



MergenHub

DOCUMENTATION

User Guide

Installation, publishing, Windows services and logs, licensing, every module, MergenHub Mobile and device templates.

Envest Enerji ve Su Teknolojileri Ltd. Şti.

mergenhub.envest.com.tr/docs · info@envest.com.tr

October 2026 · The online documentation is always the most up-to-date version.

Contents

GETTING STARTED

1	What MergenHub is
2	Installation
3	Quick start: first live data
4	Publishing on a domain
5	Windows Firewall and ports
6	Windows services and the tray app
7	Logs and diagnostics data
8	Licensing and trial mode
9	Backup, restore, upgrade, uninstall

MODULES

10	Channels, devices and tags
11	Protocol drivers
12	OPC UA server
13	REST API
14	Data Bridge (IoT gateway)
15	Advanced (virtual) tags
16	Alarms and notifications
17	Historian (data logger)
18	SCADA / HMI
19	Users, groups and security
20	AI assistant (MCP server)
21	Automation (workflows)
22	In-app AI assistant
23	Troubleshooting

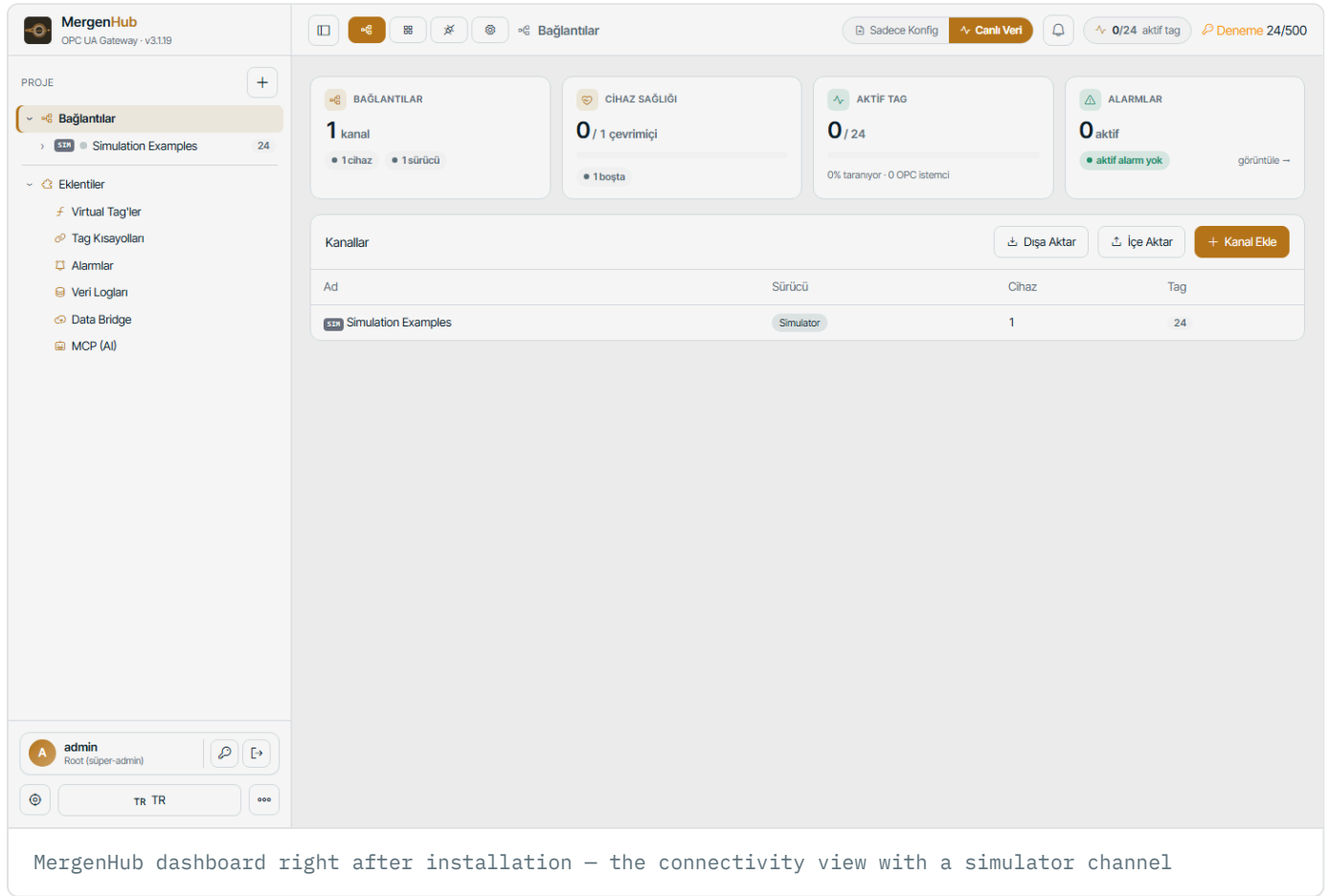
MOBILE AND DEVICE TEMPLATES

24	MergenHub Mobile (iOS and Android)
25	Creating device templates (packages)

What MergenHub is

Architecture, modules, services, data flow and where things live on disk.

MergenHub is a software-based **industrial data acquisition platform and OPC UA aggregation server** for Windows. It reads field devices over 11 industrial protocols, keeps every value in one live tag store, and publishes that data northbound as an **OPC UA server**, a **REST API**, a **web SCADA/HMI**, and outbound **Data Bridge** agents (MQTT / REST). Alarms, historian, virtual tags, users and audit are built in. One installer, three Windows services, no client software.

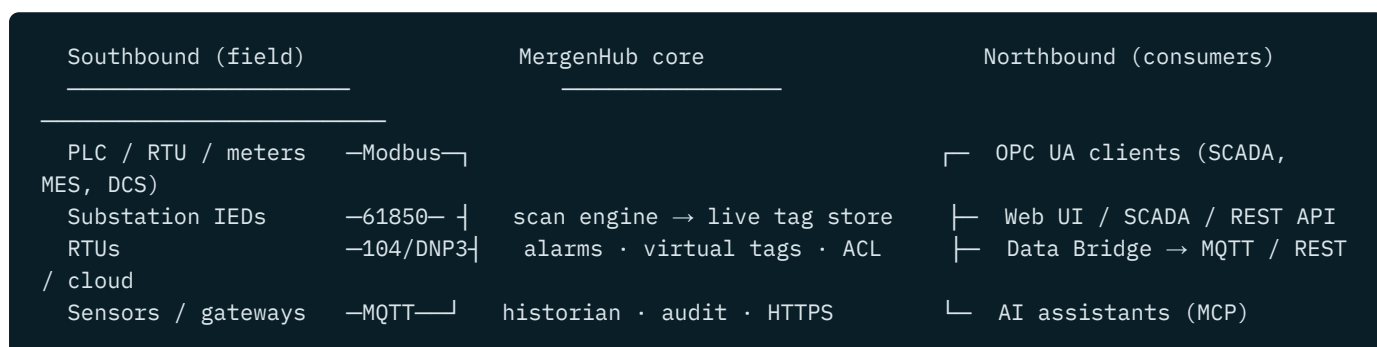


What you get in one installation

AREA	WHAT IT DOES
Drivers	Modbus TCP/RTU/ASCII, Siemens S7, OMRON FINS & NJ/NX CIP, DNP3, IEC 60870-5-104, IEC 61850 MMS, MQTT, OPC UA client, simulator
OPC UA server	All channels in one address space; multiple endpoints; security policies; certificates; per-user access; history access; alarms as UA conditions
REST API	Configuration and live values over HTTPS/JSON with session or token authentication; Swagger UI built in
Web SCADA / HMI	Browser editor with symbol libraries, pipes, flow animation, faceplates, templates; operator view for desktop and mobile

AREA	WHAT IT DOES
Alarms	ISA-18.2 lifecycle, conditions with deadband, e-mail and SMS, per-user notification preferences, history
Historian	SQLite or ODBC targets, log groups with triggers, store-and-forward, trends and CSV/Excel export
Data Bridge	REST server (Kepware-compatible), REST client and MQTT client agents that push data to clouds or third-party systems
Virtual tags	Expression engine: average, min/max, totalizer, derived, link, cumulative; function library
Security	Users, groups, roles, resource-level ACL, HTTPS, session policy, audit log
AI assistant	MCP server so AI assistants can query the plant with your permissions

Architecture



Every driver writes into the same in-memory **tag store**. The OPC UA server, the REST API, the alarm engine, the historian, the SCADA screens and the Data Bridge all read the same live value at the same moment. Scanning is **demand-driven**: a device is only polled while something (an OPC UA subscription, an open screen, a log group, an alarm) actually needs its tags.

PROJECT HIERARCHY

Channel → Device → Tag (and tag groups) . A channel is one driver plus its transport settings (TCP port, serial port, broker). A device is one PLC/RTU/IED under that channel. Tags are the addresses you read or write. See [Channels, devices and tags](#).

Windows services

SERVICE (SCM NAME)	DISPLAY NAME	ROLE
MergenHub	MergenHub Runtime (OPC UA + Config API)	Core: scan engine, OPC UA server, REST API, web UI
MergenHubDataBridge	MergenHub Data Bridge	MQTT/REST push and pull agents (separate process)
MergenHubHistorian	MergenHub Historian	Tag history logging to SQLite/ODBC (separate process)

The two auxiliary services depend on the core and are restarted automatically by Windows if they stop. A **system tray** application (`MergenHub.Tray.exe`) starts at logon for quick status, restart and shortcuts. Details: [Windows services and the tray app](#).

Where things live on disk

PATH	CONTENT
C:\Program Files\MergenHub\	Program files (replaced on upgrade)

PATH	CONTENT
C:\ProgramData\MergenHub\	All data: project, users, settings, license, certificates, logs (kept on upgrade and uninstall)
...\MergenHub\settings.json	Server settings (ports, HTTPS, OPC UA, licensing, retention)
...\MergenHub\projects\	The project (channels, devices, tags, alarms, log groups)
...\MergenHub\scada.json	SCADA screens and zones
...\MergenHub\mergenhub.db	Users, groups, roles, sessions
...\MergenHub\events.db , audit.db , logs.db	Event log, audit log, historian (see Logs)
...\MergenHub\pki\ , webpki\	OPC UA certificates, HTTPS certificate
...\MergenHub\license\	License document and machine key

Note Back up C:\ProgramData\MergenHub and you have everything. See [Backup, restore, upgrade, uninstall](#).

Default ports

PURPOSE	DEFAULT
Web UI / REST API	8090 (HTTP) — or the HTTPS port (default 443) when HTTPS is enabled
OPC UA server	48010 — <code>opc.tcp://<host>:48010/MergenHub</code>

Both can be changed in the installer or later in *Settings* › *OPC UA Server / Web & REST API*. See [Windows Firewall and ports](#).

Editions and limits

Every feature is available in every package; packages differ only by the **number of tags**. An unlicensed installation runs in **trial mode** (500 tags, a 2-hour runtime window per start). See [Licensing and trial mode](#).

Installation

Requirements, the setup wizard step by step, what the installer does, upgrade and clean install.

MergenHub ships as a single Windows installer, `MergenHub-Setup-<version>.exe`. The package is **self-contained**: nothing else (no .NET runtime, no database server) has to be installed on the target machine.

