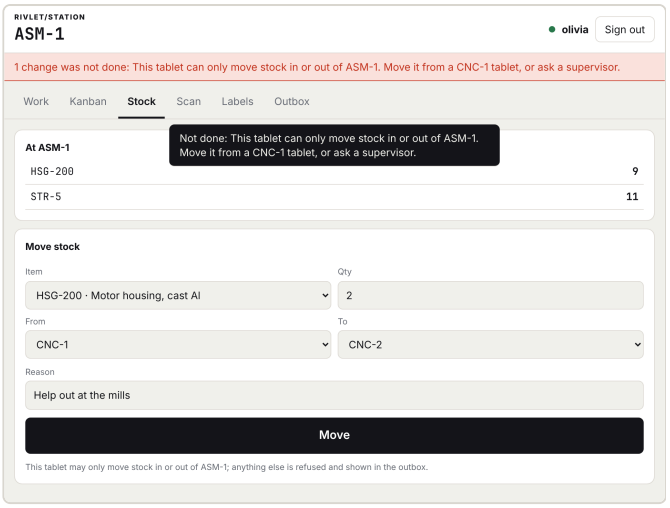
To move stock:

1. Pick the **Item** and enter the **Qty**.
2. **From** starts at your station. Pick **To**.
3. Write a **Reason**.
4. Tap **Move**.

One end of the move must be your station. A move that does not touch your station is refused: "This tablet can only move stock in or out of ASM-1. Move it from a CNC-1 tablet, or ask a supervisor."



Move stock. Stock at the station on top; From starts at this station. Fill in item, quantity, from, to and a reason, then tap Move. `floor-stock-move-filled-olivia-tablet`



Move refused, in plain words. A move that does not touch this station is refused; the red bar says why and what to do, and opens the outbox. `floor-stock-refused-olivia-tablet`

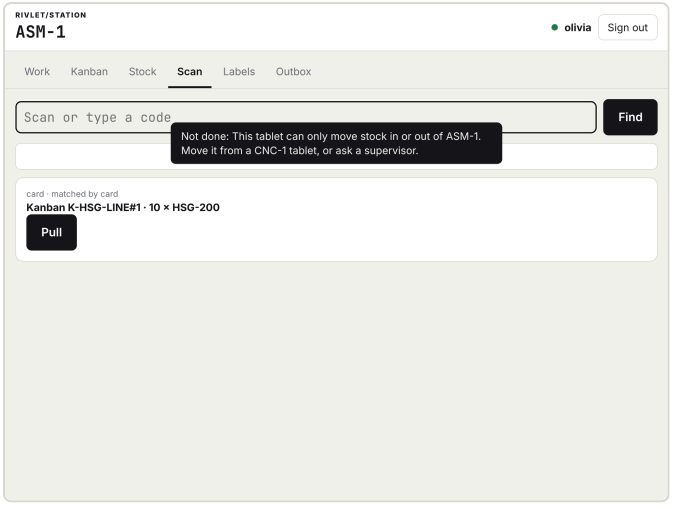
A move that would put more at a station than its WIP limit allows is refused too, with the limit ("WIP limit at ASM-1 is 60...").

Booking goods in from outside the plant (a receipt) is for buyers and planners. A floor operator is not offered it.

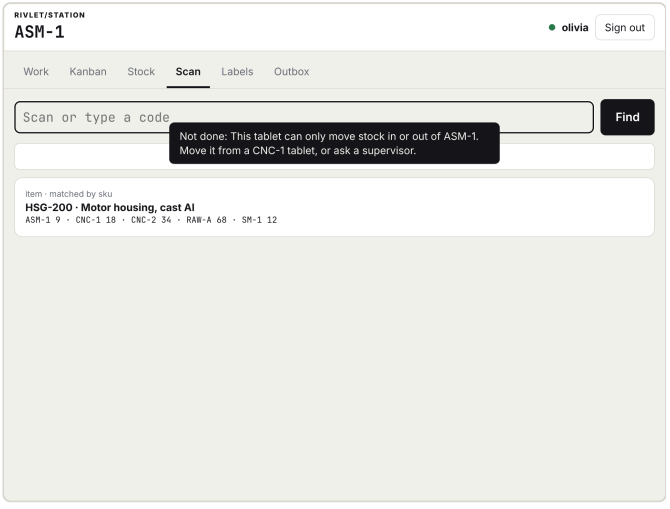
Scanning and labels

On the **Scan** tab, scan or type any code:

- A kanban card offers **Pull** or **Receive** right there.
- An item shows where it is in stock across the plant.
- A code that matches nothing says so: "Nothing found for NOPE-0000."

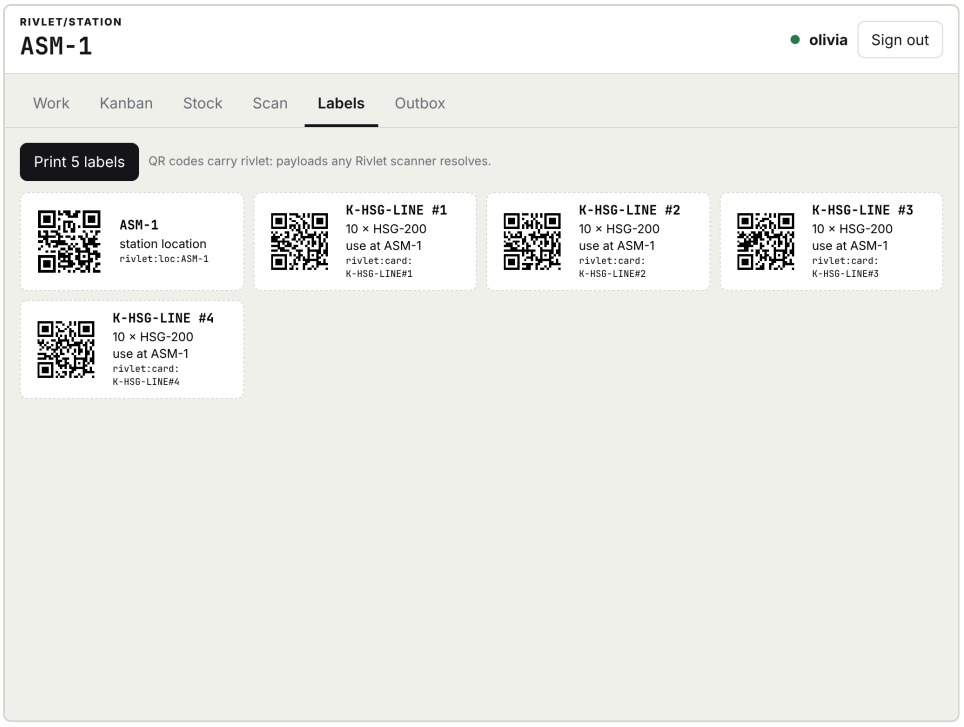


Scan a kanban card. Scanning a card's QR label finds it and offers Pull or Receive right there. `floor-scan-card-olivia-tablet`



Scan an item. Scanning an item shows where it is in stock across the plant. `floor-scan-item-olivia-tablet`

The **Labels** tab prints QR labels for the station and its kanban cards.



Labels to print. The Labels tab prints QR labels for the station and its kanban cards. `floor-scan-labels-olivia-tablet`

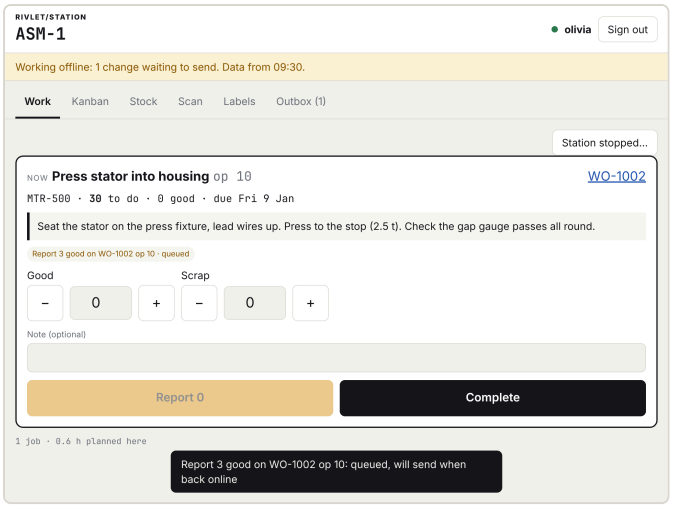
A scanned Rivlet label is never taken for a badge. If you scan one before signing in, the tablet says: "That is a Rivlet label, not a badge: sign in first, then scan it on the Scan tab."

Working offline: the outbox

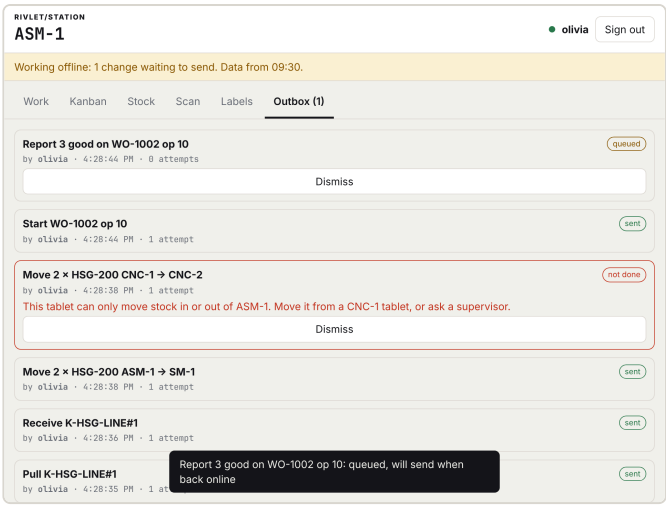
Every change you make goes through the tablet's **Outbox**: one item per tap, owned by the person who made it. When the tablet is online it is sent at once. A change counts only once the node has it.

When the network drops:

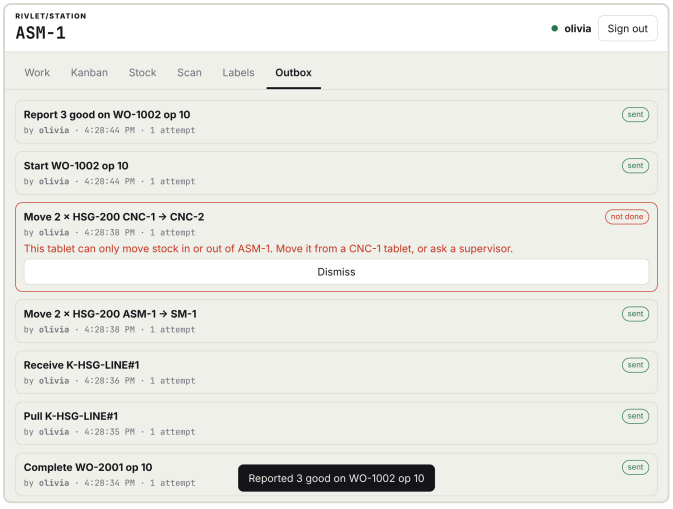
1. An amber bar says the tablet is offline.
2. Keep working. What you do waits on the tablet as "queued", not done.
3. When the tablet is back online, everything waiting is sent once, in order, by itself.



Working offline. Without a connection the amber bar says so; the report waits on the tablet as "queued", not done. `floor-offline-queued-olivia-tablet`

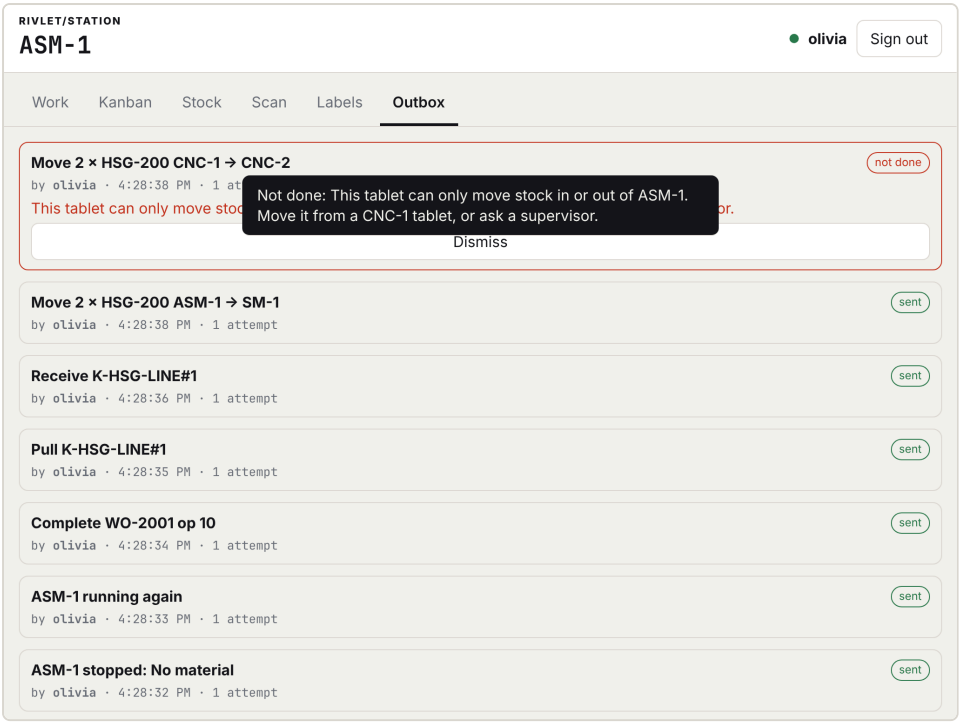


Outbox while offline. The outbox shows the waiting report, as olivia's; it is sent once, automatically, when the tablet is back online. `floor-offline-outbox-pending-olivia-tablet`



Back online. Back online, the waiting report is sent once and marked sent. `floor-offline-outbox-sent-olivia-tablet`

The **Outbox** tab lists everything this tablet sent, newest first, with who sent it: **sent**, waiting, or **not done**. A refused change shows the reason in red and **Dismiss**. The red bar at the top belongs to the person who made the change: it goes when they sign out.



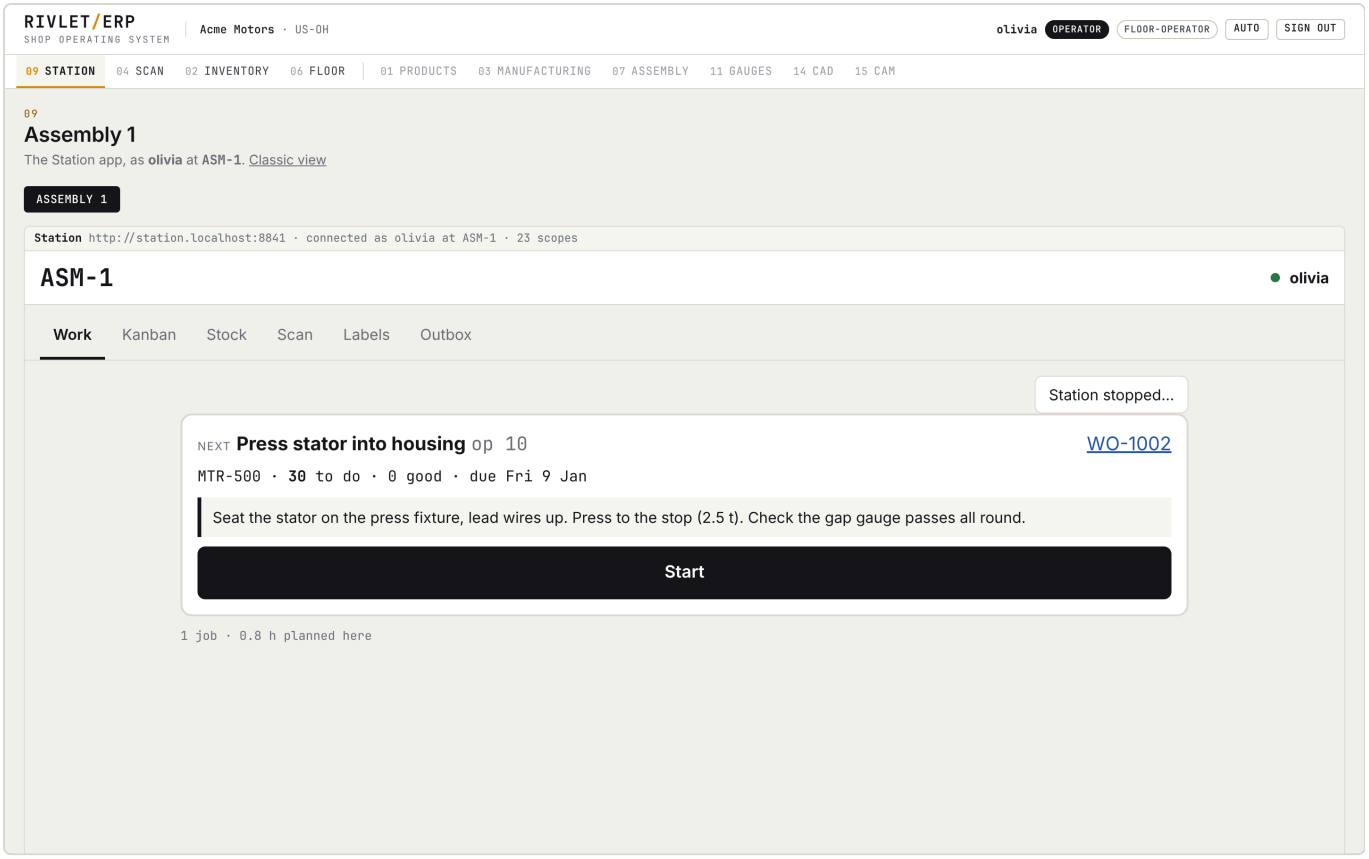
Outbox with a refusal. The outbox lists everything this tablet sent, newest first, with who sent it; a refused change shows the reason and Dismiss. `floor-offline-outbox-refused-olivia-tablet`

On a phone, scroll the tab bar to reach the Outbox.

The Station in the shell

You can also open the Station in the shell, as **09 Station**, signed in as yourself. It works as on the tablet.

If you work at one station, 09 Station opens it at once. If you work at several, pick the one you are at.



Station as Olivia. The Station app embedded in the shell, for the person signed in. Seen by Olivia (floor operator). `shell-station-page-olivia-desktop`

Running a station

This chapter is for station managers and line supervisors. You do everything an operator does (see On the floor), and you also approve, set targets, run the TV boards and keep the line moving.

	Station manager	Line supervisor
Works at	Their own stations	Every station
Approves up to	$1,000	$5,000
Approves TV boards	At their own stations	At any station
Sets daily targets	Yes	Yes
Starts on	06 Floor	06 Floor

Approving

Agents and people propose things that the plant's rules hold for a person: a kanban refill bought from a supplier, work the demand planner wants to make, a purchase order. You may approve what is worth up to your limit. Nobody may approve their own proposal.

At the tablet

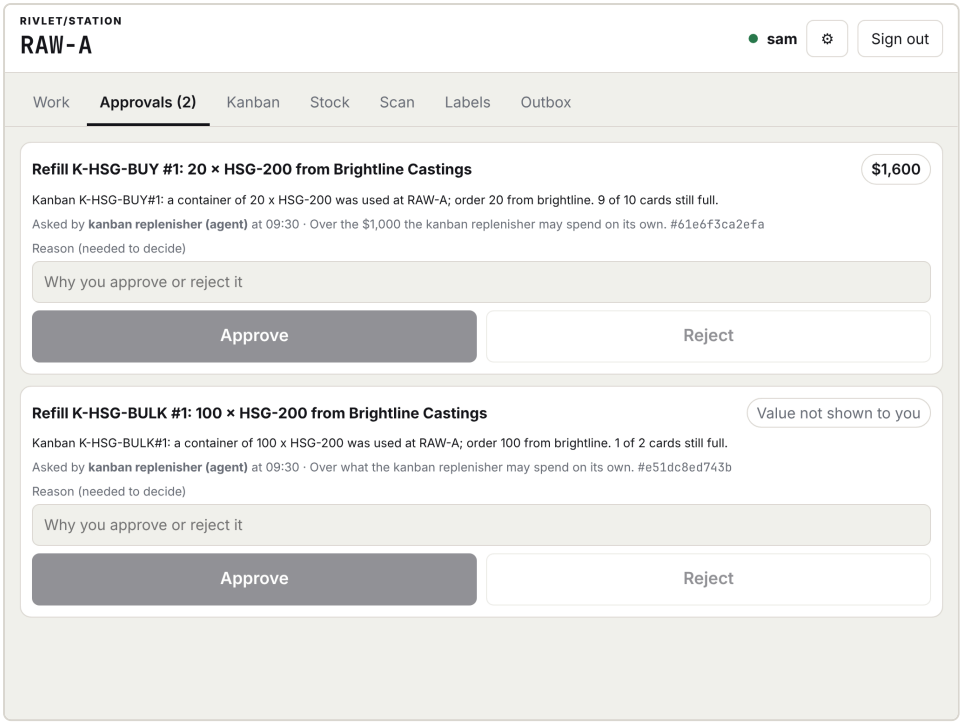
Signed in at a Station tablet, you have an **Approvals** tab. It lists what waits for a person and concerns this station, with a count on the tab:

- a kanban refill for a loop used here, or refilled from here;
- work to make at this station, or work whose units come here (FG, for an order that ships from there);
- cancelling such work.

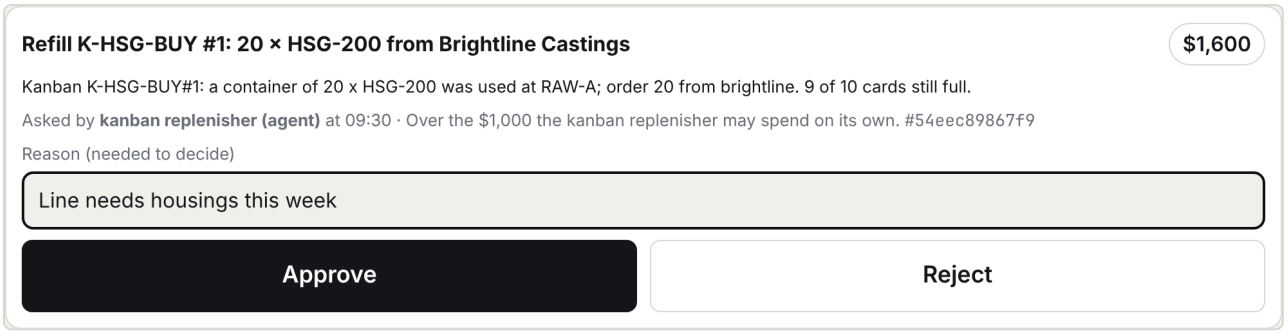
Purchase orders and RFQs concern no station: decide them in the shell (below). Floor operators have no Approvals tab, and nor does a tablet with nobody signed in.

To approve or reject at the tablet:

1. Sign in with your badge and tap **Approvals**.
2. Read the card: what it asks ("Refill K-HSG-BUY #1: 20 × HSG-200 from Brightline Castings"), what it is worth, who asked, when, and why it waits.
3. Type a **Reason**. **Approve** and **Reject** stay off until you do.
4. Tap **Approve** or **Reject**. A note says what happened ("Approved: … Purchase order PO-… placed."), and the card leaves the list.



Approvals at the station. Sam, a line supervisor, signs in at RAW-A, where Brightline's housings arrive: the Approvals (2) tab lists what waits for a person and concerns this station, each card saying what it asks, what it is worth as far as he may see, who asked and why it waits. `floor-approvals-tab-sam-tablet`

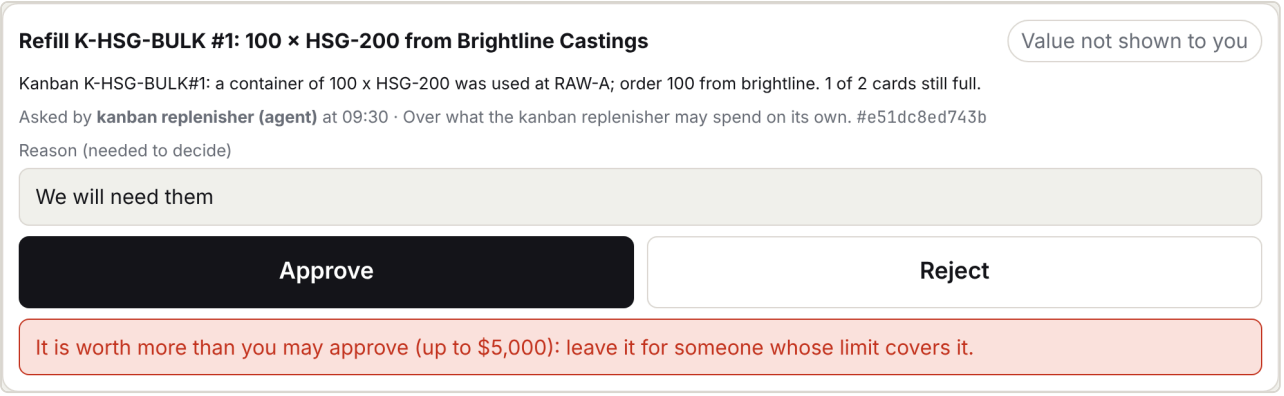


A refill to approve. The $1,600 refill of K-HSG-BUY from Brightline: within Sam's $5,000, over the $1,000 the kanban replenisher may spend on its own. Approve and Reject wait for a reason, kept with the decision. `floor-approvals-card-sam-tablet`

A tablet never shows prices, so a card shows what it is worth only when that is within your limit. Otherwise it says "Value not shown to you".

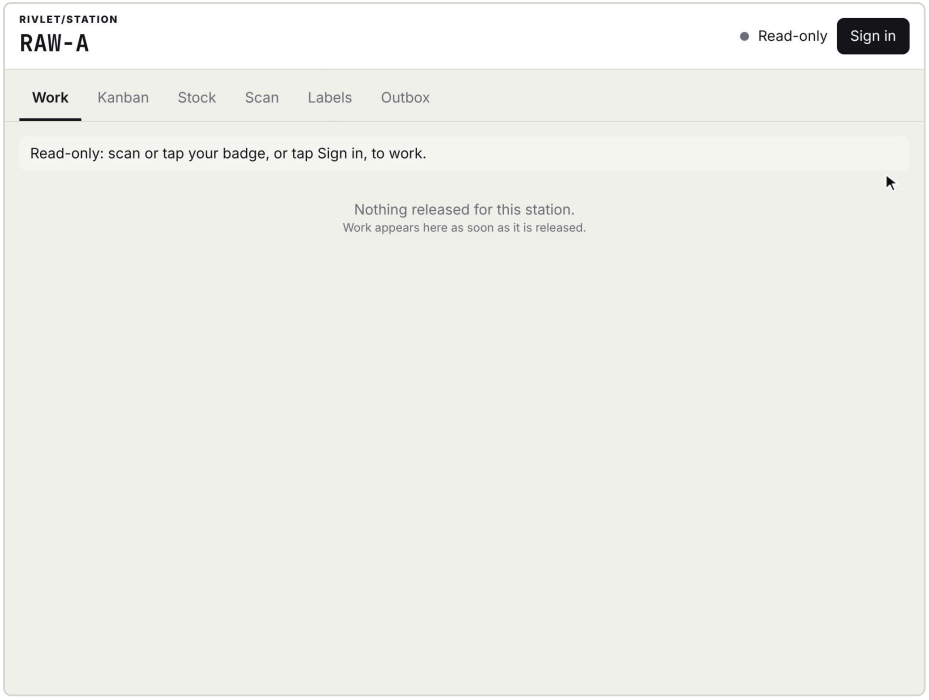
The tablet decides as the shell does. When it refuses, the card says why:

- "It is worth more than you may approve (up to $5,000): leave it for someone whose limit covers it."
- "You asked for this: someone else approves it." **Approve** is off on your own requests; you may still reject them.
- "Only plant-manager approves this: leave it for them." A rule named who approves it.
- "Already decided: someone approved, rejected or withdrew it in the meantime."
- "Cannot reach the node right now: decide once the tablet is back online." Decisions are never queued in the outbox.



Over his limit, in plain words. The bigger refill (100 housings) is over what Sam may approve: its card says "Value not shown to you" and why it waits, and approving is refused in plain words, without naming the amount. `floor-approvals-over-limit-sam-tablet`

When nothing at the station waits for a decision, the tab says so: "Nothing at RAW-A waits for a decision."



Movie: Approve at the station (53 s). Sam, a line supervisor, badges in at the RAW-A tablet: an Approvals tab lists the two refills waiting there. He approves the $1,600 refill from Brightline with a reason, and its purchase order is placed; the bigger refill is over his limit, its value not shown to him, so approving it is refused in plain words and he rejects it with his reason. `floor-approve-at-the-station`

Chapters: An Approvals tab 0:00 · Within his limit 0:13 · Over his limit 0:30

In the shell

Everything else, and what waits at stations other than this one, is in the shell:

1. Open the shell, on a phone or a desk computer, and sign in as yourself.

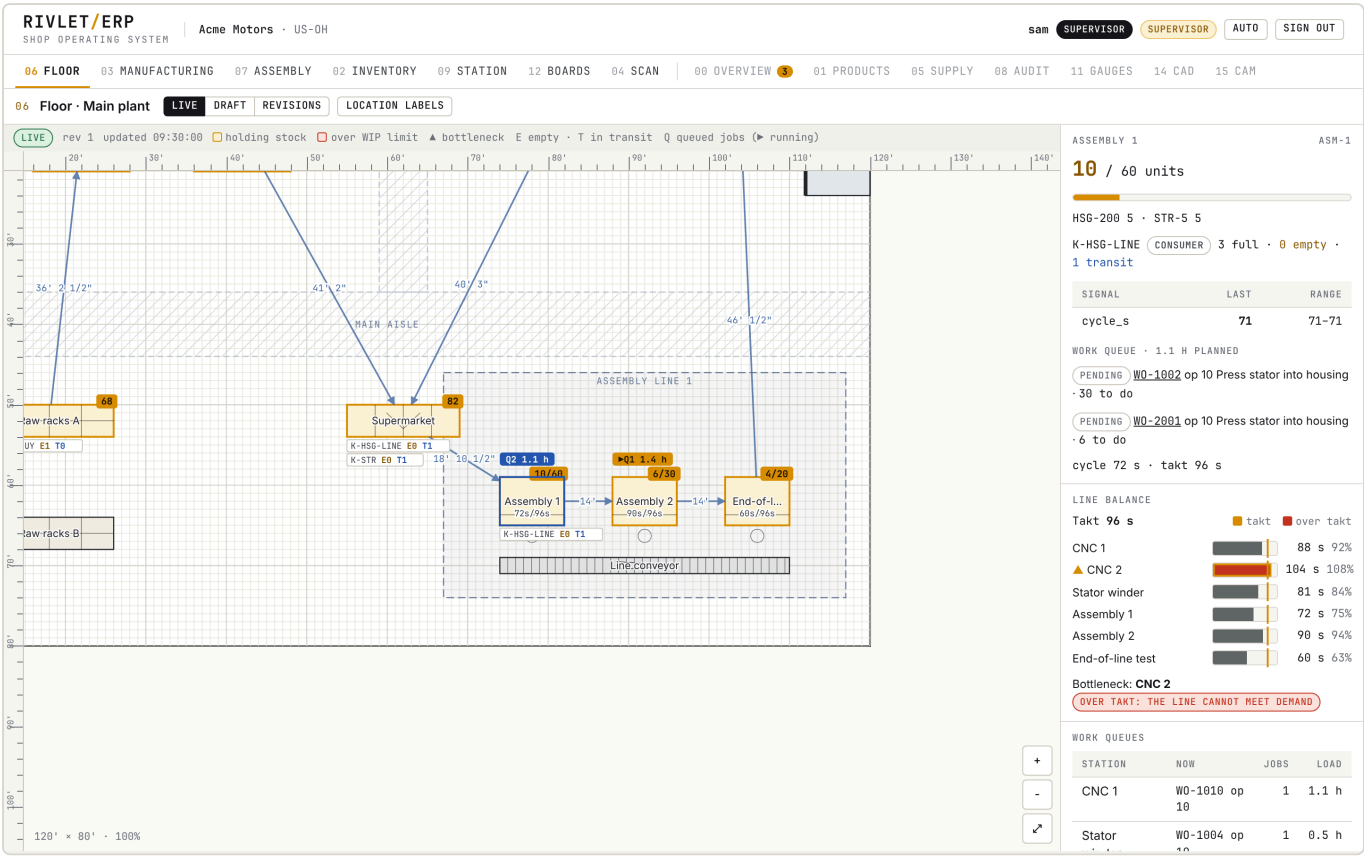
2. Open **00 Overview**. Its number counts what waits for a decision. Panel **A Approvals** lists each one.
3. Read the card: the action, its value, who proposed it and why it waits.
4. Type a reason, then press **Approve** or **Reject**.

The full steps, the limits and the history are in Managers and the CFO.

> **Note:** Quality holds are not approvals. Only quality may sign off a held unit, whatever your limit. See Quality.

The live floor

06 Floor › Live shows every station on the plan with its queue, what it is running, its WIP against its limit, and the bottleneck. Click a station to see its queue and planned hours. A stoppage recorded at a tablet shows here too.



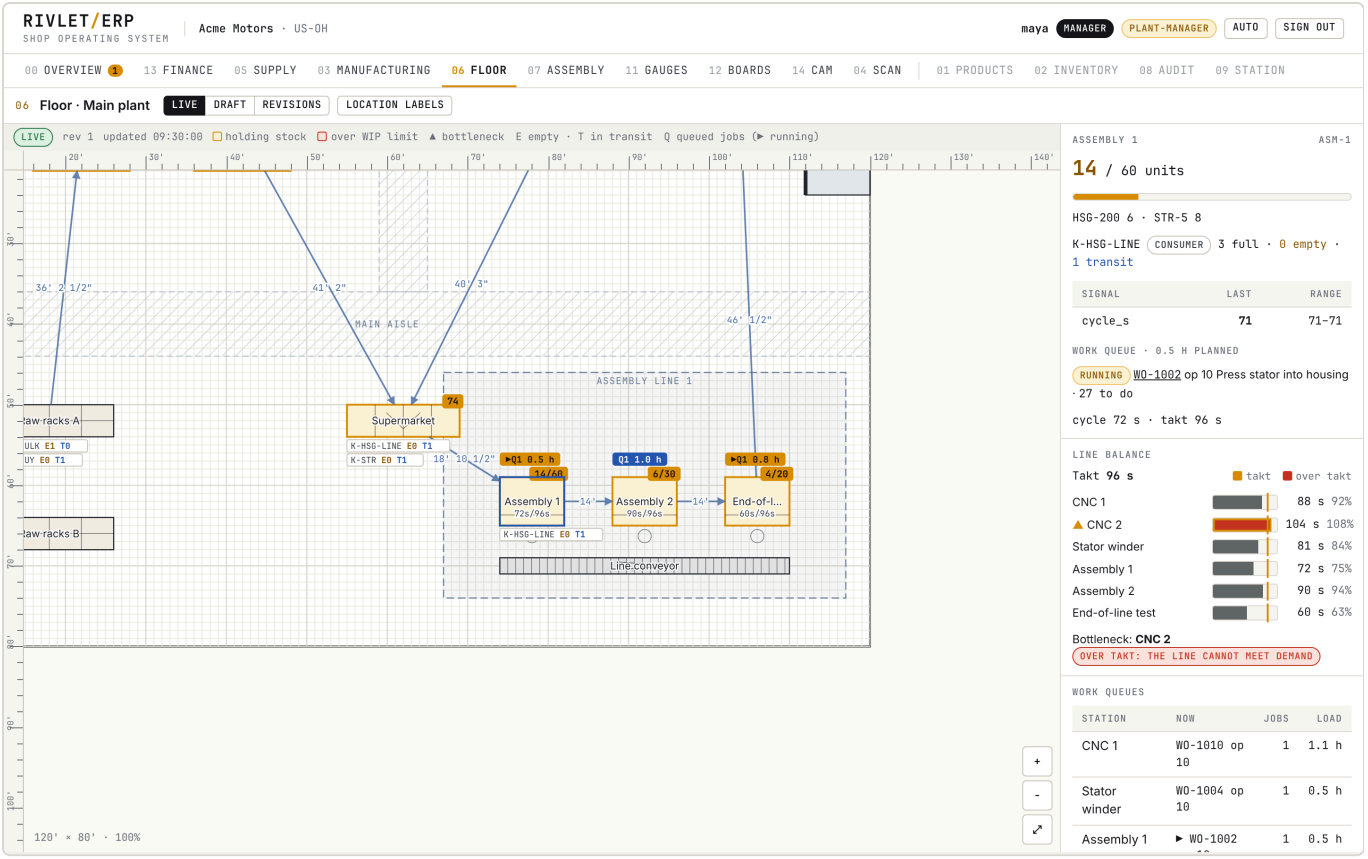
Live floor, a station selected. The live floor with Assembly 1 selected: its queue, what is running and the planned hours. `shell-floor-live-station-sam-desktop`

How work travels between stations

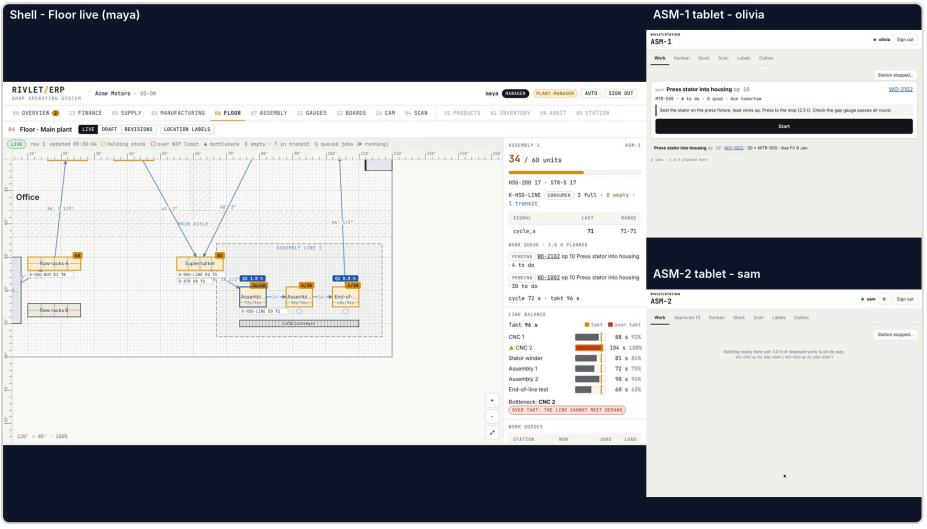
A work order follows its routing, one operation at a time. The MTR-500 motor, for example, is pressed at ASM-1 (op 10), wired at ASM-2 (op 20, with a torque check) and tested at TEST (op 30).

- A station's queue holds the orders whose current operation is there.

- When an operator completes an operation, the next one is current at once: it appears on the next station's tablet and moves on the live floor.
- Reporting at a station uses up that operation's materials there.



The live floor. Floor › Live: every station on the plan with its queue; the side panel lists what each station is working on now. `floor-travel-live-floor-maya-desktop`

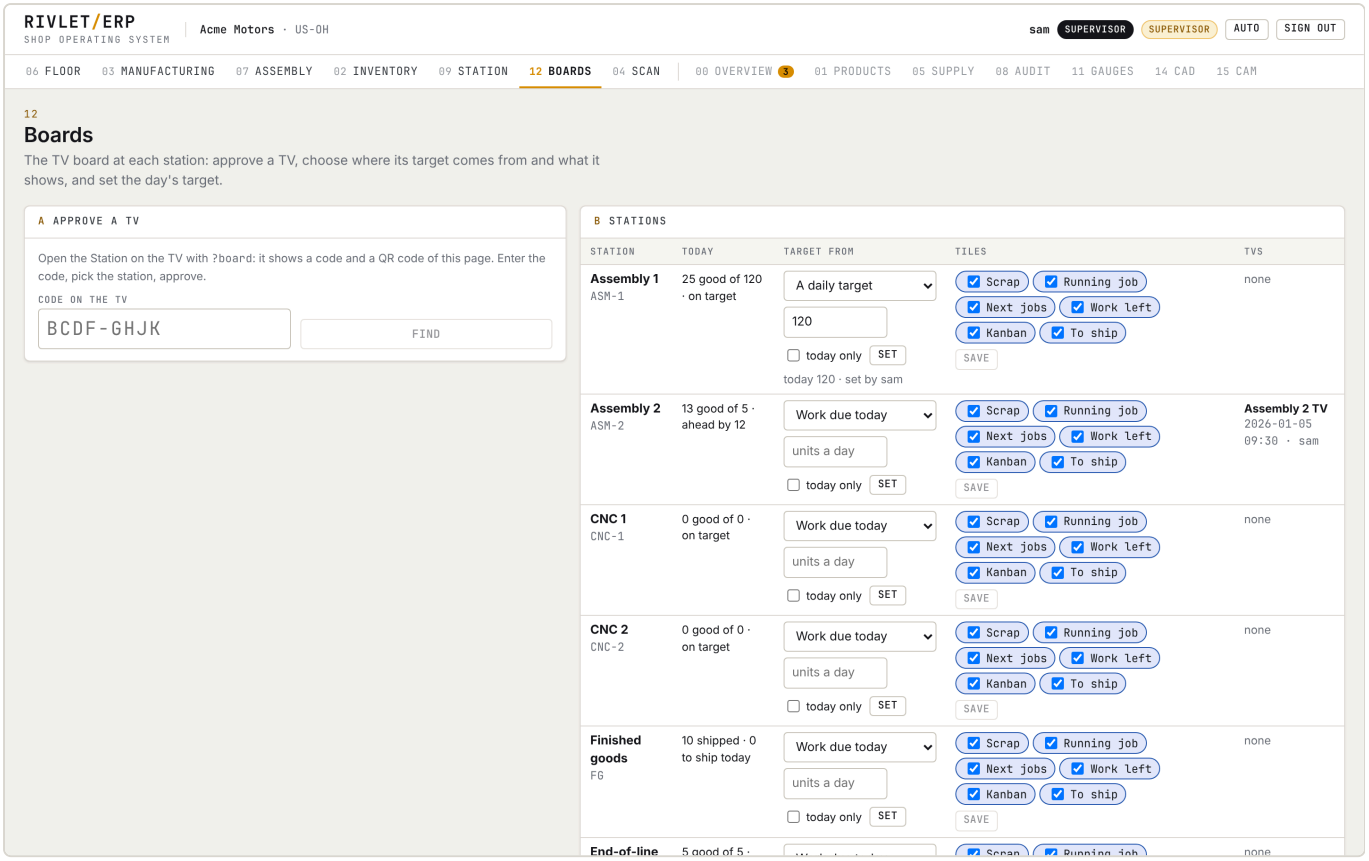


Movie: How work travels between stations (45 s). WO-2102 op 10 is pressed and completed at ASM-1; it appears at once in the ASM-2 queue as op 20, and the shell's live floor shows it move from Assembly 1 to Assembly 2. `floor-work-travels`

Chapters: One order, three stations 0:00 · Op 10 at ASM-1 0:06 · Op 20 is ready at ASM-2 0:23 · Op 20 starts 0:32

Daily targets

Each station's TV board measures the day against a target. You set where the target comes from in **12 Boards › B Stations**.



12 Boards. Boards: each station's day so far, where its target comes from (set, or the work due today), the units a day, and which tiles its TV shows. `shell-boards-sam-desktop`

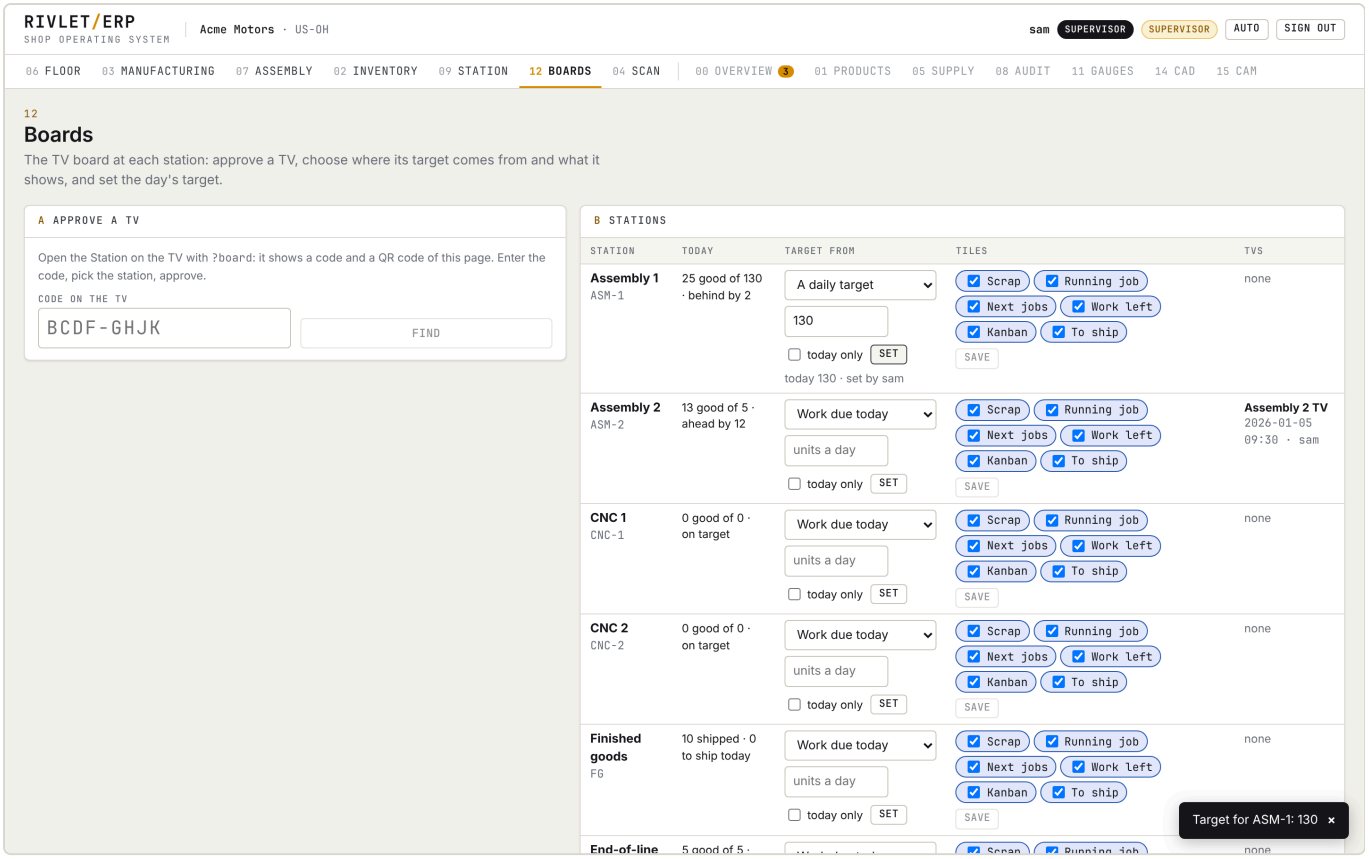
For each station you see:

- **Today**: the day so far, for example "25 good of 120 · on target".
- **Target from**: **Work due today**, **Capacity of the shifts** (what fits in today's shifts at the work's cycle time), or **A daily target** that you set.
- **Tiles**: which tiles its TV shows: **Scrap**, **Running job**, **Next jobs**, **Work left**, **Kanban** and **To ship**.
- **TVs**: the TV boards approved for it.

To set a daily target:

1. In the station's row, choose **A daily target** under **Target from**.
2. Type the units a day.
3. Tick **today only** if it is for today alone. Leave it clear to keep it from now on.
4. Press **Set**. The station's TV shows the new target at once.

To change the tiles, tick or clear them and press **Save**.

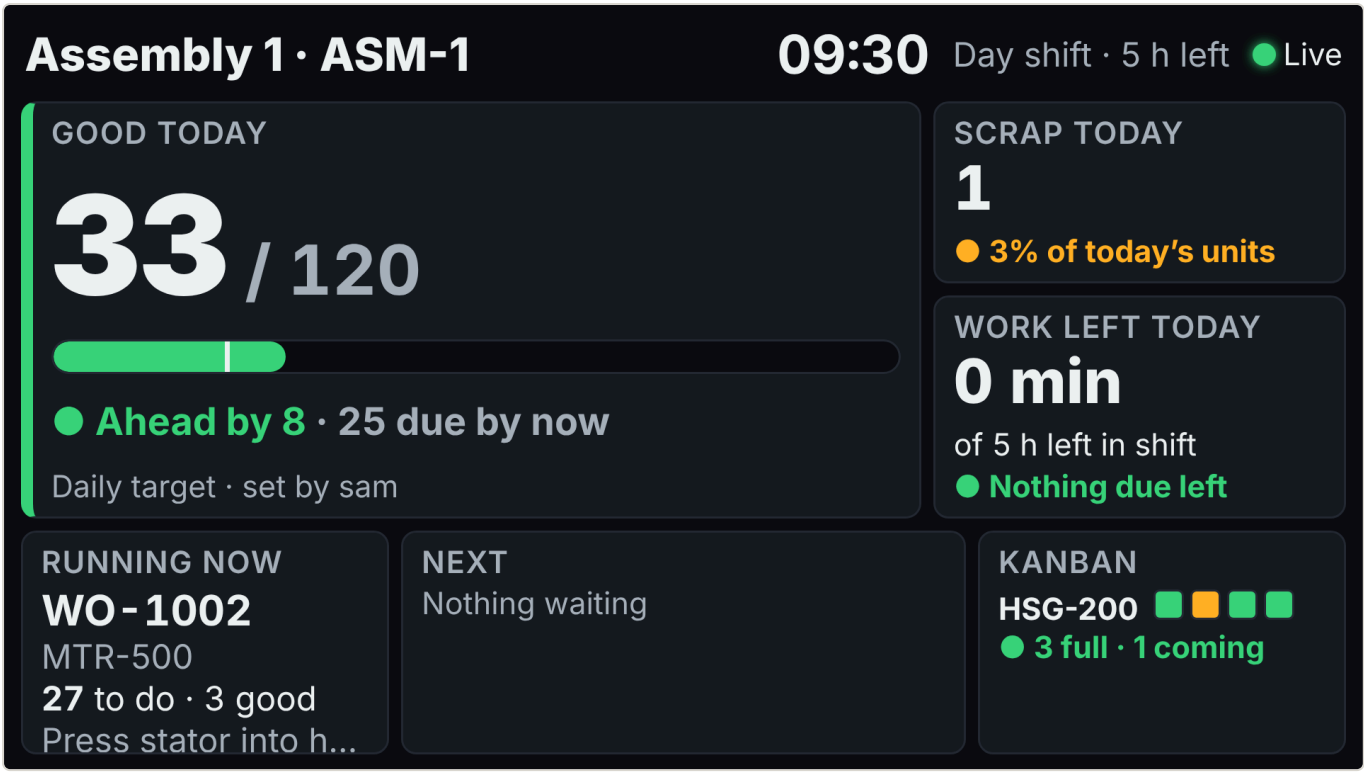


Set a daily target. Sam sets Assembly 1's target to 130 a day; its TV shows the new target at once. `shell-boards-target-sam-desktop`

A floor operator cannot approve a TV or set targets.

The TV board

A TV board shows one station's day, readable from five metres. Nobody signs in on it.

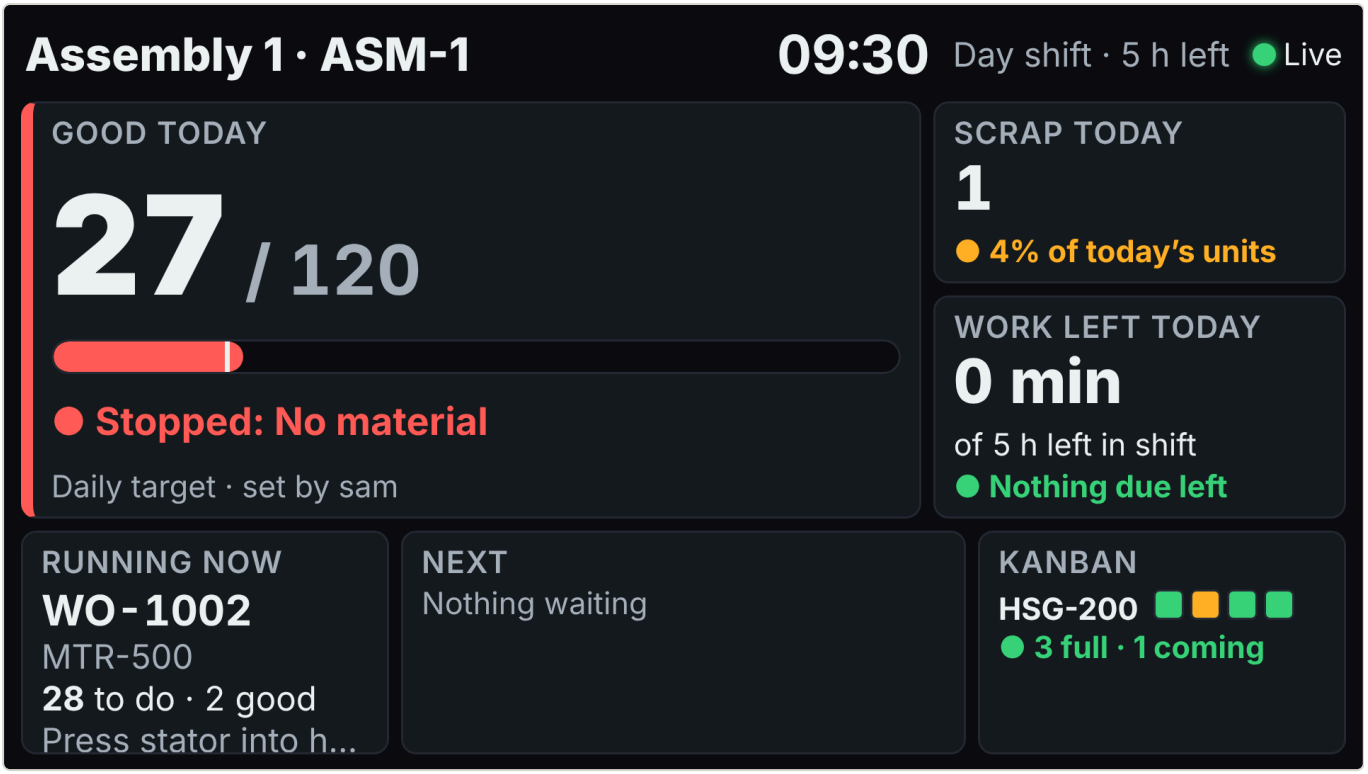


Assembly 1's board. The board: the station, the plant clock, the shift and its time left; good today against the target with the state in words; scrap, running, next, work left and kanban. `floor-tv-board-asm1-none-tv`

The board shows:

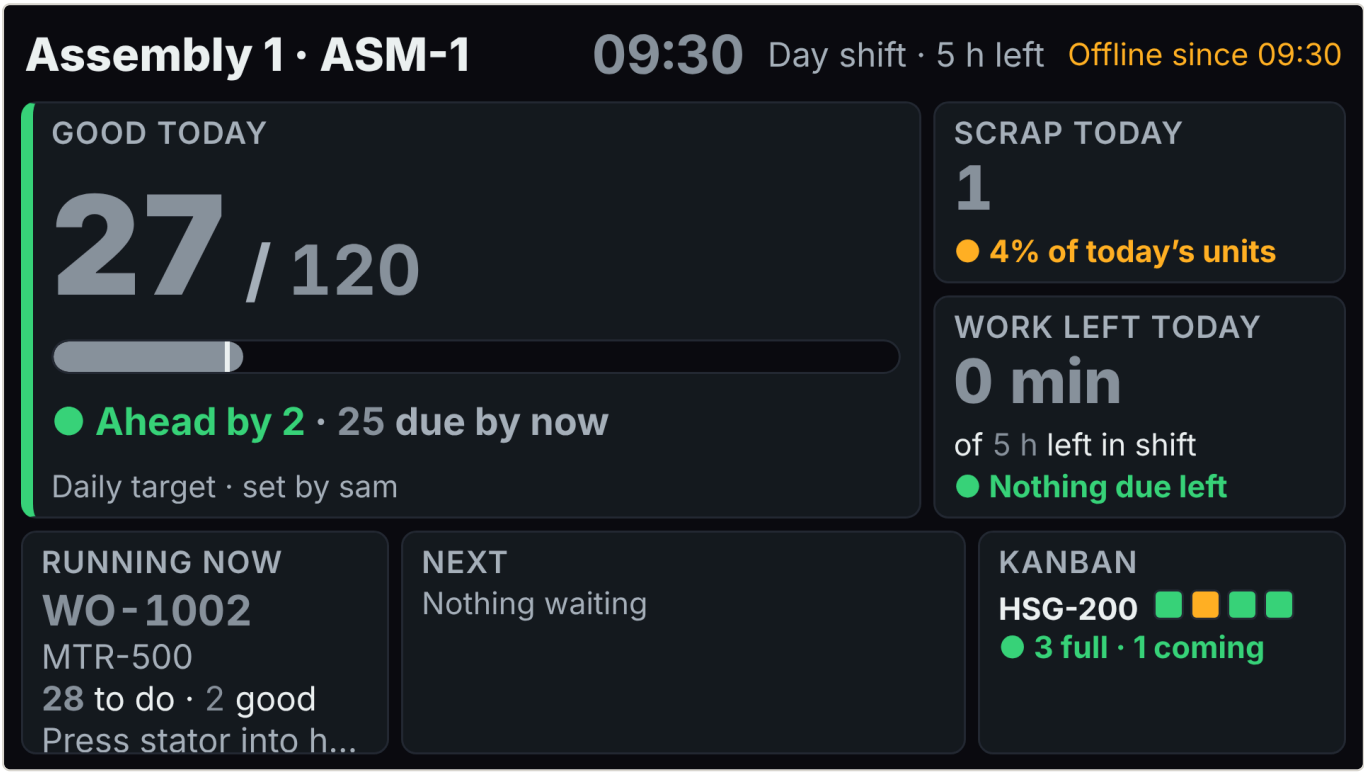
- **At the top:** the station, the plant's clock, the shift and its time left, and a live dot.
- **Good today** against the target, with a bar, and the state in words: **On target**, **Ahead by N** or **Behind by N**. Green, amber or red, always with the words.
- **Tiles:** scrap today, the job running now, the next jobs (each with the order it is made for), the work left in the shift, and the kanban cards.

Reports, scrap and stoppages at the tablet reach the board at once. A stoppage turns it red with the reason.



Stopped: No material. A stoppage at the tablet turns the board red, with the reason. `floor-tv-board-stopped-none-tv`

If the TV loses the network, it keeps its last numbers and says "Offline since 09:30" in plant time. After a minute the figures grey out, so nobody reads old numbers as live. When the network is back, it is live again by itself.



