PSBMS Engineering Tool — User Manual

A field commissioning tool for BMS/controls engineers: BACnet and Modbus device testing, network discovery, live trending and bench calculators, in one app.

- Windows 10 / 11 (64-bit) · © PSBMS Ltd 2026 · the app footer shows your installed version
- Download & support: https://tool.psbms.com

---

1. Installing, editions & first run

Install. Run `PSBMS Engineering Tool Setup.exe` and follow the prompts (accept the Windows elevation prompt). It installs to Program Files and adds Start-Menu and Desktop shortcuts. The release also includes a portable ZIP: extract the complete folder, keep its `_internal` directory beside the signed `PSBMS Engineering Tool.exe`, and double-click the executable.

Terms of use. Before anything is installed, the installer shows the terms of use; click **I Agree** to continue. The portable build has no installer, so the first time it starts it shows the same terms and asks you to agree — **I agree** starts the tool, **Exit** closes it without doing anything else. It asks once per Windows user, and again only if a later release changes the wording. The terms are short: the tool is an aid for competent engineers; you use it at your own risk and are responsible for how it is used — being authorised to connect to what you connect to, making sure any change is safe, and checking results before relying on them; it is not a safety system; and, as far as the law allows, PSBMS Ltd accepts no liability for its use or misuse. The **About** window shows them in full at any time.

> The public installer and portable executable are Authenticode-signed by **PSBMS LIMITED**. Stop and download the release again from https://tool.psbms.com if Windows reports a different or unknown publisher.

Editions. The tool is free for everyday work and unlocks its professional features with a licence:

- **Community (free)** — the complete read-the-plant kit: BACnet, Modbus, IP Scanner, Tools, the single-device simulators and a 2-channel Trend preview. No licence needed. **Importing and exporting data files — CSV import/export, the Excel/JSON round-trip and the branded PDF report — is a Pro/Trial feature**; clicking one shows a short upsell.
- **Pro** — unlocks all data **import/export** (clean, no watermark), the **Converters** module and branded report, plus unlimited Trend channels.
- **14-day trial** — unlocks the full Pro feature set including import/export; Trial exports stay watermarked until you buy.

First run. The welcome screen offers three choices:

1. **Continue Free** — start using the Community tier straight away, no code.
2. **Start 14-day trial** — the app unlocks the Pro feature set locally for 14 days. No email or account is required.

3. **Activate** — paste a licence code. If you're offline, the screen shows **Your Machine ID** (e.g. `DEMO-0000-0000-0000`) - send it to PSBMS, paste the code we return, and click **Activate**.

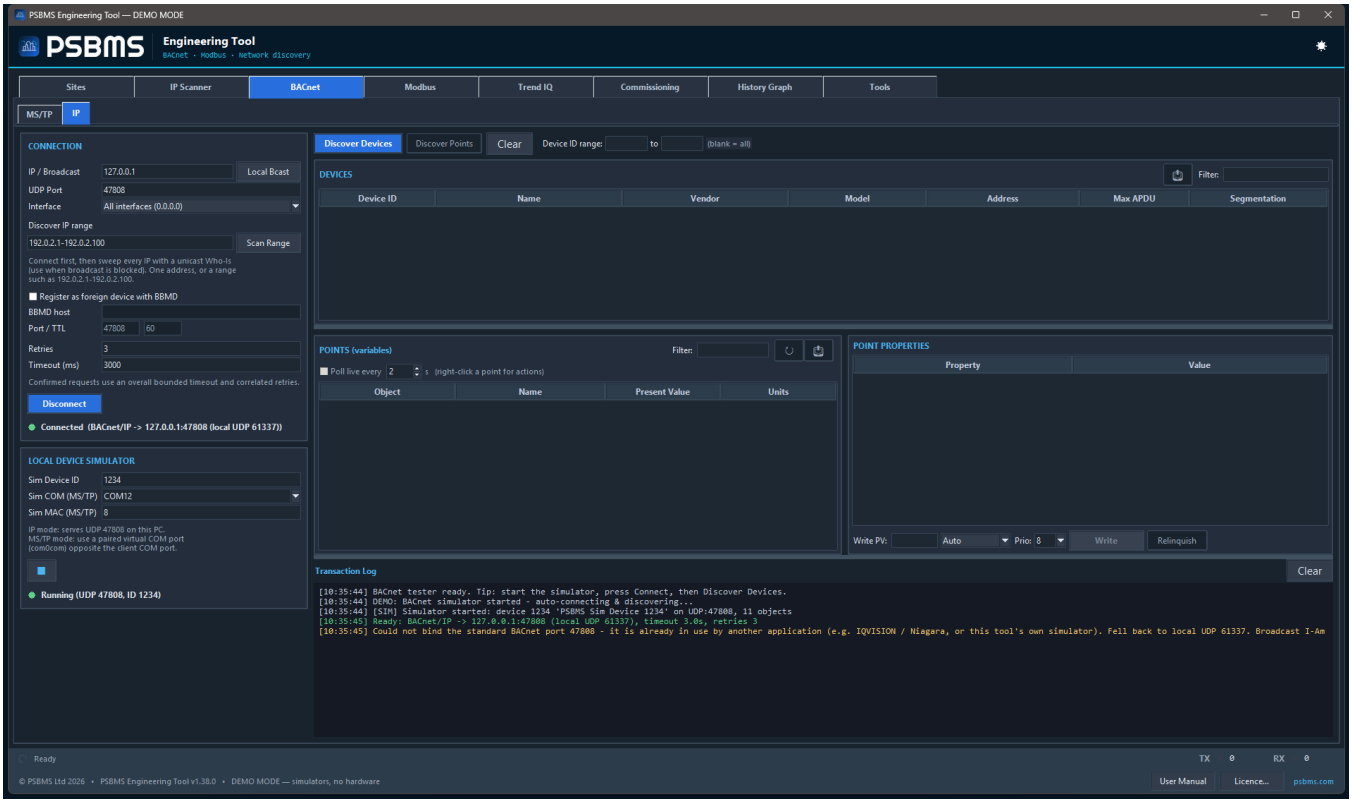
The footer always shows the current edition and, for a licence, when it expires. When a subscription lapses the app simply drops back to Community — it is never locked out, and your saved work is kept.

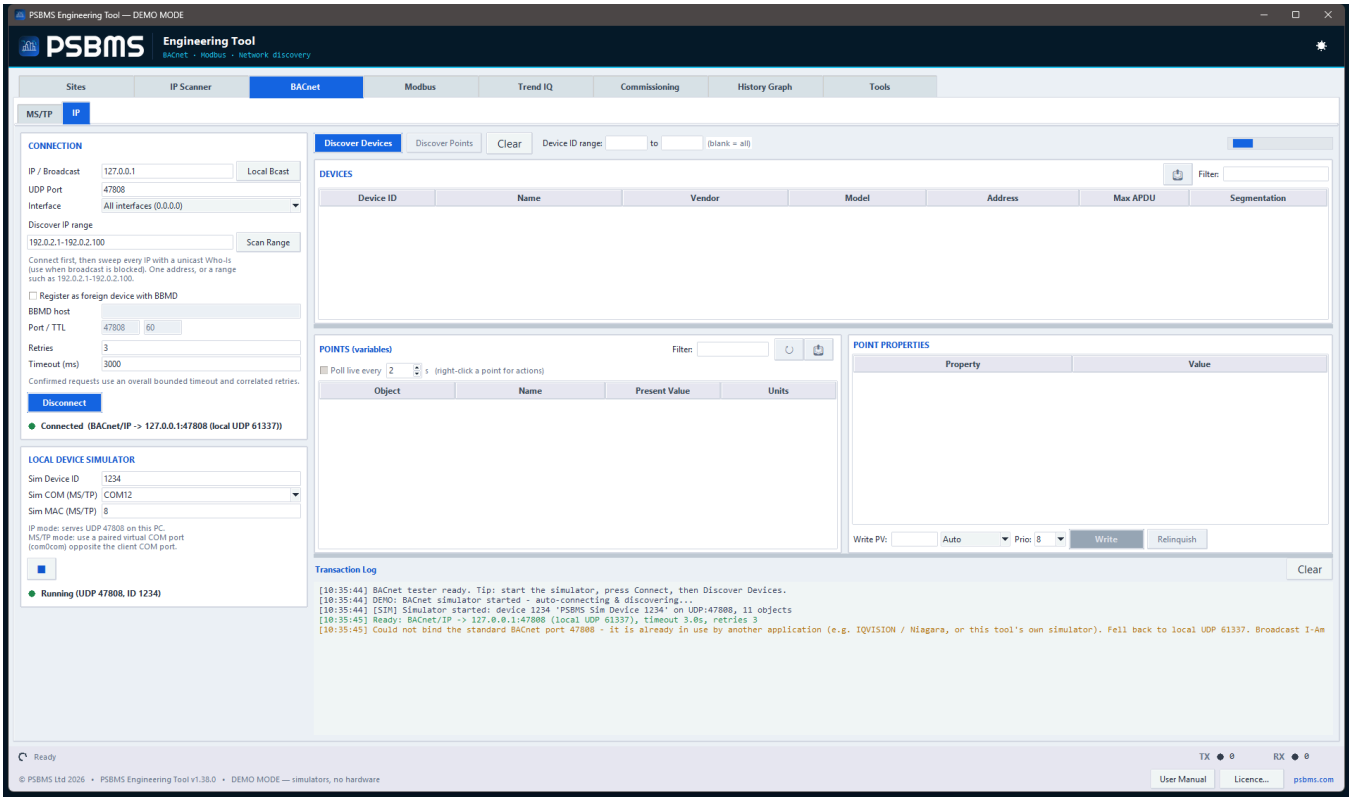
About, and checking for updates. The **About** button in the footer shows the version you are running and your edition, the terms of use, and buttons that open the **third-party notices** and the **data-handling** notice installed with it. **Copy details** puts the version, edition and Windows version on the clipboard for a support email — nothing about your sites, and no machine ID (the Licence window copies that when it is wanted).

Check for updates asks tool.psbms.com whether a newer release has been published. It only does so when you press it — the tool never checks by itself, at start-up or on a timer — and the request carries the version you are running and nothing about you, your sites or your controllers. The answer is one of:

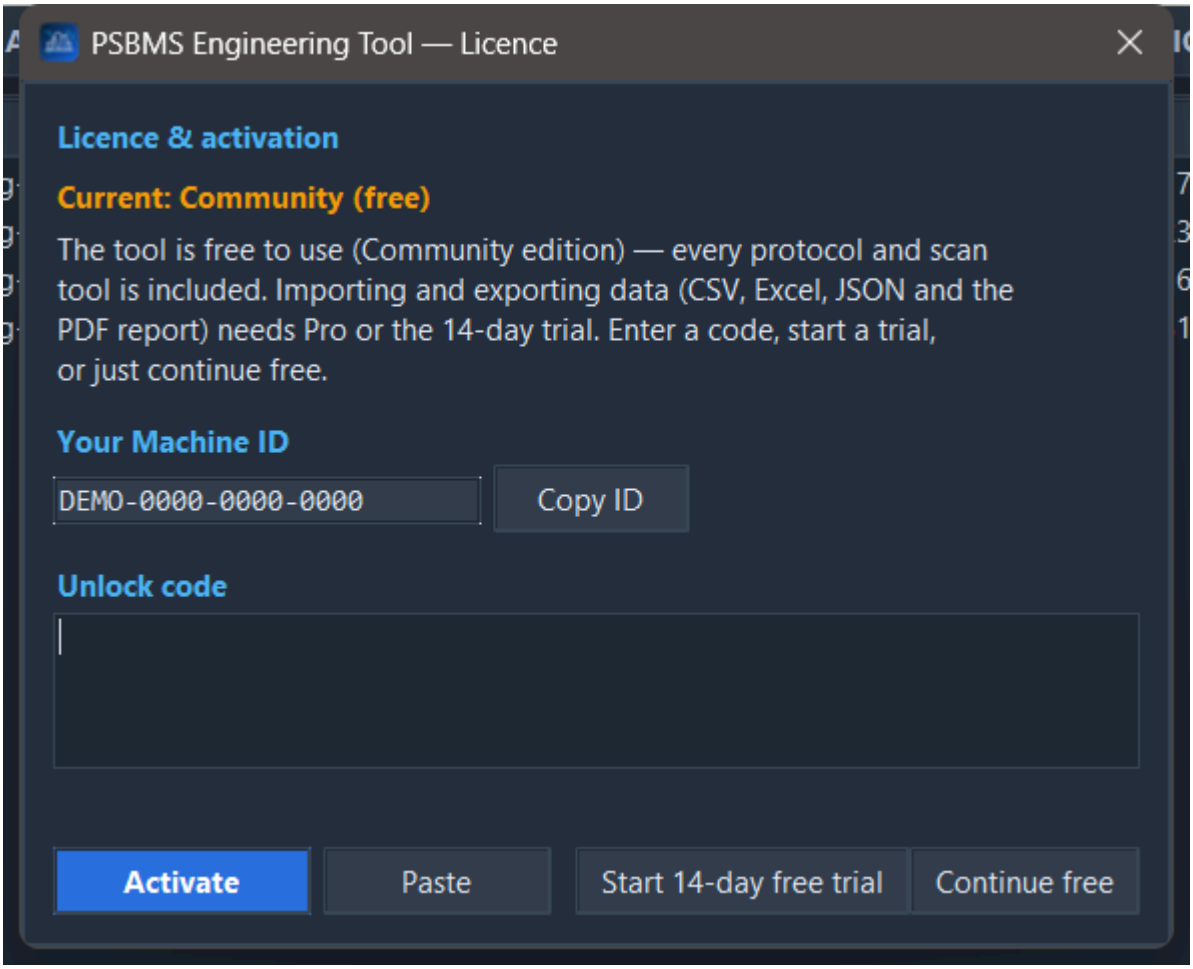
- **Version … is available** — **Get the update** opens the download page. The tool downloads and installs nothing itself: you run the new installer from the website as usual, and it is signed by **PSBMS LIMITED** like every release.
- **You have the latest version.**
- **Couldn't check for updates**, with the reason — on a site network with no internet access this is expected. **Open download page** lets you look in the browser instead — useful where a company network's proxy lets the browser out but not other programs.

Appearance. The app ships in a **dark** theme; a sun/moon toggle on the brand bar (top-right) switches to a **light** theme, and your choice is remembered — logs and the trend chart follow the theme too. Many button rows are compact **icons with hover tooltips** — hover any button to see its label. The window remembers its **last position, size and maximized state** and restores them next launch. Each tab has a **drag handle above the Transaction Log** — drag it to make the log taller or shorter against the content above it; the Modbus tab adds a divider **between Read Results and Discovered Devices**, and the BACnet tab one **between the Devices list and the Points area**. A **progress bar** appears while a scan or discovery runs. Scrollbars **appear only when a table or log overflows**, so empty views stay clean. Every result table shades **alternate rows** a light blue (a muted slate in the dark theme) so a long list reads across without losing the row; a row coloured for a reason — an alarm, a point not in normal control — keeps its own colour.

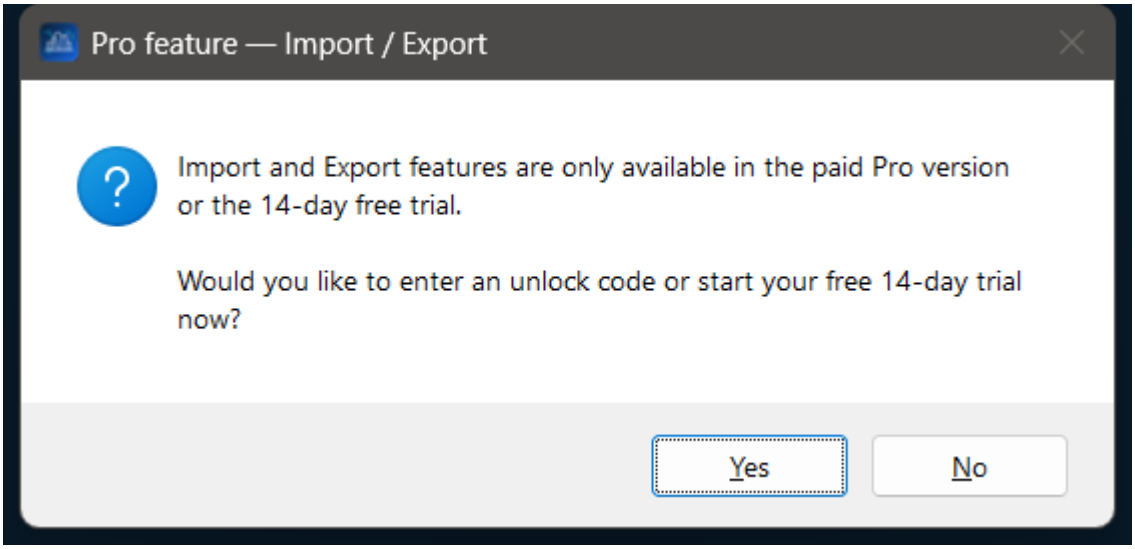




The same tab in dark and light themes — toggle with the sun/moon on the brand bar.