Requirements

	MINIMUM	RECOMMENDED
Operating system	Windows 10 / 11, Windows Server 2016+ (64-bit)	Windows Server 2019+
CPU	2 cores	4 cores for 50K+ tags
RAM	4 GB	8 GB for 100K+ tags
Disk	2 GB free	SSD; historian growth depends on log groups
Rights	Local administrator for installation	—
Network	TCP access to devices; open ports for clients (see Firewall)	—

Browsers: current Chrome, Edge, Firefox or Safari for the web UI. No client installation is needed on operator PCs.

Running the setup wizard

Run the installer as an administrator. Every page has sensible defaults; you can click **Next** through all of them.

- License agreement and trial notice** — the installer explains trial mode (500 tags, 2-hour runtime window) and that entering a license key later removes the limits without reinstalling.
- Server address and ports**
 - Host / Domain** — `localhost` for local use only; `0.0.0.0` to listen on all interfaces (LAN access); a machine name or a real domain when publishing on the internet.
 - Web UI / REST API port** — default `8090`.
 - OPC UA port** — default `48010`.
- Administrator account** — the user name is fixed: `admin`.
 - Enter a password now, or leave it empty to start with `admin / admin` and be forced to change it at first login.
 - Generate strong password** creates a 16-character password for you.
 - When the host is anything other than `localhost`, a strong password is **mandatory** (10+ characters, three of: upper, lower, digit, symbol). The installer will not open the server to the network with a weak password.
- Clean install** (off by default) — when ticked, everything under `C:\ProgramData\MergenHub` (project, users, settings, logs) is **deleted** and you start with an empty project. Leave it off to upgrade in place.

5. **OPC UA server security** — both options are on by default for an easy first connection: *Allow anonymous* and *Auto-accept untrusted client certificates*. Restrict them in *Settings > OPC UA Server* before going to production.

6. HTTPS / web security

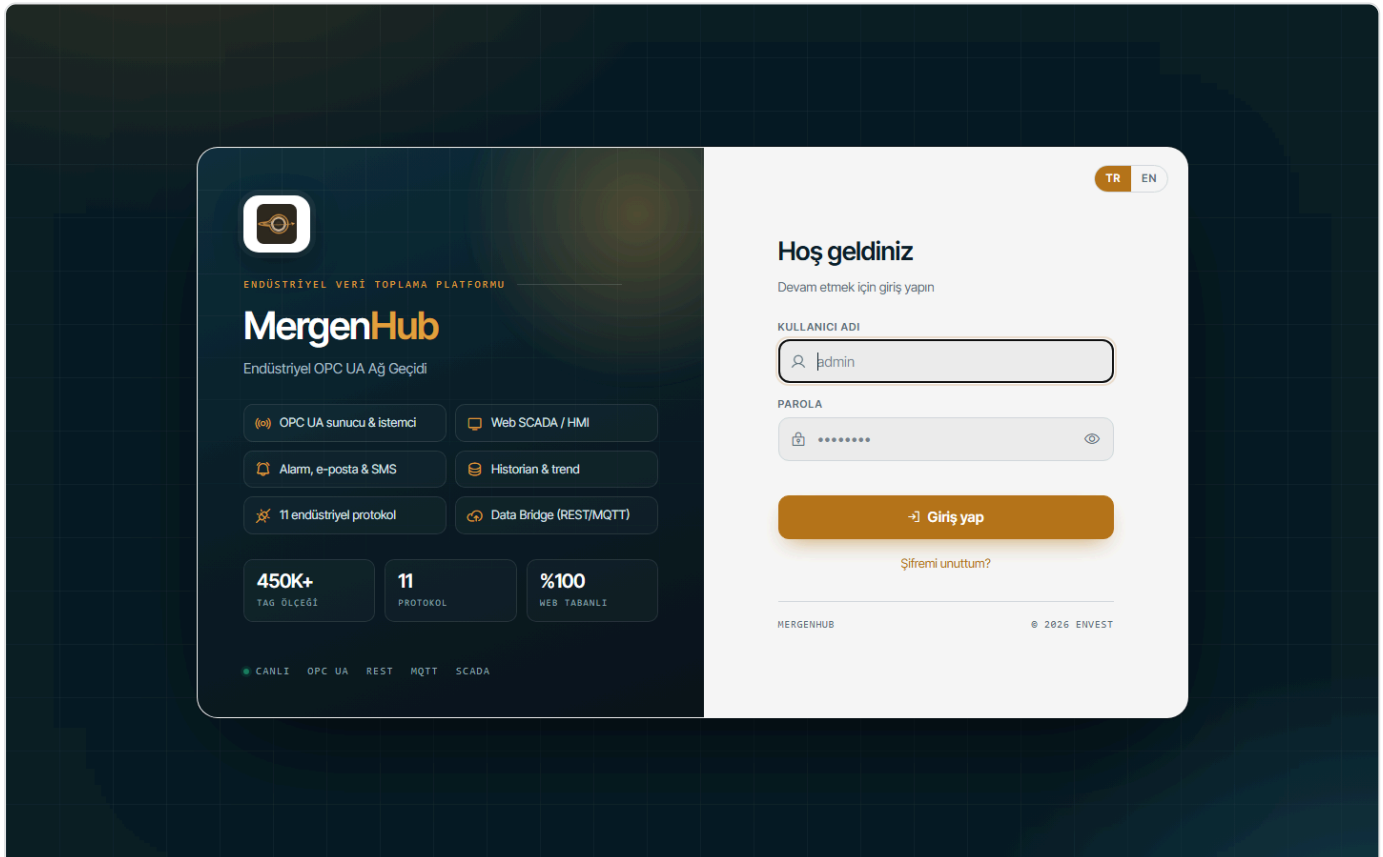
- HTTP only (default; fine for local and test use)
- HTTPS with a self-signed certificate generated during setup (browsers warn once)
- HTTPS with your own PFX certificate (path and password are asked)
- The HTTPS port (default 443) is asked when HTTPS is chosen.

7. **Domain and DNS check** — appears only when a real domain was entered. It resolves the domain and compares it with the machine's public IP so you know whether the A record already points here.

8. **Service startup behaviour** — start at boot (default) and automatic crash recovery with a configurable delay.

At the end the installer:

- writes `settings.json` and `appsettings.Production.json` (data root), creates `C:\ProgramData\MergenHub`,
- generates the OPC UA server certificate and, if chosen, the HTTPS certificate,
- adds **Windows Firewall** rules for the web, OPC UA and (if enabled) HTTPS ports,
- registers and starts the three Windows services,
- installs the system-tray manager for all users and opens the web UI.



First screen after installation: the login page

First login

Open `http://localhost:8090` (or the address you configured). Sign in with `admin` and the password you chose. If you left the password empty, you sign in with `admin / admin` and are asked to set a new password immediately.

Continue with [Quick start: first live data](#).

Upgrading an existing installation

Run the new installer over the old one. It detects the existing installation, stops the services, replaces the program files and **keeps everything under** `C:\ProgramData\MergenHub` : project, users, settings, certificates, license and history. Old service names from early versions (`MergenHubIoTGateway` , `MergenHubDataLogger`) are removed and re-registered under the new names automatically.

Tip Licensed installations are told about new releases inside the application (*Settings › License › Check updates*, and a once-a-day notice). Upgrades are free for the life of the license.

Clean install

Tick **Clean install** in the wizard when you want to discard an old project on a machine that was used for tests. This is irreversible; export the project first if there is anything worth keeping (*Connectivity › Export*).

Silent / unattended installation

The installer is built with Inno Setup and accepts the standard switches:

```
MergenHub-Setup-3.1.19.exe /VERYSILENT /SUPPRESSMSGBOXES /NORESTART /LOG="C:\Temp\mh-setup.log"
```

Wizard values fall back to the defaults (localhost, 8090, 48010, `admin/admin` with forced change). For a scripted deployment prepare `C:\ProgramData\MergenHub\settings.json` beforehand; the installer keeps an existing file.

Installation log

`%TEMP%\MergenHub-Setup.log` records every step, including service registration and firewall commands. Attach it to a support request if setup fails.

Quick start: first live data

Sign in, add a channel, a device and tags, see live values, connect an OPC UA client.

This walk-through takes you from a fresh installation to live values in an OPC UA client in about ten minutes. The screenshots use the built-in **simulator** so you can follow along without any hardware.

1. Sign in

Open the web UI (`http://localhost:8090` by default) and sign in as `admin` . The left tree is your **project**: *Connectivity* holds channels, devices and tags; *Extensions* holds virtual tags, aliases, alarms, historian, Data Bridge and MCP.

The screenshot displays the MergenHub web interface. On the left, a project tree under 'PROJE' shows 'Bağlantılar' (Connections) expanded, with 'Simulation Examples' selected (24 items) and 'Functions' (24 items). Below this, 'Ekentiler' (Extensions) are listed: 'Virtual Tag'lar, 'Tag Kısıyolları', 'Alarmlar', 'Veri Logları', 'Data Bridge', and 'MCP (All)'. The main area shows a table titled 'Simulation Examples cihazları' (Simulation Examples devices) with columns 'Ad' (Name), 'Model', 'ID', and 'Tag'. The table contains one entry: 'Functions' with ID '4' and Tag '24'. Above the table are buttons for 'Özellikler' (Properties), 'Yeniden Adlandır' (Rename), 'Kopyala' (Copy), 'Sil' (Delete), and '+ Cihaz Ekle' (Add Device). The top right shows '0/24 aktif tag' (0/24 active tags) and 'Deneme 24/500' (Trial 24/500). The bottom left shows the user 'admin' (Root (super-admin)) and a 'TR TR' language selector.

Connectivity view – the project tree and the channel list

2. Add a channel

Click **+ Add channel**, choose a driver and give it a name. For a real device pick the protocol (for example *Modbus TCP/IP Ethernet*); for this tour pick **Simulator**. Every driver has its own channel settings (network adapter, serial port, timeouts, retry policy) — the defaults are safe to keep.

Tip A channel is one physical or logical connection. Devices on the same Modbus TCP switch can share one channel; devices on different serial ports need one channel per port.

3. Add a device

Select the channel and click **+ Add device**. Enter the device's address (for Modbus TCP `10.0.0.5:502` plus the unit ID, for S7 the IP and rack/slot, for 61850 the IED address). Set the **scan rate** (how often the device is polled, in milliseconds) and, if needed, timeouts and retries.

4. Add tags

Select the device and use **+ Add tag** or **Generate tags**:

- **Add tag** — name, address, data type, read/write, optional scaling and description.
- **Generate tags** — creates many tags at once from an address range or from the device itself (online browse for OPC UA, 61850, DNP3, IEC 104, OMRON NJ symbols, MQTT topics).
- **Address hints** — the ? next to the address field lists the address formats the driver understands, with examples.

The screenshot shows the MergenHub OPC UA Gateway interface. The top bar displays the device name 'Simulation Examples.Functions' and its status '24/24 aktif tag'. The sidebar on the left shows the project structure with 'Functions' selected. The main table lists 24 tags, including Ramp1 through Ramp8, Random1 through Random8, and Sine1. Each tag row shows its address, data type, scan rate, current value, quality (Good/Bad/Uncertain), and timestamp.