Offline. Offline, the board keeps the last numbers, greys them and says "Offline since 09:30" in plant time. `floor-tv-board-offline-none-tv`

At FG, where orders ship from, the board is about shipping: what is to ship today, what shipped today, and what is due later. See Shipping.

To set up a TV board

You need a phone (or any device) where you can sign in to the shell.

1. On the TV, open the Station's address with `?board` at the end (your administrator has it), or tap **This is a TV board** on its enroll screen. The TV shows a short code and a QR code.
2. Scan the QR code with your phone. **12 Boards** opens with the TV waiting. (Or open **12 Boards** and type the code under **Code on the TV**, then press **Find**.)
3. Pick the **Station it shows** and give the TV a **Name**.
4. Press **Approve** (or **Refuse**).

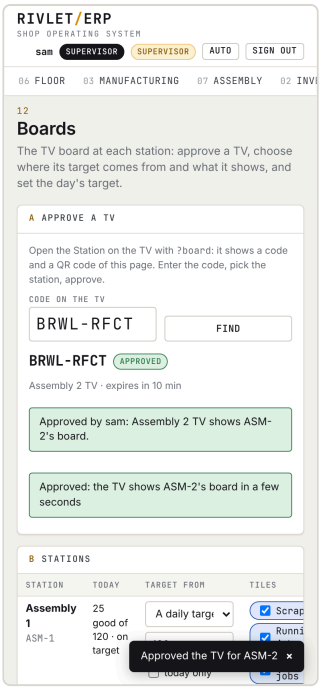
Within seconds the TV enrolls itself as a display for that station and shows its board. It may read that station's board and nothing else.



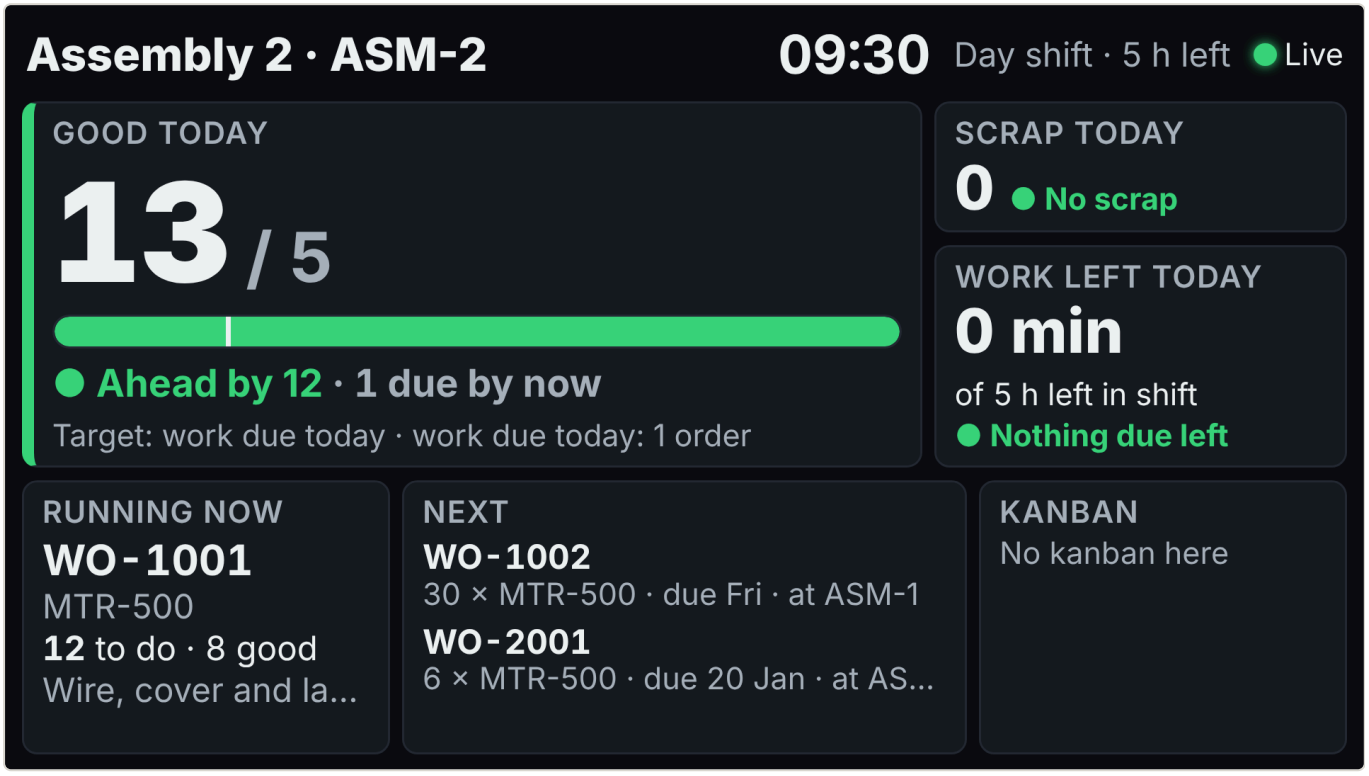
A TV asks to be a board. Open the Station on the TV with ?board: it shows a short code, a QR code of the approval page and where to approve it. `floor-tv-code-none-tv`



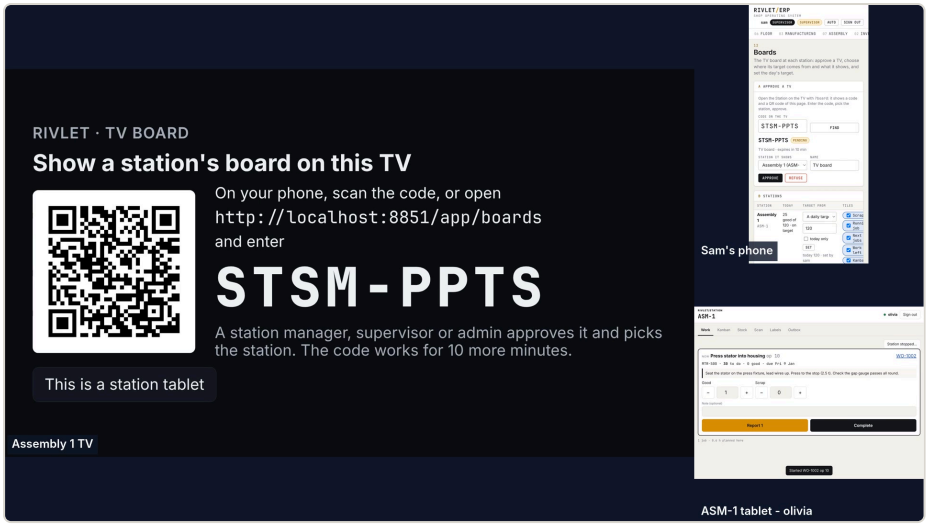
Approve a TV from a phone. Sam scans the TV's QR code with his phone: Boards opens with the TV waiting. He picks the station it shows and names it. `shell-boards-approve-tv-sam-phone`



The TV is approved. Approved: the TV enrolls itself as a display for Assembly 2 and shows its board. `shell-boards-approved-tv-sam-phone`



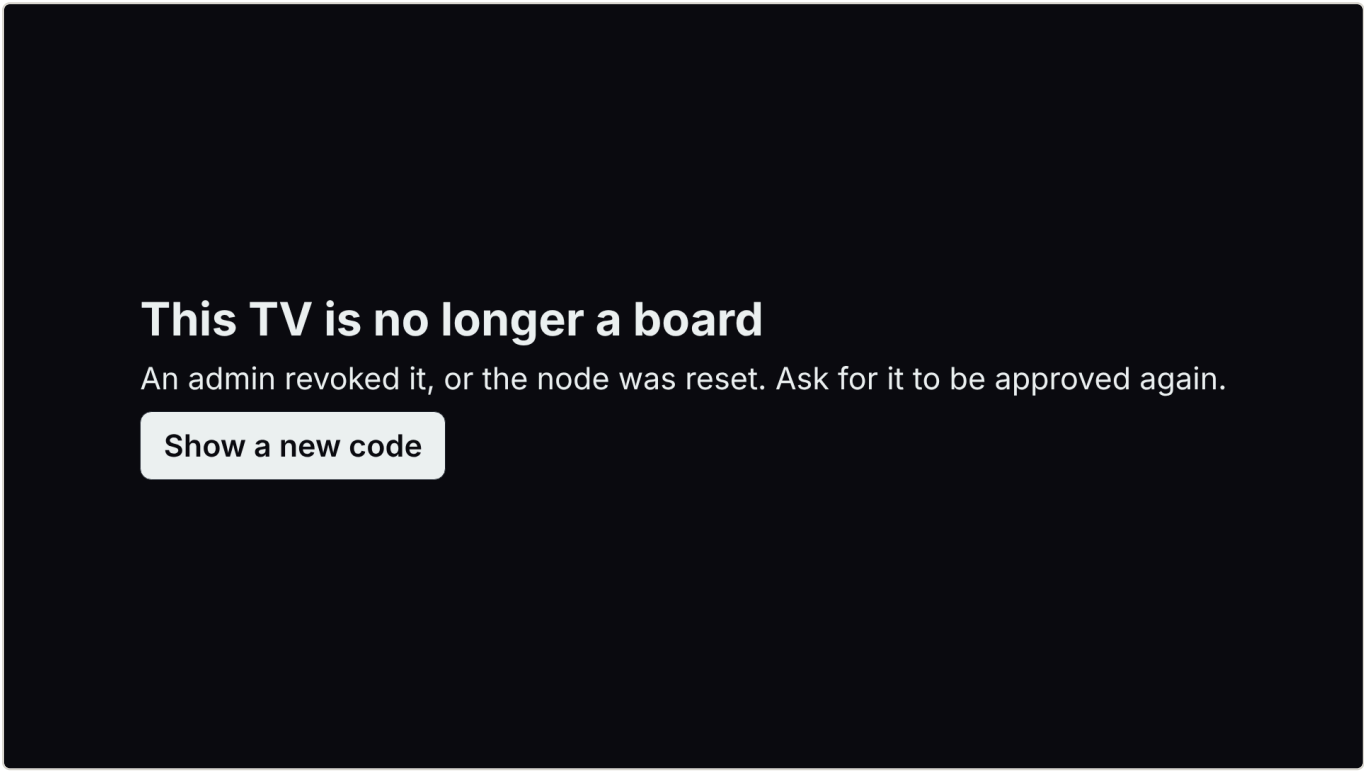
The TV shows its board. Seconds later the TV shows Assembly 2's board. `shell-boards-tv-board-none-tv`



Movie: A TV board at Assembly 1 (1 min 14 s). A TV opens the Station's board page and shows a code; sam approves it from his phone for Assembly 1; the board shows the day against the target; olivia's reports at the tablet move it live, a scrap and a stoppage too; offline it keeps its numbers and says since when; back online it is live. `floor-tv-board`

Chapters: The TV asks to be a board 0:00 · Approve it from a phone 0:05 · The board 0:17 · Live from the tablet 0:28 · Offline 1:00

If an administrator revokes the TV (Admin › Station devices), it says "This TV is no longer a board" at once. To use it again, tap **Show a new code** and approve it again.



A revoked TV. Revoked by an administrator, the TV says so at once and asks to be approved again. `floor-tv-revoked-none-tv`

The webcam badge scanner

On a tablet or kiosk with a camera, the camera can watch for badges while nobody is signed in. Operators then just hold their badge up to it.

To turn it on at a tablet:

1. Sign in at the tablet.
2. Tap the gear (⚙, Tablet settings). Operators do not have it.
3. Tick **Webcam badge scanner**. If the tablet has more than one camera, press **Find cameras** and pick one.
4. Tap **Done** and sign out. The camera starts.

The browser asks once for the camera. If the camera fails, the panel says why (no camera, camera blocked for this site, camera in use) with **Try again**; the scanner, NFC and typing keep working. Badges for the camera read best printed as QR codes, at least 2.5 cm square, in even light without glare.

Shipping

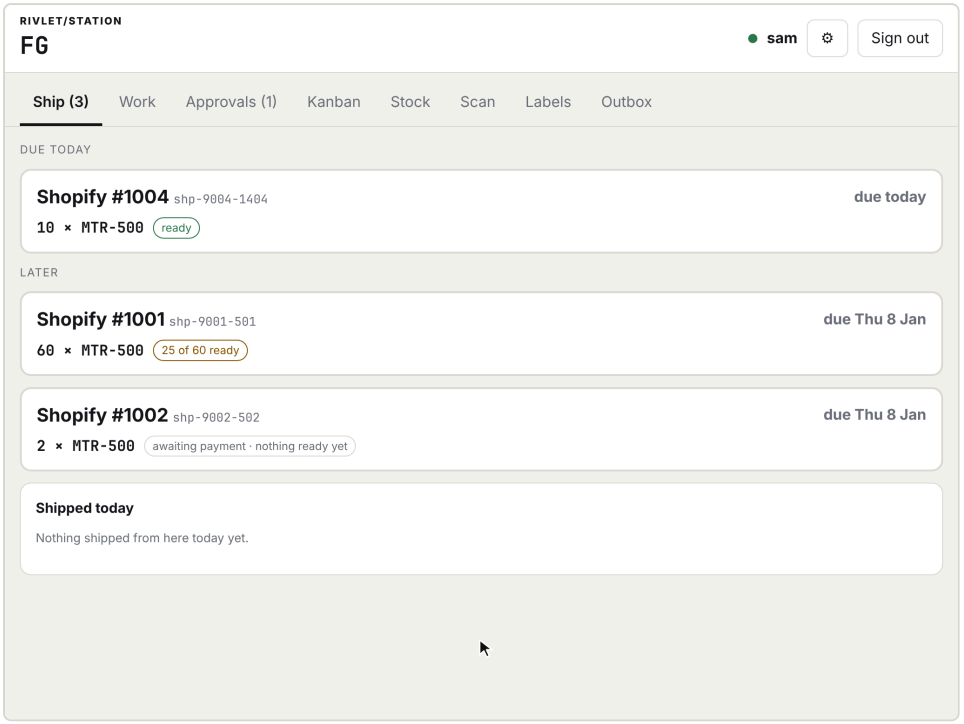
This chapter is for whoever packs and ships orders at finished goods (FG). In the demo that is Sam, the line supervisor.

Before you start

- Orders ship from FG. The FG tablet has a **Ship** tab; other stations do not.
- You need a role that may ship (floor operators, station managers and line supervisors may) and FG must be one of your stations.
- With nobody signed in, the FG tablet says: "Orders ship from FG: sign in with your badge to ship them."
- Each order comes with a label printed in your shop (in Shopify, for example). The label holds the address and the tracking number.

The Ship tab

Sign in at the FG tablet. It opens on **Ship**, with the number of orders to ship, for example **Ship (3)**.



The Ship tab. At FG the Ship tab comes first: each order with its label, what is ready for it, what is in work and its due day; one awaiting payment says so. `floor-ship-list-sam-tablet`

The orders are grouped: **Late** first (in red, marked LATE), then **Due today**, then **Later**. Each order shows:

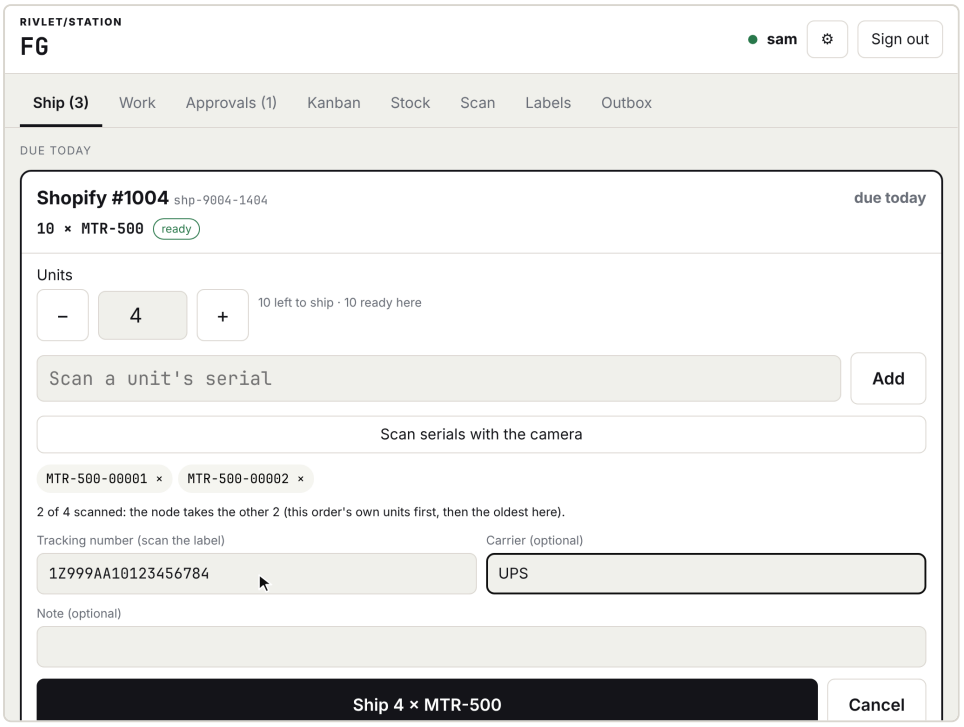
- its label, for example **Shopify #1004**, and its order id;
- what is left to ship, for example "10 × MTR-500";

- how much of it is here for it: "ready", "35 of 60 ready · 25 in work", or "nothing ready yet";
- what shipped so far, for example "4 of 10 shipped";
- "awaiting payment", while the shop has not been paid.

Under the orders, **Shipped today** lists each shipment from FG today, newest first: the time, the order, the units, the carrier and tracking number, and who shipped it.

To ship an order

1. Tap the order. The form starts with as many units as are ready for it.
2. Set **Units** with **+** and **–**, if you ship fewer now.
3. Scan each unit's serial as it goes in the box, in the **Scan a unit's serial** field (or tap **Scan serials with the camera**). Each serial counts once. Scanning the item's own barcode counts one more unit. Units you do not scan are the node's choice: this order's own units first, then the oldest ones here.
4. Scan or type the **Tracking number (scan the label)**. The **Carrier (optional)** and a **Note (optional)** can be added.
5. Tap **Ship N × item**, for example **Ship 4 × MTR-500**. Or **Cancel**.

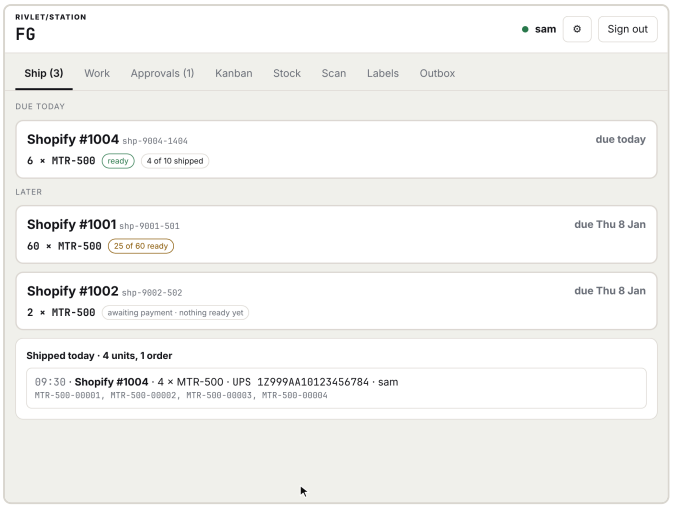


Ship 4 of Shopify #1004. The form starts at what is ready; set the units, scan serials (each once) and the tracking number, then Ship. `floor-ship-form-sam-tablet`

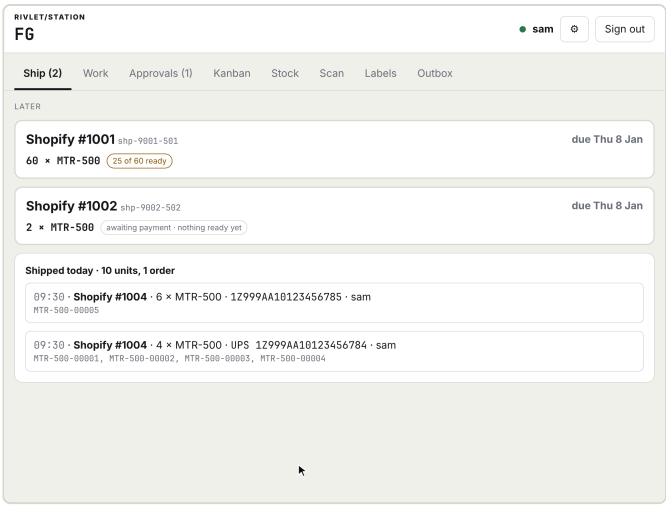
The form shows what will happen, for example "2 of 4 scanned: the node takes the other 2 (this order's own units first, then the oldest here)."

Shipping part of an order

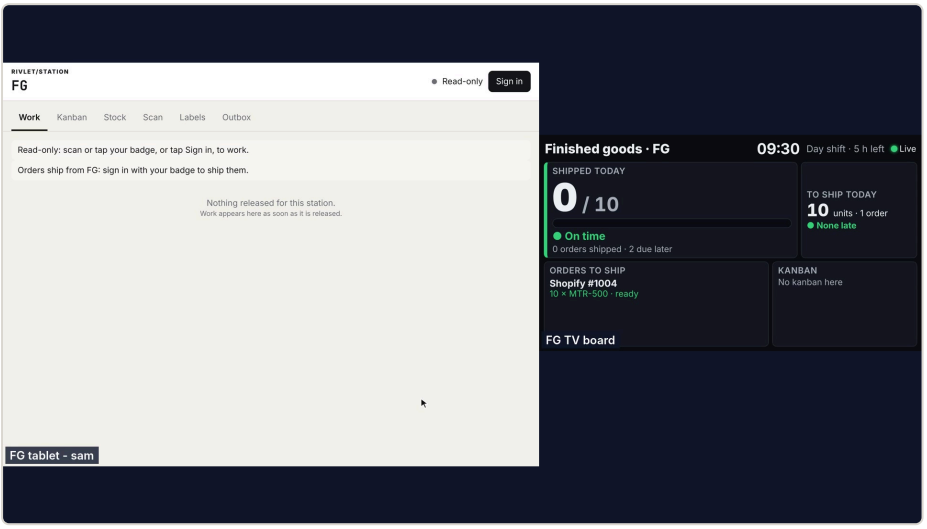
You can ship an order in parts. After a partial shipment, the order stays on the list with what is left ("4 of 10 shipped"), and the shipment is listed under **Shipped today** with its tracking number. Ship the rest later the same way. Shipped in full, the order leaves the list.



A partial shipment. After shipping 4: 4 of 10 shipped, 6 to go, and the shipment under Shipped today with its tracking number. `floor-ship-partial-sam-tablet`



Shipped in full. Shipped in full, the order leaves the list; Shipped today shows both shipments. `floor-ship-done-sam-tablet`



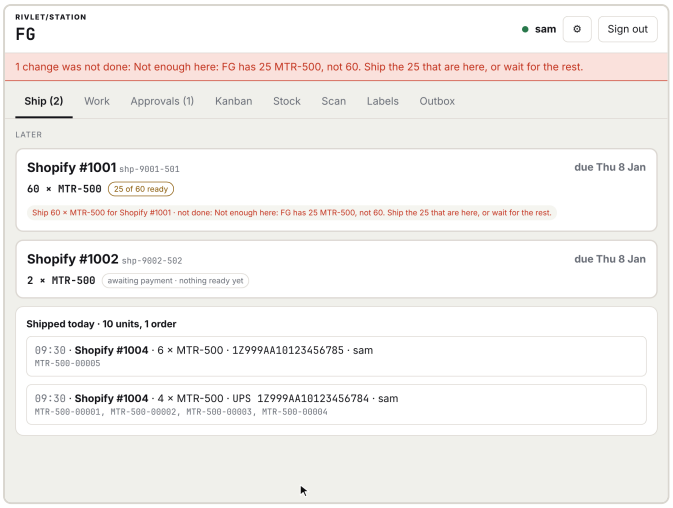
Movie: Ship at FG (1 min 25 s). Sam signs in at the FG tablet and it opens on Ship: Shopify #1004, 10 motors due today, first. He ships 4 now (two serials scanned, the tracking number from the label), then the other 6; the FG TV board's "to ship today" falls from 10 to 0 and "shipped today" climbs to 10. Shipping more than FG holds is refused in plain words. `floor-ship-at-fg`

Chapters: Orders to ship 0:00 · A partial shipment 0:24 · The rest 0:54 · More than is here 1:09

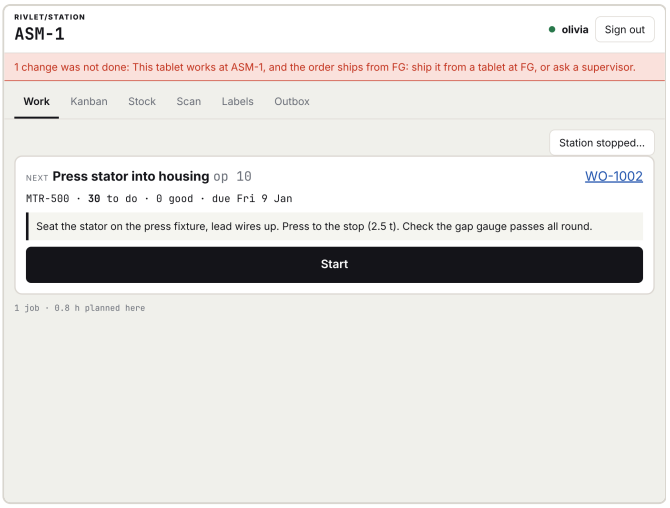
When the node says no

A shipment is all or nothing: if anything is wrong, nothing ships, and the tablet says why on the order and in the bar at the top.

What you read	What to do
"Not enough here: FG has 30 MTR-500, not 55. Ship the 30 that are here, or wait for the rest."	Ship what is here, or wait. Setting more units than are ready is only a warning ("Only 25 ready here"); the node then decides from what FG holds.
"FG has no MTR-500 now: wait until they arrive, or ask a supervisor to check the stock."	Wait, or have the stock checked.
"Only 55 left to ship on this order (some shipped already, or the shop changed it): ship 55 at most."	Lower the units.
"Unit MTR-0995-03 was shipped in shp-9001-501: take it out of the box and scan another unit."	Take that unit out and scan another.
"No unit ... is known: check the label, or leave the serial out and the node picks the unit."	Check the serial label.
"This order is closed (done, or cancelled in the shop): nothing more ships on it."	Do not ship it. Check with the office.
"This tablet works at ASM-1, and the order ships from FG: ship it from a tablet at FG, or ask a supervisor."	Ship from the FG tablet.



More than is here. Shipping more than FG holds is warned, then refused as a whole in plain words; nothing ships. `floor-ship-short-sam-tablet`



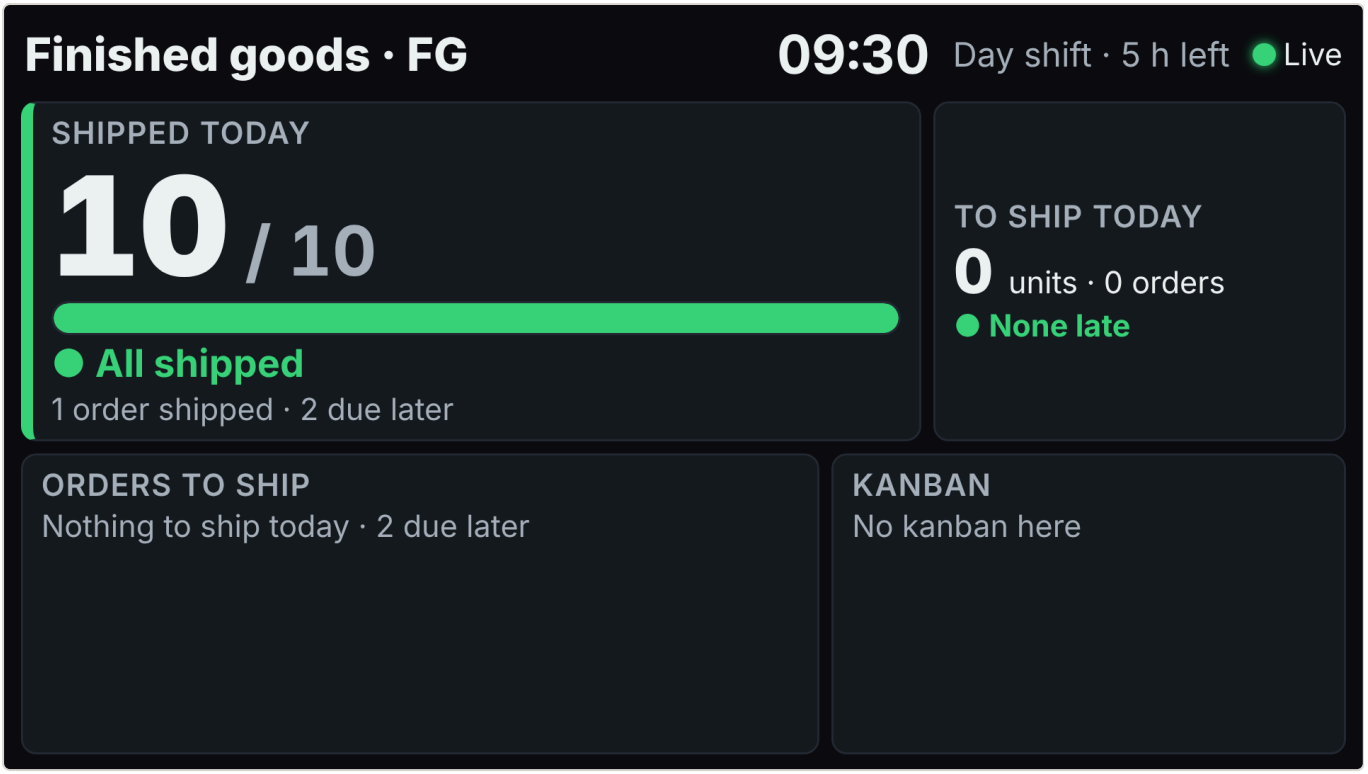
No shipping at ASM-1. ASM-1 has no Ship tab (work, kanban, stock, scan, labels, outbox); a shipment sent from it anyway is refused: the order ships from FG. `floor-ship-refused-asm1-olivia-tablet`

Offline, and shipping twice

Each shipment carries an id made on the tablet. If it is sent twice (a retry, or a dropout), it ships once. Offline, the shipment waits in the **Outbox**, and the order shows what is still sending. If the node already has it, the tablet says "Already shipped: ... was recorded before, so it shipped once."

The FG board

FG's TV board is a shipping board: what is to ship today, what shipped today, and the orders due later. It follows each shipment at once: "to ship today" falls and "shipped today" climbs.

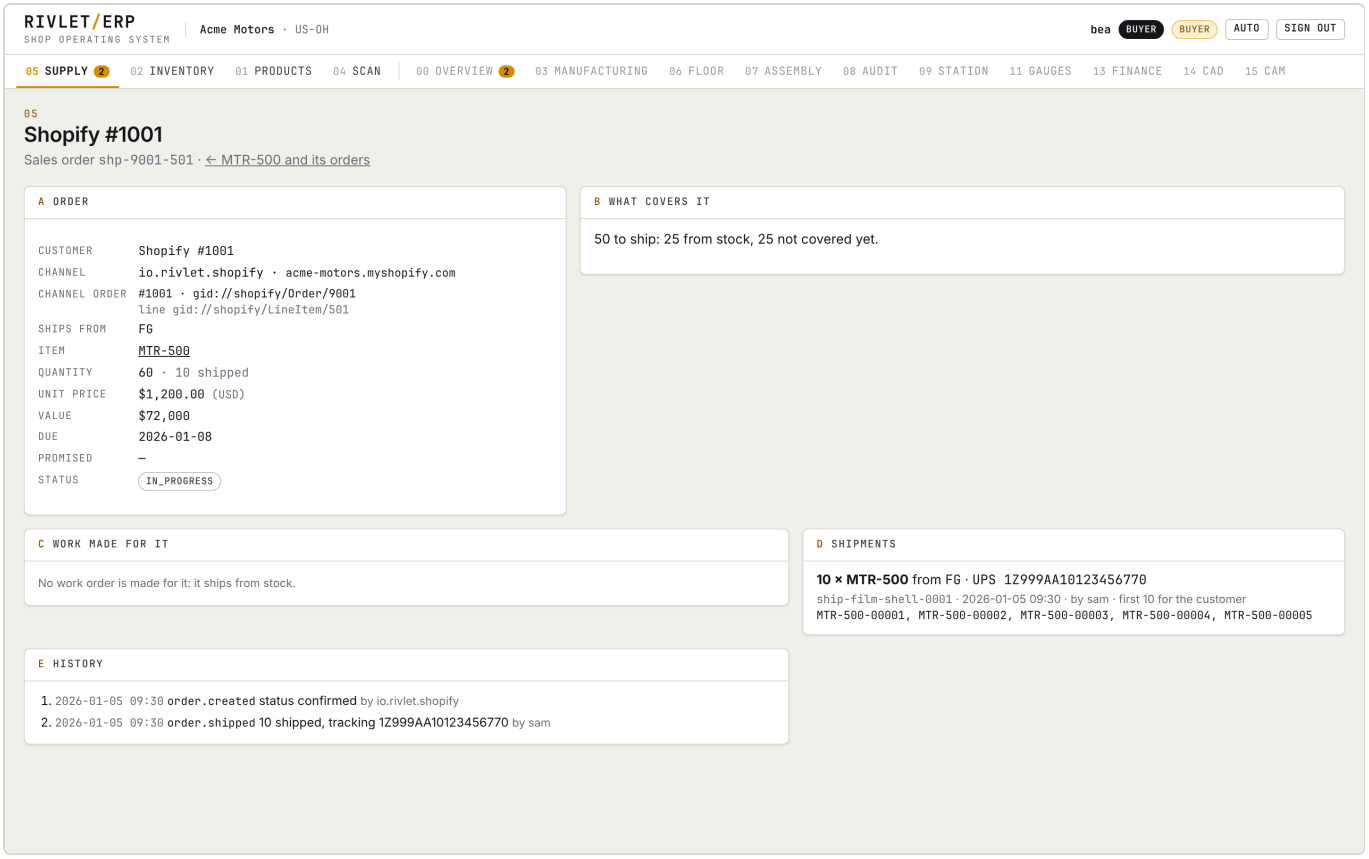


The FG board. FG, where orders ship from: to ship today, shipped today, and what is due later. `floor-tv-board-fg-none-tv`

After the shipment

When an order came from a web shop, its app is told about the shipment and marks the order fulfilled in the shop, with the tracking number. See Selling through Shopify.

In the shell, the order's record lists each shipment: its units, serials, carrier and tracking number, and who shipped it, beside the work made for the order. See Selling through Shopify.



A web order opened. Shopify #1001 as a sales-order record: its channel and channel order, where it ships from (FG), what covers it, the work made for it with its status and station, and each shipment with units, serials, carrier and tracking. `shell-record-sales-order-bea-desktop`

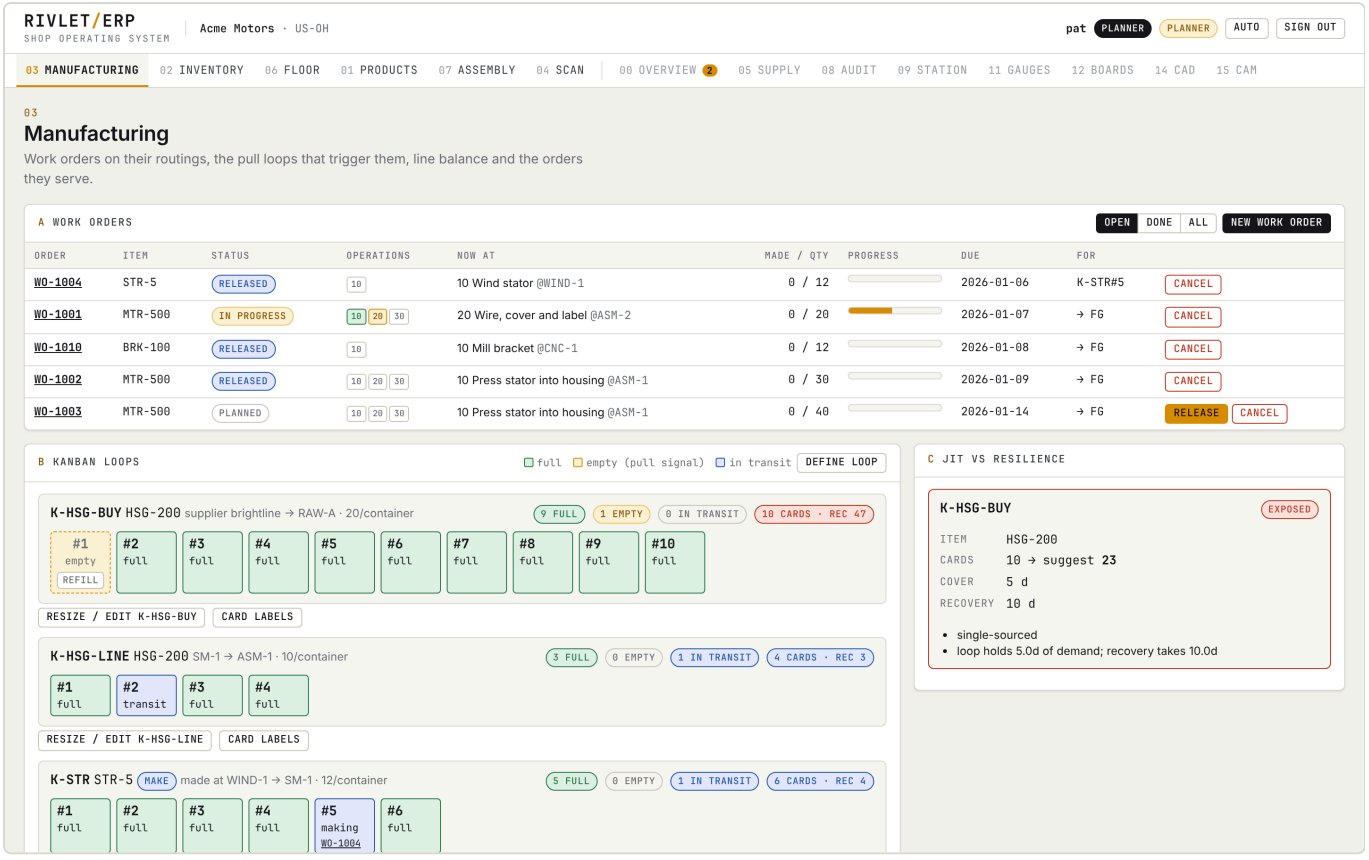
Planning

This chapter is for planners. You release work to the floor, keep kanban loops sized, and approve the work agents propose. In the demo the planner is Pat, who approves up to $10,000.

Your page: 03 Manufacturing

You start on **03 Manufacturing**. It has five panels:

- **A Work orders**: every work order, with a pip per operation and its progress.
- **B Kanban loops**: each loop's cards (full, in use or pulled), with **Pull** and **Receive**.
- **C JIT vs resilience**: how lean each loop is against the risk of a supplier failing.
- **D Line balance**: each station's cycle time against takt.
- **E Open orders**: customer orders still to make or ship.



Manufacturing as Pat. Work orders on their routings, the kanban loops that trigger them, line balance and open orders. Seen by Pat (planner). `shell-manufacturing-page-pat-desktop`

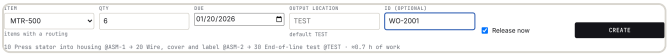
Work orders

To create a work order

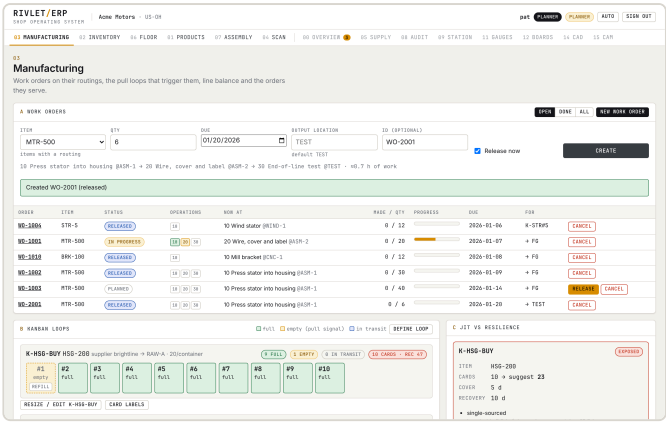
1. On **03 Manufacturing**, press **New work order**.

2. Pick the **Item**. Only items with a routing are offered.
3. Enter the **Qty** and the **Due** date.
4. Check the **Output location**. It defaults to the routing's last work centre (FG for motors).
5. Give it an **Id (optional)**, or leave it blank and one is made for you (WO-...).
6. Tick **Release now** to send it to the floor at once. Leave it clear to keep it as a draft and release it later.
7. Press **Create**.

The form shows the route and the estimated hours of work before you create it.



New work order. A new work order: the item (only routed items are offered), quantity, due date, output location and an id; Release now sends it to the floor. `shell-manufacturing-new-wo-form-pat-desktop`



Work order released. WO-2001 joins the list released, with a pip per operation and its progress. `shell-manufacturing-wo-created-pat-desktop`

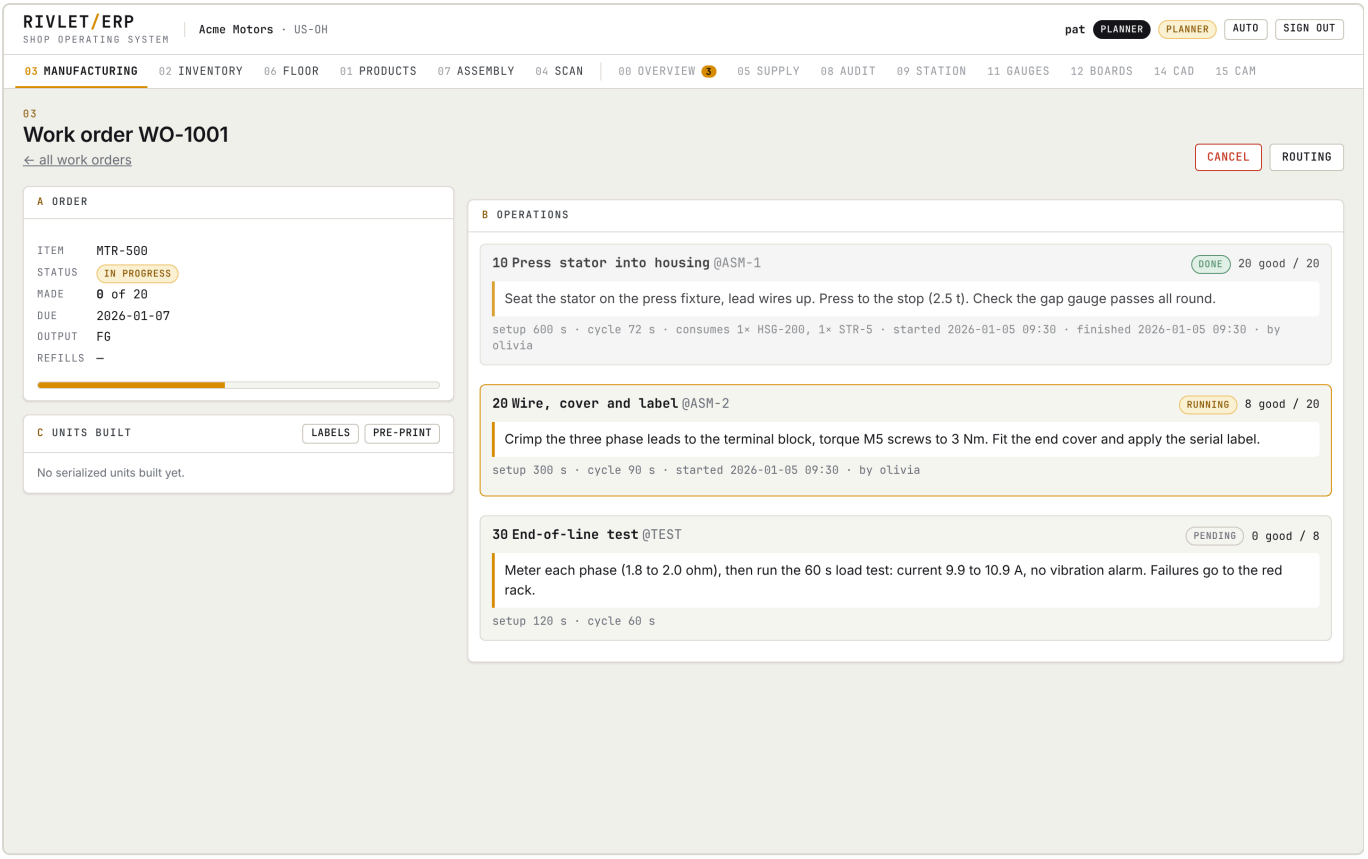
An id that is already used is refused under the form, for example "409 Conflict: order WO-2001 already exists". Choose another id, or leave it blank.



Duplicate id refused. Reusing a work-order id is refused, with the reason under the form. `shell-manufacturing-wo-duplicate-pat-desktop`

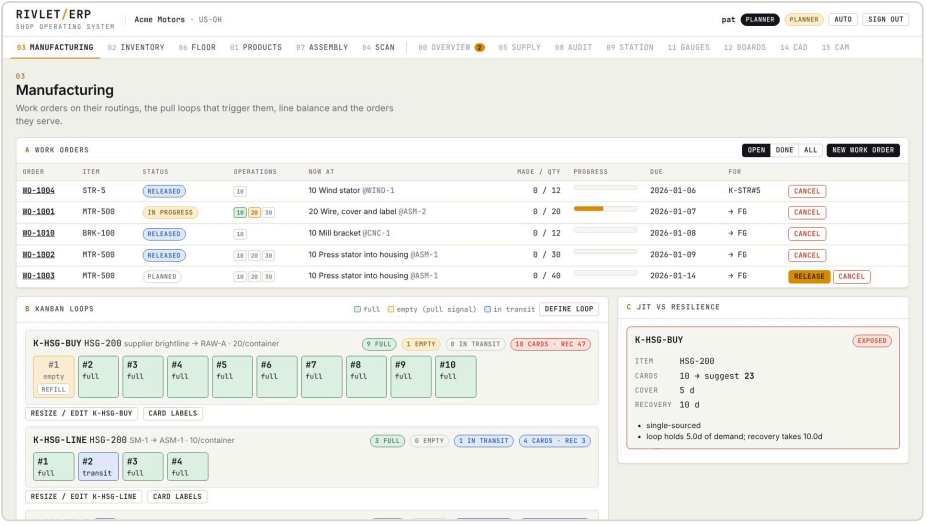
To follow a work order

Click a work order to open it. You see each operation with its station, status, good and scrap counts, who did it and the instructions. From here you can **Release** a draft, **Cancel** an order, or open its **Routing**.



Work order detail. A work order opened: each operation with its station, status, good and scrap counts, who did it and the instructions. `shell-manufacturing-wo-detail-pat-desktop`

A released order appears at once in the queue of the station that does its first operation. You can find it on **06 Floor**, in that station's queue.



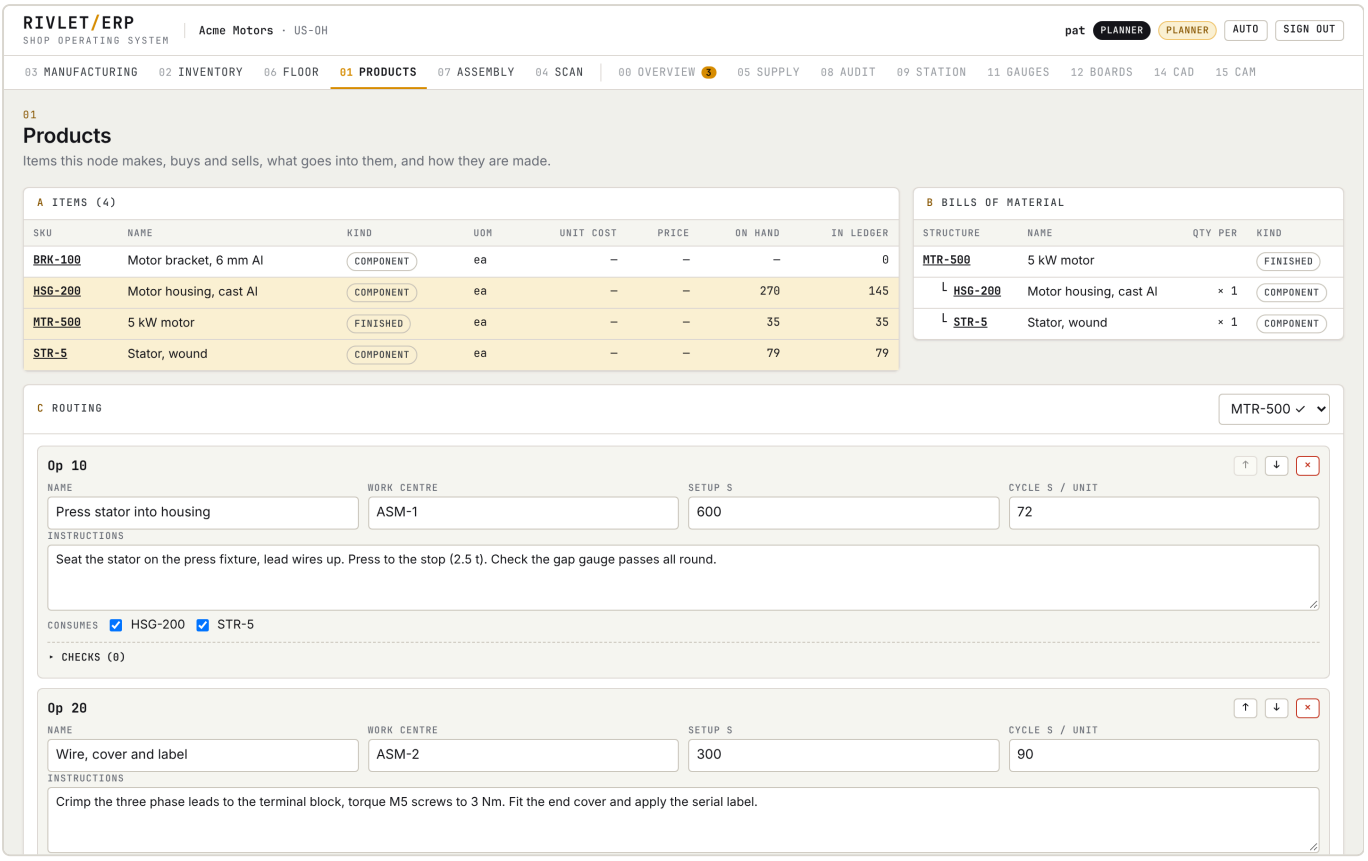
Movie: Plan the work (1 min 2 s). Pat creates and releases a work order, opens it, checks the item routing and the kanban loops, and finds the order in the station queue on the live floor. `shell-plan-the-work`

Chapters: New work order 0:00 · Open the order 0:31 · Routing 0:37 · Kanban loops 0:45 · On the floor 0:52

On a phone, the work-order table keeps each id, item code, station and date on one line, and scrolls sideways in its own frame.

Routings

A routing is the list of operations that make an item: each operation's station, setup and cycle times, instructions and checks. It lives with the item, on **01 Products**. Planners and engineers may edit it.



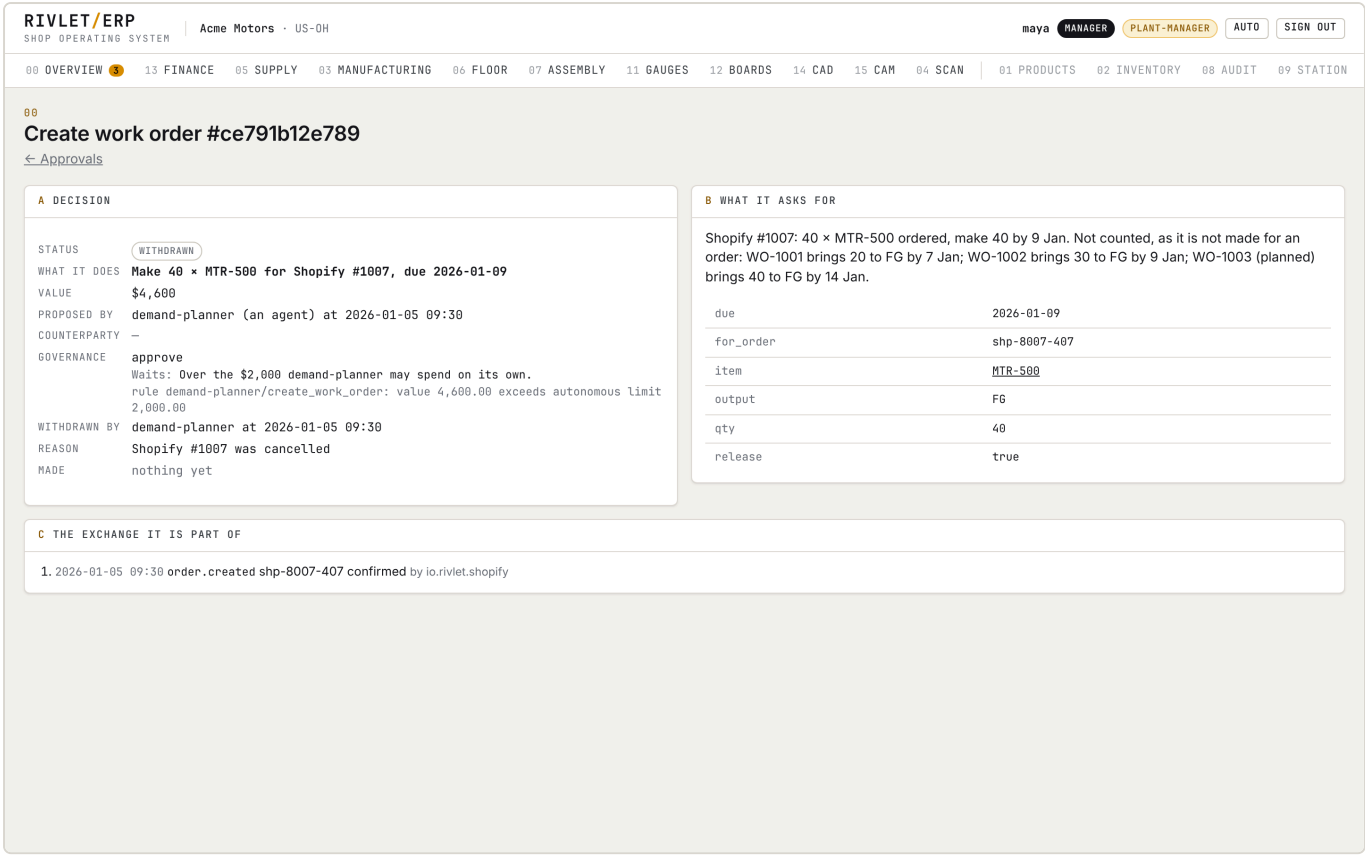
Item routing. An item's routing on Products: MTR-500, operation by operation, with the station, cycle time and checks of each. `shell-products-routing-mtr500-pat-desktop`

The demand planner

The demand planner is an agent. It turns customer orders into work. When an order cannot be covered from stock, it proposes a work order for the rest (or a purchase order, for a bought item), made for that order and due when the order is.

- It only plans for paid orders. An order awaiting payment holds its stock, but nothing is made for it yet.
- Work made for an order is labelled with it, for example "Shopify #1001", on the station queues and boards.
- If an order is cut or cancelled, it cancels unstarted work nobody needs. Work already started is left for people to decide.
- What a person declined, it does not ask again until the need changes.

- What it proposed and is still waiting for a person, it takes back when the need is gone: the order was cancelled or shipped in full, or now needs less than the proposal would make. The proposal shows as withdrawn with its reason, for example "Shopify #1002 was cancelled", and a proposal for the right amount takes its place.



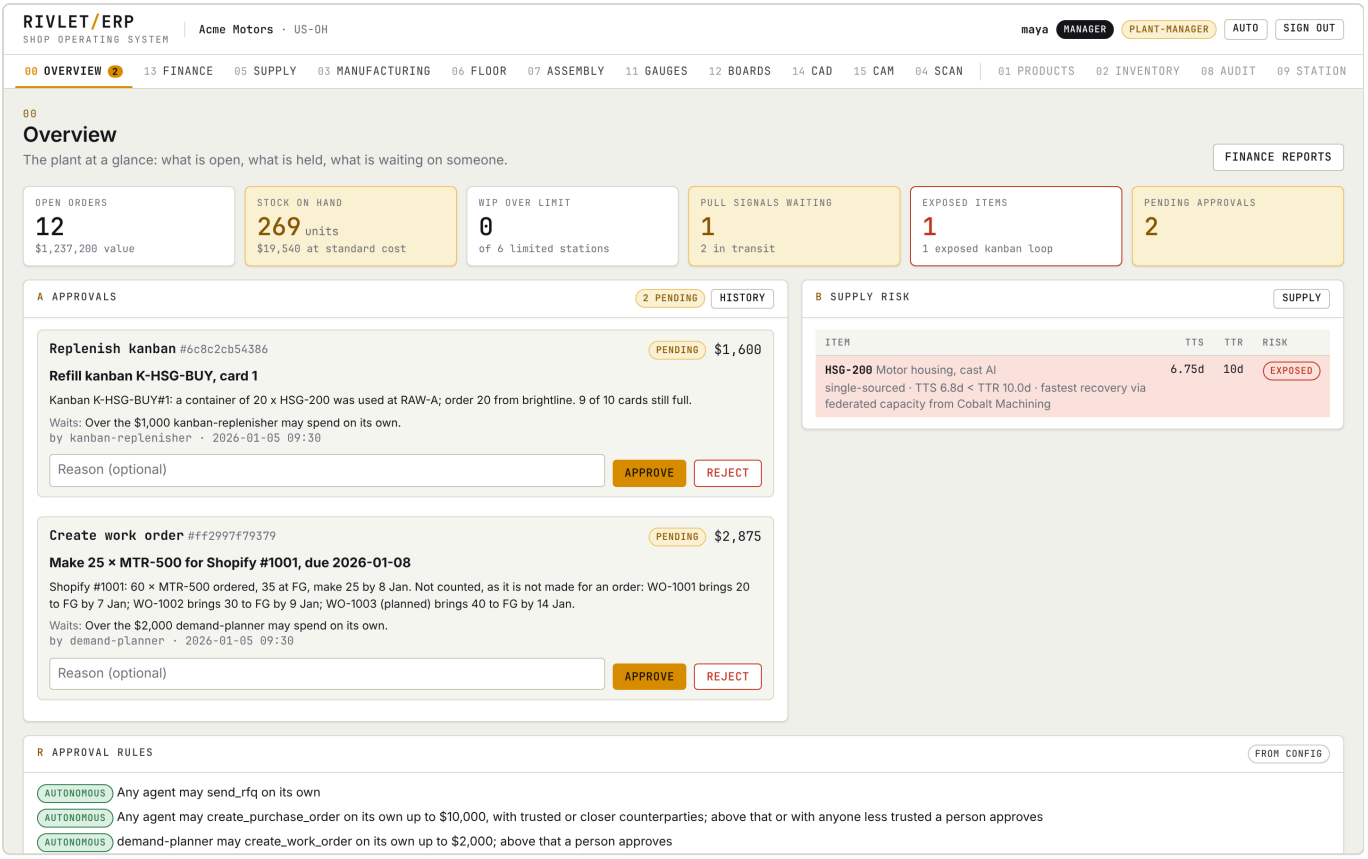
A withdrawn proposal. The planner's withdrawn proposal as a record: what it would have made, for which order, who withdrew it (the demand planner) and why. `shell-record-proposal-withdrawn-maya-desktop`

The plant's rules decide what it may do alone. In the demo it may make work on its own up to $2,000 and never buys without a person. A work order is worth its standard cost.