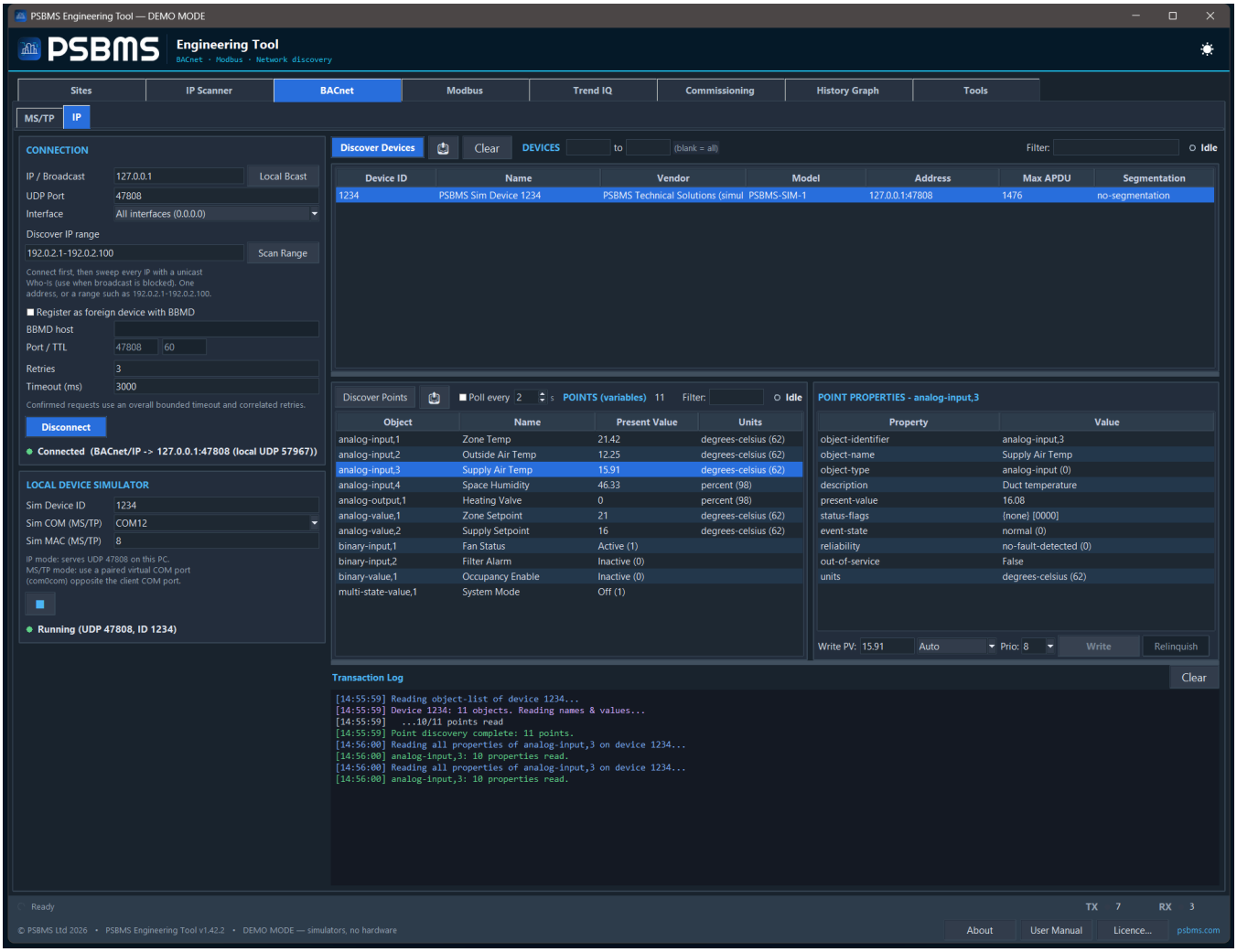


Synthetic documentation example: the Licence window with an unmistakably fake Machine ID and non-functional demo unlock code.



Community keeps every tool; importing/exporting a data file prompts to activate Pro or start the 14-day trial.

2. BACnet — the MS/TP and IP pages



Test BACnet devices, browse points, and write values. A **built-in simulator** lets you try everything with no hardware.

Connect

1. Choose **BACnet/IP** or **MS/TP (serial)**.
2. **IP:** set **IP / Broadcast** — the device's IP for a single device, or press **Local Bcast** to fill your subnet broadcast (e.g. `192.0.2.255`) to find everything. `127.0.0.1` only searches your own PC. **MS/TP:** set COM port, baud and this tool's MAC ID.
3. **Interface** (IP only) — on a laptop with more than one network (e.g. Wi-Fi *and* a USB-Ethernet on the panel network), pick which adapter to send from. Leave it on *All interfaces* unless broadcasts are going out of the wrong NIC.
4. **Register as foreign device with BBMD** (IP only) — tick it to reach a network whose broadcasts only travel through a BBMD, and give **BBMD host** the BBMD's own address, with its port and a TTL. That box takes **one address**: a range belongs in *Discover IP range* and a subnet in *IP / Broadcast*. Anything else is refused by name — with the cursor put in the box, so the next thing you type replaces it — rather than sent to the socket to come back as "getaddrinfo failed".

   A registration carries broadcasts onto a network this laptop cannot broadcast onto, so it is only *needed* for a broadcast. If that box holds something that is not an address, a broadcast is refused — there is no other way through — while **one address is still read**, with the log saying no registration was made. A

host handed over from the **IP Scanner** is likewise read directly: the scanner found it from this laptop. Your setting is left as you had it in every case.

5. Set retries/timeout if needed and press the ▶ **Connect** button (a single toggle — it becomes ■ to disconnect once connected).

> **Not finding real devices?** Three things silence a device that is plainly there. The tool must own UDP port 47808 to hear broadcast replies: if another BACnet program (e.g. IQVISION/Niagara) or this tool's own simulator is using it, the log shows an amber warning — close the other program, or point at the device's IP directly, which needs no broadcast. A **Device ID range** in the discovery bar is carried inside the Who-Is itself, so anything numbered outside it will not answer; clear both boxes. And a broadcast does not cross an *IP* router: devices on another subnet never hear **Discover Devices** at all — use **Scan Range** on that subnet, type that subnet's own broadcast address, or go through a BBMD. A *BACnet* router is different, and is crossed: the Who-Is goes out as a global broadcast, which a BACnet router — a metering gateway serving its meters as virtual devices, say — carries onto the networks behind it. The **NETWORK** box chooses that scope.

> **A subnet in the IP / Broadcast box** (`192.0.2.0/24`) is read as its broadcast address (`192.0.2.255`) and the log says so, rather than failing as an unresolvable host name.

Try it with the simulator

In the **Local Device Simulator** card press **Start Simulator** (IP mode serves UDP 47808 on this PC), keep IP `127.0.0.1`, **Connect**, then **Discover Devices**. A simulated AHU controller (device 1234) appears.

Discover & browse

Two rows, laid out as the IP Scanner's is. Over the devices table: **Discover Devices**, **⭳ Export**, **Clear**, the **DEVICES** heading with the device ID range beside it and how many devices were found - `11 found`, or `4 of 11 shown` while the filter narrows the table - then the search's progress, the filter and whether a search is running. At the top of the points card: **Discover Points**, **⭳ Export**, **Poll every N s**, the **POINTS (variables)** heading with how many there are, then that table's filter and state. On a window too narrow for a row, the filter drops below the buttons rather than being cut off.

- **Discover Devices** lists devices. The **DEVICES** range limits the search to a band of device instance numbers, and blank means all.

  > **The range is a filter, not a search.** A Who-Is carries the range with it, and a device numbered outside it is required by the protocol to stay silent - so a range left in the boxes from an earlier job turns eleven devices into two with nothing on screen to say why. The tool now says it: devices found inside a range are reported with the range that was in force, and a range that finds nothing is asked again with no limit. Clear both boxes to ask every device. The range belongs to **Discover Devices**; **Scan Range** ignores it and says so in the log, because a sweep already names every address it asks.

- **The NETWORK box** chooses which BACnet network the Who-Is asks. Blank is **global**: the question crosses every BACnet router, so devices served behind one - a metering gateway's virtual meter devices, an MS/TP trunk behind a router device - answer too, and the devices table's **Network** column shows which network each came from. A number (1-65534) asks that one remote network through its router. The word `local` keeps the question on this laptop's own network - the behaviour of every version before 1.47. A global ask that hears nothing is automatically asked again plainly, because some devices

mishandle an NPDU that carries a destination. Each discovery log line repeats the scope in force, so a number left in the box cannot silently narrow a later scan; and when one Device ID answers from more than one place - which a router-crossing scan can make audible for the first time - the log names both answers, because the table can show only one. Devices found behind a router are read, polled and written through it with no further steps.

- **A device that answers no Who-Is at all** - one that listens only for broadcasts, or sits behind a BBMD - is read directly. Point the tab at its address, press **Discover Devices**, and if nothing answers the tool reads the Device object at the wildcard instance (4194303), which every BACnet device recognises. The device appears with its instance, name, vendor and model, and **Discover Points** works on it as usual.
- **A device that answers but will not say who it is.** Some refuse that wildcard read - a Priva Compri HX answers it, and any Device ID but its own, with *service-request-denied* - and announce themselves only by broadcast on their own subnet, which does not cross a router or a VPN. The log says so, rather than suggesting the device is off, and gives the two ways in:
  - **Through the BBMD on its subnet.** Put the BBMD in **BBMD host**, tick **Register as foreign device**, press **Connect** and then **Discover Devices**: the BBMD passes the device's I-Am on to you. The IP Scanner marks BBMDs in its Open Ports column, and when it has found one on the device's subnet the log names it. If the registration goes unanswered, the log says whether **BBMD host** holds the device itself - a BBMD is a different host on its subnet - or a host that is probably not a BBMD.
  - **By its Device ID.** Put the same number in both **DEVICES** boxes - `9` to `9` - and press **Discover Devices**. At a single address the tool then reads that Device ID directly, which needs no I-Am and no BBMD.
- **A device that serves no points** - whose object list holds only its own Device object - and that was reached at its own address is asked Who-Is-Router-To-Network before anything is said about it. One that answers for itself is a **BACnet router**: the log names the networks behind it and says to press **Discover Devices** with the NETWORK box blank, because that is where its points live. One that does not answer - or a point-less device that itself sits behind a router - is reported as before: it may be a BACnet *client*, and the POINTS count reads *none served*, rather than an empty table that looks like a read that failed.
- **Discover an IP range** (BACnet/IP): in the Connection card type one address or a range into **Discover IP range** (e.g. `192.0.2.1–192.0.2.100`, or `192.0.2.1–100` for short) and press **Scan Range**. Use it when broadcast is blocked or filtered, and for any subnet other than this laptop's own. (Connect first; max 1024 addresses.) Every range box in the tool takes the same spelling.

  The sweep asks twice, so it finds what the IP Scanner finds. First a unicast Who-Is to every address, then — for the addresses that stayed silent — a direct read of the Device object at the wildcard instance, which is the same question the IP Scanner puts to port 47808. Devices found that second way are logged as a *Device read* rather than an I-Am, an address that answers BACnet but will not identify itself is named with its reason, and the **Device ID range** is not applied to either pass. The second pass adds a few seconds per batch of 64 addresses; **Disconnect** stops it where it stands.
- Select a device and press **Discover Points** to list its objects with live **Present Value** and **Units**. Binary points read *Active/Inactive* and multi-state points show their *state text*.
- Click a point to see **all its properties** on the right.
- **Filter the devices** with the box beside the DEVICES heading — matches device ID / name / vendor / model / address / network.

- **Filter the points** with the box at the end of the points row — matches object / name / present value / units (the count beside the heading then reads "X of Y"). Handy on large controllers with hundreds of points. Both filters work for BACnet/IP and MS/TP.
- **Clear** empties the tables before a fresh search.

Live values

- **Right-click a point** → **Refresh present value** re-reads them once.
- Tick **Poll every N s** to keep them updating — ideal while you force a point and watch it move.

Right-click a point

- **Read all properties**, **Refresh present value**
- **View priority array…** (commandable points) — a 16-row window; the green highlighted level is the one in control. `–` = relinquished.
- **Copy object id / name**

Write a value

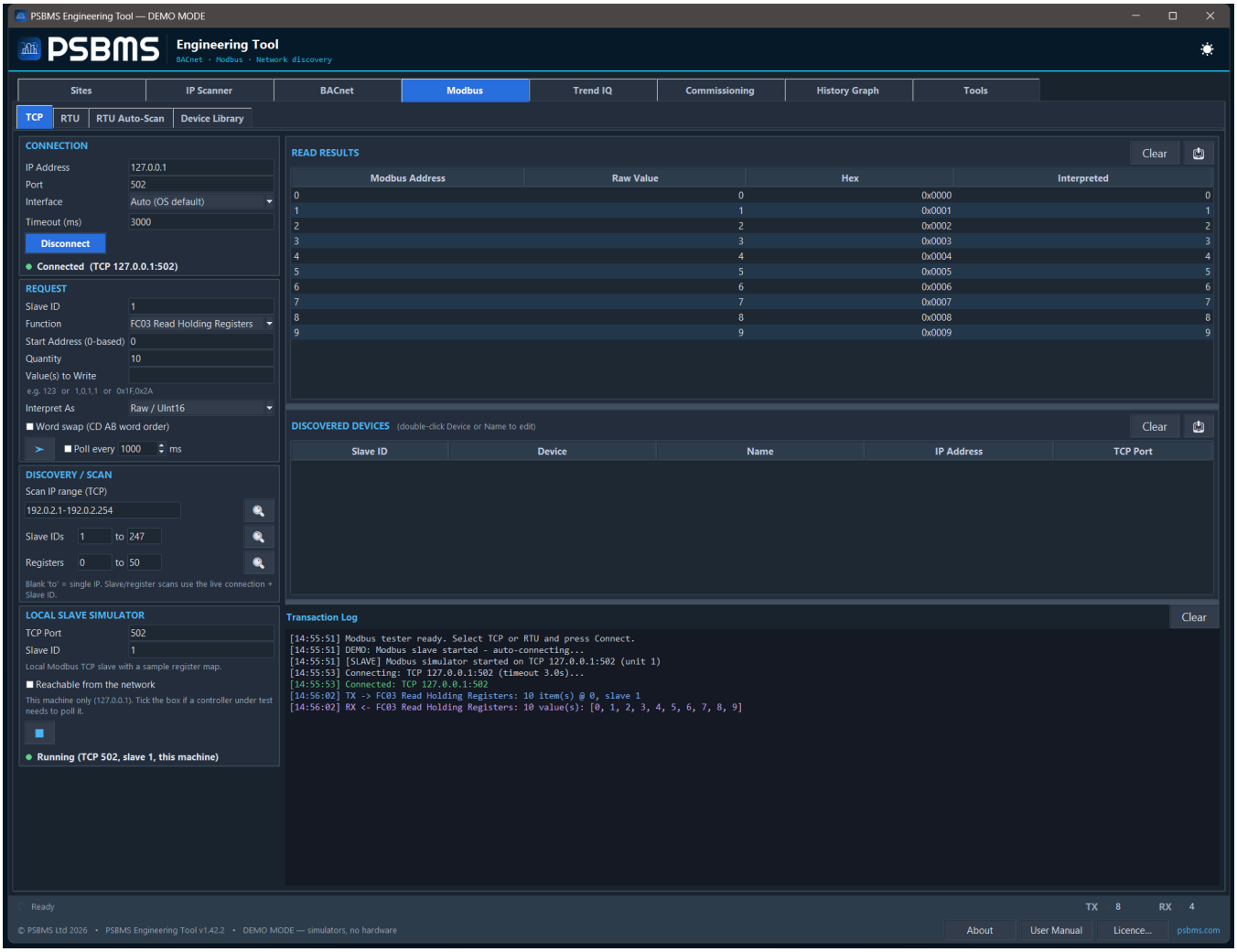
Select a commandable point (setpoint, valve, mode…). The **Write PV** box is pre-filled with the current value. Choose a value type (Auto is fine) and a **Priority** (1–16), then **Write**. Choose **Null (relinquish)** to release a priority level. Before commanding plant, enter a valid **Project ID**, **Job ID** and **Engineer** on the Commissioning page. A commandable BACnet override is recorded durably before it is queued, kept in the recovery audit after the device acknowledges it, and can only be relinquished against the matching controller, object and priority. If the audit journal or context is unavailable, the command is blocked before network dispatch.

Export

⭳ Export over the **Devices** table and **⭳ Export** over the **Points** table each open a menu naming the same three formats every Export icon offers: an **Excel workbook**, a **branded PDF** or a **CSV**. Both records say how the tab was talking to the network — BACnet/IP with the interface and port, or the MS/TP port with its baud rate and MAC — and the points record names the device they were read from. Point discovery that was cancelled or came back partial is refused rather than filed half-read.

---

3. Modbus — the TCP and RTU pages



A slave-ID scan of a gateway — each hit lands in the Discovered Devices table (Device + Name are editable), with the scan progress bar under the buttons.

A full Modbus master with a built-in slave simulator and discovery scans.

Connect

- **Modbus TCP:** IP + port (default `127.0.0.1:502`). **Interface** picks the source NIC on a multi-homed laptop (default *Auto* lets Windows route); it also applies to the IP-range scan.
- **Modbus RTU:** COM port, baud, data bits, parity, stop bits (↻ refreshes the port list).
- Press the ▶ **Connect** button. It's a single toggle — once connected it becomes ■ to disconnect (the same style as the LOCAL SLAVE SIMULATOR Start/Stop button).

