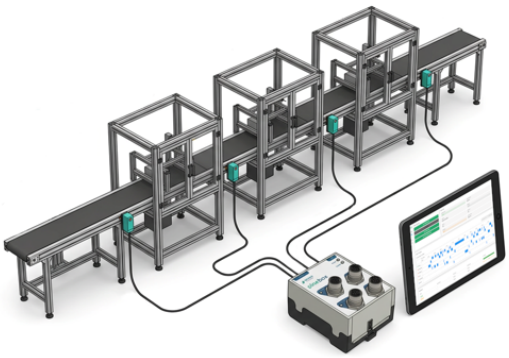
CedarInsight

User manual

www.pinetek-networks.com



CedarInsight

Pinetek Networks GmbH, Bachstelzenweg 8, 88677 Markdorf
info@pinetek-networks.com

2026-07-28

Table of Contents

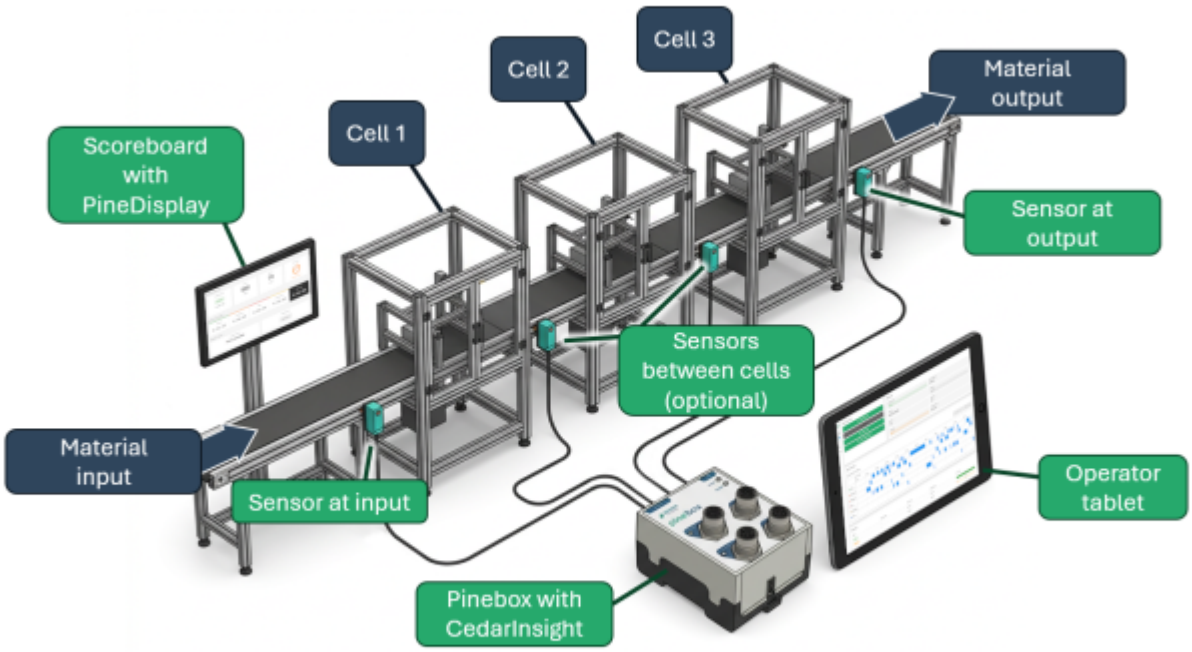
1. Introduction

1.1 About CedarInsight

CedarInsight is a manufacturing OEE (Overall Equipment Effectiveness) monitoring system by Pinetek Networks. It runs on the Pinebox, Pinetek Network's industrial edge device next to your production line and provides:

- Real-time production state tracking: the line state (Running, Down, Changeover, ...) is derived automatically from counter signals.
- OEE calculation: Availability, Performance and Quality per production run, aggregated over any time period.
- Downtime analysis: reason capture at the line, Pareto analysis, history drill-down.
- Bottleneck detection: live and historical, per process stage.
- Sensor, takt and cycle monitoring: throughput sensors, takt timers and energy-based cycle detection.
- Automated PDF reporting: run-end, scheduled or on-demand reports with e-mail dispatch.
- AI Assistant: chat-based analysis of your production data (licensed feature).

All data acquisition at the machine (counters, timers, sensor and cycle signals) is performed by the Pinebox core functions. CedarInsight consumes this data over the network and turns it into OEE metrics, analyses and reports.



1.2 Intended audience and user roles

This manual addresses three audiences, which correspond to the access levels in the application:

Role	Typical user	Relevant chapters
Operator	Line personnel: starts/stops runs, records downtime reasons, confirms notifications	4, 5
Supervisor	Shift/production lead: analyses OEE, downtime and bottlenecks, configures the line	4..7
Admin	Plant IT / system owner: installation, users, reports, backup, updates, licensing	all, especially 3, 8, 9

A fourth level, Basic, is a view-only account for wall displays (Scoreboard only, no password).

1.3 Scope of this manual

This manual covers the CedarInsight application: its user interface, configuration and administration.

It does not cover the Pinebox device itself: wiring, IO-Link master configuration, counter/timer setup and device-level networking are described in the separate Pinebox Manual.

1.4 Related documents

Reference	Document	Content
[PBX]	*Pinebox Manual* (separate document)	Pinebox hardware installation, sensor wiring, counter/timer/cycle configuration, device network setup

Throughout this manual, the marker [PBX] indicates that the described functionality depends on configuration performed on the Pinebox, described in the Pinebox Manual.

1.5 Typographic conventions

- **UI elements** are written in bold: press **Save**.
- *Menu paths* are written as: *Line Config → Counters*.
- `Monospace` is used for values, keys, file names and URLs: `http:<device-ip>:8000`.
- [PBX] marks a cross-reference to the Pinebox Manual.

1.6 Notes and warnings

This symbol is used to show information that is relevant for the operation of CedarInsight.

This symbol is used to show areas of special attention that need to be considered for the safe operation of CedarInsight.

2. System Overview

2.1 What is OEE?

OEE (Overall Equipment Effectiveness) is the industry-standard metric for how effectively a production line is used. It is the product of three factors:

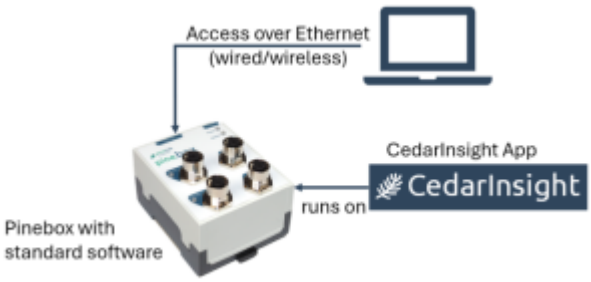
Factor	Question answered	Losses captured
Availability	Did the line run when it should have?	Unplanned downtime (breakdowns, faults)
Performance	Did it run at full speed?	Slow cycles, minor stops
Quality	Did it produce good parts?	Scrap, rejects

```
OEE = Availability × Performance × Quality
```

An OEE of 100% means: producing only good parts, at maximum speed, with no unplanned stops. World-class discrete manufacturing is typically considered to be around 85%.

CedarInsight computes these factors automatically from the counter signals of your line; the exact formulas are documented in chapter 10. OEE Calculation Reference.

2.2 System architecture at a glance



2.2.1 CedarInsight application

CedarInsight runs as an app on the Pinebox. It consists of:

- a backend (data acquisition, state machine, database, report generation), and
- a web interface served on port `8000`. No client installation is needed; any modern browser on connected network can access it at http://<device-ip>:8000. See the connection option in the Pinebox manual: Pinebox Manual, Configuration Update [PBX].

Production data is stored in a MariaDB database on the device itself. Normal operation has no cloud dependency. Data leaves the device only where explicitly configured: backups (chapter 9.3), e-mailed reports (chapter 8) and the AI Assistant (chapter 6.5).

2.2.2 Pinebox as data source [PBX]

The Pinebox standard software (running in parallel to the CedarInsight app) is the field-level data acquisition source. It connects to the IO-Link machine sensors and makes the acquired signals available to CedarInsight in two ways:

- **Polled services**: the Pinebox exposes counters, timers and cycles as simple network services; CedarInsight reads them cyclically.

- **Pushed values**: process sensor values are sent *by* the Pinebox side *to* CedarInsight's data interface.

Signal	Interface	Used by CedarInsight for
Piece counters	polled from Pinebox `/counter/values`	Production quantities, state detection, throughput per stage
Takt timers	polled from Pinebox `/timer/values`	Takt/cycle-time monitoring
Machine cycles (energy-based)	polled from Pinebox `/cycles`	Cycle stations: cycle duration, energy, stall detection
Process sensors	pushed to the CedarInsight API	Additional analogue/process values (temperature, pressure, speed, …)

Using more than 4 sensor inputs In the standard single-device installation the Pinebox services run on the same host as CedarInsight and are reached at `localhost:18080`; a Pinebox can equally be addressed by its IP on the plant network. Multiple Pinebox devices can be used; each counter, takt source and cycle station stores its own host/port. The “external” Pineboxes are standard devices without the CedarInsight app running.

The details for integrating additional Pinebox devices are described in chapter 11.

How the signals are wired and configured on the device side is described in the Pinebox Manual [PBX]. Chapter 11 of this manual describes exactly which Pinebox services CedarInsight consumes.

2.2.3 Data flow

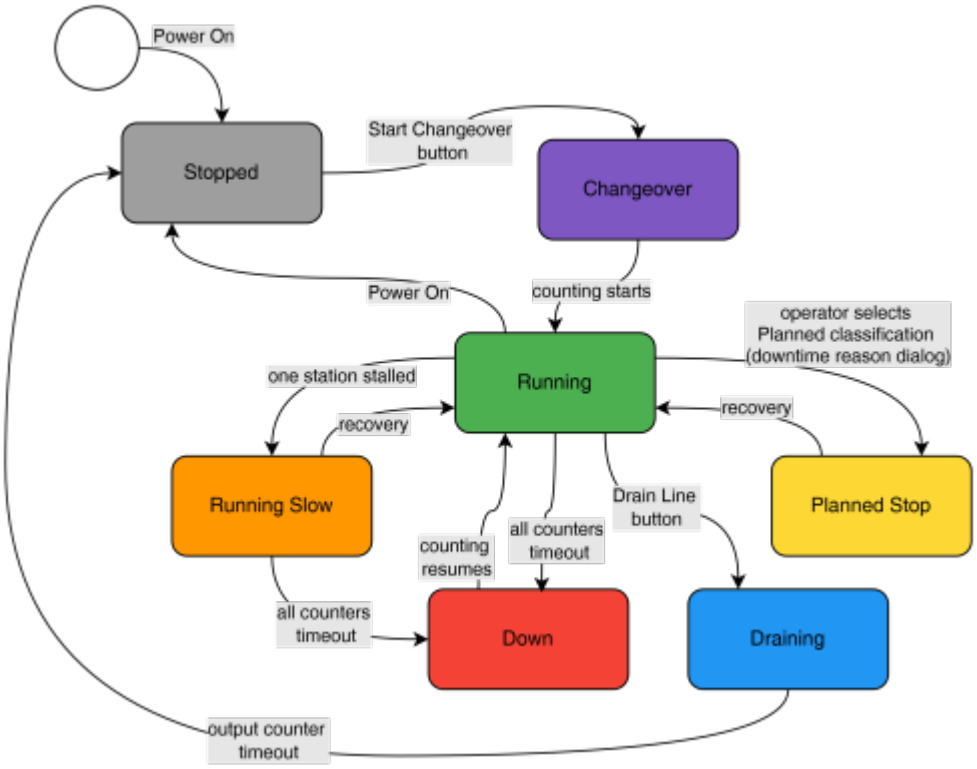
1. The Pinebox counts parts at each process stage and measures timers/cycles. [PBX]
2. CedarInsight polls the counter values cyclically (see chapter 11).
3. The state machine compares current with previous values, detects activity and timeouts, and derives the production state.
4. Every state change is written to the database as a state event; counter readings are stored as time series.
5. When a production run ends, CedarInsight computes and stores the run's OEE figures.
6. The analysis views, the AI Assistant and the report engine all read from this database.

2.3 The production state model

2.3.1 States

The line is always in exactly one of seven states:

State	Meaning
Stopped	No production run active
Changeover	Product change / setup in progress; run is prepared
Running	Production run active, counters advancing normally
Running Slow	Run active, but throughput below expectation
Down	Unplanned stop; counters stopped unexpectedly
Planned Stop	Deliberate pause (break, planned maintenance)
Draining	Input stopped, line runs empty before the run ends (no bottleneck detection)



2.3.2 How counter signals drive state transitions

CedarInsight derives states from the movement of the stage counters [PBX]:

- **Changeover → Running**: production starts when a product is selected and counting begins.
- **Running → Down**: a stage counter stops advancing for longer than its configured **timeout** (see *Line Config → Program → Stages*, chapter 7.1) → planned/unplanned stop. The state is set by the operator. When counting resumes, the state returns to **Running** automatically.
- **Running → Running Slow**: throughput falls below the expected rate without stopping completely (one station stalls minimum)
- **Running → Draining**: triggered manually; the input stage is expected to stop while the rest of the line runs empty. When the output counter times out, the run ends and the state becomes **Stopped**.
- **Stopped**: ending the run (manually or after draining) always returns to Stopped.

Every period spent in **Down** must be justified with a downtime reason (chapter 5.4). Time in **Planned Stop** counts as planned downtime and does not reduce Availability.

2.3.3 State events

Each contiguous period in a state is stored as one state event with start, end and duration. State events are the basis for the shift timeline (chapter 5.2), the state distribution and downtime analyses (chapter 6) and the Availability calculation (chapter 10).

2.4 Key concepts and terms

Term	Meaning
Program	A line configuration: the set of process stages, their counters, sensors, takt sources and detection parameters. Typically one program per line layout or product family.
Product	An article produced with a program. Carries part number and ideal cycle time. Several products can share one program.

Term	Meaning
Production run	One continuous production of one product: from changeover/start to run end. OEE is calculated per run.
Process stage	One counting point in the line (e.g. filler, capper, packer). Each stage has a role: *Input*, *Output*, *Defect/Reject* or *Intermediate*.
Counter	A piece counter acquired by the Pinebox and mapped to a stage. [PBX]
Takt	Cycle-time measurement from a Pinebox timer. [PBX]
Cycle station	A machine monitored via energy-based cycle detection (power rise/fall), e.g. a press or moulding machine. [PBX]
Sensor	A process value (temperature, pressure, …) pushed into CedarInsight. [PBX]
Downtime reason	Category + reason assigned to a Down period, from the catalogue configured in chapter 7.7.
Units in / Units out	Parts entering the line (input stage) / good parts leaving the line (output stage).
Target quantity	The planned quantity for a run, entered at changeover.

A full glossary is provided in Appendix A.

3. Installation & Commissioning

CedarInsight is normally delivered pre-installed on the Pinebox. In that case, start at section 3.2 (connecting the Pinebox) and 3.5 (license activation).

3.1 Prerequisites

Item	Requirement
Pinebox	Pinebox with CedarInsights installed and licensed
Network	Clients need HTTP access to port `8000` of the device
Browser/End device	Current Chrome, Edge or Firefox

Network ports used by the system:

Port	Service
`8000`	CedarInsight web interface and API
`18080`	Pinebox data services (counters, timers, cycles) [PBX]
`18090`	Pinebox Software update manager

3.2 Connecting the Pinebox [PBX]

Install and configure the Pinebox as described in the Pinebox Manual: sensor wiring, counter/timer configuration and network setup are performed there.

For CedarInsight two things matter:

1. The Pinebox must be reachable over the network. In the standard single-device installation the Pinebox services run on the same host (`localhost:18080`).
2. The **counter keys** configured on the Pinebox [PBX] are what you will later map to process stages in CedarInsight (chapter 7.3). Note them down or use the built-in scan functions.

CedarInsight can read from several Pinebox devices simultaneously. Each counter, takt source and cycle station stores its own host and port.

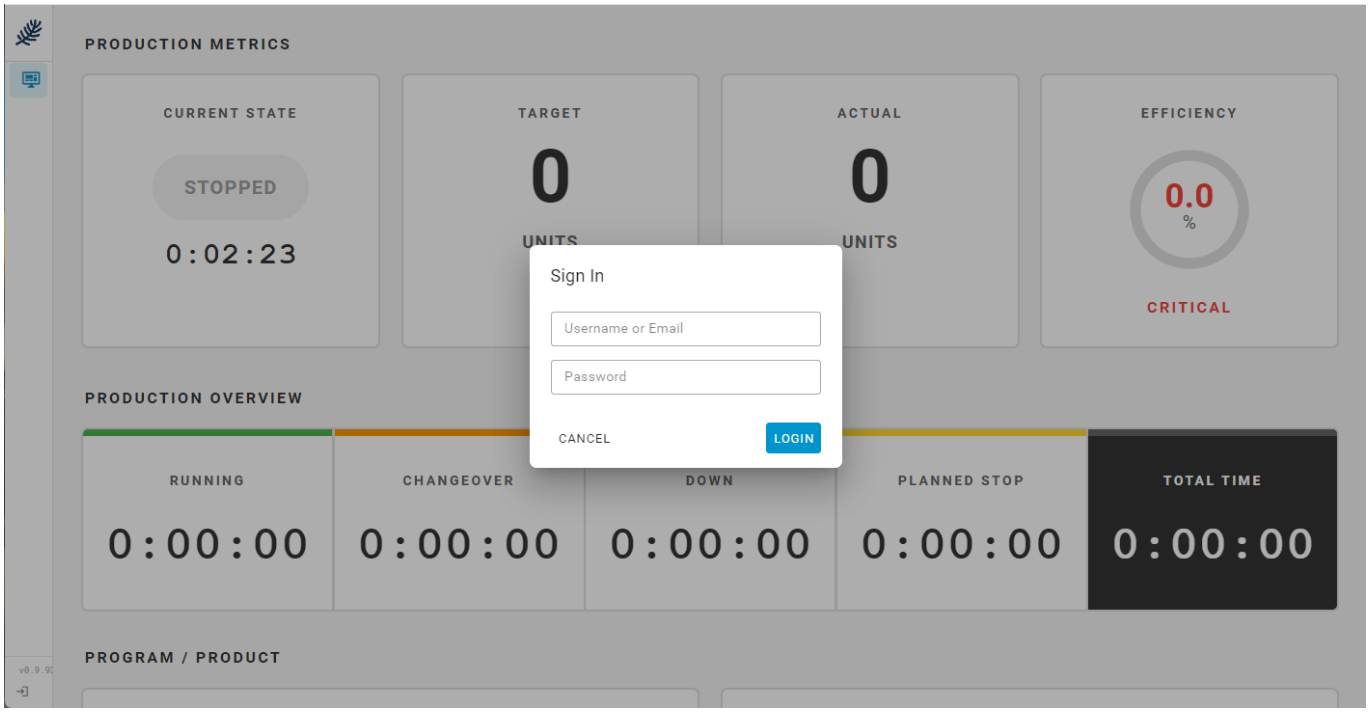
3.3 Installing CedarInsight

If the software is not pre-installed, install it from the Pinetek installation package following the instructions delivered with the package. The default installation system on the Pinebox is used (see Pinebox Manual, Configuration Update [PBX]). The pfw-File will:

- install system dependencies (including MariaDB if missing),
- create the database and application user,
- register CedarInsight as a `systemd` service, and
- start the service on port `8000`.