To approve the planner's work

1. Open **00 Overview** or **05 Supply**. The **Approvals** panel lists what waits.
2. The card says what the planner saw, for example "Shopify #1001: 60 × MTR-500 ordered, 35 at FG, make 25 by 8 Jan", and its value ($2,875).
3. Type a reason and press **Approve** (or **Reject**).

The work order is released at once, labelled with the order it is for.



Overview: waiting for a decision. The plant manager's overview leads with what waits on a person: a $1,600 kanban refill from Brightline over the agent's $1,000 limit, and the demand planner's work order for a web order over its $2,000. `network-overview-approvals-maya-desktop`

The whole story, from a web order to shipping it, is in Selling through Shopify.

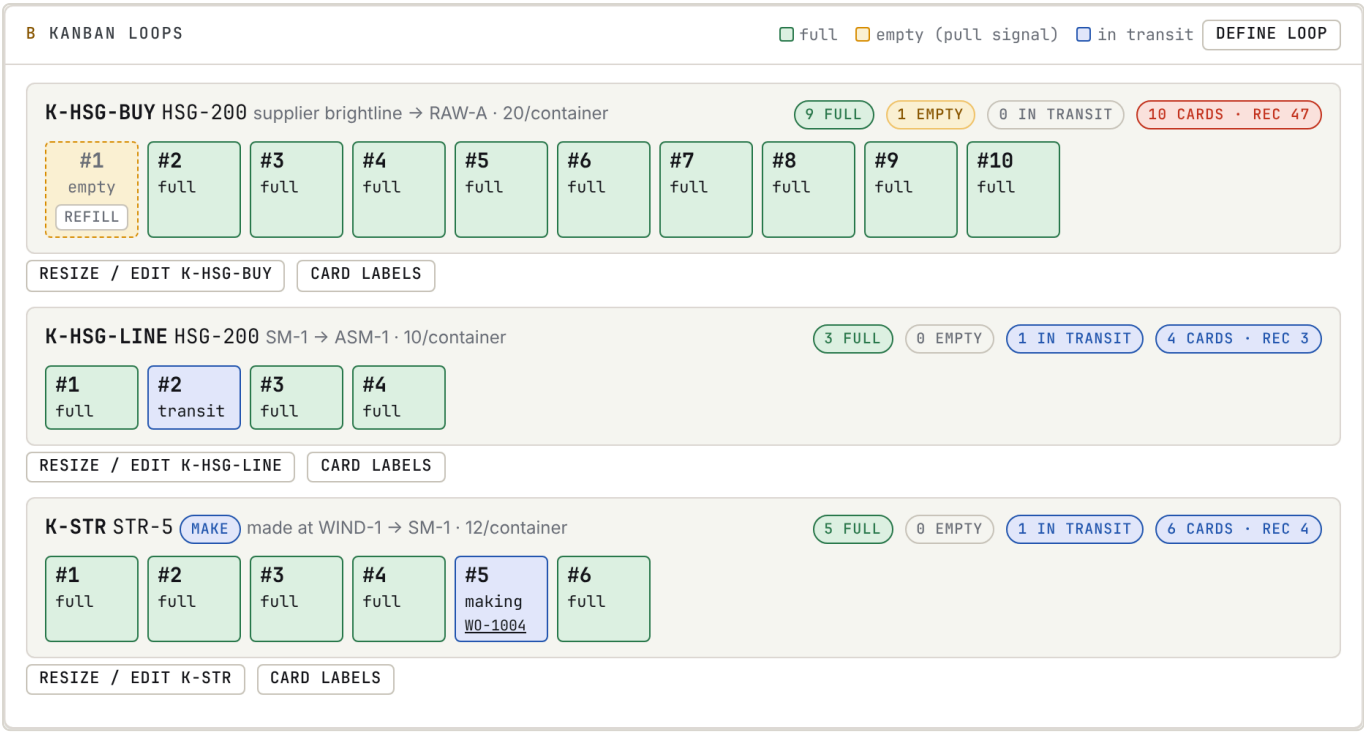
Kanban loops

A kanban loop keeps a part flowing to the place that uses it. Each loop has a number of cards; each card is one container. When a container is opened, the operator pulls its card; the empty card is the signal to refill.

Where the refill comes from depends on the loop's supply:

- **Location**: a move from another location, such as the supermarket SM-1.
- **Supplier**: a purchase order to a supplier.
- **Make (work order)**: a work order.

The kanban agent answers each pull. A refill under its limit ($1,000 in the demo) runs on its own; above that it waits for a person.

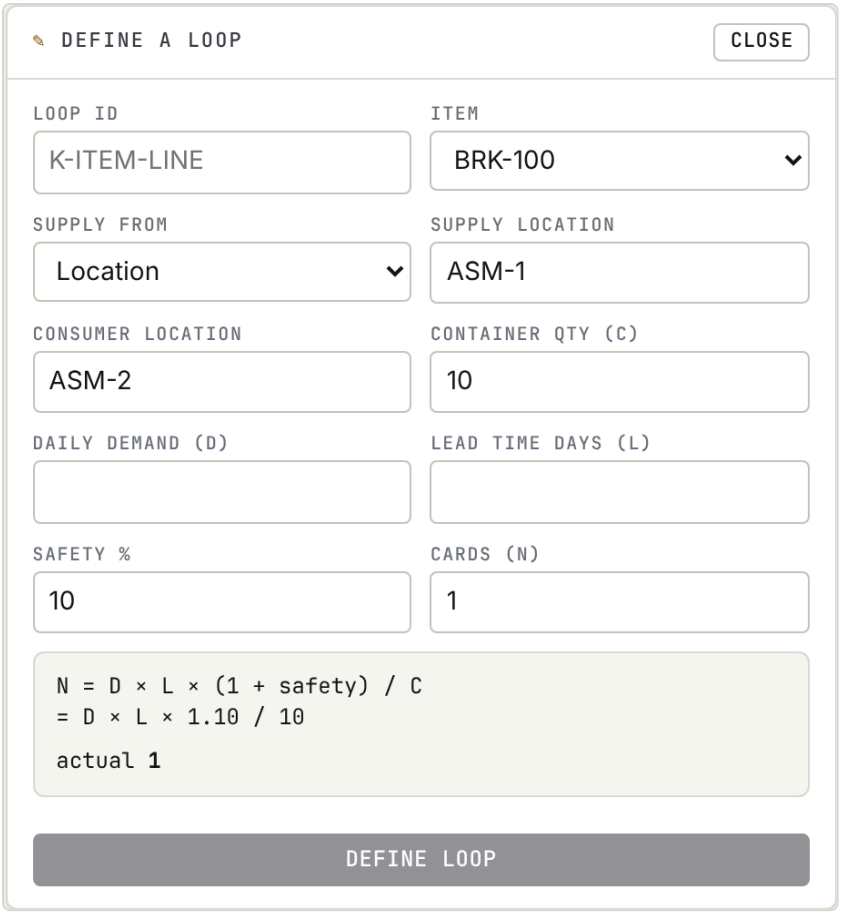


Kanban loops. Kanban loops: each card is full, in use or pulled, with Pull and Receive actions. `shell-manufacturing-kanban-pat-desktop`

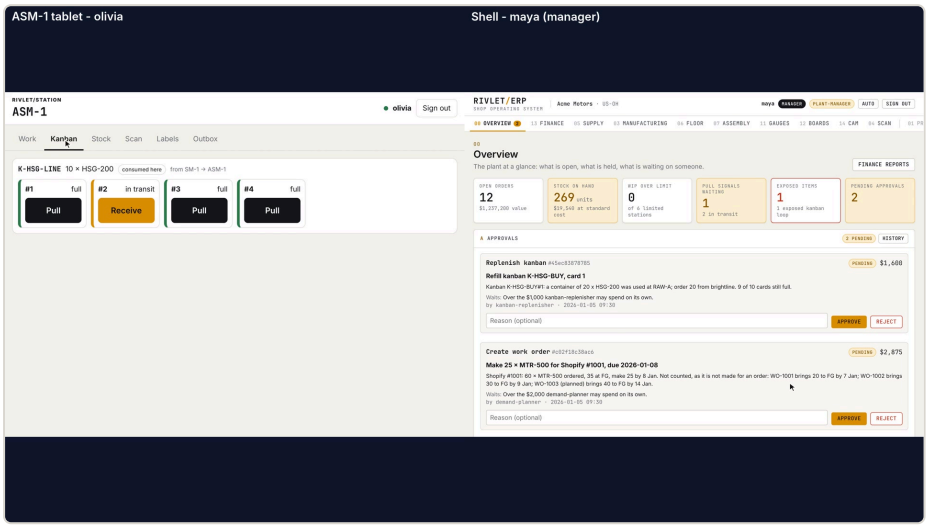
To define or resize a loop

1. In **B Kanban loops**, press **Define loop**, or **Resize / edit** on a loop.
2. Fill in the **Loop id**, the **Item**, where it is supplied from (**Supply kind**, then the supply location or supplier, or what makes it) and the **Consumer location**.
3. Enter the **Container qty (C)**, the **Daily demand (D)**, the **Lead time days (L)** and the **Safety %**.
4. The number of **Cards (N)** is worked out for you.
5. Press **Define loop** (or **Save loop**).

Card labels prints the QR labels for a loop's cards.



Define a kanban loop. Defining a kanban loop: item, supply, consumer, container size, demand and lead time; the card count is worked out for you. `shell-manufacturing-loop-form-pat-desktop`



Movie: The kanban loop (40 s). A pull at ASM-1 empties a card; the kanban-replenisher proposes a refill from the supermarket, which runs on its own (under $1,000); a bought refill of housings from Brightline ($1,600) waits for maya to approve; the line card is received back at ASM-1. `floor-kanban-loop`

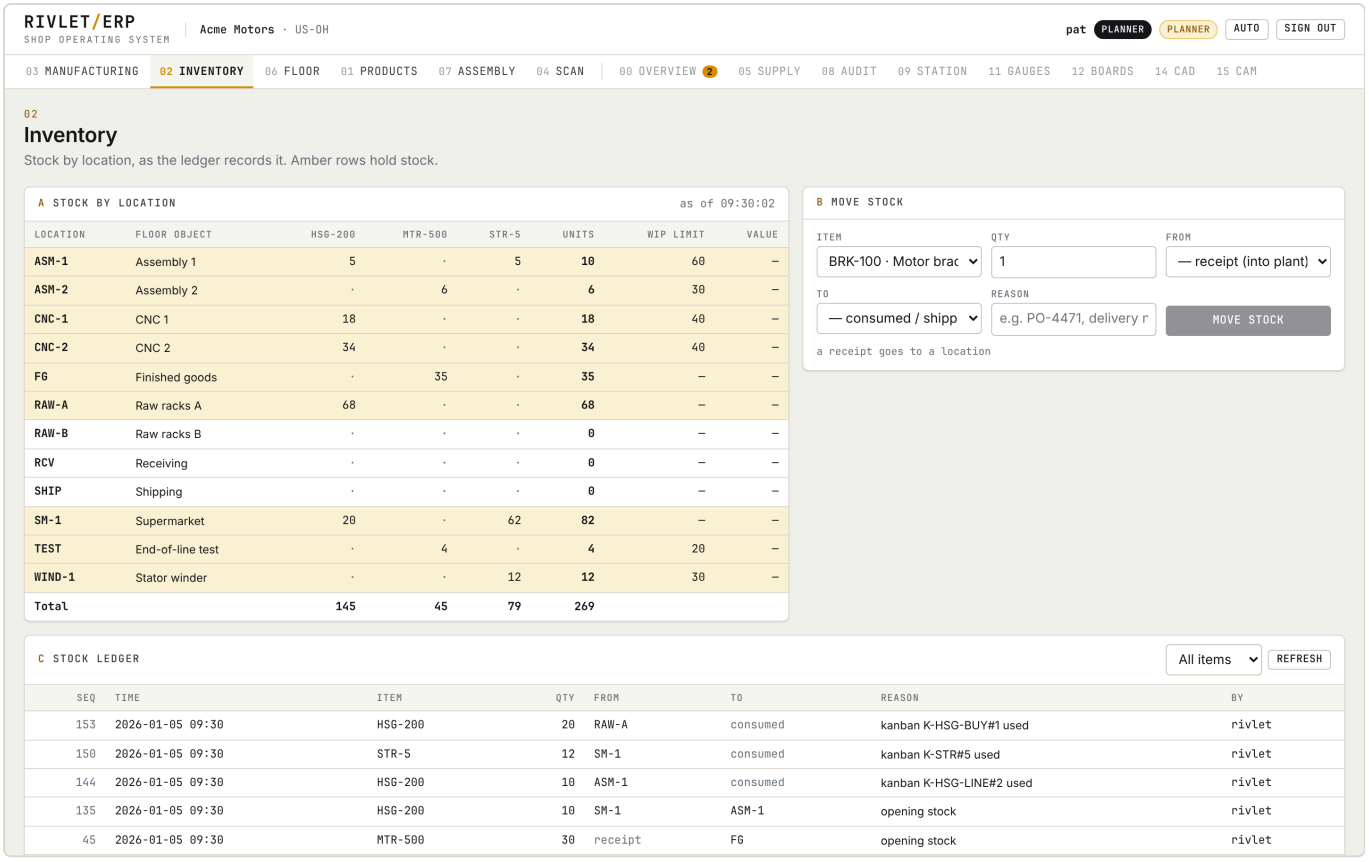
Chapters: Pull a card 0:00 · The refill runs on its own 0:09 · A bought refill waits for approval 0:14 · Receive the card 0:31

Stock

02 Inventory shows stock by location from the stock ledger, a form to move stock, and the ledger itself.

Planners and buyers may also book goods in from outside the plant (a receipt). To book goods in:

1. In the move form, pick the **Item** and the **Qty**.
2. Under **From**, pick **— receipt (into plant) —**.
3. Under **To**, pick where the goods go.
4. Write the **Reason**: the PO or delivery note, or why the goods are here. A receipt needs one.
5. Press **Move stock**.



Inventory as Pat. Stock by location from the ledger, a stock-move form and the ledger itself. Seen by Pat (planner). `shell-inventory-page-pat-desktop`

Approving

You approve up to $10,000: kanban refills, the demand planner's work and other proposals. See Managers and the CFO for how approvals work.

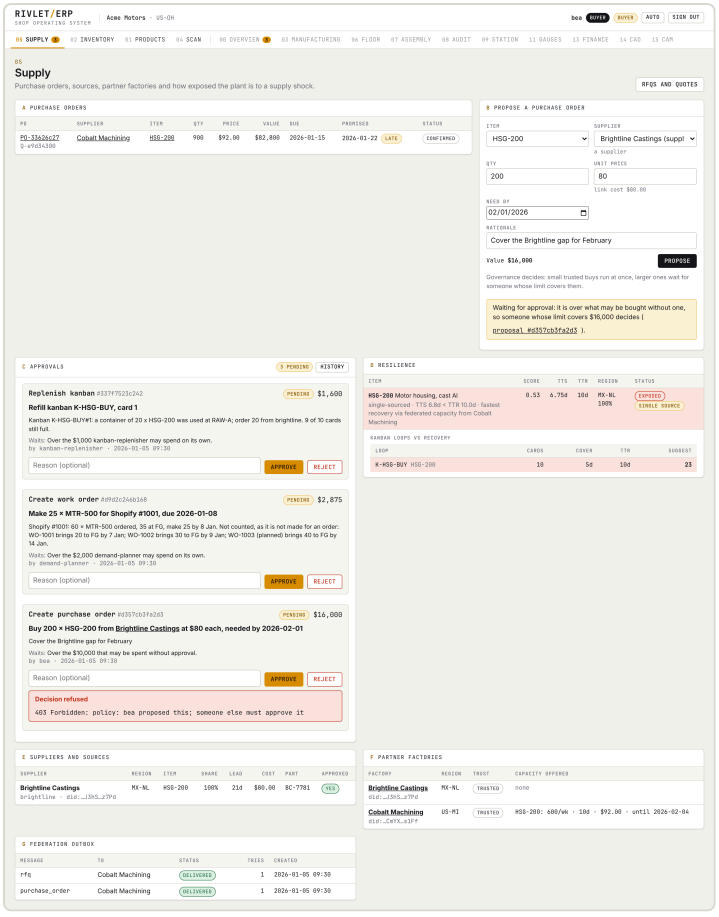
Buying

This chapter is for buyers. You buy parts, follow quotes from partner factories, and watch supply risk. In the demo the buyer is Bea, who approves up to $25,000.

Your page: 05 Supply

You start on **05 Supply**. It has these panels, and the link **RFQs and quotes** at the top:

- **A Purchase orders**: every purchase order (PO), with its supplier, status and dates.
- **B Propose a purchase order**: the form to buy.
- **C Approvals**: what waits for a decision, and its history.
- **D Resilience**: which kanban loops are exposed if a supplier cannot deliver, and for how long.
- **E Suppliers and sources**: suppliers and the items each one sources.
- **F Partner factories**: the partner nodes you trade with, their trust and the capacity they offered.
- **G Federation outbox**: the signed messages sent to partners, and whether each was delivered.

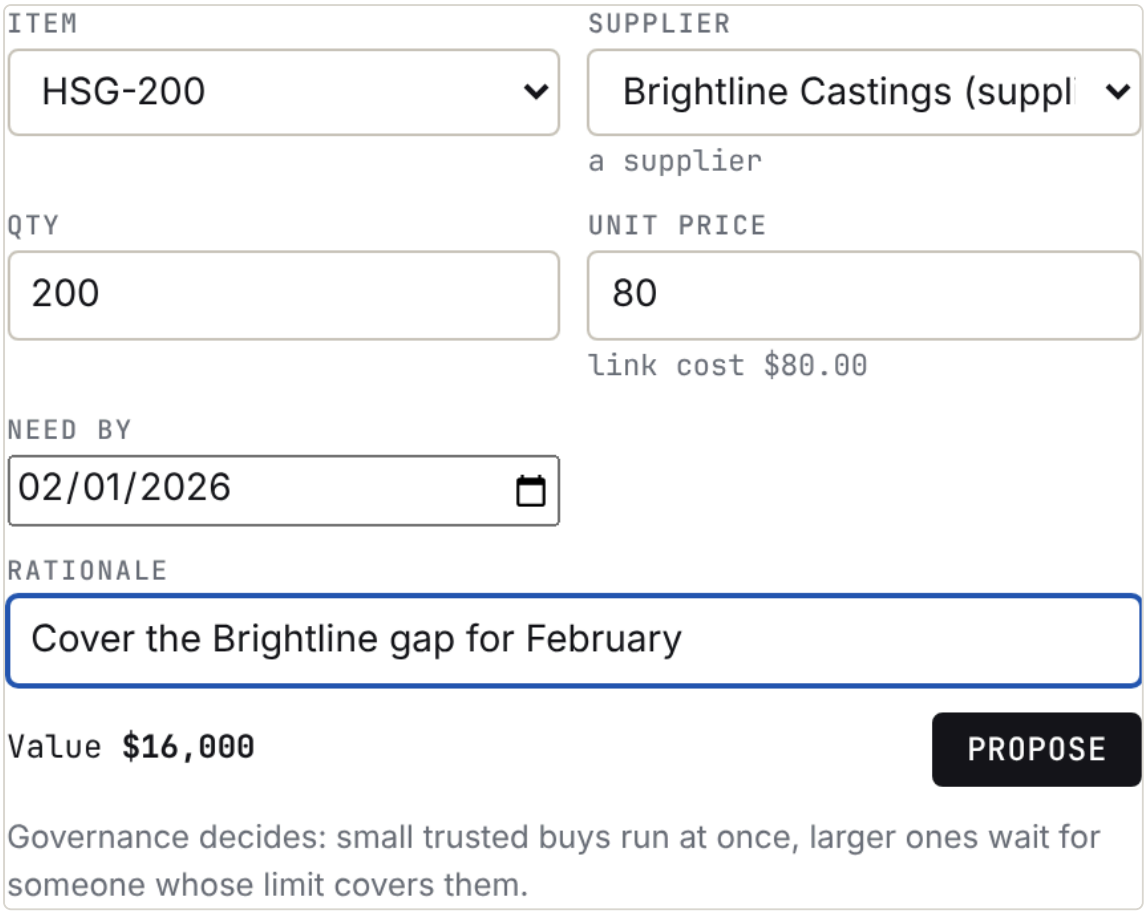


Supply, whole page. The whole Supply page, with the link to RFQs and quotes at the top. `shell-supply-full-bea-desktop`

Purchase orders

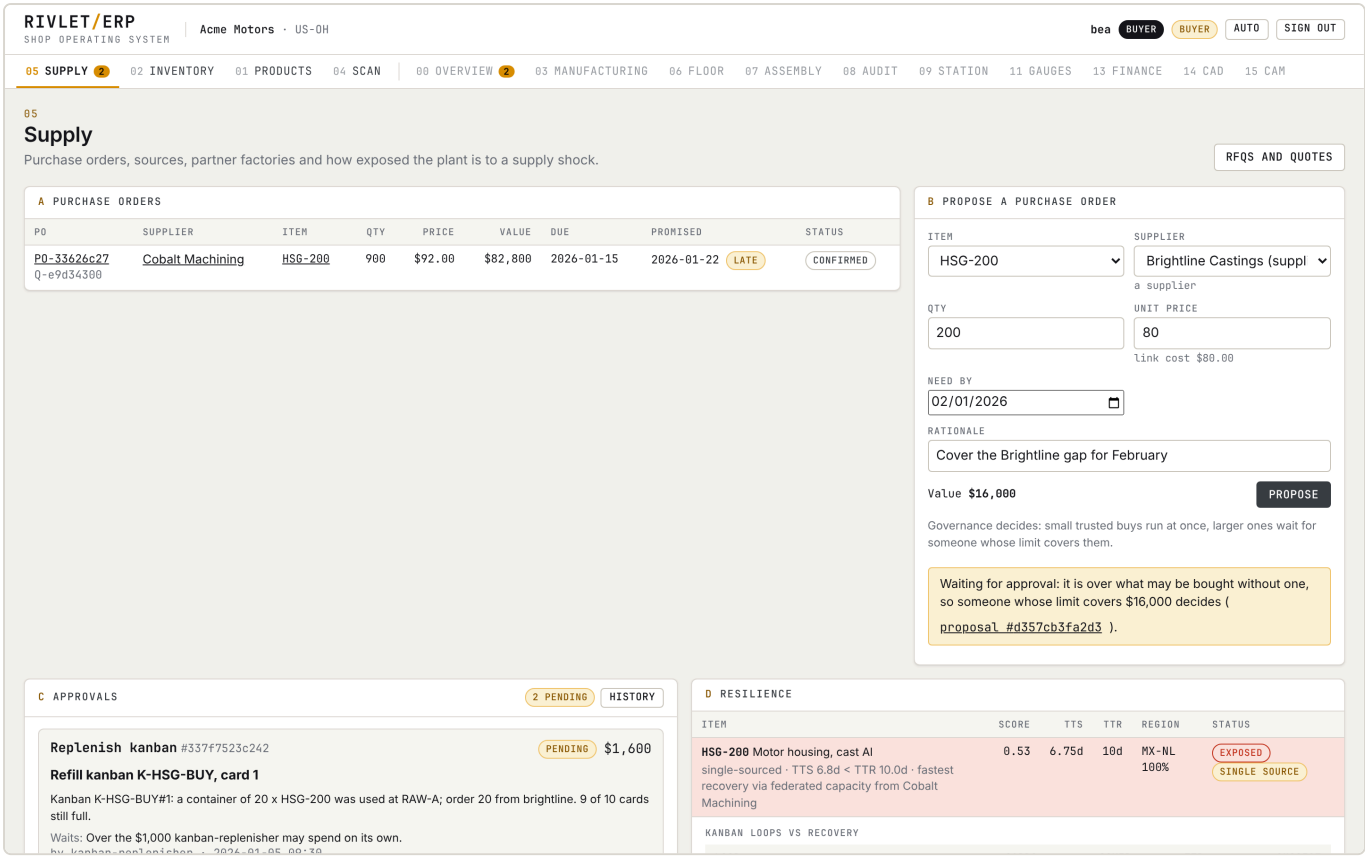
To propose a purchase order

1. In **B Propose a purchase order**, pick the **Item**.
2. Pick the **Supplier**: the item's supplier, or a partner factory that offered capacity for it (with its offered price). The **Unit price** fills in.
3. Enter the **Qty** and the **Need by** date.
4. Write a **Rationale**. The approver reads it.
5. Check the **Value**, then press **Propose**.



Propose a purchase order. Bea picks the supplier (the item's supplier, or the partner that offered capacity), the quantity and need-by date; the form shows the order value before she proposes it.
`shell-supply-po-form-filled-bea-desktop`

The plant's rules decide what happens next: small buys from trusted suppliers may run at once; larger ones wait for someone whose limit covers them. A purchase you propose yourself always waits for someone else. The form then says so in amber: "Waiting for approval: it is over what may be bought without one, so someone whose limit covers $16,000 decides (proposal #…)".



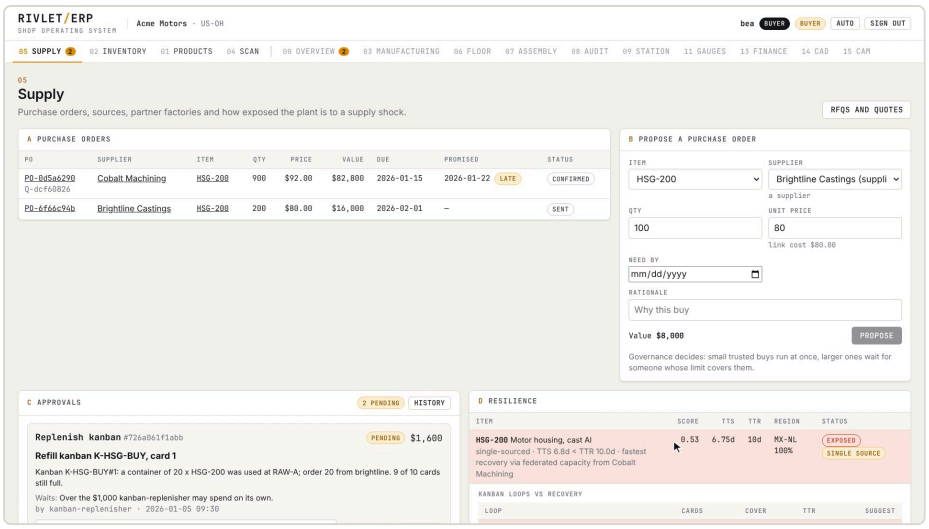
Waiting for approval. A person's own proposal always waits for someone else: the form says so in amber, with the value and a link to the approvals queue. `shell-supply-po-pending-bea-desktop`

You cannot approve your own proposal, even within your limit. The node refuses: "Decision refused · 403 Forbidden: policy: bea proposed this; someone else must approve it".



You cannot approve your own. Four eyes: when Bea tries to approve her own proposal the node refuses and says why. `shell-approvals-own-refused-bea-desktop`

When someone approves it, the PO is placed (and, to a partner factory, signed and sent). The approvals history shows who decided and why, and links to the PO it made.



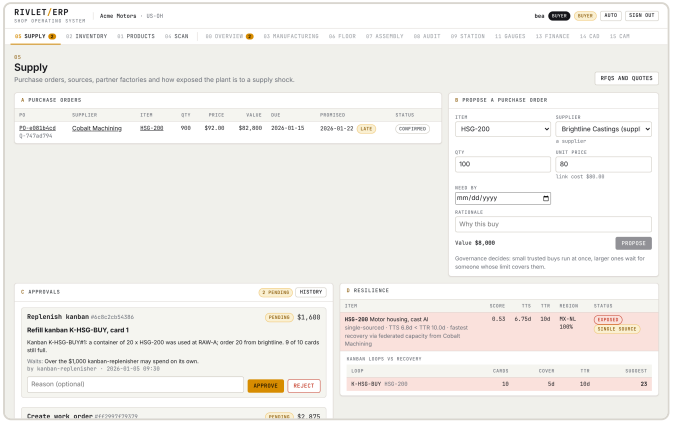
Movie: Propose and approve a purchase (1 min 35 s). Bea proposes a $16,000 purchase order; it waits for someone else; Bea cannot approve her own; Maya approves it with a reason; the history shows who decided and why, opens the PO it made, and shows work the demand planner withdrew when its web order was cancelled. `shell-approve-a-purchase`

Chapters: Propose 0:00 · It waits 0:28 · Not your own 0:34 · Approve 0:45 · History 1:10

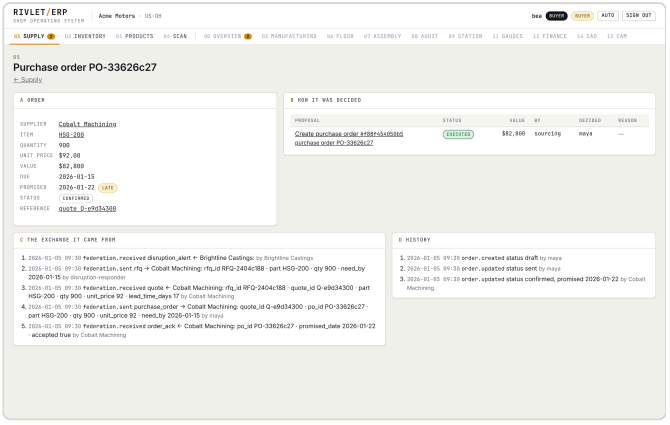
To open a purchase order

Click a PO's row in **A Purchase orders**. Its record shows:

- the supplier, item, quantity, price, and the due and promised dates;
- **How it was decided**: the proposal, who decided, and their reason;
- **The exchange it came from**: for a partner, each signed message (alert, RFQ, quote, PO, acknowledgement);
- its history.



Supply: purchase orders. The purchase order the scenario placed with Cobalt Machining, confirmed with Cobalt's promised date. `network-supply-pos-bea-desktop`

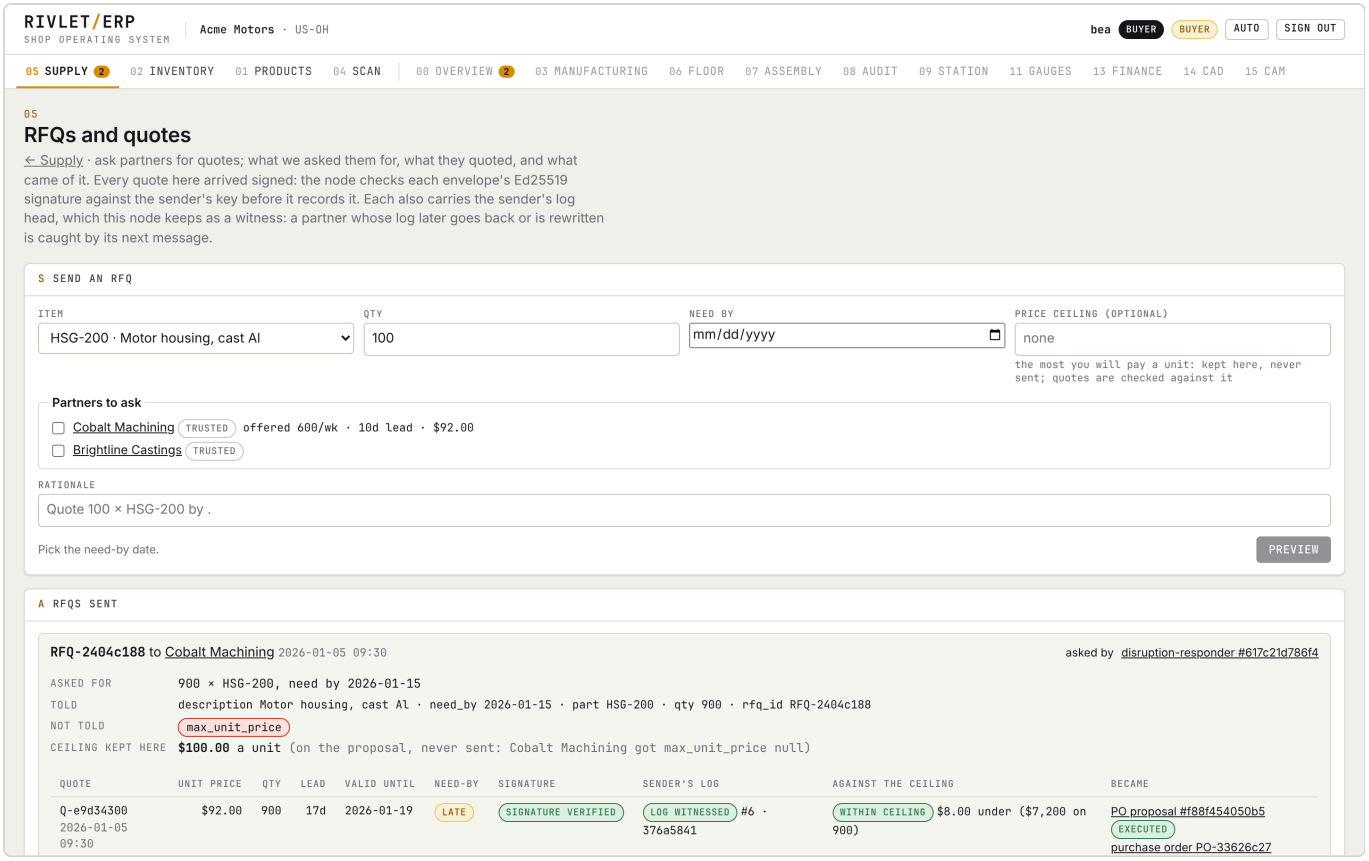


A purchase order opened. PO-33626c27 to Cobalt: the order, the proposal that made it, the exchange it came from (alert, RFQ, quote, PO, acknowledgement) and its history. `shell-record-po-bea-desktop`

Purchase orders made by approvals and agents get ids like PO-9d03c0a7.

RFQs and quotes

05 Supply › RFQs and quotes is where you ask partners for quotes (**S Send an RFQ**), and see what Acme asked partners for, what they quoted, and what came of it.



RFQs and quotes. Each RFQ sent: to whom, what it was not told (max_unit_price) and the ceiling kept here; each quote with its signature checked, the sender's log witnessed here, within or over the ceiling and by how much, and the PO proposal it became. `shell-quotes-bea-desktop`

A RFQs sent. For each request for quotation (RFQ):

- to whom, and what it **Asked for**;
- what it **Told** the partner, and what it did **Not told**. By default an RFQ never tells a partner the most you will pay (`max_unit_price`);
- the **Ceiling kept here**: your price ceiling, kept at your node and never sent.

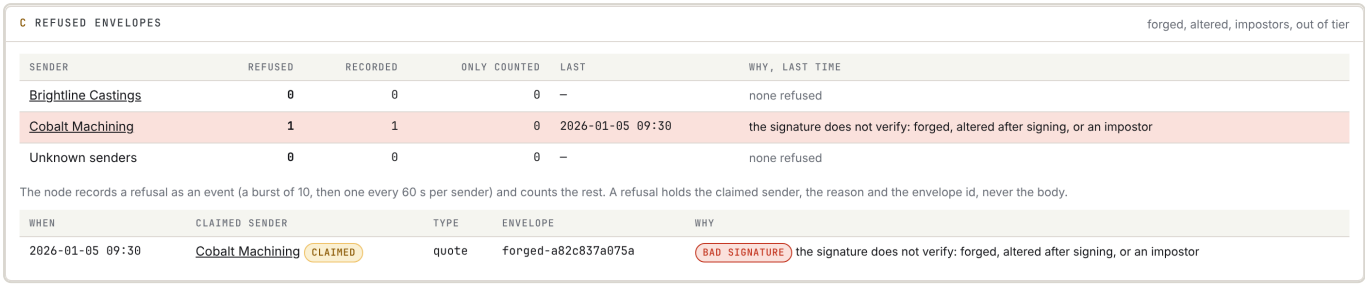
Under each RFQ, each quote:

- its unit price, quantity, lead time, valid until, and need-by (marked LATE if it cannot meet the date);
- **Signature verified**: the quote really came from that partner and was not changed;
- **Sender's log**: the head of the partner's own log when it signed, which your node keeps as a witness. "log witnessed #57" says the partner's log only grew since the heads it signed before; "log went back" or "log rewritten" in red says it was cut short or changed. Hover over it for the words. A partner's record page says how many of its heads are kept;
- **Against the ceiling**: within it, or over it, and by how much ("$8.00 under ($7,200 on 900)");
- what it **Became**: the PO proposal, and the PO.

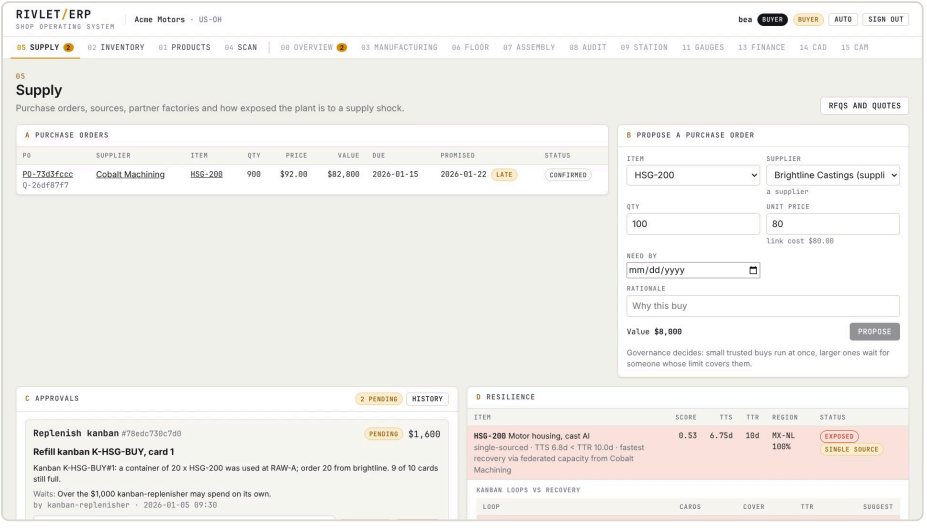
C Refused envelopes. Messages that claimed to come from a partner but failed the checks: a forged or changed signature, a stranger, or a message type the partner's tier does not allow. For each partner, and for unknown senders: how many were refused, the last time, and why. Only the sender, the reason and the message id are kept, never its contents.



The sender's log, witnessed. The "Sender's log" column: each quote carries the head of its sender's hash-chained log, which this node keeps; "log witnessed #n" says the partner's log only grew, and a log that went back or was rewritten would say so here. `shell-quotes-witness-bea-desktop`



Refused envelopes. A quote in Cobalt's name whose price was changed after signing is refused (bad signature), counted on Cobalt's row and listed with the reason, never its body. `shell-quotes-refused-bea-desktop`

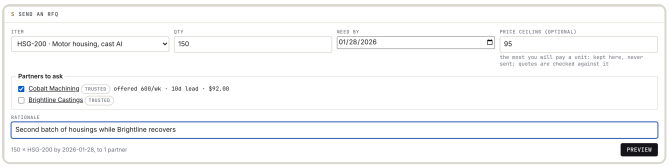


Movie: Quotes and records (1 min 3 s). Bea opens RFQs and quotes: the RFQ to Cobalt, what it was not told and the ceiling kept here, Cobalt's signed quote within the ceiling, and the PO it became; the PO opens as a record with the exchange it came from; then the partner, and the item with its sources. `shell-records-and-quotes`

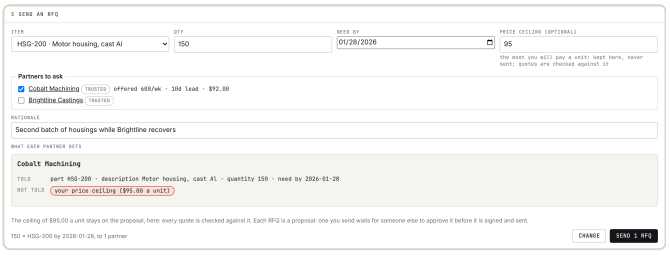
Chapters: RFQs and quotes 0:00 · The purchase order 0:28 · The partner 0:45 · The item 0:52

To send an RFQ

1. In **S Send an RFQ**, pick the **Item**, and enter the **Qty** and the **Need by** date.
2. Under **Partners to ask**, tick the partners. Only partners your plant sends RFQs to are offered (the others are listed under them, with why). Those that offered capacity for the item come first, with their offer ("offered 600/wk · 10d lead · $92.00"). An unknown partner may be asked, but its quotes are refused until your administrator makes it known.
3. If you like, enter a **Price ceiling**: the most you will pay a unit. It is kept here, never sent, and every quote is checked against it.
4. Write a **Rationale** if it helps the approver.
5. Press **Preview**. For each partner it shows what it will be told (part, description, quantity, need-by) and what it will **not** be told: your price ceiling, and anything else its redactions blank out. If an administrator lets a partner see the ceiling, the preview says so in amber.
6. Press **Send** (**Send 2 RFQs** for two partners), or **Change** to go back.

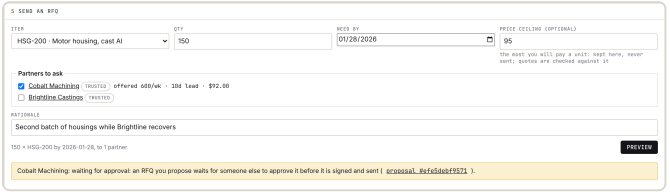


Send an RFQ. Send an RFQ: the item, how many and by when, the partners our policy sends RFQs to (the one that offered capacity first, with its offer) and, if you like, the most you will pay a unit. `shell-quotes-send-rfq-form-bea-desktop`

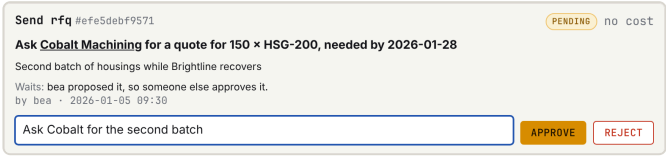


What Cobalt is told, and not. Before anything is sent, the preview shows what Cobalt will be told (part, description, quantity, need-by) and not told: the $95.00 price ceiling, which stays here on the proposal and checks every quote. `shell-quotes-send-rfq-preview-bea-desktop`

Each RFQ is a proposal, governed like any other. One you send waits for someone else to approve it before it is signed and sent: the form says "waiting for approval", with a link to each proposal. Once it is approved, it appears under **A RFQs sent**, with what the partner was not told and the ceiling kept here.



The RFQ waits for approval. Each RFQ is a proposal: one Bea sends waits for someone else to approve it before it is signed and sent. `shell-quotes-send-rfq-sent-bea-desktop`



An RFQ to approve. Bea's RFQ in Maya's approvals: what it asks of whom, and why it waits (a person asked for it, so someone else approves it). `shell-approvals-rfq-maya-desktop`

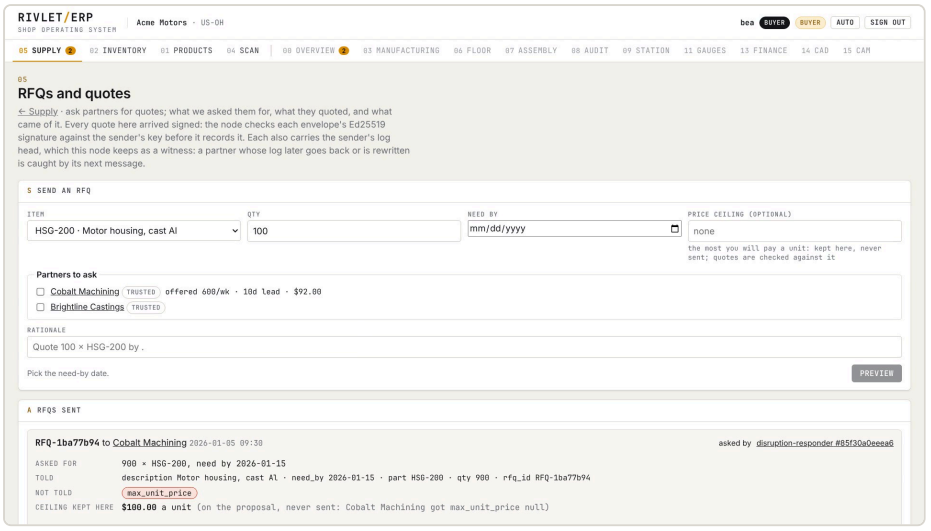
In Approvals, its card is headed "Send rfq", then says what it asks: "Ask Cobalt Machining for a quote for 150 × HSG-200, needed by 2026-01-28".

Rivlet's agents also send RFQs on their own, for example when a supplier warns of a disruption.

Each partner's own node answers an RFQ with a signed quote. In the demo, Brightline's and Cobalt's nodes run inside the demo and answer the same way: an RFQ to Cobalt is answered within seconds.



Cobalt's quote, live. Seconds later Cobalt's node answers the RFQ with a signed quote: its signature checked, its sender's log witnessed, $92.00 within the $95.00 ceiling kept here, and the purchase-order proposal it became. `shell-quotes-live-quote-bea-desktop`



Movie: Send an RFQ and get a quote (1 min 20 s). Bea asks Cobalt for a quote on 150 housings with a $95 ceiling; the preview shows what Cobalt is told and that the ceiling stays home; Maya approves the RFQ; Cobalt's node answers live with a signed quote, its log witnessed, within the ceiling, and it becomes a purchase-order proposal that waits for a person. `shell-send-an-rfq`

Chapters: Ask for a quote 0:00 · What Cobalt is told 0:22 · A second person approves 0:42 · Cobalt answers 0:56

When a quote comes in

The sourcing agent checks each signed quote against the ceiling you kept:

- Within the ceiling, it proposes a PO. Under the plant's limit for agents ($10,000 with a trusted partner in the demo), the PO runs on its own. Over it, the PO waits for a person.
- Over the ceiling, it is held for a person to decide, with how much it is over.

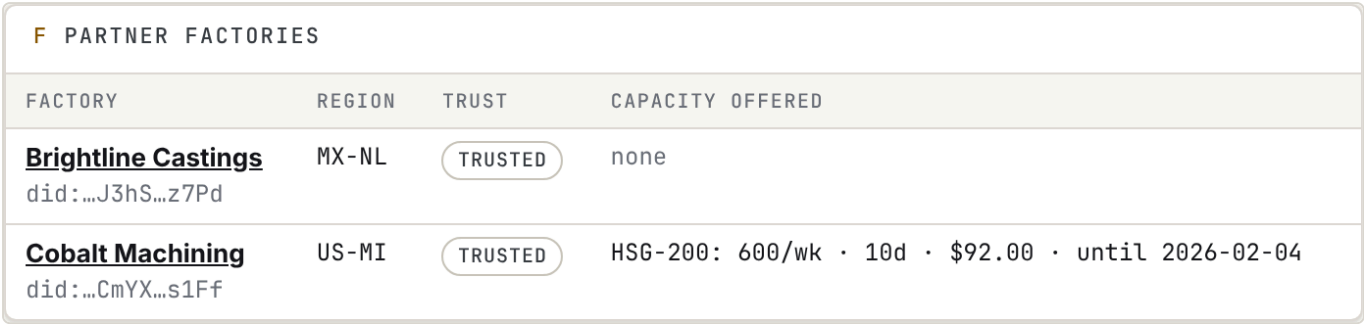
The whole exchange is shown step by step in Security in plain words.

Partners and their trust tiers

A partner factory is another company's Rivlet node. Each has a trust tier, set by your administrator. The tier decides which kinds of message you exchange with it:

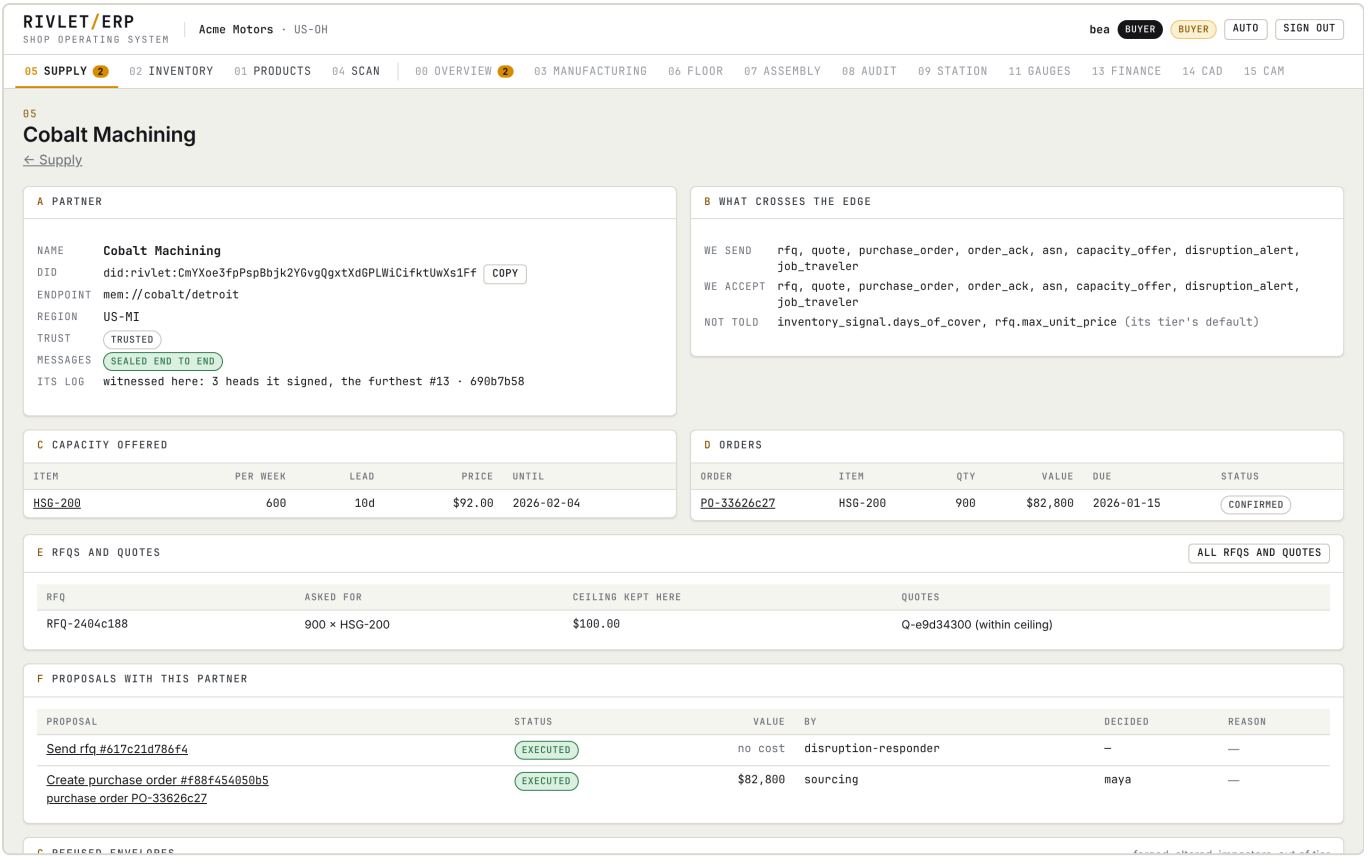
Tier	May exchange
unknown	RFQs, job travellers, disruption alerts
known	the above, and quotes and capacity offers
trusted	the above, and purchase orders, acknowledgements and ship notices
partner	the above, and inventory signals and demand forecasts

So a known partner may quote; only a trusted one may take a purchase order. Messages of other kinds from it are refused and counted.



Partner factories. Partner factories this node federates with, their trust and the capacity each has offered. `shell-supply-partners-bea-desktop`

Click a partner's name anywhere to open its record: its identity (a DID), trust tier, whether messages to it are sealed, what it is not told, the capacity it offered, and its orders, RFQs, quotes and refusals.



A partner opened. Cobalt Machining as a record: its DID, trust tier, sealing, what it is not told (rfq.max_unit_price), the capacity it offered, its orders, RFQs, quotes and refusals. `shell-record-partner-bea-desktop`

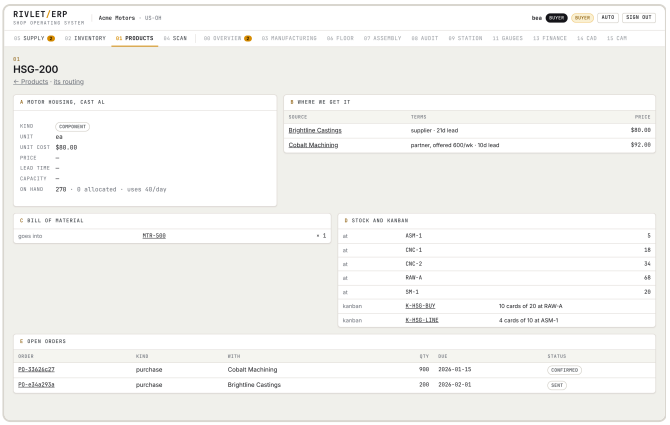
Screens name a partner by its name: an approval card for a PO to a partner reads "Buy … from Cobalt Machining", with its DID (`did:rivlet:…`) when you hover over the name, and **08 Audit** names partners too. The partner's record page shows both.

Suppliers, items and supply risk

E Suppliers and sources lists your suppliers and what each sources. Click an item anywhere to open its record: its cost, its sources (suppliers and partner offers), what it goes into, stock by location and kanban, and open orders.

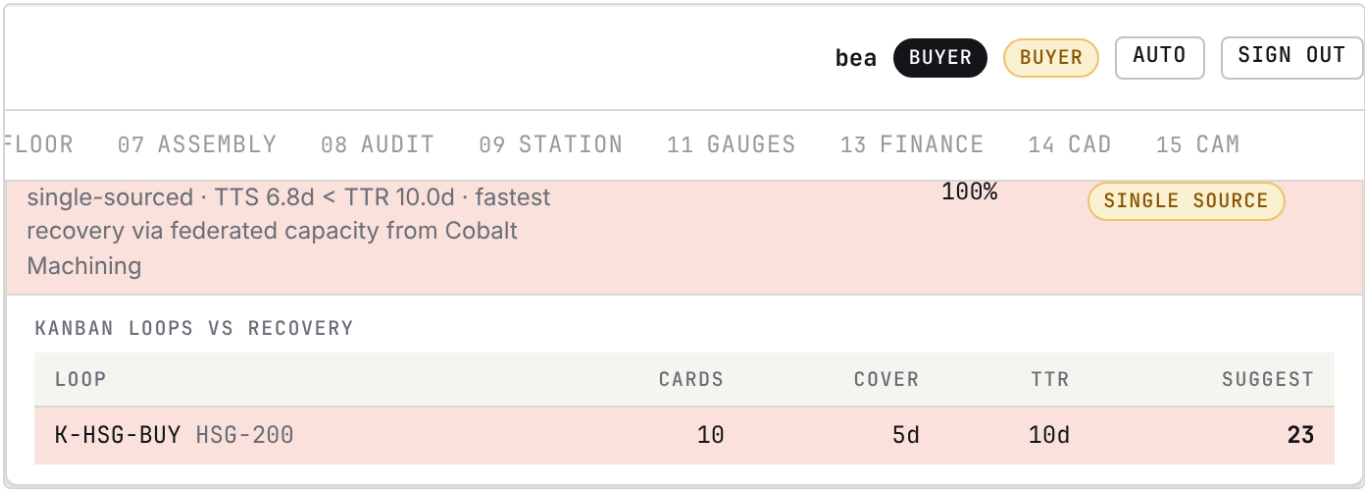


Suppliers and sources. Suppliers and the items each one sources. `shell-supply-suppliers-bea-desktop`



An item opened. HSG-200 as a record: its cost, its sources (Brightline, and Cobalt's offer), what it goes into, stock by location and kanban, and open orders. `shell-record-item-bea-desktop`

D Resilience shows which kanban loops would run dry if a supplier could not deliver, and for how long. Use it after a supplier warns of a disruption.



Resilience. Resilience: which kanban loops are exposed to a supplier that cannot deliver, and for how long. `shell-supply-resilience-bea-desktop`

The federation outbox

G Federation outbox lists each signed message sent to a partner, and whether the partner's node took it. A message the partner's node could not take yet (it is offline, say) stays **pending**: the node tries again after 2 s, then 4 s, 8 s and so on, up to every 5 minutes, for as long as the message is valid, and the row says when the next try is.