Read / write

1. Set **Slave ID**.
2. Choose a **Function** (FC01–06, 15, 16).
3. Set **Start Address** (0-based) and **Quantity**.
4. For writes, put value(s) in **Value(s) to Write** — a single value, or a comma list (`1,0,1,1`); hex like `0x1F` is accepted.
5. **Send Request.** Results appear in **Read Results**; **Clear** empties the table before the next read.

- **Interpret As** re-reads registers as Int16/UInt16, 32-bit (Int32/UInt32/Float32) or 64-bit (Int64/UInt64/Float64); tick **Word swap** to reverse the word order (e.g. CD-AB).
- Tick **Poll every … ms** to read continuously.

Try it with the simulator

In **Local Slave Simulator** set a TCP port + unit and press **Start Slave**. Then connect TCP to `127.0.0.1` on that port and read holding registers.

The slave listens on this machine only unless you tick **Reachable from the network**. Tick it when a controller under test has to poll the simulator over the LAN, and untick it afterwards — the simulator accepts writes from anyone who can reach it and has no password. The setting is not remembered between sessions, so the slave always starts local.

Discovery / Scan

- **Scan IP range (TCP)** - type one address or a range (`192.0.2.1–192.0.2.254`, or `192.0.2.1–254`) and sweep it for Modbus devices at *different IP addresses* (no connection needed).
- **Scan Slave ID range** — find which slave addresses answer on the link. The log shows the value read from each unit.
- **Scan Register range** — read a block on the selected Slave ID to map which addresses hold data (great for undocumented devices). Results fill the table.

Discovered Devices table. The IP-range and Slave-ID scans list each device they find in the **Discovered Devices** table (Slave ID · Device · Name · IP Address · TCP Port), so you don't have to read them out of the log, and the heading counts them (`2 found`). As on the IP Scanner, **⤒ Export** and **Clear** come first on each table's heading row - Read Results, Discovered Devices, and Found devices on RTU Auto-Scan alike. **Double-click the Device or Name** cell to label a device (e.g. Device = *Demo AHU Controller*, Name = *Demo AHU*), use **Clear** to empty the list, and **⤒ Export** to save it as an **Excel workbook**, a **branded PDF** or a **CSV** (`Slave ID, Device, Name, IP Address, TCP Port`), headed with the link it was found on. Rows de-duplicate on (slave, IP, port), so re-running a scan won't create duplicates or lose the labels you typed. (Slave-ID scans on an RTU link leave IP/Port blank.) A **progress bar** under the scan buttons fills while a scan runs.

Scanning a Modbus/TCP gateway for its slaves

A gateway hosts many serial (RTU) slaves behind **one IP address**, addressed by **Slave ID** — not by IP. So:

1. Set **Modbus TCP**, enter the **gateway's IP** and port (usually 502), and press **Connect**. (Or right-click the gateway in the IP Scanner and choose "test in Modbus TCP tab" to fill this in for you.)
2. In **Discovery / Scan**, set the **Slave ID range** (e.g. 1–247) and press **Scan Units**.
3. Read the log. Every slave ID listed **answered** — that is all a scan can tell you. A value of **0** is a successful read of a register that happens to hold zero; it is an answer like any other, not an empty address. An *illegal function* or *illegal data address* exception is also an answer: something is there, it just does not have that register.

   The two *gateway* exceptions are the exception. **Gateway path unavailable** and **Gateway target device failed to respond** come from the gateway itself, not from the slave you probed, so they prove nothing about that slave — the scan does not list them as discovered.

   To tell a live slave from an empty gateway slot, set the Request card's **Start Address** to a register you know that slave holds, then scan again: an empty slot will stop answering, a live slave will return its value.

Auto-detect serial settings (Auto-Scan)

Handed an unknown RS-485 device and don't know its baud rate, parity or address? Open the **Modbus → RTU Auto-Scan** page (§5), which brute-forces the serial settings and device ID until a device answers.

---

4. Commissioning Suite

*(the tab is labelled **Commissioning**)*

The **Commissioning Suite** brings schedule verification, network-health review, BACnet operations and field analysis together in one workspace. It is designed for repeatable, evidence-backed site work and does not replace the live BACnet or Modbus connection pages. Connect and discover there first, then return to the suite to analyse the cached observations.

> **Safety and qualification boundary.** Commissioning verification, Network Health, controller logs, events, clock audit and Field Tools are read-only. A COV subscription creates only a notification relationship; it does not write a point value. Point/schedule writes and alarm acknowledgement are not provided here. Controller-dependent BACnet Operations results must be qualified against the actual controller, firmware and network before they are used as project acceptance evidence. Use a controlled test window and the normal site change process.

Commissioning

1. Enter the **Project**, **Site**, **Job** and engineer details, then save the workspace. The project journal stores restart-safe commissioning state.
2. Import the design point schedule as strict **CSV** or **XLSX**. Ambiguous columns, duplicate identifiers, malformed values and non-finite tolerances fail closed instead of being guessed.
3. Refresh the relevant BACnet or Modbus data on its connection page, then use **Acquire live points**. Cached samples must be complete, time-stamped and no more than two minutes old. Stale, partial, future-dated or unverified values are excluded from passing evidence.
4. Match the design schedule to live points and run read-only verification. The result distinguishes passed, failed, missing and ambiguous points and records the evidence source and time.
5. Capture **As Found**, perform the authorised site work, refresh the protocol data, and capture **As Left**. The suite requires a newer observation for the second phase; it will not certify the same cached sample twice. Compare the snapshots to show substantive value/configuration/quality changes while retaining acquisition timestamps as provenance.
6. Export deterministic JSON/CSV evidence or a branded PDF report. Exports use the application's licence/export policy; trial outputs retain the normal watermark.

Network Health

Analyse current state correlates normalized BACnet/Modbus transaction events, flags retries, latency, timeouts, exceptions and partial operations, and analyses BACnet identity/IP/network conflicts when the necessary observations are present. You can save a baseline, compare it with the current state, or import normalized transaction JSON captured by an approved adapter. Export options include deterministic JSON, formula-safe CSV and a redacted support ZIP.

BACnet Operations

Connect and discover controllers on the **BACnet MS/TP & IP** page first, open **Commissioning** → **BACnet Operations**, and press **Refresh targets**.

- **COV** — subscribe to an object or property, choose the process ID, confirmed or unconfirmed delivery and lifetime. A lifetime of `0` requests an indefinite subscription; finite subscriptions renew automatically. Cancel using the same device/object/property/process identity. Notifications are bounded in the UI and retain UTC time, invoke/process ID, array index, priority and quality data.
- **Controller Logs** — retrieve Trend Log (object type 20), Event Log (25) and Trend Log Multiple (27) with bounded pagination, duration, byte and record limits. Partial/rolling-log results are labelled incomplete and exported with controller and object identity.
- **Events & Clock** — read active event summaries and audit controller clock drift. This view does not acknowledge alarms or set the controller clock.

Use **Export CSV** only after the displayed target and completion state have been checked. A failed or newly queued request clears the previous result so evidence from one controller cannot be mistaken for another.

Field Tools

The Field Tools page provides protocol-neutral commissioning analyses:

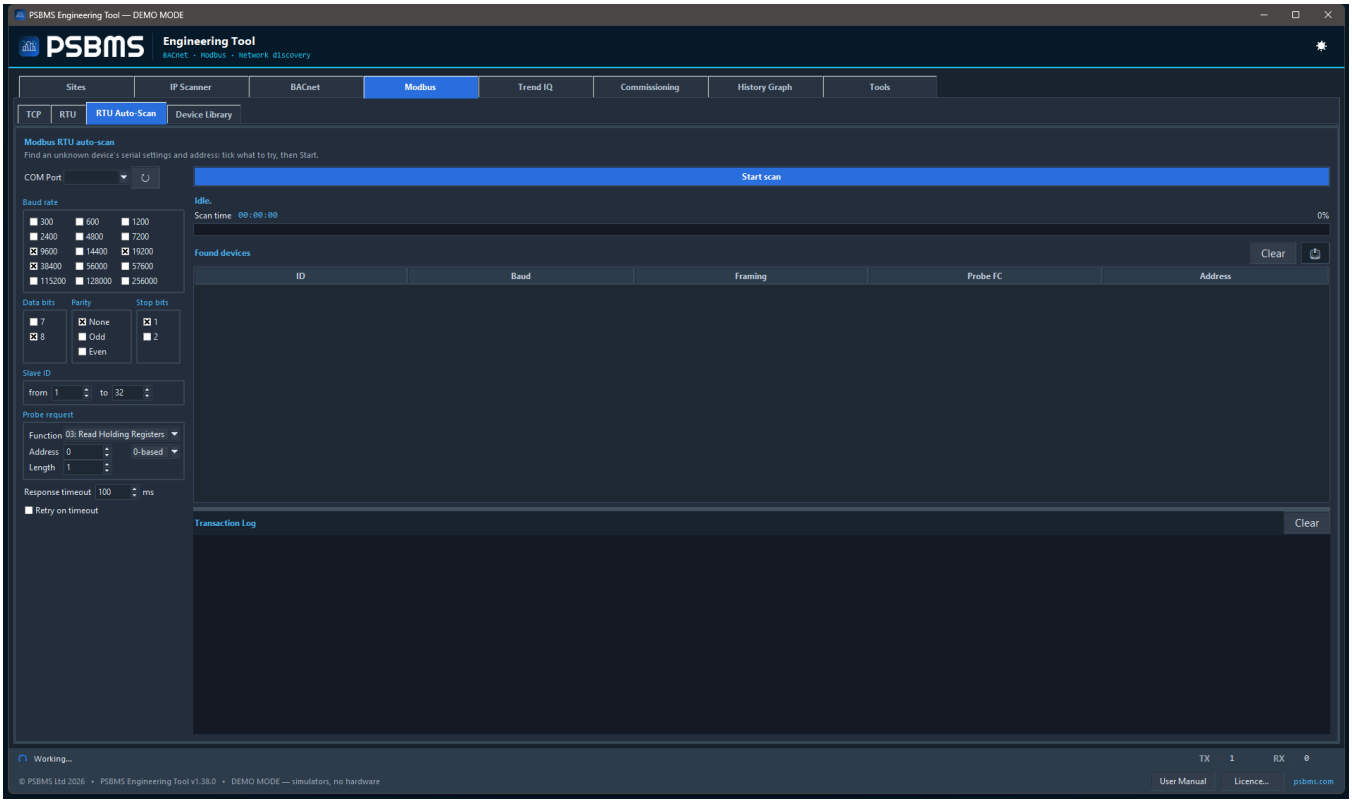
- calibration error, offset and pass/fail checks;
- actuator/VFD stroke, feedback and command checks;
- control-loop response metrics.

Inputs and outputs are explicit engineering values; the page does not command a controller. CSV exports are atomic, formula-safe and covered by the same licence/watermark policy as other application exports.

Loop Analysis works from supplied observed samples. The separate Tools PID Simulator generates synthetic educational response data only; its traces cannot be used in Commissioning Suite evidence exports.

---

5. Modbus — the RTU Auto-Scan page



Brute-forcing an unknown RTU device's serial settings — the current combination, timer and progress bar update as it scans.

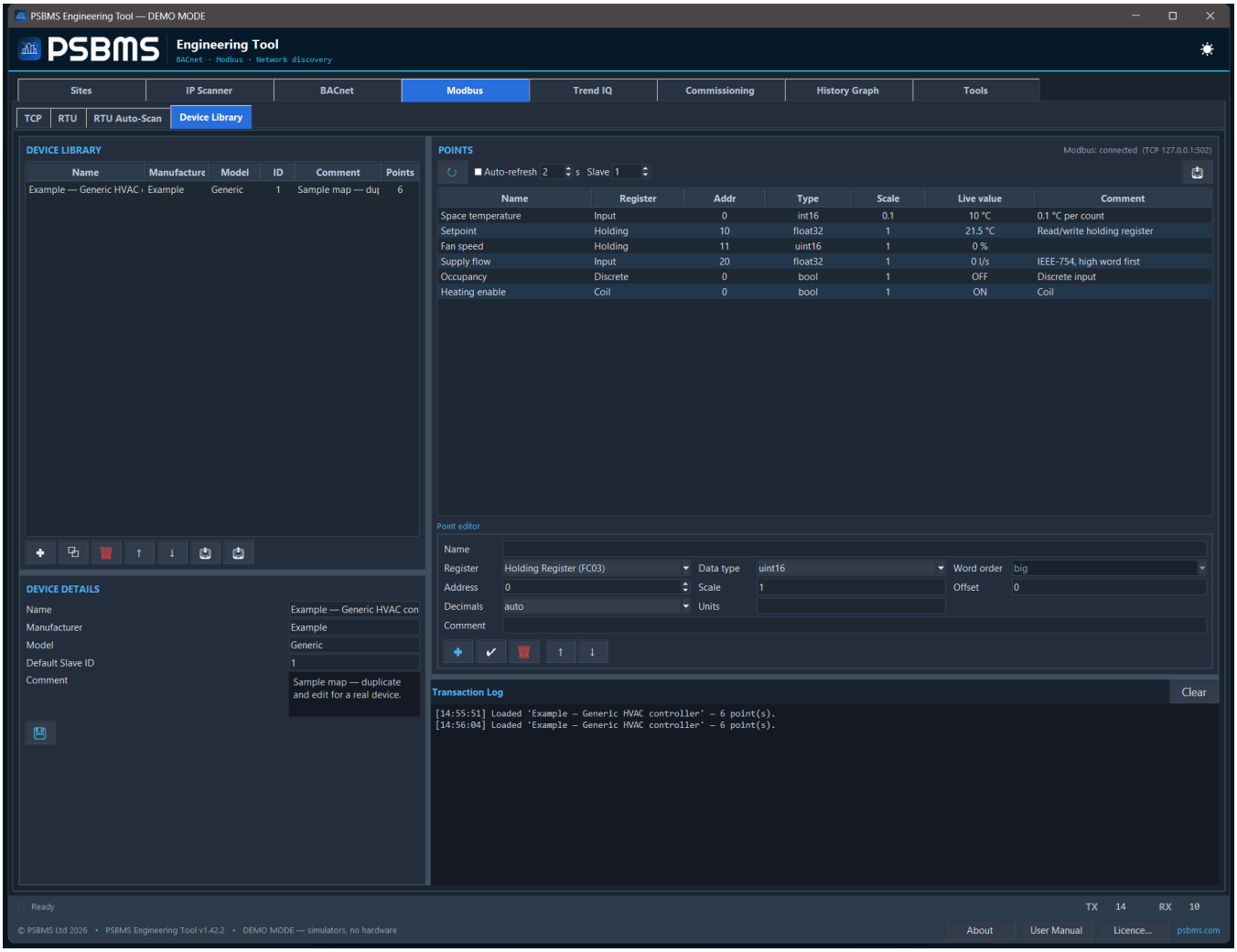
Find an unknown RTU device's serial settings and address by trying every combination until one answers.

1. Pick the **COM Port**.
2. **Tick what to try** — the **baud rates**, **data bits**, **parity** and **stop bits** to sweep (a quick, sensible set is pre-ticked; add more to widen the search), and the **Device ID** range (from / to).
3. Set the **Probe request** it knocks with (function — Read Holding Registers by default — address, 0/1-based, length), the **Response timeout** in milliseconds, and **Retry on timeout** if the link is marginal.
4. Press **Start scan**. It shows the settings it's **trying now**, the **scan time**, a **0–100 % progress bar**, and lists every device it **finds** in the **Found devices** table — any valid reply (real data *or* a Modbus exception) means a device is there at those settings, so you get its working baud/parity and address. **Stop** ends it early.
5. **⭳ Export** saves the found-devices table (ID, baud, framing, probe function, address) as an **Excel workbook**, a **branded PDF** or a **CSV**. The record names the port and the sweep behind it — every baud rate, framing and slave ID tried, and the probe used — so two devices found says as much as the hundreds of combinations that answered nothing. **Clear** empties the table (and the export with it) before another scan.

> The scan tries every combination, so it can take a while — a full sweep of all baud rates × parities × 247 IDs is thousands of probes. Start narrow (a few likely bauds, one parity) and widen only if nothing answers. The footer TX/RX LEDs flash on every frame; click them to watch the raw packets (see §12).

---

6. Modbus — the Device Library page



Synthetic documentation example: demo devices and register values beside the Point Editor; the only endpoint shown is in the RFC 5737 documentation range.

A reusable database of Modbus register maps. Build a device's point table once — names, register kinds, addresses, data types, scaling and comments — store it, and reuse it on every visit. When you're connected to a slave, it reads every point and shows the value **beside its label**, turning a wall of raw registers into a named, commented point list.