Please note that installation may take several minutes.

After installation, open `http://<device-ip>:8000` in a browser.

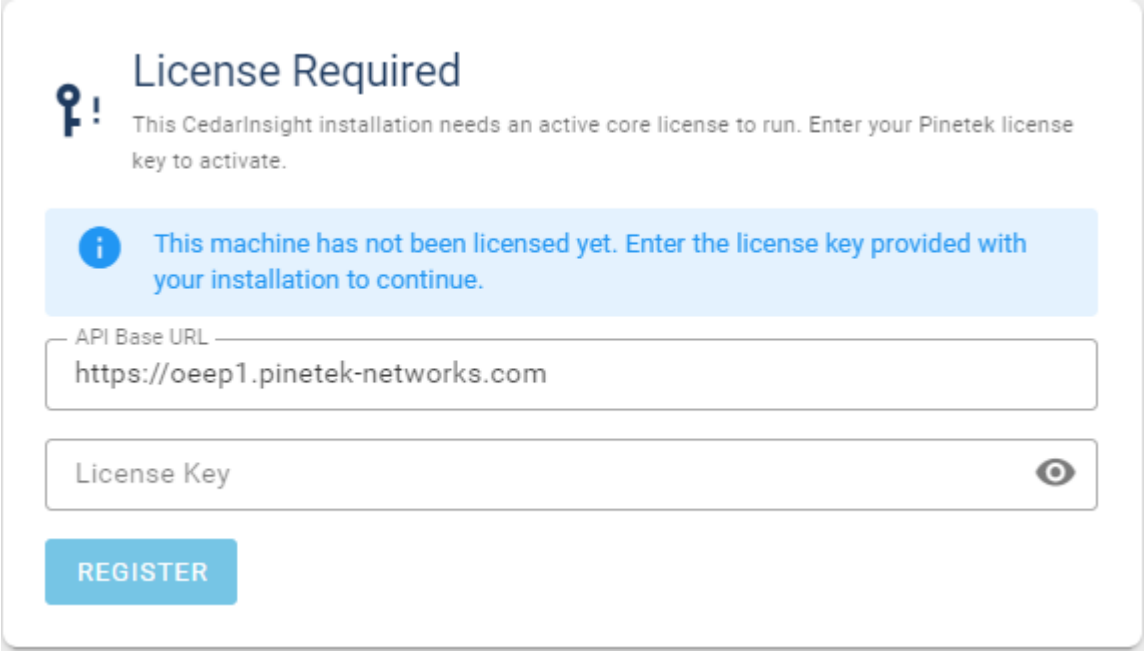


3.4 First start

On first start the application initialises its database automatically and seeds the defaults (timezone, predefined report blocks, default configuration). No manual database steps are required.

3.5 License activation

CedarInsight requires an active core license to run. Without it, the web interface shows only the **License Required** screen:



To activate:

1. Open the web interface. If the machine is not yet licensed, the License Required screen appears automatically.

2. Enter the **Pinetek license key** delivered with your installation.
3. Press **Register**. The device registers itself with the Pinetek license server using its unique machine ID.

The license is verified periodically in the background (once per day) and cached locally, so short internet outages do not interrupt operation. Optional features (Cloud Backup, AI Chat, managed SMTP) are separate entitlements on the same license, see chapter 9.6.

Registration requires internet access to the Pinetek license server. If the device is offline, contact Pinetek Networks.

3.6 Commissioning checklist

Recommended order for setting up a new line. Details for each step are in the referenced chapters.

1. License activated (3.5) and application reachable at `http://<device-ip>:8000`.
2. System settings: timezone and language (*System* → *System*, chapter 9.1).
3. Users created with appropriate roles (*System* → *Users*, chapter 9.2).
4. Counters discovered from the Pinebox and registered (*Line Config* → *Counters*, chapter 7.3). [PBX]
5. Program created: process stages defined and counters mapped, timeouts set (*Line Config* → *Program*, chapter 7.1).
6. Products created with part number and ideal cycle time (*Line Config* → *Product*, chapter 7.2).
7. Optional: sensors, takt sources and cycle stations registered and assigned (chapters 7.4–7.6). [PBX]
8. Downtime catalogue: categories and reasons defined (*Line Config* → *Downtime*, chapter 7.7).
9. SMTP configured and test mail sent (*System* → *SMTP*, chapter 9.4), required for e-mailed reports.
10. Report templates created or adapted (*System* → *Reports*, chapter 8).
11. Backup target configured and first backup executed (*System* → *Backup*, chapter 9.3).
12. Test run: perform a changeover, produce a few parts, verify state detection and counter values on the Production Timeline (chapter 5.2).

4. Getting Started

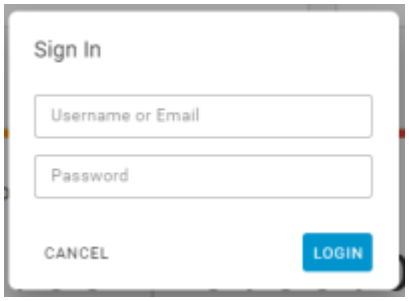
4.1 Accessing the web interface

Open a browser on any device in the plant network and navigate to:

```
http://<device-ip>:8000
```

No client software is required. The interface is responsive and can be used on desktop PCs, panel PCs and tablets.

4.2 Signing in and user roles



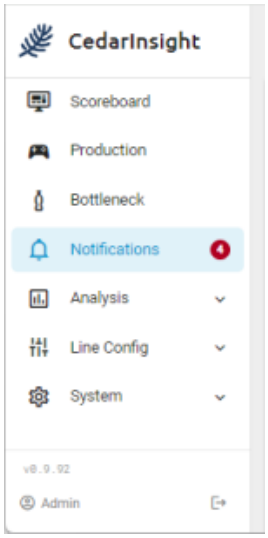
Sign in with your **username or e-mail** and **password**. Accounts are managed by an administrator (chapter 9.2).

Access levels build on each other. Every level includes the permissions of the levels below:

Level	Password	May access
Basic	no	Scoreboard only (intended for wall displays)
Operator	yes	+ Production Timeline, live Bottleneck view, Notifications
Supervisor	yes	+ all Analysis views, AI Assistant, Line Config
Admin	yes	+ System configuration, Users, Reports, Backup, SMTP, Version, License

If you navigate to a page your role does not permit, you are redirected to the Scoreboard.

4.3 Navigating the user interface



The sidebar on the left is the main navigation:

- **Scoreboard**: large-format live overview (chapter 5.1)
- **Production**: Production Timeline with state controls (chapter 5.2)
- **Bottleneck**: live bottleneck analysis (chapter 5.3)
- **Notifications**: system and production notifications(chapter 4.5)
- **Analysis** (expandable): OEE History, Bottleneck History, Production Analysis, Downtime, AI Assistant (chapter 6)
- **Line Config** (expandable): Program, Product, Counters, Cycle Stations, Sensors, Takt, Downtime (chapter 7)
- **System** (expandable): System, Backup, Users, Reports (Templates / Blocks / Layout), SMTP, Version (chapters 8–9)

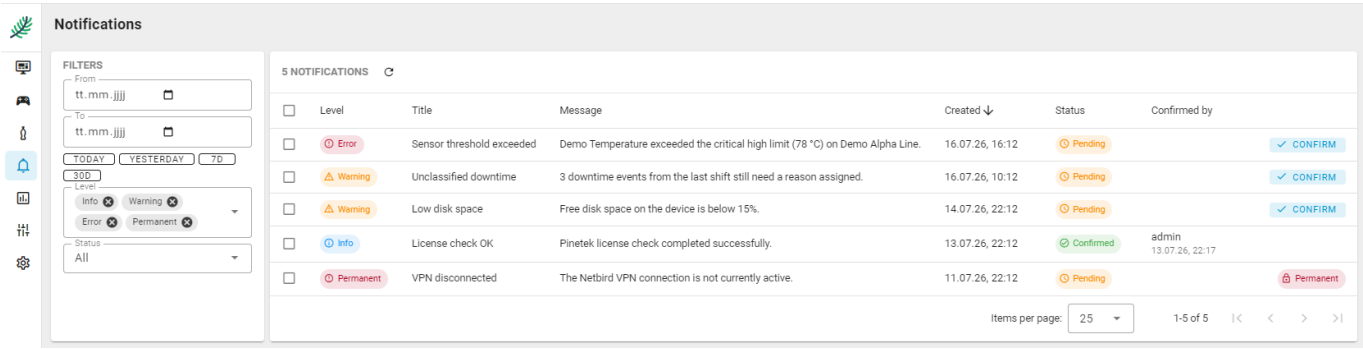
Menu entries are only shown if your role permits them.

4.4 Changing the display language

CedarInsight is available in **English** and **German**. The language is a system-wide setting; it applies to the entire web interface and to generated PDF reports.

An administrator changes it under *System* → *System* → *Language* (chapter 9.1).

4.5 Notifications



The system raises notifications for events that need attention, e.g. disk-space warnings or production events. Open them via **Notifications** in the sidebar.

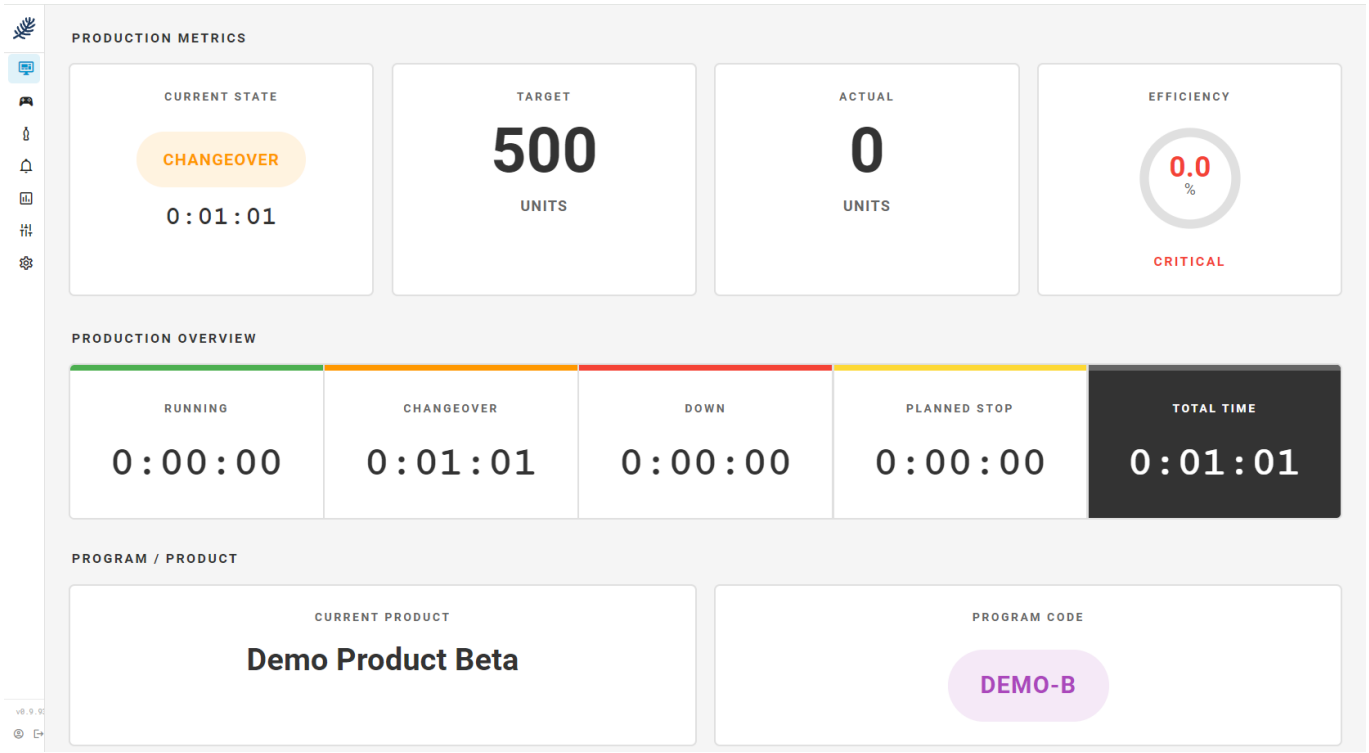
- Each notification has a **level**: *Info*, *Warning*, *Error* or *Permanent*.
- Filter by status: *All*, *Unconfirmed*, *Confirmed*, *Permanent*.
- Select one or more notifications and press **Confirm** to acknowledge them. An optional **note** can be stored with the confirmation (applied to all selected entries). The confirming user and time are recorded.
- *Permanent* notifications represent persistent conditions and stay visible while the condition lasts.

Unresolvable warnings (e.g. low disk space) should be reported to your administrator.

5. Live Operation

This chapter describes the day-to-day views used at the line. Required role: **Operator** (Scoreboard: any signed-in user).

5.1 Factory Scoreboard

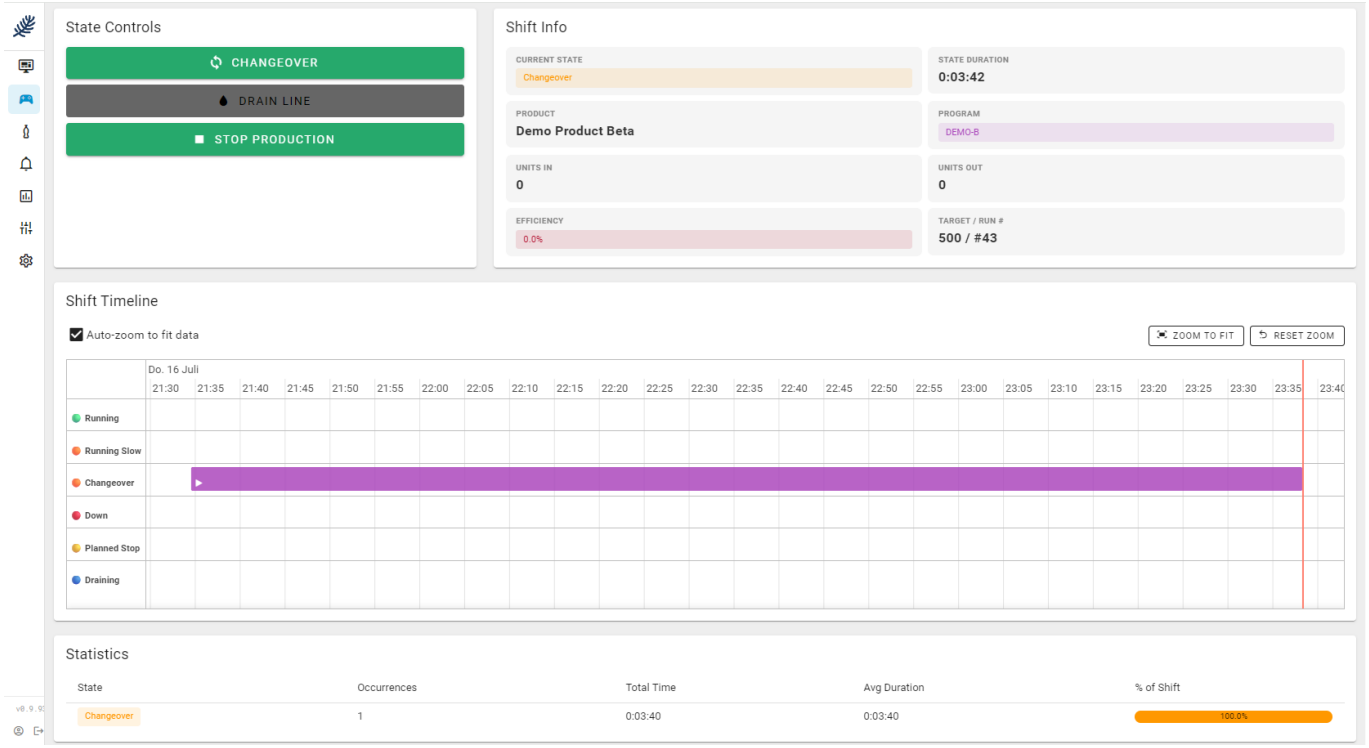


The Scoreboard is a large-format live display intended for wall monitors. It shows:

- **Current State** with state color and duration,
- **Current Product** and **Program Code**,
- Production Metrics: **Target, Actual, Efficiency**,
- Production Overview of the current shift: time in **Running, Changeover, Down, Planned Stop**, and **Total Time**.

The Scoreboard requires no interaction and refreshes automatically. A Basic account (no password) can be used to leave it permanently open on a wall display.

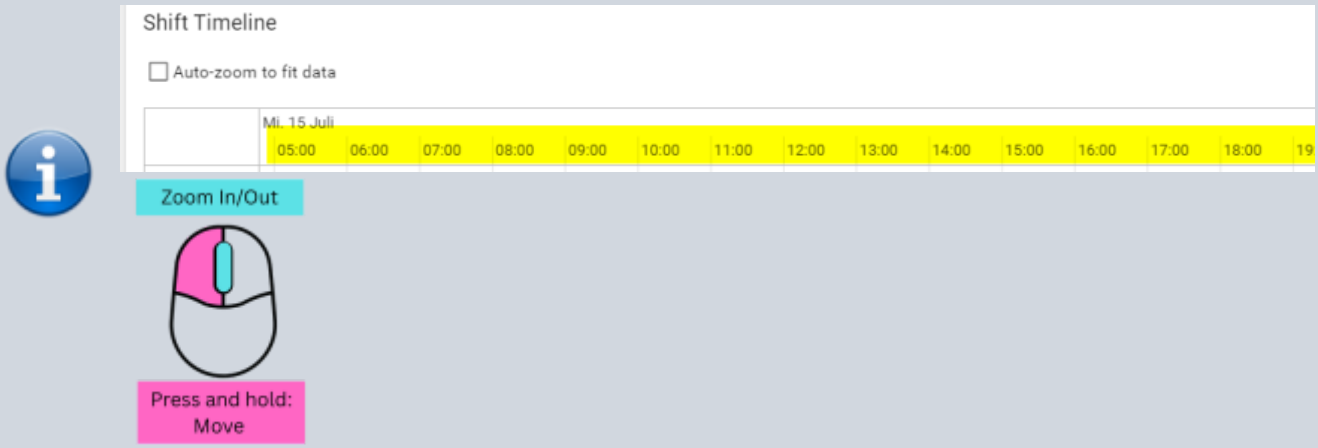
5.2 Production Timeline



The Production Timeline is the operator's main working view. It combines:

- **State Controls**: buttons to drive the production run (below),
- **Shift Info**: current state, state duration, program/product, Units In, Units Out, Efficiency, Target/Run #,
- **Shift Timeline**: a horizontal timeline of all state segments; use *Zoom to Fit* / *Reset Zoom* or enable *Auto-zoom to fit data*,
- **Statistics**: per state: occurrences, total time, average duration, % of shift,
- **Stage table**: per process stage: stage #, name, total units, throughput.

To zoom in and out with the mouse wheel, you need to put the mouse pointer over the timeline (yellow marked):



5.2.1 Starting a run (changeover)

1. Press **Start Changeover**. The line state changes to **Changeover** (purple).
2. Select the **Product** (mandatory): the program is determined by the product.
3. Enter the **Target Quantity** (leave blank for an unlimited run).
4. Confirm. The production run is created.

As soon as the line starts counting parts [PBX], the state changes automatically to **Running**; no further operator action is required.