G FEDERATION OUTBOX

MESSAGE	TO	STATUS	TRIES	CREATED
rfq	Delta Foundry	PENDING next try in 294 s	9	2026-01-05 09:30
purchase_order	Delta Foundry	PENDING next try in 61 s	8	2026-01-05 09:30
rfq	Cobalt Machining	DELIVERED	1	2026-01-05 09:30
purchase_order	Cobalt Machining	DELIVERED	1	2026-01-05 09:30

A message waiting to be tried again. Messages to Delta Foundry, which has no route from here, wait in the outbox: each says how many tries it has had and "next try in N s", on a timer of its own whatever the plant's clock says. `network-supply-outbox-retry-bea-desktop`

Receiving goods

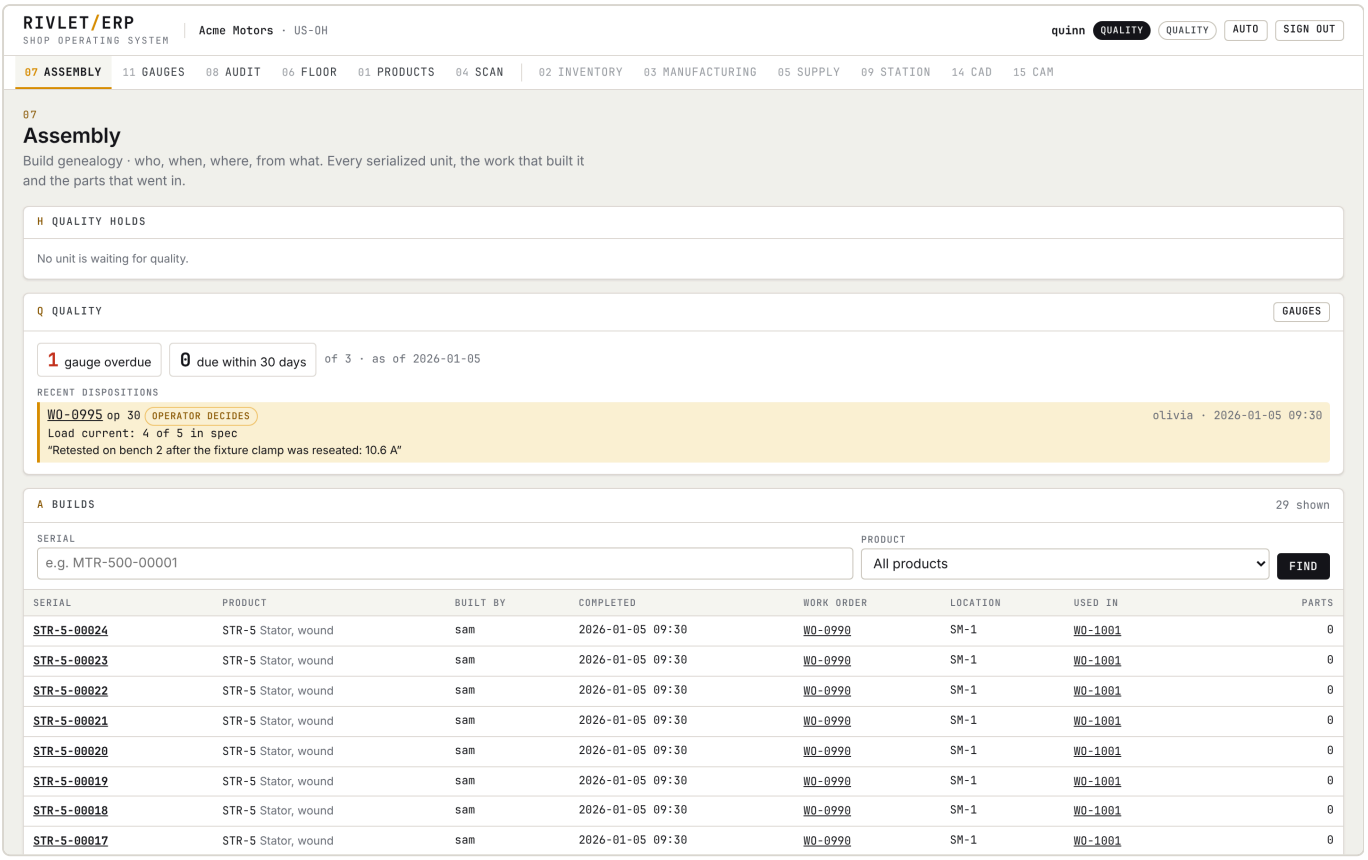
Buyers may book goods in from outside the plant. See Planning for the steps.

Quality

This chapter is for quality. Only the quality role may sign off a held unit, whatever anyone else's approval limit. In the demo that is Quinn.

Your page: 07 Assembly

You start on **07 Assembly**: builds, each unit's history (genealogy), and the held units waiting for you. Its number counts the held units.



Assembly as Quinn. Build genealogy: every serialized unit, who built it, where, when and from what. Seen by Quinn (quality). `shell-assembly-page-quinn-desktop`

Your other sections are **11 Gauges** (instruments and calibration) and **08 Audit** (the event log).

Checks and what happens when one fails

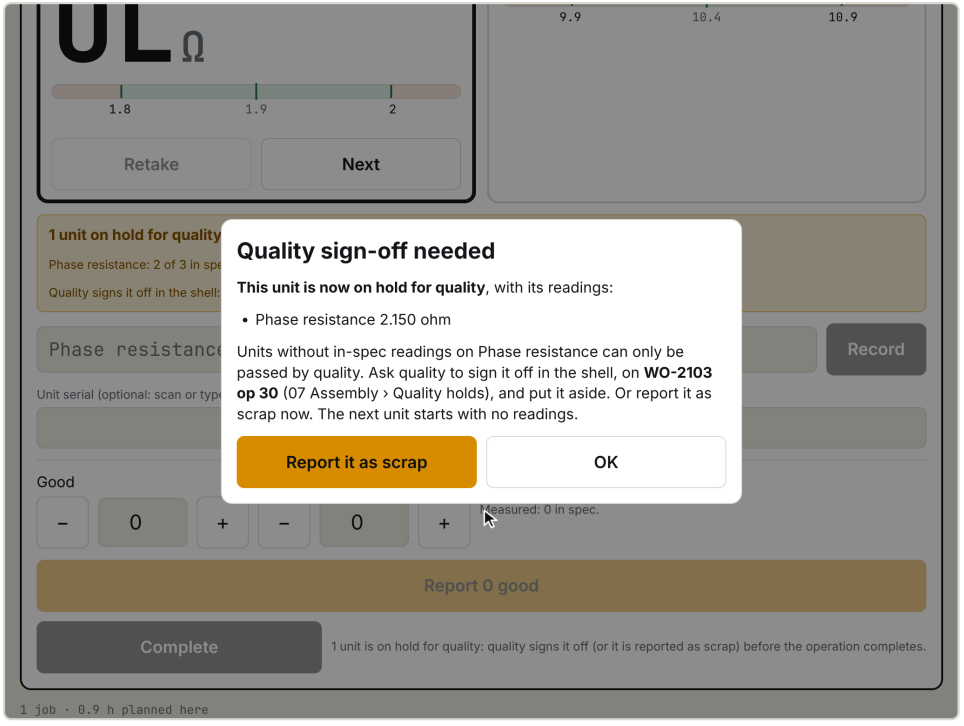
A check is a measurement each unit must pass at an operation, for example a phase resistance between 1.8 and 2.0 Ω at TEST. Checks are part of the item's routing (**01 Products**). Each check says what happens when a unit fails it (its *on fail*):

On fail	What happens
Operator decides	The operator may report the unit good with a reason, kept with its readings. Or as scrap.
Quality decides	The operator cannot pass it. Reported good, the unit goes **on hold for quality**, with its readings.
Scrap	The unit is reported as scrap.

Holds

How a unit goes on hold

1. At the station, a reading falls outside the band, for example phase B at 2.15 Ω against 1.8–2.0. The tablet marks the unit failed, in red.
2. The check is set to **Quality decides**. When the operator reports the unit good, it goes on hold for quality, with its readings. The dialog says so, and where quality signs it off.
3. The operator taps **OK** and puts the unit on the red rack.

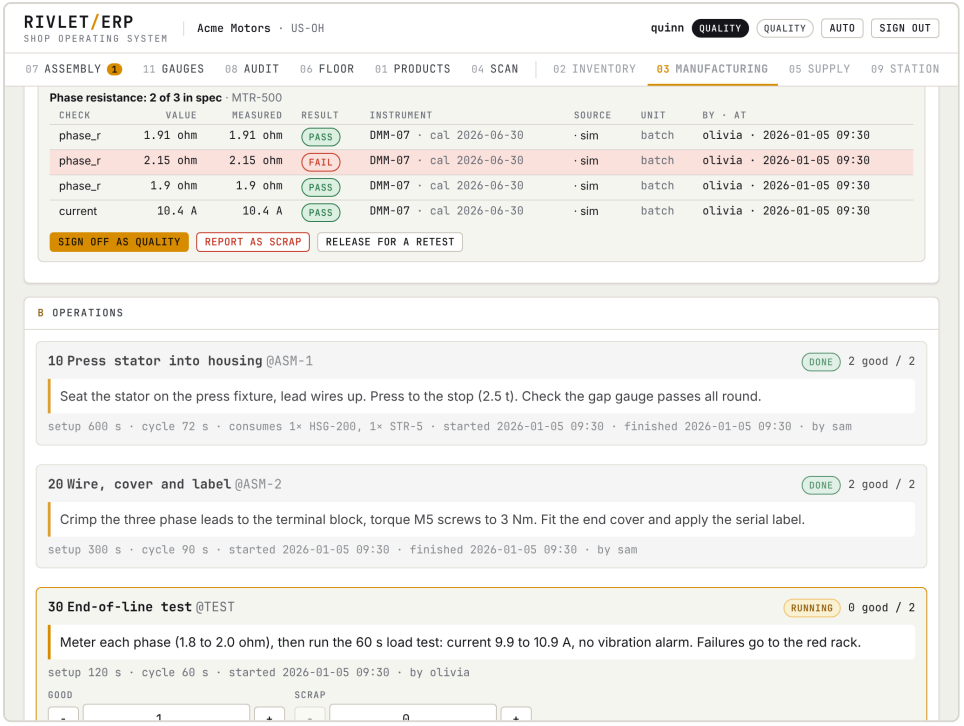


On hold for quality. A unit failing a "Quality decides" check goes on hold for quality with its readings; the dialog says where quality signs it off. `floor-quality-signoff-dialog-olivia-tablet`

To sign off a held unit

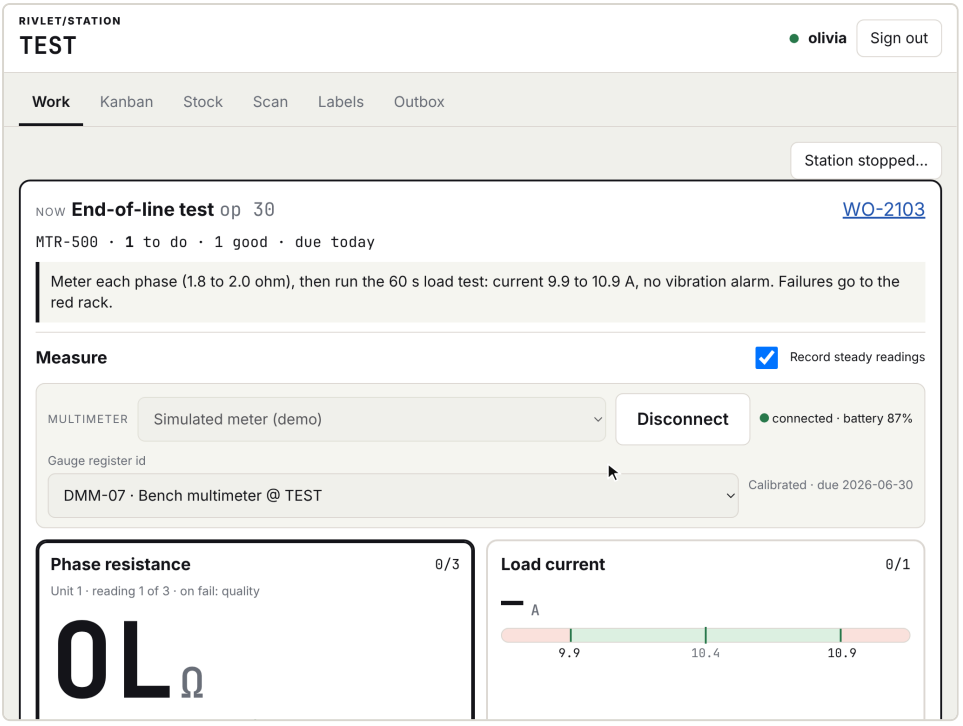
1. Open the work order (from **07 Assembly**, or **03 Manufacturing**). The held unit is a hold card, with its readings, what failed and who held it.
2. Re-test the unit if you need to.

3. Choose one:
    - **Sign off as quality**: write the reason first (for example "MRB 114: phase B 2.15 Ω within the motor design margin, use as is"), then press **Pass this unit**. The unit passes with its own readings, and your reason is kept with them.
    - **Report as scrap**: the unit is scrapped.
    - **Release for a retest**: write why (for example "probe slipped: measure again"), then press **Release it**. The unit goes back to be measured again.

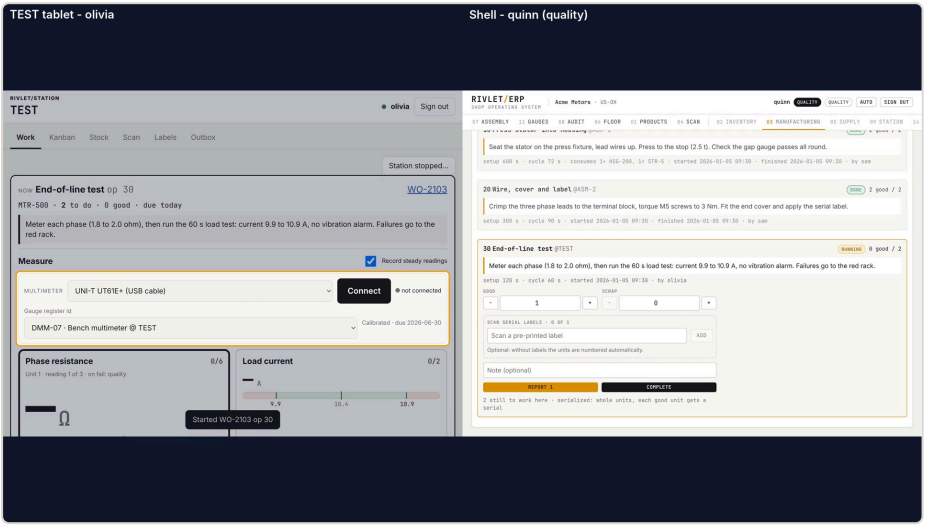


The hold, in the shell. Quality sees the held motor on the work order: its readings, what failed and who held it, with Sign off. `floor-quality-hold-card-quinn-tablet`

Back at the station, the tablet counts the signed-off unit good. The next unit starts with no readings.



After the sign-off. After quality's sign-off the tablet counts the unit good; the next motor starts with no readings. `floor-quality-after-signoff-olivia-tablet`



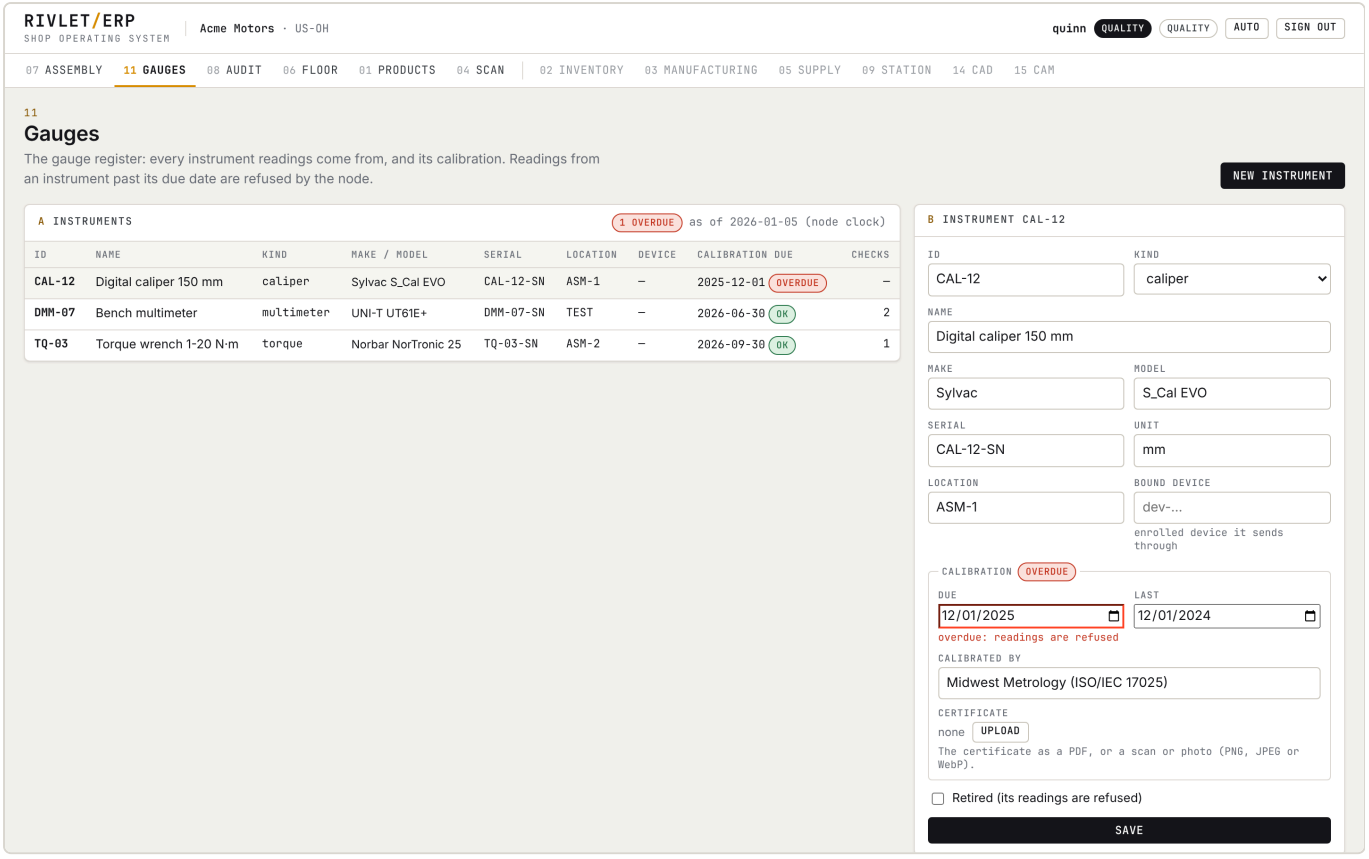
Movie: An out-of-spec unit goes to quality (53 s). At TEST a phase resistance reads 2.15 ohm. The check is set to Quality decides, so the motor goes on hold for quality with its readings; quinn sees the hold on the work order in the shell and signs it off with a reason; the tablet counts it good and the next motor starts with no readings. `floor-quality-hold`

Chapters: Measure the unit 0:00 · The unit is held 0:15 · Quality signs it off 0:29 · Back at the tablet 0:46

Gauges and calibration

11 Gauges is the gauge register: every instrument readings come from, its kind, where it is, and when its calibration is due.

- Readings from an instrument past its due date are refused by the node. At the tablet the operator is told to pick a calibrated instrument and measure again.
- The tablet warns when an instrument is due within 14 days.
- An instrument can be bound to an edge bridge, so its readings arrive with the bridge's identity. See Administration.

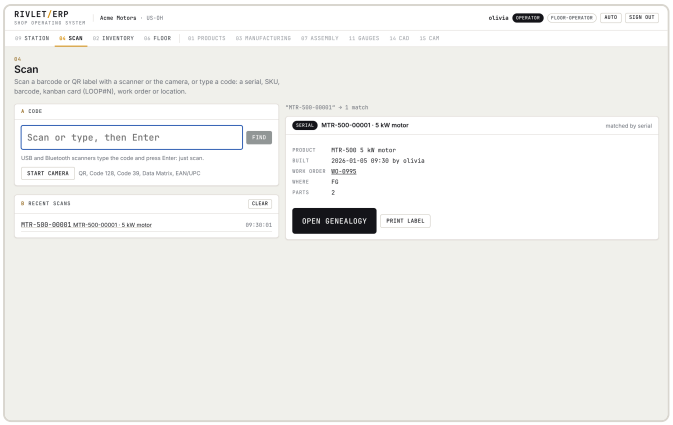


An instrument. An instrument opened in the gauge register, with its calibration record. `shell-gauges-instrument-quinn-desktop`

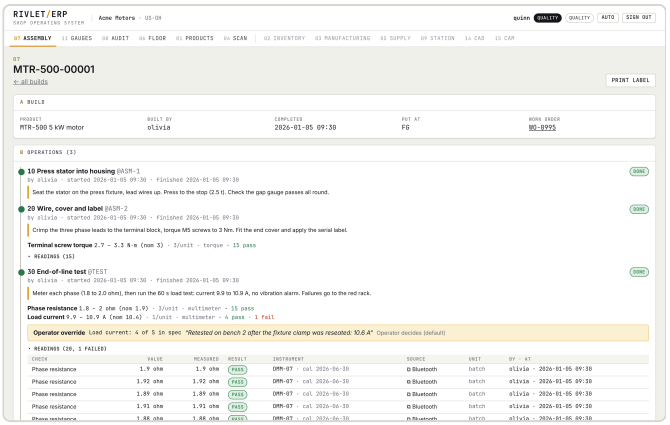
Readings and genealogy

Every reading is kept with the unit, the check, the instrument and who took it. To see a unit's history:

1. Open **04 Scan** and scan or type its serial, or find it in **07 Assembly**.
2. Open its genealogy: the build, each operation with who did it and where, the parts that went in, and the readings taken.



A serial scanned. Scanning a serial opens that unit: what it is, where it was built and a link to its genealogy. `shell-scan-unit-olivia-desktop`

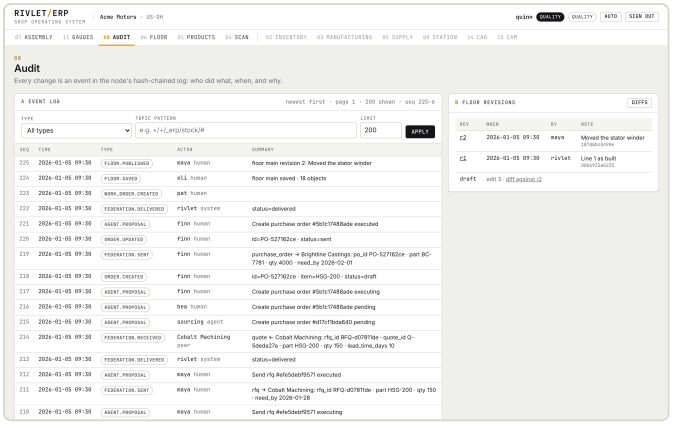


A unit's genealogy. One serial opened: the build, each operation with who did it and where, the parts that went in and the readings taken. `shell-assembly-genealogy-quinn-desktop`

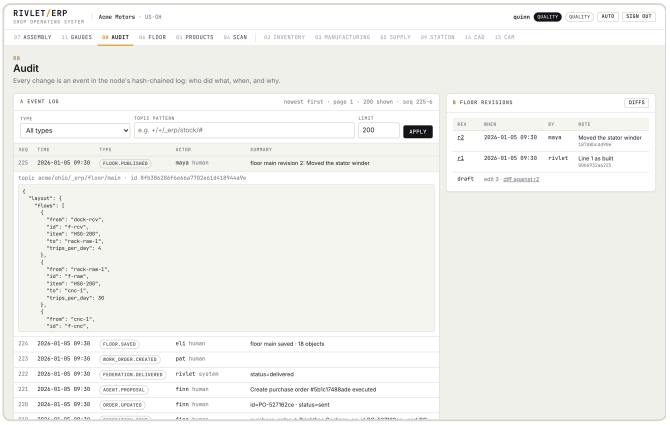
A shipped unit's history also says which order it shipped in.

The audit log

08 Audit is the event log, newest first: every change with who made it, when and why. Filter it by topic or type; click an event to see its full record.



Event log. The event log, newest first: every change with who made it, when and why. `shell-audit-events-quinn-desktop`



An event opened. Clicking an event shows its full record. `shell-audit-event-open-quinn-desktop`

The log cannot be changed after the fact without it showing. See Security in plain words.

Engineering

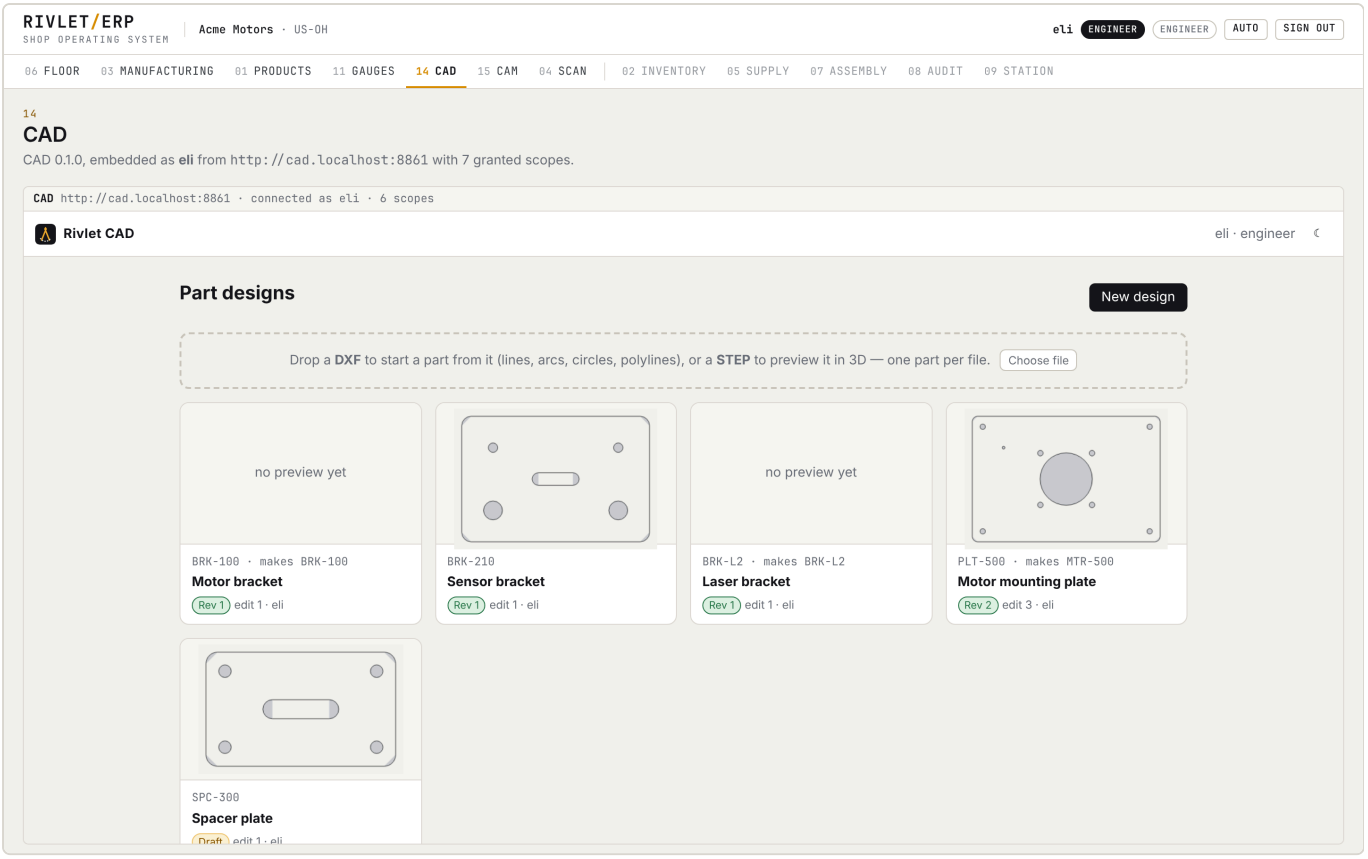
This chapter is for industrial engineers, and for the plant managers who release their work. It takes a part from a drawing to a program loaded in a machine, and covers the floor designer.

From drawing to machine

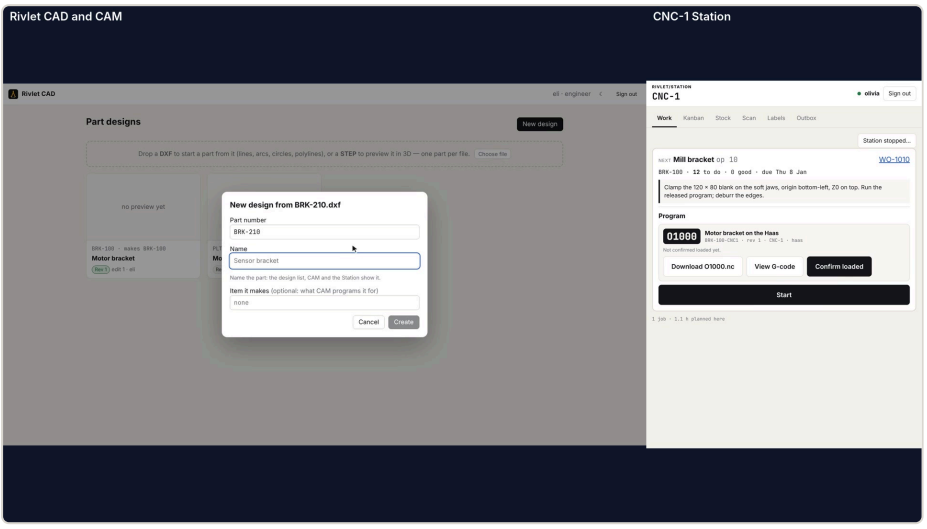
1. **CAD** draws or imports the part and saves it as a design, with revisions. A plant manager releases it.
2. **CAM** turns a released design into a machine program and previews it live. The node generates the program; a plant manager releases it with a program number.
3. **The Station tablet** at the machine shows the released program on the job. The operator loads it and confirms which revision is loaded.

Who	Does what
Industrial engineer	Draws designs, makes programs, edits items, routings and the floor layout. Cannot release, and does not report work.
Plant manager	Releases designs, programs and floor layouts; adds machines.
Floor operator at the machine	Reads and downloads programs, and confirms them loaded.

CAD and CAM open inside the shell as **14 CAD** and **15 CAM**, signed in as you. No token is needed.



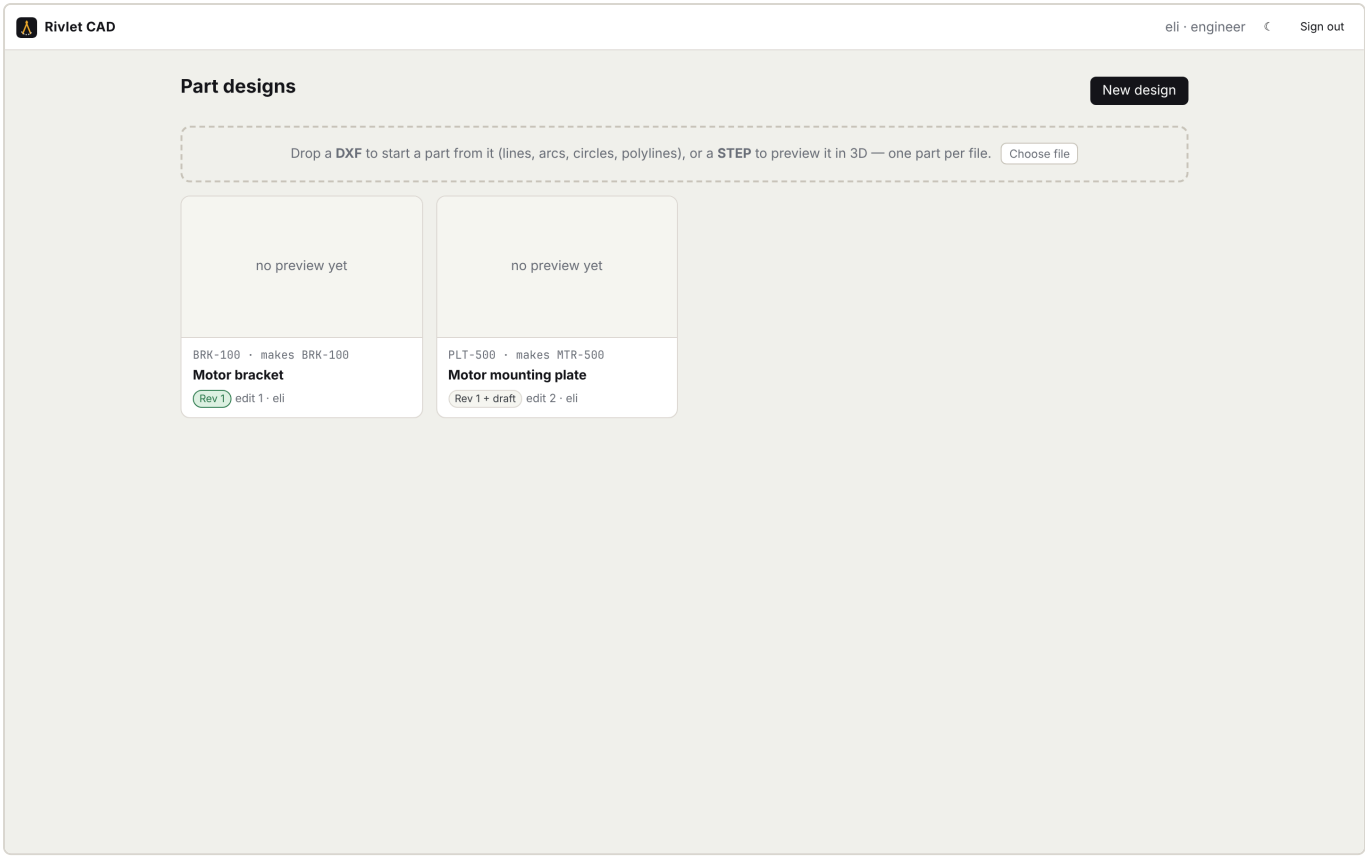
CAD in the shell. CAD runs inside the shell with Eli's session; no token is needed. `design-shell-cad-designs-eli-desktop`



Movie: From drawing to machine (43 s). A bracket drawing goes into CAD, becomes a CNC program with a live toolpath preview in CAM, Maya releases it, and the CNC-1 tablet shows it and confirms it is loaded. `design-promo-drawing-to-machine`

CAD: designs

CAD lists every part design on the node. Each card shows its picture, its id and the item it makes, its name, and its revision: green "Rev N" when the draft matches the released revision, "Rev N + draft" when the draft is ahead, amber "Draft" if never released. It also shows the last edit and who saved it.

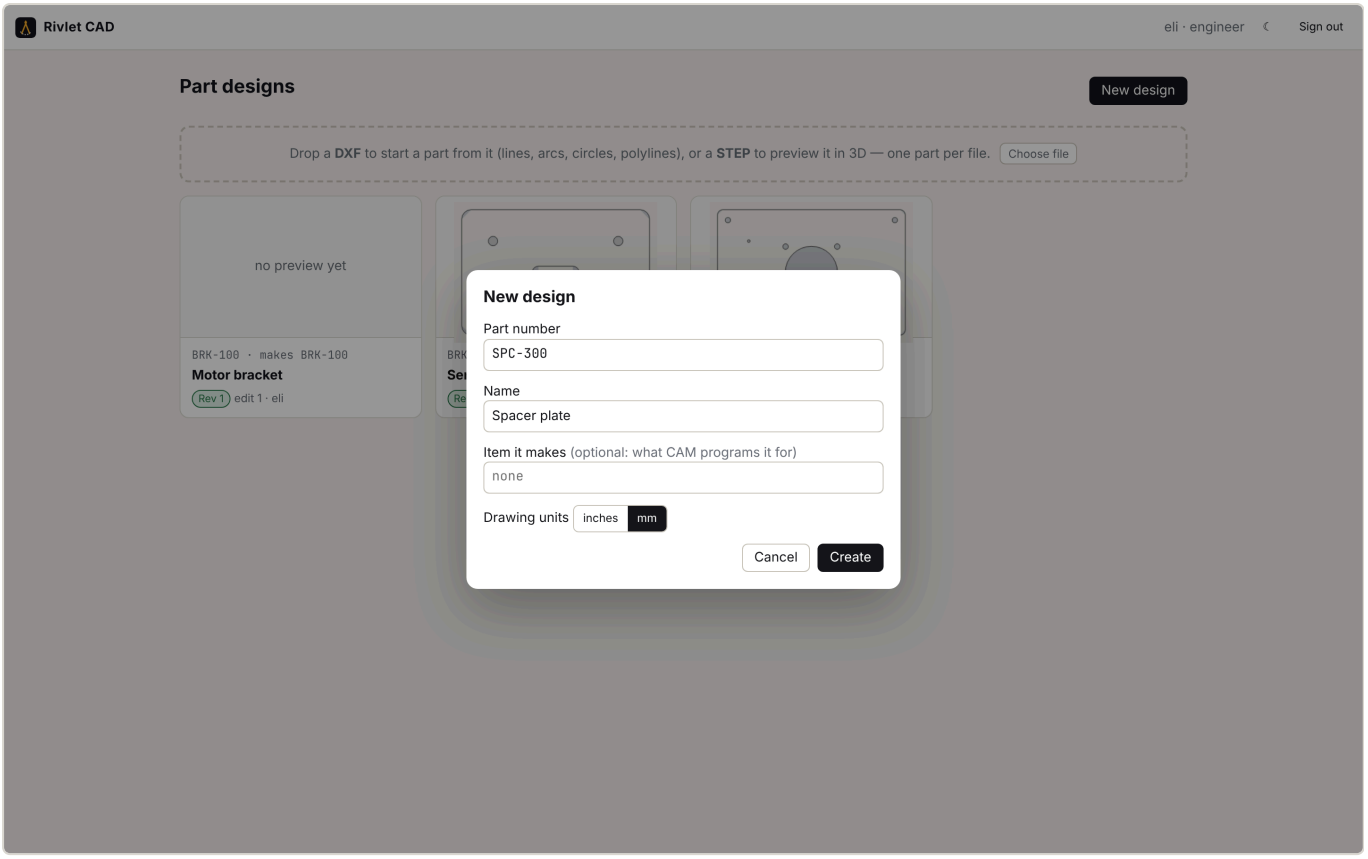


Part designs. The CAD app lists every part design on the node with its thumbnail, the item it makes, and whether the draft matches the last released revision. `design-cad-designs-eli-desktop`

To start a new design

1. Press **New design**.
2. Enter the part number, the name, **the item it makes**, and the drawing units (inches or mm).
3. Press **Create**.

You can change the item later in Inspect.

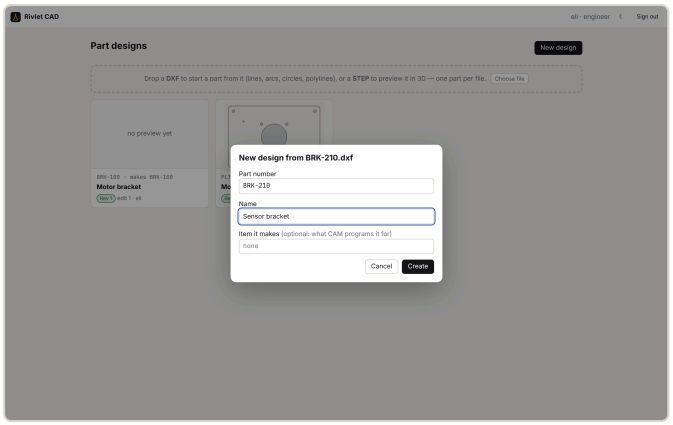


A new design. A new design needs a part number, a name, the item it makes and the drawing units.
`design-cad-new-eli-desktop`

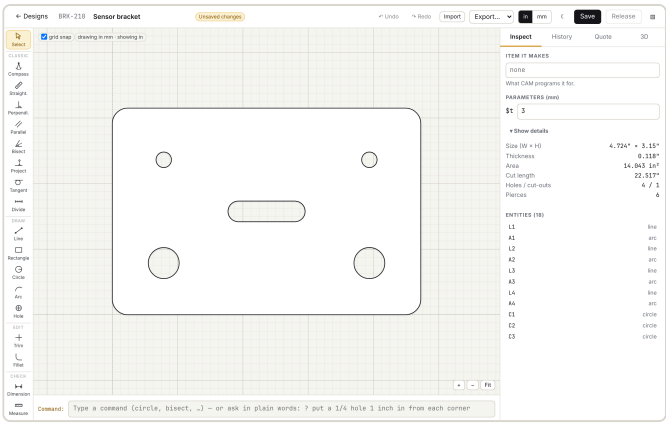
To start from a DXF

1. Drop a DXF on the dashed box, or press **Choose file**.
2. The part number is taken from the file name. The **Name** is the name of the item with that part number, if there is one; otherwise it is empty, and you give the part its name (the design list, CAM and the Station show it).
3. Press **Create**.

The outline becomes the stock and the profile. Every closed shape inside it becomes a hole or a cut-out. Ends that miss by up to 0.01 mm are joined.



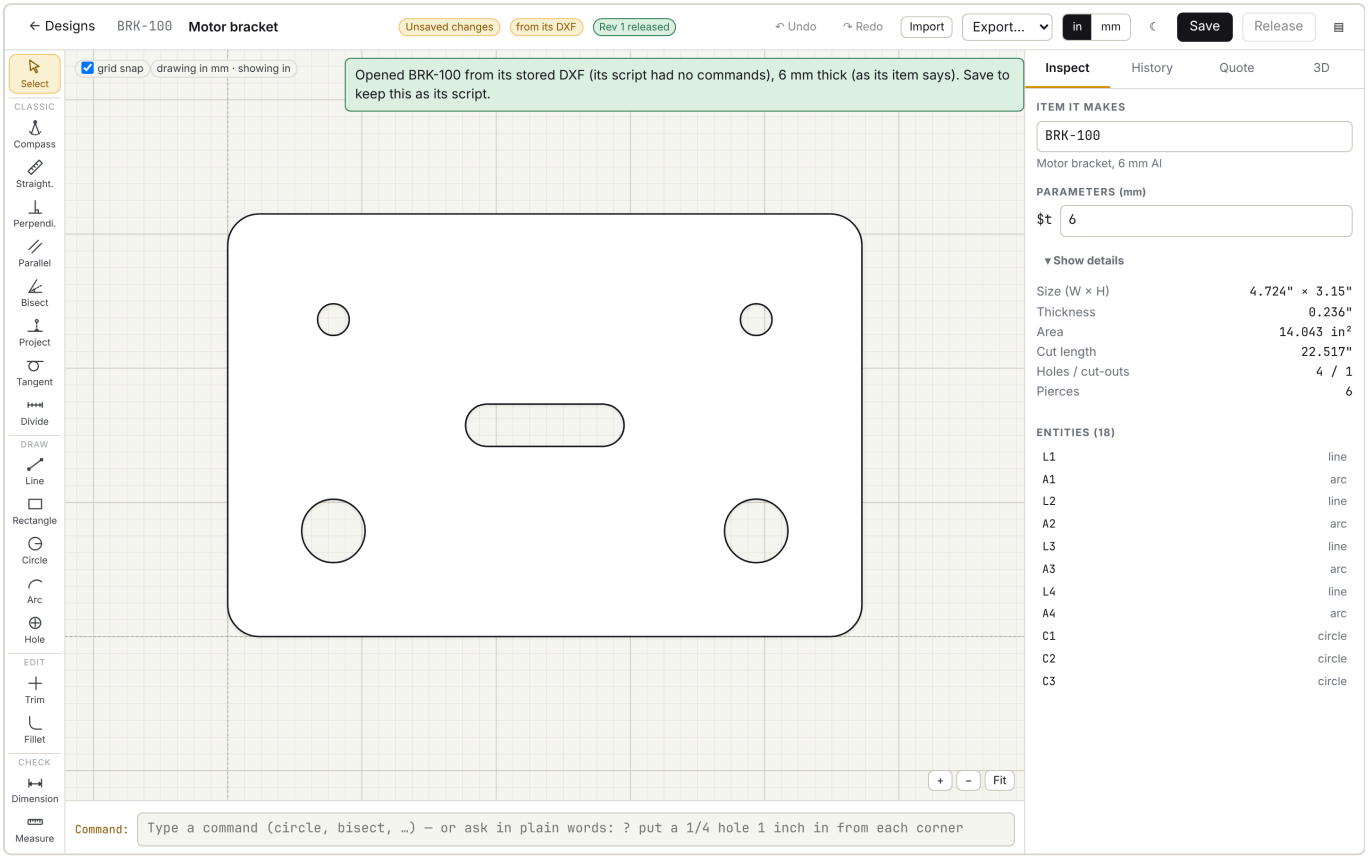
A new design from a DXF. Dropping a DXF on the list starts a new design; the part number comes from the file name. The item it makes is optional and must already be on the node. `design-cad-dxf-new-eli-desktop`



The imported DXF. The DXF's outline becomes the stock and profile, and the holes and slot inside it become cut-outs, so the imported part can be edited and checked like any drawing. `design-cad-dxf-imported-eli-desktop`

A design made outside CAD that has only a DXF opens as that DXF, so you can edit it like any drawing.

A design's thickness is the parameter `$t`. When a DXF does not say how thick the part is, CAD takes the thickness from the item's name ("Motor bracket, 6 mm Al" is 6 mm), or else a default (3 mm, or 1/8"). Check `$t` in Inspect before you save: CAM starts its stock from it.

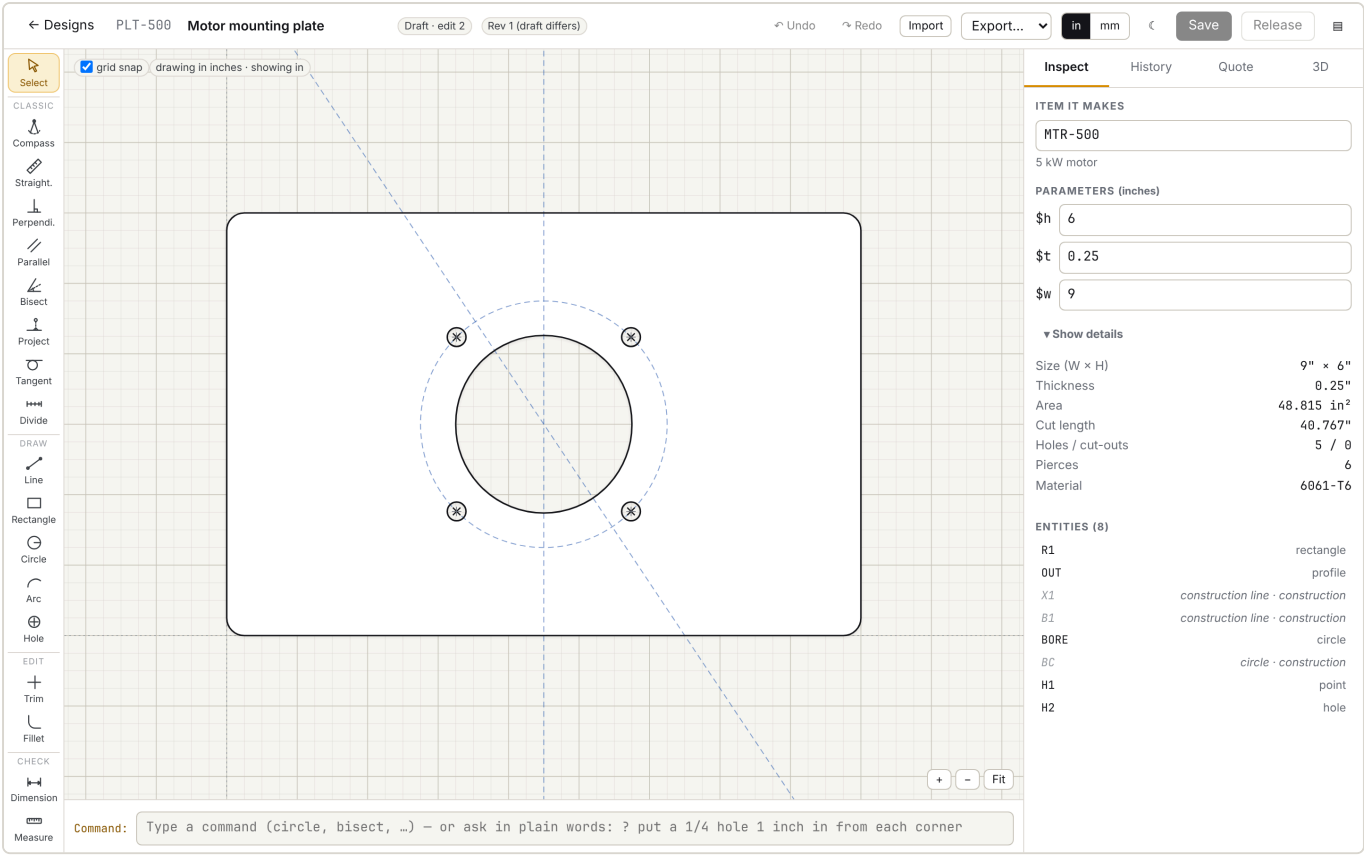


BRK-100 opens as its DXF. BRK-100 was made outside CAD and has only a DXF: CAD opens it as that DXF, so it can be edited like any drawing. `design-cad-brk100-eli-desktop`

To preview a STEP file, drop it: it opens in the 3D tab only.

CAD: drawing

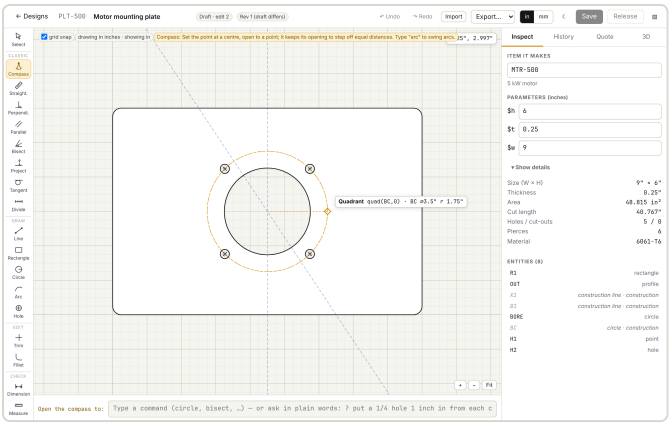
The screen has the tools on the left (Classic: compass, straightedge, perpendicular, parallel, bisect, project, tangent, divide; then Draw, Edit and Check), the drawing in the middle, the command line below it, and the the **Inspect**, **History**, **Quote** and **3D** panels on the right. The header has Undo, Redo, Import, Export, in/mm, light/dark, **Save** and **Release**.



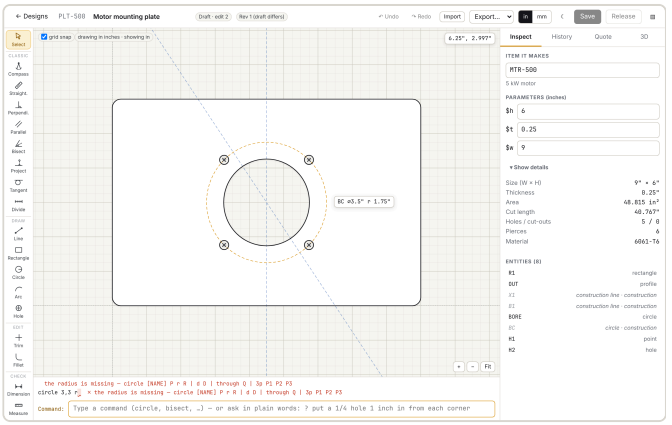
The drawing. The editor: tools on the left, the drawing in the middle, the command line below and the Inspect, History, Quote and 3D panels on the right. `design-cad-drawing-eli-desktop`

Every click is written as a command, so a drawing is also a readable script you can edit.

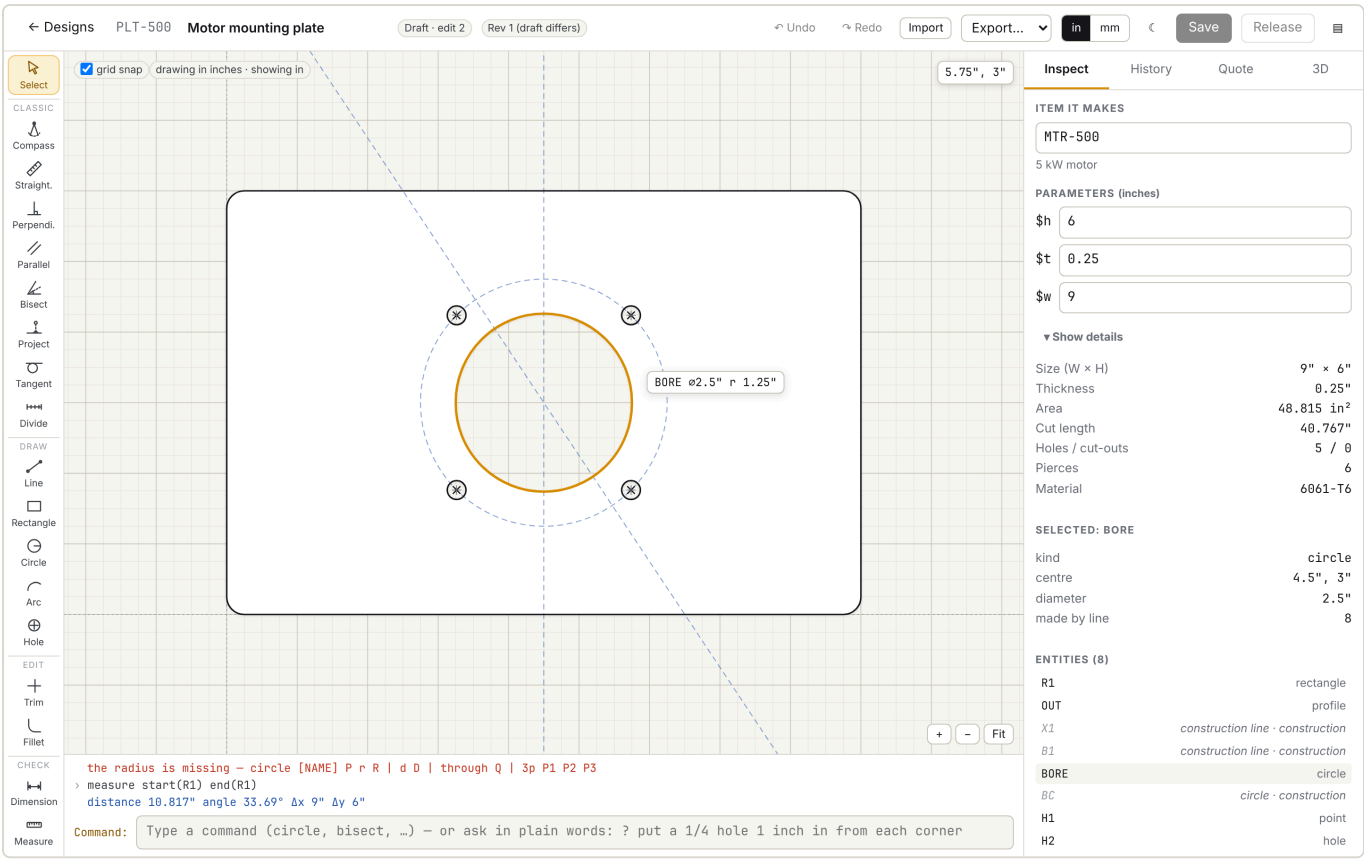
- **Compass:** pick a centre, then the point it opens to. Snaps lock onto named points. It keeps its opening, so the next click steps off the same distance.
- **Typed commands:** type a command at the command line, for example `circle 3,3 r 0.4`. An incomplete one is underlined, with what is missing.
- **Parameters:** values such as `$w`, `$h` and `$t` drive the drawing. Change one and the drawing follows. **Inspect** shows them, with the item, size, thickness, cut length, holes and cut-outs, pierces, material and the named shapes.



Compass: opening to a snap. The compass is centred on the bore and opens to a quadrant of the bolt circle; the prompt at the bottom says what it needs next. `design-cad-compass-eli-desktop`



A typed command with an error. Commands can be typed; an incomplete one is underlined and the message says exactly what is missing. `design-cad-command-error-eli-desktop`

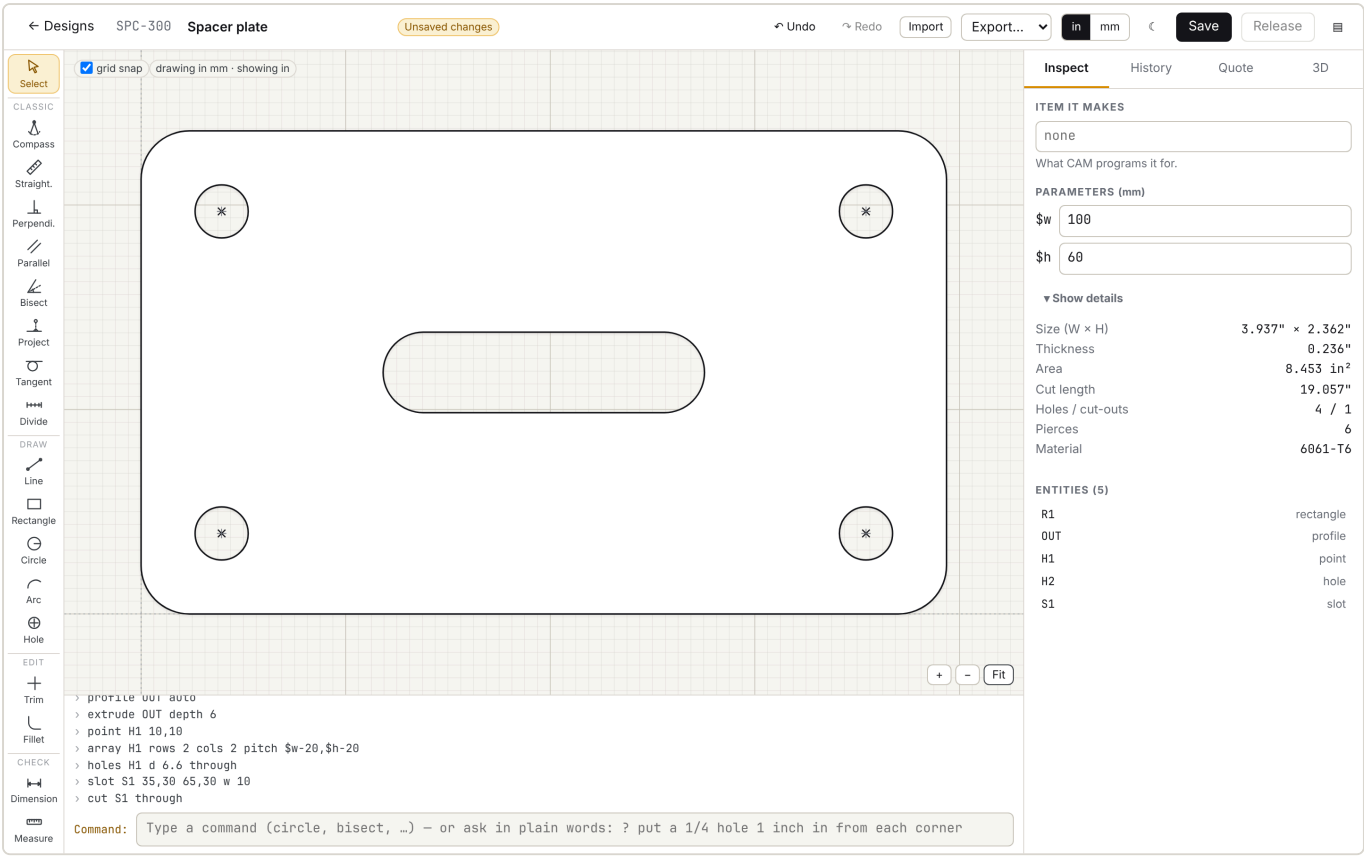


Inspect: parameters, item and entities. Inspect shows the parameters ($w, $h, $t) that drive the drawing, the item the design makes, the part's size, cut length and pierces, and the named entities. `design-cad-inspect-eli-desktop`

A whole part can be typed. This spacer plate is twelve commands:

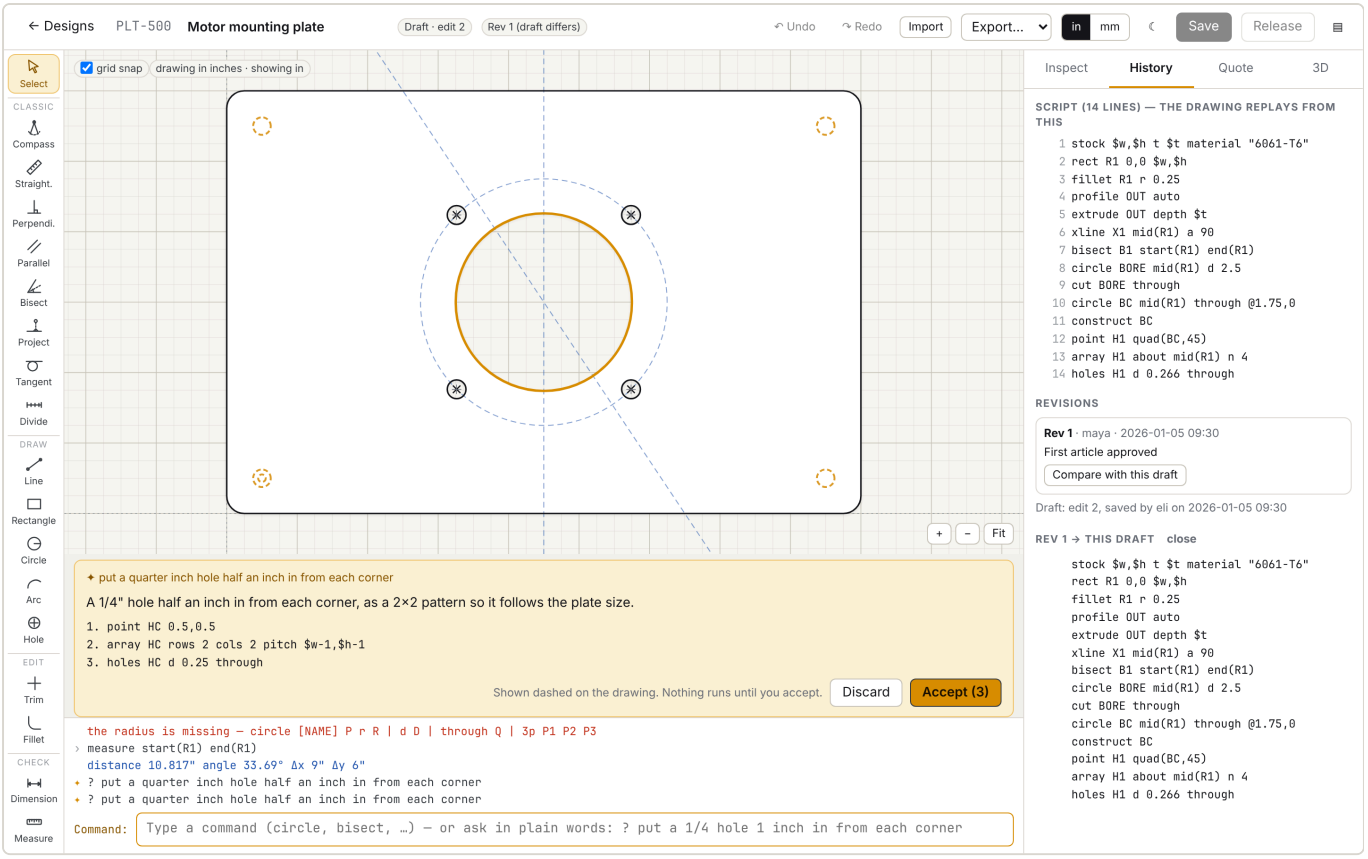
```
param w = 100
param h = 60
stock $w,$h t 6 material "6061-T6"
```

```
rect R1 0,0 @$w,$h
fillet R1 r 6
profile OUT auto
extrude OUT depth 6
point H1 10,10
array H1 rows 2 cols 2 pitch $w-20,$h-20
holes H1 d 6.6 through
slot S1 35,30 65,30 w 10
cut S1 through
```



A part sketched from typed commands. A spacer plate typed as twelve commands: parameters, stock, an outline with filleted corners, a hole pattern and a slot. `design-cad-sketch-eli-desktop`

You can also ask in plain words. The assistant's suggestion is drawn dashed, with its commands listed; nothing runs until you press **Accept**, and one undo takes it back. A node without assistant credentials says so, and everything else keeps working.