Build a device

1. **New** creates a device; give it a **Name**, and optionally **Manufacturer**, **Model**, default **Slave ID** and a **Comment**. **Save** stores it in the library on the left (files live in `%APPDATA%\PSBMS Engineering Tool\modbus_devices\`). The library lists every device in a table — **Name, Manufacturer, Model, ID, Comment, Points** — and the ↑ ↓ **Move device up / down** buttons reorder it; that order is remembered between sessions.
2. Add points with the **Point editor**: **Name**, **Register** (Holding / Input / Coil / Discrete), **Address** (0-based), **Data type** (bool, uint16, int16, uint32, int32, float32, **uint64/int64/float64** — 4 registers, for energy totalisers etc.), **Word order** (for 32/64-bit values, *big* = most-significant word first), **Scale** and **Offset** (engineering value = raw × scale + offset), **Decimals**, **Units** and a **Comment**. **Add** appends it; select a row and the ✓ **update** / 🗑 **remove** / ↑ ↓ move buttons edit it. Remember to 💾 **Save**.

3. ⧉ **Duplicate** clones the open device (handy for a family of similar units); 🗑 **Delete** removes one from the library. (Hover any icon button for its label.)

> **Decimals** controls how the live value is shown: **auto** prints whole numbers as integers and trims trailing zeros (and never uses scientific notation — a large run-hours count reads `262446`, not `2.62e+05`); set **0–6** to fix the number of decimal places.

Import / export

- ↧ **Import CSV** creates a device from a points CSV (columns: Name, Register, Address, DataType, WordOrder, Scale, Offset, Decimals, Units, Comment — register may be a name like *Holding Register* or a function-code number).
- ⭱ **Export** opens a menu with three choices: a **Device list** summary CSV (one row per device — Name, Manufacturer, Model, Slave ID, Comment, Points), **This device's registers** (the open device's point map), or **Every device → folder** (one re-importable CSV per device). Share maps with colleagues or keep them under version control. These three stay CSV on purpose: a register map is what ↧ **Import CSV** reads back, and a workbook is not.

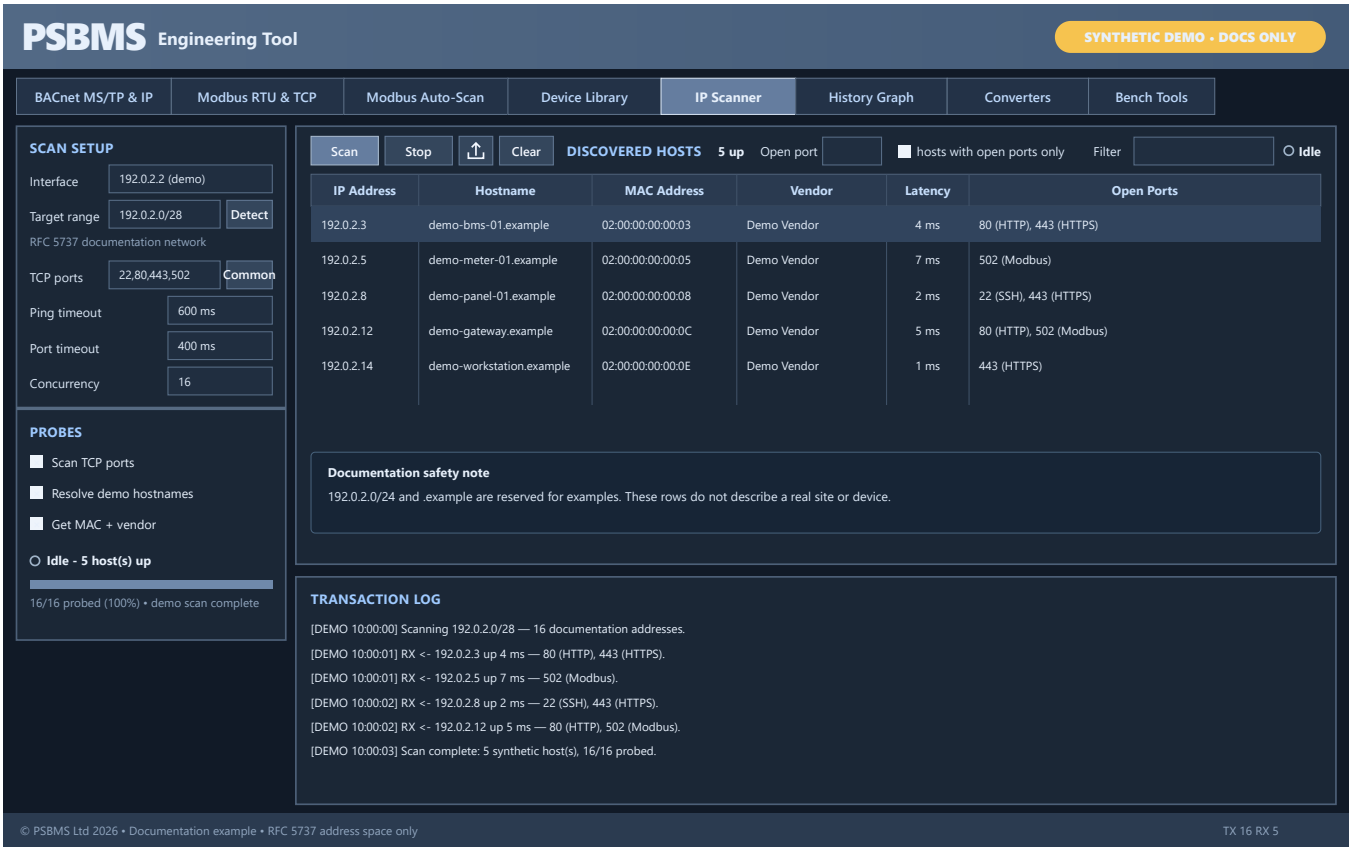
Read live values

1. Connect to the slave on the **Modbus → TCP** or **RTU** page (§3) — the Device Library shares that connection, so this works for both TCP and RTU.
2. Open the device, set the **Slave** to read (defaults to the device's), and press ⟳ **Read live** — the **Live value** column fills with each point's scaled value **and its units** (e.g. `23.5 °C`). The points table no longer has a separate Units column — units are still set in the Point Editor, exported to CSV, and shown next to each value. Tick **Auto-refresh** to re-read every few seconds. While it reads, the footer **RX/TX LEDs** flash and the packet inspector captures the traffic, exactly as on the Modbus tab.
3. Points that couldn't be read (wrong address, not connected) show **—** and are greyed. Bits show **ON/OFF**; registers show the engineering value.
4. ⭱ **Export live** writes the current readings (name, register, address, value, units, raw and comment) as an **Excel workbook**, a **branded PDF** or a **CSV** — a snapshot of what's on screen, headed with the device, its slave ID and how many of its points answered.
5. **Right-click** one or more points and choose **Add to History Graph** to chart them live — the channels carry their data type, word order and scale.

> The live read uses the Modbus tab's own connection, so you never open a second COM port — essential for RS-485, where the port can only have one master.

---

7. IP Scanner



Synthetic documentation example: fictional `.example` hosts in `192.0.2.0/24`, with demo MAC vendors, ports, and progress.

Discover and map devices on the local network.

1. Press **Detect** to fill your local subnet, or type a range — CIDR (`192.0.2.0/24`), a range (`.10-.60`), or single IPs comma-separated.
2. Set **TCP Ports** — single ports and/or ranges (`80, 443, 8000-8100`); press **Common** for a sensible default set. **47808** is the exception: BACnet/IP listens on UDP, so a TCP connect would never find it. The scanner asks it in BACnet instead — one read of the device's own identifier, sent to that host alone — and any BACnet reply marks the port open.
3. Tick the probes you want (ports / hostname via DNS + mDNS, with a NetBIOS + LLMNR fallback / MAC + vendor) and press **Scan**, at the left of the row above the results — beside **Stop**, **⭱ Export** and **Clear**. The status at the right-hand end of that row says whether a scan is running; the PROBES panel shows its progress.
4. Results stream in: IP, hostname, MAC, vendor, latency, open ports. Click a column heading to sort, and **⭱ Export** to save the scan.

The **Vendor** column is resolved from the device's MAC against a bundled IEEE manufacturer database (~52,600 prefixes) — it works fully **offline**, so manufacturers appear even on isolated plant networks. Randomised/private MACs (common on phones) have no manufacturer and are shown blank.

MAC addresses come from the network you are plugged into. A device's MAC never crosses a router, and a VPN is one. On your own network segment the scanner reads each device's MAC from Windows' ARP table. Through a VPN or a router that table holds the tunnel's or the router's own address for every host — GlobalProtect answers for all of them as `02:50:41:00:00:02` — so there the scanner leaves MAC and Vendor blank, and the log says which adapter answered for them. It then asks the hosts that can report their

own MAC, which works through a VPN: a host with a web or BACnet port open is sent Trend's discovery request, whose reply from an IQ3 or IQ4 carries its MAC, and a Windows host with file sharing open is asked over NetBIOS. An IQ5 answers that request only by broadcast, which a VPN does not carry. For the rest, scan from a laptop on the site network, or ask the network team for the switch's MAC address table.

Export the scan with **⭱ Export**, which opens a menu with the same three choices a Trend IQ scan offers: an **Excel workbook**, a **branded PDF** (landscape — open ports make a wide table), or a **CSV**. Pick one and the save dialog offers that format; a name you type ending in another of the three still wins. Each one carries the same header — the range scanned, the interface used, the TCP ports asked for (or *not scanned*), how many hosts answered, and when it was exported — and holds exactly what is on screen, in the sort order you left it. Export from a filtered screen and the filter is named in the header, so a short list is never mistaken for a short network.

Filter the results from the right-hand end of the same row: enter an **Open port** to show only hosts with that port open, tick **hosts with open ports only**, or type in **Filter** to match IP / hostname / vendor / port text. The count reads "X shown / Y up" while a filter is active. On a narrow window the filters take a row of their own under the buttons.

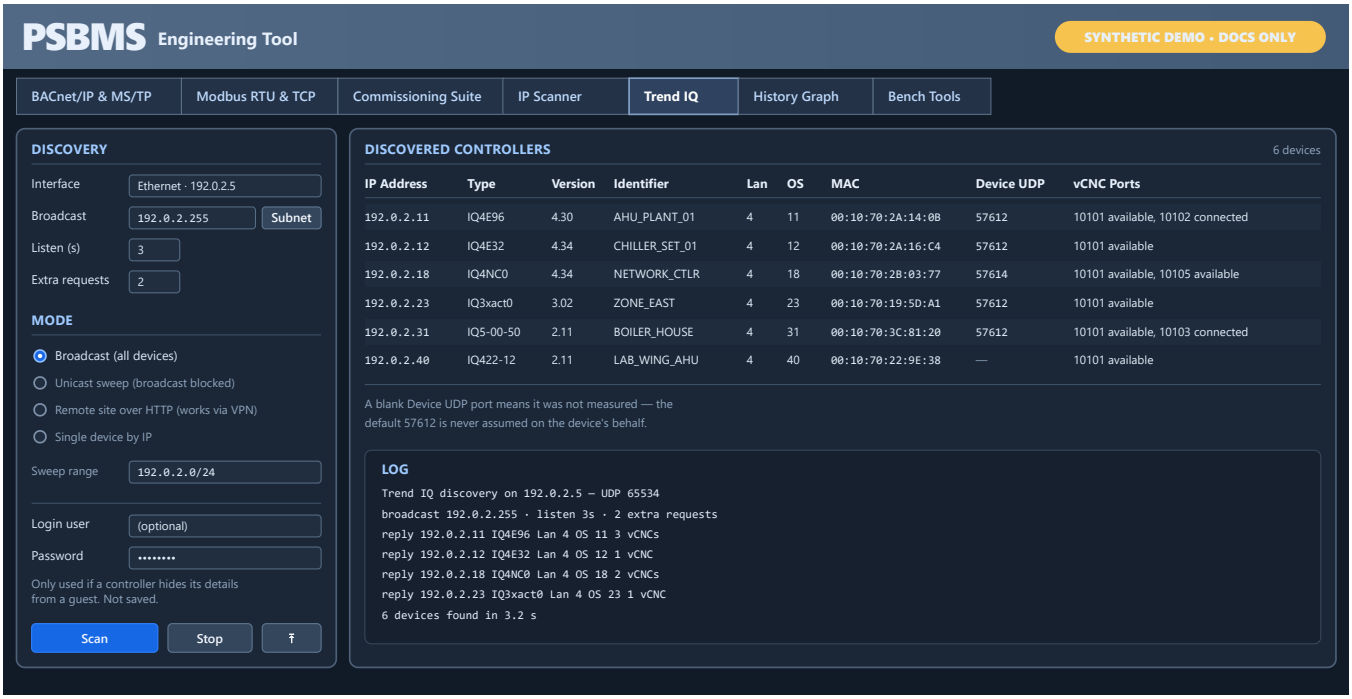
Right-click a host for actions based on its open ports: open its web page (HTTP/HTTPS) in the browser, launch Remote Desktop (RDP), open a file share (SMB), copy an ssh/telnet/VNC command, or **test in the Modbus/BACnet tab**. **Double-click** a host to open its web UI.

Handing a host to the **BACnet/IP** tab does the next two steps for you: it switches to the tab, connects, and reads what is on that address — by Who-Is if the device answers one, otherwise by reading its Device object directly. The device lands in the table ready for **Discover Points**. (The Modbus hand-over still fills the address in for you to connect, because a Modbus read has to be aimed at a register before it means anything.)

A host that is its subnet's **BBMD** shows as `47808 (BACnet/IP, BBMD)` in **Open Ports**: each host that answers BACnet/IP is also asked for its Broadcast Distribution Table, which is a read. The BACnet tab is told, so a device on that subnet that will not identify itself can be pointed at it by name.

---

8. Trend IQ



Synthetic documentation example: fictional Trend controllers on the reserved 192.0.2.0/24 documentation network.

Find IP-networked Trend IQ3, IQ4 and IQ5 controllers and record how a site is addressed — on the local network, or across a VPN.

1. Choose the **Interface** to send from. On a laptop with a VPN up this matters: the wrong adapter finds nothing.
2. Pick a **Mode**:
   - **Broadcast (all devices)** — fastest local discovery; every controller on the subnet answers at once. Raise **Listen (s)** or **Extra requests** on a busy or lossy network.
   - **Unicast sweep (broadcast blocked)** — asks each address in **Sweep range** in turn, for networks where broadcast is filtered.
   - **Remote site over HTTP (works via VPN)** — reads each controller's own web interface instead of broadcasting. This is the mode that reaches another site.
   - **Single device by IP** — ask one controller, given in **Device IP**. With **Login user** filled in, or **Diagnose single device over HTTP** ticked, it reads that controller's web pages, which reach it through a VPN. Otherwise it asks over UDP, which works on the local network only.
3. Press **Scan**. Rows appear as replies arrive, the count beside **Clear** reads "N devices", and the scan's state shows at the right of the toolbar. Click a column heading to sort, or **Stop** to end early.
4. Type in **Filter**, at the right of the toolbar, to narrow the table as you type. It matches any column in any case: an address, a type such as `IQ5`, an identifier, `L11-OS2`, a MAC, GUID or serial. The count then reads "2 shown / 40 devices". Sorting, the right-click menu and the export work on the rows shown; a filtered export names the filter and says how many of the controllers it holds.
5. Use the export button to save the table as a workbook, PDF or CSV.

Reading the columns. Type and **Version** are the controller's hardware designation and firmware — `IQ4E96` and `4.30`. When a controller states its own designation it is shown verbatim, trailing zero and all; when it does not, the type falls back to the family the reply identifies, or to a plain `Trend` when even that is

unavailable. It is never guessed from the firmware number — the same IQ4E ships on 3.33, 3.70 and 4.34, so that guess would be wrong as often as right. **Lan/OS** is its Trend network and outstation address, written `L11-OS22` as every other screen and report writes it, and sorted by number. **GUID** is the site code the controller states on its Address page — read on the same request as the serial, so it costs the scan nothing and is kept when the controller is added to a site. It fills whenever the Address page is read: over HTTP, or with **Identify models over HTTP** ticked. **Device UDP** is the inter-device port the controller reports: 57612 by default, though sites often use a nearby 57xxx to keep networks apart. It is left **blank when it could not be measured**, never filled in with the default, so an exported list can be trusted for housekeeping. **vCNC Ports** lists each configured virtual CNC as port and state — `10101 available, 10102 connected` — so you can see at a glance which slot is free. Slots that are not configured are left out rather than listed as unused.

Each column is measured — its heading, or its widest value where that is wider — and the window's width is shared out in proportion, so the table fills the screen while a short column stays short beside a long one. With too little room the longest columns give way first and no heading is ever cut. The widths follow the scan as results arrive, and the window as it is resized. The Sites list and the module viewer's tables work the same way.

> Discovery itself runs on UDP port **65534** — the port Trend's own tools use. That is not the 57612 inter-device port reported in the results; listening on 57612 for discovery finds nothing.

Credentials are optional. Most controllers describe themselves to a guest. Some hide their identity and address until you log in — fill in **Login user** and **Password** for those. They are sent only to the controller being scanned, are never written to `settings.json` or the log, and are not remembered after the app closes. Tick **Diagnose single device over HTTP** to see a per-request trace when one controller will not answer.

Identify models over HTTP (ticked by default) asks each controller it just found what model it is, and fills in the **Type** column. The discovery reply carries no model field at all, so this is the only way to get one — it reads the controller's own Address page, the same source the site scan uses, and only for the handful just discovered rather than the whole subnet. To a guest most controllers report only the family, or decline entirely and leave the Type as discovery reported it; with **Login user** and **Password** filled in you get the exact designation, `IQ4E64` or `IQ5-00-150`. Untick it to skip the extra HTTP.