5.2.2 During the run

The state is maintained automatically from the counter signals (chapter 2.3.2):

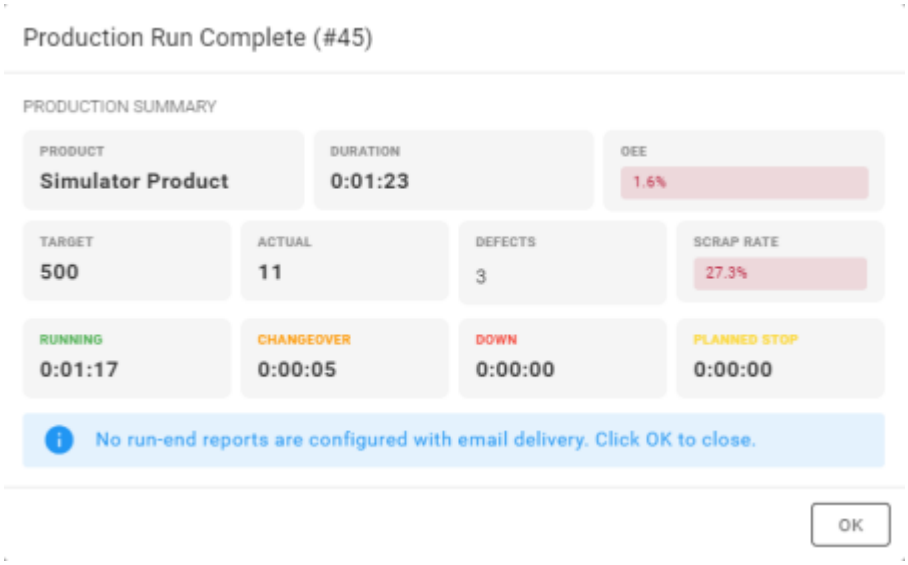
- If a stage counter stalls beyond its timeout, the state becomes **Down** and a downtime reason is requested (section 5.4).
- If throughput drops without a full stop (single stations stall), the state becomes **Running Slow**.
- For breaks or planned interventions, use **Planned Stop**; press **Resume** to continue. Planned stops do not reduce Availability.

5.2.3 Ending a run

There are two ways to end a run:

- **Drain Line**: the input stage stops while the rest of the line runs empty (state **Draining**, blue). When the output counter times out, the run ends automatically.
- **Stop Production**: ends the run immediately (confirmation required).

5.2.4 Run-end summary and bad parts



When the run ends, the **Production Run Complete** dialog shows the run summary: Target, Actual, Defects and Scrap Rate.

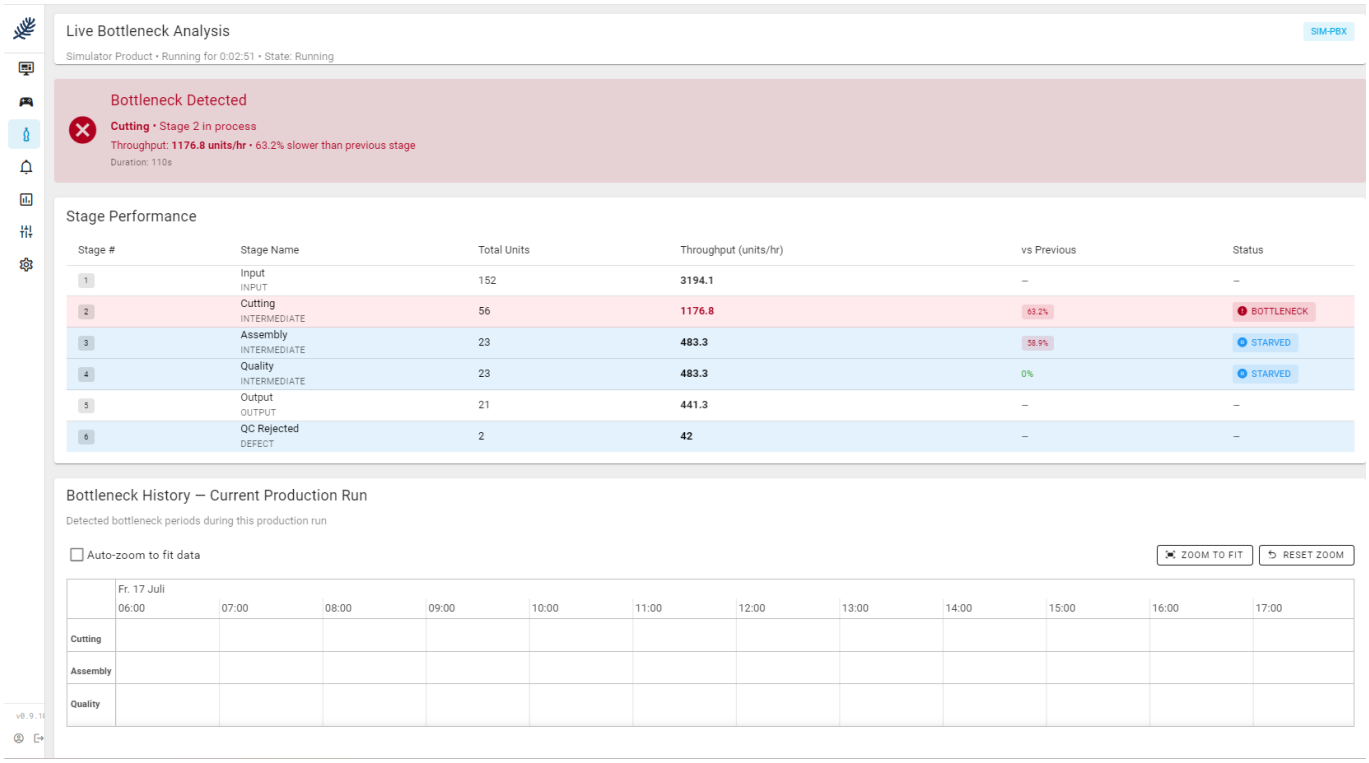
The defect parts are calculated as *input parts - output parts* after the line was drained or production was stopped.

Actions on run-end:

- Correct the defect quantity here if the automatic defect count does not match reality (e.g. manually sorted-out parts). The OEE Quality factor is recalculated with the corrected value.
- If run-end reports with e-mail delivery are configured (chapter 8), the dialog lists the reports and recipients. Choose **Send Email** to dispatch them, or **OK, do not send email** to skip.

The finished run is then available in all analysis views (chapter 6).

5.3 Live Bottleneck Analysis



The Bottleneck view compares the live throughput of all intermediate process stages of the current run and identifies the stage that limits the line.

How detection works: a stage is flagged as a bottleneck if its throughput is more than a configured percentage slower than the fastest stage, and this condition persists longer than a configured minimum duration. Both thresholds are set per program (chapter 7.1). Analysis begins after a warm-up of at least 60 seconds of running time.

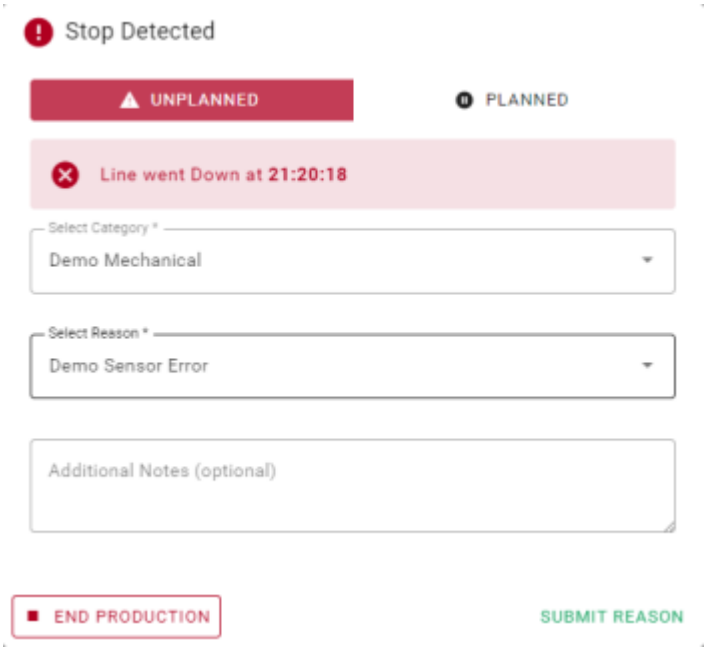
Each stage is shown with total units, throughput (units/h), deviation vs. the previous stage, and a status:

Status	Meaning
NORMAL	Stage operating within threshold
OBSERVING	Deviation detected; a countdown shows when it becomes a confirmed bottleneck if it persists
BOTTLENECK	Deviation persisted beyond the minimum duration; confirmed bottleneck
STARVED	Downstream of the bottleneck, waiting for upstream material

When a bottleneck is detected, all downstream stages are marked *STARVED*, since they cannot run faster than the bottleneck allows. Only the earliest slow stage is the true bottleneck. Input and output stages are shown for reference but are not analysed.

The lower part of the view lists the bottleneck periods detected during the current run. Historical analysis across runs is available under *Analysis → Bottleneck History* (chapter 6.2).

5.4 Recording downtime reasons



Every **Down** period must be justified. When a stop is detected, the timeline view shows a **Downtime Occurred** dialog (“Line went Down at …”):

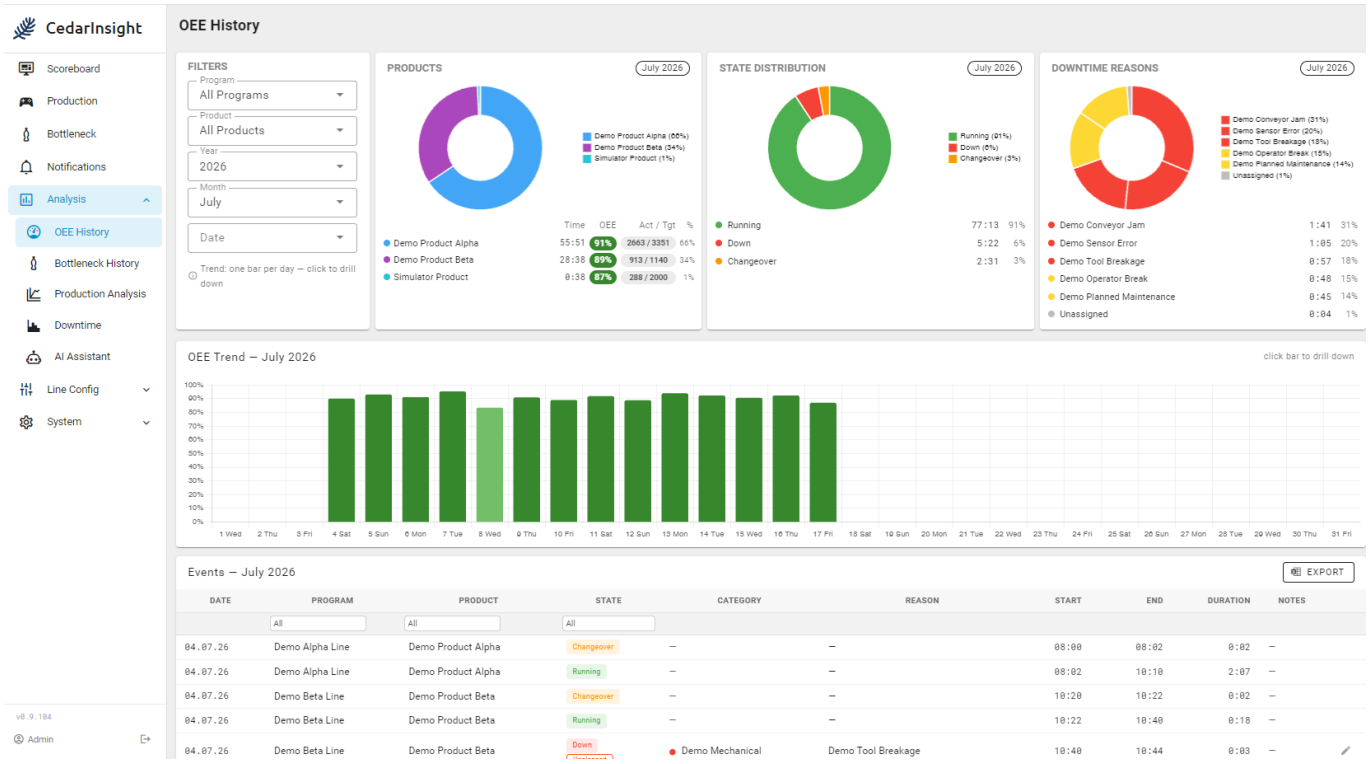
1. Choose whether the stop was **Planned** or **Unplanned**.
2. Select the **Category** and then the **Reason** (both mandatory; the catalogue is configured under *Line Config → Downtime*, chapter 7.7).
3. Optionally add **Notes**.
4. Press **Submit Reason**.

If a stop is not classified immediately, it remains **pending** and can be assigned or corrected later in *Analysis → OEE History* (chapter 6.1). Complete reason data is what makes the Downtime Pareto (chapter 6.4) meaningful.

6. Analysis

The Analysis views are available to **Supervisor** and **Admin** roles. All views share the same filter pattern: **Program**, **Product** and a time drill-down (year → month → day).

6.1 OEE History

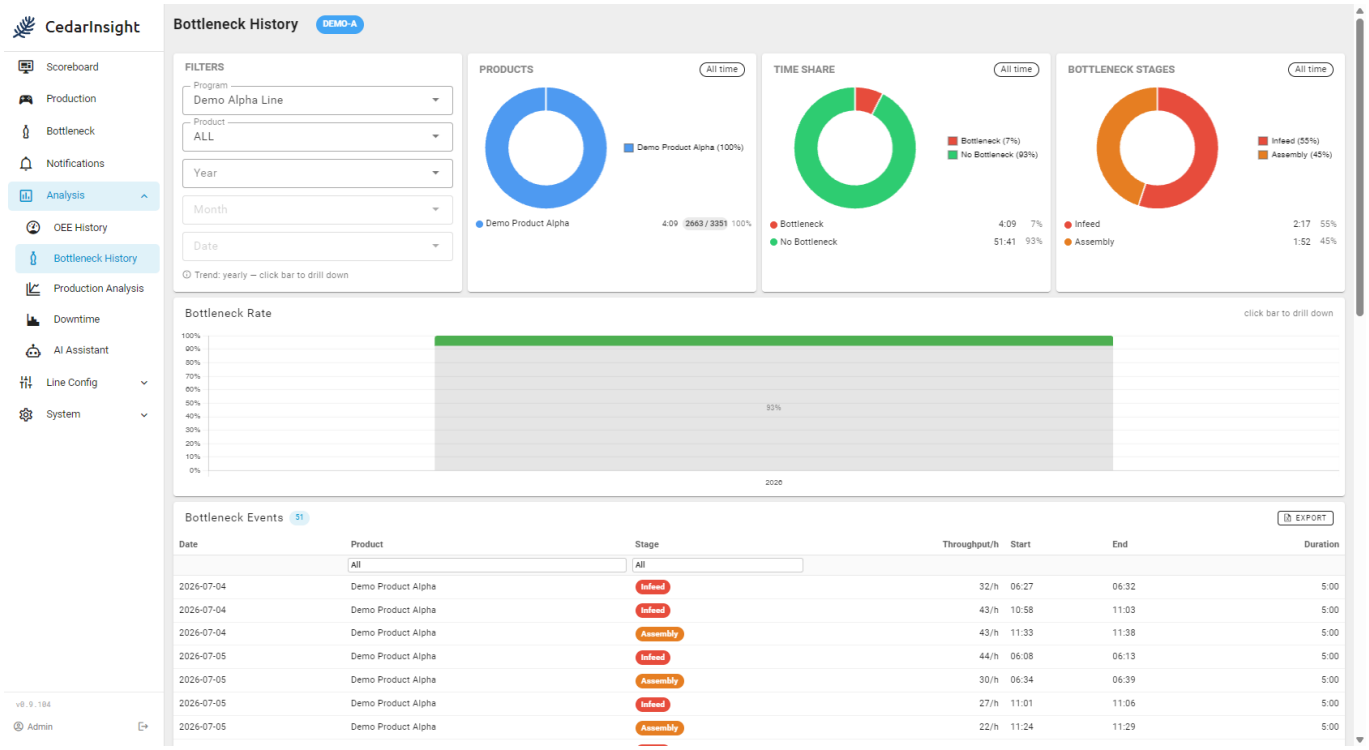


Analysis → OEE History shows the OEE development over time.

- **KPI cards**: OEE per product, states, downtime reasons for the selected period.
- **OEE Trend:** one bar per year, month or day; click a bar to drill down to the next level. At day level, an **Hourly OEE** table shows OEE, actual/target and events per hour; click a row to filter the pies and event list.
- **Events**: the downtime events of the selection, with duration, planned/unplanned flag, category, reason and notes.

Editing downtime reasons: in the events list you can **assign** a reason to a pending event or **edit** an already assigned one (pencil icon): select category, reason and optional notes. This is the place to clean up unclassified stops after the shift.

6.2 Bottleneck History

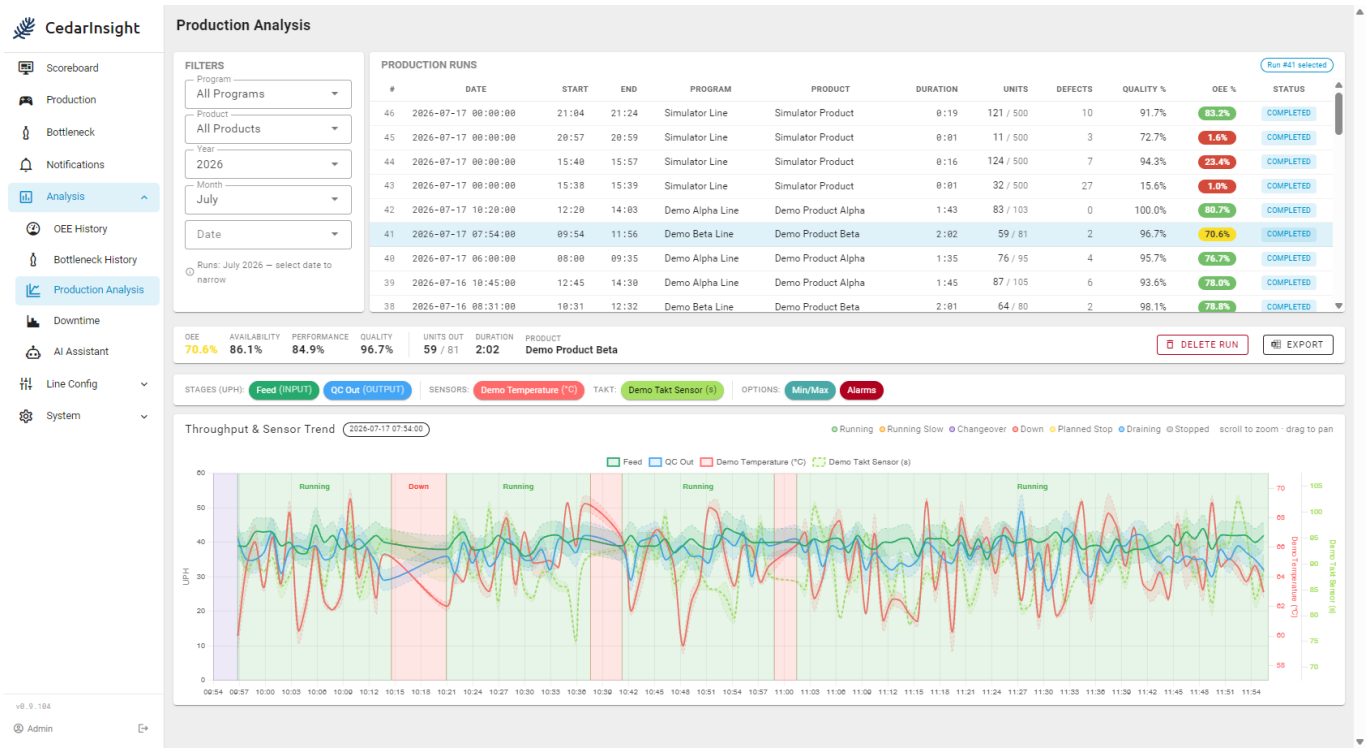


Analysis → Bottleneck History aggregates the bottleneck events (chapter 5.3) across runs.