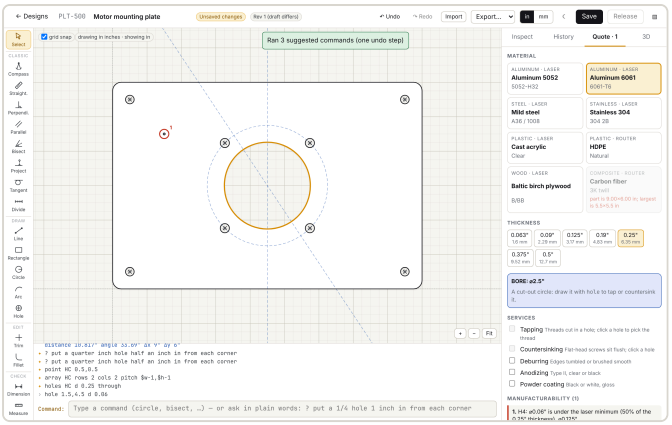


An assistant suggestion, previewed. A suggestion is drawn dashed with its commands listed; nothing runs until you accept it, and one undo takes it back. `design-cad-assistant-preview-eli-desktop`

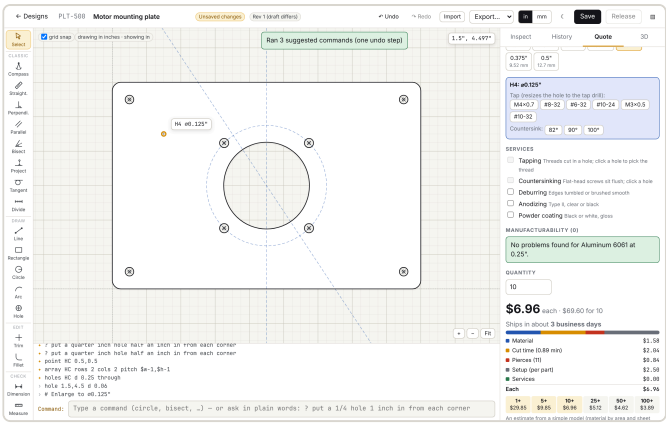
CAD: checks and the estimate

The **Quote** panel checks that the part can be made, and estimates what it costs.

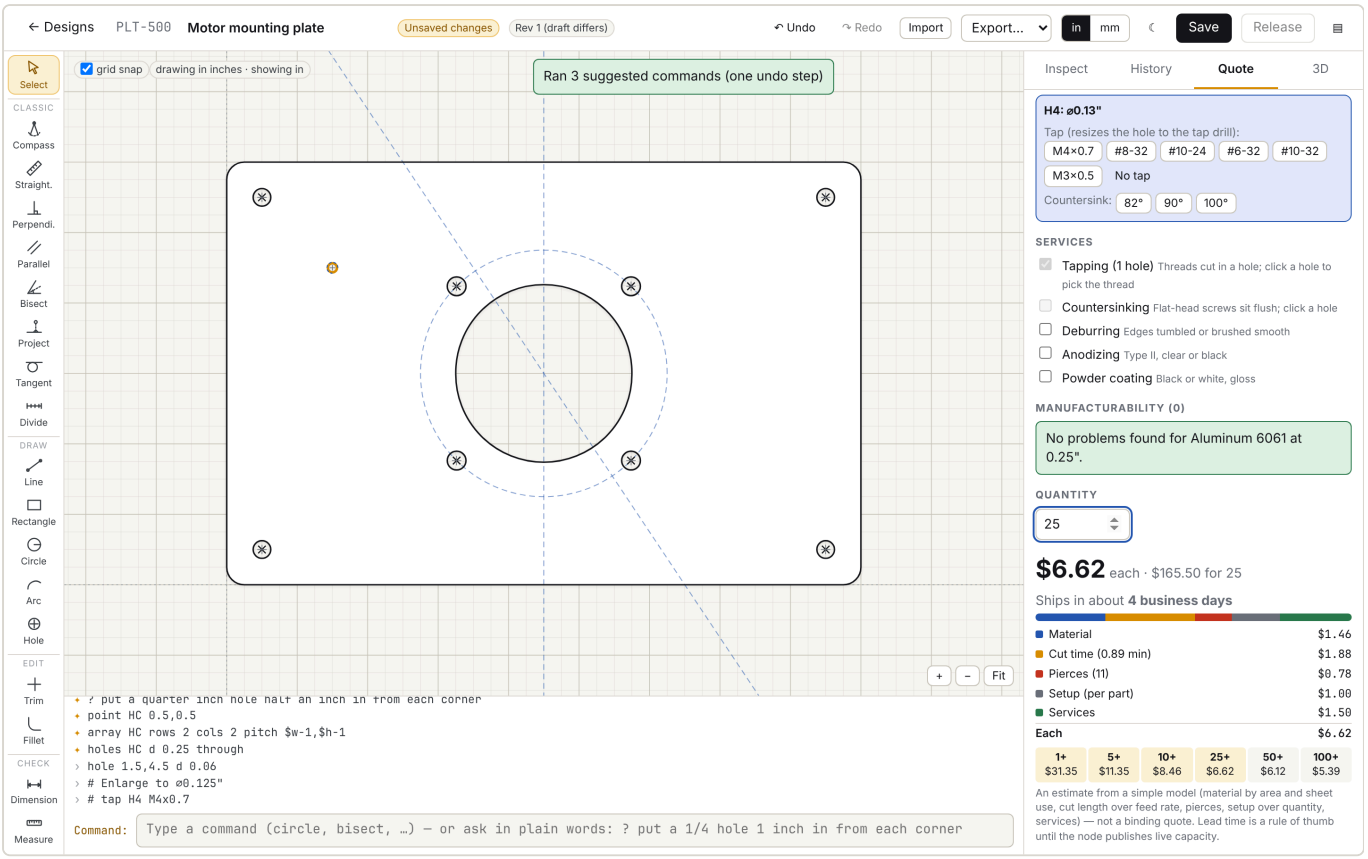
- **Manufacturability:** each issue is numbered on the drawing, with a one-click fix. For example, a hole too small for the sheet.
- **Taps and countersinks:** click a hole with Select. Taps (nearest drill first) and countersinks are offered; choosing one rewrites the hole's command.
- **The estimate:** pick the material and quantity. It prices material, cutting, pierces, setup and services. It is an estimate, and says so.



Manufacturability checks. A hole too small for the sheet is marked on the drawing, with a one-click fix in the Quote panel. `design-cad-quote-dfm-eli-desktop`



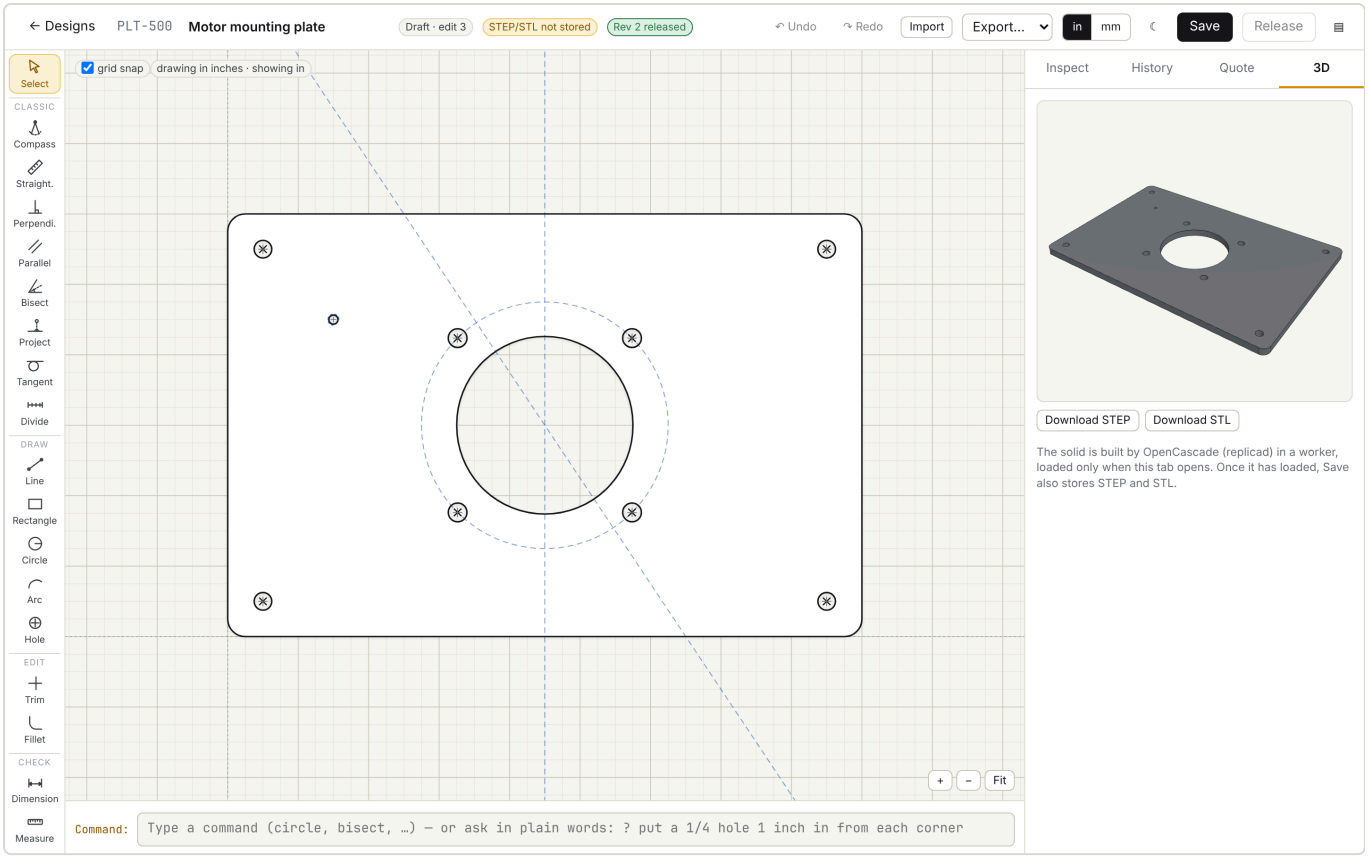
Tap or countersink a hole. Clicking a hole offers taps (nearest drill first) and countersinks; choosing one rewrites the hole's command. `design-cad-hole-tap-eli-desktop`



An instant estimate. The estimate prices material, cutting, pierces, setup and services for the chosen quantity, and is labelled as an estimate. `design-cad-quote-price-eli-desktop`

CAD: the 3D view

The **3D** tab builds the solid in your browser. Drag to turn it. The buttons below it download STEP or STL. Once the 3D view has loaded, **Save** also stores STEP and STL with the design.

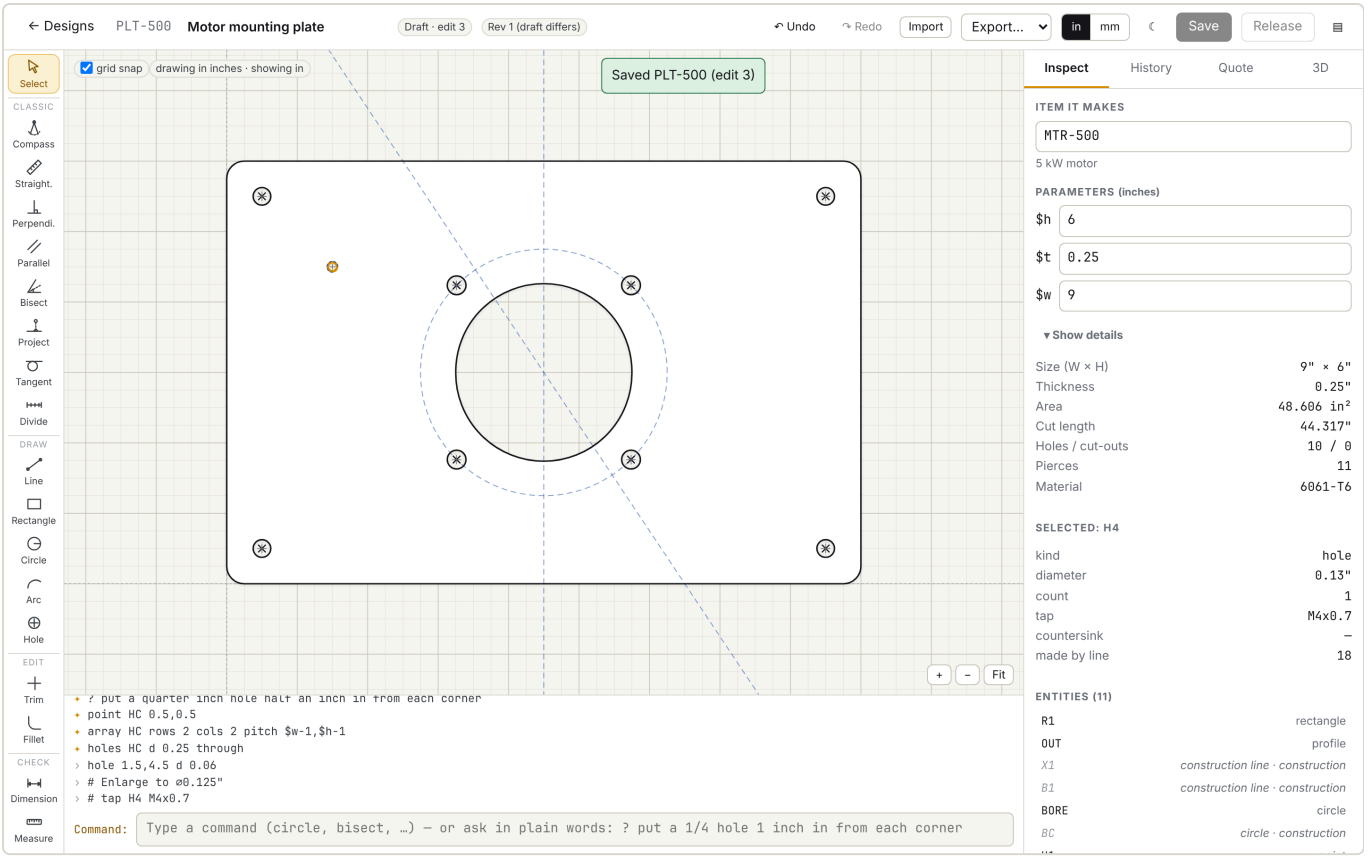


The part in 3D. The 3D tab builds the solid with OpenCascade in the browser, served by the node; once it has loaded, saving also stores STEP and STL. `design-cad-3d-eli-desktop`

CAD: saving and releasing

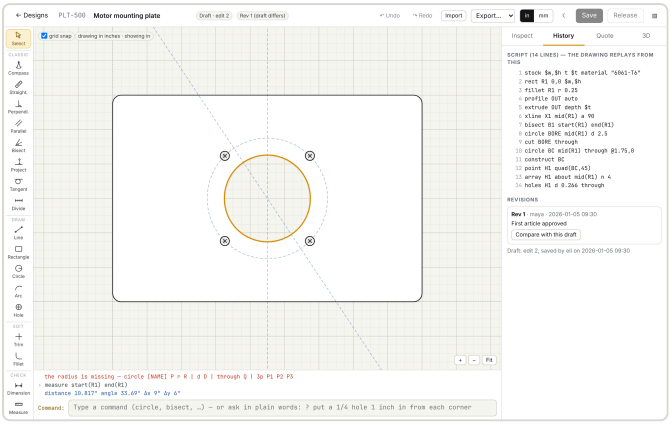
To save a design

Press **Save**. CAD stores the script, the DXF (with OUTLINE, HOLES, POCKET and SKETCH layers, and the thickness), an SVG, a picture and, once 3D has loaded, STEP and STL. Each save is the next edit of the draft.

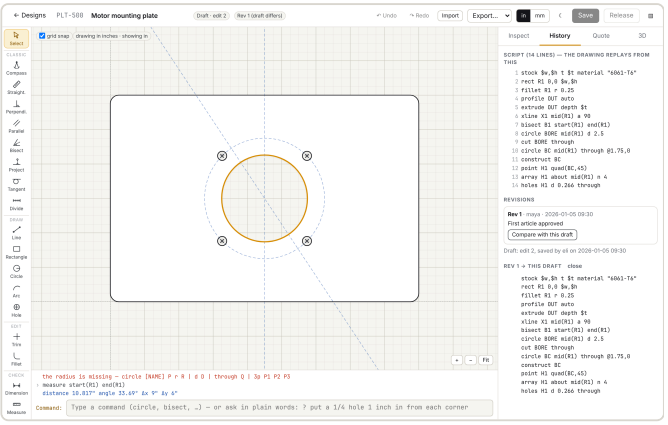


Saved to the node. Save stores the script, DXF (with OUTLINE, HOLES, POCKET and SKETCH layers), SVG, a thumbnail and, once the 3D view has loaded, STEP and STL as the next edit of the draft. `design-cad-saved-eli-desktop`

History shows the draft as a numbered script, and the revisions. **Compare** shows what the draft adds or changes since a revision.



History: the drawing as a script. Every click is written as a command, so the drawing is a readable script; the revisions are listed below it. `design-cad-history-eli-desktop`



Compare a revision with the draft. Compare shows which commands the draft adds or changes since revision 1. `design-cad-history-diff-eli-desktop`

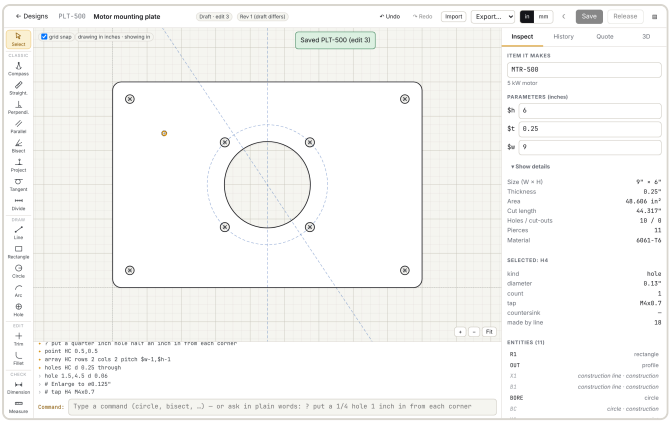
To release a design

Releasing is for a plant manager. For an engineer, **Release** is off: "Releasing needs an approver".

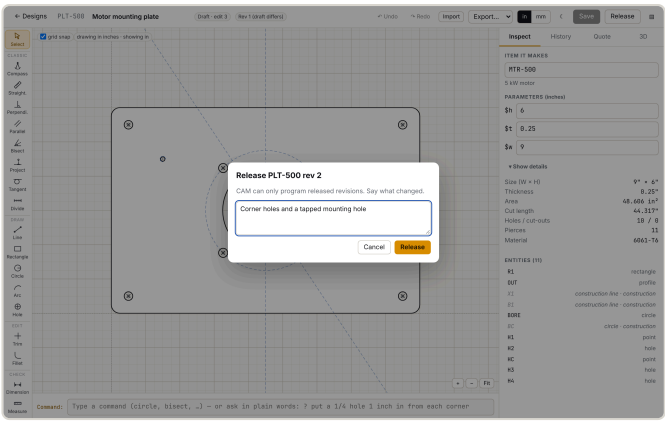
1. Open the saved draft.

2. Press **Release**.
3. Write what changed.
4. Press **Release** in the dialog. The draft becomes the next revision.

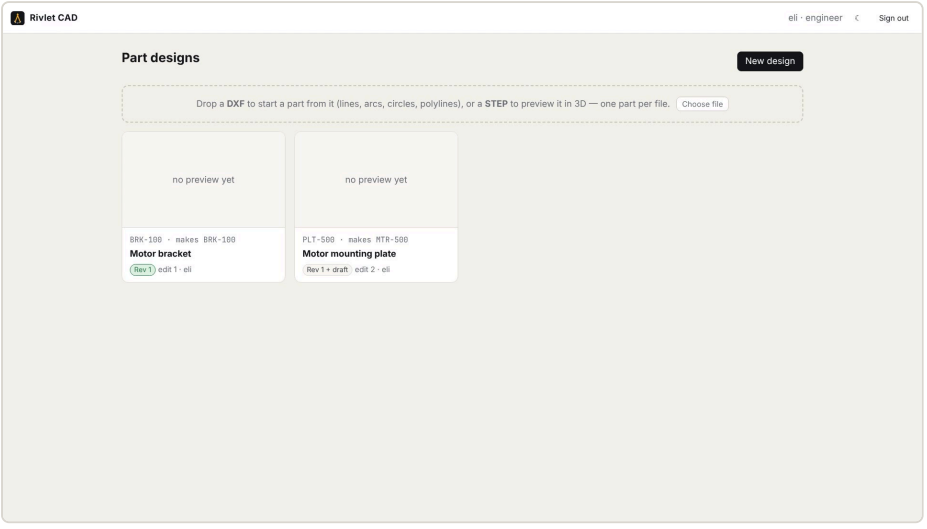
Only released revisions with a DXF can be programmed in CAM.



Release is for a plant manager. An engineer saves drafts; releasing is designs:release, a plant manager's right ("Releasing needs an approver"). `design-cad-release-engineer-eli-desktop`



Release a design. A plant manager releases the saved draft as the next revision, with a note saying what changed; CAM can only program released revisions. `design-cad-release-dialog-maya-desktop`

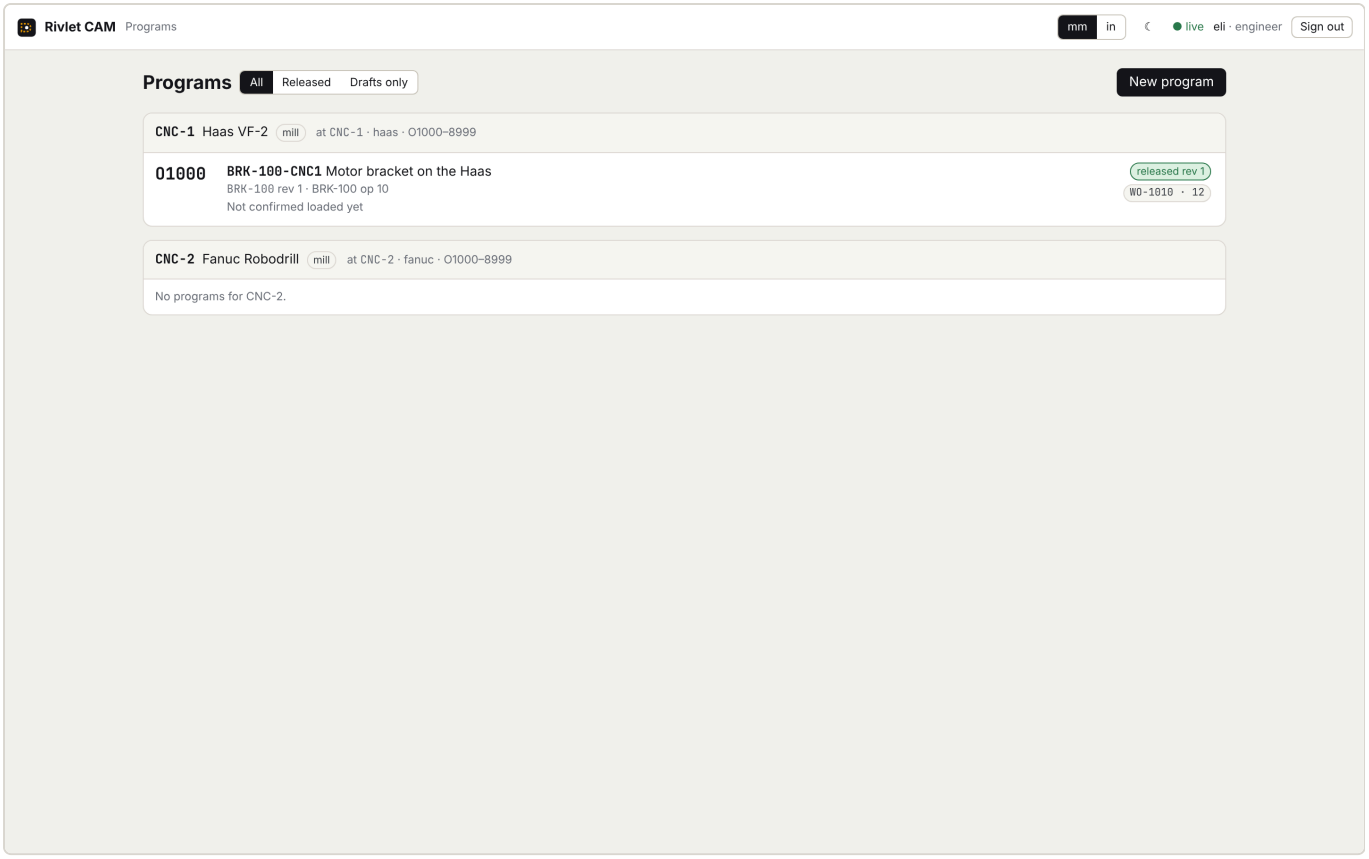


Movie: CAD basics (2 min 0 s). A tour of the CAD app on the motor mounting plate: tools and snaps, the compass, typed commands, parameters, the script and its history, manufacturability checks and the estimate, the 3D view, saving and releasing. `design-cad-basics`

Chapters: Designs 0:00 · Tools and the compass 0:08 · Typed commands 0:26 · Parameters 0:40 · History 0:48 · Checks and the estimate 0:59 · 3D 1:19 · Save and release 1:32

CAM: programs

15 CAM lists programs by machine: the program number once released, the design revision, the item and operation, whether it was confirmed loaded, and the work orders queued for it.

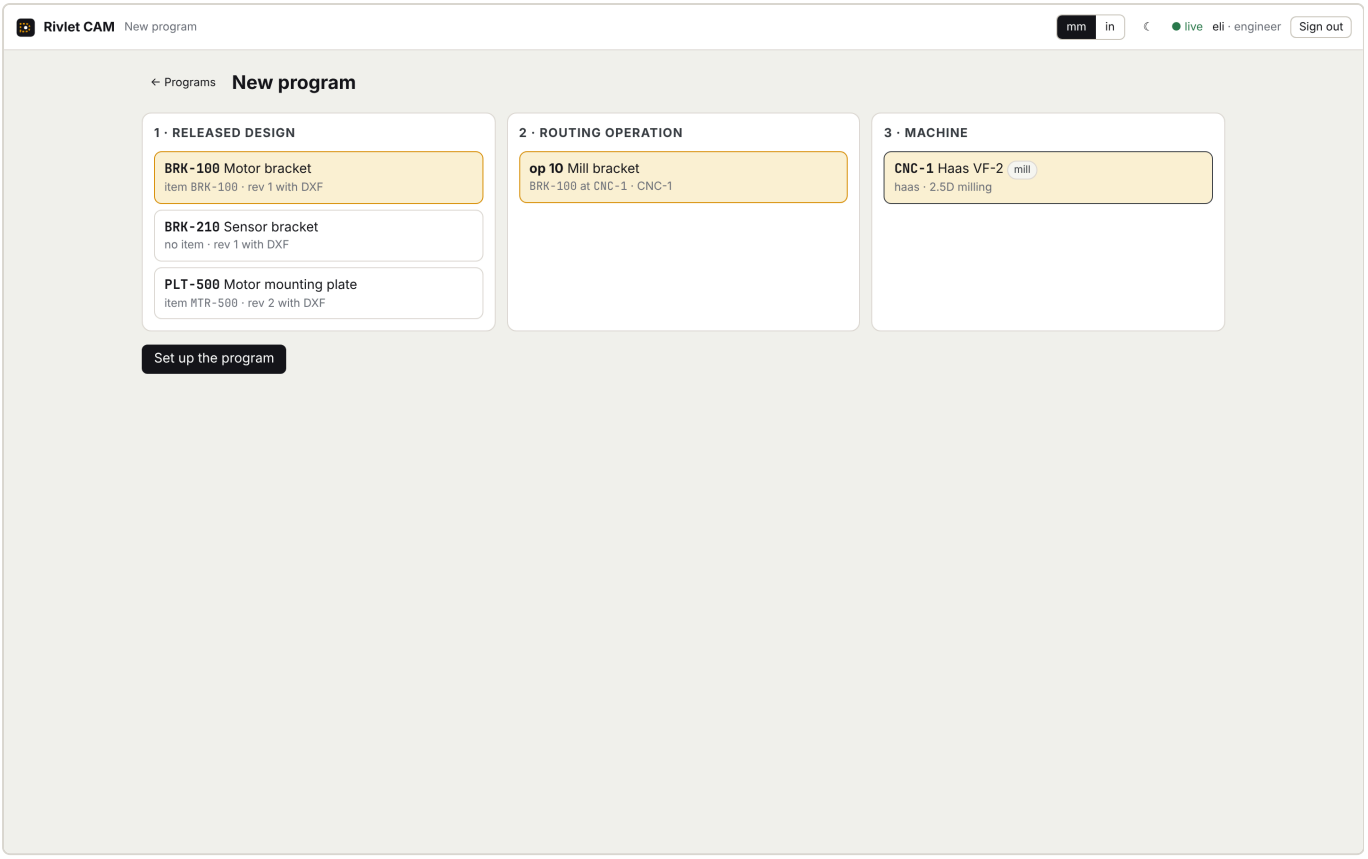


Programs by machine. Programs are grouped by machine, with the program number, released revision, whether it was confirmed loaded and the work orders queued for it. `design-cam-programs-eli-desktop`

CAM: making a mill program

To make a program

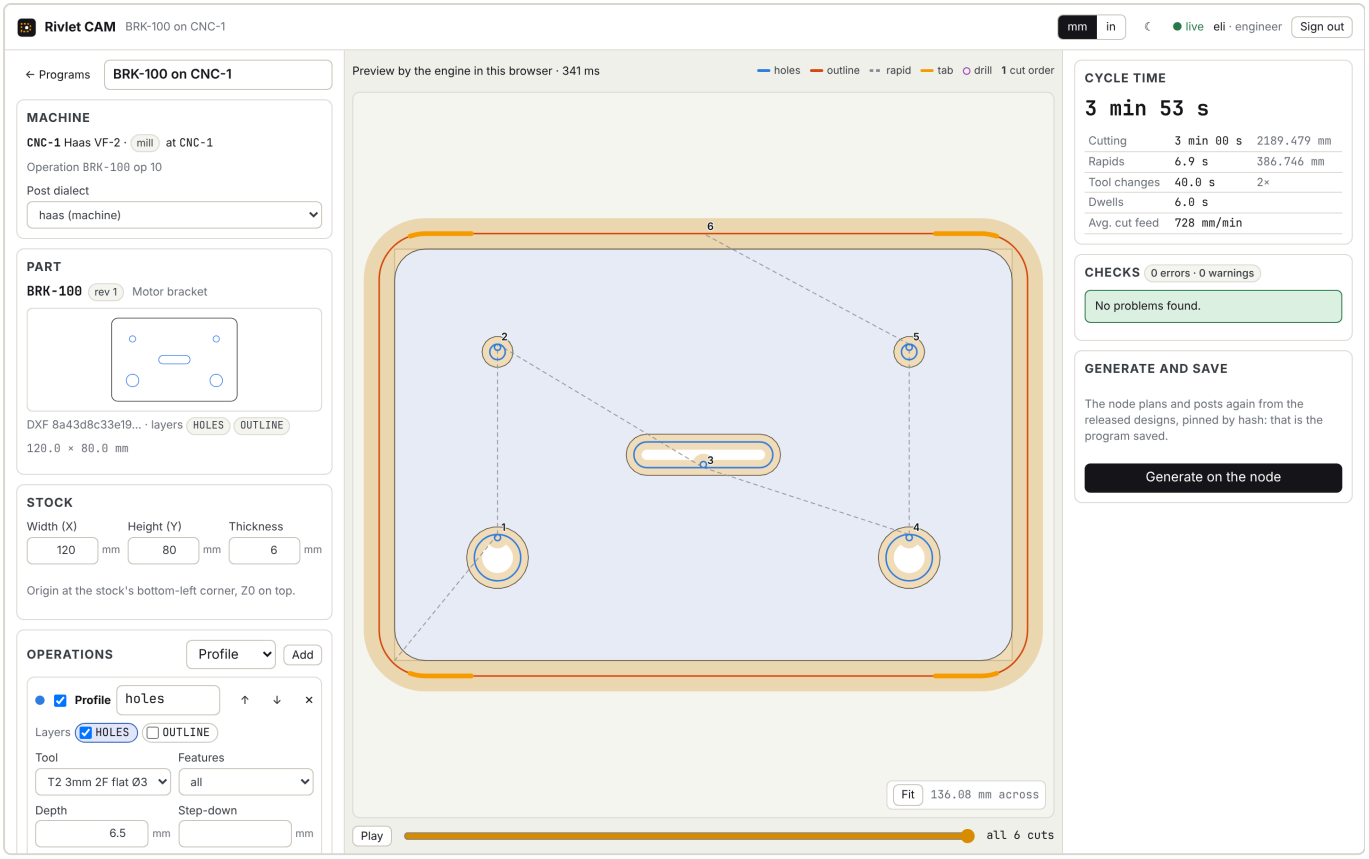
1. On the programs list, press **New program**. Pick a released design.
2. Pick the routing operation it runs at.
3. Pick a machine at that operation's station. Then **Set up the program**.



Start a program. A program starts from a released design, the routing operation it runs at, and a machine at that operation's location. `design-cam-start-eli-desktop`

To read the preview

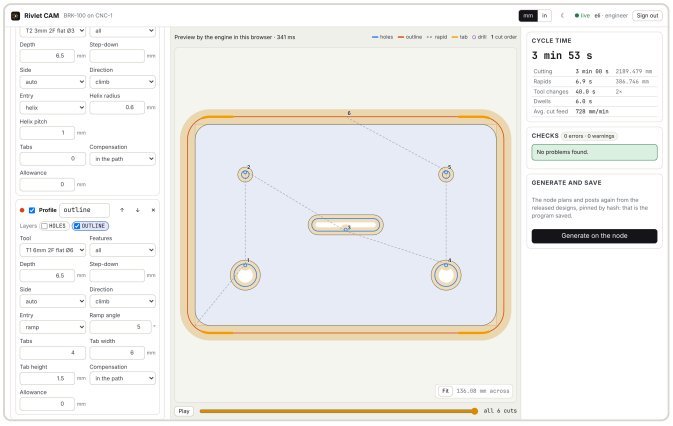
The engine plans the toolpath in your browser, about a quarter of a second after each change. Rapid moves are dashed; cuts are coloured by operation. The numbers give the cut order. The cycle time is broken down beside it.



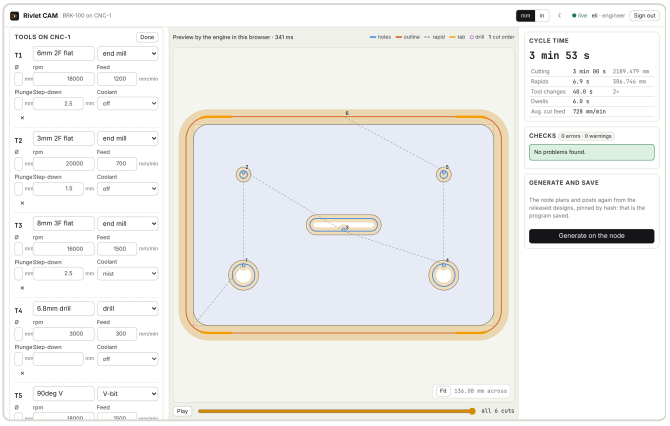
The workbench: setup, preview and cycle time. The engine plans the toolpath in the browser: rapids dashed, cuts coloured by operation, tabs, cut-order numbers, and the cycle time broken down beside it. `design-cam-workbench-eli-desktop`

To set up the operations

The setup has the machine and its post, the part and its layers, the stock, the operations and the machine's tool library. The default operations come from the DXF layers: the holes first, with a small end mill on a helix, then the outline with a ramp and four tabs. Change a value (six tabs instead of four, say) and the preview follows. Lengths can be shown in inches; the engine still works in millimetres.



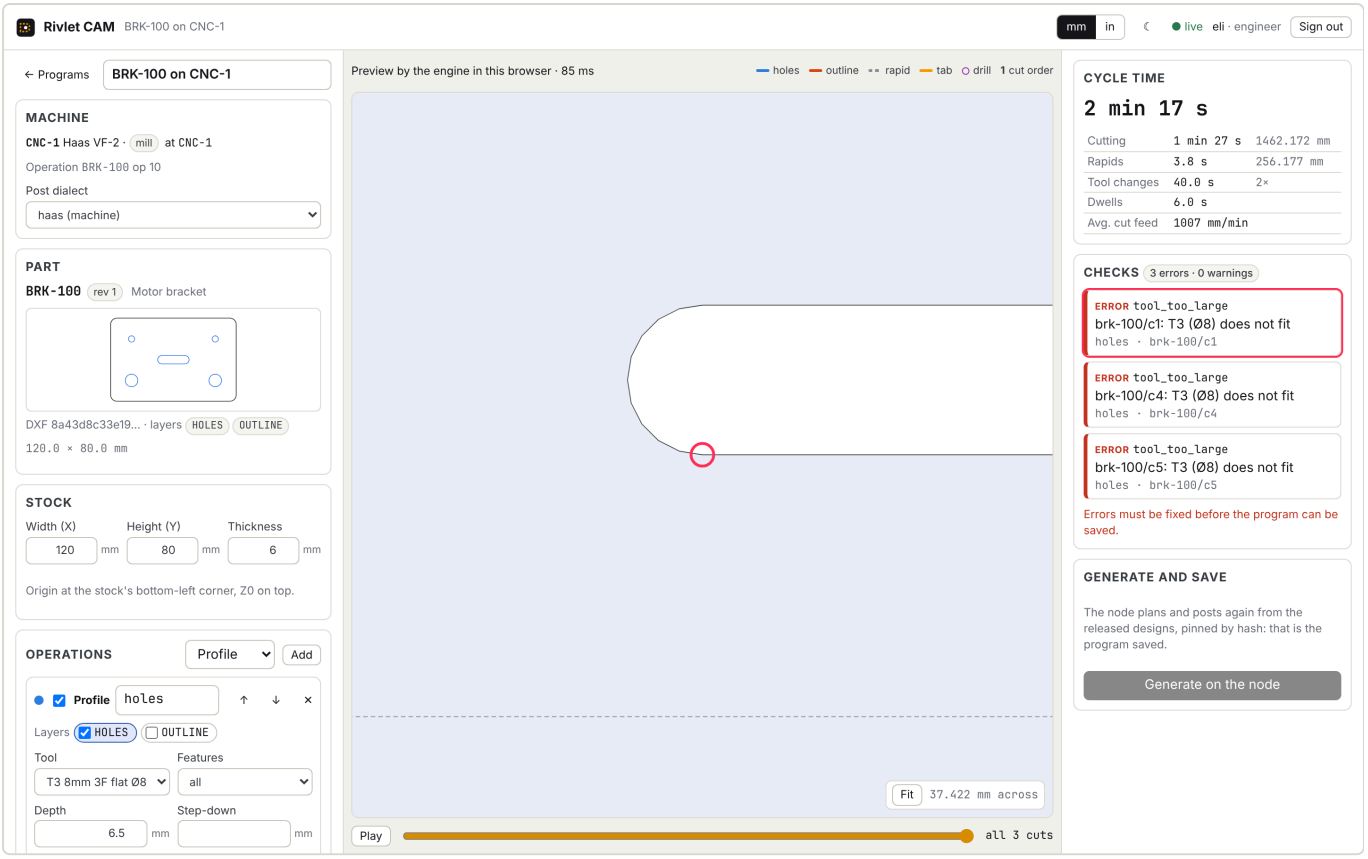
Operations. Default operations come from the DXF layers: the holes with a small end mill on a helix, then the outline with a ramp and four tabs. `design-cam-operations-eli-desktop`



The machine's tool library. Each machine has a tool library (kept on this device) that the operations pick their cutters from. `design-cam-tools-eli-desktop`

To fix a check

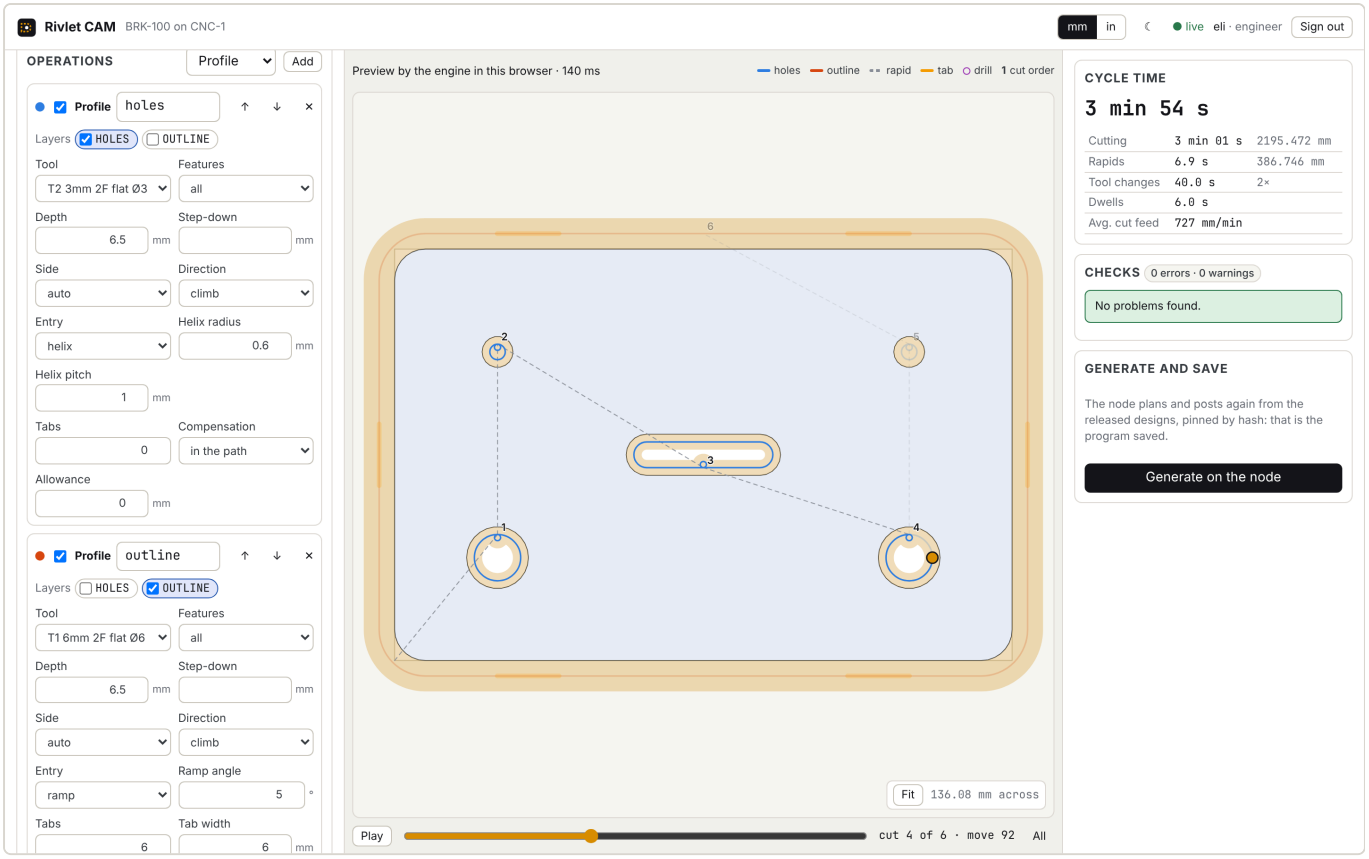
Checks sit beside the preview. Errors block **Generate**: for example an 8 mm cutter does not fit 6 mm holes. Click a check to see the moves it is about. Fix the cause (here, pick the 3 mm cutter again) and the error goes.



A check that blocks generating. An 8 mm cutter does not fit the 6 mm holes: the error is marked on the drawing and Generate is disabled until it is fixed. `design-cam-check-error-eli-desktop`

To walk through the cuts

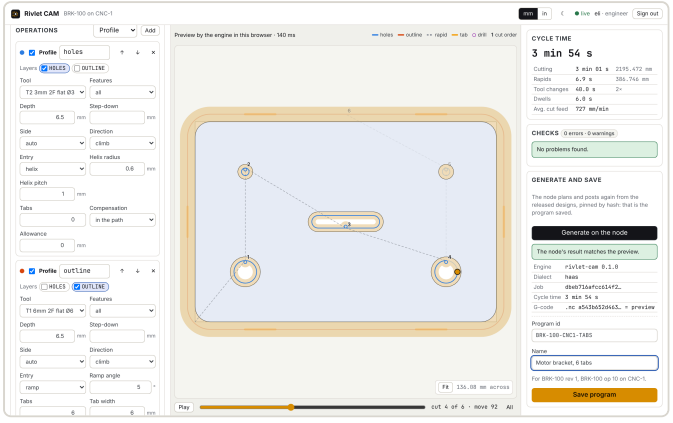
Drag the scrubber, or press **Play** (and **Pause**). Finished cuts are solid, the rest faded, and the tool is where it is.



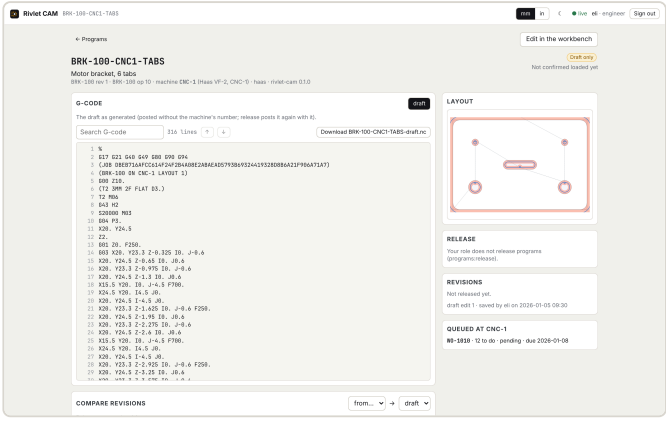
Scrubbing the cut order. The scrubber walks the cut order: finished cuts solid, the rest faded, and the tool where it is. `design-cam-scrub-eli-desktop`

To generate and save

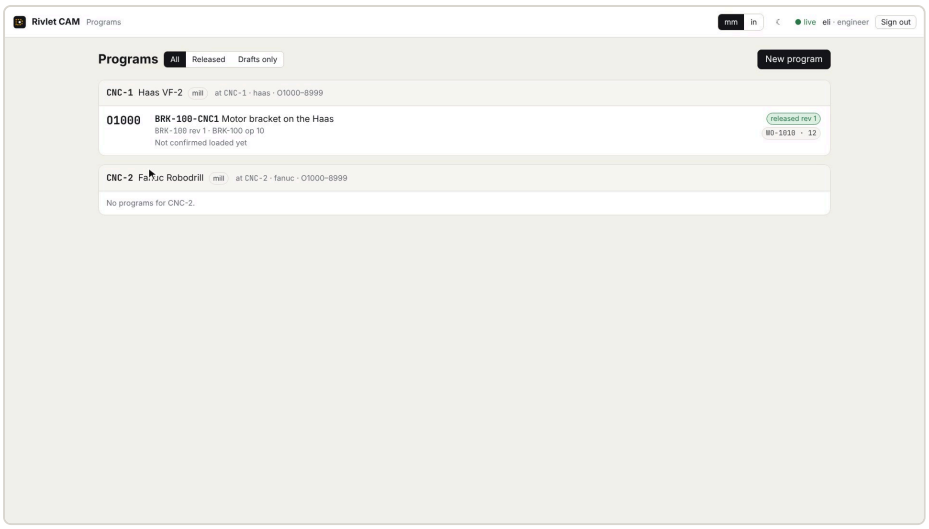
1. Press **Generate on the node**. The node plans again from the released design and says whether its result matches the preview. The node's result is what is saved.
2. Give the program an id and a name.
3. Press **Save program**. It is a draft until a plant manager releases it.



Generated on the node. The node plans and posts the program again from the released design and confirms its result matches the preview, then you name and save it. `design-cam-generated-eli-desktop`



The saved program (draft). A saved program is a draft until an approver releases it; an engineer can see it but not release it. `design-cam-program-draft-eli-desktop`



Movie: Make a mill program (1 min 37 s). Eli makes a CNC-1 program for the BRK-100 bracket: pick the design, operation and machine, read the preview and cycle time, change the tabs, fix a check, scrub the cut order, generate on the node and save. `design-make-a-mill-program`

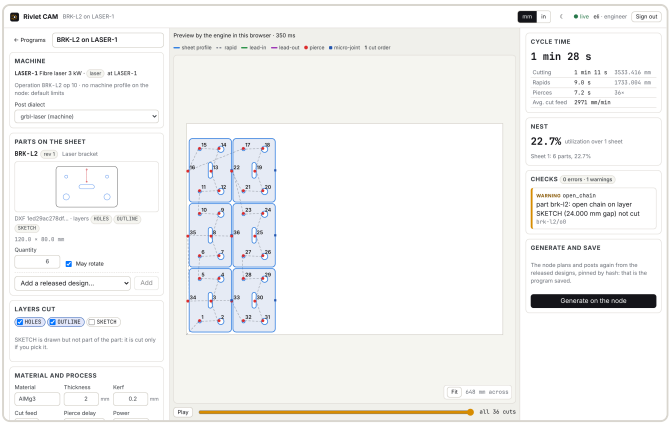
Chapters: Programs 0:00 · Pick the design, operation and machine 0:06 · Read the preview 0:17 · Set up the operations 0:30 · Fix a check 0:44 · Scrub the cut order 1:02 · Generate and save 1:15

CAM: nesting a sheet

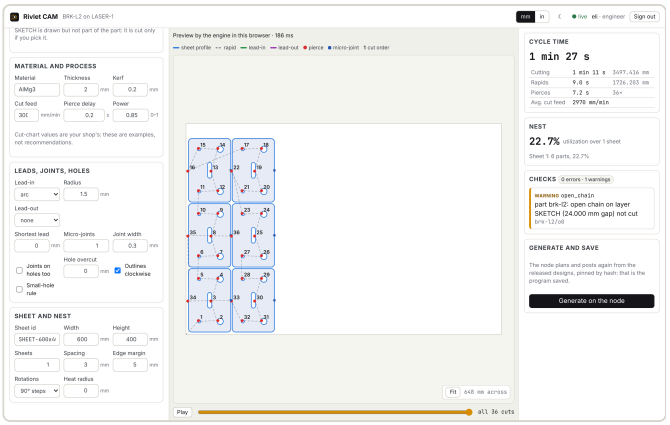
A sheet machine, such as a laser, cuts parts from a sheet. Pick a sheet machine when you start a program, and the workbench becomes a nest:

- the parts and their quantities, the material and process;
- lead-ins and lead-outs, micro-joints that hold each part in the sheet, and the small-hole rule;
- the sheet size and count, and the nest settings;
- the layers it cuts: every layer of the drawings but SKETCH (what is drawn but not part of the part), as for a mill. Tick SKETCH to cut it too.

Utilization says how much of the sheet becomes parts. More parts than one sheet holds spill onto a second; switch between sheets above the preview. Each sheet is saved as its own program (`-S1` , `-S2`): the button reads **Save 2 programs**.



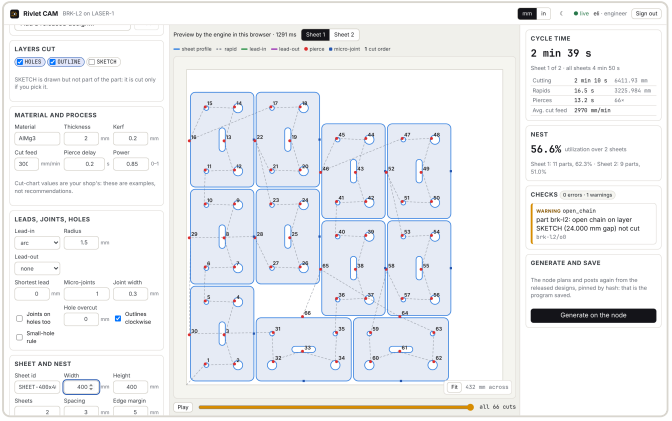
Nesting on one sheet. Six brackets nested on a 600 × 400 sheet for the laser, with pierce points, lead-ins, micro-joints and the sheet utilization. `design-cam-sheet-nest-eli-desktop`



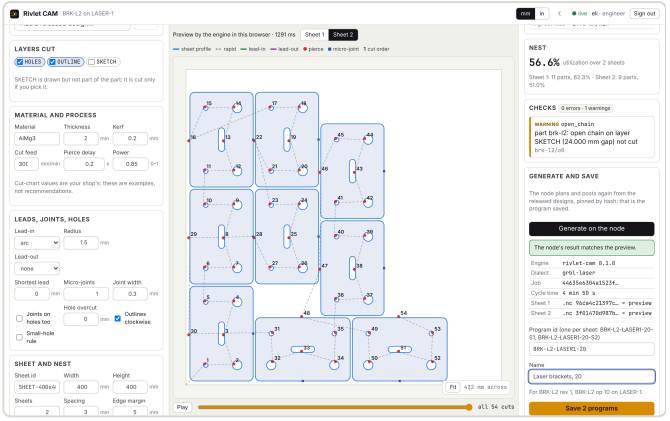
Leads, joints and holes. Lead-in and lead-out, micro-joints and the small-hole rule are set per job; the preview follows each change. `design-cam-sheet-leads-eli-desktop`



The layers a laser cuts. Layers cut: the laser cuts OUTLINE and HOLES; SKETCH (here a centre mark, drawn but no part of the part) is cut only if you pick it. `design-cam-sheet-layers-eli-desktop`

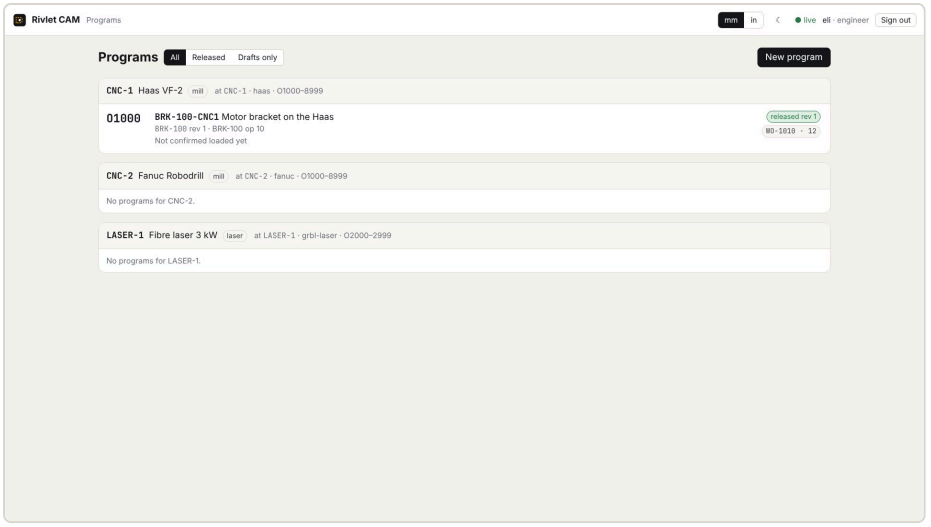


Nesting over two sheets. Twenty brackets do not fit one 400 × 400 sheet, so the nest uses two; the tabs above the preview switch between them. `design-cam-sheet-two-eli-desktop`



One program per sheet. A nest over two sheets saves as two programs, -S1 and -S2. `design-cam-sheet-save-two-eli-desktop`

The sheet's id names its size, for example SHEET-600×400, and follows the size when you change it (SHEET-400×400). An id you type yourself, such as a stock code, stays.