Ad	Adres	Tip	Tarama	Değer	Kalite	Zaman
Ramp1	RAMP (120, 35, 100, 4)	Long	100	37.63	Good	18:48:42.167
Ramp2	RAMP (300, 150.750000, 200.500000, -0.250000)	Float	100	198.81	Good	18:48:42.167
Ramp3	RAMP (250, 0, 1000, 1)	Word	100	8.12	Good	18:48:42.167
Ramp4	RAMP (2000, -1000, 1000, -5)	Long	100	994.93	Good	18:48:42.167
Ramp5	RAMP (250, -1000000, 1000000, -500)	Date	100	995942.00	Good	18:48:42.167
Ramp6	RAMP (1000, 0, 1000000000, 1250)	LLong	100	2536.25	Good	18:48:42.167
Ramp7	RAMP (1000, -1000000000, 1000000000, -5555)	Date	100	999988728.90	Good	18:48:42.167
Ramp8	RAMP (1000, 150.750000, 200.500000, -0.250000)	Double	100	199.99	Good	18:48:42.167
Random1	RANDOM (30, -20, 75)	Short	100	-10.13	Good	18:48:42.167
Random2	RANDOM (100, 0, 1000)	Word	100	381.20	Good	18:48:42.167
Random3	RANDOM (100, -1000, 0)	Long	100	-518.52	Good	18:48:42.167
Random4	RANDOM (1000, -999, 999)	Long	100	-630.22	Good	18:48:42.167
Random5	RANDOM (100, -1000000000, 1000000000)	Long	100	431506399.97	Good	18:48:42.167
Random6	RANDOM (1000, -1000000, 1000000)	Date	100	451194.95	Good	18:48:42.167
Random7	RANDOM (1000, 0, 1000000000)	LLong	100	979424945.12	Good	18:48:42.167
Random8	RANDOM (100, 0, 1000000)	LLong	100	584115.86	Good	18:48:42.167
Sine1	SINE (10, -40.000000, 40.000000, 0.050000, 0)	Float	100	23.81	Good	18:48:42.167

Tag table of a device – value, quality and timestamp update live

5. Watch live values

Switch the top bar to **Live data**. Each tag shows its current value, **quality** (Good / Bad / Uncertain) and timestamp. A red quality means the device could not be read; the *Diagnostics* button on the device shows the raw request/response frames and the last error.

6. Connect an OPC UA client

Point any OPC UA client (UaExpert, Kepware, Ignition, a PLC) to:

```
opc.tcp://<host>:48010/MergenHub
```

Browse *Objects* › *<channel>* › *<device>* to find your tags. With the installer defaults anonymous access and untrusted certificates are accepted; harden this later in *Settings* › *OPC UA Server* (security policy, user authentication, trusted client list). Details: [OPC UA server](#).

7. Read the same values over REST

```
curl -c c.txt -X POST http://localhost:8090/auth/v1/login -H "Content-Type: application/json" -d "{\n  \"username\": \"admin\",\n  \"password\": \"<password>\"\n}"
curl -b c.txt "http://localhost:8090/runtime/v1/values?tags=Simulation%20Examples.Functions.Ramp1"
```

The full API is documented in the application under ... › *REST API* (Swagger) and in [REST API](#).

8. Build a first screen

Open **SCADA**, create a screen, drag a tank and a pump from the symbol library, bind their properties to tags and press **Run**. See [SCADA / HMI](#).

What to configure next

- **Alarms** — thresholds on critical tags with e-mail/SMS notification: [Alarms](#)
- **Historian** — log groups for the tags you need trends on: [Historian](#)
- **Users** — operator accounts with limited rights: [Users, groups and security](#)
- **Publishing** — access from other sites or the internet: [Publishing on a domain](#)
- **License** — enter your key to lift the trial limits: [Licensing](#)

Publishing on a domain

DNS, port forwarding, HTTPS certificates, reverse proxy (IIS / nginx / Caddy), remote OPC UA.

MergenHub runs as Windows services, so it keeps working whether or not anyone is logged in. Publishing it under a domain name means: a DNS record that points to the machine, the right ports reachable through routers and firewalls, and HTTPS so that passwords never travel in clear text. This page covers all three plus the reverse-proxy options.

Decide how the server is reached

SCENARIO	WHAT TO DO
Only this PC	Keep the installer defaults (<code>localhost</code>).
Office / plant LAN	Set host to <code>0.0.0.0</code> (all interfaces); clients use the machine's IP or name; keep the firewall rules the installer added.
Internet, own public IP	DNS A record → public IP; forward ports on the router; enable HTTPS; harden users and OPC UA.
Internet, behind a reverse proxy	DNS → proxy; proxy terminates HTTPS and forwards to <code>127.0.0.1:8090</code> ; OPC UA port forwarded separately (TCP, not HTTP).
VPN only	Recommended for OPC UA and write access from outside; no port forwarding needed.

Warning Never expose the web UI over plain HTTP on the internet. Enable HTTPS (below) and use strong passwords; the installer refuses to bind to a non-local host with a weak admin password.

1. DNS

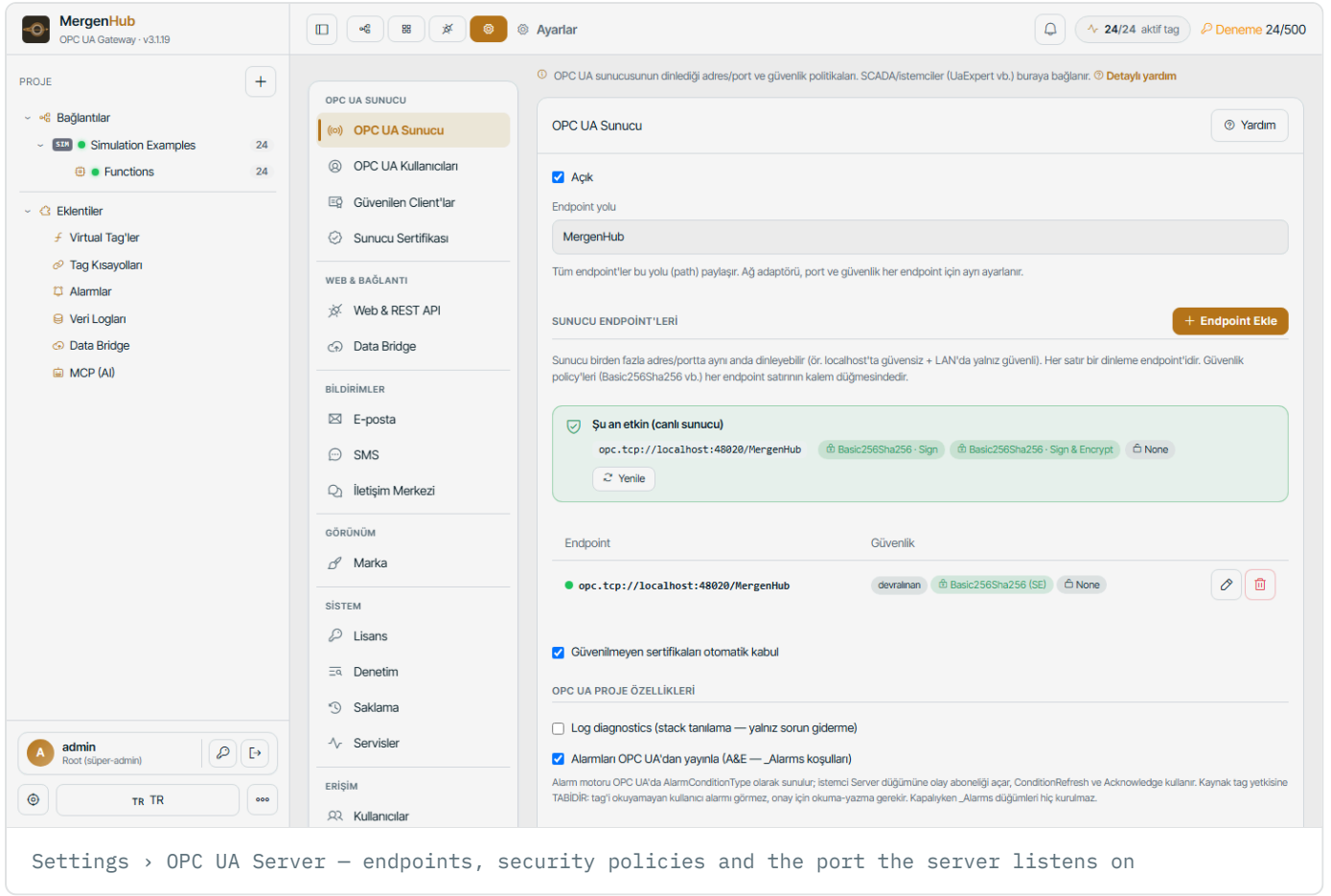
Create an **A record** for the name you want, for example `scada.example.com` , pointing to the public IP of your internet connection (or of the reverse proxy / cloud VM). If your ISP address changes, use a dynamic-DNS service and keep the record updated.

The installer has a **Domain and DNS check** page: enter the domain and it tells you whether the name already resolves to this machine's public address.

2. Ports and forwarding

TRAFFIC	PORT (DEFAULT)	PROTOCOL	NEEDED FOR
Web UI + REST API	8090 or 443 with HTTPS	TCP	Browsers, REST clients, Data Bridge REST server
OPC UA	48010	TCP	OPC UA clients (SCADA, MES, historian)

On the router forward the external port to the MergenHub machine's LAN IP. On the machine itself the installer already created Windows Firewall rules; see [Windows Firewall and ports](#). Change the ports in *Settings* › *Web & REST API* and *Settings* › *OPC UA Server* if you prefer non-default values.



Settings > OPC UA Server – endpoints, security policies and the port the server listens on

3. HTTPS

You have three options.

A) SELF-SIGNED CERTIFICATE (FASTEST)

Choose *HTTPS – self-signed* in the installer, or later run as administrator:

```
"C:\Program Files\MergenHub\MergenHub.exe" gencert "scada.example.com"
```

This writes `C:\ProgramData\MergenHub\webpki\web.pfx`. Browsers show a one-time warning because the certificate is not signed by a public authority; fine for internal use.

B) YOUR OWN CERTIFICATE (PFX)

Copy the PFX to `C:\ProgramData\MergenHub\webpki\web.pfx` and its password to `web.pfx.pass` (plain text file), or pick the file in the installer. Restart the `MergenHub` service.

C) FREE PUBLIC CERTIFICATE WITH LET'S ENCRYPT

- **win-acme** on the MergenHub machine: `wacs.exe --target manual --host scada.example.com --store pfxfile --pfxfilepath C:\ProgramData\MergenHub\webpki --pfxfilename web.pfx --pfxpassword <pw>` and a renewal task; write the password to `web.pfx.pass`. Port 80 must be reachable during validation (or use DNS validation).
- **Caddy** as a reverse proxy handles certificates automatically — see below. This is the simplest production setup.

Enable HTTPS and the port in *Settings › Web & REST API* (or the installer). Keep HTTP redirect on so old bookmarks still work.

4. Reverse proxy (optional but recommended)

A reverse proxy on the same machine or a gateway host terminates TLS, hides the internal port and lets you host several applications behind one IP.

CADDY (AUTOMATIC HTTPS)

```
scada.example.com {  
    reverse_proxy 127.0.0.1:8090  
}
```

Caddy obtains and renews the certificate itself. Keep MergenHub on HTTP `127.0.0.1:8090` behind it.