That same pass reads each controller's own report of its health from the Address page: corrupt strategy, failed RTC or RAM, low memory or flash, archive failure, sequence table overrun, licence error and the rest. A fault is **named in the log**, per controller, where it cannot be scrolled past — an estate scan is read once, and a fault on a controller nobody opens is the one that goes unnoticed.

Right-click → Site comms map lists every cross-controller link on the site, gathered from the module viewers you have open. A link lives on the controller that owns it, so no single controller can show the site's own wiring.

It reports the two things a per-controller view cannot: a link that is **not getting through**, and a link to a controller that **never answered the scan** — the far end has gone, or was never on that lan, and either way the link is writing into nothing. A missing *reverse* link is not reported: a `To` comms writes into the far end and nothing there has to match it, so flagging one would send you looking for something that is not supposed to exist.

After a scan the log lists the site's own address book — which outstations answered on each lan — built from what replied rather than from a drawing, and the two disagreeing is itself worth knowing.

Anything claiming an address **twice** is reported as an error and its rows are coloured red, until the next scan or **Clear**, and in the export. Two controllers on one Lan/outstation is not a tidy-up job: they fight over every message sent to that address, and the symptom usually shows up on a third controller that is doing nothing wrong. A site scan is the only place it is visible at all. A controller seen twice — once by broadcast, once over HTTP — is one controller, not a conflict.

Right-click a controller to view its modules (below), open its web page in your default browser, add it to a site, copy its IP address or the whole row, or copy its vCNC connection details.

Select several — Ctrl-click or Shift-click — and right-click inside the selection for the three actions that make sense for each of them: **View modules** opens a tab per controller, **Open web pages** opens each one's page, and **Add controllers to site** files them all in one site, asked once. The entries that answer a question about one controller are greyed. A right-click outside the selection starts a new one, as it always has.

⭳ Export opens a menu with three choices — **Excel workbook**, **PDF report** or **CSV** — and the one you pick is what the save dialog offers. (A name you type ending in another of the three still wins, so `.csv` is a CSV whichever choice opened the dialog.) The PDF is landscape, because twelve columns of MACs and serials cannot be squeezed onto a portrait page, and it records what was scanned: the mode, the range or broadcast, how many controllers answered and on which Lans. A controller claiming an address another also claims is flagged in both files, as it is on screen.

Viewing a controller's modules

Right-click a controller → View modules opens that controller in a tab of its own, listing what is configured inside it. Right-clicking the same controller again returns to the tab already open rather than starting a second one.

Close it with **Close tab**, or with a **middle-click on the tab itself**, as you would a browser tab. Chart and trace tabs close the same way, and a module tab takes its own chart tabs with it. The fixed tabs along the top ignore a middle-click.

The tab has three panes:

- **Modules** (left) — the module types this controller has, named as it names them, with the number of items once a type has been read. The list comes from the controller itself, so an IQ3 shows its *Time Zones* and *XNC Interfaces* where an IQ4 shows *Time Schedules* and *Interface Modules*.
- **Items** (middle) — every item of the selected type, in that type's own columns: sensors have Value, Units and Alarm; IO modules have Configured Type, Actual Type, Serial Number and State. Click a heading to sort — value columns sort as numbers, so the outlier comes to the top rather than 100.0 sorting between 1.0 and 2.0 — and type in **Filter** to narrow the list by any column, label or reading alike.

  Digital Inputs and Sensors gain a Type column reading **Internal** or **External** — whether the point is wired to real terminals or driven by strategy inside the controller. Tick **External only** in the toolbar to show just the physical points. The controller words the two modules differently (a digital input says `0: External`, a sensor says `1: Internal Analog`), but the question is the same, so both are shown in the same words; anything the tool cannot place is passed through exactly as the controller wrote it rather than guessed at.

  External only works on Drivers too. A driver's Type says Analogue or Digital - the kind of output, not how it is wired - so it is never read as a wiring answer. A driver is wired when it has terminals, which is what its **Module-Channel** holds; a driver the strategy alone drives leaves that blank, and External only hides it.

Those are the only two types that state it, so the switch is **only shown on Digital Inputs and Sensors**. A driver's Type says `Analogue` or `Digital`, which is the kind of output rather than anything about wiring, and a logic is not wired to anything at all — so elsewhere the switch would have nothing to read, and one that can only report that it does not apply invites the click that proves it.

Digital Inputs, Sensors and Drivers gain a Module-Channel column — the terminal the point is wired to, as `module-channel`. A driver that also drives an anti-phase output shows both, `2-6/7`. A point with no hardware behind it is left blank rather than shown as `0-0`, so everything in the column is a terminal you can actually find on the panel.

Drivers gain a Type column — `Analogue`, `Digital` — saying what kind of output the driver is. That is a different question from a digital input's Internal/External, which is about wiring, so it is a column of its own.

IC Comms gain three columns, in the order you would trace the link: **Local Module** naming the module in *this* controller the comms is wired to (`S100S`), **Destination** saying which way it talks and to which controller (`From OS11`, `To L12 OS9`, `Global To L128`), and **Remote Module** naming the module at the far end (`S500V`). Where that far end is on another lan the remote module is named with it — `L71 A22V` — because `A22V` alone is a complete reference only while the module is on this lan.

The controller's own list heads its first column "Local Module", but that column holds the *value* currently passing through the comms (`26.63`) rather than a module at all, so it is retitled **Value** and the module the comms is wired to takes the name it was using. That module is read from the item's **Connection**.

Which *end* of the link it is depends on the direction: a `To` comms reads that module and sends it out, while a `From` comms writes the arriving value into it. The Destination column beside it tells you which way round you are looking - it leads with `To` or `From`.

Lan 0 is Trend's "my own lan", so a link to outstation 11 on this lan reads `To OS11` rather than `To L0 OS11`; any other lan number is shown. Worth knowing that a controller may also write its own lan out in full: on one live IQ3 — itself Address 9 on Lan 14 — N1 says lan 0 and N9 says lan 14, and both mean outstation 24 on the local lan.

Knobs gain Range and Drives — what the setpoint may be set to (`20.00 - 28.00`) and which module moves when it changes (`F13E`). A setpoint on its own says where it is set; those two turn a knob list into the setpoint schedule a handover pack asks for.

Every list gains a Status column saying whether the item is in normal automatic control — `Normal`, `Disabled`, `Overridden (21.50)`, `Hand`. An IQ4 driver states this outright and the tool keeps the controller's own column; a sensor, a digital input and every IQ3 state it as separate Disable and Override rows, which are read and combined. Tick **Not normal only** to leave just the items that are not running automatically — the override left in from a previous visit, or the module somebody disabled to stop it nuisance-alarming.

An **IC Comms that is not getting through** reports `Failed` in the same Status column, so **Not normal only** finds a broken cross-controller link without a switch of its own. On one live IQ3 that is one of its ten.

Tick In alarm only to leave just the items the controller has flagged. This costs no extra reading: the controller marks an alarmed item in the list it already served. It works on every module type, including Logics, Functions, Drivers and Knobs, which carry no Alarm column of their own. On Sensors and Digital Inputs the controller's own Alarm column is read as well, which is where an IQ4 puts the wording `Digin` for an alarmed input.

The two states are coloured in the list, so they can be told apart without ticking anything: an item **in alarm is red**, and one **not in normal automatic control — overridden, disabled or in hand — is amber**. They are different jobs. The red is the plant telling you something; the amber is an engineer having told the plant something, and an override left on after a visit is the more common of the two faults. An item that is both is drawn red, because that is the one to look at first. An item whose Status has not been read yet is left uncoloured rather than being called normal, and gains its colour as the Status column fills in behind it.

The Plots list gains Last Record and Records. A plot that has stopped logging looks exactly like a healthy one in every list the controller serves; the date it last stored a sample is the only thing that says otherwise. Records reads `410/1000` — how full the ring buffer is, and so how much history is left before what is on the controller stops covering the fault being chased.

Every list gains a Plot column, last, naming the plot that records the item — `15 Minutes (P11)`, or `COV (P13)` for a change-of-value plot. Where the controller serves a column of its own for this it is retitled to match, so there is only ever one. A plot can record any kind of module, so sensors, drivers, knobs, logics and the rest all carry it; an item nothing records has an empty cell, which is an answer rather than a gap. Where the controller serves a Graph column of its own and leaves it empty — which varies by firmware — that one is filled rather than a second added beside it, and it is moved to the end of the list so the column sits in the same place on every module type whichever way it got there.

The **Plots** list is the exception: a plot records something else, never itself, so a Graph column there says nothing. Right-click any plot and **Open chart** instead.

Functions and Logics gain a Type column, third in the list beside the label, saying what the module actually does — `Comparator`, `Adder Scaler`, `Gate`, `Average`, `Timer`, `Combination`. A list of forty functions all labelled "Function 23" is otherwise unreadable without opening each one.

A function's type also states the formula it computes — a comparator's type reads `Comparator` followed by the expression it evaluates. That is the definition rather than the answer to "what is this module", and long enough to push the name out of sight, so only the name is shown. Types that carry no formula — `Average`, `A to D`, `Hysteresis Band` — are shown whole.

A Plots list fills in the blank Periods. The controller writes Period only where there is an interval to print, so a change-of-value plot and a triggered one both arrive blank — which reads as "not known" rather than what it means. Those cells now say **COV** and **Triggered**. Only blank cells are filled; a period the controller stated is never overwritten, and only the blank rows cost a read.

These columns are the one thing here that is not free. The controller does not put them in the list, only on each item's own page, so they cost one read per item, and the Graph column additionally reads each plot once. On a large controller that is tens of seconds, and over a slow VPN longer. They fill in as the answers arrive, and **Stop** ends the read and keeps what has arrived.

A module that is a single page — Address Page, Time, Program — has no item list, because it *is* the parameters. Those pages use the whole right-hand side for the parameters instead of showing an empty item list headed "0 items" above them, and **Export** offers those parameters as what is displayed: the first option reads "Export 54 displayed parameters (Address Page)" and writes them as Parameter/Value.

Refreshing one item

Right-click an item → Refresh reads that one item again — its row in the list and its own page — instead of the whole module type. After an override, a commissioning change or an item that did not answer the first time, that item is the one you want to see again, and re-reading four hundred sensors to see one of them is

minutes on a slow link. The entry names the item it will read, *Refresh S26*. The row is updated where it stands, so the list keeps its scroll and selection; if the item is selected, its parameters are read again too, and the log says when it was read.

The Plot column is filled from the plots this tab has already read: refreshing one item never re-reads every plot on the controller. If the item's own page does not answer, or the item is no longer listed, the log says so and the row is left as it was — a blank put back over it would read as "not wired".

Trend points on the History Graph

Right-click an item → **Add to History Graph** starts sampling that point over time on the **History Graph** tab, beside whatever BACnet and Modbus points are already logged there — one time base across the protocols, which is what a mixed-site commissioning check needs and what no single controller can give you.

The History Graph reads Trend points **through a session of its own**. It uses a module viewer's session where one is open — an IQ controller has few to give — and otherwise logs in itself, with the login held for that controller. So closing the viewer tab does not stop the recording, and a point can be trended from the **Sites** tab without opening a viewer at all (below). The log says which controller it opened and whether it got in as a named user or a guest.

Select several items first — click, then Ctrl-click or Shift-click — and they all go in one action: right-clicking inside a selection keeps it, and the entry then reads *Add to History Graph (4 points)*. The four duct temperatures of an AHU belong on one chart, and adding them one at a time is four trips through the menu.

It is a different question from **Open chart**: that shows what the controller has *already recorded* in a plot, at whatever interval the plot uses. This shows what the point is doing *now*, at the interval you choose, whether or not anything is logging it.

Digital points chart as 1 and 0 — the controller says `On` and `Off`, and a chart needs a number. A value the tool cannot read as a number leaves a gap in the trace rather than being drawn as zero. Sampling goes through the module viewer's own session, so the controller is not logged into twice; close the viewer and the channel simply stops answering rather than disappearing.

Trend points in the Commissioning Suite

Everything read into an open module viewer is offered to the **Commissioning Suite** as live points, alongside the BACnet and Modbus ones. So a Trend controller can now be matched against an imported design schedule, verified as-found and as-left, and written into the same JSON, CSV and branded PDF evidence pack — which until now the suite could only do for the two protocols a Trend site is *not* built in.

A point is named by the controller's network address rather than its IP — `trend:L14-OS9:S26` — because that is what a point schedule carries and what survives the site being re-addressed.

Two things worth knowing. Trend points are always reported as **not writable**: that is this tool's own guarantee rather than a judgement about the point. And a table is stamped with the moment its read finished; the suite fails closed on anything it cannot date, so a list read hours ago is marked stale rather than quietly witnessed as current.

Following a value through the strategy

Right-click an item → **Trace** opens a tree in a tab of its own, rooted on that module: what feeds it above, what it feeds below. Expanding a branch reads one more page, so you walk the strategy hop by hop instead of reconstructing it from a drawing that is not in the van. Each node shows the module, its label, its value, its status and its terminal where it has one.

Trace to hardware follows the value back on its own until it reaches a module wired to a real terminal, or an IC Comms — which means the value arrives from another controller and this box cannot say any more about it. The trail is listed in the log with what stopped it. It gives up after two dozen hops rather than going round a strategy that loops back on itself.

This works because the controller draws the direction rather than writing it: `jumpin.gif` beside a connection it reads, `jumpout.gif` beside one it writes. An item's own detail pane shows the same thing under **Wired to**.

The terminal map

IO Channels, near the top of the module types, lists every terminal on the controller and what is wired to it — module, channel, item, label and which list it came from. The terminals with nothing on them read `spare`, which is the question an engineer with a cable in their hand is actually asking.

It is built from lists already read rather than by asking the controller again, because the Module-Channel column is where the answer lives. So read Digital Inputs, Sensors and Drivers first (or use **Read all**) — and if any of them has not been read, the map says so, because **a terminal offered as spare because its list was never read is how a second cable ends up on an occupied channel**.

Two honesty notes. A module the controller lists gets all of its channels, so the spare ones show; the **onboard** I/O gets only the channels something is on, because a controller states how many it is *using* and never how many it *has*. And a raise/lower driver holds two terminals — both are shown as taken.

On one live IQ4 the map came to 43 terminals with 5 spare, and reconciled exactly with the controller's own counts of 15 onboard and 23 external in use.

Reading a time schedule

Select a **Time Schedule** and its normal week appears under the parameters, one line a day, as hours rather than as the controller's own list of changeover points:

```
Normal week

    Monday        05:00-23:00 On

    Saturday      -
```

The week lives on a page of its own, so it costs one extra read — made only when a schedule is actually selected, never for a whole list.

`null` is the controller's word for "this schedule has nothing to say at this hour", which is **not** the same as commanding Off: another schedule may be driving the point. Only the hours the schedule is actually commanding something are listed, and a day it never commands reads as `-`.

The controller's clock

Opening a controller reads its Time module once and says whether its clock agrees with the laptop. A controller an hour out runs **every time schedule on it** an hour out, and nothing else on site tells you.

It also says when no summer-time changeover has been set — which is worth knowing before the clocks change rather than after. Controllers state this two ways: a modern `Daylight Savings Applied` setting,

and the older Daylight Start/End day fields. The modern one wins where it exists, because a controller can be applying summer time through it while every legacy field reads zero.

The alarm log

Alarm Log sits at the top of the module types, because it is the first thing worth reading on a fault call. It is the controller's own alarm history — module, label, type, value, time, transition and current state — read in one request and listed like any other module type, so the filter, the column sort and the export all work on it. Clicking a row opens the parameters of the module that raised it.

What repeats is called out in the log as it is read. That matters more than it sounds: on one live IQ3, **292 of the 300 retained entries were a single flapping network node**. That is both the fault and the reason no other fault is still on record — the buffer holds a fixed number of entries, so one noisy module quietly erases everything else.

Exporting every plot at once

The export button's third option writes **one CSV per plot** into a folder you choose — the whole controller's recorded history, filed against the job in one go rather than a hundred right-clicks. Each file carries the same provenance header as any other export, and the samples at full precision.

It costs two requests per plot, so a controller holding a hundred is two hundred: **Stop** works between plots, and what was written is reported rather than only what failed. A plot recording nothing gets no file — a file of headings with no rows reads as a failure — and is named in the log instead, as is any plot that could not be read.

Keeping history the controller throws away

A controller holds a fixed number of samples per plot — a thousand on the ones measured here — in a ring: at capacity every new sample discards the oldest. A 15-minute plot therefore covers about ten days, so the fault you are asked about in March happened in January and is long gone.

Archive plot histories merges each visit's samples into a folder you keep, one file per plot, rather than replacing it. Visits overlap heavily — most of what the controller still holds was archived last time — so only the new samples are added, and the log says how many and what the archive now covers:

```
P9 Pri LTHW Req: +768 new, 1768 held (12 Jan 2026 to 30 Jan 2026)
```

Point it at the same folder each visit. Two things it will not do: a timestamp already held keeps the value already held, so re-reading never rewrites history; and a line that cannot be read is skipped rather than taken as the end of the file, because an archive is appended to over years.

