11. Pinebox Interface Reference

Hardware manual

www.pinetek-networks.com

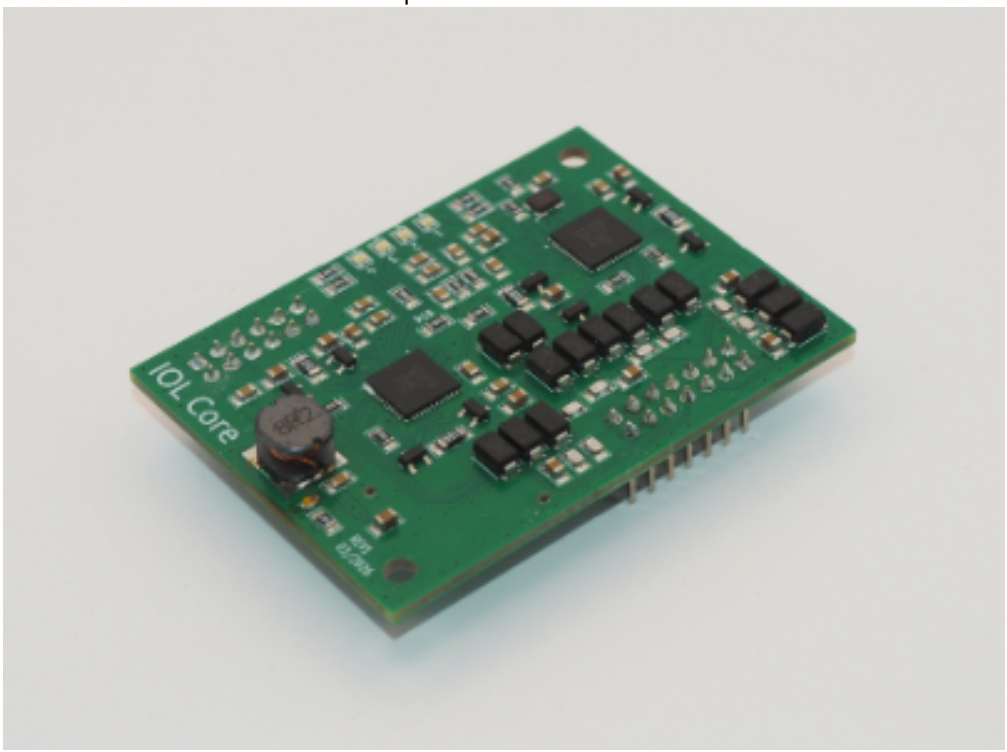


Table of Contents

11. Pinebox Interface Reference 5
11.1 Overview 5
11.2 Reachability and multiple Pinebox devices 5
Network requirements 5
Discovering devices 6
Verifying reachability 6
11.3 Counter values 6
11.4 Takt timers 6
11.5 Cycle records 6
11.6 Sensor values, push via Node-RED 7
The ingest endpoint 7
Setting up the Node-RED flow 7
11.7 Behaviour when a source is unreachable 8

11. Pinebox Interface Reference

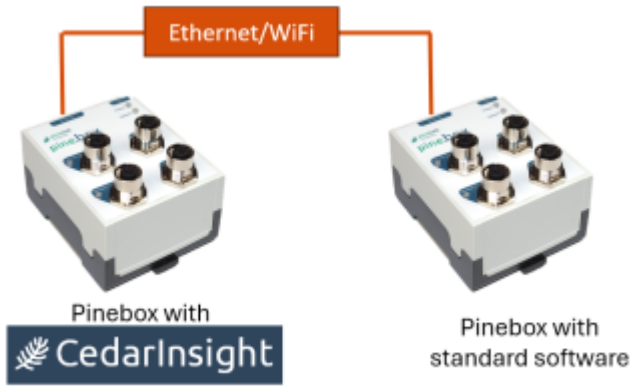
This chapter describes which Pinebox services **Pinetek Insight** consumes and how it behaves when they are unavailable. Everything on the device side (wiring, IO-Link configuration, counter/timer/cycle setup, device networking) is documented in the Pinebox Manual [PBX].

11.1 Overview

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

11.2 Reachability and multiple Pinebox devices

Pinetek Insight 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 **Pinetek Insight** 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 **Pinetek Insight** device to each Pinebox on TCP port `18080` (no connections from the Pinebox to **Pinetek Insight** 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 **Pinetek Insight** 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 **Pinetek Insight** device for hosts answering on port `18080` and lists them with hostname (the **Pinetek Insight** 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 **Pinetek Insight** host, and check the device itself [PBX].



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

11.3 Counter values

Pinetek Insight 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 **Pinetek Insight** 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, **Pinetek Insight** 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]. **Pinetek Insight** 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, **Pinetek Insight** 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]; **Pinetek Insight** 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 **Pinetek Insight** 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://<pinetek-insight-host>:8000/api/sensors/ingest
Content-Type: application/json
```

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

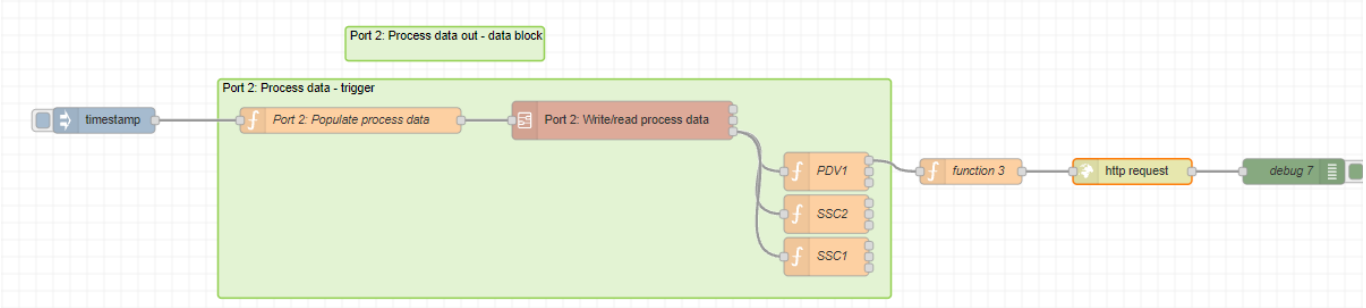
```
[
  { "sensor_key": "line1_temp",  "value": 74.2, "unit": "°C",  "display_name": "Oven
temperature" },
  { "sensor_key": "line1_press", "value": 5.8,  "unit": "bar", "display_name": "Air pressure"
}
]
```

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://<pinetek-insight-host>:8000/api/sensors/ingest` (use `http://localhost:8000/...` when Node-RED runs on the **Pinetek Insight** 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 **Pinetek Insight** 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].