NGINX

```
server {  
    listen 443 ssl http2;  
    server_name scada.example.com;  
    ssl_certificate      /etc/ssl/scada.crt;  
    ssl_certificate_key  /etc/ssl/scada.key;  
    client_max_body_size 64m;  
    location / {  
        proxy_pass http://10.0.0.20:8090;  
        proxy_http_version 1.1;  
        proxy_set_header Host $host;  
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
        proxy_set_header X-Forwarded-Proto https;  
        proxy_read_timeout 300s;  
    }  
}  
  
server { listen 80; server_name scada.example.com; return 301 https://$host$request_uri; }
```

IIS (URL REWRITE + APPLICATION REQUEST ROUTING)

1. Install *URL Rewrite* and *ARR*, enable *Server Proxy Settings › Enable proxy*.
2. Create a site bound to `scada.example.com:443` with your certificate.
3. Add an inbound rule: match `(.)`, action **Rewrite* to `http://127.0.0.1:8090/{R:1}`, tick *Append query string*.
4. In *Request Filtering* raise the allowed content length if you upload large SVG symbol libraries.

Note OPC UA is a binary TCP protocol, not HTTP — it cannot go through an HTTP reverse proxy. Forward port 48010 directly (or use a VPN).

5. OPC UA from the internet

- Add an **endpoint** in *Settings › OPC UA Server* whose host is the public DNS name (`opc.tcp://scada.example.com:48010/MergenHub`). Clients validate the server certificate against the name they connect to.

- Use security policy **Basic256Sha256 – Sign & Encrypt** and disable *Allow anonymous*; create an OPC UA user with only the rights it needs.
- Turn off *Auto-accept untrusted certificates* and trust clients explicitly in *Settings › Trusted clients*.
- Regenerate the server certificate after changing the host name (*Settings › Server certificate*).

6. Checklist before going live

- ☐ Strong admin password; operator accounts with limited roles ([Users and security](#))
- ☐ HTTPS enabled; HTTP redirected
- ☐ Only ports 443 and 48010 (if needed) forwarded; everything else closed
- ☐ OPC UA anonymous access off, trusted client list maintained
- ☐ Session policy and audit retention reviewed ([Logs](#))
- ☐ Backups of `C:\ProgramData\MergenHub` scheduled ([Backup](#))
- ☐ License activated so the trial window does not stop scanning ([Licensing](#))

Windows Firewall and ports

Which ports MergenHub uses, the rules the installer adds, how to open or restrict them.

MergenHub listens on two TCP ports by default and the installer opens them in Windows Firewall. This page lists every port, the rules the installer creates, and how to open, restrict or remove them by hand.

Ports MergenHub uses

PORT	DIRECTION	PURPOSE	CHANGE IN
8090 TCP	inbound	Web UI, REST API, Data Bridge REST server, MCP endpoint	Installer › <i>Server address and ports</i> ; <i>Settings</i> › <i>Web & REST API</i>
443 TCP	inbound	HTTPS for the same services (only when HTTPS is enabled)	Installer › <i>HTTPS</i> ; <i>Settings</i> › <i>Web & REST API</i>
48010 TCP	inbound	OPC UA server (<code>opc.tcp://host:48010/MergenHub</code>)	<i>Settings</i> › <i>OPC UA Server</i> › <i>Endpoints</i>
device ports	outbound	Modbus 502, S7 102, OMRON 9600 (FINS) / 44818 (CIP), DNP3 20000, IEC 104 2404, IEC 61850 102, MQTT 1883/8883, OPC UA client to remote servers	per channel/device
25 / 465 / 587	outbound	SMTP for e-mail notifications	<i>Settings</i> › <i>E-mail</i>
443	outbound	SMS provider API, license portal (activation, refresh, update check)	—

Outbound connections are normally allowed by Windows Firewall; only the inbound ports need rules.

Rules the installer creates

The installer runs `netsh advfirewall` and adds these inbound *allow* rules (all profiles, TCP):

RULE NAME	PORT
MergenHub Web	the web port you chose (8090)
MergenHub OPC UA	the OPC UA port (48010)
MergenHub HTTPS	the HTTPS port (only when HTTPS was enabled)

They are removed again by the uninstaller. You can see them in *Windows Defender Firewall with Advanced Security* › *Inbound Rules*.

Opening or changing ports manually

Run in an **administrator** command prompt:

```
netsh advfirewall firewall add rule name="MergenHub Web" dir=in action=allow protocol=TCP
localport=8090
netsh advfirewall firewall add rule name="MergenHub OPC UA" dir=in action=allow protocol=TCP
localport=48010
netsh advfirewall firewall add rule name="MergenHub HTTPS" dir=in action=allow protocol=TCP
localport=443
```

PowerShell equivalent:

```
New-NetFirewallRule -DisplayName "MergenHub Web" -Direction Inbound -Protocol TCP -LocalPort 8090
-Action Allow
```

If you changed a port in *Settings*, add a rule for the new port and delete the old one:

```
netsh advfirewall firewall delete rule name="MergenHub Web"
```

Restricting who may connect

Limit a rule to specific networks instead of the whole world:

```
netsh advfirewall firewall set rule name="MergenHub OPC UA" new
remoteip=10.0.0.0/24,192.168.10.0/24
netsh advfirewall firewall set rule name="MergenHub Web" new profile=domain,private
```

For internet-facing servers prefer a reverse proxy on 443 and keep 8090 bound to `127.0.0.1` (set the host to `127.0.0.1` in *Settings* › *Web & REST API*). See [Publishing on a domain](#).

Third-party firewalls and antivirus

- Endpoint security suites (Kaspersky, ESET, Sophos ...) have their own firewall; add the same TCP ports there.
- Some products flag the unsigned installer. Right-click the file › *Properties* › *Unblock*, or add an exclusion for `C:\Program Files\MergenHub`.
- On corporate PCs with AppLocker / WDAC the installer may be blocked entirely; ask the administrator to allow the publisher or the path.

Verifying

- From another machine: `Test-NetConnection <host> -Port 8090` (PowerShell) — *TcpTestSucceeded: True* means the port is open.
- On the server: `netstat -ano | findstr :8090` shows the listening socket and the process ID of the `MergenHub` service.
- The *Event log* (... › Event Log) shows `Web API listening on ...` and `OPC UA server started` lines when the services are up.

Troubleshooting

SYMPTOM	LIKELY CAUSE
Web UI works on the server but not from other PCs	Host set to <code>localhost</code> — change it to <code>0.0.0.0</code> in <i>Settings</i> › <i>Web & REST API</i> and restart the service; or the firewall rule is missing
OPC UA client sees the endpoint but cannot connect	Endpoint host name differs from what the client uses; certificate not trusted; security policy mismatch
Browser says "connection reset" on HTTPS	HTTPS enabled but no valid <code>web.pfx</code> in <code>webpki\</code> ; check the event log
Port already in use	Another program (often another OPC server) uses 48010 or 8090; change the port in <i>Settings</i>

Windows services and the tray app

The three services, startup and crash recovery, the system tray manager, running without a logged-in user.

MergenHub is installed as three Windows services and a small system-tray manager. Services run in the background under the *Local System* account, start at boot and keep running when nobody is logged in — an operator PC, a server in a cabinet or a virtual machine all behave the same.

The three services

SCM NAME	DISPLAY NAME	PROCESS	WHAT STOPS IF IT IS DOWN
MergenHub	MergenHub Runtime (OPC UA + Config API)	MergenHub.exe	Everything: device scanning, OPC UA, web UI, REST API, alarms, SCADA
MergenHubDataBridge	MergenHub Data Bridge	MergenHub.IoTGateway.exe	Outbound MQTT/REST agents and the Data Bridge REST server
MergenHubHistorian	MergenHub Historian	MergenHub.Historian.exe	Writing tag history (values are still live; nothing is logged)

The two auxiliary services **depend on** the core service (`depend= MergenHub`): Windows starts the core first, it writes an internal token, then the others connect to the core over the local REST API. Splitting them into processes means a failure in a cloud connector or a database driver can never take the OPC UA server down.

Settings > Services – status, uptime, restart counters and controls for every runtime service

Startup and crash recovery

- All three services are set to **Automatic** start. On the installer's *Service startup behaviour* page you can switch to *Automatic (delayed)* or manual.
- **Recovery** is configured for each service: if the process crashes, Windows restarts it after the delay you chose (default a few seconds). Inside the core, every driver channel runs under a supervisor as well; a dead channel worker is recreated within seconds without restarting the service.
- The core also has a **runtime restart** that restarts the scan engine in-process (used by the trial window and by *Settings > Services*). It does not touch the Windows service.

Managing services

Settings > Services in the web UI shows each runtime service with start/stop/restart and the auto-restart switch. From an administrator prompt:

```
sc query MergenHub
sc stop MergenHub
sc start MergenHub
sc qc MergenHubHistorian
```

or with PowerShell: `Get-Service MergenHub*` , `Restart-Service MergenHub` .

Note Stopping the core stops the dependent services too; starting the core does not start them automatically until their own start command runs (Windows handles this at boot).

The system-tray manager

`MergenHub.Tray.exe` is registered to start at logon for all users and sits next to the clock as *MergenHub Administration*.

- Shows the service state (green/yellow/red) and the trial countdown when unlicensed.
- **Restart runtime** — restarts the scan engine in-process (new trial window, reloads drivers) without stopping the Windows service.
- **Start / Stop / Restart service** — the Windows services (requires elevation).
- **Open web UI, Open data folder, Event log**, version and license summary.

The tray application is optional; the services run without it.

Running without a logged-in user

Nothing needs a desktop session. After a reboot the services start and devices are scanned before anyone signs in. Only the tray icon requires a user session.

Resource usage

Memory grows with the tag count: roughly 250–350 MB for a few thousand tags, 2–3 GB at 450K tags with many OPC UA subscribers. CPU is dominated by the drivers' scan rates. *Settings* › *Services* and the dashboard show CPU, RAM and disk of the machine; a capacity banner warns when RAM gets tight.

Service accounts and permissions

The services run as *Local System* by default and need write access to `C:\ProgramData\MergenHub`. If you switch them to a dedicated account (for example to reach a network share for ODBC logging), grant that account **Modify** on the data folder and *Log on as a service*.

Windows Event Viewer

Service start/stop and unhandled crashes are also written to *Windows Logs* › *Application* with source *MergenHub*. MergenHub's own detailed logs are described in [Logs and diagnostics data](#).