1. Select a **Program** (mandatory as stages/stations are program-specific).
2. The trend chart shows bottleneck time per year/month/day; click a bar to drill down or, at day level, to filter the event list.
3. The **Bottleneck Events** table lists each event with date, product, stage, throughput/h, start, end and duration.

Use this view to find the chronically limiting stage of a line: the prime candidate for improvement measures.

6.3 Production Analysis



Analysis → Production examines individual production runs.

The graph data shows data for a production run: throughput of the single stations and the takt and sensor data connected with the program. The takt and sensor data is used to correlate events and data with the throughput, e.g. throughput goes down when temperature goes up.

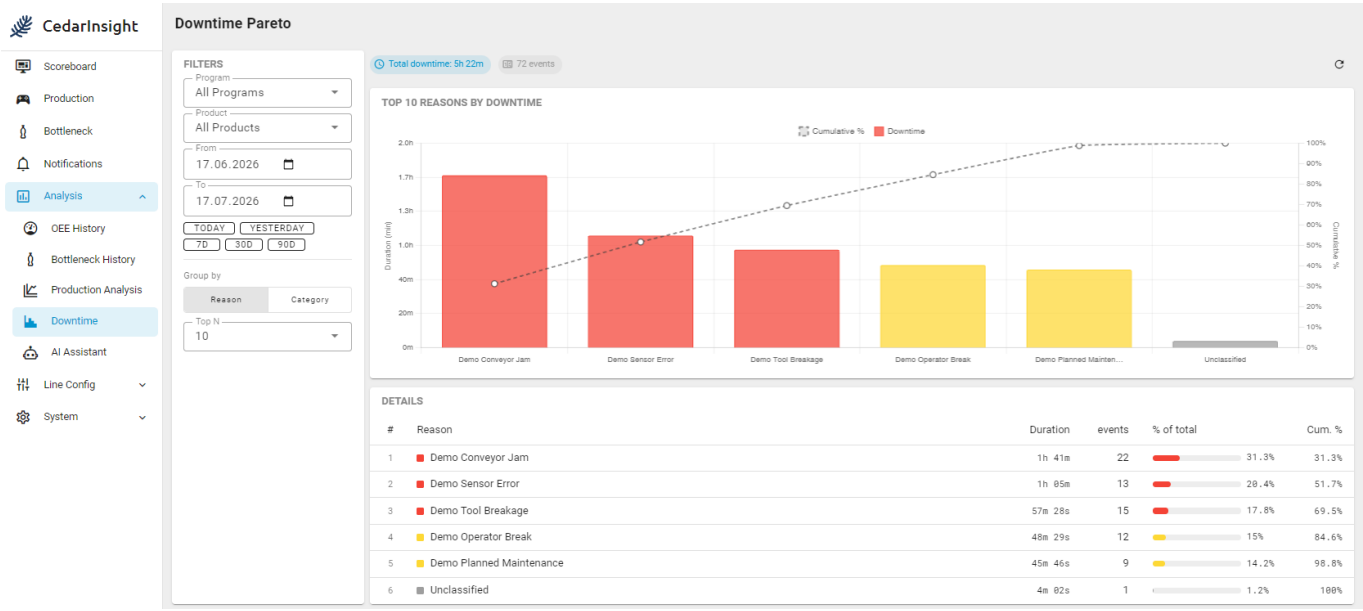
To analyze production runs, the following actions are available:

- Narrow the run list by year, month, date, program and product. The **Production Runs** table shows date, start/end, program, product, duration, units, defects, Quality % and OEE % per run, plus its status.
- Click a run to load its detail below:
  - **Production Summary:** the run's OEE figures,
  - **Throughput & Sensor Trend:** a time chart of the run. Select which **stages** (throughput) and **sensors** to plot; options include min/max bands and alarm markers, and a **takt** overlay. Scroll to zoom, drag to pan (on the timeline)

Run maintenance (use with care):

- **Close Run**: marks a still-ACTIVE run as COMPLETED and recalculates its OEE metrics. Use this when a run was left open (e.g. after a power failure).
- **Delete Run:** permanently deletes the run and all associated data. This cannot be undone.

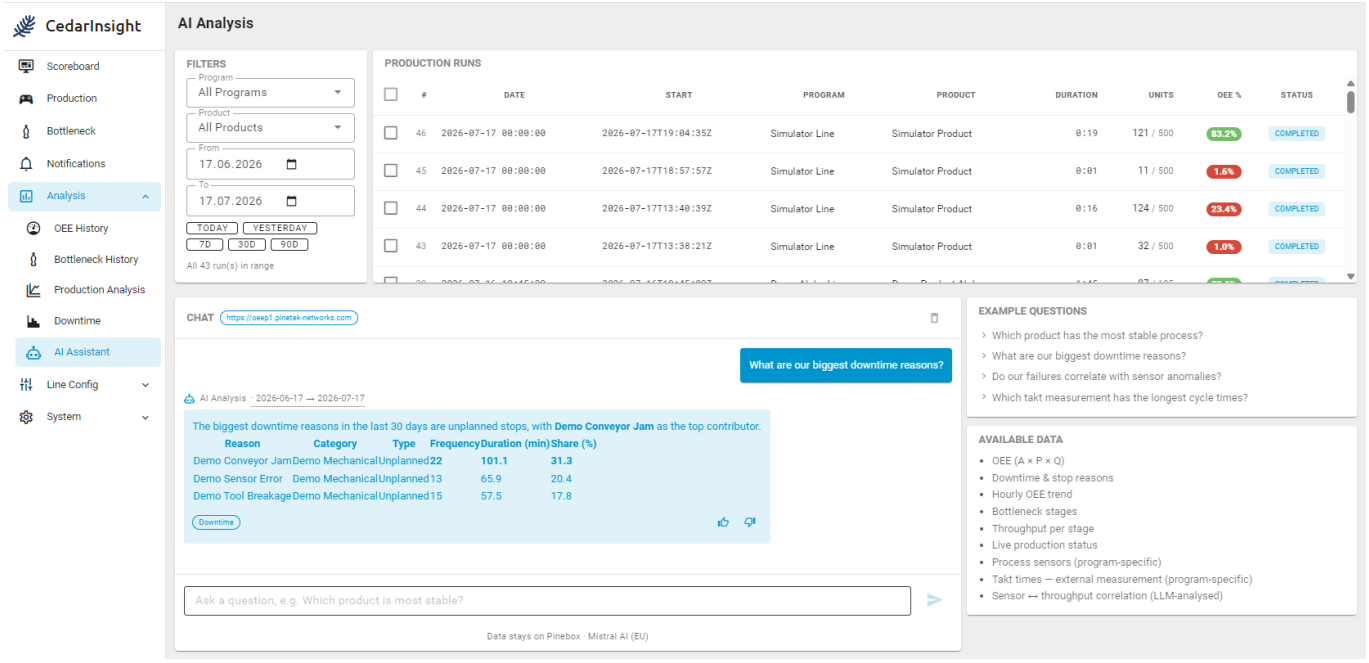
6.4 Downtime Pareto



Analysis → Downtime ranks downtime causes by total duration, the classic Pareto (“80 % of downtime comes from 20 % of causes”).

- Filters: date range, program, product.
- **Group by** *Category* or *Reason*; select **Top N**.
- The chart shows duration bars plus the **cumulative %** line; the header shows total downtime and number of events.
- The **Details** table lists rank, name, duration, number of events, % of total and cumulative %.

6.5 AI Assistant



Analysis → AI Assistant provides chat-based analysis of your production data: ask questions in natural language (“Which product had the worst quality last?”, “What were the main downtime causes?”) and receive answers computed from the database.

6.5.1 Working with the assistant

- Define the dataset first: date range, program, product, or a set of selected runs. The assistant answers only from this scope.
- The assistant uses the same aggregation logic as the analysis views and reports, so its figures match what you see elsewhere.

The AI (LLM) is weak handling date and time inputs like “today” or “last week”. To narrow down times, please use the filters. They will filter the raw data provided to the API.

6.5.2 Licensing, data privacy and anonymisation

- The AI Assistant is a licensed feature (entitlement *AI Chat*, chapter 9.6). Without it the view shows “AI Analysis not licensed”.
- Requests are processed via the Pinetek LLM gateway. Identifying names are anonymised before leaving the device and de-anonymised in the response; the external model never sees your real product or program names.

6.6 Exporting data

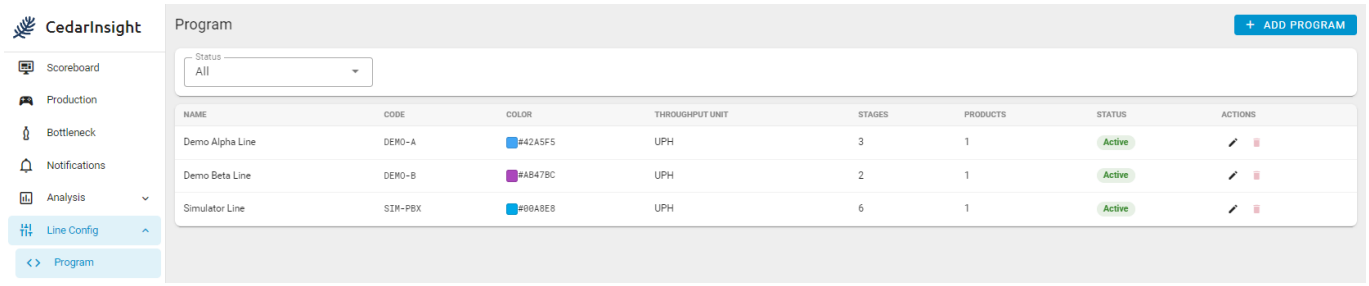
Analysis tables offer an **Export** action that downloads the current (filtered) data as an Excel file for further processing.

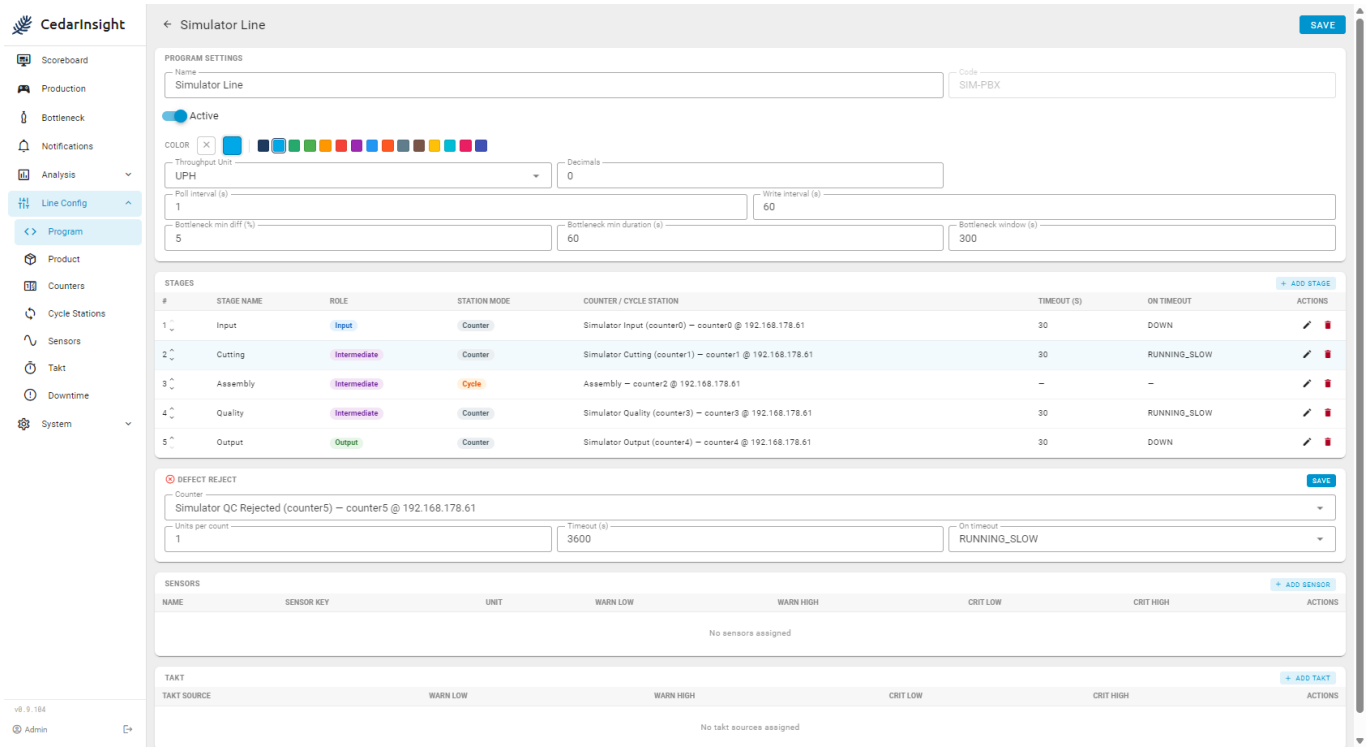
7. Line Configuration

Line Config describes your production line to CedarInsight. Required role: **Supervisor**.

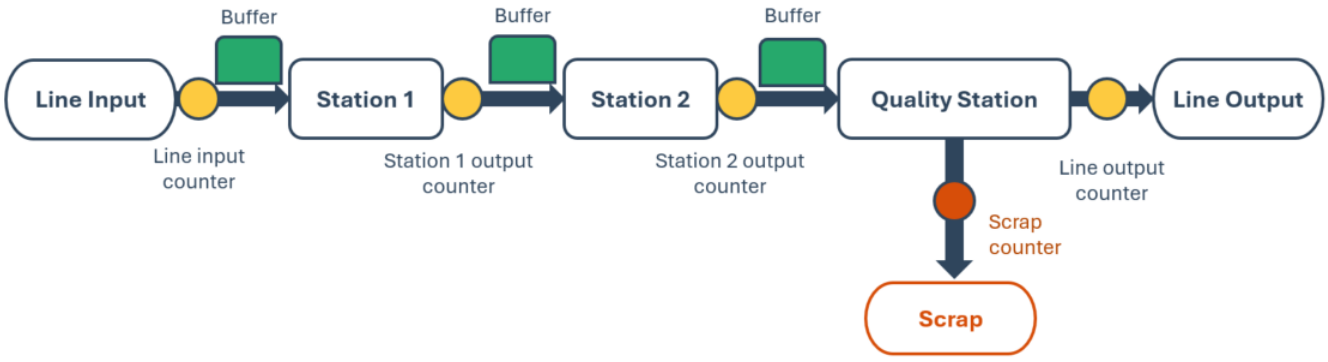
Recommended configuration order: *Counters → Program (with stages) → Products*, then optionally Sensors, Takt, Cycle Stations and the Downtime catalogue. See also the commissioning checklist in chapter 3.6.

7.1 Programs





Line Config → Program defines line configurations. Click a program to open its detail view with the tabs **Stages**, **Sensors**, **Takt**, **Products** and **Program Settings**.



Programs usually have stages that are connected to counters or cycle stations:

- An Input stage (defined by the first position in the list)
- Several processing stages with counting between the stages
- An Output stage (defined by the last position in the list)
- Optional: one or more scrap counters

The minimal configuration is one output counter. The counter inputs are used to calculate the OEE values.

The program can also include takt and sensors. Those inputs do not have impact on the OEE calculation, but can be used for correlating in the production analysis or via the AI analysis (see below) .

Program settings

Field	Meaning
Name / Code	Display name and short code (code is shown on the Scoreboard). The code cannot be changed after the first production run
Color	Default color for this program's runs
Cycle Time (s)	Default ideal cycle time (can be refined per product)
Throughput Unit	Units per hour / minute / day; affects throughput displays; the actual throughput is defined at the product
Decimals	Decimal places for throughput values
Poll interval (s)	How often counter values are sampled (default 1)
Write interval (s)	How often sampled readings are written to the database (default 60)
Bottleneck min diff (%)	A stage must be at least this much slower than the fastest stage to count as deviating
Bottleneck min duration (s)	The deviation must persist this long to become a confirmed bottleneck
Bottleneck window (s)	Averaging window for throughput comparison

Stages

Each stage is one counting point of the line, in process order:

Field	Meaning
No.	Position in the process (1 = first), automatically determined
Stage name	e.g. “Filler”, “Capper”, “Packer”
Role	*Input*, *Output*, *Defect / Reject* or *Intermediate*
Counter	The Pinebox counter or cycle station mapped to this stage (registered in 7.3) [PBX]
Units per count	Multiplier, e.g. 6 if one counter tick equals one pack of 6 units
Timeout (s)	No count for this long → stage considered stalled (drives Down detection, chapter 2.3.2)
On timeout	Behaviour when the timeout fires

Roles matter for OEE: the **Input** stage counts *Units In* (actual quantity), the **Output** stage counts *Units Out* (good quantity), a **Defect/Reject** stage counts rejected parts, and **Intermediate** stages are used for throughput and bottleneck analysis.

A stage can alternatively run in **Cycle mode** instead of Counter mode; it is then fed by a cycle station (section 7.4) rather than a piece counter.

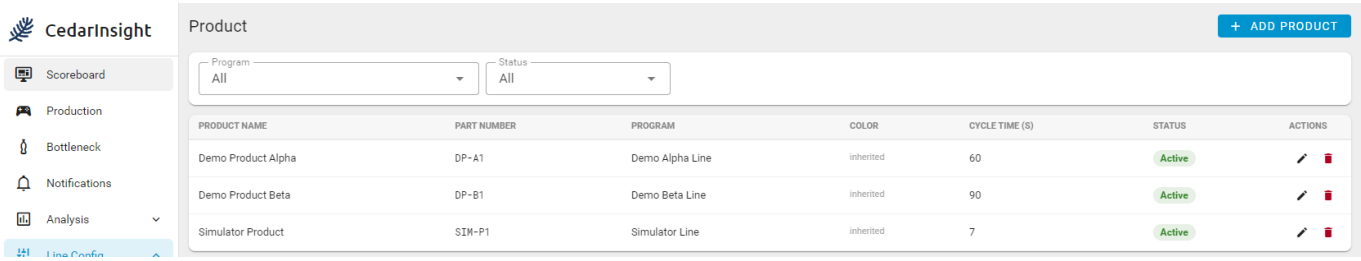
Sensors / Takt tabs

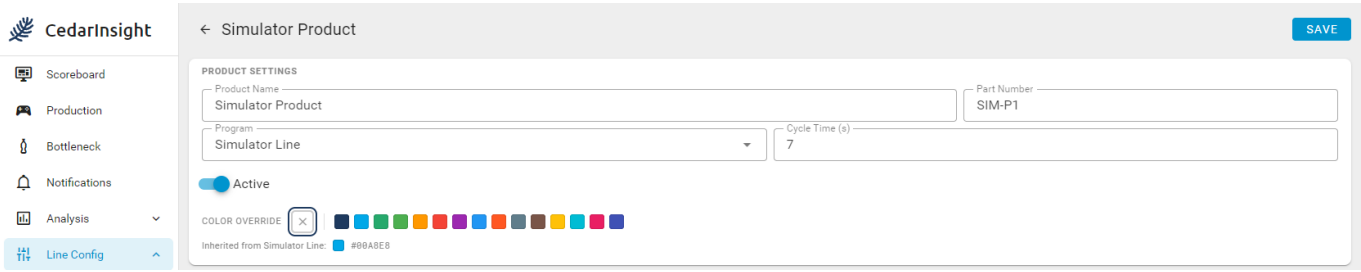
Assign registered sensors (7.5) and takt sources (7.6) to the program and set warning/critical thresholds (*Warn low/high*, *Crit low/high*). Assigned sensors and takt appear in the run trend chart (chapter 6.3). The recorded data is used to correlate takt times and sensor values with the throughput of the line, e.g. if air pressure drops, the takt times get longer.