Movie: Nest a sheet (1 min 28 s). Eli nests laser brackets on a sheet: quantities, lead-ins, micro-joints, utilization, then twenty parts over two sheets saved as two programs. `design-nest-a-sheet`

Chapters: Start a sheet program 0:00 · Read the nest 0:11 · Leads and joints 0:25 · More parts than one sheet holds 0:36 · Generate and save 1:04

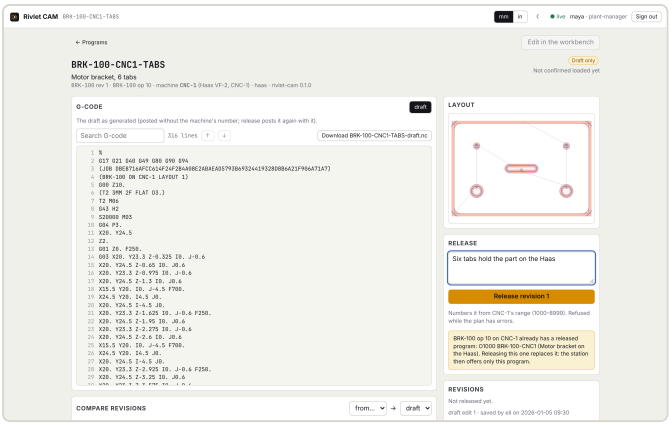
CAM: releasing and revising a program

To release a program (plant manager)

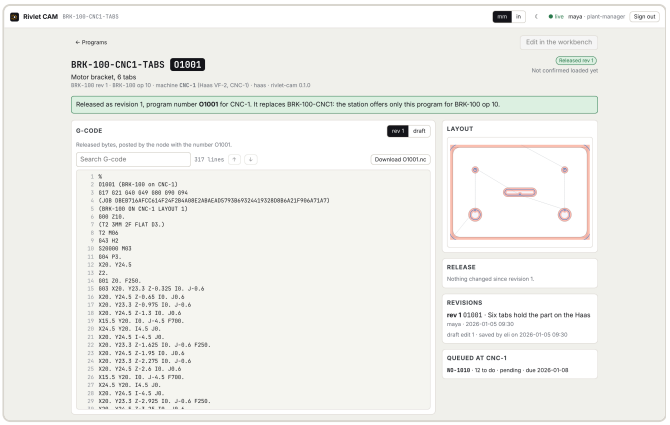
1. Open the draft program.
2. Write a note on what is approved.
3. Press **Release revision 1** (or **Release revision 2**, and so on). The machine's next program number is given, for example O1001.

One program is the one to run for each operation on each machine. If the operation already has a released program on that machine, the release panel warns that releasing this one replaces it: the station then offers only the new program. The replaced program reads "Replaced by O1001". To make it the one to run again, release it again, unchanged. (The sheets of one nest are not replaced by each other.)

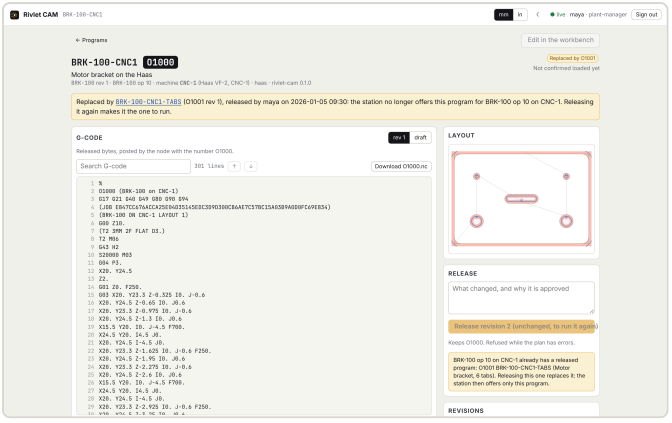
The released G-code is shown with line numbers. You can search it and download it (as `O1001.nc`).



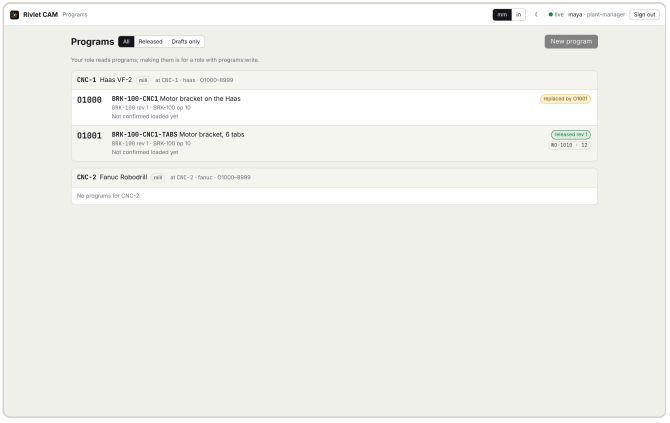
Release a program. An approver releases the draft with a note; the machine's program number is assigned on release. The panel warns that the operation already has O1000 on CNC-1, which this release replaces. `design-cam-release-maya-desktop`



Released as O1001. Released as revision 1 with program number O1001 for CNC-1; the released G-code is shown with line numbers. `design-cam-released-maya-desktop`



The program it replaced. O1000, the program the release replaced, reads "Replaced by O1001": the station no longer offers it for BRK-100 op 10 on CNC-1, and the bridge no longer delivers it. Releasing it again would make it the one to run. `design-cam-replaced-maya-desktop`



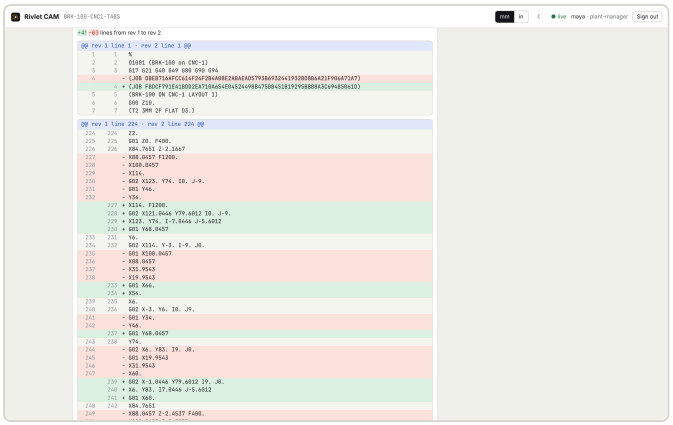
Programs: which one runs. In the programs list the replaced program is tagged "replaced by O1001", so it is clear which program the machine runs. `design-cam-programs-replaced-maya-desktop`

At CNC-1, the job card then offers O1001 and says it replaces O1000: see On the floor.

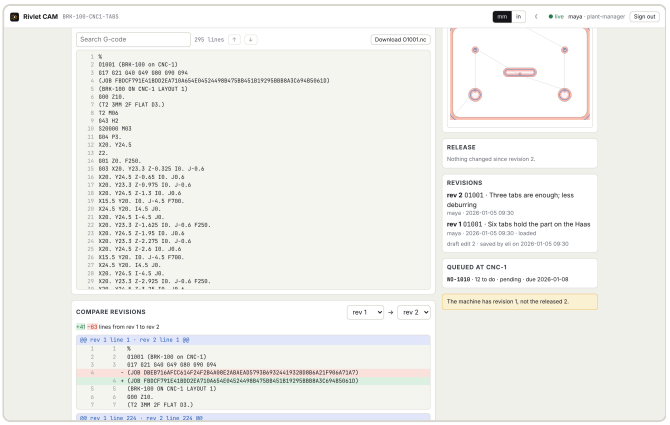
To revise a program

1. Open the program and press **Edit in the workbench**. The setup it was saved from comes back.
2. Make the change and save on the same program. It is a new draft.
3. A plant manager releases it as revision 2. The number stays the same.

Compare shows what changed in the G-code between two revisions. The revisions keep each note, who released it and when.

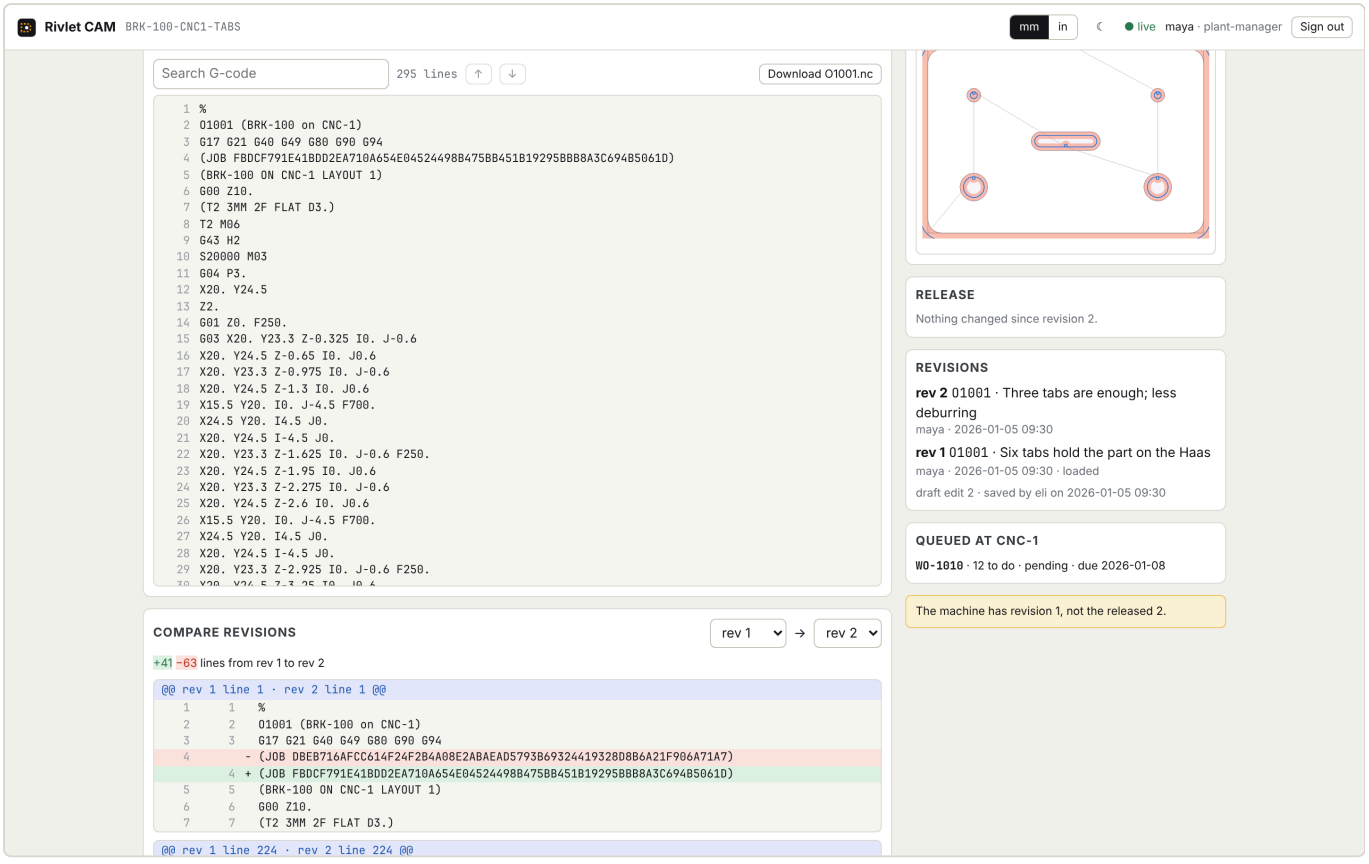


Compare revisions. The G-code diff between revision 1 and 2 of O1001: the tab moves and the job hash changed. `design-cam-diff-maya-desktop`

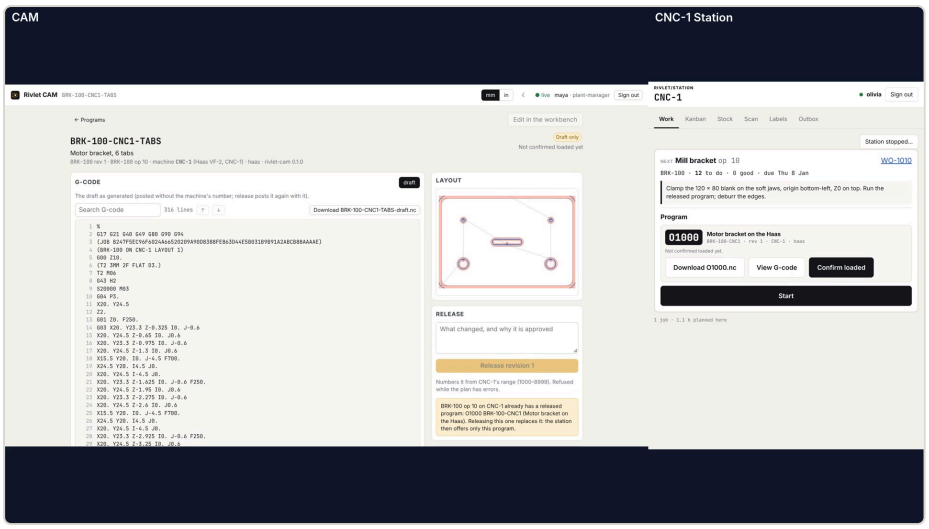


Revisions. Each release is kept as a revision with its note, who released it and when; the number stays O1001. `design-cam-revisions-maya-desktop`

The program page shows which revision the machine was last confirmed with, and warns when it is not the released one. At the machine, the job card warns too, until the operator loads the new revision. See On the floor.



Loaded state in CAM. The program page shows which revision the machine was last confirmed with, and warns when it is not the released one. `design-cam-loaded-maya-desktop`



Movie: Release and revise a program (1 min 22 s). Maya releases a draft as O1001, which replaces O1000 for the operation; at CNC-1 the job offers O1001 and says it replaces O1000; Olivia confirms it loaded; Eli changes the tabs; Maya releases revision 2 and compares it with revision 1; the tablet warns the machine still has revision 1 until Olivia loads the new one. `design-release-and-revise`

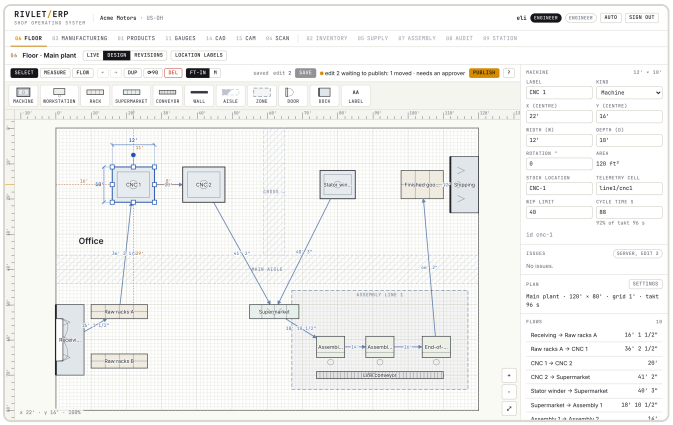
Chapters: Release 0:00 · Load it at the machine 0:14 · Edit 0:29 · Release revision 2 0:50 · Compare revisions 0:59 · The machine still has revision 1 1:09

The floor designer

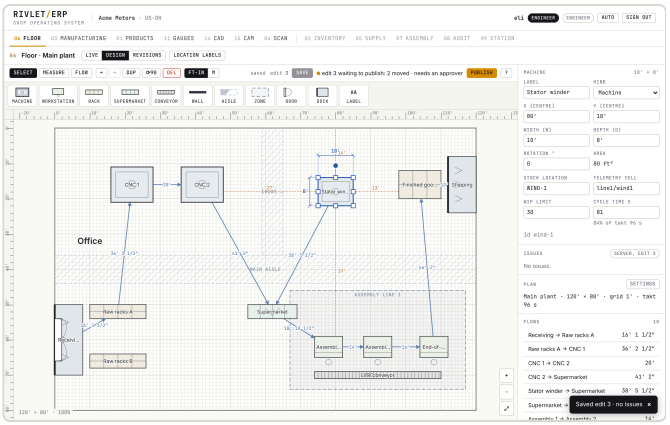
06 Floor has three tabs: **Live**, **Design** and **Revisions**. Engineers start in the designer.

To change the floor layout

1. On the **Design** tab, drag an item from the palette (machines, racks, benches and more) onto an empty spot on the plan.
2. Select it to edit it in the inspector: its size, position, station and cell. Name it.
3. Press **Save**. This keeps your draft. The designer says your edit is waiting to publish, and what changed.



Designer, a machine selected. In the designer, selecting a machine shows its size, position, station and cell in the inspector. `shell-floor-designer-selected-eli-desktop`

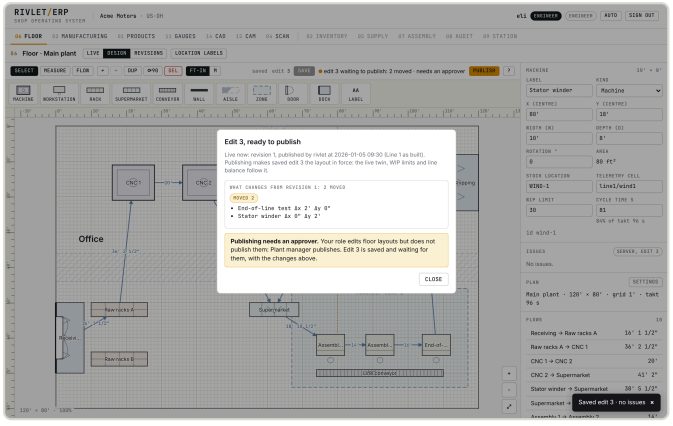


A saved edit waiting to publish. Eli saved a change; the designer says the edit is waiting to publish and needs an approver. `shell-floor-publish-waiting-eli-desktop`

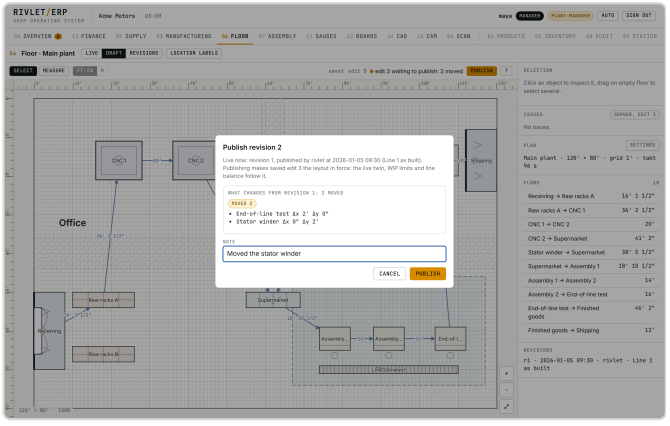
To publish it

Publishing makes the draft the live layout. It needs a plant manager.

1. Press **Publish**. The dialog lists what changes from the live revision: added, removed, moved (and how far), changed, and flows.
2. An engineer sees the same list, and "Publishing needs an approver: a Plant manager publishes it; your edit is saved and waiting."
3. A plant manager writes a note and publishes. The draft becomes the next revision, and the live floor runs on it.

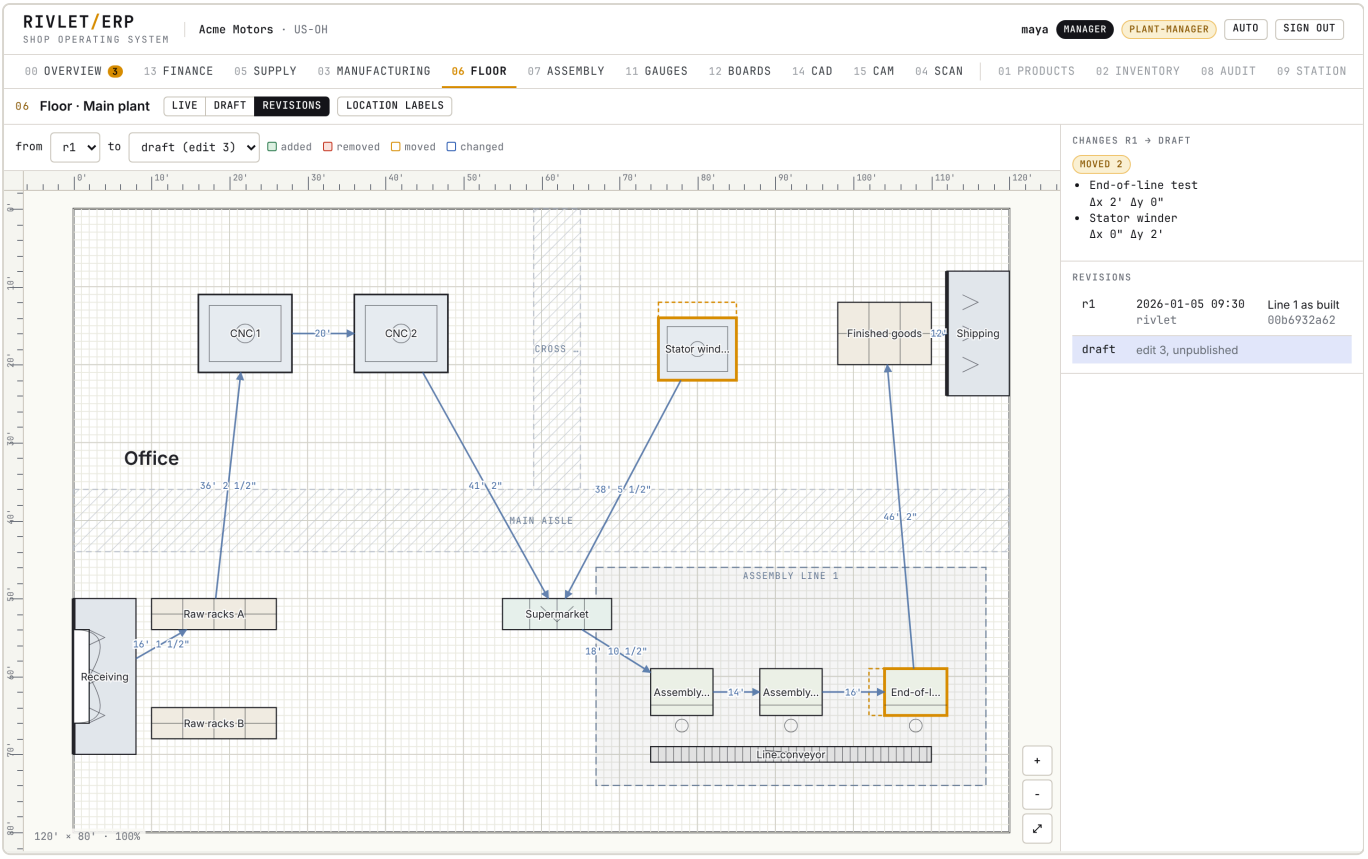


Publish: who publishes. An engineer's Publish shows what would go live against the live revision and names who publishes it (a Plant manager): his edit is saved and waiting. `shell-floor-publish-needs-approver-eli-desktop`

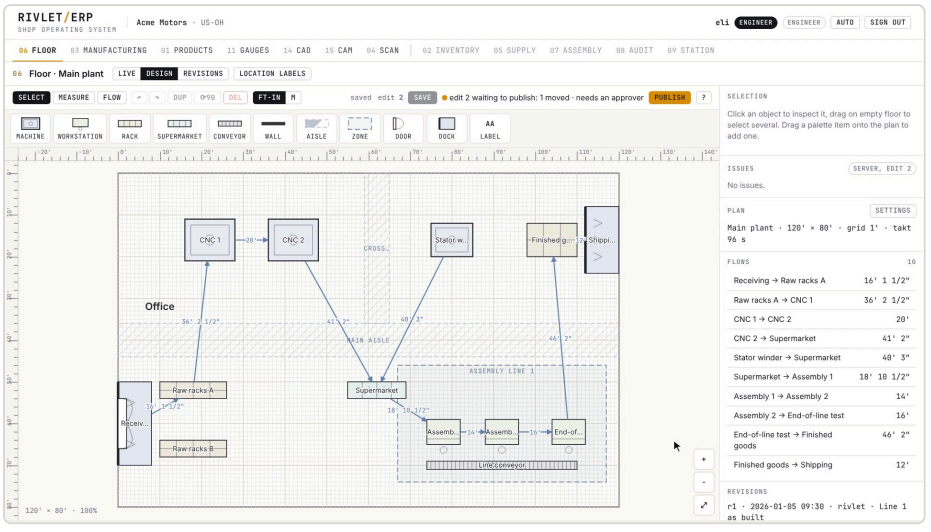


Publish a revision, with what changes. Before publishing, the dialog lists what goes live against the live revision: added, removed, moved and how far, changed, and flows. `shell-floor-publish-modal-maya-desktop`

Revisions compares any two published layouts, or the draft with the last revision.



Floor revisions. Revisions compares two published layouts, or the draft with the last revision. `shell-floor-revisions-maya-desktop`



Movie: Lay out the floor and publish it (1 min 18 s). Eli drops a new machine on the plan, names it and saves the draft; his Publish shows the change and that a plant manager publishes it; Maya's Publish lists what goes live against revision 1, she publishes revision 2 and compares it with revision 1. `shell-lay-out-the-floor`

Chapters: The designer 0:00 · Place a machine 0:07 · Save the draft 0:20 · Publish 0:40 · Compare revisions 0:59

Items and routings

Engineers may edit items and their routings on **01 Products**: each operation's station, setup and cycle times, instructions and checks. See Planning and Quality.

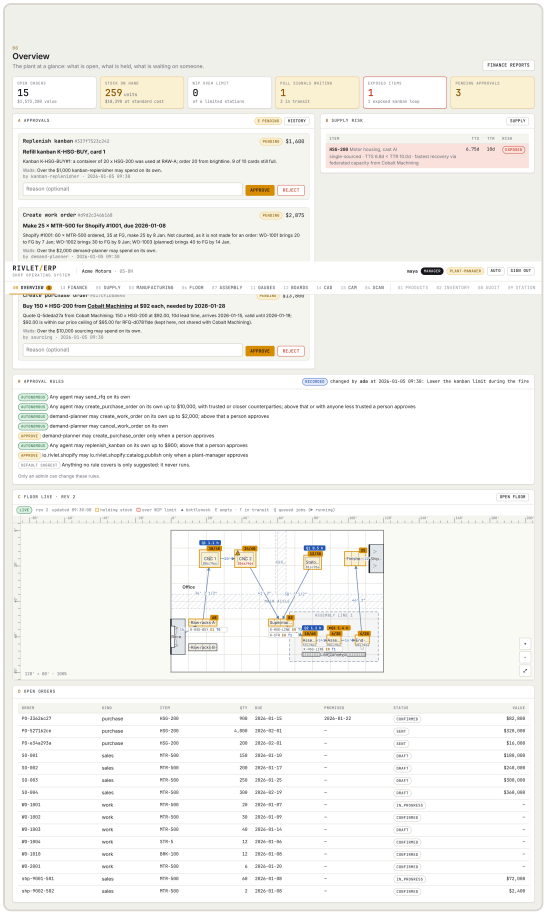
Managers and the CFO

This chapter is for the plant manager and the CFO, and for everyone who approves. It covers the Overview, approving and its limits, what agents may do on their own, who sees money, and the finance reports.

Your Overview

You start on **00 Overview**: the plant at a glance.

- **The tiles:** Open orders, Stock on hand, WIP over limit, Pull signals waiting, Exposed items and Pending approvals. Click one to open its section. With finance rights, the tiles also show values. Under Exposed items, the tile counts the kanban loops at risk too, for example "1 exposed kanban loop".
- **A Approvals:** what waits for a decision.
- **B Supply risk:** what is exposed if a supplier cannot deliver, and for how long.
- **R Approval rules:** the plant's rules, read-only, in plain words.
- **C Floor live:** every station, its queue and the bottleneck.
- **D Open orders.**



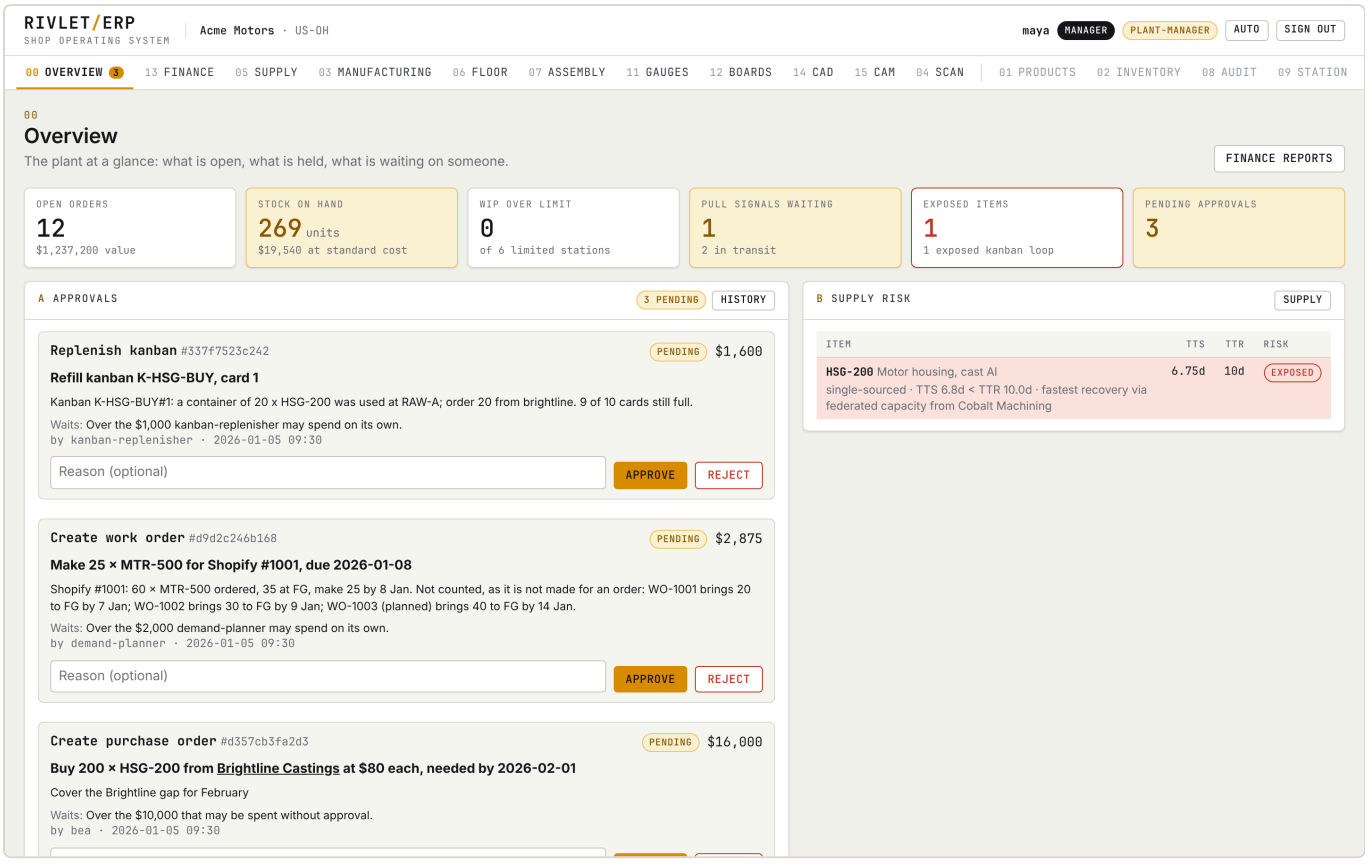
Overview, whole page. The whole Overview page for the plant manager. `shell-overview-full-maya-desktop`

Approving and rejecting

The plant's rules hold some actions until a person decides. In the demo:

- a purchase order over $10,000, or with a partner that is not trusted;
- a kanban refill over $1,000;
- work the demand planner wants to make over $2,000;
- anything the demand planner wants to buy;
- an app's own action, when its rule says so;
- anything a person proposes: a person's own proposal always waits for someone else.

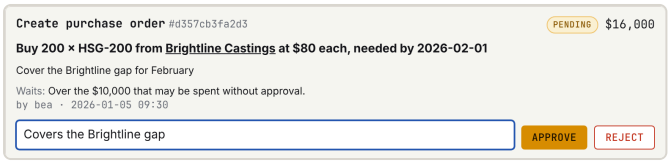
The number on **00 Overview** counts what waits for you. Only people who may approve see it.



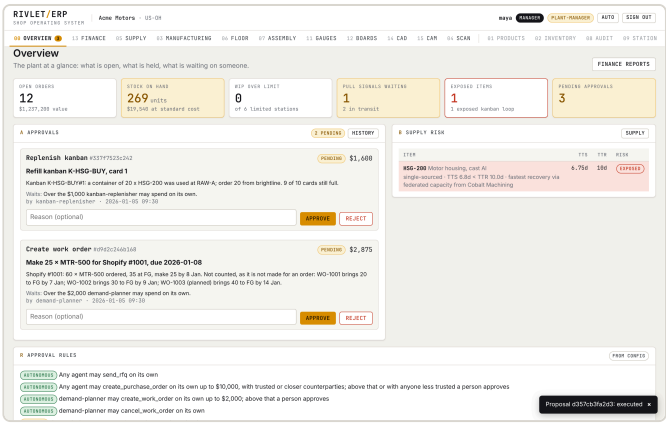
Purchases waiting for Maya. The Overview puts what is waiting for a decision first: Bea's purchase order, a kanban refill and the demand planner's work order for a web order. Each card says what it does and why it waits, in words. `shell-overview-pending-approval-maya-desktop`

To approve or reject

1. Find the card in **A Approvals** (on 00 Overview, or **C Approvals** on 05 Supply). It shows the action, its value, its details, the rationale, who proposed it and why it waits.
2. Type a reason. It is kept with the decision, and the proposer reads it.
3. Press **Approve** or **Reject**. A note confirms it, and the card leaves the queue.



Approve with a reason. Maya types a reason, which is kept with the decision, then presses Approve. `shell-approvals-reason-maya-desktop`

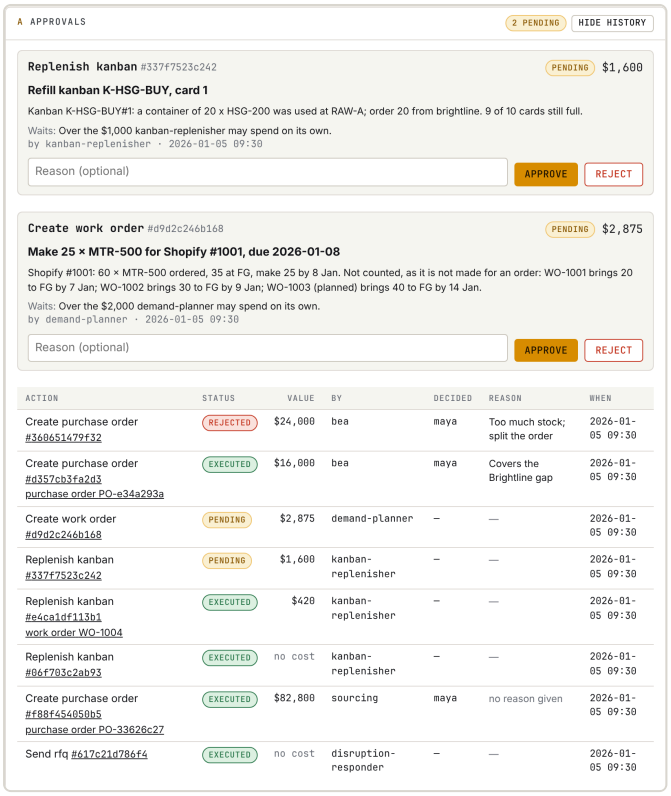


Approved. The order is approved and executed; a toast confirms it and the card leaves the queue. `shell-approvals-approved-toast-maya-desktop`

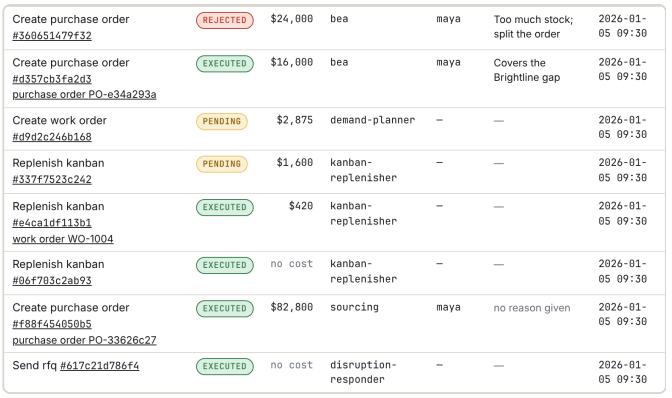
Each card says in words what the action does and why it waits for you, for example "Buy 200 × HSG-200 from Brightline Castings at $80 each, needed by 2026-02-01", then "Waits: Over the $10,000 that may be spent without approval." A kanban refill reads "Refill kanban K-HSG-BUY, card 1" and waits because it is "Over the $1,000 kanban-replenisher may spend on its own"; work for a web order says which order it is for: "Make 25 × MTR-500 for Shopify #1001". A supplier or partner factory shows by its name: hover over it for its supplier id or DID (`did:…`), and a partner's name opens its page. Hover over the "Waits" line for the rule as the system records it. Without finance rights the amounts are left out ("Over what kanban-replenisher may spend on its own"). Open the proposal (its # link) to read it as a record.

The history

Press **History** in the Approvals panel. It lists recent proposals: status, value, who proposed, who decided, the reason, and the record an approval made (a PO or a work order, as a link). A rejected proposal stays in the history in red, with its reason. A withdrawn one (whoever proposed it took it back while it waited: a person, an app or the demand planner) says who withdrew it and why.



Approvals history. History lists recent proposals with status, value, who proposed, who decided, the approver's reason and the record an approval made. `shell-approvals-history-maya-desktop`



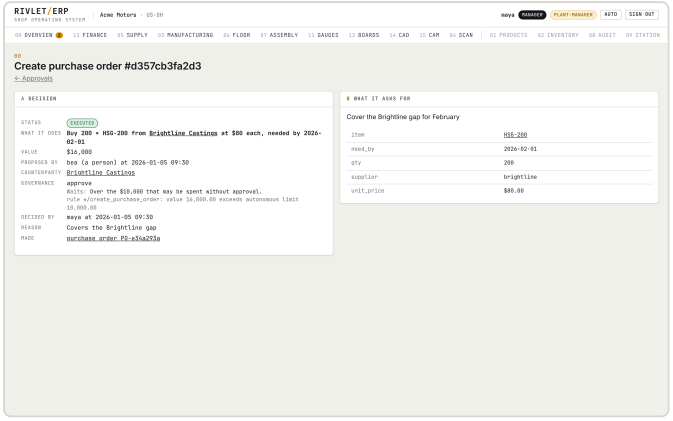
History with a rejection and its reason. A rejected proposal stays in the history in red, with the reason the buyer reads. `shell-approvals-history-rejected-maya-desktop`

ACTION	STATUS	VALUE	BY	DECIDED	REASON	WHEN
Create purchase order #12557aabeb98	WITHDRAWN	$12,000	bea	withdrawn by bea	Brightline can deliver after all	2026-01-05 09:30
Create work order #a6875dad5833	WITHDRAWN	$4,600	demand-planner	withdrawn by demand-planner	Shopify #1007 was cancelled	2026-01-05 09:30
Publish products to Shopify #4ae445b74cb4	EXECUTED	no cost	Shopify app	maya	Spring catalogue checked	2026-01-05 09:30
Create purchase order #85f60fc096b5 purchase order PO-1ebb50c3	EXECUTED	$320,000	bea	finn	Approved for the 2026 budget	2026-01-05 09:30
Create purchase order #86f8d2a8f7a4	PENDING	$13,800	sourcing	–	—	2026-01-05 09:30
Send rfq #7f8da4d1ba01	EXECUTED	no cost	bea	maya	Ask Cobalt for the second batch	2026-01-05 09:30
Create purchase order #29432cf69051	REJECTED	$24,000	bea	maya	Too much stock; split the order	2026-01-05 09:30
Create purchase order #da6889ba2e29 purchase order PO-8921a645	EXECUTED	$16,000	bea	maya	Covers the Brightline gap	2026-01-05 09:30
Create work order #2f1511391abb	PENDING	$2,875	demand-planner	–	—	2026-01-05 09:30
Replenish kanban #c721ccac3c8c	PENDING	$1,600	kanban-replenisher	–	—	2026-01-05 09:30
Replenish kanban #24cc4f50a91d work order WO-1004	EXECUTED	$420	kanban-replenisher	–	—	2026-01-05 09:30
Replenish kanban #82fc73ea7a51	EXECUTED	no cost	kanban-replenisher	–	—	2026-01-05 09:30
Create purchase order #811518a2d5cd purchase order PO-71ecfed4	EXECUTED	$82,800	sourcing	maya	no reason given	2026-01-05 09:30
Send rfq #815155efee0e	EXECUTED	no cost	disruption-responder	–	—	2026-01-05 09:30

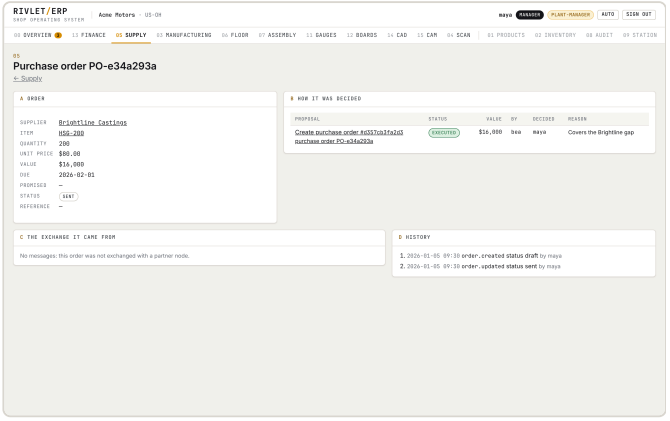
Withdrawn proposals in the history. Withdrawn while they waited: the demand planner took back its work for Shopify #1007 ("Shopify #1007 was cancelled"), and Bea her purchase order ("Brightline can deliver after all"). Each says who withdrew it and why. `shell-approvals-history-withdrawn-maya-desktop`

A proposal that spends nothing, such as a kanban refill moved from the supermarket, reads **no cost** in the Value column.

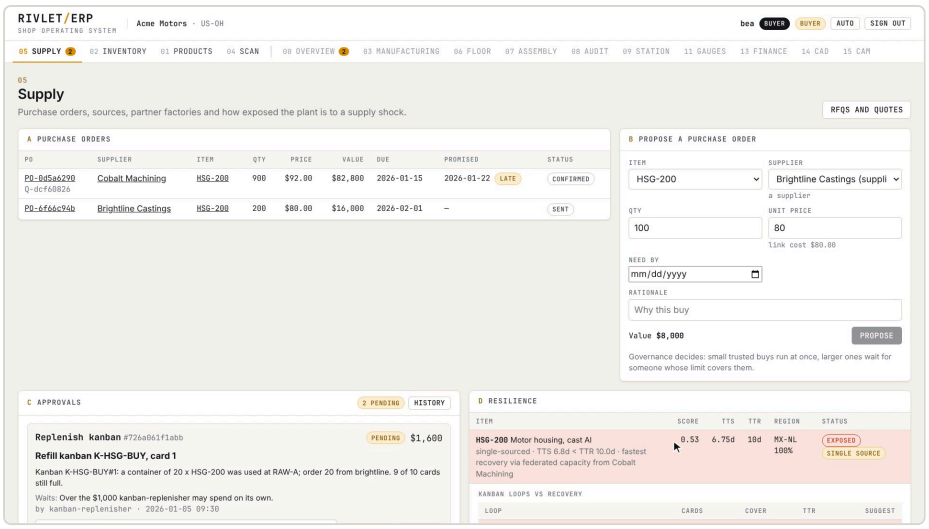
Click a proposal's # to open it as a record: who proposed it and why, the rule that held it, who decided with what reason, its details, and the record it made.



A proposal opened. A proposal as a record: who proposed it and why, the rule that held it, who decided with what reason, its arguments, and the purchase order it made. `shell-record-proposal-maya-desktop`



The purchase order it made. The PO an approval made: supplier, item, quantity, price, the proposal behind it and its history. `shell-record-po-from-approval-maya-desktop`



Movie: Propose and approve a purchase (1 min 35 s). Bea proposes a $16,000 purchase order; it waits for someone else; Bea cannot approve her own; Maya approves it with a reason; the history shows who decided and why, opens the PO it made, and shows work the demand planner withdrew when its web order was cancelled. `shell-approve-a-purchase`

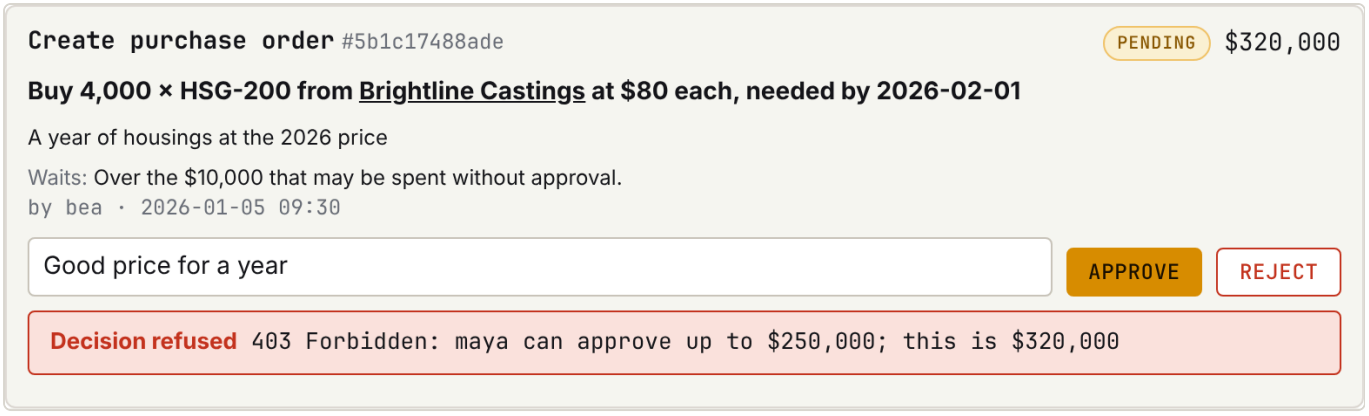
Chapters: Propose 0:00 · It waits 0:28 · Not your own 0:34 · Approve 0:45 · History 1:10

Approval limits

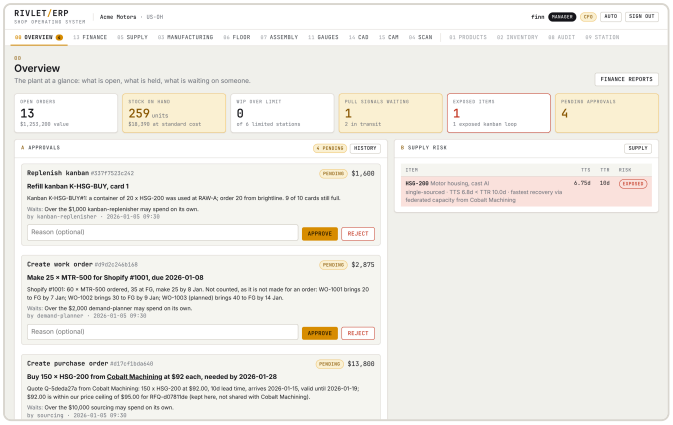
Each role approves up to its limit:

Role	Approves up to
Station manager	$1,000
Line supervisor	$5,000
Planner	$10,000
Buyer	$25,000
Plant manager	$250,000
CFO / finance	no ceiling

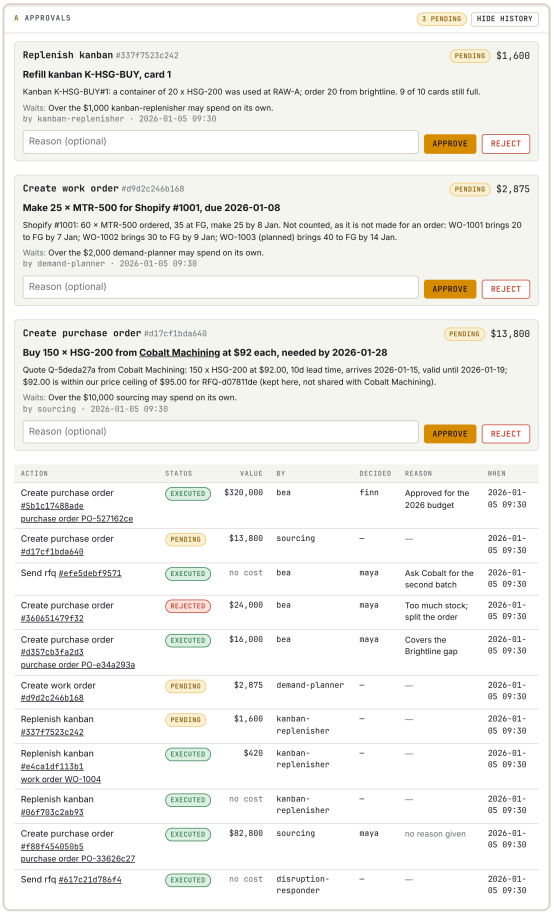
Something over your limit is refused when you approve it, and the card says so: "Decision refused 403 Forbidden: maya can approve up to $250,000; this is $320,000". Without finance rights you do not see what such a proposal is worth, and the refusal does not tell you either: "sam can approve up to $5,000; this is worth more". Leave it for someone whose limit covers it. The CFO has no ceiling, so what is over everyone else's limit comes to the CFO. So does a proposal that does not say what it is worth.



Over the plant manager's limit. A $320,000 purchase is over the $250,000 a plant manager may approve, so the node refuses Maya's approval and says so. `shell-approvals-over-limit-maya-desktop`



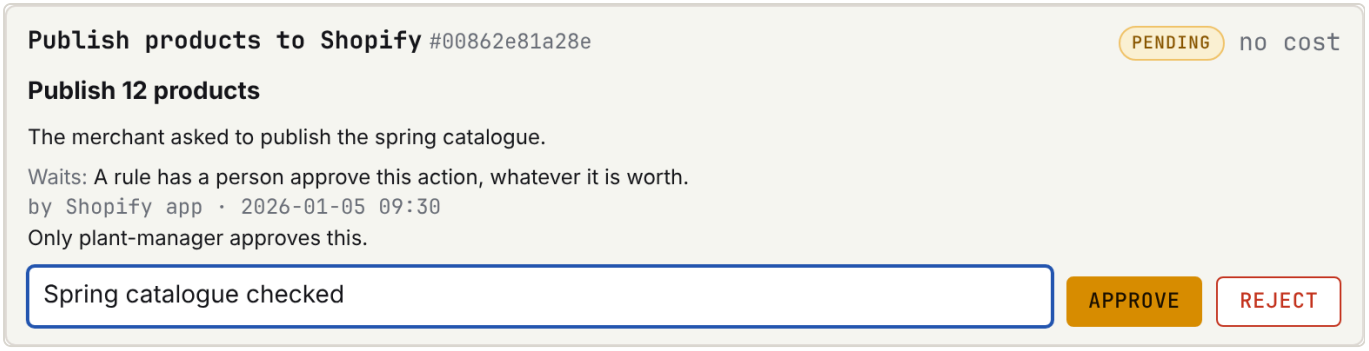
The CFO's overview. Finn, the CFO, lands on the same Overview; his role has no approval ceiling, so what is over everyone else's limit comes to him. `shell-overview-cfo-finn-desktop`



The CFO approves it. Finn approves the $320,000 order with a reason; the history shows it decided by finn. `shell-approvals-cfo-approved-finn-desktop`

Named approvers

A rule may name the roles that approve what it holds. The card then says, for example, "Only plant-manager approves this." **Approve** is off for anyone else, who may still reject it. Named approvers decide within their own limits.



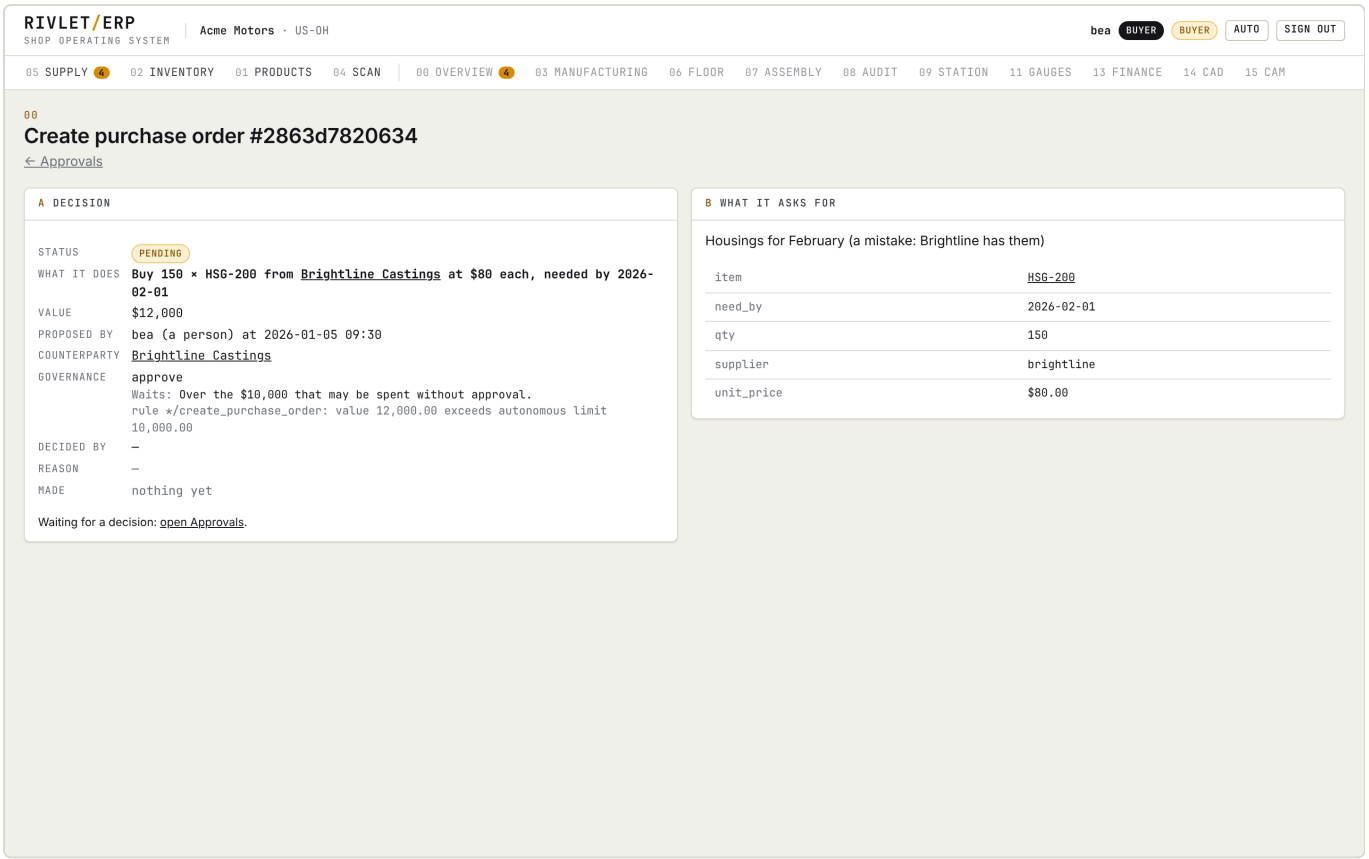
The plant manager decides. Maya, a plant manager, can approve the app's action; its value is $0, and still only her role decides it. `shell-approvals-named-approver-maya-desktop`

Your own proposals

Nobody approves their own proposal: "Decision refused · 403 Forbidden: policy: bea proposed this; someone else must approve it". This is the four-eyes rule.

While your own proposal waits, only you may withdraw it, with a reason. It leaves the queue, and the history shows it as withdrawn by you, with your reason.

The shell has no **Withdraw** button yet: a proposal is withdrawn through the node's API (`POST /proposals/{id}/withdraw` with a reason), as the Shopify app and the demand planner do. Until then, ask an approver to reject it, giving your reason.

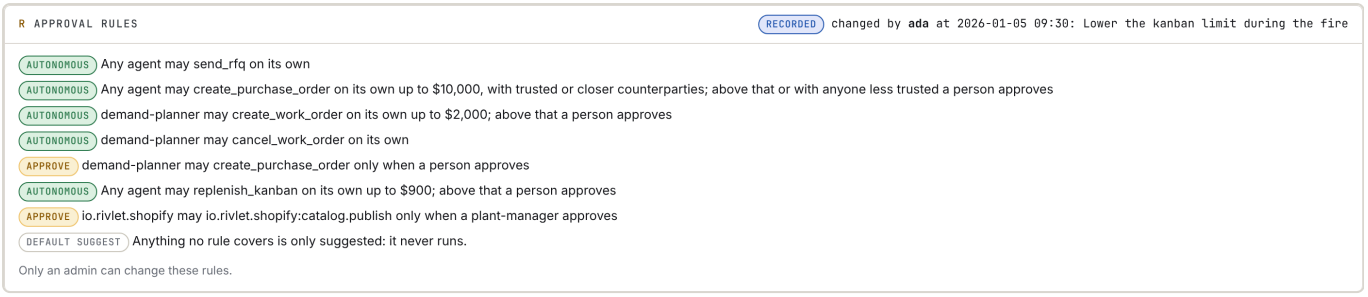


Your own proposal, waiting. Bea's own purchase order, waiting for someone else: its record says what it does, its value and why it waits. `shell-record-proposal-own-pending-bea-desktop`

People whose role does not approve see what is waiting, with "Your role does not approve proposals.", but cannot decide.