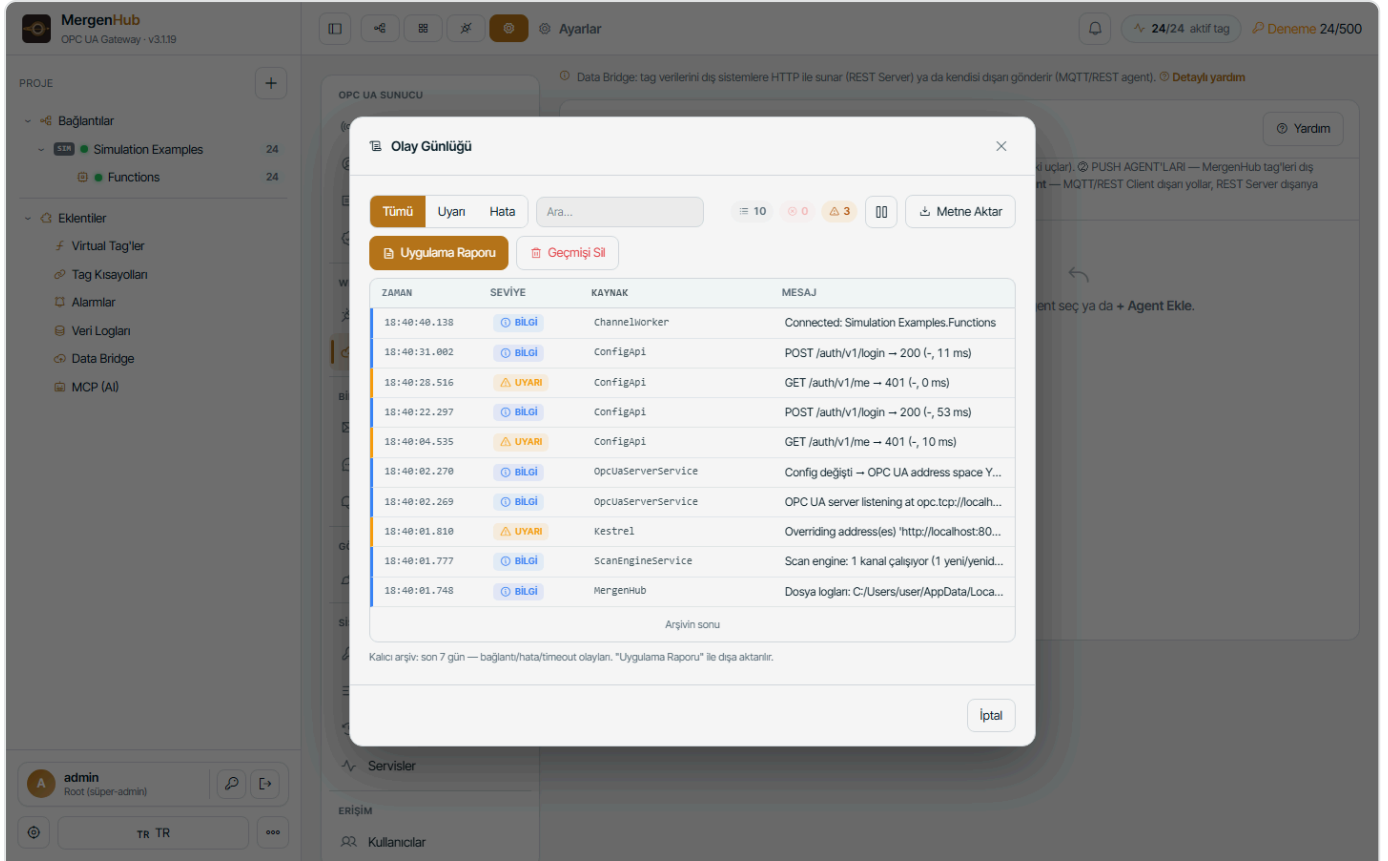
Logs and diagnostics data

Every log MergenHub writes: file logs, event log, audit log, historian database, installer log; retention and locations.

MergenHub keeps several kinds of logs, each with its own purpose, location and retention. All of them live under the data folder `C:\ProgramData\MergenHub`.

Overview

LOG	WHERE	WHAT IS IN IT	RETENTION	WHERE TO VIEW
Application file log	<code>logs\mergenhub-YYYYMMDD.log</code>	Everything the core process logs: startup, driver connections, errors, stack traces, API requests	One file per day; old files pruned automatically	Any text editor; attach to support requests
Event log	<code>events.db</code> (SQLite)	Operator-facing events: device connected/lost, timeouts, service start/stop, OPC UA sessions, alarms raised	7 days by default (<i>Settings › Retention</i>)	<i>... › Event Log</i> in the web UI, with filter, search and <i>Export to text</i>
Audit log	<code>audit.db</code>	Who changed what: every configuration write (POST/PUT/DELETE), OPC UA and REST writes to tags, logins, MCP tool calls, license actions	90 days by default	<i>Settings › Audit</i> (chip filters by category, user, time)
Alarm history	inside the project store	Every alarm transition: raised, acknowledged, cleared, by whom	90 days by default	<i>Alarms › History</i>
Historian	<code>logs.db</code> (SQLite) or your ODBC database	Tag values recorded by log groups	Until you delete or downsample	<i>Historian › Trend</i> , CSV/Excel export, OPC UA HistoryRead
Device diagnostics	in memory	Raw request/response frames (hex) per device, last error, timing	Rolling buffer	<i>Device › Diagnostics</i> ; Wireshark-style trace export
Installer log	<code>%TEMP%\MergenHub-Setup.log</code>	Every installer step, service and firewall commands	Until you delete it	Text editor
Windows Event Viewer	Application log, source <i>MergenHub</i>	Service lifecycle, unhandled crashes	Windows policy	<code>eventvwr.msc</code>



The Event Log window – filter by level, search, export, and the Application Report button

Application file log

The core writes a rolling text log to `C:\ProgramData\MergenHub\logs\`. Each line has a timestamp, level (`INF` , `WRN` , `ERR`), the source component and the message. Typical entries:

```
2026-09-05 10:14:02.117 INF Web API listening on http://0.0.0.0:8090
2026-09-05 10:14:03.402 INF OPC UA server started: opc.tcp://0.0.0.0:48010/MergenHub
2026-09-05 10:14:05.930 INF ChannelWorker Connected: Pumping_Station.PLC_1
2026-09-05 10:20:41.008 WRN ModbusTcp Read timeout 10.0.0.5:502 unit 1 (block 40001..40064),
retry 1/3
2026-09-05 10:20:44.012 ERR ChannelWorker Device Pumping_Station.PLC_1 marked Bad: 3 consecutive
failures
```

Settings › *Retention* controls how many days are kept. For a support case send the file of the day the problem happened.

Event log

Events are what an operator needs to see without reading log files: connection state changes, timeouts, service restarts, OPC UA client sessions, configuration reloads, licence/trial messages. Open it from the top-right ... menu › **Event Log**.

- Filter **All / Warning / Error**, free-text search, pause auto-refresh.
- **Export to text** saves the current view.
- **Application Report** builds a single text file with version, settings summary, channel/device health and the last events — the fastest thing to attach to a support ticket.

- Events are persisted in `events.db` for the configured number of days (default 7) and survive restarts.

Audit log

Every change made through the web UI, the REST API, OPC UA writes or an MCP tool is recorded with user, IP, time, action and target. Open *Settings* › *Audit*. Examples of what you will find:

- `channel.create`, `device.update`, `tag.delete`, `project.import`
- `tag.write` from OPC UA or REST with old and new value
- `user.login`, `user.login_failed`, `user.password_change`
- `license.activate`, `license.release`, `settings.update`

Retention is 90 days by default (*Settings* › *Retention*). Audit records are never editable from the UI.

Settings › Audit – filter chips by category, user and time; every configuration change is listed

Alarm history

Every alarm transition (raised, acknowledged, cleared) with timestamp, value, user and comment is kept for the alarm-history retention (default 90 days) and shown under *Alarms* › *History*. It is also available to OPC UA Alarms & Conditions clients.

Historian data

Log groups write to `logs.db` (SQLite) by default or to any ODBC database (SQL Server, MySQL/MariaDB, PostgreSQL, Oracle). When the target is unreachable, records are buffered on disk (**store-and-forward**) and written later. Trends, downsampling and CSV/Excel/PNG export are in the *Historian* module; see [Historian](#).

Device diagnostics and packet traces

Select a device and open **Diagnostics**: the last request/response frames in hex, byte counts, timing and the last error text. For cache-based protocols (DNP3, IEC 104, IEC 61850) the raw TX/RX tap shows the actual link traffic. A trace can be saved for analysis. See [Troubleshooting](#).

Retention settings

Settings › Retention:

SETTING	DEFAULT
Event log days	7
Audit log days	90
Alarm history days	90
File log days	rolling, automatic

MergenHub

OPC UA Gateway - v3.1.19

Ayarlar

24/24 aktif tag

Deneme 24/500

PROJE

+

Bağlantılar

Simulation Examples

24

Functions

24

Eklentiler

Virtual Tag'lar

Tag Kısıyolları

Alarmlar

Veri Logları

Data Bridge

MCP (AI)

OPC UA SUNUCU

OPC UA Sunucu

OPC UA Kullanıcıları

Güvenilen Client'lar

Sunucu Sertifikası

WEB & BAĞLANTI

Web & REST API

Data Bridge

BİLDİRİMLER

E-posta

SMS

İletişim Merkezi

GÖRÜNÜM

Marka

SİSTEM

Lisans

Denetim

Saklama

Servisler

ERİŞİM

Kullanıcılar

Detaylı yardım

Saklama

Loglar/geçmiş verilen günden eski olunca sistem OTOMATİK siler (saatlik). 0 = süresiz sakla.

Denetim Günlüğü - 7 kayıt

Kim ne yaptı (audit.db) — güvenli izi

90

≈ 3 ay

Olay Günlüğü - 10 kayıt

Sistem/iletişim olayları (events.db)

7

7 gün

Alarm Geçmişi - 0 kayıt

Geçmiş alarm olayları (raise/clear/ack)

90

≈ 3 ay

Veri Logları (historian) kendi saklama süresini her log grubunda ayrı tutar.

Kaydet

Şimdi Temizle

0 gidersen o günlük hiç silinmez.

Settings › Retention – how long events, audit and alarm history are kept

Sending logs to support

1. ... › *Event Log* › *Application Report*— save the file.
2. Add the day's `logs\mergenhub-YYYYMMDD.log` .
3. For a device problem add the *Diagnostics* trace of that device.
4. Send them through the customer portal (*Support*) or by e-mail.

Licensing and trial mode

Trial limits, online and offline activation, releasing and moving a license, subscriptions and maintenance.

Every feature of MergenHub is available in every package; packages differ only by the **number of tags** a project may contain. Licenses are perpetual or annual subscriptions, bought from the [pricing page](#) or through a dealer, and activated in the application online or offline.

Packages

PACKAGE	TAG LIMIT
Starter	1,000
Standard	5,000
Professional	25,000
Corporate	50,000
Enterprise	unlimited

Tags are counted as the total static tags in the project (channels › devices › tags including groups). Virtual tags, system tags, aliases and SCADA screens do not count.

Trial mode

An installation without a license runs in **trial mode**:

- all features enabled,
- **500 tags** maximum,
- the runtime (scan engine) runs for **2 hours after each start**; when the window expires scanning stops and tag quality becomes *Bad* until you start a new window (top-bar button, tray › *Restart runtime*, or restarting the service).

The top bar shows a *TRIAL* pill with a countdown. Entering a license key lifts both limits instantly; no reinstall or restart is needed.

Activating online

1. Buy a license; the key looks like `MH-XXXX-XXXX-XXXX-XXXX` and is shown in the customer portal and sent by e-mail.
2. In MergenHub open *Settings › License*, paste the key and click **Activate**.
3. The application contacts the license portal, binds the key to this machine's fingerprint and stores a signed license document locally.

Settings > License – status, tag usage, activation, offline files and update check

The document is refreshed automatically every few days (subscription end, maintenance status, revocation). If the portal is unreachable the license keeps working; after 14 days a warning is shown, nothing is blocked.

Activating offline (no internet on the server)

1. *Settings > License > Offline activation* — enter the key and click **Create request file** (`.mhreq`).
2. Sign in to the customer portal on any internet PC, open the license and upload the request under **Offline activation**; download the license document (`.mhlic`).
3. Back in MergenHub click **Import document** and select the `.mhlic` file.

The document is bound to the machine that created the request.

Releasing and moving a license

A license is valid on one machine at a time. To move it:

- **Online:** *Settings > License > Release* — the portal frees the key immediately; activate it on the new machine.
- **Offline:** *Release (offline)* creates a signed proof file (`.mhrel`); upload it in the customer portal under the license, then activate elsewhere.
- **Machine crashed / unavailable:** open the license in the customer portal and use *Force release* (or ask us); the old activation is blacklisted.

Hardware changes: the fingerprint tolerates one changed component (for example a replaced disk). After a major rebuild release and re-activate.