A record for the client

The export menu's last option writes a **branded PDF record** of the controller. Its header says which panel this is and how to reach it: Controller Identifier, Lan-Outstation, Controller Version, Serial Number, Site GUID, IP Address, MAC Address, Hostname, UDP Port, vCNCs and when it was Exported. Then, in order:

Section	What it holds
Outstation Summary	Each module type read, and how many items it holds
Hardware I/O Summary	The terminal map, spare channels marked
IC Comms	Its cross-controller links, any not talking flagged
Alarm Log	The entries themselves, newest first — Item, Item label, Type, Event, Time, Status — with the most frequent alarms named in the note
Alarm Destinations	Item, Label, Type, To, and whether it has **Failed**
Sensors	Value, units, type, scaling, terminal, the two alarm levels, alarm, status and plot
Digital Inputs	State, type, terminal, the state it is judged against, alarm, status and plot
Drivers	Value, type, terminal, alarm, status and plot
Knobs	Value, units, range, the **Destination** it drives, and plot
Switches	**State** and plot

A row in alarm or not in normal control is flagged, by the same rule the screen and the workbook colour it by.

The header's facts come from three pages no module list carries — the Address page, the Ethernet network and the Virtual CNCs list — read once when the viewer connects, and everything else from what has already been read. So a record, like a workbook or CSV export, is **written straight away**, even in the middle of a Read all: the log says *Writing …* at once and names the file when it is done, and the read carries on alongside. Only the plot histories have to wait for the controller — they start when the read in hand finishes its current module type, ahead of anything else queued, and **Stop** ends the read without throwing them away. A type the controller offers but that was not read this visit still has its section, saying so: silence about a section reads as "there is none of this here", which is a different claim from "this was not read". A type the controller does not offer at all has no section.

Every page carries a hairline at its foot with the credit, the version it was made with and the page number, so a page printed on its own still says where it came from and where it belongs.

Long tables are cut at sixty rows with the cut declared, because a point list of four hundred rows is an export rather than a report anybody reads — and showing sixty as though they were all of it would be worse than either.

What changed while I was on site

The **snapshot** button beside Export answers the question you are asked on the way out. Take a snapshot when you arrive, do the work, re-read the module types you care about, and **Compare** — every setpoint, limit, schedule and switch that moved is listed by module, item and column:

```
Knobs K3: Value '24.00' → '21.00'
```

It compares what has been read, so read what you want compared — **Read all** before and after gives you the whole controller. It reports what somebody else changed from a supervisor while you were in the plant

room just as readily as what you changed yourself, which is the point.

The alarm log is deliberately left out: it grows on its own between two captures, and every new entry would be reported as a change you made.

A snapshot can be **saved to a file** and compared against on the next visit. A file that is not one of ours is refused rather than read as an empty snapshot, which would compare as "every item on this controller is new".

What an export says about itself

Every CSV now carries its provenance in the header: which controller, its Trend address and firmware, and when the file was written. Each module type's section adds when that list was read, **which columns this tool derived** rather than the controller serving them, the items that never answered, and whether the list was truncated.

That last group matters most. A derived column is this tool's reading of the controller's pages; a file that does not distinguish them invites whoever opens it in six months to treat both the same way.

Charting a logged point

Right-click an item with a Graph entry → Open chart opens that plot's recorded history in a tab of its own. The controller stores the samples itself — up to a thousand per plot — and this is the same data its own graph page draws.

On the **Plots** list itself the entry is always available, because every plot keeps its own history: the row *is* the plot, so it charts itself. What the plot records is read off its own page.

- **Chart / Table** switches between the trace and the raw samples. Hover the chart to read off the nearest sample's timestamp and value.
- **⭳ Export** writes the samples as an **Excel workbook**, a **branded PDF** or a **CSV**, always at full precision — not rounded to six significant figures, which would cost you 40 units on a seven-digit meter reading and quietly break a consumption calculation. The record names the plot, its units, the period covered and anything the controller declared but did not send.
- The trace is **broken wherever the controller recorded nothing**, rather than joined across the gap. A straight line through a two-day outage looks exactly like a slow genuine ramp, and that is not a distinction to leave to the eye.

If a read fails, the chart says so. It will not report a failure as "this plot has recorded nothing" — those need different responses from you, and only one of them is about the controller.

- **Detail** (bottom) — the selected item's own parameter page, the same parameters the controller shows for it.

Read all walks every module type, which is what you want before an export or when taking a record of a controller before working on it; **Stop** ends it early.

The **export button** asks what you want:

- **Export displayed rows** — only what is on screen, so ticking **External only** and choosing this gives you exactly the external digital inputs and nothing else.
- **Export everything read** — every module type read so far, one section each, because the types genuinely have different columns.

Both options carry their counts, so you are choosing between "29 displayed rows (Digital Inputs)" and "all 3 module types read" rather than between two words. An option that would write nothing is greyed out.

Each of the two names its formats one step in — **Excel**, **PDF** and **CSV** — so the format is chosen on the menu rather than in the save dialog's file-type box. **PDF** on *displayed rows* is that one list, dressed like the controller report; on *everything read* it **is** the controller report, the curated one with the outstation summary and the I/O map, rather than a second document saying something slightly different.

The **Excel workbook** carries a title band naming the controller, the provenance block (controller, address, Trend address, firmware, when it was read, and that it was read by this tool, read-only), and one sheet per module type with a frozen, filterable header, alternate rows shaded, channel numbers stored as numbers so they sort properly, alarmed items in red and overridden ones in amber as on screen, and a landscape page set-up that repeats the header on every printed page. Exporting several types adds a **Contents** sheet that links to each. The **CSV** is the same record as plain text.

Both are named for the controller and what came out of it, with the extension following the format picked:

```
L17 OS16 Block C AHU 6 Kitchen Digital Inputs.xlsx
```

Lan and outstation lose the zeros Trend pads them with — you write L17, not L017 — but only the ones in front, so L10 stays L10.

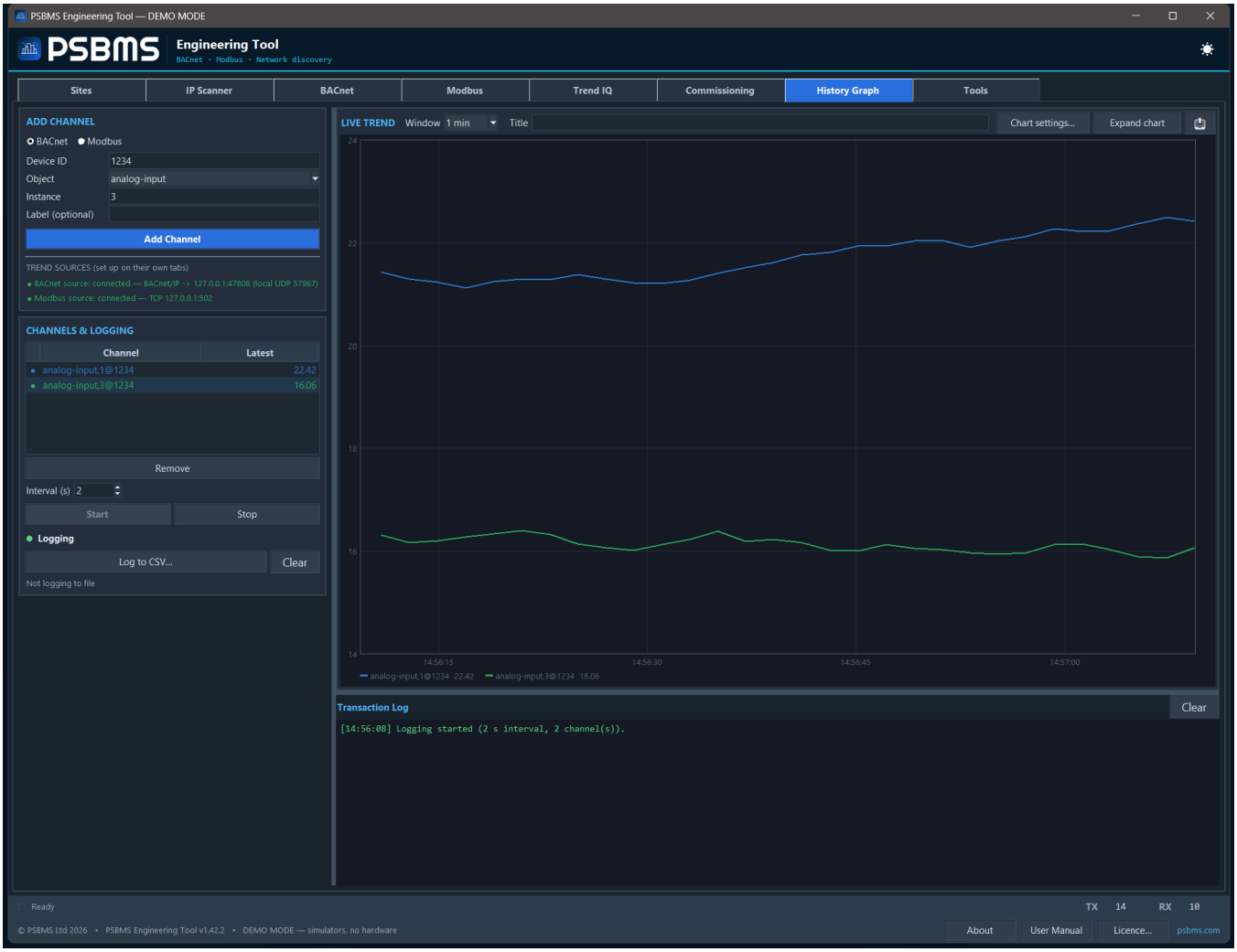
> **A blank derived cell is not always "nothing there".** Some controllers drop requests when several are in flight; each read is retried, and anything that still will not answer is **named in the log** rather than left as a blank you would read as "internal, not wired". If you see that warning, press refresh.

Log in for the full picture. This is the one thing worth knowing before you rely on what you see. A controller does not simply allow or refuse a guest — it answers one with a *reduced* menu and blanks the values on the pages it does serve. One IQ4 offers 9 module types to a guest and 35 to a named user, with readings shown for all of them. An IQ5 shows a guest no modules at all. So fill in **Login user** and **Password** on the Trend IQ tab *before* right-clicking, or the tab will show a genuinely populated controller as a nearly empty one. The heading says which view you are looking at, ending in `- guest` when no login was used.

> **The viewer never writes.** The controller's module pages carry editable fields — a knob's setpoint, a switch's state — because they are the same pages used to change them. The tool only ever reads them: there is no code path in it that can send a change back, deliberately, because these pages command live plant.

---

9. History Graph



Synthetic documentation example: fictional BACnet and Modbus channels on reserved documentation endpoints, plotted and logged like live values.

Watch and record values over time to inspect a control loop's behaviour, and turn what you recorded into a report you can hand over.

1. First **connect** on the BACnet and/or Modbus tab. The **Trend Sources** line on this tab shows what each is connected as (or "not connected").
2. **Add a channel:** choose BACnet (device + object + instance) or Modbus (unit + register + type), optionally name it, and **Add Channel**.
3. Set the **Interval** and press **Start**. Values plot live on the chart and the latest reading shows next to each channel. Use the **Window** dropdown to change the visible time span: **1 min** to **1 hour** follow the readings as they arrive, and **All** holds the start of the run in place so the trace gets denser rather than sliding off the left edge.
4. **Log to CSV…** to also write every sample to a file. **Stop** to pause, **Clear** to reset the chart.
5. **Remove** takes the selected channels off. **Remove all** empties the list to set up a different graph; where the channels have recorded anything it asks first, because what they recorded goes with them.

Once logging stops, the chart ends at the last sample rather than at the current time, so the trace stays where it is while you set it up for a report or a screenshot.

> **The window is how you watch; it is not what you keep.** Every sample stays in the chart's memory — up to 100,000 per channel, about 27 hours at one a second — whatever span the Window shows, and a report covers the whole recording by default. Watch a five-minute window all afternoon and the report still holds the afternoon. The export menu says which it will be: **Report the whole recorded session** or **Report only the visible window**, and the report itself names the period under **Covers**. Ask for the window while more has been recorded and the log says what is being left out.

> **Starting the CSV log part-way through keeps what came before.** Logging is usually started once something worth keeping has happened, so everything already on the chart is written to the file first, with the same columns and the elapsed clock starting at the first sample. The file then continues live.

> **Community** charts up to two channels (a preview); **Pro** removes the limit for unlimited channels.

Giving the chart the room

Two dividers can be dragged: the one between the controls and the chart, and the one between the chart and the **Transaction Log** under it. Drag either all the way to its edge to put that panel away, and drag it back out to bring the panel back.

Expand chart — or a double-click on the chart — puts both panels away at once, so the chart fills the tab: the size to take a screenshot at. **Restore panels**, or **Esc** after clicking the chart, brings them back.

On a laptop screen the left-hand column is taller than the window. It scrolls — with the wheel, or the scrollbar that appears when it is needed — so the cards at the bottom of it are reachable.

Title and chart settings

Type in the **Title** box above the chart to title it. The title is drawn over the chart, and it names the report and the file the report is saved as.

Chart settings… opens a window whose changes show on the chart as you make them:

- **Chart** — the title again, **Grid lines**, **Draw as steps** and **Legend**. Steps hold each reading until the next one, which is how a BMS point behaves; a sloped line between two samples draws a change nobody measured.
- **Left axis** and **Right axis** — **Fit to the data**, or untick it and type a **Min** and **Max** to hold the axis to a range: 0 to 100 for a valve or a fan speed. A reading outside the range stops at the edge of the plot. **Units** labels the axis.
- **Channels** — **Show** hides a channel without removing it. Click a colour swatch to choose that channel's colour. **Axis** puts a channel on the left or the right scale, so a fan speed in % and an air flow in m^3/s can share one chart. **Line** sets how thick it is drawn.

Reset to defaults puts everything back as a fresh chart starts.

The settings window opens over the tool, on the screen the tool is on.

A graph you set up once

An AHU airflow check is the same points, the same two axes and the same interval on every visit. **Save graph…** keeps the chart as it stands - its channels with their colours, widths, axes and which of them are shown, the chart settings, and the interval - under a name, with a site. Pick it from **Saved graphs** on the next

visit and the chart is set up and recording: the channels go back on, the settings and the interval with them, and **Start** is pressed for you.

The list reads `name – site`, so the same name can be used on every site you look after. Saving under a name that site already has replaces it, which is what saving a graph again means. **Forget** takes the graph off the site and leaves the chart on screen exactly as it is.

What is kept is what is drawn, and nothing else. A Trend channel names the controller it reads and the item on it; the login for that controller is looked up in the protected store at the moment the graph is loaded. So a graph travels with the site record - to a colleague, to a job folder - without carrying a password, on the same rule as the rest of the record. A controller with no login held is read as a guest, which the log says.

Loading a graph replaces what is on the chart. Stop the CSV log first if one is open: that file's columns are fixed while it is open, and the tool refuses rather than write rows that do not match its header.

Reports

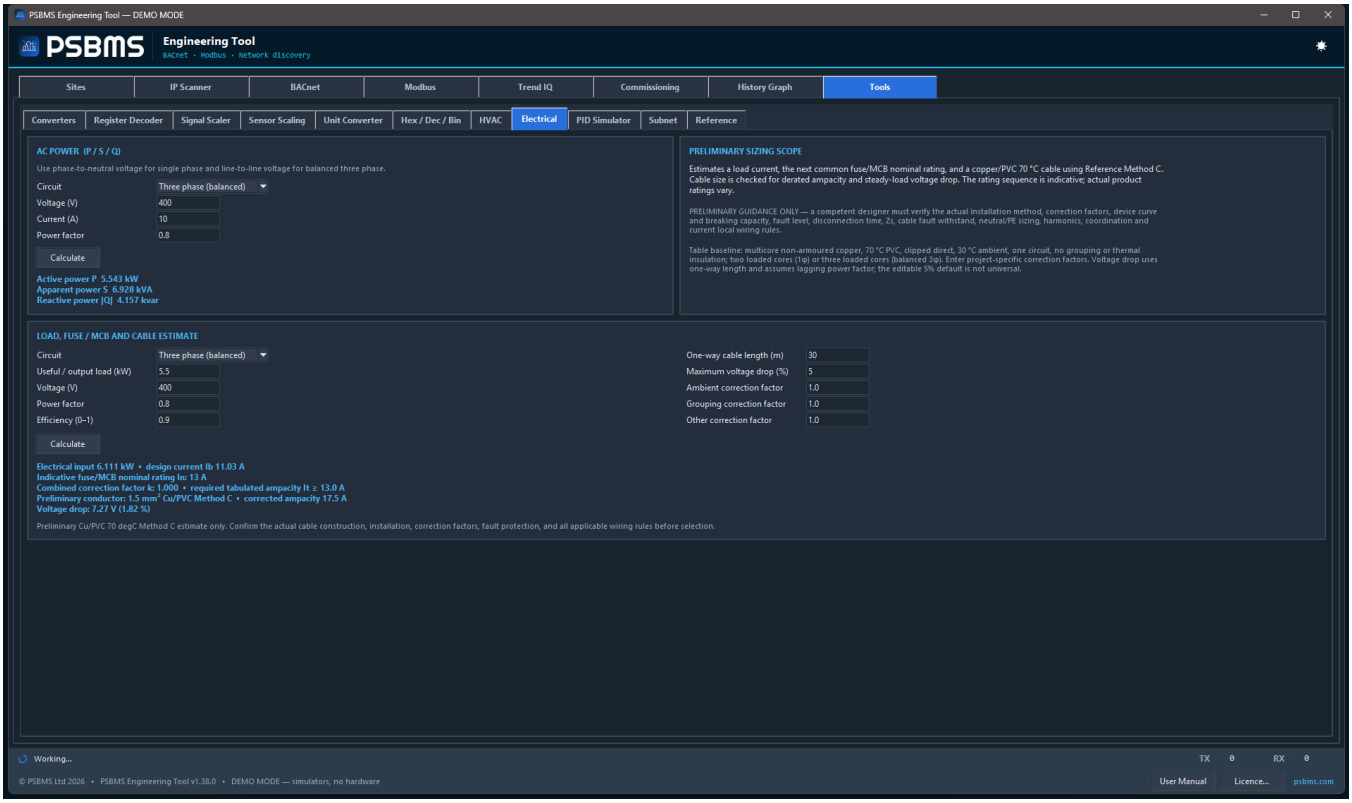
The **Export** icon above the chart saves a report as **Excel**, **PDF** or **CSV**. The report is what the chart shows: the period in the **Window**, and the channels that are shown. A hidden channel is left out.