The sensor data and takt time is recorded but does not contribute to the OEE calculation.

7.2 Products

Line Config → Product manages the articles you produce. A product runs on a defined program, but with it's own cycle time. Example: Filling lines that use the same stations for filling and labelling, but fill different container sizes (this different speeds and cycle times).





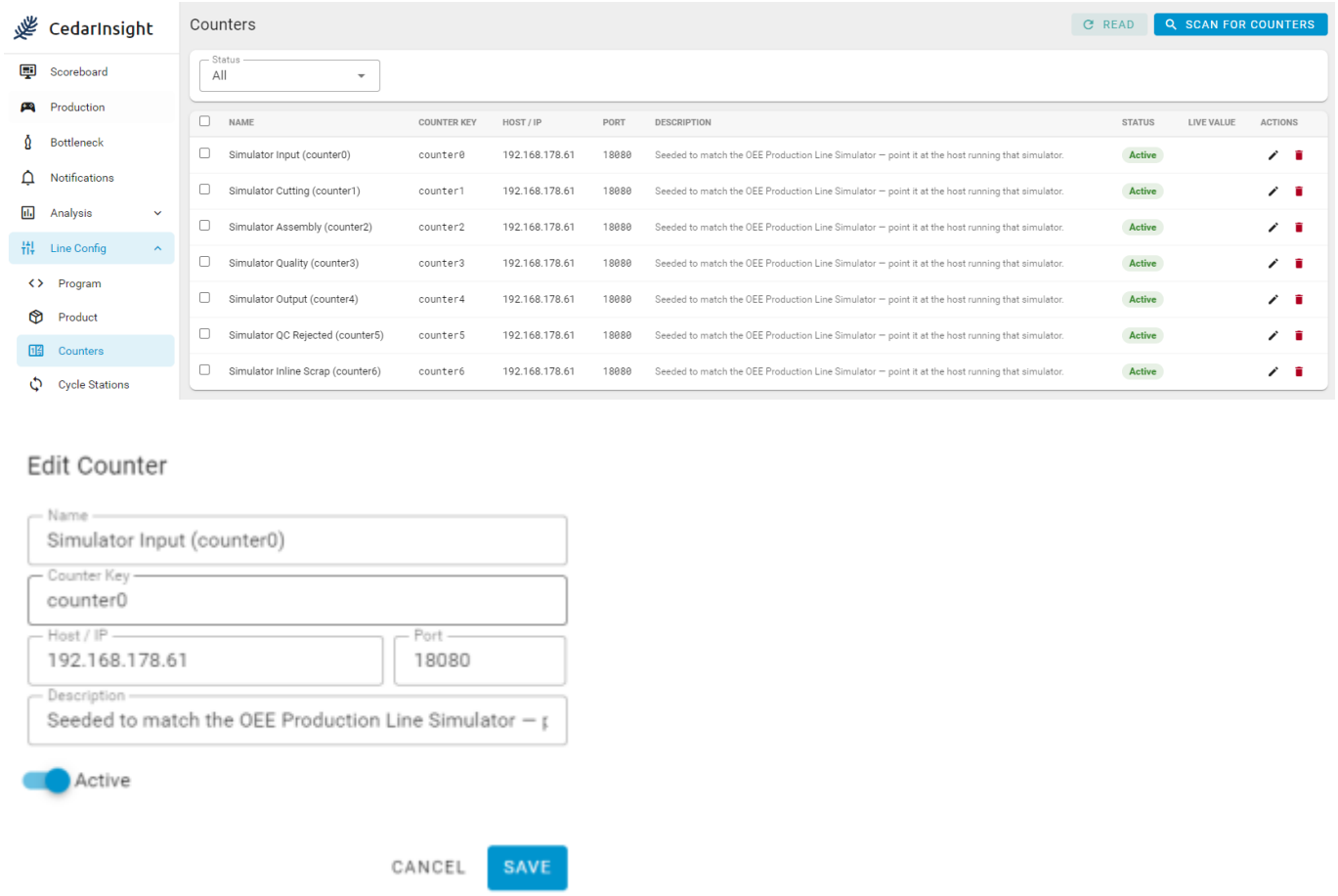
Per product:

Field	Meaning
Product Name	Display name (mandatory)
Part Number	Your article/part number
Program	The program used to produce this product (mandatory)
Cycle Time (s)	*Ideal* cycle time per unit, the basis of the Performance calculation (chapter 10.3)
Color Override	Optional colour for timelines/charts; by default the product inherits the program colour

An accurate ideal cycle time is essential; a value set too high makes Performance look better than it is, and vice versa.

7.3 Counters

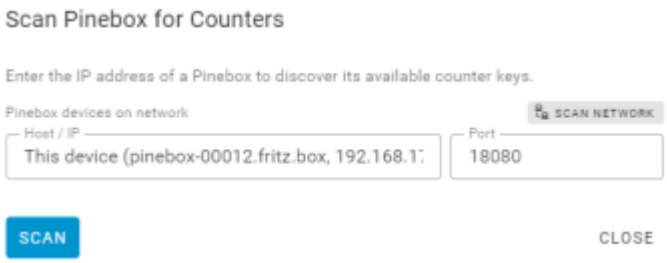
Line Config → Counters is the registry of Pinebox piece counters.The counter values are used as input to the programs. In order to use counters, they need to be defined and configured in the Pinebox.



Each entry stores:

Field	Meaning
Name	Display name
Counter Key	The key exactly as it appears in the Pinebox counter data [PBX]
Host / IP, Port	Address of the Pinebox providing this counter (default port `18080`)

Rather than typing keys manually, use the scan functions:



1. **Scan for counters:** reads the available counter keys from a selected (or manually entered) host.
2. **Scan network:** discovers Pinebox devices on the local network for remote counters
3. **Import** the keys you need; already registered keys are marked.

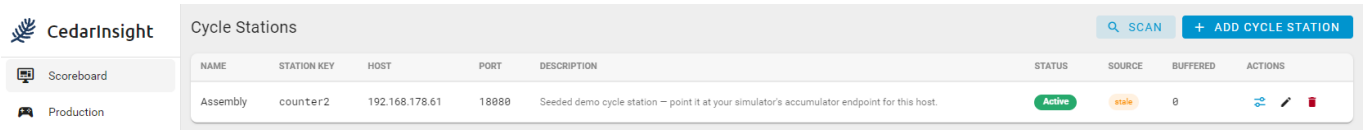
Read fetches the live values of all registered counters; use this to verify the mapping (unreachable hosts are reported).

Which physical input feeds which counter key is configured on the Pinebox, see the Pinebox Manual [PBX].
The default port on the Pinebox is 18080

7.4 Cycle Stations [PBX]

Cycle stations monitor machines whose cycles are detected from their power consumption (e.g. presses, moulding machines): the Pinebox measures electrical power/energy, and a detector recognises each machine cycle from the power rise at cycle start and the power fall at cycle end.

In the list of cycle stations, the detected buffers are shown and can be updated



7.4.1 Registering cycle stations

Line Config → Cycle Stations manages the registry (pen symbol). Cycle stations serve as input to the program counters. To use cycle stations, they need to be configured in the Pinebox.

Per station:

Parameter	Meaning
Name	Station name for identification
Station Key	Identifier on the source device (Pinebox)

Parameter	Meaning
Host/Port	Pinebox IP or hostname and port
Description	Internal description of the cycle station
On stall	*see below*

As with counters, **Scan** discovers devices on the network and imports the stations they advertise.

On stall defines what happens when the detector flags a stalled cycle:

- **Auto-scrap:** the part is automatically counted as scrap,
- **Ask operator:** a resolution prompt appears; the operator chooses *Scrap*, *Complete* or *Re-enter*,
- **Auto-continue:** the partial cycle is accepted as complete.

7.4.2 Calibration

The **Calibrate** view shows a live power scope (~2 min window; scroll to zoom, drag to pan) with the average power and its rate of change, plus the currently configured thresholds and the observed idle/peak values.

Detector parameters:

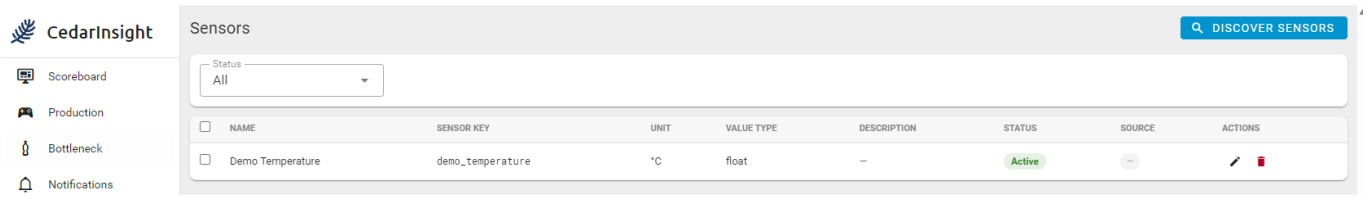
Parameter	Meaning
Baseline (W)	Machine idle power draw, used for stall detection
Start rise (W/s)	Minimum rate of power rise to begin a cycle (right scale)
End fall (W/s)	Minimum rate of power fall to end a cycle (right scale)
Min energy (kWh)	Minimum accumulated energy before the end condition may fire (noise guard)
Stall timeout (s)	Average power below baseline for this long → cycle flagged as stalled
Slow timeout (s) / Max cycle (s)	Limits for slow/overlong cycles

Parameter	Meaning
T min / T max (s)	Optional duration gate , cycles outside these bounds are not accepted as OK

Calibration procedure:

1. Let the machine run a few normal cycles.
2. Watch the **rate trace** (right axis, W/s): set **Start rise** to the leading-edge peak and **End fall** to the trailing-edge dip of a cycle.
3. Set **Baseline** to the observed idle power.
4. Use the energy measurement to determine **Min energy**: click once on the curve to set the start marker, click again for the end marker; the energy under the curve and the time span are displayed. Set **Min energy** safely below a normal cycle's energy but above noise level.
5. Save and verify that the **Detected Cycles** markers match the real machine cycles.

7.5 Sensors [PBX]



How sensor data reaches CedarInsight

Sensors are the one data type in CedarInsight that is pushed, not polled; everything else (counters, cycle stations, takt) is fetched by CedarInsight itself on a timer, and sensors instead arrive whenever the source decides to send them. The path is:

1. **Pinebox**: the IO-Link sensor data is configured and poweredread at the hardware/fieldbus level. [PBX]
2. **Node-RED**: a flow running on the Pinebox (local or remote) cyclically (via cyclic inject) reads that value as described in the Pinebox manual, packages it into the JSON shape CedarInsight expects, and sends it on.
3. **CedarInsight**: serves the receiving endpoint and stores what arrives. Nothing needs to be configured on the CedarInsight side to make it “listen”; the API is always up as part of the normal backend, and the only setup is registering the sensor here in **Line Config → Sensors** so incoming keys are recognised.

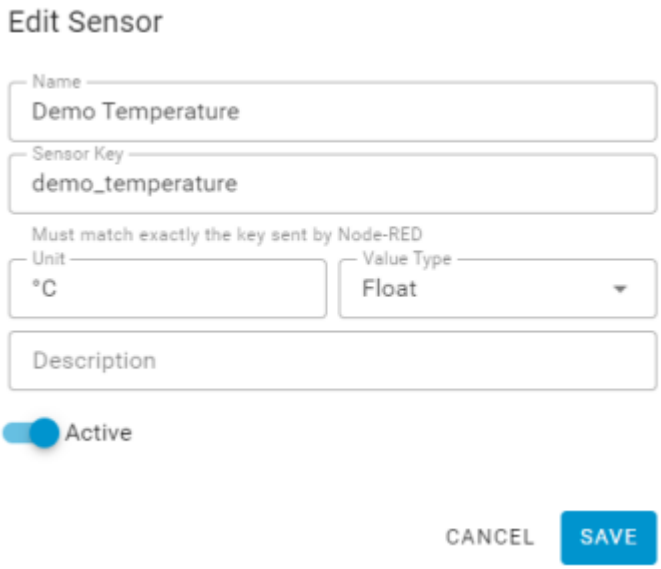
```
POST http://<cedarinsight-host>:8000/api/sensors/ingest
```

sent as a single sample object or an array of them, e.g.

```
''[{"sensor_key": "line1_temp", "value": 74.2, "unit": "°C"}]''
```

The full payload contract, field-by-field, and a worked Node-RED flow example (input node → function node → HTTP request node) are in chapter 11.6 [PBX]. This section covers the registry side: what to configure here once data is flowing.

Registering a sensor



Per sensor: **Name**, **Sensor Key** (must match the `sensor_key` the source sends, exactly; this is the only link between the incoming push and the registered row), **Unit**, **Value Type**, description.

Use **Discover sensors** to list keys currently sending live data (i.e. anything that has POSTed to the ingest endpoint recently, whether registered yet or not) and **Register** them directly, including their current value, so you can identify each signal before naming it.

If a not-yet-registered key's push includes `display_name`/`unit`, the sensor is auto-registered on first arrival; Discover will simply show it as already present. Once a sensor is registered and named through the UI, pushed `display_name`/`unit` values never overwrite what's set here, the UI always wins.

Checking that data is actually arriving

The **Source** column in the sensor list reflects whether a sample for that key has arrived recently (within the last 30 seconds):

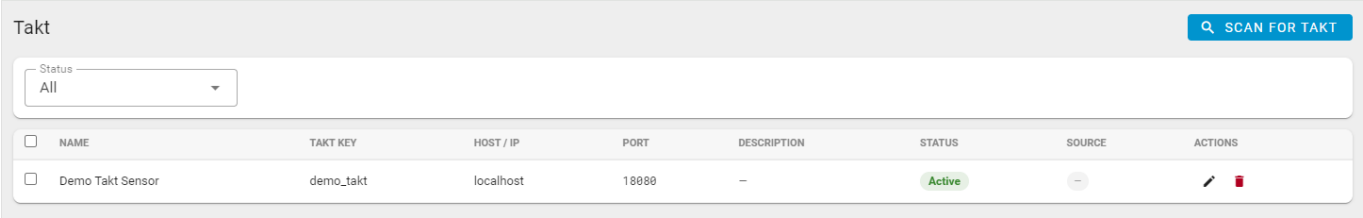
- **OK** (green): a sample was received within the last 30 seconds.
- **stale** (amber): no sample for longer than that; hover the chip for the last-seen time. Registered sensors keep their configuration regardless; this only reflects whether the push side is currently alive.
- **—** (grey): no sample has ever been received for this key since the backend last started.

This is independent of whether a production run is active; Node-RED can push (and this column will show **OK**) even with the line stopped.

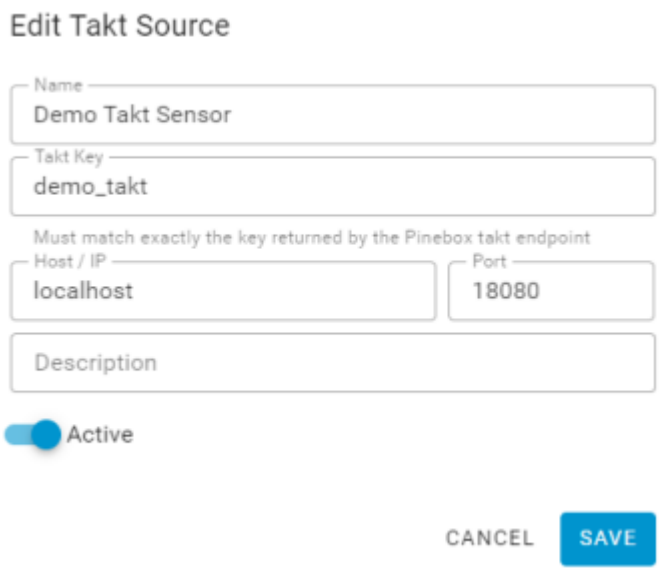
7.6 Takt

Takt is the cycle time of one individual unit (e.g. robot) how long that one process took, timed directly by the Pinebox, not derived from a counter. Takt is the time measurement between two events, e.g. a movement from A-to-B with a sensor on both ends.

Takt is complementary information, not a driver of OEE: unlike Counters and Cycle Stations, takt values never feed into Availability, Performance, or state transitions; like sensors they exist purely to catch a single slow (or suspiciously fast) cycle in the moment, via the **warn/crit thresholds** set on the program assignment (7.1), which raise notifications when breached and get overlaid against actual values in the run trend chart (6.3) for root-causing dips seen in the “real” OEE numbers.



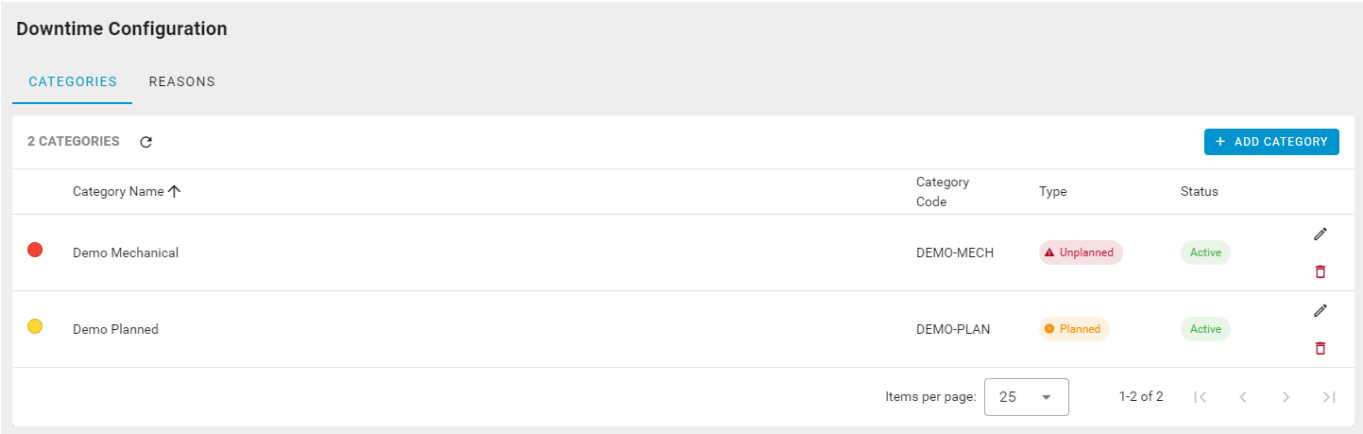
Line Config → *Takt* registers takt (cycle-time) sources measured by Pinebox timers. Per source: **Name**, **Takt Key** (must match the key of the Pinebox takt endpoint exactly [PBX]), **Host/Port**, description.