Subscriptions and maintenance

- A **subscription** license has an end date; renew it in the portal and the application picks up the new date at the next refresh (or import the new document offline). After expiry there is a 30-day read-only grace period (data keeps flowing, configuration is locked), then trial mode.
- The first year of **Maintenance & Support** is included. Without an active maintenance contract you still get every upgrade for free; only support requests are not answered.

Updates

Licensed installations check the portal once a day and show a *New version available* balloon with download and release notes links (per-version *don't show again*). *Settings › License › Check updates* runs the check on demand.

Common questions

QUESTION	ANSWER
I reached the tag limit	The API returns <i>402 Payment Required</i> with the limit and current count; upgrade the package in the portal, the new limit arrives at the next refresh.
Can I run several installations?	One license = one machine. Buy one license per server (dealers get volume discounts).
Does the license expire when maintenance ends?	No. Perpetual licenses never expire; maintenance only affects support.
Where is the license stored?	<code>C:\ProgramData\MergenHub\license\</code> — it is machine-bound, copying it elsewhere does not work.

Backup, restore, upgrade, uninstall

What to back up, project export/import, in-place upgrades, uninstalling.

Everything MergenHub owns lives under one folder, which makes backup, migration and upgrades simple.

What to back up

ITEM	LOCATION	NOTES
Whole data folder	<code>C:\ProgramData\MergenHub\</code>	Project, SCADA, users, settings, license, certificates, logs, historian
Project only	<i>Connectivity</i> › <i>Export</i> → JSON	Channels, devices, tags, alarms, log groups, virtual tags, aliases
SCADA only	<i>SCADA</i> › <i>Export project</i>	Screens, zones, symbol library, bindings
Historian in an external DB	your ODBC database	Back up with the database's own tools

Tip A nightly copy of `C:\ProgramData\MergenHub` (with the services running is fine; SQLite files are consistent) is enough to rebuild a server from scratch.

Example scheduled task (robocopy, keeps 14 daily copies):

```
robocopy "C:\ProgramData\MergenHub" "D:\Backups\MergenHub\%date:~-4%%date:~3,2%%date:~0,2%" /MIR
/R:1 /W:1 /XD logs
forfiles /P "D:\Backups\MergenHub" /D -14 /C "cmd /c rmdir /s /q @path"
```

Restore

1. Install the same or a newer MergenHub version on the target machine.
2. Stop the three services (`sc stop MergenHub`).
3. Replace `C:\ProgramData\MergenHub` with the backup (keep the new `webpki\` if the host name changed).
4. Start the services. Users, project, screens and history are back.
5. Re-activate the license on the new hardware (release it on the old machine first, or use *Force release* in the portal).

Migrating to another server

Same as restore. Points to remember:

- OPC UA clients trust the **server certificate**; a new machine has a new certificate unless you copy `pki\own`. Clients may need to trust it again.
- The web HTTPS certificate is bound to the host name; regenerate with `MergenHub.exe gencert "<name>"` if the name changed.
- Firewall rules are per machine — the installer creates them; see [Firewall](#).

Upgrading

Run the new installer. It:

- stops services and the tray app,
- replaces program files completely (no stale DLLs),
- keeps `C:\ProgramData\MergenHub` untouched,
- migrates old service names if needed,
- starts everything again.

Downtime is typically under a minute. Release notes are on the [releases page](#); licensed installations are notified inside the application. Upgrades are free for the life of the license.

Note Downgrading is not supported. Keep the previous installer and a backup if you want a way back.

Project export / import

Connectivity > Export writes the project as JSON; *Import* reads MergenHub or Kepware JSON exports. Import can **replace** the project or **merge** into it (same-named channels are replaced). Passwords inside device settings are exported encrypted with a project key; importing on another server asks for the key.

Uninstalling

Settings > Apps or the uninstaller in the program folder. It stops and removes the three services, removes the firewall rules and the tray autostart, and deletes program files. **The data folder is kept** so a later reinstall finds everything again. Delete `C:\ProgramData\MergenHub` by hand if you want a completely clean machine.

Channels, devices and tags

The project hierarchy, scan rates and scan modes, scaling, tag groups, addressing helpers, tag generation.

The project is a three-level tree: **channel** → **device** → **tag**. Everything else (alarms, log groups, virtual tags, screens) refers to tags by their full path `Channel.Device.Tag`.

The screenshot shows the MergenHub interface. On the left, the project hierarchy is displayed under 'PROJE'. It includes 'Bağlantılar' (Connections), 'Simulation Examples' (24), 'Functions' (24), and 'Ekleniler' (Added). Under 'Ekleniler', there are 'Virtual Tag'lar, 'Tag Kısıyolları' (Tag Restrictions), 'Alarmlar' (Alarms), 'Veri Logları' (Data Logs), 'Data Bridge', and 'MCP (AI)'. The main area shows 'Simulation Examples cihazları' (Simulation Examples devices) with a table listing 'Ad' (Name), 'Model', 'ID', and 'Tag'. The table has one row: 'Functions' with ID '4' and Tag '24'. At the bottom, there is a user profile for 'admin' (Root (super-admin)) and a language selector set to 'TR TR'.

Connectivity – channel list, device tree and the live tag table

Channels

A channel is one driver instance with its transport settings. Create one per physical connection:

DRIVER FAMILY	ONE CHANNEL PER...
Modbus TCP, S7, OMRON, DNP3, IEC 104, IEC 61850, OPC UA client	network segment or group of devices you want isolated
Modbus RTU/ASCII	serial port (COM1, COM2 ...) or Ethernet-encapsulation gateway
MQTT	broker connection
Simulator	as many as you like

Channel settings include network adapter binding, connection/request timeouts, retry count, inter-request delay and the driver-specific options (e.g. Modbus block size, S7 PDU size). Channels are **isolated**: a device that stops answering never slows down devices on another channel.

Devices

A device is one PLC, RTU, IED, meter or broker topic set. Key settings:

- **ID / address** — IP:port and unit ID (Modbus), IP + rack/slot (S7), IED address (61850), station addresses (DNP3/104), topic root (MQTT).
- **Scan rate** — how often the device is polled (ms). Tags inherit it unless they override.
- **Scan mode** — *respect client rate* (OPC UA subscribers can request faster/slower), *fixed*, or *client-driven*.
- **Timeouts and retries** — after the configured failures the device is marked *Bad* and retried with back-off; it recovers automatically.
- **Demand-driven scanning** — a device is polled only while some consumer needs its tags; idle devices are left alone (saves bandwidth on GSM/radio links).
- **Diagnostics** — hex request/response trace, counters, last error.
- **Copy** — duplicates a device with all tags (addresses can be shifted).

The screenshot displays the MergenHub OPC UA Gateway v3.1.19 interface. The left sidebar contains navigation options: PROJE, Bağlantılar (Simulation Examples, Functions), Eklentiler (Virtual Tag'ler, Tag Kısayolları, Alarmlar, Veri Logları, Data Bridge, MCP (AI)), and a user profile for 'admin'. The main panel shows system status with a 'Sunucu Kaynakları' section displaying CPU (0%), RAM (218 MB / 0.7%), and Thread (28 thread - 842 handle) usage. Below this, a 'MAKİNE (TOPLAM SİSTEM)' section shows CPU (12% - 32 çekirdek), RAM (22941 / 32530 MB (%70.5)), and Disk (144.8 GB boş / 930.6 GB) usage. A 'UYARI EŞİKLERİ' (Warning Thresholds) section lists CPU ≥ 85%, RAM ≥ 85%, Disk uyarı < 5 GB, and Disk kritik < 1 GB. The 'Servisler & Alt Sistemler' section lists services: OpcUaServer (Çalışıyor), ScanEngine (Çalışıyor), Historian (Veri Logları) (Boğta), and Alarmlar (Çalışıyor). The bottom status bar indicates 'Settings > Services shows every channel worker and its health at a glance'.

Tags

A tag maps an address in the device to a typed value:

FIELD	MEANING
Name	unique inside the device; used in OPC UA node ids and REST paths
Address	driver-specific (<code>40001</code> , <code>DB10.DBD24</code> , <code>D100</code> , <code>ns=2;s=...</code> , <code>AI:3</code> , <code>I0A 1001</code> , <code>LD0/MMXU1.TotW.mag.f</code> , <code>plant/line1/temp</code>) — the ? shows the formats
Data type	Boolean, Byte, Short, Word, Long, DWord, LLong, QWord, Float, Double, String, Date, BCD/LBCD, arrays where the driver supports them
Read/Write	read-only or read/write (writes go to the device immediately and are audited)
Scaling	linear or square-root scaling with raw/scaled ranges and clamping; writes are un-scaled
Scan rate	optional override of the device rate
Description / unit	shown in OPC UA (Description, EngineeringUnits) and the UI

Tag groups organise large devices into folders and appear as OPC UA folders.

Creating many tags at once

- **Generate tags** — from an address range (start, count, step, naming pattern).
- **Online browse** — reads the tag list from the device: OPC UA servers, IEC 61850 (SCL/ICD import or live GVA walk), DNP3 integrity poll, IEC 104 general interrogation, OMRON NJ/NX symbols, MQTT topic discovery with JSON pointers.
- **Import** — Kepware or MergenHub JSON exports; CSV via the tag table's import button.

Quality and timestamps

Every value carries an OPC quality: **Good**, **Bad** (device unreachable, error, or trial window expired), **Uncertain** (stale/cached). Both the *source timestamp* (device time when available) and the *server timestamp* are kept and exposed over OPC UA.

Aliases

Extensions › *Aliases* let you publish a tag under a second, friendlier name (for example `Plant.Intake.Flow` instead of `Modbus_1.RTU_3.AI_40011`). Aliases appear in OPC UA and REST like normal tags.

System tags

Each device has `_System` tags (`_Error` , `_NoError` , `_LastRead` , `_SuccessfulReads` , `_Enabled` ...) that report communication health and can be logged or alarmed like any tag. `_Enabled` can be written to pause a device.

Protocol drivers

Modbus, Siemens S7, OMRON, DNP3, IEC 104, IEC 61850, MQTT, OPC UA client, simulator — what each supports.

All drivers ship in every installation; there is nothing to buy or enable. Each driver has an *Address hints* panel in the tag editor and a help page inside the application.

DRIVER	TRANSPORT	HIGHLIGHTS	DETAIL PAGE
Modbus	TCP, UDP, RTU/ASCII serial, Ethernet encapsulation	FC01–04 reads, FC05/06/15/16/22 writes, zero/one-based addressing, Modicon 5/6-digit, word/bit order, per-area max block size, one socket per channel	Modbus
Siemens S7	ISO-on-TCP (102)	S7-300/400/1200/1500, DB/M/I/Q/T/C areas, optimised PDU packing, 64-bit types	Siemens S7
OMRON	FINS/TCP, EtherNet/IP CIP	CS/CJ/CP FINS memory areas, NJ/NX symbolic tags (CIP), bit and word access	OMRON
DNP3	TCP	Integrity/event polls, unsolicited, SBO/Direct Operate, Secure Authentication, time sync, file control, redundancy	DNP3
IEC 60870-5-104	TCP	General interrogation, single/double commands, setpoints, time sync, redundancy, RTU compatibility modes	IEC 104
IEC 61850	MMS	SCL/ICD import, LN-level reads, control model (SBO/direct), reports, VMD addresses	IEC 61850
MQTT	TCP/TLS	Broker per channel, topic devices, JSON/XML/CSV/key-value payload parsing, discovery, arrival watchdog liveness	MQTT
OPC UA client	opc.tcp	Connect to other UA servers (PLC, gateway, another MergenHub); channel = session, device = subscription, browse for tags	—
Simulator	—	Ramp, sine, random, static functions for testing screens, alarms and logging without hardware	—

Channel and device limits

Every driver limits the number of channels it can open and the number of devices in one channel. The values match the limits documented by the industry-standard OPC server for each driver, with channels additionally capped at 1024. The limit is **per driver**, not per project — channels of different drivers add up beyond it.