- **PDF** — the branded record: the title; the period, duration, number of samples, interval, channels and where they were read from; the chart exactly as it is drawn on screen, with its colours, axes and units, on white; and a table of statistics.
- **Excel** — a **Summary** sheet with the same facts and statistics, a **Data** sheet with every sample as real times and numbers, and a **Chart** sheet: an Excel chart of the same channels, on two axes when the chart has two, which can be restyled in Excel like any other.
- **CSV** — the samples: a time stamp, the seconds since the first sample, and a column for each channel, left blank where the point did not answer.

For each channel the statistics give the number of samples and how many the point did not answer, the minimum and maximum and when each occurred, the mean, and the first and last readings. Readings are written as the point holds them: 21.4 read over BACnet is written 21.4, and a meter reading of 1234567 is written in full.

---

10. Tools — the bench calculators

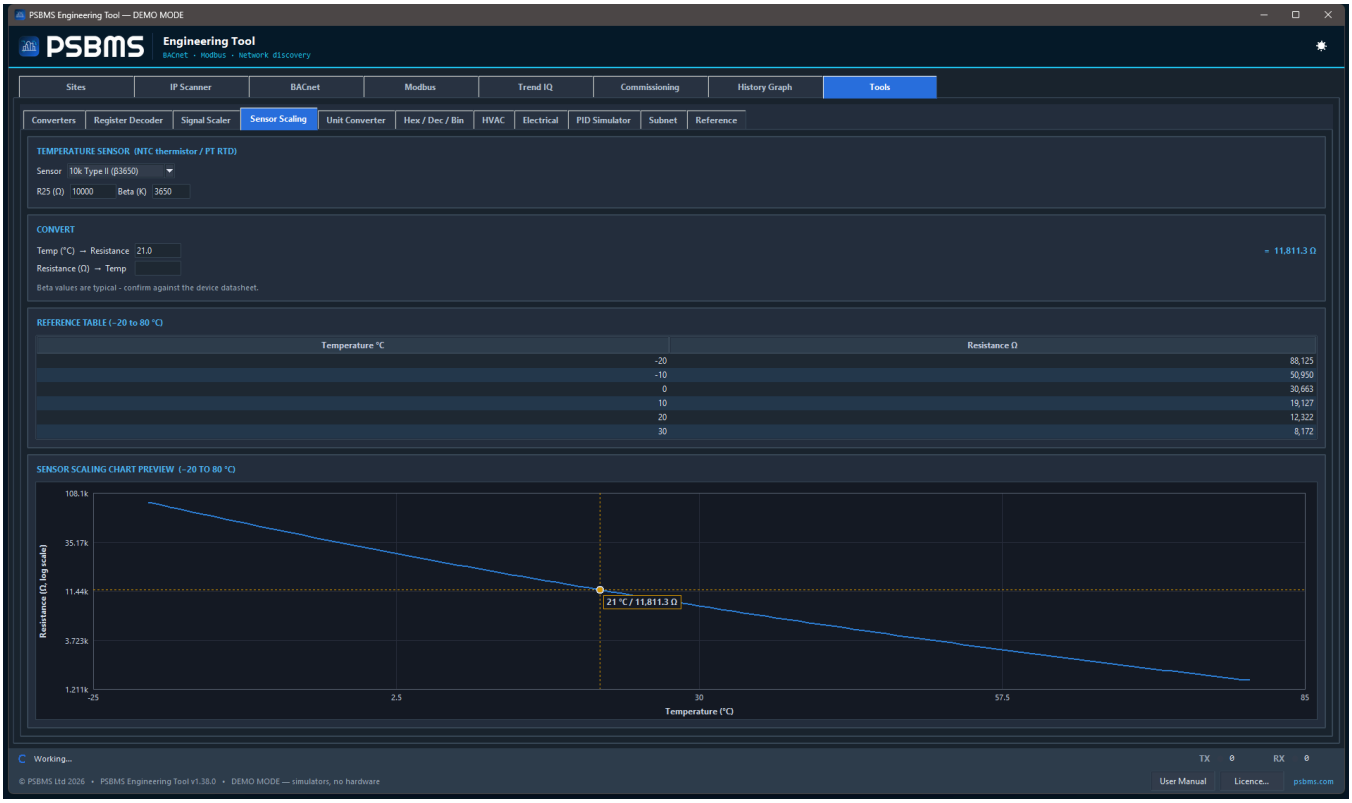


The single home for offline engineering calculations and simulation — protocol, signal, sensor, HVAC, electrical, PID, unit, number-base and subnet tools.

Offline calculators — no connection needed. Sub-tabs:

- **Register Decoder** — paste Modbus registers (decimal or `0x` hex) and see Int16/UInt16, and 32-bit Int/UInt/Float across all four byte/word orders (plus Float64). Answers "which byte order is this meter?" instantly.
- **Signal Scaler** — convert 4–20 mA / 0–10 V (etc.) to engineering units and back. Set the signal and EU spans; type a value on either side. The live chart previews the resulting linear scaling curve and updates with the selected preset and span values; a marker identifies the current conversion point.
- **Sensor Scaling** — NTC thermistor (10k Type II/III presets, editable Beta) and PT100/PT1000. Temperature ↔ resistance works both ways, with a reference table and a live −20…80 °C temperature/resistance curve in the lower part of the page. NTC curves use a logarithmic resistance axis so the full range stays readable; RTD curves use a linear axis.

Both scaling previews use higher-contrast labels in dark mode and abbreviate large axis values with `k`, `M` or `B`. The selected conversion point is shown in a protected high-visibility label so the plotted curve cannot obscure it.



The Sensor Scaling preview uses a logarithmic resistance axis and keeps the selected point readable above the curve.

- **Unit Converter** — temperature, pressure, air flow, power/heat, length.
- **Hex / Dec / Bin** — enter an integer and select its input base, or let **Auto** detect the `0x`, `0d` or `0b` prefix. The page shows canonical hexadecimal, decimal and binary forms plus unsigned and signed two's-complement interpretations at a selected 8-, 16- or 32-bit width. Inputs outside the supported 32-bit range are reported as errors rather than truncated.
- **HVAC** — psychrometrics, water/air coil duty, pump/fan affinity, mixed-air temperature and Kv/Cv conversion. These calculators were moved here from Commissioning Suite so calculations have one home.
- **Electrical** — single-phase or balanced-three-phase active power `P`, apparent power `S` and reactive-power magnitude `|Q|`; plus load current, indicative fuse/MCB nominal rating, preliminary Cu/PVC Method C conductor size and steady-load voltage drop.
- **PID Simulator** — explore a PID Controller using the familiar G/I/T parameter convention against an offline first-order-plus-dead-time plant. This tab appears immediately after Electrical and includes heating, flow and reverse-acting cooling presets. All visible durations are in seconds.
- **Subnet** — IPv4 calculator: network, mask, broadcast, first/last host, usable count, from a CIDR or `IP mask`.
- **Reference** — Modbus exception & function codes, BACnet error codes & object types, common TCP ports.

Using the PID Simulator



A paused synthetic heating response: SP/PV and P/I/D/Output share one time base, while the controller and process settings remain visible.

The automatic **PID Controller** uses `Output = G × (SP − PV) + integral − derivative-on-PV`. Both **I** and **T** are entered in seconds; **G = 0** deliberately enables integral-only control. The derivative term is filtered and limited to ±25%, and automatic output is limited to 0–100% with integrator anti-windup. Inside the configured deadband, the output holds while the integral state is back-calculated. Manual mode lets you set output directly and tracks the controller state for a bumpless return to automatic.

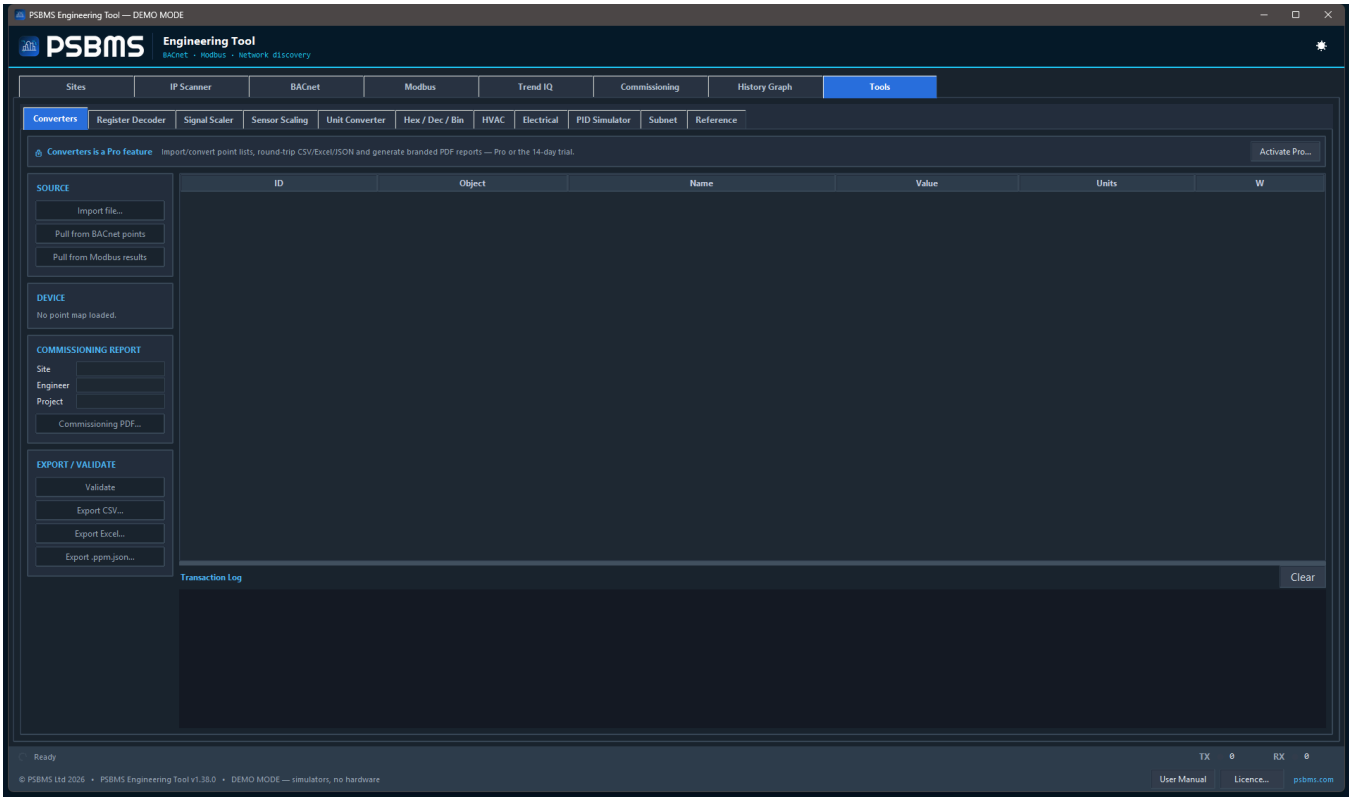
The simulated plant exposes initial and ambient PV, gain in EU per percent output, time constant and transport delay in seconds, plus a live load disturbance. Choose **Slow heating PI**, **Slow heating PID**, **Fast flow — integral only** or **Reverse cooling PID** as a starting point. Setpoint, disturbance, controller settings and process dynamics can be edited while the simulation is running; **Initial PV** is applied by **Reset**. **Start**, **Pause**, **Step** and **Reset** control time; the speed and manual-output controls also remain live. The chart window can show 60, 300 or 900 seconds. Synchronized charts show SP/PV together with the P/I/D/Output traces. The simulator has no save or export.

> **PID simulation boundary.** This is an educational offline simulation of a simplified first-order-plus-dead-time process. The G/I/T parameter convention is familiar from IQ controllers, but this does not emulate IQSET or controller firmware and is not a plant digital twin or source of tuning advice. Its settings, traces and readouts are not controller commissioning or acceptance evidence. Verify any live adjustment through manufacturer guidance, competent engineering risk assessment and a controlled site test.

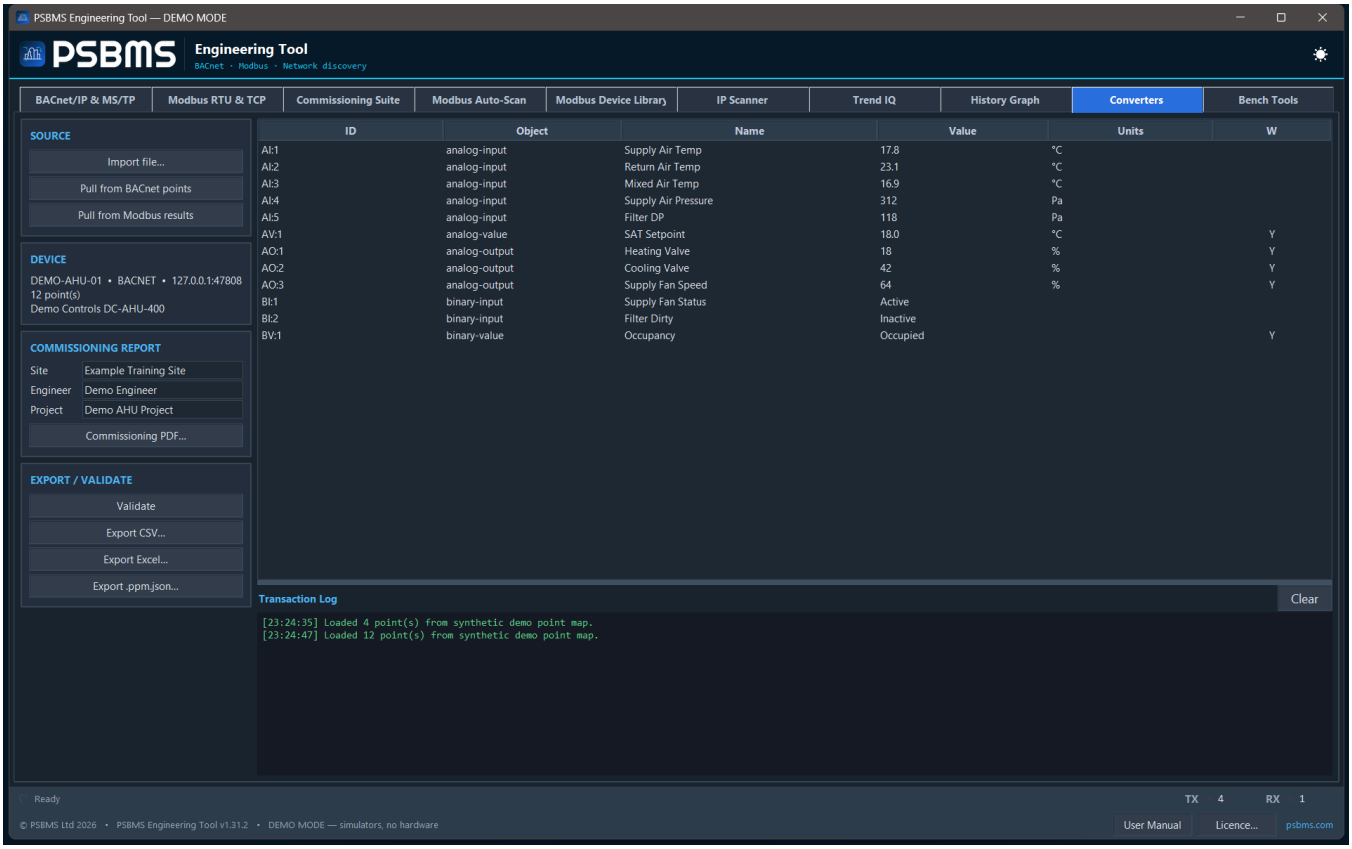
> **Electrical safety boundary.** Fuse/MCB and conductor results are preliminary engineering estimates, not a compliant circuit design or product selection. A competent designer must verify the actual installation method and factors, device curve and breaking capacity, fault level and protection, disconnection time, loop impedance, cable fault withstand, CPC/neutral sizing, harmonics, coordination and the current applicable wiring rules. The displayed Method C assumptions and voltage-drop allowance must be checked for the project.