What agents may do on their own

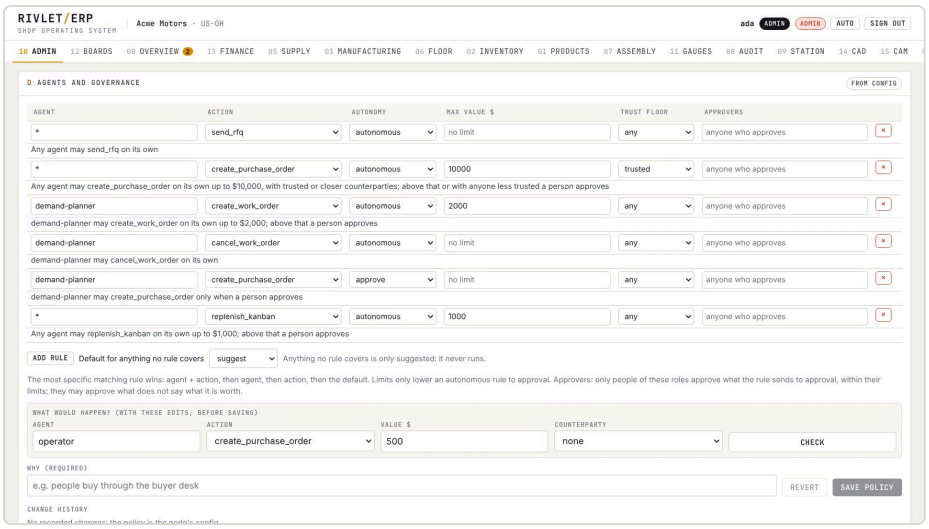
Agents never act directly. They propose, and the plant's rules decide what runs at once and what waits for a person. The rules are in **R Approval rules** on your Overview (read-only); the administrator edits them (see Administration).



Approval rules (read-only). The approval rules as an approver sees them: read-only, in plain language. `shell-overview-approval-rules-maya-desktop`

In the demo:

- an RFQ runs on its own (it is cheap and can be undone);
- a purchase order runs on its own only under $10,000, and only with a trusted partner;
- the demand planner makes work for orders on its own up to $2,000, cancels unstarted work nobody needs, and never buys without a person;
- kanban refills run on their own up to $1,000;
- anything no rule covers is only suggested: it never runs;
- a person's own proposal never runs alone, whatever the rules say.



Movie: What agents may do on their own (1 min 26 s). Ada reads the autonomy policy and checks three cases with "What would happen?"; Maya decides a refill the agent could not run alone. `network-agents-and-governance`

Chapters: The policy 0:00 · What would happen? 0:23 · A person decides 0:57

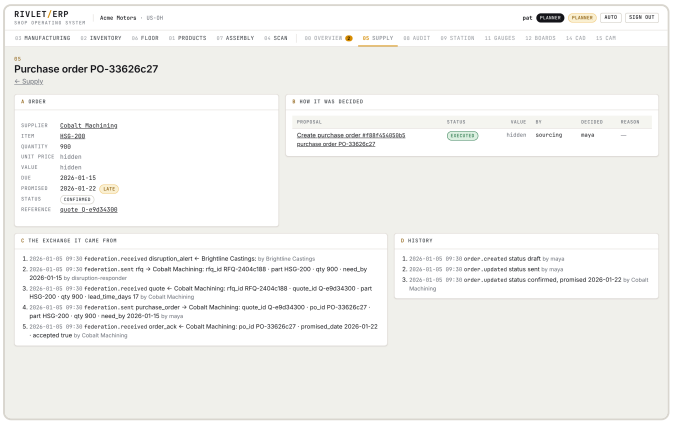
Releasing

A plant manager also releases engineering work:

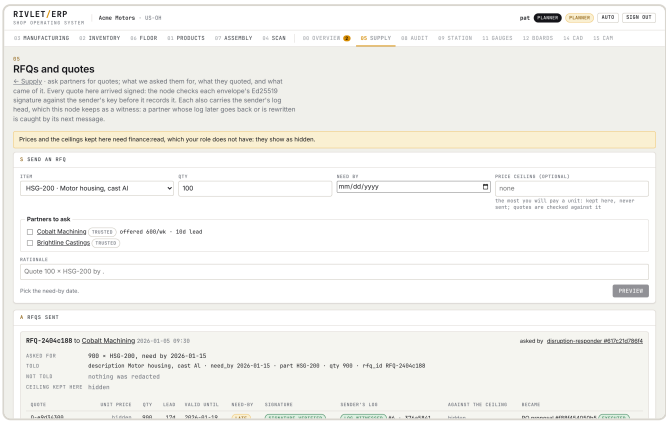
- designs in CAD, and programs in CAM: see Engineering;
- a new floor layout: see Engineering.

Finance: who sees money

Costs, prices, margins and values are shown only to roles that read finance: the buyer, the plant manager, the CFO and the admin. Everyone else sees "hidden" where a price would be, and a note says why. A Station tablet never shows prices.



A PO without finance. Pat's role has no finance:read, so the same purchase order shows its prices as hidden. `shell-record-po-no-finance-pat-desktop`



Quotes without finance. Without finance:read the quotes show no prices and no ceiling. `shell-quotes-no-finance-pat-desktop`

Someone who may approve a proposal always sees what it is worth: approving means seeing what is approved.

With finance rights you see on the Overview:

- the value of open orders;
- the value of stock on hand at standard cost (as **13 Finance** counts it), and which items have no cost;
- the value of each proposal waiting.

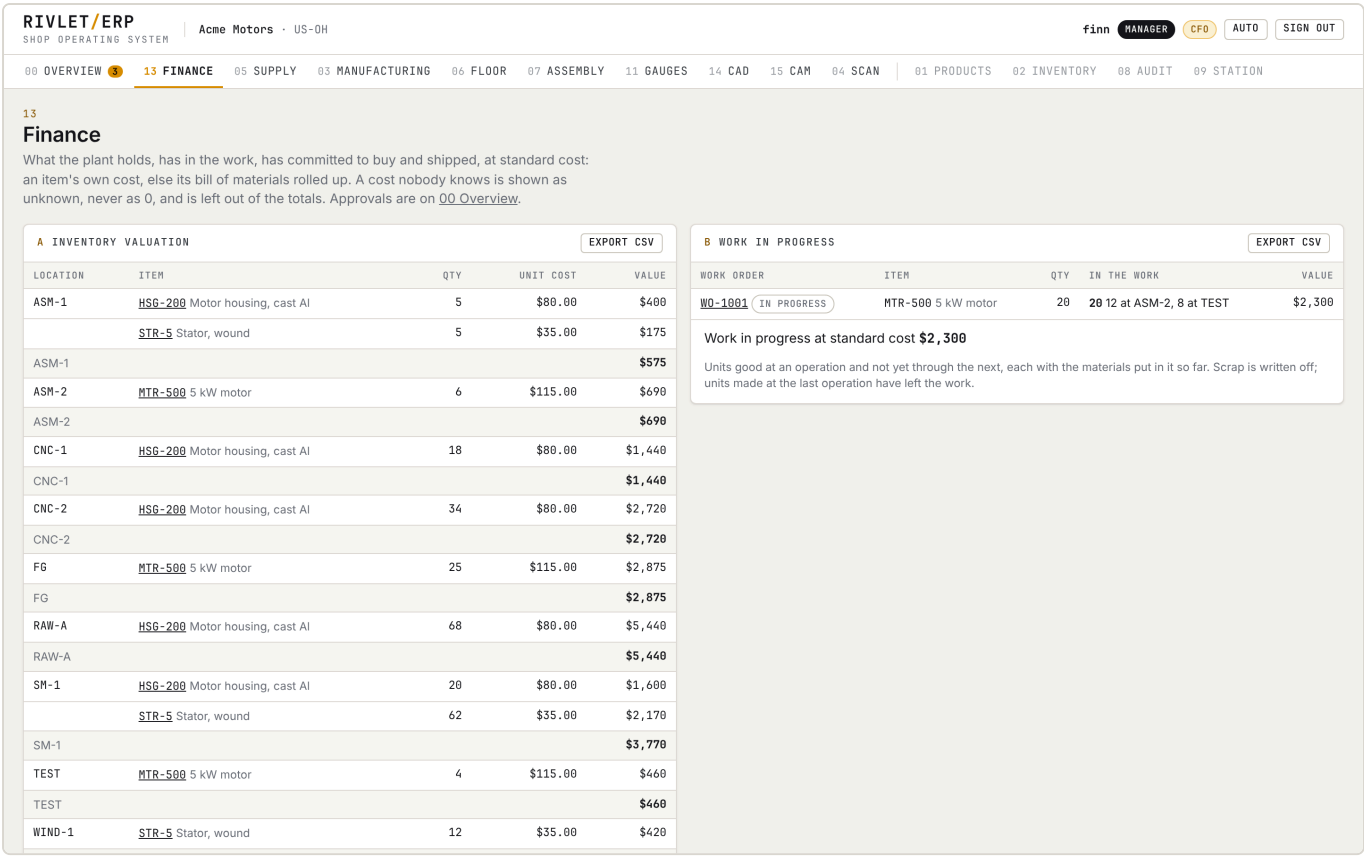
A work order is valued at its standard cost: the item's cost, or else its bill of materials rolled up. A proposal with no known value is over every limit but the CFO's.

Finance reports

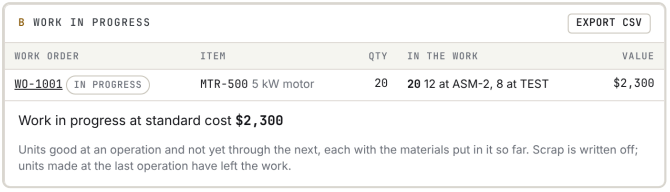
13 Finance is for roles that read finance: the CFO, the plant manager, the buyer and the admin. The CFO and the plant manager find it next to **00 Overview**; **Finance reports** on the Overview opens it too. Everyone else has no Finance section, and the node refuses them the reports.

Every report values things at **standard cost**: the item's own cost, or else its bill of materials rolled up (the demo's motor, MTR-500, has no cost of its own and rolls up to $115). A cost or price nobody knows reads **unknown**, never $0. Totals add up what is known, and say under them what they leave out ("Not counted: BRK-100 (no cost known)").

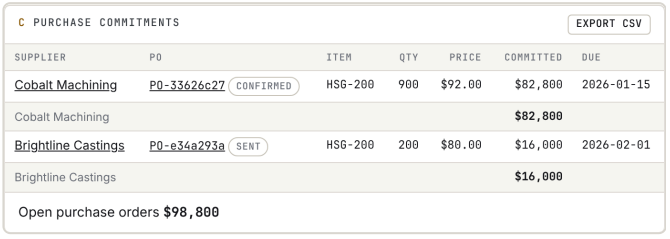
- **A Inventory valuation**: the stock at each location, item by item, with its unit cost and value, a subtotal for each location and the total.
- **B Work in progress**: each work order with units in the work (good at one operation, not yet through the next), where they are, and the standard cost of the materials put into them. Scrap is written off; finished units have left the work.
- **C Purchase commitments**: every open purchase order, by supplier, at its price: what the plant has committed to buy.
- **D Shipped this month**: each sales order shipped in the month (by the plant's days), how many, its revenue where it has a price, and its cost of goods at standard cost. Pick another month at the top of the panel.



13 Finance. The CFO's 13 Finance: inventory valued at standard cost by location and item, work in progress by work order, open purchase orders by supplier, and what shipped this month, each with Export CSV. `shell-finance-finn-desktop`



Work in progress. Work in progress by work order: the units good at an operation and not yet through the next, at the standard cost of the materials in them. `shell-finance-wip-finn-desktop`



Purchase commitments. Open purchase orders by supplier: what is committed and when it is due. `shell-finance-commitments-finn-desktop`

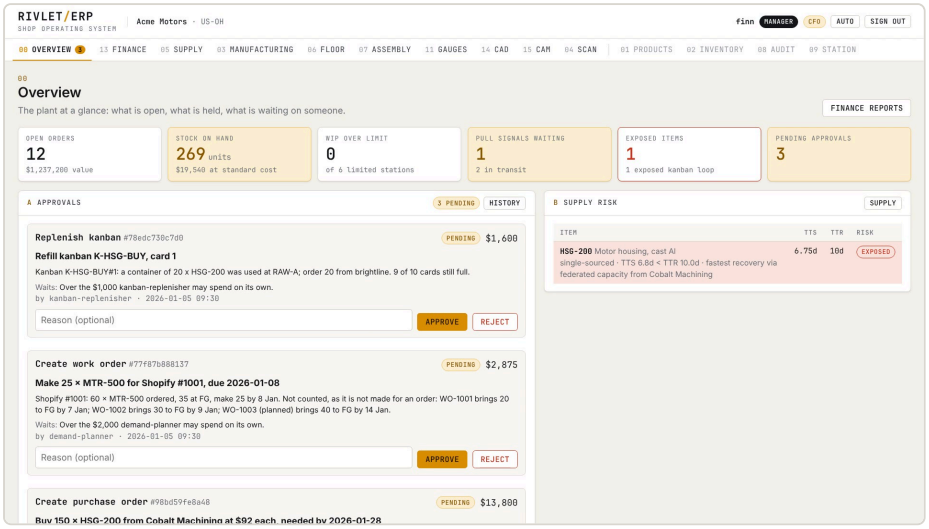
To export a report:

1. Open **13 Finance**.
2. Press **Export CSV** on the report's panel.
3. Open the file in a spreadsheet: one row per line, money as plain numbers, an unknown cost as "unknown".

rivlet-inventory-2026-01-05.csv · 11 rows

```
location,item,name,qty,unit_cost,stock_value
ASM-1,HSG-200,"Motor housing, cast Al",5,80,400
ASM-1,STR-5,"Stator, wound",5,35,175
ASM-2,MTR-500,5 kW motor,6,115,690
CNC-1,HSG-200,"Motor housing, cast Al",18,80,1440
CNC-2,HSG-200,"Motor housing, cast Al",34,80,2720
FG,MTR-500,5 kW motor,25,115,2875
RAW-A,HSG-200,"Motor housing, cast Al",68,80,5440
SM-1,HSG-200,"Motor housing, cast Al",20,80,1600
SM-1,STR-5,"Stator, wound",62,35,2170
TEST,MTR-500,5 kW motor,4,115,460
WIND-1,STR-5,"Stator, wound",12,35,420
```

The exported CSV. Export CSV downloads the report as a spreadsheet file (rivlet-inventory-2026-01-05.csv): one row per item at a location, its quantity, unit cost and stock value, money as plain numbers. `shell-finance-export-csv-finn-desktop`



Movie: Finance reports (37 s). Finn opens 13 Finance: inventory valued at standard cost by location and item, work in progress by work order, purchase commitments by supplier and what shipped this month; each exports as CSV. `shell-finance-reports`

Chapters: 13 Finance 0:00 · Inventory 0:04 · Work in progress 0:15 · Commitments 0:22 · Shipped this month 0:28

Supply risk

B Supply risk on the Overview, and **D Resilience** on 05 Supply, show what is exposed if a supplier cannot deliver: which items and kanban loops, and for how long the plant can run. After a supplier's warning (for example the demo's foundry fire at Brightline), agents may ask partner factories for quotes on their own. See Buying.

Administration

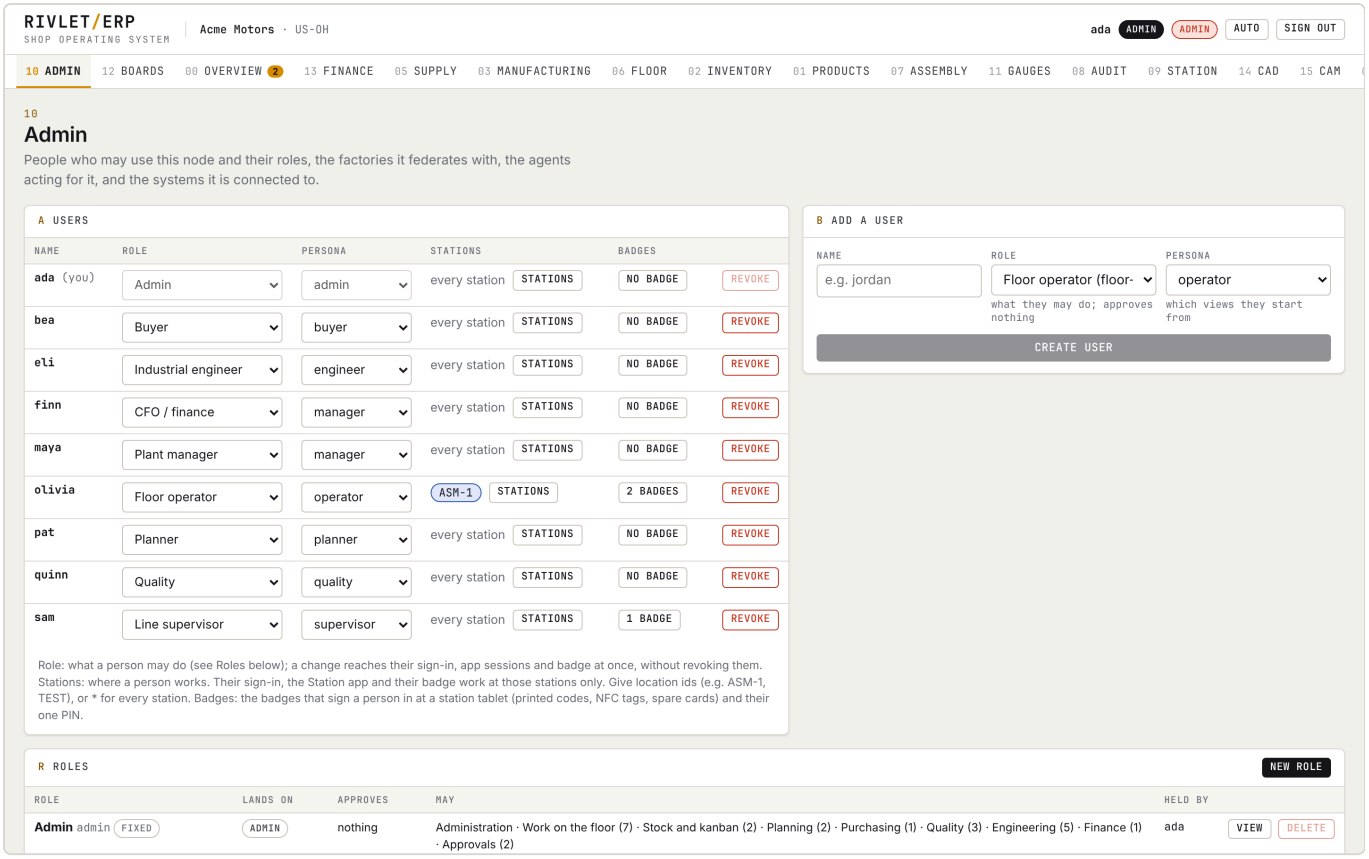
This chapter is for the administrator. You set up people, roles, badges, devices, apps, partners and the plant's rules. The admin role administers only: it reads everything, but does no work on the floor and approves no proposals. In the demo the admin is Ada.

The Admin page

10 Admin is one long page, in panels:

Panel	What it is for
A Users	Everyone who may sign in: role, persona, stations, badges
B Add a user	Add a person and get their token
R Roles	What each role may do, and up to what value it approves
C Partners	The partner factories this node trades with
D Agents and governance	The rules: what agents and apps may do on their own, and who approves the rest
E Apps	Installed apps, and **Install from manifest**
F Station devices	Enrolled tablets, TV boards and edge bridges
G Enroll a tablet	A one-time code to enroll a tablet or a bridge
H Station badges	Give a person a printed badge and set their PIN
I Boards	A link to 12 Boards, where TVs are approved and targets set
J Integrations	The MQTT mirror, the A2A agent card and MCP: read-only, with their settings

Only admins see 10 Admin. Anyone else who opens it reads "Admin needs an admin role."

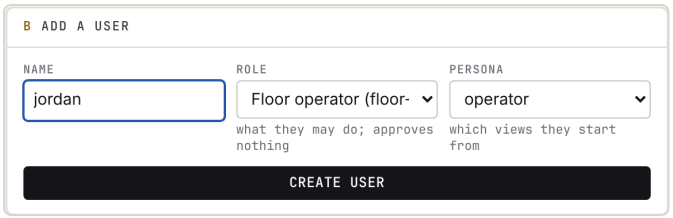


Admin as Ada. Users, roles, partners, governance, installed apps, tablets, badges and boards. Seen by Ada (admin). `shell-admin-page-ada-desktop`

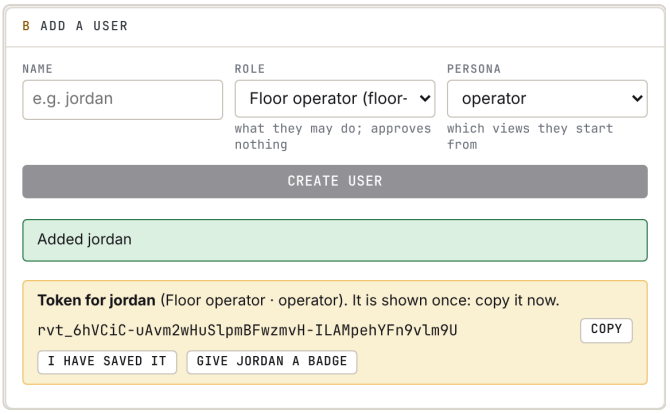
People

To add a person

1. In **B Add a user**, enter the **Name**.
2. Pick the **Role**: what they may do. The hint says what it approves.
3. Pick the **Persona**: which views they start from.
4. Press **Create user**.
5. The token is shown once. Copy it now and give it to the person privately. Then press **I have saved it**.



Add a user. Adding a user: a name, a role (what they may do) and a persona (where they start). `shell-admin-add-user-filled-ada-desktop`



The new token. The new person's token is shown once: copy it and hand it over. "Give jordan a badge" opens their badges. `shell-admin-new-token-ada-desktop`

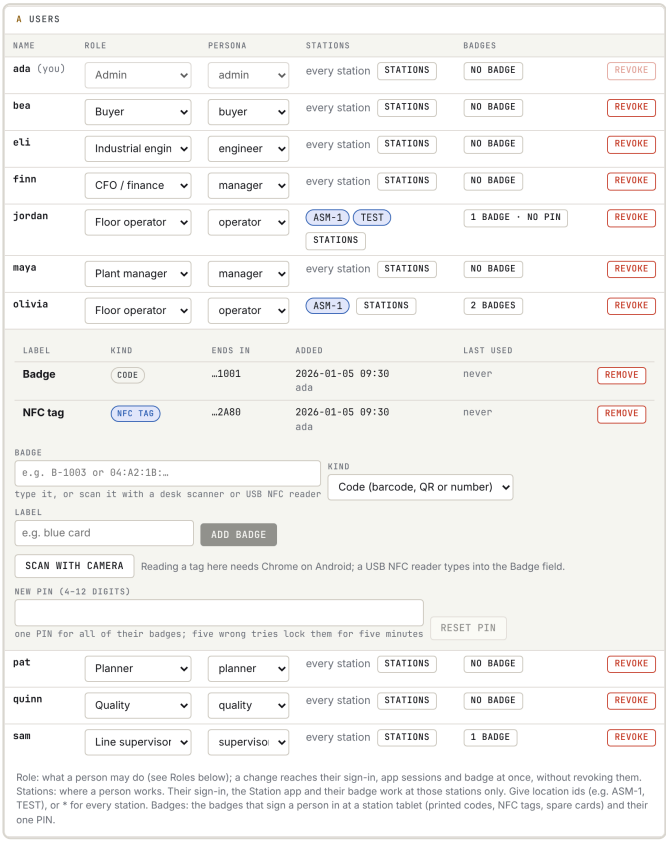
To set where a person works

A person's sign-in, the Station app and their badge work only at their stations.

1. In **A Users**, press **Stations** on their row.
2. Type the station ids, separated by commas (for example `ASM-1, TEST`), or `*` for every station.
3. Press **Save**.



Set stations. Stations: type where the person works (ASM-1, TEST), or * for every station. `shell-admin-stations-edit-ada-desktop`



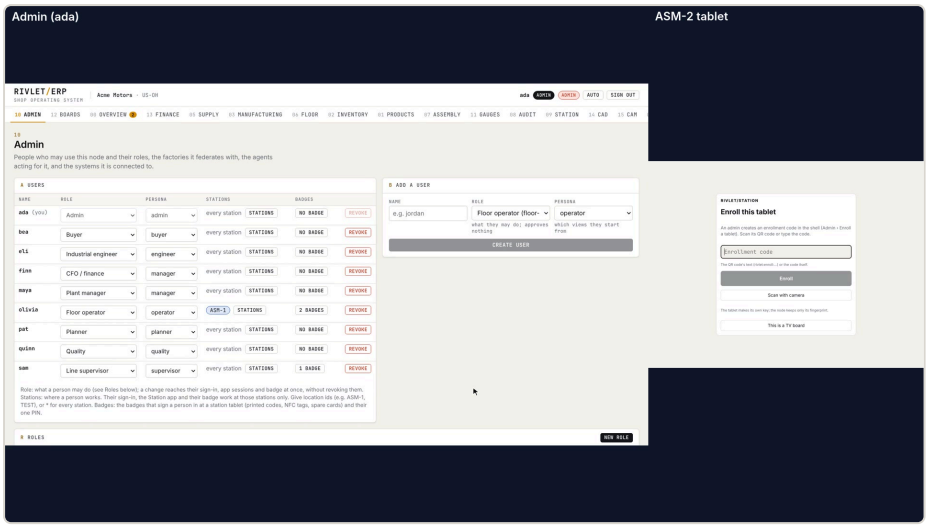
Users after adding Jordan. Jordan now works at ASM-1 and TEST only, with a badge. `shell-admin-users-after-ada-desktop`

To change a person's role or persona

Pick another role or persona on their row. The change reaches their sign-in, app sessions and badge at once. You cannot change your own role here.

To remove a person

Press **Revoke** on their row and confirm. Their token stops working at once. Their open shell says they are signed out, at their next step.



Movie: Add a person and enroll a tablet (1 min 43 s). Ada adds Jordan, sets the stations Jordan works at, gives Jordan a badge and PIN, creates an enrollment code, enrolls the ASM-2 tablet with it, Jordan badges in, then Ada revokes the tablet. `shell-admin-people-and-tablets`

Chapters: Add a person 0:00 · Set stations 0:20 · A badge and a PIN 0:30 · Enroll a tablet 0:43 · See and revoke the device 1:26

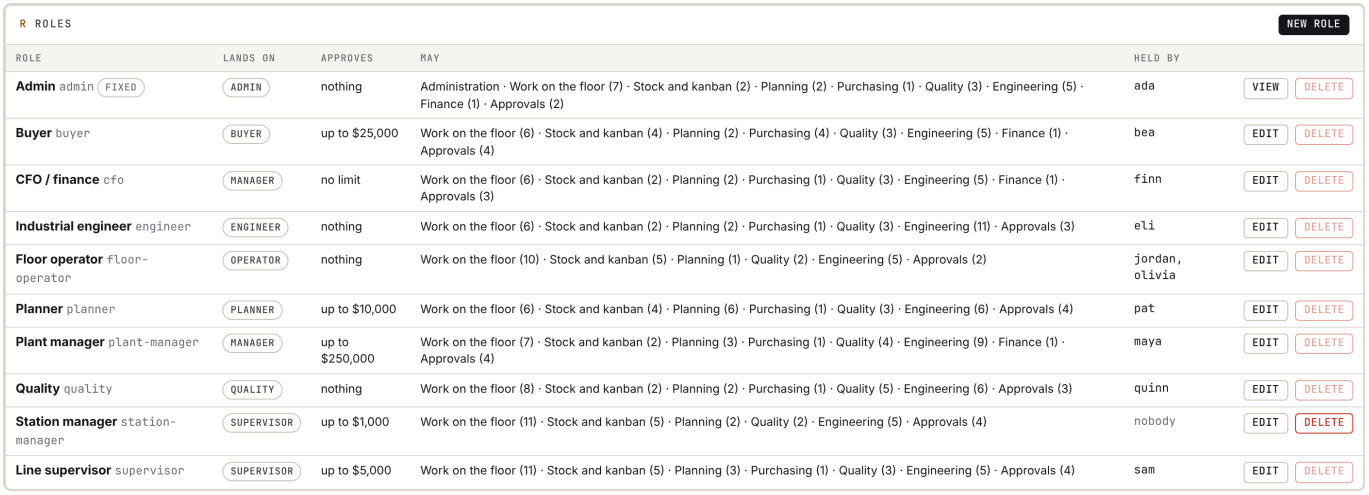
Roles and approval limits

R Roles lists the plant's roles: where each lands, what it approves, what it may do (in groups such as "Work on the floor" and "Approvals"), and who holds it. Press **Edit** to change a role's rights or limit, **New role** to add one, and **Delete** to remove one nobody holds. The default roles are listed in Getting started. Admin is fixed: **View** shows what it holds.

The **Approves** column reads:

- **nothing** for a role without the right to approve proposals ("approve proposals worth up to the role's approval limit", under Approvals), whatever its limit says. Admin is one: it administers and reads everything, but approves nothing.
- **up to $1,000** (its limit) for a role that approves.
- **no limit** for a role that approves with no ceiling, such as the CFO.

The same words are the hint under **Role** in **B Add a user**.



Roles. Roles: the plant's roles with what each may do and up to what value each approves ("nothing" for a role that does not approve, Admin included; "no limit" for the CFO); the defaults can be edited. `shell-admin-roles-ada-desktop`

Some rules hold whatever the role says:

- Nobody approves their own proposal.
- A proposal that does not say what it is worth is over every limit except the CFO's.
- At a Station tablet, a person never sees prices, and approves only if their role also works the floor (station manager, line supervisor), and only what concerns that station (its **Approvals** tab, see Running a station).
- Only the quality role signs off held units.

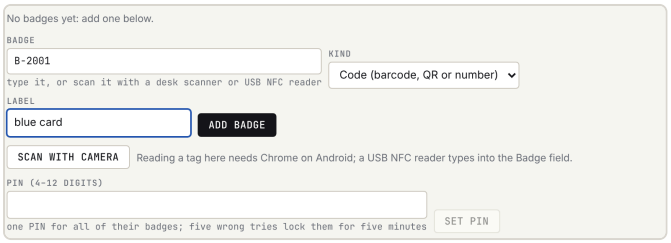
Badges

A badge signs a person in at the tablets of their stations, with their PIN. A person may have several badges (a printed card, an NFC tag, a spare card) and one PIN for all of them.

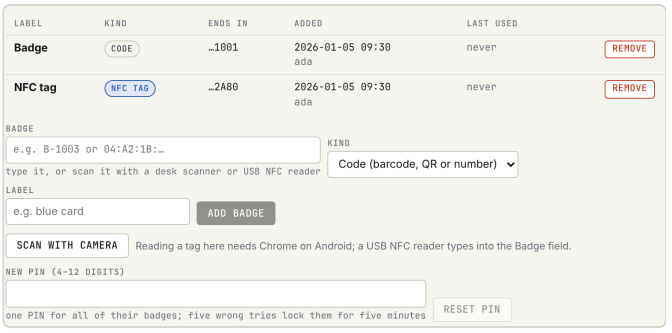
To give a person a badge

1. In **A Users**, press the badges button on their row ("No badge", or "2 badges"). Right after adding a person, **Give jordan a badge** does the same.
2. Pick the **Kind**: **Code (barcode, QR or number)** or **NFC tag (its UID)**. Enter the **Badge** value: type it, scan it into the field with a desk scanner or USB NFC reader, use the camera, or read an NFC tag on an Android phone or tablet.
3. Add a **Label** if it helps (for example "spare card"), then press **Add badge**.
4. Enter their PIN (4 to 12 digits) and press **Set PIN** (or **Reset PIN**). Without a PIN, their badges do not sign them in.

Only the last four characters of each badge are shown. Press **Remove** to stop a badge signing them in.

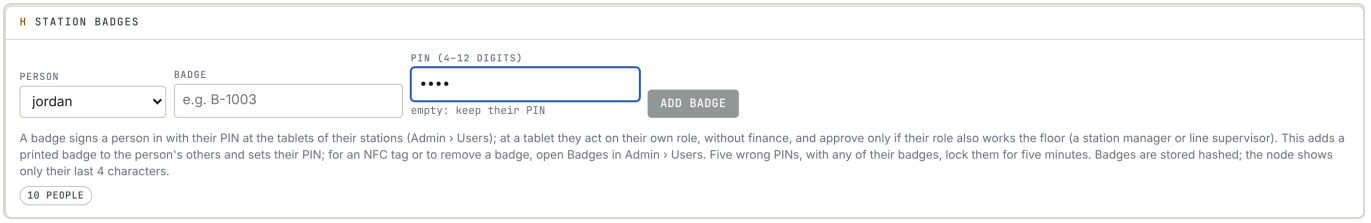


A person's badges. Users › Badges: each person may have several badges (a printed card, an NFC tag); add one by typing it, scanning it into the field or reading a tag. `shell-admin-user-badges-ada-desktop`



Olivia's badges. Olivia has a printed badge and an NFC tag; only the last four characters of each are shown. `shell-admin-user-badges-olivia-ada-desktop`

H Station badges is a quicker way to add a printed badge and set the PIN together: pick the **Person** (nobody is picked for you), the **Badge** and the **PIN (4–12 digits)**, then **Add badge**.



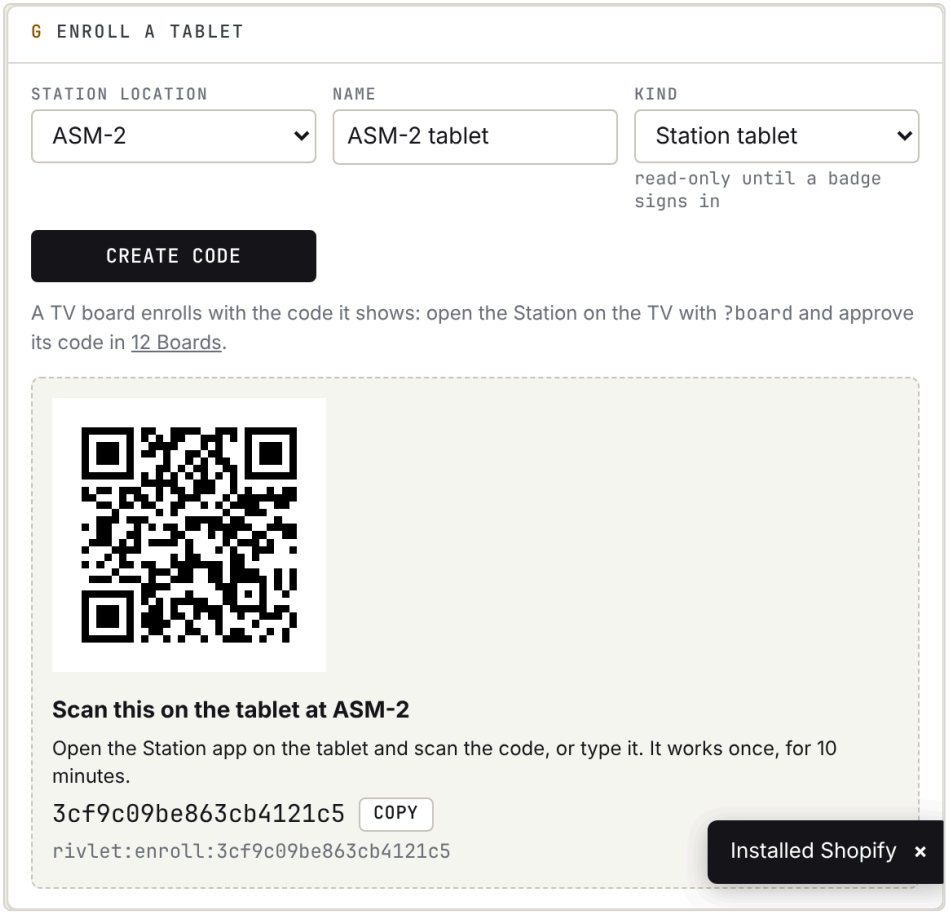
Set a badge and PIN. Station badges: pick the person (nobody is picked for you), a badge and a PIN of 4 to 12 digits. `shell-admin-badge-ada-desktop`

Five wrong PINs, with any of a person's badges, lock them for five minutes. Badges are stored scrambled (hashed); nobody can read them back.

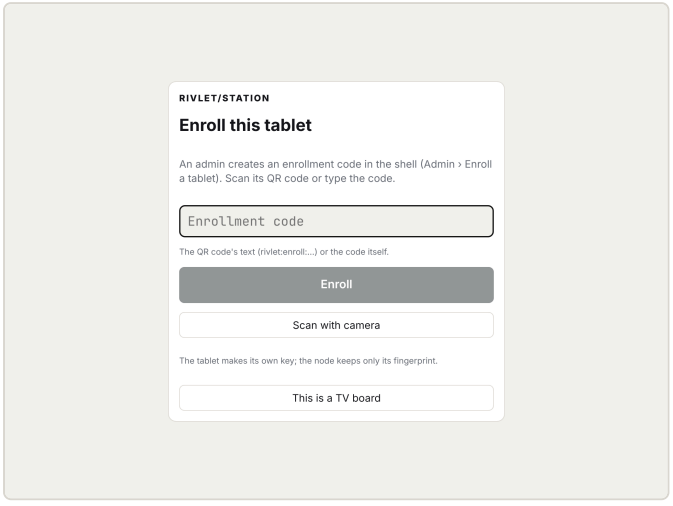
Devices

To enroll a Station tablet

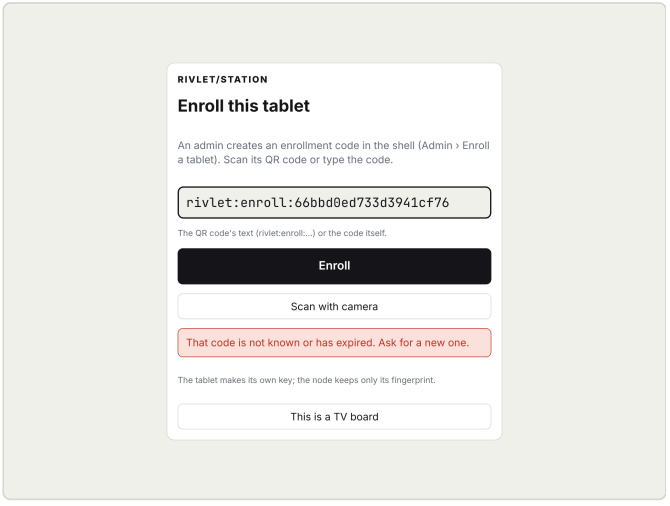
1. In **G Enroll a tablet**, pick the **Station location**, give it a **Name**, and choose the **Kind Station tablet**.
2. Press **Create code**. A QR code and a one-time code appear. The code works once, for 10 minutes.
3. On the tablet, open the Station's address. It shows **Enroll this tablet**.
4. Scan the QR code, or type the code, and tap **Enroll**.
5. The tablet says it is the station. Tap **Start**.



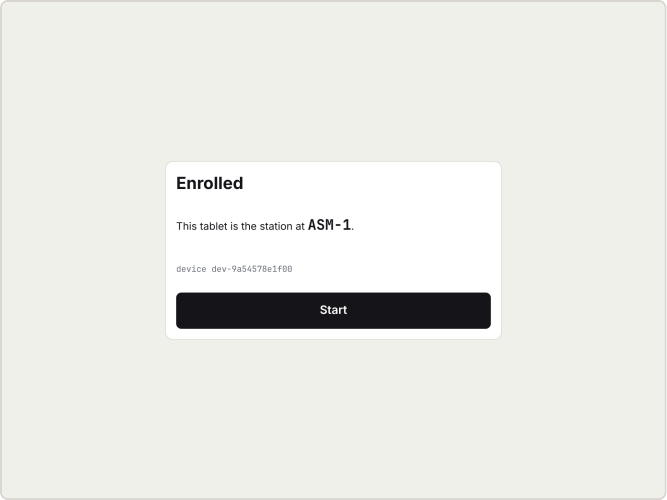
Enrollment code and QR. A one-time enrollment code and QR for the tablet at ASM-2: scan it on the tablet, or type the code. `shell-admin-enroll-code-ada-desktop`



Enroll this tablet. A new tablet asks for the enrollment code an admin created in Admin › Enroll a tablet. `floor-enroll-code-empty-admin-tablet`

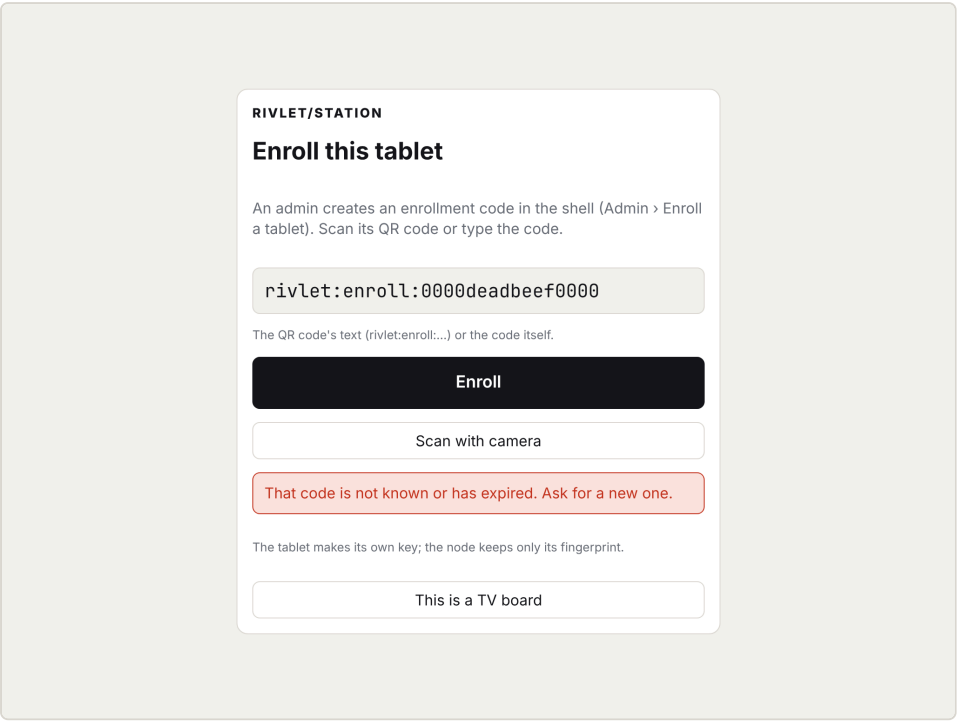


Enrollment code entered. Type the code or scan its QR code, then tap Enroll. `floor-enroll-code-filled-admin-tablet`



Enrolled. The tablet is now the station at ASM-1; tap Start to open it. `floor-enroll-enrolled-admin-tablet`

A mistyped, used or expired code is refused: "That code is not known or has expired. Ask for a new one."



Enrollment code refused. A mistyped, used or expired code is refused, and the tablet stays unenrolled. `floor-enroll-bad-code-admin-tablet`

To see and revoke devices

F Station devices lists every enrolled tablet, TV board and edge bridge, with its station and who enrolled it. Press **Revoke** to remove one: a lost or replaced tablet, for example.

- A revoked tablet says "This tablet is no longer enrolled" and asks to be enrolled again. Changes it could not send are kept on it, under their people, and go out once it is enrolled again.
- A revoked TV says "This TV is no longer a board".

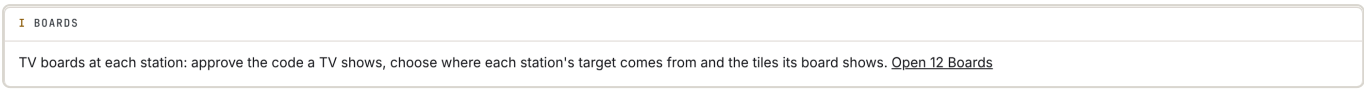
- To move a tablet to another station, revoke it and enroll it again there.



Station devices. Enrolled tablets, TV boards and edge bridges, with their station and who enrolled them. `shell-admin-devices-ada-desktop`

TV boards

TVs do not use enrollment codes. A TV shows a code of its own, and a supervisor or station manager approves it in **12 Boards**. See Running a station. **I Boards** links there.



Boards in Admin. Admin links to 12 Boards, where TVs are approved and targets set. `shell-admin-boards-ada-desktop`

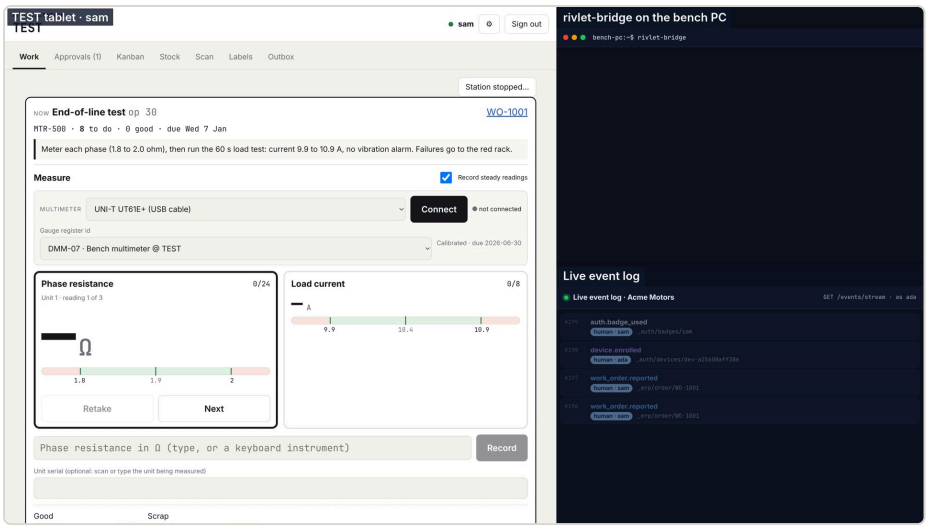
Edge bridges

An edge bridge (`rivlet-bridge`) is a small program on a computer next to an instrument or a machine. It sends readings, or delivers programs to a machine, with an identity of its own.

To set up a bridge at a test bench:

1. In **G Enroll a tablet**, pick the station (TEST), name it, choose the **Kind Edge bridge**, and press **Create code**.
2. On the bench computer, run `rivlet-bridge enroll --node <the node's address> --code <the code> --name "TEST bench"`. The bridge makes and keeps its own key.
3. In **11 Gauges**, open the instrument (for example the multimeter DMM-07) and bind it to the bridge, so readings name the right instrument. A plant manager or quality may do this.
4. Start the bridge with the instrument's driver. Each reading is sent on its own, bound to the job running at the station, and the node judges it pass or fail.

The readings appear at once on the station's tablet. In the log, each reading's actor is the bridge, not a person.



Movie: An edge bridge at the test bench (1 min 31 s). A bridge next to the TEST bench enrolls with a one-time code, its multimeter is bound to it in the gauge register, and its readings land on WO-1001 op 30: live on the TEST tablet and in the event log, judged pass or fail by the node. `network-edge-bridge`

Chapters: The job at TEST 0:00 · Enroll the bridge 0:06 · Bind the meter 0:12 · Readings, live 0:19

Apps

E Apps lists the installed apps, the rights each was granted, the links it handles, and any worker scripts it may run. CAD, CAM and the Station are apps; so is a sales channel such as Shopify.

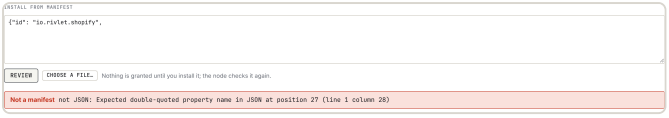


Installed apps. Installed apps, the scopes each was granted, the intents it handles and the workers that may evaluate JavaScript. `shell-admin-apps-ada-desktop`

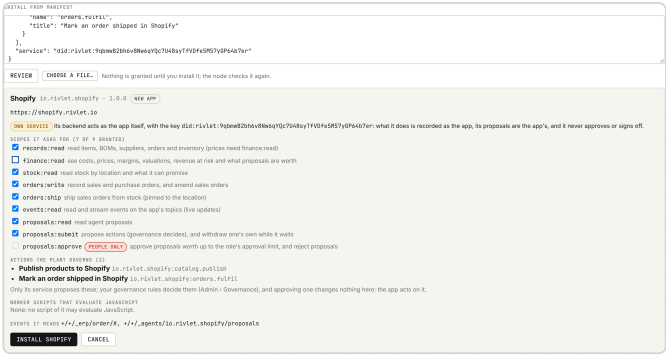
To install an app from its manifest

1. Paste the app's manifest (a JSON file, `rivlet-app.json`) in **Install from manifest**, or press **Choose a file…**.

2. Press **Review**. A manifest that is not JSON is caught here, before anything is sent ("Not a manifest").
3. Read what it asks for:
    - **Scopes it asks for**: untick any you do not grant. Approving and signing off are for people only and cannot be granted to an app.
    - **Actions the plant governs**: things only the app proposes, which your rules decide.
    - **Own service**: the key its backend uses to act as the app itself, with no person signed in.
    - **Worker scripts that evaluate JavaScript**, if any.
4. Press **Install** and the app's name (for example **Install Shopify**). Nothing is granted until you install it, and the node checks it again.



A broken manifest. A manifest that is not JSON is caught before anything is sent. `shell-admin-install-broken-ada-desktop`



Review before installing. Shopify's manifest reviewed: the scopes it asks for (finance:read unticked here; proposals:approve is for people only), its own actions for governance, and its service key. `shell-admin-install-review-ada-desktop`

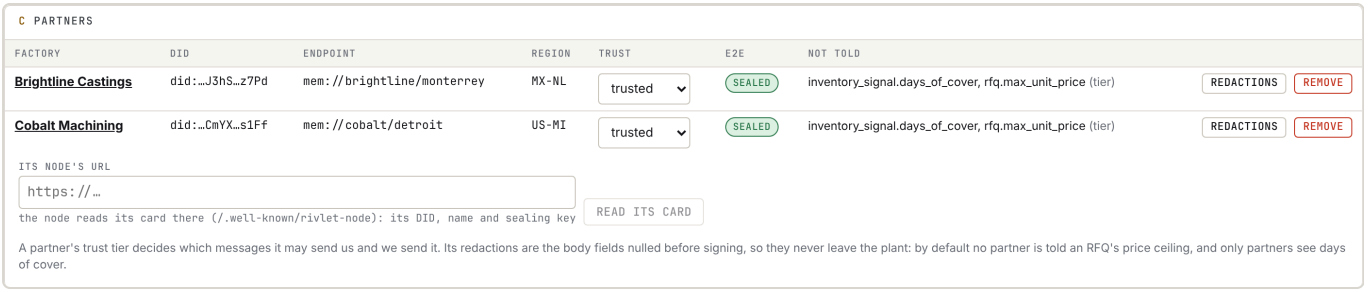


Installed, with its own service. Installed: Shopify is listed with its own service, its governed actions and what it asked for beyond its grant. `shell-admin-install-done-ada-desktop`

The same manifest again reviews as an upgrade, with what is new marked. Press **Remove** on an app to uninstall it: an app's own service is refused at once.

Partners

C Partners lists every partner factory: its DID, endpoint, region, trust tier, whether messages to it are sealed end to end, and what it is not told.



Partners. Partners: each factory this node federates with, its endpoint, trust, whether messages are sealed and what it is not told. `shell-admin-peers-ada-desktop`

To add a partner

1. Get the address of the partner's node from the partner, for example `https://rivlet.delta.example`.
2. In **Add a partner**, enter it under **Its node's URL** and press **Read its card**. The node reads the partner's public node card there and shows what it says: its **DID** (its identity, `did:rivlet:…`), name, region, **Endpoint** and **Sealing key**.
3. Check the DID with the partner by phone or email: anyone can serve another node's card.
4. Pick the **Trust** tier and press **Add** with the partner's name. Messages to it are sealed end to end with its key.

If the card cannot be read, the reason shows under the form: the address is not a URL, the node does not answer there, what it serves is not a valid node card, or it is this node's own address. Reading the card of a partner already listed offers **Update** instead: it takes the name, endpoint and key from the card and keeps the trust tier and redactions.



Add a partner. Add a partner from its node card: the node reads it from the URL typed, and shows its DID, name, endpoint and sealing key to confirm before it is added at a trust tier. `shell-admin-partner-add-ada-desktop`

To change a partner's trust

Pick another tier on its row and confirm. Known partners may quote; trusted ones may also take purchase orders; partners also get inventory signals and forecasts. The full list is in Buying.

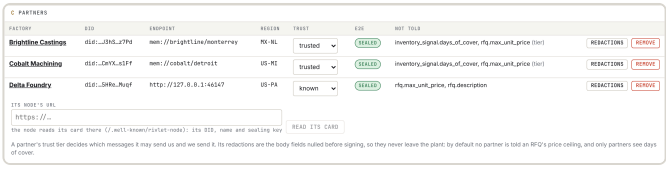
To set what a partner is not told

1. Press **Redactions** on its row.
2. For each message type, list the fields to blank out before a message to it is signed. The tier still decides which message types it may get.
3. Press **Save redactions**. **Use the tier's default** goes back to the default.

By default every partner is not told an RFQ's price ceiling (`rfq.max_unit_price`).



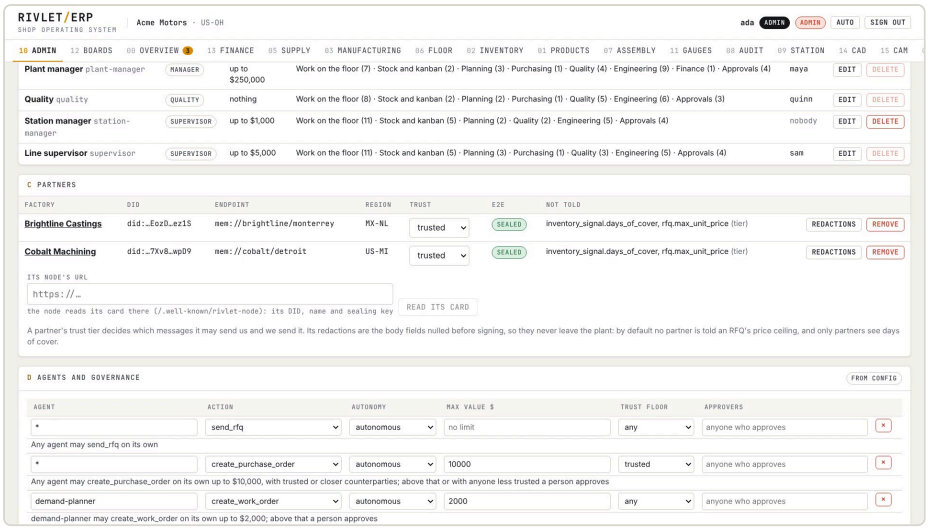
What a partner is not told. Redactions per message type: fields nulled before an envelope to Delta is signed. Its tier still decides which message types it may get. `shell-admin-partner-redactions-ada-desktop`



Partners after adding Delta. Delta Foundry, re-tiered to trusted, is not told rfq.max_unit_price, rfq.description or purchase_order.need_by. `shell-admin-partners-ada-desktop`

To remove a partner

Press **Remove** on its row and confirm. Its messages are refused from now on, and nothing more is sent to it.



Movie: Partners, apps and who approves (1 min 34 s). Ada adds Delta Foundry from its node card, makes it trusted and sets what it is not told; installs Shopify from its manifest after reviewing what it asks for; and adds a rule that only a plant manager approves Shopify's "Publish products". Shopify proposes it with its own token, and Maya decides. `shell-partners-apps-and-rules`

Chapters: Add a partner 0:00 · What it is not told 0:18 · Install an app from its manifest 0:29 · Who approves 0:50 · The app proposes, a plant manager decides 1:11

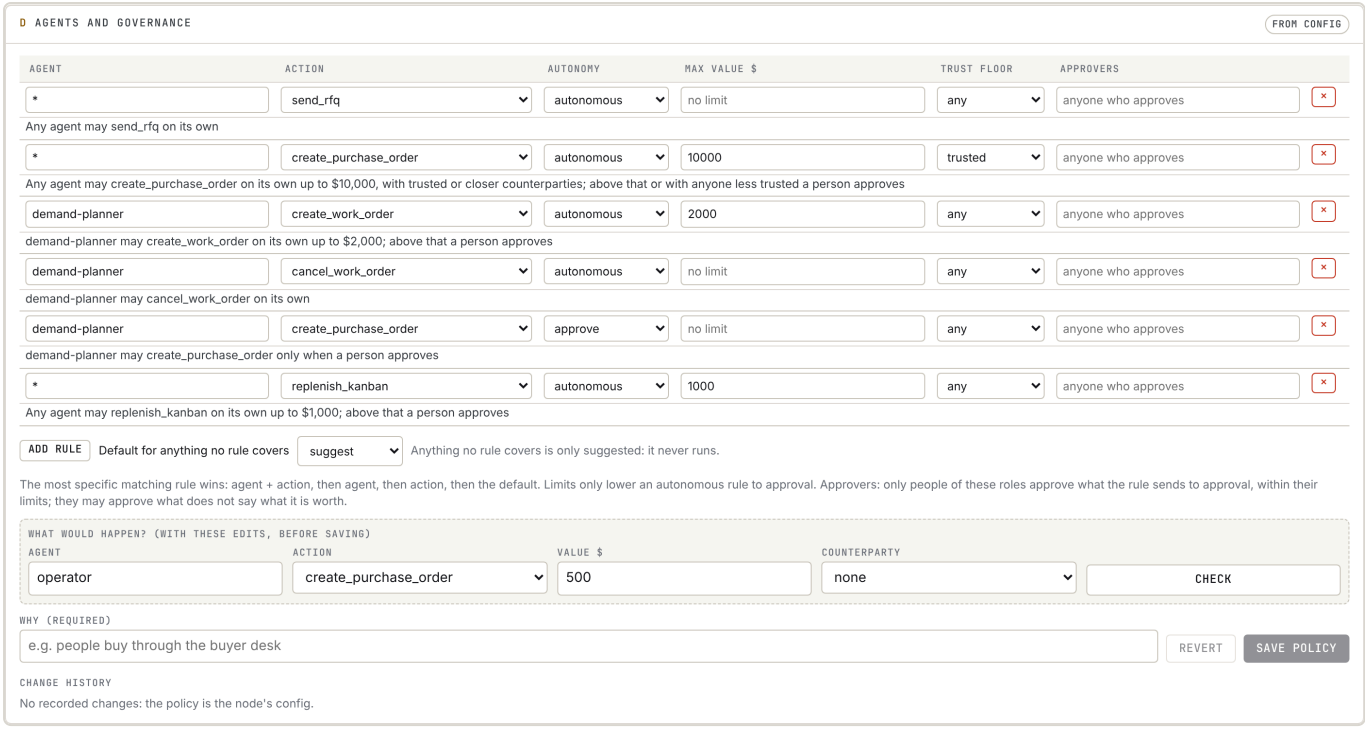
Governance rules and approvers

D Agents and governance holds the plant's rules: what agents and apps may do on their own, and who approves the rest. The rules are written per agent (or `*` for any) and action, with:

- **Autonomy**: observe, suggest, approve (waits for a person) or autonomous (runs on its own);
- **Max value $**: above it, an autonomous rule waits for a person;
- **Trust floor**: the least trusted partner it may deal with on its own;

- **Approvers**: the roles that approve what it holds. Empty means anyone who approves, within their limit.