DRIVER	MAX CHANNELS	MAX DEVICES PER CHANNEL
Modbus TCP/IP Ethernet	1024	8192
Modbus RTU Serial	1024	255
Modbus ASCII Serial	100	247
DNP3 Client Ethernet	1024	1024

DRIVER	MAX CHANNELS	MAX DEVICES PER CHANNEL
IEC 60870-5-104 Client	1024	1024
IEC 61850 MMS Client	256	256
MQTT Client	60	1024
OMRON FINS Ethernet	512	32
OMRON NJ Ethernet	256	1024
Siemens S7 TCP/IP	1024	1024
OPC UA client	256	256
Simulator	1024	999

- Limits are checked only when **adding** (UI, REST API, AI assistant, copy, import). When a limit is reached the operation is rejected and the project is left unchanged; an import is rejected as a whole, never partially applied.
- A saved project is never rejected or trimmed when it loads.
- Every value was exercised at the limit itself against a running MergenHub with protocol simulators, including all drivers at their limits **at the same time** (7,584 channels): every tag stayed Good and kept updating.
- A channel polls its devices **one after another**, so a crowded channel has a longer cycle (in testing, 8,192 Modbus TCP devices in one channel refreshed about every 40 s). Split devices that need fast updates across several channels — channels run in parallel.

Choosing timeouts and scan rates

- Start with the driver defaults and the device's documented response time. A scan rate lower than the device round-trip only creates back-pressure.
- Serial and GSM/radio links: longer request timeout (2–5 s), fewer retries, and demand-driven scanning so idle devices do not keep the link busy.
- Large Modbus devices: keep *max block size* below what the device supports (many RTUs reset above 64 registers).

Writes

Writes are executed **immediately**, ahead of the scan queue, and are verified: the driver checks the device acknowledged the write (Modbus exception codes, S7 return codes, DNP3/104 command responses). Failed writes are reported to the caller (OPC UA StatusCode / REST error) and logged in the audit log with user and value.

Redundancy and recovery

Every channel runs under a supervisor. Broken sockets are reopened, dead workers are recreated, and quarantine/back-off prevents a flapping device from flooding the log. DNP3 and IEC 104 support dual-address redundancy.

Diagnostics

Select a device › **Diagnostics** to see the raw frames (TX/RX hex), byte counters and last error. For cache-based protocols (DNP3, 104, 61850) the packet tap shows the real link traffic, not the cached values. See [Troubleshooting](#).

OPC UA server

Endpoints, security policies, certificates, the address space, system tags, history access, alarms over UA.

MergenHub is an **OPC UA server** (Data Access, Historical Access, Alarms & Conditions, Methods) that presents every channel, device and tag in one address space. Any UA client — SCADA, MES, historian, PLC, another MergenHub — connects to it.

```
opc.tcp://<host>:48010/MergenHub
```

The screenshot displays the MergenHub OPC UA Gateway - v3.1.19 interface. The left sidebar shows the project structure with 'Bağlantılar' (Connections) and 'Ekentiler' (Extensions). The main area is titled 'OPC UA Sunucu' (OPC UA Server) and contains the following sections:

- OPC UA Sunucu**: A section for managing the server, including 'OPC UA Kullanıcıları' (OPC UA Users), 'Güvenilen Client'lar' (Trusted Clients), and 'Sunucu Sertifikası' (Server Certificate).
- WEB & BAĞLANTI**: A section for web and connection settings, including 'Web & REST API' and 'Data Bridge'.
- BİLDİRİMLER**: A section for notifications, including 'E-posta' (Email), 'SMS', and 'İletişim Merkezi' (Communication Center).
- GÖRÜNÜM**: A section for appearance settings, including 'Marka' (Brand).
- SİSTEM**: A section for system settings, including 'Lisans' (License), 'Denetim' (Audit), 'Saklama' (Storage), and 'Servisler' (Services).
- ERİŞİM**: A section for access settings, including 'Kullanıcılar' (Users).

The 'OPC UA Sunucu' section is currently active, showing the 'Açık' (Open) status. The 'Endpoint yolu' (Endpoint path) is set to 'MergenHub'. The 'SUNUCU ENDPOINT'LERİ' (Server Endpoints) section shows a list of endpoints, including 'opc.tcp://localhost:48020/MergenHub' with security policies 'Basic256Sha256 - Sign' and 'Basic256Sha256 - Sign & Encrypt'. The 'Güvenlik' (Security) section shows the 'devralınan' (Inherited) security policy. The 'OPC UA PROJE ÖZELLİKLERİ' (OPC UA Project Properties) section includes checkboxes for 'Log diagnostics (stack tanılama — yalnız sorun giderme)' and 'Alarmları OPC UA'dan yayınla (A&E — _Alarms koşulları)'.

Settings > OPC UA Server – endpoints, security policies, certificates and project properties

Address space

```
Objects
├─ <Channel>
│   └─ <Device>
│       ├── <Tag> (Variable: value, quality, source/server timestamps, EngineeringUnits,
│       │         Description)
│       │   ├── <TagGroup>/...
│       │   └─ _System (_Error, _NoError, _LastRead, _SuccessfulReads, _Enabled ...)
│       ├── _Aliases
│       ├── _AdvancedTags
│       ├── _Alarms (when "publish alarms over OPC UA" is on)
│       └─ _System (server-wide statistics)
```

Node ids are string-based (`ns=2;s=Channel.Device.Tag`) so they stay stable across restarts and exports.

Endpoints

The server can listen on several endpoints at once (*Settings › OPC UA Server › Endpoints*): for example `localhost` without security for a local client and the LAN/public host with *Basic256Sha256 – Sign & Encrypt*. Each endpoint has its own host, port and allowed security policies. Changing endpoints takes effect after a runtime restart.

Security

SETTING	RECOMMENDATION
Security policies	None only on localhost; Basic256Sha256 Sign & Encrypt elsewhere
Allow anonymous	off in production
User authentication	<i>Settings › OPC UA Users</i> : user/password accounts with roles; certificates supported
Auto-accept untrusted certificates	off in production; trust clients in <i>Settings › Trusted clients</i>
Server certificate	generated at install; regenerate when the host name changes (<i>Settings › Server certificate</i>)

Per-user **resource ACL** applies to OPC UA too: a user who cannot see a channel in the UI does not see it in the address space, and writes require the write permission. See [Users and security](#).

Session and subscription limits

Defaults: 128 sessions, session timeout 20 min, up to 450K monitored items proven. Publishing intervals down to 100 ms; the server honours client-requested sampling rates (scan mode *respect client rate*) so subscribers can drive polling.

History access

Tags that belong to a historian log group answer **HistoryRead** (raw and processed aggregates) directly from the historian database, so UA clients can plot trends without a separate historian. See [Historian](#).

Alarms & Conditions

With *Publish alarms over OPC UA* enabled, every alarm definition becomes an `AlarmConditionType` under `_Alarms` ; clients subscribe to the Server object for events, use `ConditionRefresh` and `Acknowledge` . State mapping follows ISA-18.2 (active/acked/cleared). See [Alarms](#).

Methods

Server-side methods are exposed for runtime control (restart a channel, force a read) to authorised users only.

Connecting common clients

- **UaExpert** — Add server › enter the endpoint URL › choose security › trust the server certificate; if *auto-accept* is off, trust the UaExpert certificate in *Settings › Trusted clients*.
- **Keeware / Ignition / Siemens WinCC** — use the same endpoint; for signed sessions import MergenHub's certificate (`pki\own`) into the client's trust store.
- **Another MergenHub** — use the *OPC UA client* driver with this server as the channel.

Troubleshooting

SYMPTOM	FIX
<code>BadCertificateHostNameInvalid</code>	Client connects with a name not in the certificate — add an endpoint with that host and regenerate the server certificate
<code>BadSecurityChecksFailed</code> / <code>BadCertificateUntrusted</code>	Trust the client certificate (<i>Settings › Trusted clients</i>) or enable auto-accept temporarily
<code>BadIdentityTokenRejected</code>	Anonymous disabled and wrong user/password
Client sees Bad quality	Device offline or trial window expired — check <i>Connectivity</i> and the Event log
Server "not running" in the UI	Port in use by another OPC server; change the port in <i>Endpoints</i>

REST API

Authentication, configuration API, live values, writes, Swagger, examples in curl.

Everything the web UI does goes through the same REST API, so anything you can click you can script. The API is JSON over HTTP(S) on the web port (default 8090) and documented live in the application (... › *REST API*, Swagger UI).

Settings › Web & REST API – port, HTTPS, CORS and API access options

Authentication

Two ways:

1. **Session cookie** — `POST /auth/v1/login` with `{"username":"...", "password":"..."}` returns a cookie; send it with every request. Sessions follow the security policy (idle timeout, single/multi login).
2. **API token** — create a token for a user in *Settings › API tokens* and send `Authorization: Bearer <token>`. Tokens carry the user's role and resource ACL; revoke them any time.

```
curl -c c.txt -X POST https://scada.example.com/auth/v1/login \
-H "Content-Type: application/json" -d '{"username":"apiuser", "password":"...."}'
```

Main endpoint groups

GROUP	EXAMPLES	PURPOSE
/auth/v1/*	login , me , logout , forgot	authentication
/config/v1/project/*	channels , channels/{ch}/devices , .../tags (GET/POST/PUT/DELETE)	project configuration, same model as the JSON export
/config/v1/settings	GET/PUT	server settings
/config/v1/license/*	activate, refresh, release, update-check	licensing
/config/v1/scada/*	screens, library, symbols	SCADA screens
/config/v1/advanced , /aliases , /alarms , /logger	CRUD	extensions
/runtime/v1/values? tags=A.B.C,...	GET	live values with quality and timestamps
/runtime/v1/write	POST {"tag": "A.B.C", "value": 42}	write to a device (audited)
/runtime/v1/status , /metrics , /services/*	GET/POST	health, CPU/RAM, service control
/runtime/v1/events , /audit	GET	event and audit logs
/logger/v1/history?tag=... &from=...&to=...	GET	historian data
/iotgateway/*	GET/POST	Kepware-compatible Data Bridge REST server (see Data Bridge)

Configuration writes return **409** on version conflicts (someone else changed the project), **402** when the license tag limit would be exceeded, **403** when the resource ACL denies access.