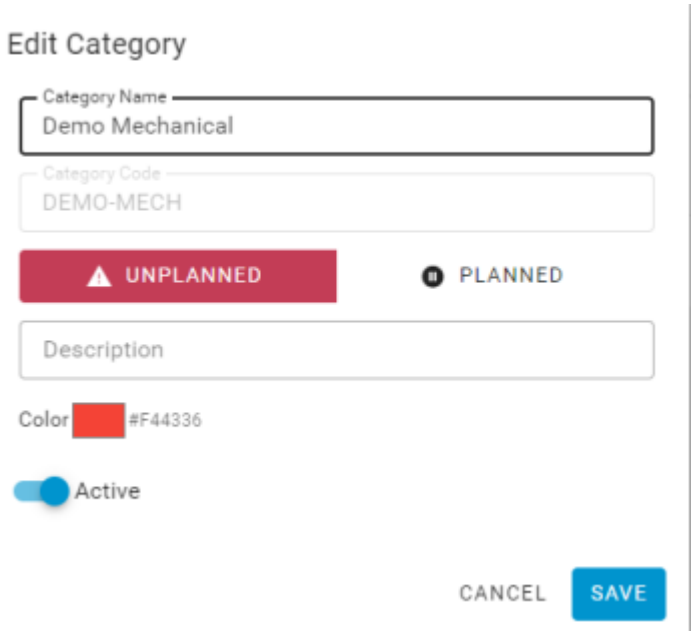
Scan for takt reads the available timer keys from a Pinebox (local or remote) and registers them directly.

Assign takt sources to a program (7.1) to monitor cycle times against thresholds and overlay them in the run trend chart.

7.7 Downtime reason catalogue



Line Config → *Downtime* maintains the two-level reason catalogue used when classifying stops (chapter 5.4): **Categories** (e.g. “Mechanical”, “Material”, “Organisational”) each containing **Reasons**.



Field	Meaning
Category Name	Display name of category/reason
Category Code	Unique short code, e.g. `MECH-FAIL`, cannot be changed after creation
Type	Planned / unplanned classification
Color	Color used in charts
Sort Order	Display order in the selection dialog

Keep the catalogue short and unambiguous; operators must find the right reason in seconds. Prefer ~5–10 reasons per category over an exhaustive list.

8. Reporting

CedarInsight generates PDF reports from your production data, automatically at run end, on a schedule, or on demand, and can send them by e-mail. Report management requires the **Admin** role and is found under *System → Reports*.

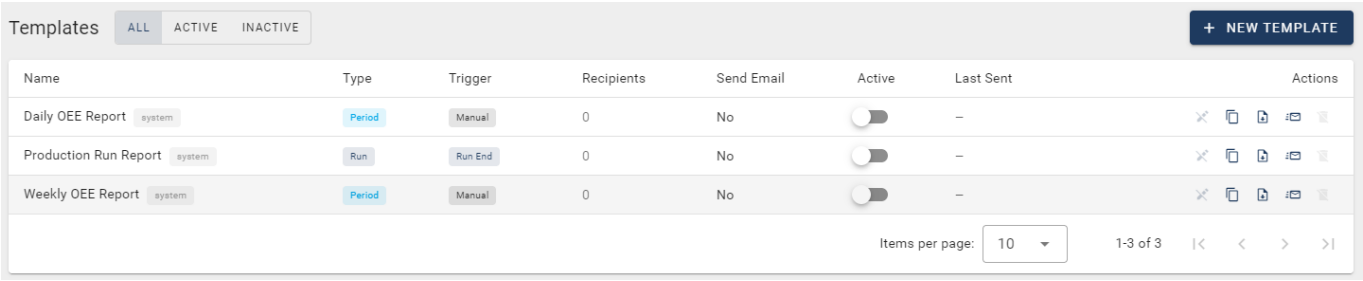
E-mail dispatch requires configured SMTP settings (chapter 9.4). Reports are generated in the system language (chapter 9.1).

8.1 Concepts

Element	Purpose
Template	A report definition: type, trigger, filters, recipients and an ordered list of sections
Block	A reusable content element (HTML). Types: *Header*, *Footer*, *Section*
Common Layout	The shared header/footer used by all templates unless overridden

A template is assembled from blocks; generated PDFs are archived per template.

8.2 Report templates

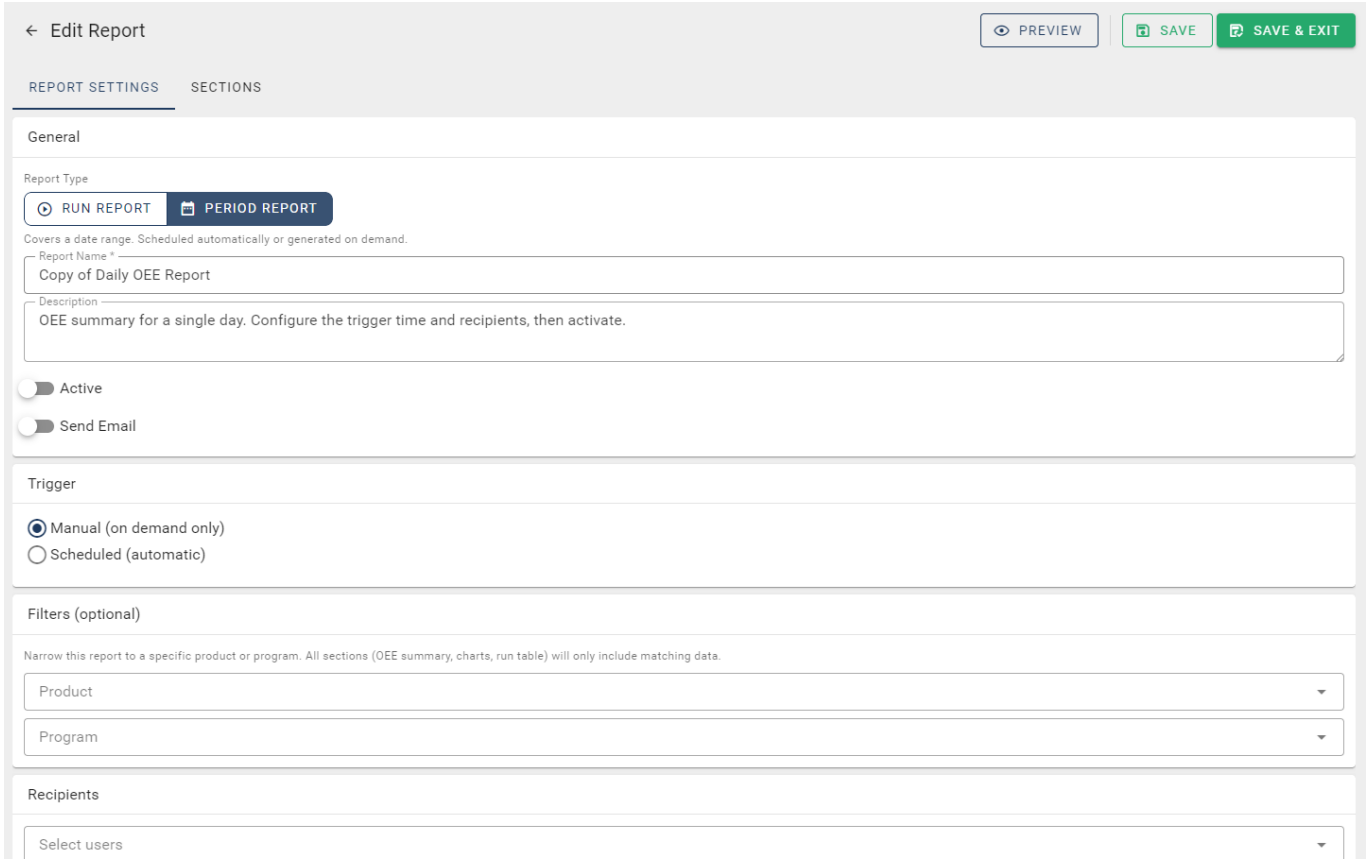


System → Reports → Templates lists all templates with type, trigger, recipients, e-mail flag, active flag and last-sent time. From the list you can **Generate / Download PDF**, send a **test e-mail**, edit, duplicate, activate/deactivate or delete a template (deleting also removes its archived PDF files). Templates marked **system** are predefined and cannot be edited or deleted.

There are two report types:

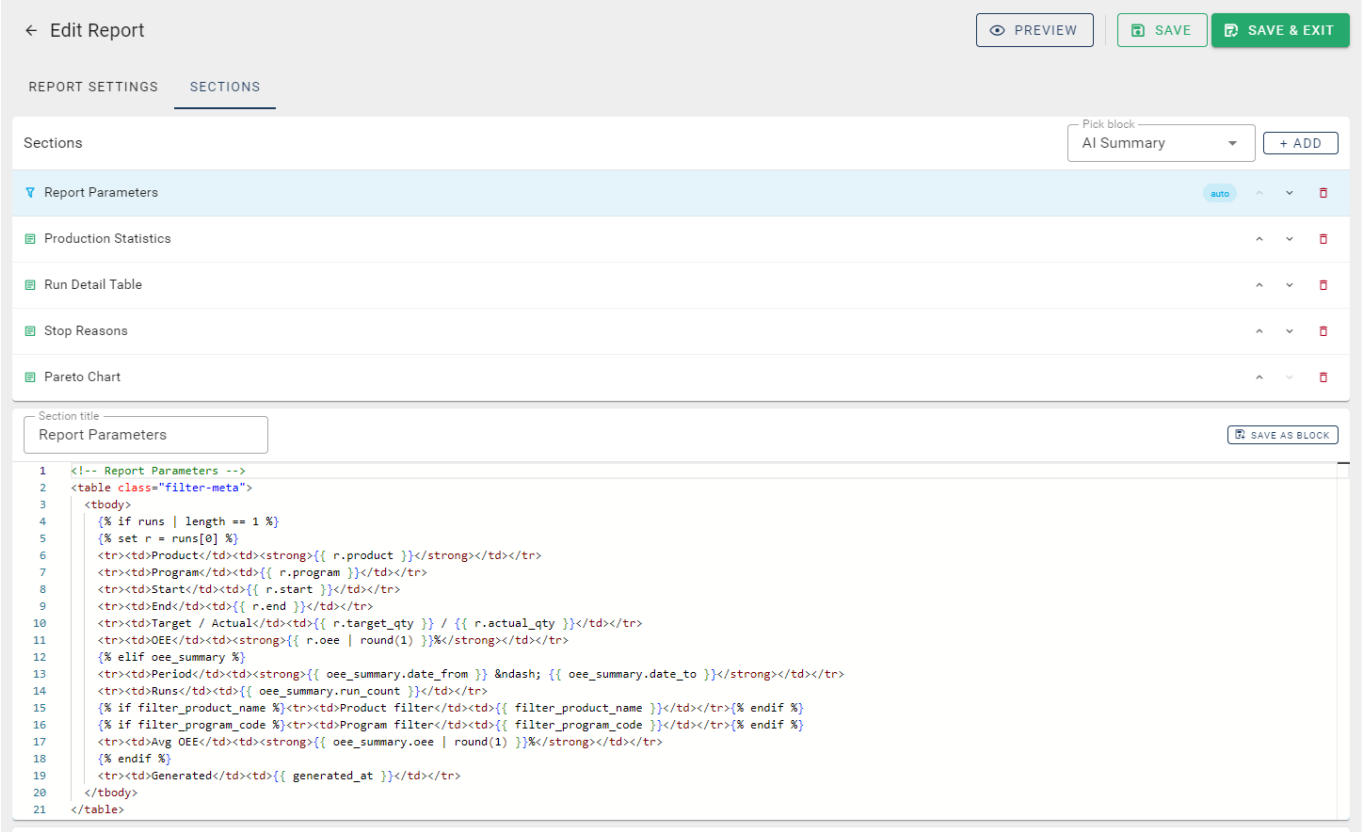
- **Run Report:** covers a single production run. Generated at run end or on demand (you pick the run; defaults to the most recent completed run).
- **Period Report:** covers a date range. Generated on schedule or on demand (you pick the range).

8.3 Creating and editing a template (Report Builder)

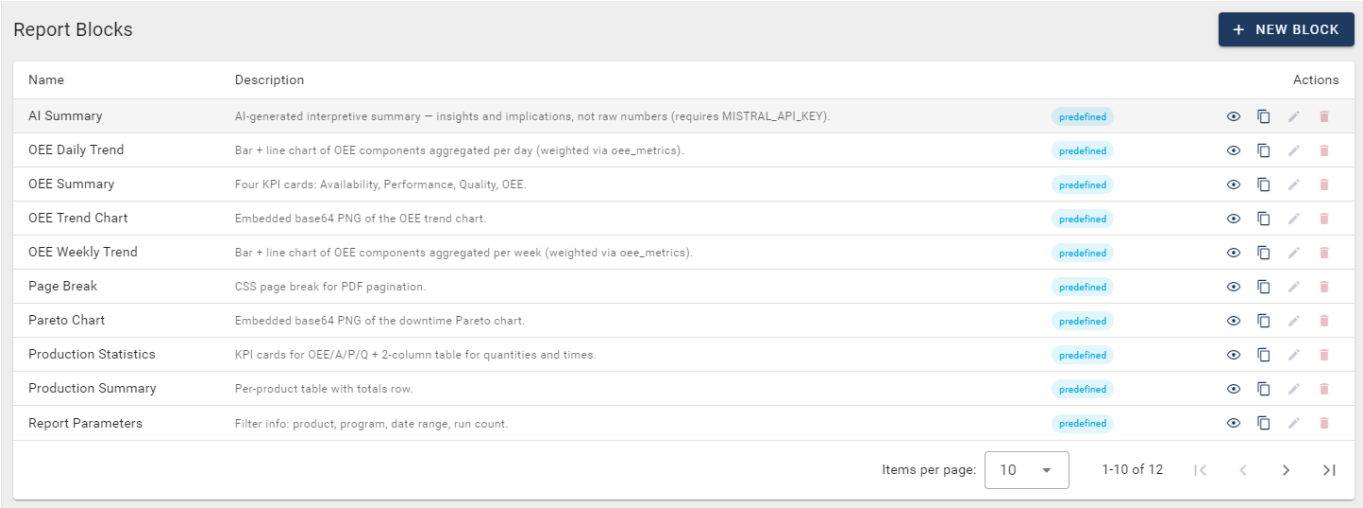


New Report opens the Report Builder:

- **General:** name, description, report type, active flag.
- **Trigger:**
  - **Manual **(on demand only), * **Triggered at Run End:** generated automatically when a production run is stopped (run reports), or
  - **Scheduled (automatic):** for period reports, choose the report period (rolling window: daily / weekly 7 days / monthly 30 days / quarterly 90 days), the run-at time (HH:MM) and, for weekly/monthly schedules, the day of week / day of month.
- **Filters (optional):** narrow the report to a specific product or program; all sections (OEE summary, charts, run table) then include only matching data.
- **Recipients:** select users; enable Send Email to dispatch the PDF automatically.
- Header & Footer: by default the shared Common Layout is used (). Enable Custom header/footer to override for this template only; you can load the common layout as a starting point.
- **Sections:** pick a block from the library and Add it; reorder with Move up / Move down. Click a section to edit its title and HTML content directly in the built-in editor; Save as Block stores an edited section back to the library for reuse.
- Generate Preview: renders the report with chosen parameters (run or date range) before saving.



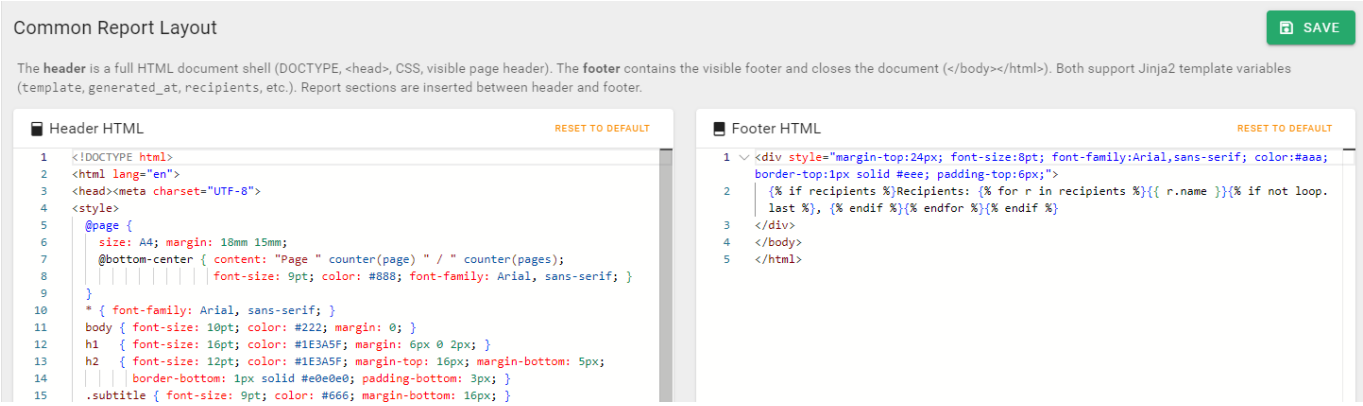
8.4 Blocks library



System → *Reports* → *Blocks* manages the reusable content elements. Each block has a name, description, type (*Header*, *Footer*, *Section*) and HTML content with Jinja2 template variables (8.6).

- **Predefined blocks** (OEE summary, run table, downtime table, charts, standard header/footer, ...) are system-owned: they can be viewed and duplicated, but not edited or deleted. To customise one, duplicate it and edit the copy.
- **Custom blocks** can be created, edited, previewed, duplicated and deleted freely.

8.5 Common Layout



System → Reports → Layout defines the shared header and footer HTML used by all templates that do not override it. **Edit**, **Preview**, or **Reset to default**. A table of the available template variables is shown in the editor.

Typical use: put your company logo and report title in the common header once; every report uses it automatically.

8.6 Template variables

Block HTML is rendered with Jinja2. The most important context variables:

Variable	Content
`runs`	List of the production runs in scope, with per-run OEE fields

Variable	Content
`runs_by_product` / `runs_by_program`	The same data grouped and aggregated per product / program
`oee_summary`	Weighted OEE totals for the whole scope (availability, performance, quality, OEE, quantities)
`reasons`	Downtime reasons with durations
`charts`	Pre-rendered charts (embedded images)
`generated_at`	Generation timestamp
`filter_*`	The active filters (period, product, program)
`t`	Translation dictionary: write `{{ t.oee }}`, `{{ t.availability }}` etc. so one block renders correctly in any report language

The full, current variable list is displayed inside the Report Builder and the Layout editor (**Available Template Variables**).