---

11. Tools — Converters, point-list conversion & reports (Pro)



On Community the tab shows a Pro/Trial upsell bar…



Once unlocked, Converters can import a point list; this synthetic 12-point demo is ready to validate, convert, and turn into a branded PDF report.

Turn a controller's point list into a clean, portable point map and a branded commissioning report. A **Pro** feature — on Community the tab shows an unlock prompt; click **Activate Pro…** to enter a code.

1. **Load points. Import file…** reads a CSV, Excel or `.ppm.json` point list; or **Pull from BACnet points** / **Pull from Modbus results** grabs the points you've just discovered on those tabs.
2. **Validate.** Press **Validate** to flag missing names, duplicate addresses and out-of-range values before you hand anything on.
3. **Export.** Save a portable `.ppm.json` point map (**Export .ppm.json…**), a **CSV** (**Export CSV…**) or an **Excel** workbook (**Export Excel…**) — or generate a **Commissioning PDF…**: a branded, dated report of the point list for the project record. (Import/export needs Pro or the 14-day trial; Trial exports are watermarked, Pro is clean.)

---

12. Sites — the address book of buildings and controllers

A **site** is a building you visit and the controllers in it, whatever they speak. It is the record that outlives a visit: the addresses that took an afternoon to find, the panel each controller sits in, and the note explaining why AHU3 is on manual.

Filling it in. Create a site, give it a name and client, then either add controllers by hand or — far quicker — discover them and file them. On the **Trend IQ** tab, right-click any discovered controller and choose **Add to site…** — or select several and add them all at once, choosing the site once: each one's identifier, IP, Lan, outstation, model, firmware, MAC, UDP port, serial and site GUID are recorded as found. Re-discovering the same address on a later visit updates that row rather than making a second one: the notes you typed survive, and the controller's own answers — its outstation name among them — are refreshed.

A controller that would not reveal its Lan and outstation is filed without them rather than as Lan 0 — Lan 0 is a real address, so a guess would read as a fact.

Notes live against the site and against each controller, which is where they belong: the note about a seized damper should be next to the controller that runs it, not in a document somebody has to find.

The two forms sit either side of the controller list: SITE DETAILS above it, the selected CONTROLLER below it. Each has its buttons beside its heading — **Save site**; **Save controller** and **Forget login** — and its notes to the right of its fields, so the list between them gets most of the height. On a short window the controller form scrolls rather than squeezing the list.

SITE DETAILS holds the site's **Name** and **Client** and its login. The CONTROLLER form holds the controller's **Type**, its **Name**, its **IP Address**, **Lan** and **Outstation**, and its own login. A controller's **Name** is its outstation name: the name it reports about itself, filled in when it is discovered, refreshed on each re-discovery, and the same name the list's **Label** column shows. Type one for a controller added by hand.

Reading what is in a controller

Select a controller and press **Read modules** - or **Read every controller** to sweep the whole site. Each module type the controller serves is recorded with its point list, so the **Modules** column reads `31 type(s), 1204 pt` and the record answers questions away from site: which sensor reads the flow temperature, what is on channel 3-4, whether that AHU has a frost stat.

The sweep runs in the background and **Stop** ends it after the module in progress. Where a login is held for the controller it is used; where none is, the controller is asked as a guest, which on an IQ4 returns a reduced menu - nine module types where a named user is offered thirty-five.

A module type that would not answer is named in the log rather than passed over: "no sensors" and "the sensors would not read" look identical in a list and mean very different things on site. A list the controller cut short is marked with a `+`, because a partial point list that reads as complete would have somebody conclude a point does not exist.

A survey shows on the **TX/RX strip** like any other comms — it is the busiest reader in the tool, one request per point — and the packet inspector lists what it asked for.

Surveying only ever reads. The tool has no way to write to a controller, and a test walks the source of every module involved to keep it that way.

Working the controller list

The list reads **Lan/OS**, **Type**, **Label**, **IP Address**, **MAC**, **UDP** and **GUID**, then what has been read, the login it will use, and your notes — where a panel sits on the Trend network first, then what it is and what it calls itself, then how to reach it. **Type** is the model the controller states (`IQ4E16`, `IQ5-00-50`), as the Trend IQ tab shows it; **Label** is what it calls itself — its outstation name, which the form below the list shows and edits as **Name**.

Click a column heading to sort, and again to reverse it; the arrow shows which way it runs. Addresses, Lan/OS and module counts sort the way they read rather than the way text sorts, so `192.0.2.9` comes before `192.0.2.10` and `L2-OS1` before `L11-OS9`. A controller with nothing in that column sits at the bottom whichever way the column runs — an unread panel is an unknown, not a small value, and sorting should not float every one of them to the top. **Save site** keeps the order the list is showing, so a site sorted by address opens that way on the next visit.

Right-click a controller for the same three actions a discovered controller offers on the Trend IQ tab, plus its reports:

View modules	Opens that controller's live module viewer, using whichever login it will actually use — its own, or the site's
Open web page	The controller's own web interface, in your browser
Save a controller report	PDF or Excel — see below
Copy IP address	

An action that would do nothing is greyed: there is nothing to open or copy without an address, and no Trend module viewer for a BACnet or Modbus row.

Looking at what was read, and getting it out

The **Modules** page beside Controllers is the module viewer you already know, pointed at the site book instead of one live controller: module types down the left with their item counts, the item table on the right carrying the same columns the controller served. Nothing new to learn, and it works with no controller in reach.

The scope decides how wide the page looks:

Scope	Shows
Selected Controller	the controller selected on the Controllers page
Selected Site	every controller in the open site
All Sites	the whole site book

Widening the scope adds **Site** and **Controller** columns to every row — the two things a row loses the moment more than one controller is on screen. On a single controller they are left off, where they would only repeat the same two values down the whole table.

The page lists what a record is made of, in the order a record reads: Alarm Log, Alarm Destinations, IC Comms, IO Channels, Sensors, Digital Inputs, Drivers, Knobs, Switches, Time Schedules and Plots. The same eleven in every scope — the Site and Controller columns are what keep them apart once more than one controller is on screen.

External only beside the filter works as the live viewer's does, on Digital Inputs, Sensors and Drivers: wired to real terminals, or driven by strategy inside the controller. An empty list with it ticked says none of them is wired, rather than looking like a list that failed to read.

IO Channels is here too, although no controller serves it: it is the terminal map, assembled from the I/O modules and the points wired to them, by the same code the reports use. What is on a terminal reads the same way here, in a site report and in a controller record.

What is left out — Functions, Logics, Directories, Sensor Types, Users, Options — describes how a controller is put together rather than what is installed in the building. A survey still reads it and ⭱ **Export** still writes it; it is just not what this page is for.

Columns are shown in the same order as the live Module Viewer, and that order is applied on the way to the screen rather than stored — so a controller surveyed by an older version reads the right way round without being read again. A column that version never read (a sensor's alarm levels, a digital input's required state) is simply absent until you re-read the controller.

Filter looks at every column, not just the label: an engineer chasing a fault types a terminal, a value or a status as readily as a name.

A type the controller cut short carries a `+` after its count, and the line above the table says when the values were last read — the *oldest* reading in view, because a table is only as current as its stalest row.

The bar reads like the live viewer's own. The **refresh** icon re-reads the selected module type from the controllers in scope — that one type rather than all thirty-five, leaving every other type in the record alone. **Read all** re-reads every type, and **Stop** ends either after the module in progress. Stored values are shown the instant a type is selected, so the page is useful on a train; these are the deliberate acts that refresh them.

If a controller was busy when it was read, some items will not have answered and their Type, terminal and Status will be blank — which looks identical to a point that genuinely has none. The line above the table says how many did not answer, so a blank you should chase is told apart from a blank that is the answer. Refresh asks again.

Right-click a row for the same actions as a live module list: **Open chart**, **Trace**, **Add to History Graph**, **Refresh**, **Copy row** and **Copy column values**. Open chart is offered only where a plot was actually recorded, and Add to History Graph only on a list of points — a configuration list such as Alarm Routes, Networks or Virtual CNCs has no value that moves, so trending it would draw a flat line and call it a trend. Trace is unaffected: a Network cannot be trended but it can still be traced. The three that need a live controller open that controller's own module viewer and land on the same module type and item — the site book holds a record, not a connection, so the viewer does the talking.

Refresh reads that one point again from its controller, with the login held for it, and keeps it in the site record — no viewer is opened, and the rest of the list, and when it was read, stay as they were. It is the point-sized version of the refresh icon and follows its rule: it reads from the open site only, so it is not offered under **All Sites**. Nor is it offered on **IO Channels**, which is assembled from several lists rather than read from one page.

⭱ Export writes exactly what is on screen — the same columns, the same scope, the same filter — as an **Excel workbook**, a **branded PDF** or a **CSV**. That is the shape an O&M point schedule, a Niagara import or a handover spreadsheet wants, and the workbook opens straight into Excel. The site, the controller, the scope, any filter in force and when the modules were read are named in the record, so a filtered list cannot be mistaken for a short module type.

The site record and the controller record

Right-click a site on the Controllers page for a **site report**, or a controller for **its own record** — each as a PDF to hand over or an Excel workbook to work on.

The site report answers what somebody asks next week: what is in this building, what each panel is, where it lives on the network and on the Trend Lan, what I/O is fitted, and how much of it is spare. The controller record goes a level down — its identity and how to reach it, then its alarm log, alarm destinations, IC comms, terminal map, sensors, digital inputs, drivers, knobs, switches, time schedules and plots.

Neither reads a controller. Both are built from what is already in the site book, so they are instant, work at a desk, and work with the site network nowhere in reach. The consequence is worth knowing: a controller nobody has surveyed is *named as not read* rather than reported as having no I/O. "No spare terminals" and "nobody looked" are different answers and only one of them sends an engineer to site without a cable. You are told which controllers those are before the save dialog opens, not after the file exists. Read them and export again for a complete record.

The same rule runs through the controller record: a module type the record does not hold is listed at the end under **Not in this record**, because an empty section and an unread one look identical on a page.

The terminal map in both is built by the same code the live module viewer uses, so a site record and a controller record cannot disagree about what is on a terminal.

Sharing a site, and why logins are not in it

Export writes one or more sites to a JSON bundle you can email a colleague, keep in a job folder, or hand to whoever picks up the site next. **Import** merges a bundle into what you already have: a controller you both recorded is matched on address and not duplicated, a blank field is filled from theirs, and a note you have already written is never overwritten by an imported one.

The bundle contains no passwords, and cannot. The format has nowhere to put one. That is deliberate rather than an omission: anything a colleague can decrypt, anyone holding the file can decrypt, and a file carrying a customer's plant passwords is the keys to their building. Logins are instead held in a separate store protected by Windows DPAPI, which ties them to your Windows account on your machine — a colleague could not open it even if they had it.

So the split is:

	Where it lives	Shared?
Site, controllers, addresses, notes	`%APPDATA%\PSBMS Engineering Tool\sites\`	Yes — that is what Export is for
Usernames, passwords, PINs, levels	A DPAPI store under your Windows account	**Never**

One login for the whole site. Most sites are ninety controllers on one password and two that are different. The **Site login** boxes hold a user, password and PIN for the site; a controller's own login is used first and the site's is the fallback, so setting a site login answers for the many without overriding the one panel whose password is not the same. It lives in the same protected store, under your Windows account, and is no more shareable than a controller's.

The **Login** column shows what is held without revealing it — `eng, level 4, password and PIN held` — because somebody is usually standing behind you in a plantroom. A controller with no login of its own that the site login covers reads `(site)`: the column answers whether this controller will get in, which is the operational question, rather than whether this row has boxes filled. Clearing every login box and saving forgets the login; so does **Forget login**, and so does deleting the controller or the site. The CONTROLLER boxes only ever edit that controller's own login — saving there never copies the site's onto it.

A Trend IQ controller's login level is set in the controller's own User Settings, so the form does not ask for one; a level recorded by an earlier version is kept when the controller is saved.

The CONTROLLER form belongs to the site on screen. Opening another site empties it, and **Save controller** asks you to select one rather than writing whatever the form last held - which is not necessarily a controller this site has.

Trend a saved point straight from the record. On the **Modules** page, select one or more points and right-click → **Add to History Graph**. No module viewer is opened: the record holds the controller's address and the login kept for it, and the History Graph opens its own session to read them. Points from different controllers can go in together — each is read from its own. Right-clicking inside a selection keeps it, and the menu says how many points it will add; **Open chart**, **Trace** and **Copy row** act on the row you right-clicked. A

record with no IP address says so rather than failing later, and a controller with no login held is read as a guest, which the log states.

A History Graph belongs to a site too. A graph saved from the History Graph tab is kept in that site's record and offered in its **Saved graphs** list - the points, colours, axes, chart settings and interval of a check you run on every visit. Export the site and the graph goes with it; the logins it needs stay where they are.

Hover a **Password** or **PIN** box, on the site or the controller, for where its login is kept. If Windows cannot provide a protected store, the tab says so under the form rather than offering a Save button that fails when pressed.

13. The status bar, transaction log & packet inspector

Status indicators. The marker beside "Connected", "Stopped" and the like has five states, and each has its own shape as well as its own colour — so you can still read it on a washed-out laptop screen in a plant room, and it survives being printed or pasted into a report in black and white:

Mark	State	Means
● filled circle, green	OK	Connected, running, the read succeeded
◆ diamond, amber	Busy	Working now — connecting, scanning, reading
○ hollow ring, grey	Idle	Not started, or stopped on purpose. Nothing is wrong
▲ triangle, amber	Warning	It worked, but not completely
■ square, red	Error	It was tried and it failed

The distinction that matters on site is the last two rows against the third: a **hollow ring** means you have not connected yet, or you pressed Disconnect. A **red square** means the tool tried and could not — a refused port, a dropped link, a device that stopped answering. A scan that finishes having found nothing is idle, not an error: finding nothing is an answer.

The transaction log. Every tab has a colour-coded log at the bottom: blue = sent, purple = received, green = success, amber = warning, red = error, with timestamps. It's your first stop for diagnosing anything — and the best thing to copy when asking a device manufacturer's support for help. **Clear** empties it.

The status bar (bottom of the window) shows the tool is working and the comms traffic live:

- An **activity spinner** turns while a request is in flight, reading **Working… / Ready**.
- The active **serial settings** appear when connected or scanning (e.g. `9600 8N1 · ID 31`).
- **TX** and **RX** counters, each with an **LED that flashes on every frame** — real-time send/receive traffic at a glance. All three protocols feed it: Modbus frames, BACnet datagrams, and every page a Trend module viewer reads or address a site scan tries - including what the History Graph reads for itself while it records.

The packet inspector. Click the TX or RX counter to open a live window listing every frame sent and received as **hex** — a running packet monitor, newest at the bottom. It keeps updating while you scan or poll; **Clear** empties the capture, **Close** dismisses it (the traffic keeps being captured). **Auto-scroll**, ticked, keeps the newest frame in sight; untick it to hold the view still and read a line - the capture carries on underneath, and ticking it again goes straight back to the live edge.

Frames are counted where they pass, so a bulk read moves the counters for every one of its exchanges — discovering 2,000 points is 4,000 frames, and the strip says so. A few frames at a time each get their own line

in the window; a flood is gathered into one line per tick (`31 x ReadProperty`) with the count still added in full, because a line per frame is a window nobody can read.

---

14. Tips & troubleshooting

- **BACnet finds nothing:** four things silence a device that is there. You are pointed at `127.0.0.1` or the wrong subnet; another app owns UDP 47808 (watch the log warning) - close IQVISION/Niagara, or use the device IP, which needs no broadcast; the **Device ID range** boxes are set, and a Who-Is carries that range inside it, so anything numbered outside will not answer - clear both boxes; or the devices are on another subnet, where a broadcast cannot reach them - use **Scan Range**, which asks each address in turn and then reads the Device object of any address that did not answer.
- **A button seems to do nothing:** if the application hits an internal fault it now says so and writes the details to `%APPDATA%\PSBMS Engineering Tool\faults.log`. Send that file with a note of what you were doing.
- **What did the tool do on site last Tuesday?** Every line any Transaction Log shows is also kept, with its date, time and tab, in `%APPDATA%\PSBMS Engineering Tool\logs\activity-YYYY-MM-DD.log` — one file per day, kept for 90 days (a `--demo` session writes its own `activity-demo-` file instead). Attach the day's file to commissioning evidence, or read it back after a visit. Passwords and PINs never appear in a Transaction Log, so they never appear here either; a login *user name* can appear where a tab names which login it used, exactly as on screen — review the day's file before sharing it outside the project team, as with any export.
- **Modbus times out:** check unit ID, 0-based addressing, and (RTU) the serial settings match the device.
- **Values look wrong (Modbus):** try a different **Interpret As** / **Word swap**, or use the Tools **Register Decoder** to see every option.
- **Windows Firewall** may prompt on first network use — allow it on private networks.
- **Licence lapsed:** the app drops back to the free **Community** tier — it is never locked out, and your saved work is kept. Activate a new code to restore Pro.