Each rule is also written out in plain words under it. **Default for anything no rule covers** decides the rest. The most specific rule wins: agent and action, then agent, then action, then the default.



Governance rules. Governance rules decide which actions run at once, which wait for an approver, up to what value, and which roles approve. `shell-admin-governance-ada-desktop`

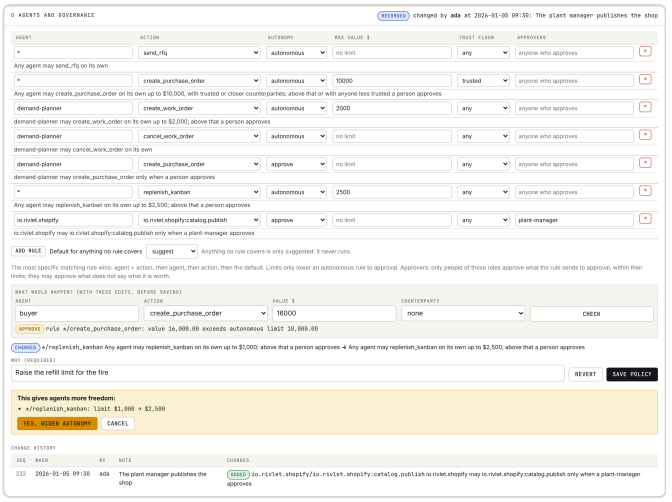
To change the rules

1. Edit a rule, or press **Add rule**.
2. Check it: in **What would happen?**, enter an **Agent**, **Action**, **Value $** and **Counterparty**, then press **Check**. It runs that case through your edited rules before you save.
3. Write **Why (required)**: the change note.
4. Press **Save policy**. If you give agents more freedom, you are asked to confirm first, with the difference shown: **Yes, widen autonomy**, or **Cancel**. **Revert** drops your edits.

Every change is kept in the **Change history**: who, when, the note, and what changed (for example "changed $1,000 → $900").



Check a rule. Check what governance would do with an action before anyone tries it. `shell-admin-governance-check-ada-desktop`



Confirm wider autonomy. Widening what runs without approval asks for a second confirmation and shows the difference. `shell-admin-governance-confirm-ada-desktop`

233	2026-01-05 09:30	ada	Lower the kanban limit during the fire	CHANGED */replenish_kanban Any agent may replenish_kanban on its own up to $1,000; above that a person approves → Any agent may replenish_kanban on its own up to $900; above that a person approves
232	2026-01-05 09:30	ada	The plant manager publishes the shop	ADDED io.rivlet.shopify/io.rivlet.shopify:catalog.publish io.rivlet.shopify may io.rivlet.shopify:catalog.publish only when a plant-manager approves

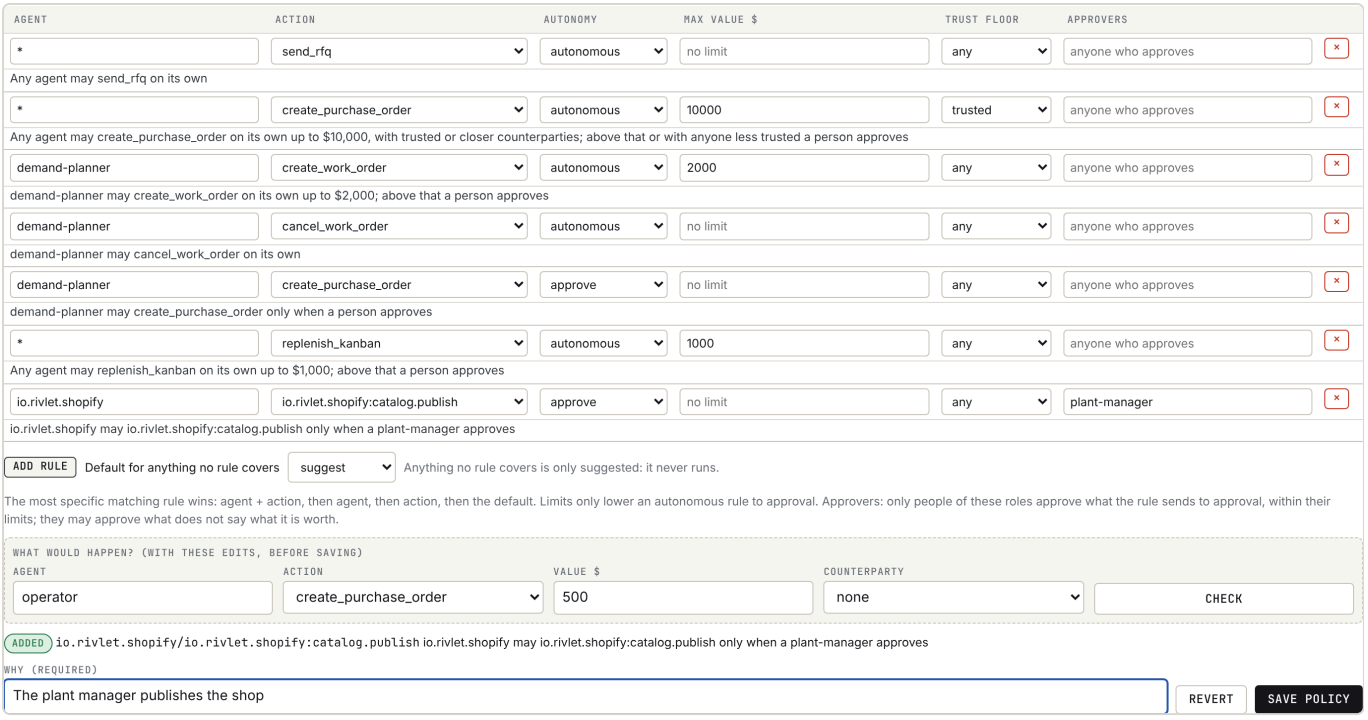
Governance history. Every change to the rules is kept, with who made it, the note and what changed. `shell-admin-governance-history-ada-desktop`

To name who approves an app's action

An app's own actions are governed like an agent's. For example, to have only a plant manager approve Shopify publishing its products:

1. Add a rule with agent `io.rivlet.shopify` and action `io.rivlet.shopify:catalog.publish`.
2. Set **Autonomy** to approve.
3. Under **Approvers**, enter `plant-manager`.
4. Save with a note.

Others then see "Only plant-manager approves this." on the card, and may reject it but not approve it.

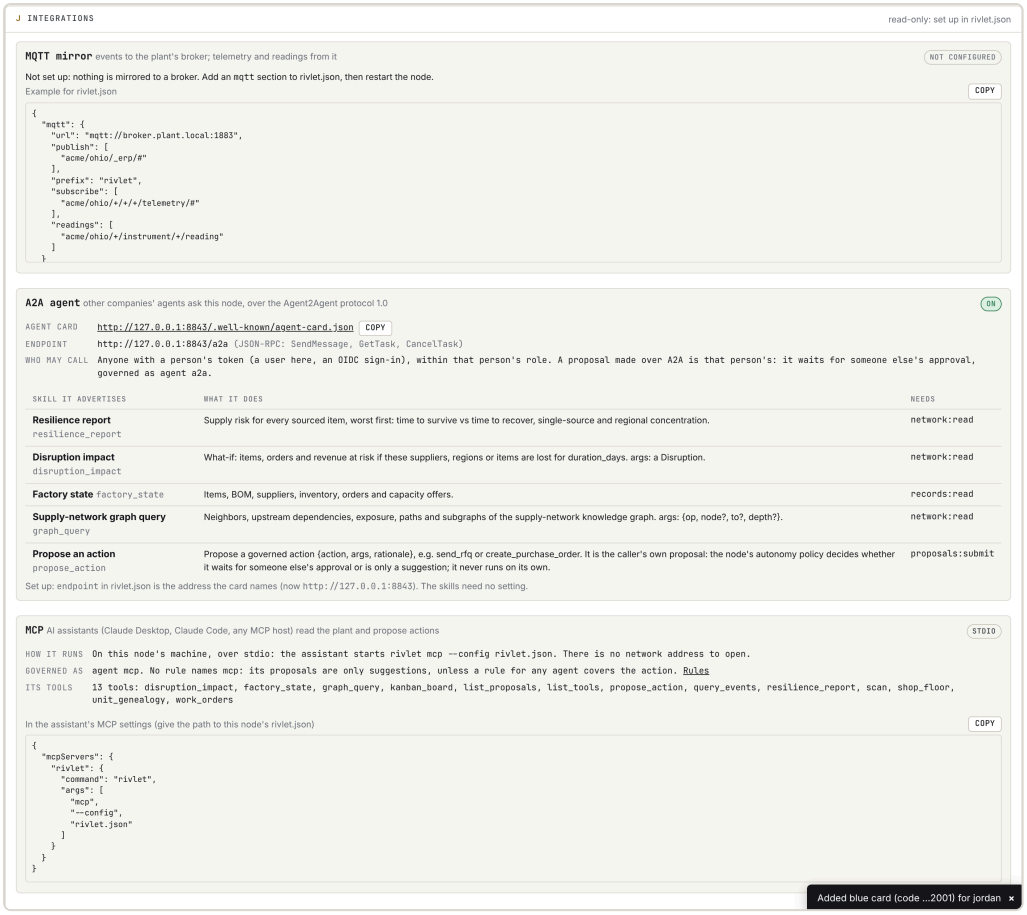


A rule that names its approvers. A rule for Shopify's own "Publish products" action: it waits for approval, and only a plant manager approves it. `shell-admin-governance-approvers-ada-desktop`

Integrations

J Integrations shows, read-only, what the node offers other systems, and the settings that set each up. Settings are changed in the node's configuration file, `rivlet.json`, by whoever runs the node.

- **MQTT mirror:** whether the node mirrors its events to the plant's MQTT broker, and whether it is connected (since when, how many events it mirrored and messages it took in, the last error). It lists what is mirrored and to which broker topic, and the telemetry and instrument readings it takes in from the broker. Below are the `mqtt` settings in `rivlet.json` (the broker's password is never shown), or an example when there is none. Access control (users, badges, devices) is never mirrored.
- **A2A agent:** the address of the node's agent card, which other companies' agents read, the skills it advertises and what each needs. A proposal made over A2A is the caller's own, and waits for someone else's approval. The card names the address in the `endpoint` setting.
- **MCP:** how an AI assistant (Claude Desktop, Claude Code or another MCP host) reads the plant and proposes actions: it runs `rivlet mcp --config rivlet.json` on the node's machine. It shows the tools the assistant gets, how its proposals are governed (as agent `mcp`), and the settings to give the assistant.



Integrations. Admin › Integrations, read-only: the MQTT mirror to the plant's broker and its state, the A2A agent card and the skills it advertises, and MCP for AI assistants with its tools and how their proposals are governed. `shell-admin-integrations-ada-desktop`

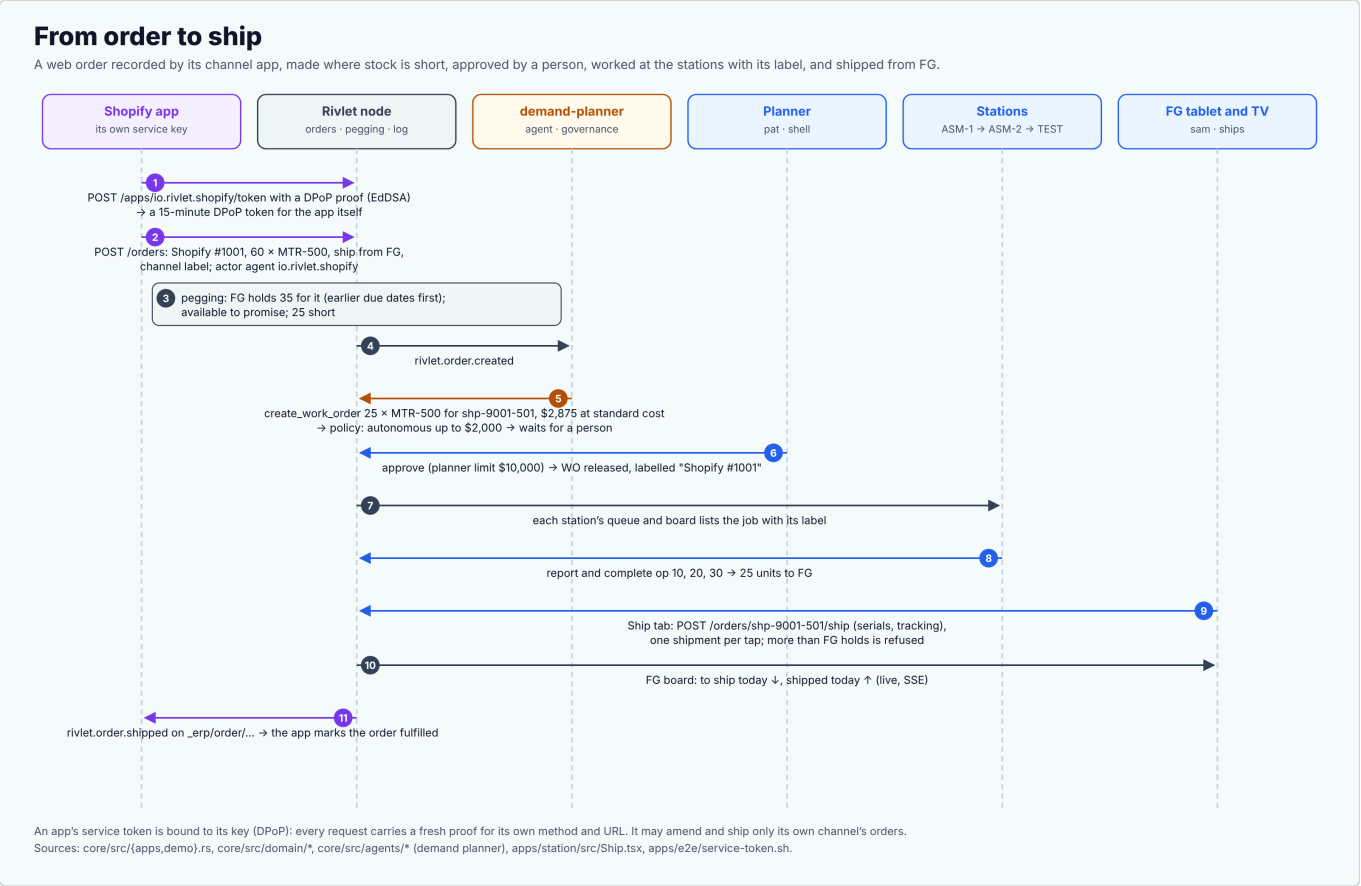
What has no screen yet

The plant's time zone and shifts (`calendar`) live only in `rivlet.json`, with no screen in Admin. Ask whoever runs the node to change them.

Selling through Shopify

This chapter is for everyone at the plant who touches a web order: the planner who approves the work, the operators who make it, the packer at FG who ships it, and the administrator who connects the shop. It describes the plant's side. The shop's side (the storefront, the catalogue, the shop owner's console) is run by the Rivlet Shopify app.

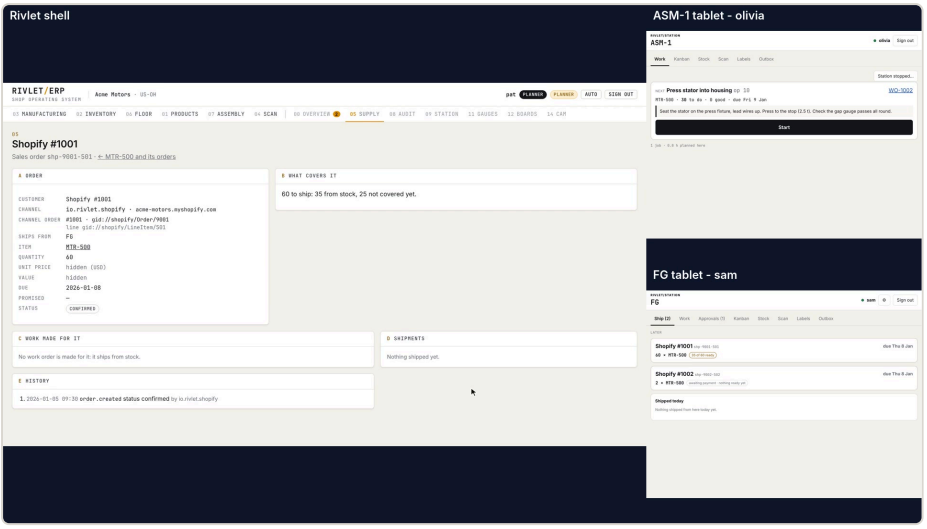
How an order travels



From order to ship. A web order recorded by its channel app with its own token, made where stock is short, approved by a person, worked at the stations with its label, and shipped from FG. `network-order-to-ship-sequence-all-1600`

1. **A customer orders on the web shop.** In the demo, Shopify #1001: 60 MTR-500 motors.
2. **The Shopify app records it in Rivlet, as itself.** No person signs in to do this. Each line of the web order becomes a sales order, labelled "Shopify #1001", shipping from FG within the plant's usual days (three in the demo). The log names the app as the one who recorded it.
3. **Rivlet works out what covers it.** FG holds 35 motors for it; 25 are short. Orders due earlier are covered first.
4. **The demand planner proposes making the rest.** A work order for 25 motors, made for Shopify #1001, worth $2,875 at standard cost. That is over what it may do on its own ($2,000), so it waits for a person.

5. **The planner approves it.** Pat approves it within his $10,000. The work order is released, labelled with the order it is for.
6. **The work reaches the stations at once.** Each station's queue and TV board show the job with its label: "WO-1012 · Shopify #1001 · 25 × MTR-500 · due Thu". Operators work it as any other job: pressed at ASM-1, wired at ASM-2, tested at TEST. The good units go to FG.
7. **The packer ships it at FG.** On the FG tablet's **Ship** tab, Shopify #1001 shows as ready. The packer scans the serials and the tracking number from the label, and ships. The FG board counts it.
8. **The shop is told.** The Shopify app sees the shipment and marks the order fulfilled in Shopify, with the tracking number.



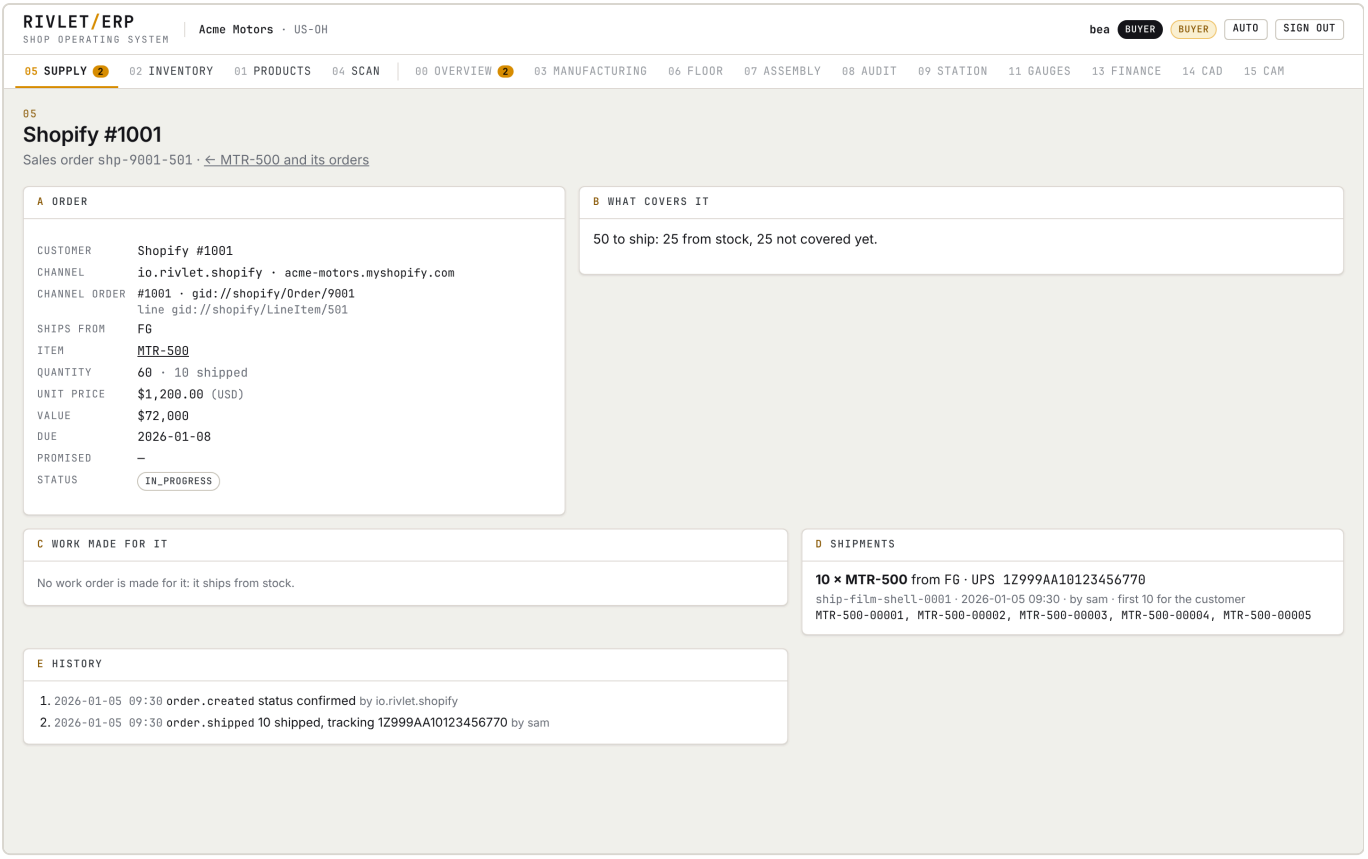
Movie: From order to ship (1 min 13 s). Shopify #1001, 60 motors, is recorded by the Shopify app; FG holds 35, so the demand planner proposes making 25 and, over its own limit, it waits for pat; pat approves; the job reaches ASM-1 labelled with the order; olivia makes it; it is wired and tested; and sam ships all 60 at FG. `floor-promo-order-to-ship`

Chapters: The order 0:00 · The planner proposes, a person decides 0:09 · At the station, with its label 0:20 · Wired and tested 0:33 · Shipped 0:37

The web order in Rivlet

A web order opens as a sales order record, titled by its label ("Shopify #1001", never a name or address), with its id under it. Open it from its item's record (under its orders), or from **13 Finance › Shipped this month** once it has shipped. It shows:

- the item, quantity, price (for those who read finance), due date and status;
- its **Channel** (the app and the shop's account) and **Channel order**, the shop's own reference;
- where it **Ships from** (FG), and **awaiting payment** while the shop is not paid;
- what covers it: how many are shipped, ready at FG and in work;
- the work made for it: each work order with its status and the station it is at now;
- each shipment: the units, serials, carrier and tracking number, and who shipped it;
- its history: recorded by the app, later shipped by a person.



A web order opened. Shopify #1001 as a sales-order record: its channel and channel order, where it ships from (FG), what covers it, the work made for it with its status and station, and each shipment with units, serials, carrier and tracking. `shell-record-sales-order-bea-desktop`

Until work is made for an order, **Work made for it** reads "No work order is made for it: it ships from stock", even when part of it is not covered yet and the demand planner's proposal to make the rest waits for approval. **What covers it** says how much is not covered ("50 to ship: 25 from stock, 25 not covered yet"), and the planner's proposal is in **Approvals**.

The FG tablet's **Ship** tab and **Shipped today** show the same orders and shipments at the station.

Paid or not

An order the shop has not been paid for yet is recorded as **awaiting payment**. It holds its stock, as the shop counts it, but nothing is made or bought for it. The FG tablet shows "awaiting payment" on it. When the shop is paid, the app updates the order and planning starts.

Changes and cancellations

When the customer changes or cancels the order, or is refunded, the app updates the plant's order to what the customer still wants (0 for a cancellation). Rivlet does the rest:

- stock held for it is freed, and the shop can sell it again;
- work for it that nobody has started is cancelled by the demand planner;

- work the demand planner proposed for it that still waits for a person is withdrawn, with why ("Shopify #1002 was cancelled"), so nobody approves it for nothing;
- work already started is left for the plant to decide;
- an order cannot go below what has already shipped.

A cancelled or completed order is closed. The FG tablet refuses to ship more on it: "This order is closed (done, or cancelled in the shop): nothing more ships on it."

A returned item comes back into stock only when a person books it in at the plant (a receipt, with the reason, for example "Return Shopify #1001").

What the shop may sell

The app tells Shopify how many it may sell from Rivlet's figure for what can be promised: stock on hand less open orders, at the location the shop ships from. It changes within seconds of any change in the plant. For a product made to order, the figure can go below zero: each order then becomes work.

What the app may and may not do

The Shopify app acts as itself, with only the rights the administrator granted when installing it:

- It may record its own shop's orders, update them, and mark them shipped.
- It may not touch another channel's orders, or orders that came through no channel: those are refused.
- It may not approve anything or sign off quality. Those stay with people.
- It cannot reach admin screens.
- If the administrator removes the app, its access ends at once.

Its own actions wait for the plant

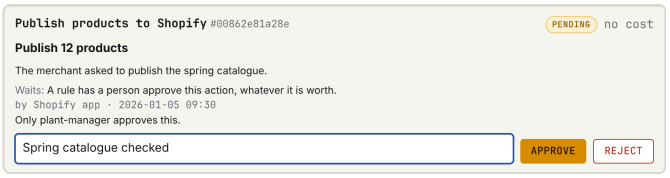
Some things the app does change the shop, not just the plant: publishing products, for example. These are the app's own actions, and the plant's rules decide them like an agent's. A rule may say that only a plant manager approves them (see Administration).

When the app proposes one, it appears in **Approvals** like any other proposal. Only the named role may approve it; anyone else who approves may reject it. While it waits, the app may take it back, for example when the shop is disconnected: it leaves the queue and the history shows it as withdrawn by the app, with the app's reason.

The card reads the action's title from the app's manifest, "Publish products to Shopify", asked for by "Shopify app". The history and the proposal's page say the same, also after the app is removed.



Only the named role approves. Shopify proposed publishing its catalogue, as itself. The rule names the plant manager, so Bea may reject but not approve: "Only plant-manager approves this." `shell-approvals-named-approver-bea-desktop`

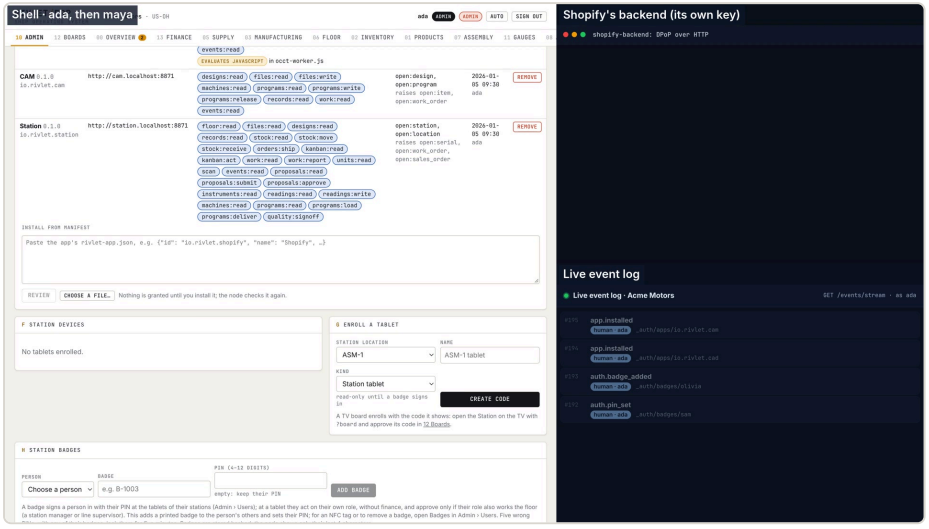


The plant manager decides. Maya, a plant manager, can approve the app's action; its value is $0, and still only her role decides it. `shell-approvals-named-approver-maya-desktop`

Connecting the shop

The administrator connects the shop once:

1. The Shopify app's operator gives you its manifest. The manifest names the key of the app's own service.
2. In **10 Admin › Apps**, install it from the manifest (see Administration). Grant only what it needs.
3. From then on, the app's backend proves who it is with its own key on every request, and gets short-lived access (15 minutes at a time). No password or token is passed between the two.



Movie: An app's own service identity (1 min 14 s). The Shopify app's backend makes its own Ed25519 key; Ada installs its manifest naming that key; the backend gets a 15-minute DPoP token with a proof of the key and uses it with a fresh proof per request; replayed proofs, other keys and admin routes are refused; it records a web order as itself and may amend only its own channel's orders; its own action waits for the plant manager; removing the app ends its token at once. `network-app-service-identity`

Chapters: Its own key 0:00 · The admin installs its manifest 0:04 · A token, for a proof of the key 0:21 · Every request carries a fresh proof 0:29 · It records and amends its own orders 0:37 · Its own action, governed in the plant 0:44 · Removed: its token ends at once 1:09

Security in plain words

This chapter is for everyone who wants to know why they can trust what Rivlet shows: who can do what, how a partner's quote is secured, and what the record keeps. No technical knowledge is needed.

Who can do what

- **Everyone signs in as themselves.** In the shell with a personal token; at a tablet with a badge and PIN. What you do is recorded as yours.
- **Your role decides what you may do.** The shell shows only what your role may use, and the node refuses anything else, whatever screen it comes from.
- **Your stations limit where you may work.** A floor operator's sign-in, badge and tablet work only at their own stations. A tablet works only at its own station.
- **Money is shown only to roles that read finance.** Everyone else sees "hidden". Tablets never show prices.
- **The admin administers.** The admin sets up people, devices and rules, and reads everything, but does no work and approves nothing.

Who can approve what

- **Big things wait for a person.** Agents and apps propose; the plant's rules decide what runs on its own, and the rest waits for a person.
- **Four eyes.** Nobody approves their own proposal. A purchase you propose always waits for someone else.
- **Limits.** Each role approves up to its limit: $1,000 for a station manager up to $250,000 for a plant manager. The CFO has no ceiling. Something with no known value is over every limit but the CFO's.
- **Named approvers.** A rule may say that only certain roles approve it, whatever its value.
- **Quality is quality's.** Only the quality role signs off a held unit.
- **Every decision is kept** with who decided, when, and their reason.

The details are in Managers and the CFO.

Signatures

Every message between partner factories (an RFQ, a quote, a purchase order, an acknowledgement, an alert) is signed with the sender's private key. Each node's identity, its DID, is made from its public key. Nobody else can sign in its name, and there is no outside authority to trust.

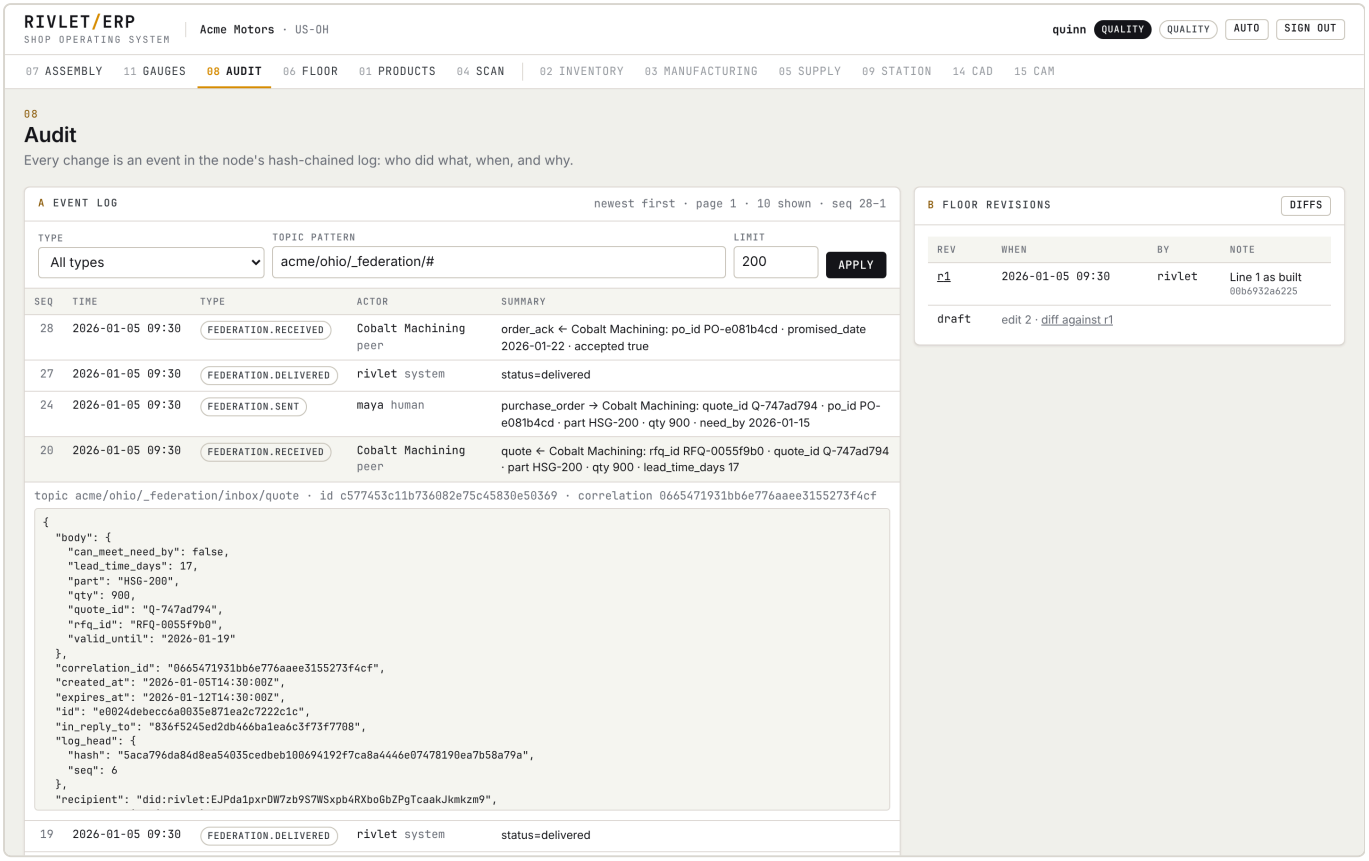
When a message arrives, Rivlet checks, in this order:

1. the signature matches the sender it claims;
2. the message is addressed to this node;

3. it is recent;
4. the sender's trust tier allows this kind of message;
5. it is well formed;
6. it has not been received before.

A message that fails any check is refused and nothing is recorded from it. A message received twice is recorded only once.

Each refusal is logged with the claimed sender and the reason, never the message's contents. Buyers see the count for each partner in **RFQs and quotes › C Refused envelopes**. A burst of refusals from one sender is counted, not logged one by one, so an attack cannot flood the log.



A signed quote, as recorded. Expanding a received message shows the whole signed envelope: sender and recipient DIDs, the quote, its correlation id and the Ed25519 signature. `network-audit-envelope-quinn-desktop`

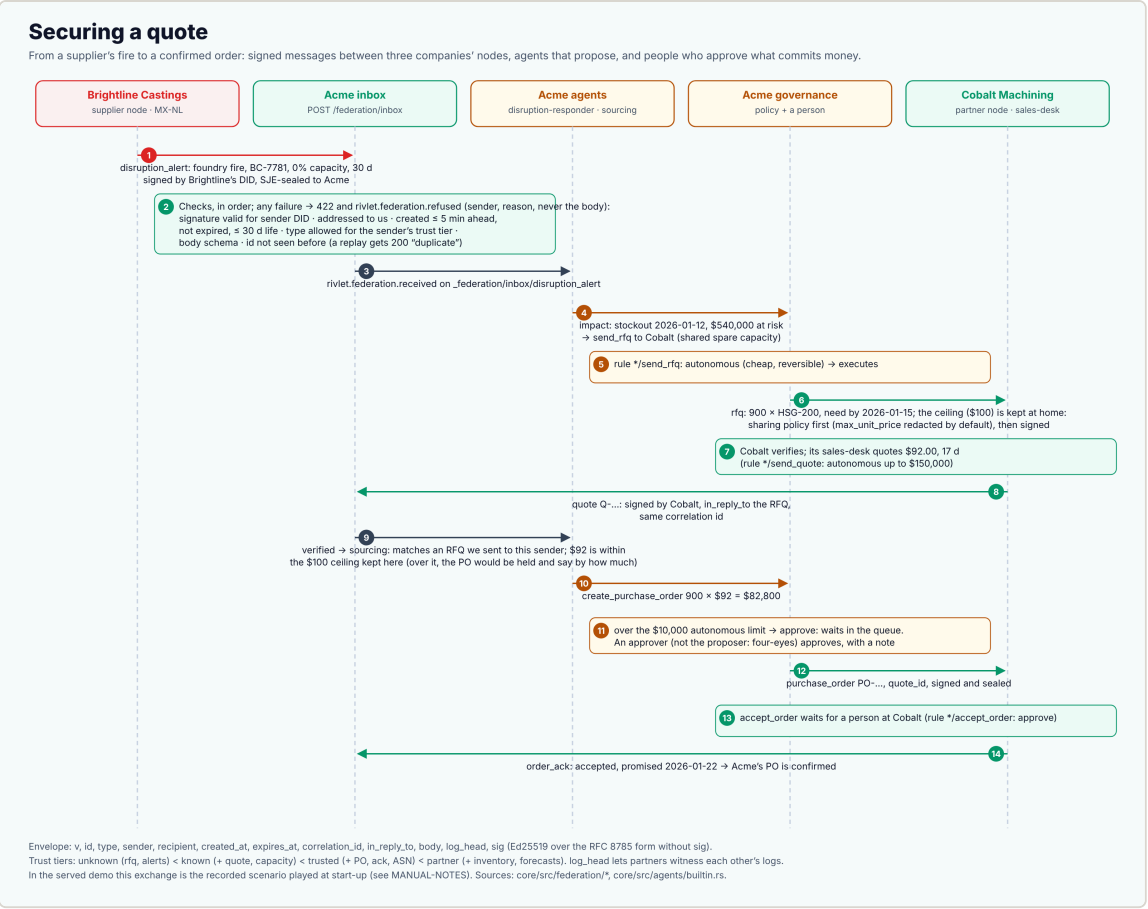
What partners are not told

Before a message is signed, fields the partner should not see are blanked out (redacted). Each partner has a list of what it is not told, set by the administrator.

By default, no partner is told your price ceiling: the most you would pay on an RFQ. Your node keeps the ceiling at home and checks every quote against it. The partner quotes without knowing it.

Messages to a partner whose encryption key is known are also sealed: only that partner's node can read them on the way.

How a quote is secured



Securing a quote. From a supplier's fire to a confirmed order: signed, verified messages between three nodes, the price ceiling kept at home, agents that propose and people who approve. `network-quote-sequence-all-1600`

In the demo, a supplier's foundry burns down, and a partner factory fills the gap:

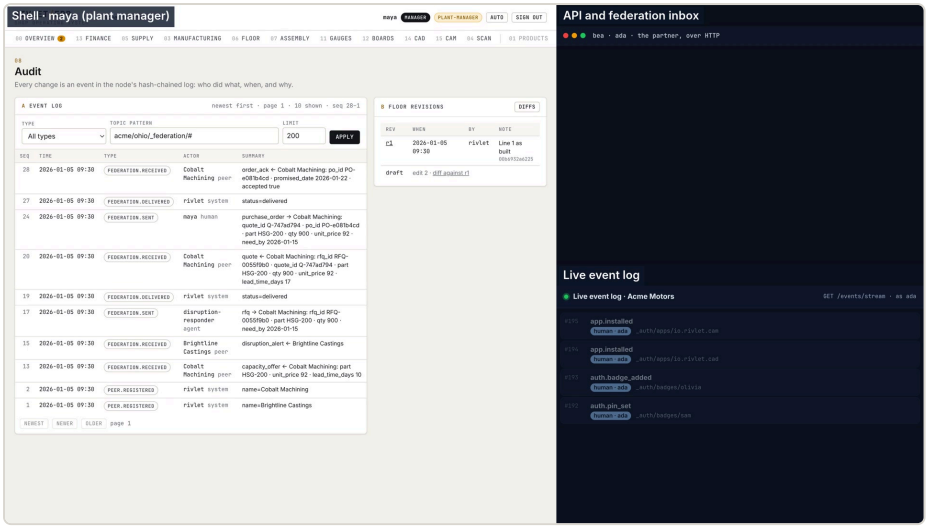
1. **The alert.** Brightline Castings, a supplier, sends a signed disruption alert.
2. **The RFQ.** Acme's agent works out what is at risk, and asks Cobalt Machining, a partner, for a quote: 900 housings. RFQs are cheap and can be undone, so the rules let the agent send one on its own. The ceiling, $100 a unit, stays at Acme: the RFQ goes out without it.
3. **The quote.** Cobalt answers with a signed quote: $92.00 a unit, 17 days. Acme checks the signature.
4. **The check at home.** Acme's sourcing agent compares $92.00 with the $100 ceiling it kept: within it. It proposes an $82,800 purchase order, which is over what agents may spend alone.
5. **A person decides.** A person approves the order, with a reason.
6. **The order and the acknowledgement.** Acme signs the purchase order and sends it. Cobalt answers with a signed acknowledgement and a promised date. The order is confirmed.

Every step is a signed record in Acme's log, and each is on the PO's record page under **The exchange it came from**.

The movie repeats the exchange live with a new partner, then shows the attacks being refused:

- a quote whose price was changed after signing: the signature no longer matches;

- a quote signed in the partner's name with another key: refused the same way;
- a stranger with a valid signature: it may ask and warn, but not quote;
- the real quote sent again: a duplicate, recorded once.



Movie: Securing a partner's quote (2 min 13 s). The recorded supplier fire and Cobalt exchange; then live: a partner registered as trusted, an RFQ approved by a second person and sent without the price ceiling, a signed quote verified and checked against the ceiling kept at home, forged and replayed quotes refused and counted, the PO approved and a signed acceptance confirming it.
`network-securing-the-quote`

Chapters: Recorded: the fire and the first quote 0:00 · Live: register a partner 0:19 · An RFQ needs a second person 0:27 · A signed quote, checked at home 0:56 · Forged and replayed quotes are refused, and counted 1:12 · Four-eyes approval and a signed acceptance 1:43

Tablets, TV boards and bridges

- **Each device has its own key.** A tablet, TV or bridge makes its own key when it is enrolled. The node keeps only a fingerprint of it. A copied token is useless without the device's key.
- **Enrollment codes work once,** for 10 minutes.
- **A tablet reads its own station only,** and does nothing until a person signs in with a badge and PIN.
- **Badges are stored scrambled.** Only their last four characters are ever shown. Five wrong PINs lock a person's badges for five minutes.
- **A TV board can only read its own station's board.** Nobody signs in on it.
- **Revoking a device ends it at once.** It must be enrolled again.

Apps acting as themselves

An app's backend (the Shopify app, for example) acts with no person signed in. It proves who it is with its own key on every request, and gets access for 15 minutes at a time. A proof is good once. It has

only the rights the administrator granted, never the right to approve, and only for its own orders. Removing the app ends its access at once.

The record

Everything that happens is an event in one log. Each event says what happened, when, and who did it: a person, an agent, an app, a device, or a partner.

Each record in the log carries a fingerprint of the one before it. Changing or removing an old record breaks the chain, and a check of the log shows where. Partners keep a note of how far your log has reached, so a log cut short is caught too.

An event made at a Station tablet names the person who made it and the tablet they used. The same person at a desk computer is recorded without a device, and a device acting alone (an edge bridge) is recorded as the device.

What you can check yourself

- **08 Audit** shows the log: filter it and open any event.
- **05 Supply › RFQs and quotes** shows each quote's signature check and how it compares with your ceiling.
- **10 Admin › Partners** shows what each partner is not told, and whether messages to it are sealed.
- Every record (a PO, a proposal, a partner, an item, an order) shows how it was decided and its history.

Troubleshooting

When Rivlet cannot do something, it says why in plain words. This chapter lists what you may read, where, what it means and what to do. Look for the first words of the message.

At a Station tablet: signing in

What you see	What it means	What to do
"Wrong PIN. Try again."	The PIN does not match. Your badge is still read.	Enter your PIN again.
"Too many wrong PINs: your badges are locked for a few minutes."	Five wrong PINs, with any of your badges.	Wait five minutes, or ask your supervisor.
"This badge isn't registered. Ask your supervisor."	Nobody holds this badge, or it was removed.	Ask your supervisor or administrator to add it (Admin › Users › Badges).
"You do not work at this station. Sign in at one of yours, or ask an admin to add it."	This station is not one of yours.	Use a tablet at one of your stations, or ask an admin to add this one.
"This person can no longer sign in."	Your access was removed.	Ask your administrator.
"That is a Rivlet label, not a badge: sign in first, then scan it on the Scan tab."	You scanned a label before signing in.	Sign in with your badge first.
"Tap to sign in needs Chrome on Android or a USB NFC reader."	This tablet cannot read NFC tags.	Scan or type your badge instead.
"Signed out after 10 min without activity"	Nobody tapped for 10 minutes.	Sign in again. Work you queued is still yours.
"Read-only: scan or tap your badge, or tap Sign in, to work."	Nobody is signed in.	Sign in.
"Your role can only look here."	Your role does not do this station's work.	Ask your supervisor if you should.

At a Station tablet: work, stock and kanban

What you see	What it means	What to do
"Nothing ready here yet: 1.0 h of released work is on its way."	No job's current operation is at this station yet.	Wait; the jobs on their way are listed under it.
"Nothing left to report here."	Every unit of this operation is reported.	Tap **Complete**.
Report stays off after you set Scrap	A scrap needs a reason.	Pick a reason under "Why is it scrap?".
"This tablet can only move stock in or out of ASM-1. Move it from a CNC-1 tablet, or ask a supervisor."	One end of a move must be this station.	Move it from a tablet at the other station, or ask a supervisor.
"This tablet works at ASM-1 only. Do it at a tablet where the work is, or ask a supervisor."	The change is for another station.	Do it at that station's tablet.
"WIP limit at ASM-1 is 60 and it holds ...: ... more would exceed it."	The station would hold more than its WIP limit.	Move less, or move work on first.
"Only people who receive goods can book stock in from outside. Move it from where it is, or ask a buyer."	Receipts are for buyers and planners.	Move it from where it is, or ask a buyer.
"Nothing found for NOPE-0000."	The code matches nothing in the plant.	Check the label, or type the code.
"...: a refill is already requested (waiting for approval)"	A refill for this card already waits for a person.	Nothing; it is on its way once approved.
A reading "refused: Instrument DMM-07 was due for calibration on ...: recalibrate it before use."	The instrument is past its calibration date.	Choose a calibrated instrument above and measure again. Tell quality.
"Machine has rev 1" on a program	A newer revision is released than the one loaded.	Load the released revision and tap **Confirm loaded**.
"Run this one: it replaces O1000 (...), which is no longer offered here."	A newer program replaced the one you may have loaded.	Load the program the card shows, and tap **Confirm loaded**.

At a Station tablet: approvals

What you see	What it means	What to do
No **Approvals** tab	Your role does not approve at a tablet (only station managers and line supervisors do), or nobody is signed in.	Approve from your own sign-in in the shell, or leave it for a supervisor.
"Value not shown to you"	It is worth more than you may approve, and a tablet never shows prices.	Leave it for someone whose limit covers it.
Approve and **Reject** are off	A decision needs a reason.	Type one under **Reason**.
"It is worth more than you may approve (up to $5,000): leave it for someone whose limit covers it."	It is over your approval limit.	Leave it, or reject it with a reason.
"You asked for this: someone else approves it."	Nobody approves their own request.	Leave it for another approver.
"Only plant-manager approves this: leave it for them."	A rule names who approves it.	Leave it for that role. You may still reject it.
"Already decided: someone approved, rejected or withdrew it in the meantime."	Someone else decided first.	Nothing: it has left the list.
"Cannot reach the node right now: decide once the tablet is back online."	The tablet is offline. Decisions are never queued.	Decide again once it is back online.

At a Station tablet: the outbox and the network

What you see	What it means	What to do
An amber bar: offline	The tablet cannot reach the node.	Keep working. Changes wait as "queued" and are sent once, by themselves.
"Cannot reach the node right now."	The network or the node is down.	Wait. If it lasts, tell your supervisor.
"The node could not do it just now: try again in a moment."	The node was busy or had a problem.	It is tried again by itself; wait.
"Too many tries: wait a moment, then try again."	Too many requests in a short time.	Wait a moment.
A red bar, and **not done** in the Outbox	The node refused one of your changes. The reason is on it.	Read the reason, do what it says, then tap **Dismiss**. The bar goes when you sign out.
"Not done: ..."	A change was refused; the reason follows.	As above.
"This tablet is no longer enrolled"	An admin revoked the tablet, or the node was reset. Unsent changes are kept.	Ask for a new enrollment code, then tap **Re-enroll this tablet**.
"That code is not known or has expired. Ask for a new one."	The enrollment code is mistyped, used or older than 10 minutes.	Ask your administrator for a new code.

At the FG tablet: shipping

What you see	What to do
"Orders ship from FG: sign in with your badge to ship them."	Sign in.
"Not enough here: FG has 30 MTR-500, not 55. Ship the 30 that are here, or wait for the rest."	Ship what is here, or wait for the rest.
"FG has no MTR-500 now: wait until they arrive, or ask a supervisor to check the stock."	Wait, or have the stock checked.
"Only 55 left to ship on this order (some shipped already, or the shop changed it): ship 55 at most."	Lower the units.
"Nothing is left to ship on this order: it shipped already, or the shop changed it."	Do not ship it.
"Unit … was shipped in …: take it out of the box and scan another unit."	Swap the unit and scan another.
"No unit … is known: check the label, or leave the serial out and the node picks the unit."	Check the serial label.
"This order is closed (done, or cancelled in the shop): nothing more ships on it."	Check with the office.
"This tablet works at ASM-1, and the order ships from FG: ship it from a tablet at FG, or ask a supervisor."	Ship from the FG tablet.
"Already shipped: … was recorded before, so it shipped once."	Nothing: it shipped once.

On a TV board

What you see	What it means	What to do
A code and a QR code	The TV is not a board yet.	A supervisor approves it in **12 Boards**.
"Offline since 09:30", figures greyed	The TV lost the network. The numbers are old.	Check the TV's network. It is live again by itself.
"This TV is no longer a board"	An admin revoked it, or the node was reset.	Tap **Show a new code** and have it approved again.

In the shell

What you see	What it means	What to do
An error under the sign-in field	The node does not know this token.	Check you copied all of it. Ask your administrator for a new one.
"You are signed out. Your access to this node was removed, or your token is no longer valid. If you still need it, ask your administrator for a new token."	An administrator removed your access, or your token changed.	Ask your administrator.
"Admin needs an admin role."	Only admins open 10 Admin.	Ask your administrator.
"Your role does not approve proposals."	You may see the queue but not decide.	Leave it for an approver.
"Decision refused · 403 Forbidden: policy: bea proposed this; someone else must approve it"	You cannot approve your own proposal.	Ask someone whose limit covers it.
"Decision refused 403 Forbidden: maya can approve up to $250,000; this is $320,000"	It is over your approval limit.	Leave it for someone with a higher limit (the CFO has none).
"Decision refused 403 Forbidden: sam can approve up to $5,000; this is worth more"	It is over your approval limit, and your role does not see what it is worth.	Leave it for someone with a higher limit.
"Only plant-manager approves this." and **Approve** is off	A rule names who approves it.	Leave it for that role. You may still reject it.
"Waiting for approval: it is over what may be bought without one, so someone whose limit covers $16,000 decides"	Your proposal waits for a second person.	Nothing; it is in the approvals queue.
"409 Conflict: order WO-2001 already exists"	That work-order id is used.	Choose another id, or leave it blank.
"Publishing needs an approver: a Plant manager publishes it; your edit is saved and waiting."	Your floor layout change is saved; a plant manager publishes it.	Tell the plant manager.
"Releasing needs an approver" (CAD's Release is off)	Releasing designs is for a plant manager.	Save, then ask a plant manager to release it.

What you see	What it means	What to do
Generate is off in CAM	A check has an error.	Click the error to see where; fix it.
"Not a manifest"	The pasted app manifest is not valid JSON.	Paste the whole manifest file again.
"hidden" instead of a price	Your role does not read finance.	Ask someone who does.
"Some figures could not load"	Part of the page could not reach the node.	Reload the page.
A time looks hours off from your watch	Screens tell the plant's time, not your device's: you are in another time zone than the plant.	See Times on the screen.
"You do not work at CNC-1: its queue is not shown here." in CAM	CAM shows the queues of the stations you work at only.	Nothing is wrong with the programs. Ask an engineer or supervisor for the queues at other stations.
"Your role reads programs; making them is for a role with programs:write." in CAM	Your role may read programs but not make or change them.	Ask an engineer.

Cameras

What you see	What to do
"No camera found: plug in a webcam (or check it is switched on), then try again."	Connect a camera, then **Try again**.
"The camera is blocked for this site: allow it in the browser's site settings, then try again."	Allow the camera in the browser, then **Try again**.
"The camera is in use by another app or browser tab: close it there, then try again."	Close the other app or tab.
"The camera needs a secure page: open this over https or on localhost."	Ask whoever set up the tablet to use the https address.
"This browser cannot read barcodes from the camera."	Use a scanner or type the code.

Still stuck?

- Read the record's history: every record shows who changed it and why.
- **08 Audit** shows every event, with who made it.
- Ask your supervisor, then your administrator. Tell them the exact words you saw and the time.

Glossary

The words Rivlet uses, in plain terms.

Word	What it means
Acknowledgement	A partner's signed answer to a purchase order, with the date it promises. It confirms the order.
Actor	Who did something, as the log records it: a person, an agent, an app, a device (such as a bridge) or a partner.
Agent	A small program on the node that watches the plant and proposes actions, such as the kanban replenisher, the demand planner or the sourcing agent. The plant's rules decide what it may do on its own.
App	A program installed on the node, such as CAD, CAM, the Station or the Shopify app. It has only the rights the administrator granted.
Approve, approval	A person's decision to let a held proposal run. Each role approves up to its limit.
Approval limit	The most a role may approve. The CFO has none.
Approvers	The roles a rule names to approve what it holds.
Audit	Section 08: the event log.
Autonomy	How much an agent may do on its own under a rule: observe, suggest, approve (waits for a person) or autonomous (runs).
Awaiting payment	A web order the shop has not been paid for. It holds stock, but nothing is made for it yet.
Badge	A printed card or NFC tag that, with your PIN, signs you in at a Station tablet.
Bill of materials (BOM)	The parts that go into an item.
Board, TV board	A TV at a station that shows its day: good today against the target, what is running, what is next.
Bridge, edge bridge	A small program on a computer next to an instrument or machine. It sends readings and delivers programs, as a device of its own.
CAD	The app where parts are drawn and saved as designs.
CAM	The app that turns a released design into a machine program.
Ceiling, price ceiling	The most you would pay a unit on an RFQ. It is kept at your node and never told to the partner.
Check	A measurement each unit must pass at an operation, with a spec band and what happens on fail.

Word	What it means
Cover	What covers an order: units made for it, stock held for it, work still in progress for it, and what is short.
Demand planner	The agent that turns customer orders into work orders (or purchase orders) for what stock does not cover.
DID	A node's identity (`did:rivlet:…`), made from its public key.
Draft	A design, program or floor layout that is saved but not released or published.
Enroll	To make a device (tablet, TV, bridge) part of the plant, at one station.
Event	One thing that happened, recorded in the log with its actor and time.
FG	Finished goods: the location orders ship from.
Four eyes	The rule that nobody approves their own proposal.
Genealogy	A unit's history: where and by whom each operation was done, what went in, and its readings.
Governance	The plant's rules for what agents and apps may do on their own, and who approves the rest.
Hold, on hold for quality	A unit that failed a "Quality decides" check. Only quality may pass it.
Kanban card, loop	A card per container of a part. Pulling the card when you open the container signals a refill. A loop is the cards for one part at one place.
Label (order label)	How an order is shown on queues and boards, for example "Shopify #1001".
Manifest	An app's description: who it is, what rights it asks for, its own actions and its service key.
Node	The Rivlet Core server for one site. It keeps the plant's record.
O-number	A machine program's number, given when it is released (for example O1001).
Operation (op)	One step of a routing, done at one station (op 10, op 20…).
Outbox	Where a tablet keeps each change until the node has it. It works offline.
Partner (factory)	Another company's Rivlet node you trade with.
Persona	Which sections you see first and where you start. It gives no rights.

Word	What it means
PIN	Your personal number, 4 to 12 digits, used with any of your badges.
Proposal	An action an agent, an app or a person asks for, held or run by the rules.
Publish	To make a floor layout draft the live layout. A plant manager publishes.
Purchase order (PO)	An order to buy from a supplier or partner.
Queue	The jobs whose current operation is at a station.
Receipt	Booking goods into the plant from outside.
Redaction	A field blanked out before a message to a partner is signed.
Release	To make a draft design or program the next official revision. A plant manager releases.
Replaced (program)	A released program that a later release for the same operation and machine took over from. The station no longer offers it; CAM marks it "Replaced by O1001". Releasing it again makes it the one to run.
Revision (rev)	A numbered released version of a design, program or floor layout.
RFQ	Request for quotation: asking a partner for a price.
Role	What you may do, and up to what value you approve.
Routing	The operations that make an item, each at a station.
Sealed	Encrypted so only the receiving partner's node can read it.
Scope	One right, such as reporting work or reading finance. A role is a set of scopes.
Service (an app's own)	An app's backend acting as the app itself, with no person signed in.
Shell	The web app for the office, at the node's address under `/app/`.
Signature	Proof that a message came from its sender and was not changed.
Standard cost	What an item is valued at in 13 Finance and in approval limits: its own cost, or else its bill of materials rolled up. A cost nobody knows is "unknown", never 0.
Station	A work centre (ASM-1, TEST, FG...), and the tablet that is enrolled there.
Stoppage	A time a station is down, with its reason.

Word	What it means
Supermarket (SM-1)	The store of parts near the line that kanban refills come from.
Target, daily target	The units a station should make today.
Token	Your personal key to the shell. Keep it private.
Trust tier	How much a partner is trusted: unknown, known, trusted or partner. It decides what messages you exchange.
Withdraw	To take back a proposal while it waits for a person. Only whoever proposed it may (a person, an app, or the demand planner when the need is gone); the history shows who withdrew it and why.
Witness (log head)	The head of a partner's log, signed into each message it sends, which your node keeps. If the partner's log later goes back or is rewritten, the heads disagree and RFQs and quotes says so.
Work order (WO)	An order to make a quantity of an item, following its routing.
WIP limit	The most work in progress a station may hold.