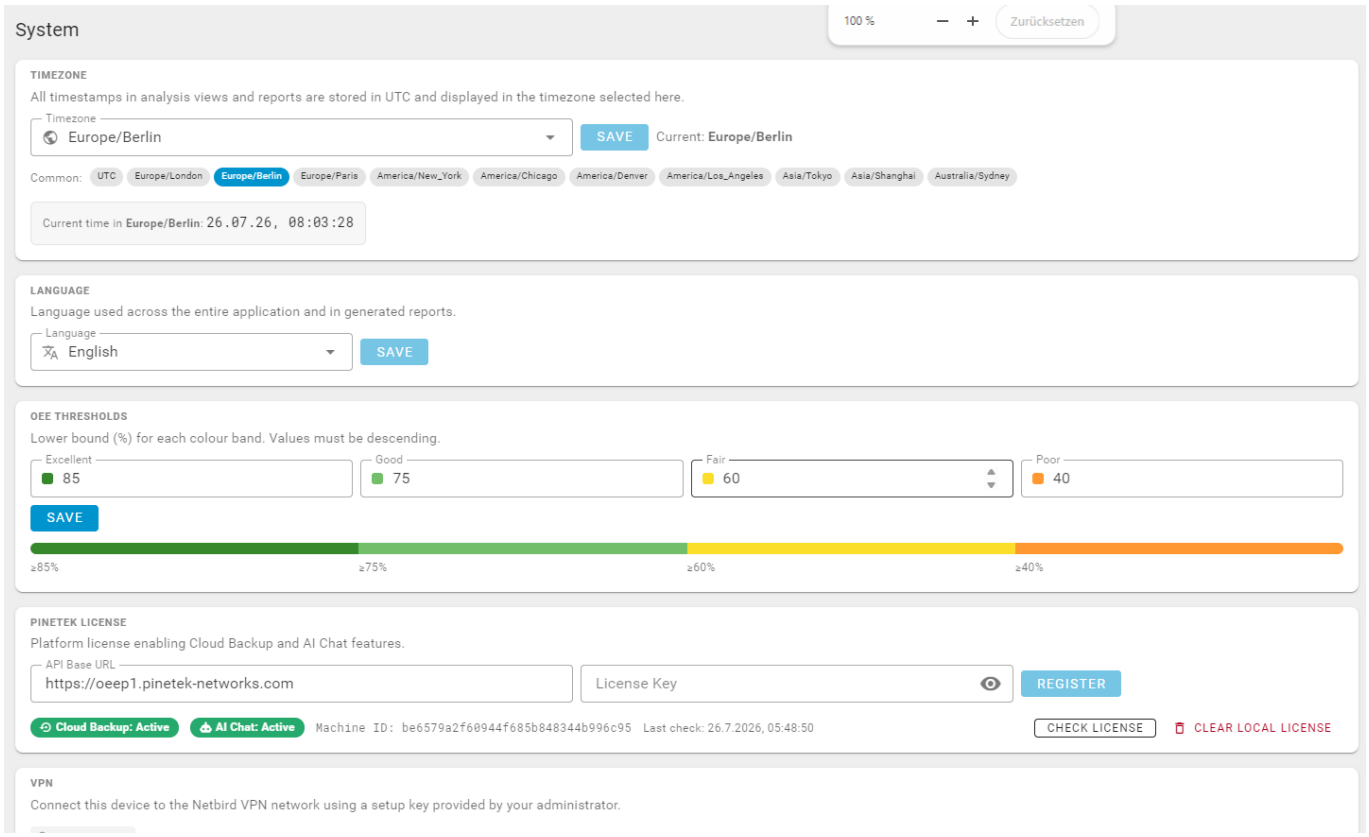
8.7 Generating, testing and dispatch

- **On demand:** *Templates* list → **Generate / Download PDF**; pick the run (run report) or date range and optional filters (period report).
- **Test e-mail:** sends a generated report to a single address of your choice; use this to verify SMTP and layout before enabling automatic dispatch.
- **Run end:** active run-end templates are offered for dispatch in the Run Complete dialog (chapter 5.2.4).
- **Scheduled:** active scheduled templates are generated and e-mailed automatically at the configured time; *Last Sent* in the list shows the most recent dispatch.

9. System Administration

All functions in this chapter require the **Admin** role.

9.1 System configuration

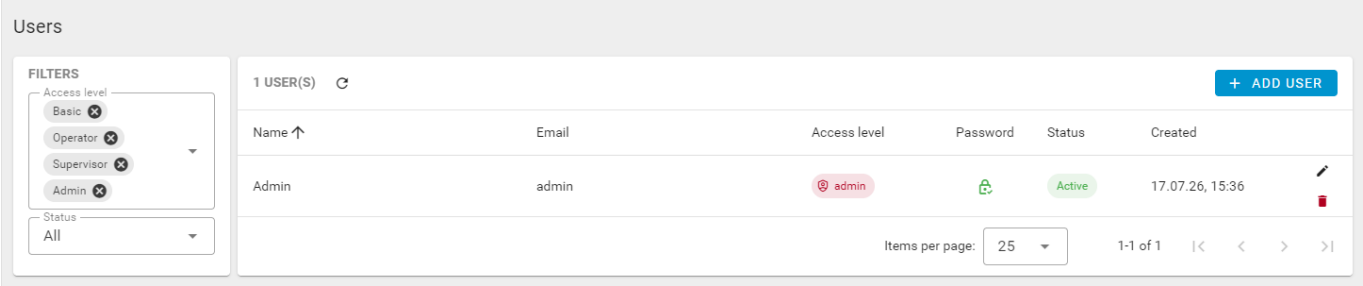


System → System holds the global settings:

- **Timezone:** all timestamps are stored in UTC and displayed in the timezone selected here (analysis views and reports alike).
- **Language:** English or German; applies to the entire web interface and to generated reports.
- **OEE Thresholds:** the lower bounds (%) of the color bands *Excellent / Good / Fair / Poor* used wherever OEE values are color-coded. Values must be descending.
- **VPN:** optional remote-maintenance connection via Netbird: install the client and connect with a one-time **setup key** provided by Pinetek Networks or your Administrator. The card shows connection status, VPN IP, FQDN and peers.
- **Data Retention:** automatically delete production data older than the configured number of days (`0` = keep forever). The purge runs daily at 02:30 (local time); **Purge Now** runs it immediately.
- **Danger Zone:** irreversible maintenance actions:
  - **Delete Database:** permanently deletes all production data and configuration and re-creates an empty schema. Requires typing `DELETE` to confirm. The default login is restored to admin/admin
  - **Seed Test Data:** fills the database with 60 days of realistic demo data (products, programs, runs, downtime and state events) for training/demos. Existing data is kept.

Delete Database cannot be undone. Make a backup first (9.3).

9.2 User management

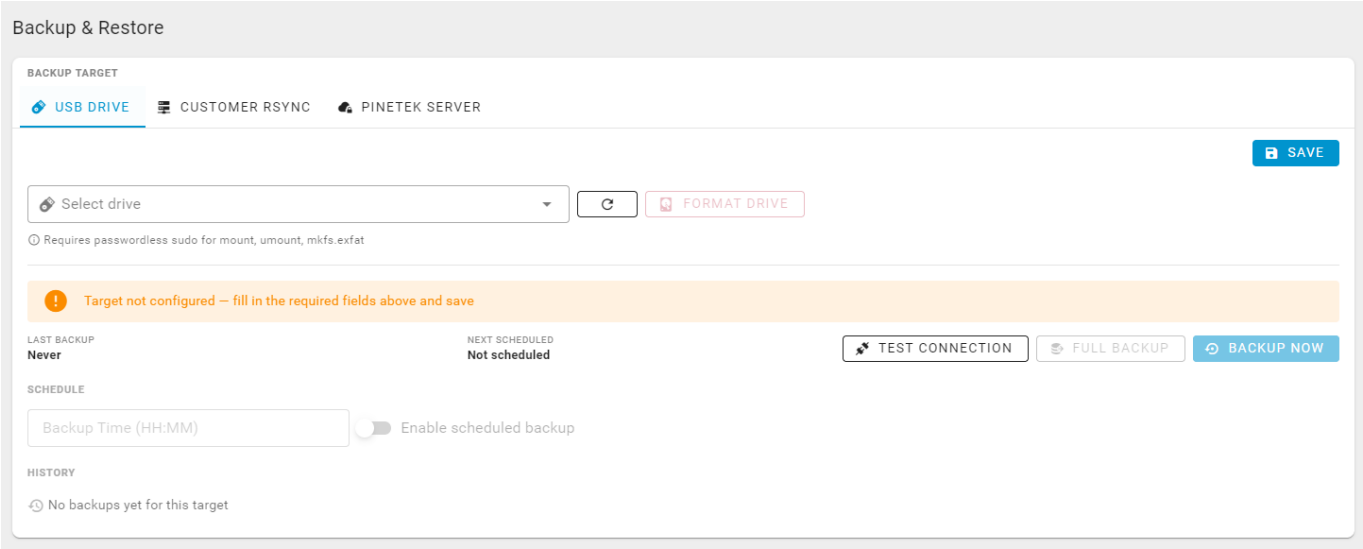


System → Users manages accounts. Per user: **name**, **e-mail**, **password** and **access level** (*Basic*, *Operator*, *Supervisor*, *Admin*; see chapter 4.2). Basic accounts have no password; all other levels require one. When editing, leave the password field blank to keep the current password. Users can be deactivated instead of deleted to preserve their history.

9.3 Backup & Restore

System → Backup protects your production data. The status card shows the last backup and the next scheduled one. Configure one or more targets and mark one as **active**:

USB Drive



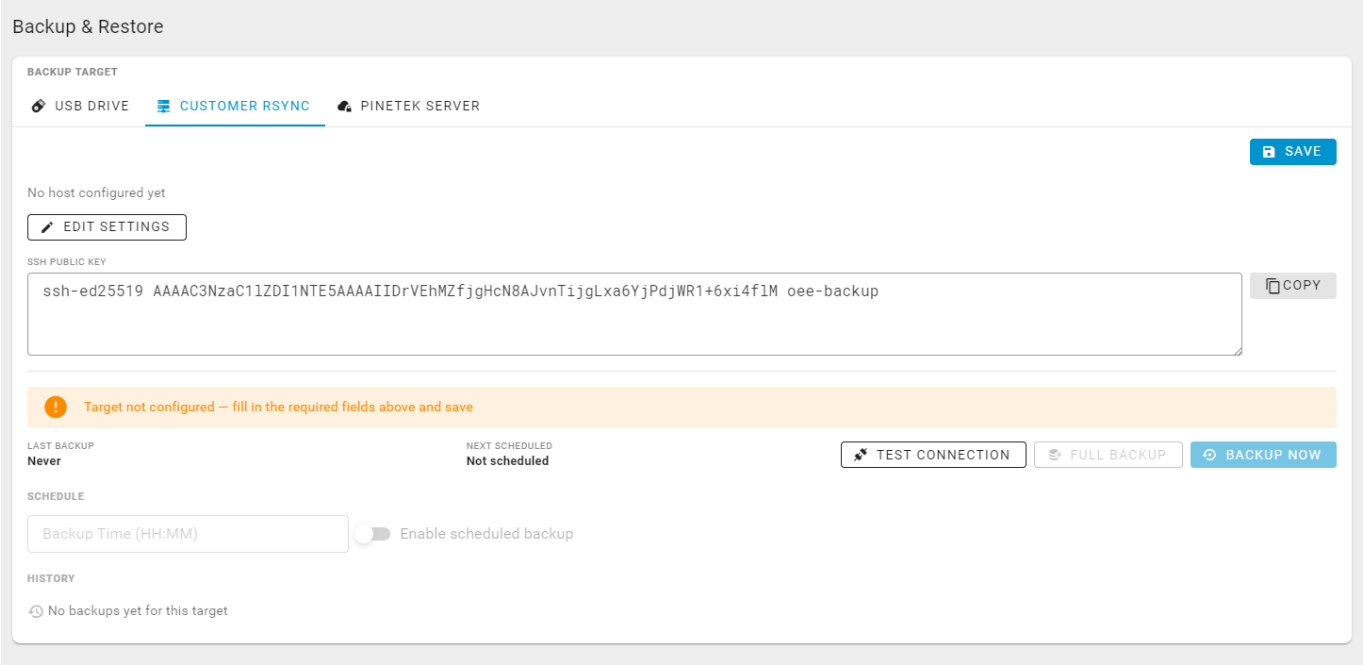
The USB Drive backup stores the data on a USB drive connected to the USB port of the Pinebox.

The view lists detected drives and can **format** a new drive (!).

On format, all data on the USB stick on it is erased. The USB Drive backup is the least secure backup. In case of issues with the Pinebox (overvoltage or similar), there is risk of data loss on the Pinebox and the backup media at the same time.

Customer rsync

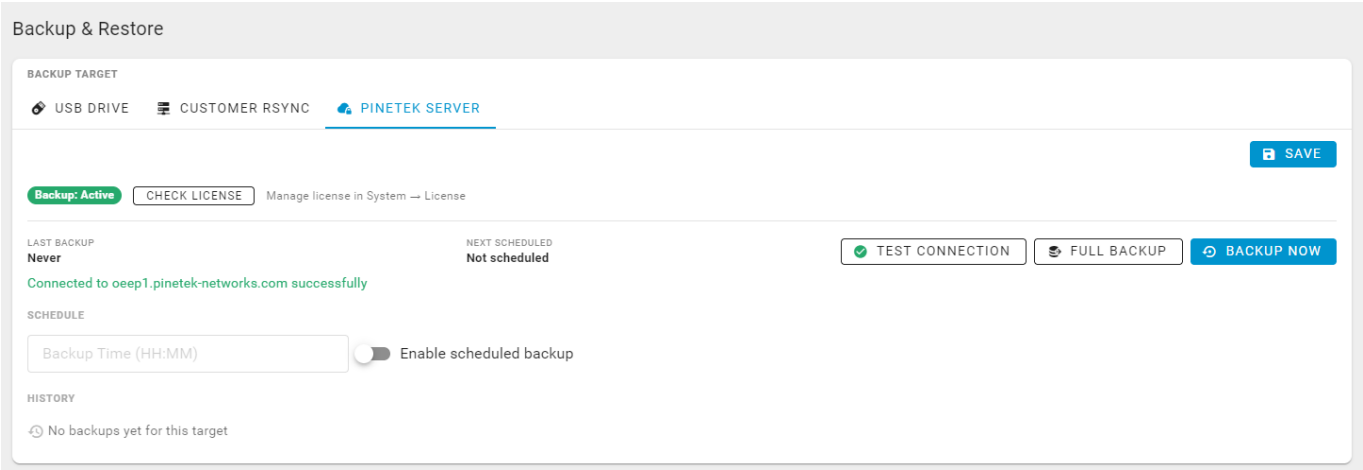
Backup to your own server via rsync/SSH



You need to obtain the settings (host, user, remote path, port) for the remote rsync from your local Administrator, matching to your rsync setup. **Test Connection** verifies the setup.

Pinetek Server

Managed cloud backup to Pinetek Networks infrastructure.



Requires the **Cloud Backup** license entitlement (9.6). The backup target are the Pinetek servers, no further customer infrastructure is required.

Running backups

- **Backup Now**: immediate incremental backup (only data since the last backup).
- **Full Backup**: forces a complete dump regardless of what is already on the destination.
- **Schedule**: enable daily automatic backup at a set time (HH:MM).
- **History**: every backup with target, trigger, covered data range, size and status (*Running / Success / Failed*).

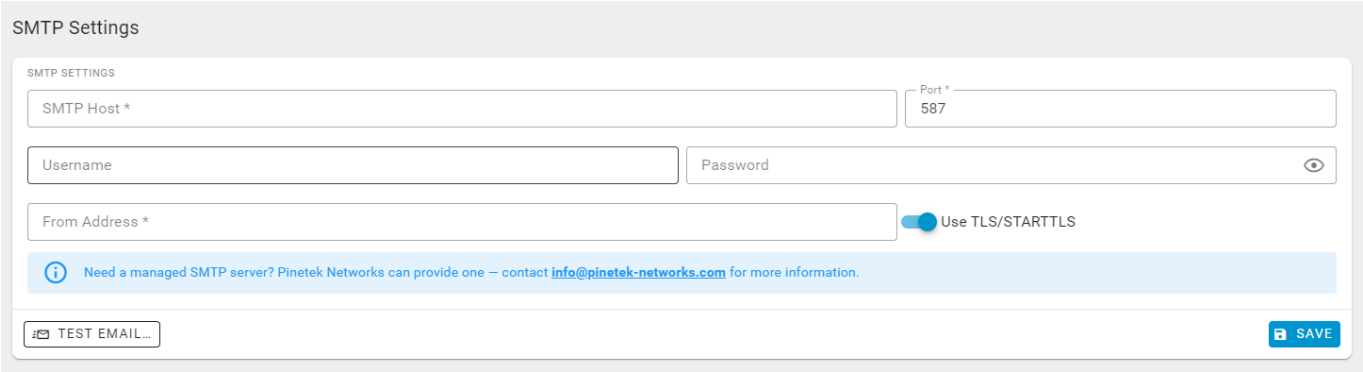
Restore

Load Backups lists the backups found on the active target; select one and press **Restore**.

Restoring overwrites the current data and cannot be undone.

9.4 E-mail / SMTP settings

System → SMTP configures the mail server

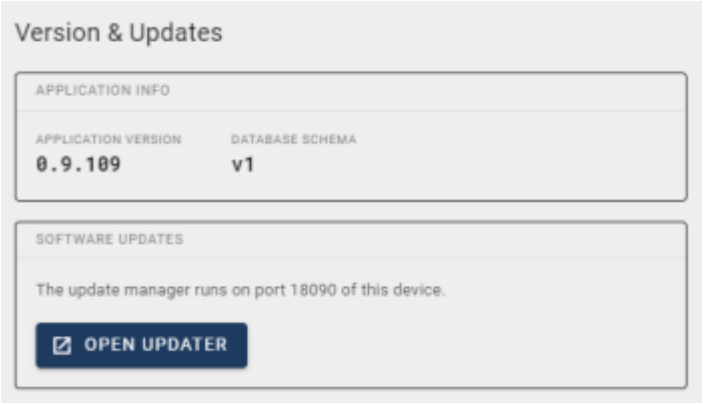


To send mails from the CedarInsight system (reports, alarms, ..), an external SMTP server is required that can be reached from the Pinebox that CedarInsight runs on. This SMTP is usually provided by your local Administrator or can be provided by Pinetek Networks (contact us for details).

In either case, the configuration data needs to be entered: **host**, **port**, **username/password**, **from address** and **TLS/STARTTLS**.

Use **Test Email** to verify the configuration.

9.5 Software version & updates



System → Version shows the **application version** and **database schema version**. This is relevant information if you need to contact support.

Software updates are handled by the separate update manager, which runs on port `18090` of the device; **Open Updater** takes you there.

CedarSense uses the same updater as the Pinebox. The file format are .pfw files.

9.6 License management

The Pinetek license shows the **machine ID**, registration state, time of the **last check** and the entitlement status:

Entitlement	Enables
Core	Running CedarInsight at all (chapter 3.5)
Cloud Backup	Backup target *Pinetek Server* (9.3)
AI Chat	The AI Assistant (6.5)

Actions: **Register** (with a license key), **Check License** (immediate re-validation, otherwise automatic every 24h, cached locally), and **Clear Local License** (removes the locally stored key and entitlements; the machine stays registered on the server).

If the CedarInsight system is operated in a non-online, isolated environment, the license does not expire.

9.7 Disk-space monitoring

The system monitors free disk space in the background and raises notifications (chapter 4.5) when space runs low. If warnings appear, reduce the data-retention period (9.1), purge old data, or contact support.

10. OEE Calculation Reference

This chapter documents exactly how CedarInsight computes its figures, so every displayed value can be audited. All analysis views, reports and the AI Assistant use the same calculation functions; figures are consistent everywhere.

10.1 Data captured per production run

Quantity	Source
Actual quantity (units in)	Input-stage counter × units-per-count [PBX]
Good quantity (units out)	Output-stage counter × units-per-count [PBX]
Defect quantity	Defect/Reject-stage counter; can be corrected by the operator at run end (5.2.4)
Target quantity	Entered at changeover
Running time	Total time in states *Running* and *Running Slow* (from state events)
Unplanned downtime	Total time in state *Down*
Planned production time	Wall-clock run duration (start → end)
Ideal cycle time	From the product (or program default) configuration

10.2 Availability

```
Availability = Running time / (Running time + Unplanned downtime)
```

Availability relates run time to operating time. Time in *Planned Stop*, *Changeover* and *Draining* is not an availability loss, only unplanned *Down* time reduces Availability. This follows the standard OEE definition in which planned stops are excluded from the loss calculation.

10.3 Performance

```
Performance = (Ideal cycle time × Actual quantity) / Running time
```

Performance compares the time the production *should* have taken at ideal speed with the time it actually ran. Slow cycles and minor stops (shorter than the stage timeout) reduce Performance. The value is capped at 100%.

Performance is only as accurate as the configured **ideal cycle time** (chapter 7.2).

10.4 Quality

```
Quality = Good quantity / Actual quantity
```

Good quantity is the output-stage count; actual quantity is the input-stage count. Operator corrections of the defect quantity at run end are reflected here.

10.5 OEE

```
OEE = Availability × Performance × Quality
```

10.6 Aggregation across runs

When several runs are combined (period views, grouping by product/program, reports), the factors are weighted, not simply averaged:

Factor	Aggregation
Quality	Volume-weighted: Σ good / Σ actual over all runs
Availability	Time-weighted: Σ running / Σ (running + unplanned downtime)
Performance	Running-time-weighted mean of the per-run values, mathematically equal to Σ (ideal cycle × count) / Σ running time
OEE	Recomputed from the three aggregated factors: A × P × Q

This means a long run influences the aggregate more than a short one; a simple arithmetic mean of run OEEs would distort the result.

10.7 OEE colour scale

OEE values are colour-coded throughout the application. The default bands:

Band	Default threshold	Colour
Excellent	≥ 85 %	dark green
Good	≥ 75 %	light green
Fair	≥ 60 %	yellow
Poor	≥ 40 %	orange
Critical	< 40 %	red

The thresholds are configurable under *System → System → OEE Thresholds* (chapter 9.1).

11. Pinebox Interface Reference

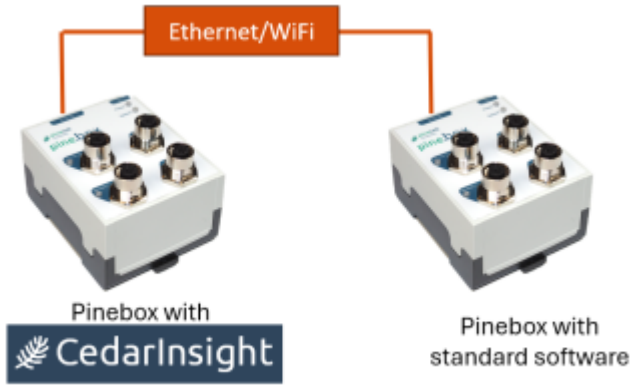
This chapter describes which Pinebox services CedarInsight consumes and how it behaves when they are unavailable. Everything on the device side (wiring, IO-Link configuration, counter/timer/cycle setup, device networking) is documented in the Pinebox Manual [PBX].

11.1 Overview

Service	Direction	Default endpoint	Used for
Counter values	CedarInsight polls	`http://<pinebox>:18080/counter/values`	Quantities, state detection, stage throughput
Counter reset	CedarInsight calls at production run start	`http://<pinebox>:18080/counter/reset`	Zeroing stage counters for a new run
Takt timers	CedarInsight polls	`http://<pinebox>:18080/timer/values`	Takt/cycle-time monitoring
Cycle records	CedarInsight polls	`http://<pinebox>:18080/cycles`	Cycle stations (duration, energy, stalls)
Sensor values	Pinebox side pushes (Node-RED)	CedarInsight API `/api/sensors/ingest`	Process values (temperature, pressure, ...)

11.2 Reachability and multiple Pinebox devices

CedarInsight does not have a single “Pinebox address”. Instead, every registered counter, takt source and cycle station stores its own host and port (default port `18080`). This is what allows one CedarInsight to collect data from several Pinebox devices, for example one Pinebox per machine section of a longer line.



Network requirements

- All connections are outgoing HTTP from the CedarInsight device to each Pinebox on TCP port `18080` (no connections from the Pinebox to CedarInsight are needed for counters/takt/cycles). Firewalls between the devices must permit this direction.
- In the standard single-device installation, the Pinebox services run on the same host and are addressed as `localhost:18080`; no network configuration is needed.
- Additional Pinebox devices should have static IP addresses (or DHCP reservations), because the address is stored with each counter/takt/cycle-station entry. If a Pinebox changes its IP, every entry referring to it must be updated. Assigning the address is done on the device, see the Pinebox Manual [PBX].
- The exception is the sensor push (section 11.6): there the direction is reversed: the Pinebox/Node-RED side must be able to reach CedarInsight on TCP port `8000`.

Discovering devices

The **Scan network** function in the Counters, Cycle Stations and Takt views probes the local /24 subnet of the CedarInsight device for hosts answering on port `18080` and lists them with hostname (the CedarInsight host itself is marked “This device”).

- Only devices in the same subnet are found by the scan. A Pinebox in a different subnet or VLAN is still usable, enter its host/IP and port manually in the corresponding dialog.
- After selecting a device, the second scan step reads its available keys (counter keys, timer keys or cycle stations) so you can import them without typing.

Verifying reachability

- *Line Config* → *Counters* → **Read** fetches live values from all registered hosts in one step and explicitly lists hosts that could not be reached; the quickest overall health check of a multi-Pinebox setup.
- Cycle-station panels show *“Source unreachable — showing last known data”* when their device stops responding (section 11.5).
- For persistent problems, verify with `ping` / a browser call to `http://<pinebox>:18080/counter/values` from the CedarInsight host, and check the device itself [PBX].

Keep a written mapping of *Pinebox IP* ↔ *machine section* in your line documentation. In CedarInsight itself, use the **description** field of counters, takt sources and cycle stations to record which physical device they live on.

11.3 Counter values

CedarInsight polls the counter endpoint cyclically (sampling interval per program, default 1 s; readings are persisted at the program's write interval, default 60 s). The response is a set of **counter keys** with monotonically increasing values [PBX].

From consecutive readings CedarInsight derives:

- **Deltas**: produced quantities per stage (multiplied by *units per count*),
- **Activity/inactivity**: state transitions: a stage whose counter does not advance within its **timeout** is considered stalled; input/output stage stalls drive the *Down*, *Draining* and run-end logic (chapter 2.3.2),
- **Throughput per stage**: bottleneck detection (chapter 5.3).

At the start of each production run, CedarInsight resets the counters of the active program via the reset endpoint, so each run starts counting from zero.

11.4 Takt timers

The timer endpoint provides named **takt keys** with measured cycle/interval times [PBX]. CedarInsight polls registered takt sources, stores the series and evaluates them against the per-program warning/critical thresholds (chapter 7.1). Takt series can be overlaid in the run trend chart (chapter 6.3).

11.5 Cycle records

For each registered cycle station, CedarInsight polls the cycles endpoint (with the station key and a *since* watermark) and receives the detected machine cycles: start time, duration, energy, peak power and status. Station discovery uses the station list the device advertises. Stall handling (auto-scrap / ask operator / auto-

continue) and detector calibration are described in chapter 7.4.

The raw power measurement and cycle detection run on the source device [PBX]; CedarInsight stores, displays and integrates the results into the production run (a stage in *Cycle mode* counts cycles instead of counter ticks).

11.6 Sensor values, push via Node-RED

Unlike the polled services above, sensor values are pushed into CedarInsight by the data-source side, typically a Node-RED flow running on the Pinebox [PBX], which reads the physical sensors (IO-Link, Modbus, …) and forwards the values over HTTP.

The ingest endpoint

```
POST http://<cedarinsight-host>:8000/api/sensors/ingest
Content-Type: application/json
```

The body is a single sample object or an array of them:

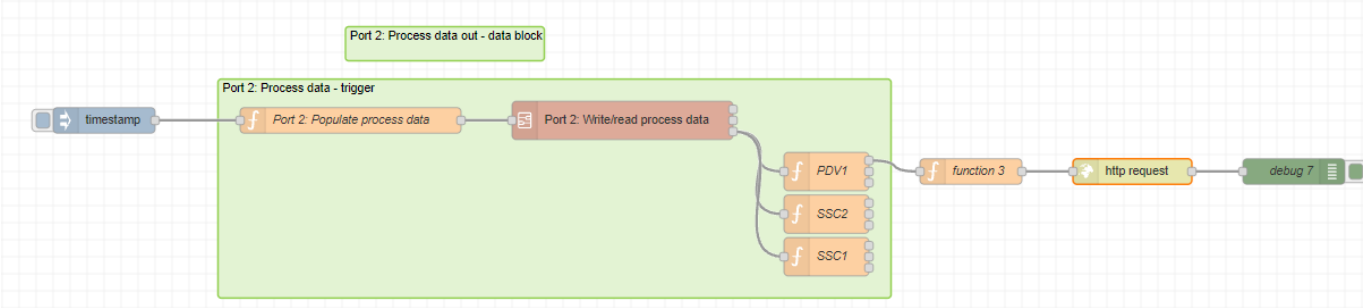
```
[
  { "sensor_key": "line1_temp",  "value": 74.2, "unit": "°C",  "display_name": "Oven
temperature" },
  { "sensor_key": "line1_press", "value": 5.8,  "unit": "bar", "display_name": "Air pressure"
}
]
```

Field	Required	Meaning
`sensor_key`	yes	Unique key of the sensor; must match the key registered in *Line Config → Sensors* (chapter 7.5) exactly
`value`	no	Numeric sample value
`unit`	no	Physical unit, e.g. `°C`, `bar`
`display_name`	no	Human-readable label
`timestamp`	no	ISO-8601 sample time; if omitted, the server receipt time is used
`quality`	no	Optional quality flag

Behaviour:

- The endpoint answers `202 Accepted` immediately; values are buffered and written to the database in a 60-second flush cycle.
- **Auto-registration:** if a `sensor_key` is not yet known and the sample carries `display_name`/`unit`, the sensor is registered automatically. Names/units already set in the UI are never overwritten by pushed values; the UI wins.
- Keys that are sending data appear in *Line Config → Sensors → Discover sensors* with their current value, ready to **Register** (chapter 7.5).

Setting up the Node-RED flow



A minimal flow consists of three nodes:

1. **Sensor input node**: whatever delivers your raw value (in the image above, the process data of a connected sensor) [PBX]
2. **Function node**: build the sample object:

```
msg.payload = [{
    sensor_key:   "line1_temp",
    value:        Number(msg.payload),
    unit:         "°C",
    display_name: "Oven temperature"
}];
return msg;
```

3. **HTTP request node**: Method: `POST`, URL: `http://<cedarinsight-host>:8000/api/sensors/ingest` (use `http://localhost:8000/...` when Node-RED runs on the CedarInsight device itself), Content-Type `application/json`.

Recommendations:

- **Batch** several sensors into one array per request instead of one request per sensor.
- A push rate of 1 sample per second per sensor matches the storage resolution; pushing faster brings no benefit.
- Add a small retry/queue (e.g. Node-RED `catch` + delay) if the network between Node-RED and CedarInsight is unreliable; samples not delivered are lost (section 11.7).
- Verify the flow with *Line Config* → *Sensors* → *Discover sensors*: the key must appear with a live value within seconds.

11.7 Behaviour when a source is unreachable

Situation	Behaviour
Counter endpoint unreachable during a run	No counter advance is detected, after the stage timeout the state machine reports **Down**. The stop appears as unplanned downtime and should be classified accordingly (e.g. a dedicated reason “Data source failure”).
Counter endpoint unreachable while Stopped	No effect on data; the counter registry's **Read** function reports which hosts could not be reached.
Cycle station source unreachable	The station panel shows *“Source unreachable — showing last known data”*; polling resumes automatically and missed cycles are fetched via the *since* watermark.
Sensor data stops arriving	Gaps appear in the sensor series; no state impact. Samples that were never delivered are not recoverable, buffer on the Node-RED side if needed.

Persistent connection problems are usually network or device issues, see section 11.2, the troubleshooting steps in chapter 12.1 and the Pinebox Manual [PBX].

12. Troubleshooting & FAQ

12.1 No counter data / state stuck

Symptom: counters on the Production Timeline do not advance, or the line shows *Down* although the machine is running.

1. Check the Pinebox: is the device powered, connected and are the sensors counting? → Pinebox Manual [PBX].
2. *Line Config* → *Counters* → **Read**: are the live values reachable and advancing? Unreachable hosts are listed explicitly.
3. Verify the **counter key** and **host/port** of the affected stage's counter (chapter 7.3), a renamed key on the Pinebox breaks the mapping. Re-run **Scan for counters** to compare.
4. Check the stage **timeout** (chapter 7.1): a timeout shorter than the real production rhythm causes false *Down* states (e.g. slow lines, batch transfer).
5. Ping the Pinebox from the network; check cabling/switch.

12.2 Login and permission problems

- **“Invalid email or password”**: check username/e-mail spelling; an admin can set a new password (*System* → *Users*).
- **A menu entry is missing / redirect to Scoreboard**: your access level does not permit the page (see roles matrix, Appendix C). An admin can raise your level.
- **No admin available**: contact Pinetek support.

12.3 Reports are not generated or not e-mailed

1. Is the template **active** and its **trigger** correct (chapter 8.3)?
2. *System* → *SMTP*: settings complete? Send a **test e-mail** (chapter 9.4).
3. Send a **test report e-mail** from the template list, this isolates report generation from dispatch.
4. Run-end reports: were they skipped in the Run Complete dialog (**OK, do not send email**)?
5. Scheduled reports: check *Last Sent* in the template list; verify the device **timezone** (chapter 9.1).
6. Check *Notifications* for error entries.

12.4 License problems

- **License Required screen** although previously licensed: press **Check License** (chapter 9.6). If the check fails, verify the device's internet access to the Pinetek license server.
- **AI Assistant or Pinetek backup “not licensed”**: the corresponding entitlement is not active; contact Pinetek.

12.5 Disk space and database

- Disk-space warnings in Notifications: reduce the **data-retention** period and run **Purge Now** (chapter 9.1); check the backup history for failed backups accumulating locally.
- The daily retention purge runs at 02:30.

12.6 Behaviour after restart / power failure

After a restart, the application resumes automatically. If a production run was active during the outage:

- The run may remain **ACTIVE** without a live line. Close it in *Analysis* → *Production* with **Close Run** (chapter 6.3); the OEE metrics are then calculated.

- The outage period appears as it was recorded; assign a downtime reason where required (chapter 6.1).

12.7 Frequently asked questions

Why is Availability 100 % although the line stood still? Planned stops, changeover and draining do not count as availability losses (chapter 10.2). Only unplanned *Down* time reduces Availability.

Why does the aggregated OEE differ from the average of the run OEEs? Aggregation is time- and volume-weighted (chapter 10.6); long runs weigh more than short ones.

Why does Performance never exceed 100 %? The value is capped; if it would exceed 100 %, the configured ideal cycle time is too long (chapter 7.2).

Can I correct a wrong defect count after the run? At run end via the Run Complete dialog (5.2.4). Later corrections require closing/re-evaluating the run; contact your administrator.

Which browser should I use? A current Chrome, Edge or Firefox. For wall displays, use a **Basic** account with the Scoreboard.

Appendices

A. Glossary

Term	Definition
Availability	OEE factor: running time / (running time + unplanned downtime)
Bottleneck	The process stage whose throughput limits the whole line
Changeover	Setup/product-change phase before a production run
Cycle station	Machine monitored via energy-based cycle detection [PBX]
Defect / Reject	Part not meeting quality requirements; counted by the reject stage or corrected manually
Downtime reason	Category + reason classifying a Down period
Draining	Run-out phase: input stopped, line runs empty
Entitlement	A licensed feature on the Pinetek license (Core, Cloud Backup, AI Chat, SMTP)
Ideal cycle time	Theoretical time to produce one unit at full speed; basis of Performance
OEE	Overall Equipment Effectiveness = Availability × Performance × Quality
Pareto analysis	Ranking of causes by impact with cumulative percentage
Performance	OEE factor: (ideal cycle time × actual quantity) / running time
Pinebox	Pinetek field data-acquisition device; source of all machine signals [PBX]
Process stage	One counting point of the line, with role Input/Output/Defect/Intermediate
Product	Produced article, linked to a program, with part number and ideal cycle time
Production run	Continuous production of one product; unit of OEE calculation
Program	Line configuration: stages, counters, sensors, takt, detection parameters
Quality	OEE factor: good quantity / actual quantity
Scrap rate	Defect quantity / actual quantity
State event	Stored record of one contiguous period in a production state
Takt	Measured cycle/interval time from a Pinebox timer [PBX]
Target quantity	Planned quantity of a run, entered at changeover
Units per count	Multiplier from counter ticks to units (e.g. one tick = pack of 6)

B. State reference

State	Color	Entered by	Left by	OEE effect
Stopped	grey	Run end, drain complete	Start Changeover	outside any run
Changeover	purple	**Start Changeover** button	counting starts	not an availability loss
Running	green	counting detected	stall, slow-down, operator action	running time
Running Slow	orange	throughput drop	recovery or stall	running time; reduces Performance
Down	red	stage counter timeout	counting resumes	**unplanned downtime, reduces Availability**; reason required
Planned Stop	yellow	operator action	**Resume** button	planned; no Availability loss

State	Color	Entered by	Left by	OEE effect
Draining	blue	**Drain Line** button	output counter timeout → run ends	not an availability loss

C. Roles and permissions matrix

Function	Basic	Operator	Supervisor	Admin
Scoreboard	✓	✓	✓	✓
Production Timeline (run control, downtime reasons)	–	✓	✓	✓
Live Bottleneck Analysis	–	✓	✓	✓
Notifications	–	✓	✓	✓
Analysis (OEE History, Production, Bottleneck History, Downtime Pareto)	–	–	✓	✓
AI Assistant	–	–	✓	✓
Line Config (Program, Product, Counters, Cycle Stations, Sensors, Takt, Downtime)	–	–	✓	✓
System configuration, Backup, Users, SMTP, Version, License	–	–	–	✓
Reports (Templates, Blocks, Layout)	–	–	–	✓

D. Technical data

Item	Value
Web interface	`http://<device-ip>:8000`
Supported browsers	current Chrome, Edge, Firefox
Pinebox data services	port `18080` (counters, timers, cycles) [PBX]
Update manager	port `18090`
Database	MariaDB, local on the device
Timestamps	stored in UTC, displayed in the configured timezone
Languages	English, German (system-wide, incl. reports)
Report output	PDF (A4), optional e-mail dispatch via SMTP
Data export	Excel (.xlsx) from analysis tables
License check	Pinetek license server, once per 24 h, cached locally

E. Contact and support

Supplier information

Pinetek Networks GmbH
Bachstelzenweg 8
DE-88677 Markdorf

www.pinetek-networks.com
info@pinetek-networks.com

Data privacy

The device does not collect, store, or transmit data outside of the physical device, unless the user programs the device accordingly (AI feature, online backup, email; see sections in this manual).

Support

Support for CedarInsight is available through multiple channels:

- The Pinetek Knowledge Base (online manuals): doc.pinetek-networks.com
- Latest firmware, PDF manuals and other information: download.pinetek-networks.com/cedar-insight

For individual support, please use our support portal: support.pinetek-networks.com