Examples

Read three tags:

```
curl -b c.txt "http://localhost:8090/runtime/v1/values?  
tags=Plant.PLC1.Flow,Plant.PLC1.Pressure,Plant.PLC1.Pump1_Run"
```

```
[{"tag": "Plant.PLC1.Flow", "value": 412.6, "quality": "Good", "ts": "2026-09-05T10:20:41.008Z"}, ...]
```

Write a setpoint:

```
curl -b c.txt -X POST http://localhost:8090/runtime/v1/write \  
-H "Content-Type: application/json" -d '{"tag": "Plant.PLC1.Setpoint", "value": 55}'
```

Create a device with tags in one call:

```
curl -b c.txt -X POST http://localhost:8090/config/v1/project/channels/Plant/devices \
-H "Content-Type: application/json" \
-d '{ "name": "PLC2", "idString": "10.0.0.6:502", "unitId": 1, "scanRateMs": 1000,
      "tags": [ { "name": "Flow", "address": "40001", "dataType": "Float" },
                { "name": "Run", "address": "00001", "dataType": "Boolean", "readWrite": "RW" } ] }'
```

Export / import the whole project:

```
curl -b c.txt http://localhost:8090/config/v1/project/export -o project.json
curl -b c.txt -X POST http://localhost:8090/config/v1/project/import?mode=merge -H "Content-Type:
application/json" --data-binary @project.json
```

Rate limits and safety

- Login attempts are rate-limited (429 after repeated failures).
- Every mutating call is written to the audit log with user, IP and payload summary.
- Live-value reads are cheap (in-memory); avoid polling faster than the device scan rate — subscribe over OPC UA or MQTT instead for high-rate data.

Swagger / OpenAPI

Open ... › *REST API* in the application for the interactive OpenAPI document (login required). It lists every route with schemas and lets you try calls with your session.

Data Bridge (IoT gateway)

REST server, REST client and MQTT client agents; payload formats; Kepware-compatible endpoints.

The Data Bridge (formerly IoT Gateway) moves data **out of** MergenHub to clouds, brokers and third-party systems, and lets simple HTTP clients read or write tags without OPC UA. It runs as its own Windows service (`MergenHubDataBridge`), so a slow cloud endpoint can never affect scanning.

Extensions > Data Bridge – agent list and the agent editor

Agents

AGENT	DIRECTION	WHAT IT DOES
REST server	inbound	Kepware-compatible HTTP endpoints (<code>/iotgateway/read</code> , <code>/iotgateway/write</code> , <code>/iotgateway/browse</code>) with the same JSON shapes as Kepware's IoT Gateway, so existing integrations work unchanged
REST client	outbound	Pushes selected tags to a URL (POST/PUT) on change or on a schedule; JSON/XML/CSV/key-value formats; custom headers and auth
MQTT client	outbound	Publishes selected tags to a broker (QoS, retain, TLS, username/password or certificate); JSON payloads, one topic per tag or grouped; optional Sparkplug-style templates

Each agent has its own tag list, publish rate, *on change* / *deadband* filters and a status page with counters and the last error.

REST server quick start

1. Add a *REST server* agent, choose the tags, allow writes if needed.
2. Log in (session cookie or API token) and read:

```
curl -b c.txt "http://localhost:8090/iotgateway/read?ids=Plant.PLC1.Flow&ids=Plant.PLC1.Pressure"
```

```
{"readResults":[{"id":"Plant.PLC1.Flow","s":true,"r":"","v":412.6,"t":1757067641008}, ...]}
```

1. Write (when *allow write* is on):

```
curl -b c.txt -X POST http://localhost:8090/iotgateway/write -H "Content-Type: application/json" \
-d ' [{"id":"Plant.PLC1.Setpoint","v":55}] '
```

MQTT client quick start

- Broker `mqtt://broker.example.com:1883` (or `mqtts://` with TLS), client id, credentials.
- Topic template such as `plant/{channel}/{device}/{tag}` or a single topic with a JSON array.
- Publish mode: *on change* (with deadband) or *every N ms*.
- Payload example:

```
{"id":"Plant.PLC1.Flow","v":412.6,"q":"Good","t":"2026-09-05T10:20:41.008Z"}
```

Reliability

- Agents buffer when the destination is down and flush when it returns (store-and-forward in memory with a bounded queue).
- The agent status shows *connected* / *retrying* and the last error text; failures are also in the [Event log](#).
- Because the bridge is a separate process, restarting it (*Settings* › *Services*) does not interrupt OPC UA or scanning.

Security notes

- The REST server uses the same users, tokens and resource ACL as the main API; only tags the user may read are returned.
- Prefer HTTPS and API tokens for external systems; never expose the web port without TLS.
- MQTT: use TLS and per-device credentials on the broker; MergenHub validates the broker certificate unless you disable it for testing.

Advanced (virtual) tags

Expression engine, average/min/max, totalizer, derived, link, Oracle cumulative tags; function reference.

Advanced (virtual) tags compute new values from existing tags. They live under *Extensions* › *Virtual tags*, appear in OPC UA under `_AdvancedTags`, and can be logged, alarmed and shown on screens like any other tag.

Extensions › Virtual tags – list and expression editor

Types

TYPE	PURPOSE	NOTES
Expression (Derived)	any formula over input tags	evaluated on change or at a fixed period; math, logic, comparison, string and time functions
Average	rolling or window average	run-gated: only accumulates while a <i>run</i> tag is true
Minimum / Maximum	min/max over a window	resettable, run-gated
Totalizer	integrates a rate over time (m ³ /h → m ³)	units per hour/minute/second, persistent across restarts, reset tag
Cumulative (Oracle-style)	difference-based counters	handles rollover

TYPE	PURPOSE	NOTES
Link	copies a tag to another (write-back)	for mapping between devices or protocols
Trigger	raises a boolean pulse on conditions	for screens and Data Bridge

Expression language

```
IF(Plant.PLC1.Flow > 100 AND Plant.PLC1.Pump1_Run, Plant.PLC1.Flow * 1.05, 0)
ROUND(AVG(Tank1.Level, Tank2.Level), 1)
CLAMP((Raw - 4) / 16 * 100, 0, 100)           // 4-20 mA to percent
LERP(0, 100, Raw / 65535)                     // linear interpolation
POLY(Raw, 0.0012, -0.3, 12.5)                 // polynomial
MAP(Raw, 0, 1023, 0, 10)                      // range mapping
```

Available functions include arithmetic, `ABS` `ROUND` `FLOOR` `CEIL` `MIN` `MAX` `AVG` `SUM`, trigonometry, `IF` `AND` `OR` `NOT`, comparisons, string functions, `NOW()` `HOUR()` `MINUTE()`, `PREV(tag)` (previous value), `AGE(tag)` (seconds since last update), `QUALITY(tag)`, `RANDOM()`. The editor has *Validate* which checks syntax and shows a live evaluation.

Quality propagation

If any input is *Bad*, the result is *Bad* unless the expression handles it explicitly with `QUALITY()`. Missing inputs are reported in the tag's status column and in the event log.

Persistence

Totalizers and cumulative tags save their state to disk every 30 s and on shutdown, so a restart or power loss does not reset accumulated volumes.

Performance

Virtual tags evaluate in the core process on their own schedule; thousands of them are fine. Prefer *on change* evaluation for expressions that depend on slowly changing tags.

Alarms and notifications

ISA-18.2 lifecycle, conditions, deadband, e-mail and SMS, per-user preferences, alarm history.

The alarm module implements the **ISA-18.2** lifecycle (active → acknowledged → cleared), evaluates conditions on live tags, notifies people by e-mail and SMS, keeps history, and publishes alarms to OPC UA clients.

Alarms – active list, definitions and history tabs

Defining an alarm

Extensions › Alarms › + Alarm:

FIELD	MEANING
Source tag	the tag to evaluate (numeric or boolean)
Condition	<code>> >= < <= == !=</code> a limit, <i>between/outside</i> a range, boolean <i>true/false</i> , <i>rate of change</i> , <i>deviation from another tag</i> , <i>bad quality</i>
Deadband	hysteresis so a value hovering around the limit does not flap
On delay / off delay	seconds the condition must persist before raising / clearing
Severity / priority	Low, Medium, High, Critical (1–1000 for OPC UA)

FIELD	MEANING
Area › Source	organisation like WinCC (plant › unit › equipment) for filtering
Message	text with placeholders {value} {limit} {tag} {unit}
Auto-acknowledge	clear without operator ack
Notifications	which groups/users get e-mail and/or SMS, with optional escalation delay

Operator view

The *Active* tab lists raised alarms with colour by severity, time, value and an **Acknowledge** button (comment optional). The bell in the top bar shows the active count everywhere in the application; SCADA screens show alarm badges on bound objects and an alarm banner widget.

Notifications

- **E-mail** — SMTP settings in *Settings › E-mail*; templates per severity.
- **SMS** — pluggable providers (NetGSM and generic HTTP) in *Settings › SMS*.
- **Per-user preferences** — each user chooses which severities and areas they want by e-mail/SMS and quiet hours.
- **Communication centre** — test messages, templates, invitations and bulk messages (*Settings › Communication*).

MergenHub
OPC UA Gateway - v3.139

PROJE

+

Bağlantılar

Simulation Examples

24

Functions

24

Ekentiler

Virtual Tag'ler

Tag Kısayolları

Alarmlar

Veri Logları

Data Bridge

MCP (AI)

admin

Root (super-admin)

TR TR

OPC UA SUNUCU

OPC UA Sunucu

OPC UA Kullanıcıları

Güvenilen Client'lar

Sunucu Sertifikası

WEB & BAĞLANTI

Web & REST API

Data Bridge

BİLDİRİMLER

E-posta

SMS

İletişim Merkezi

GÖRÜNÜM

Marka

SİSTEM

Lisans

Denetim

Saklama

Servisler

ERİŞİM

Kullanıcılar

Alarm bildirimlerini e-posta ile göndermek için SMTP ayarları. [Detaylı yardım](#)

E-posta

Alarm bildirimleri için SMTP gönderici ayarları.

E-posta gönderimi etkin

SMTP Sunucu

smtp.gmail.com

Port

587

SSL/TLS kullan

☒

Kullanıcı adı

Parola

Gönderen adres

alarm@firma.com

Gönderen ad

MergenHub

TEST

test alıcı e-posta

Test Gönder

Not: port/bind/güvenlik değişiklikleri yeniden başlatmada uygulanır.

Kaydet

Settings › E-mail – SMTP server, sender and test message

MergenHub User Guide - mergenhub.envest.com.tr/docs

49 / 112

History

Every transition (raised, acknowledged, cleared, notification sent) is stored with timestamp, value, user and comment for the configured retention (default 90 days) and can be exported. See [Logs](#).

OPC UA Alarms & Conditions

Enable *Publish alarms over OPC UA* in *Settings › OPC UA Server*. Each definition becomes an `AlarmConditionType` under `_Alarms` ; UA clients receive events, refresh conditions and acknowledge. Acknowledgements from UA are audited with the UA user name.

Tips

- Use `_System._NoError` alarms per device to be notified about communication loss.
- Combine *bad quality* conditions with an off delay to avoid alerts on short GSM dropouts.
- For KPI alarms create a virtual tag first and alarm on that.

Historian (data logger)

Log groups, triggers, SQLite or ODBC targets, store-and-forward, trends and export.

The historian records tag values into a database and serves trends, exports and OPC UA HistoryRead. It runs as its own service (`MergenHubHistorian`) and reads live values from the core over the local API.

Extensions › Historian – log groups and the trend viewer

Targets

TARGET	WHEN TO USE
SQLite (<code>logs.db</code> , default)	single server, up to a few hundred million rows; zero setup
ODBC	SQL Server, MySQL/MariaDB, PostgreSQL, Oracle — corporate databases, reporting tools, long retention

Settings › Historian (or the log group) holds the connection; the *Browse* button discovers SQL Server instances on the network, lists databases and can create the table for you.

Log groups

A log group is a set of tags with one storage rule:

- **Trigger** — periodic (every N ms), on change (with deadband), on a trigger tag edge, or a time window (e.g. 06:00–22:00). Advanced latched triggers match Kepware's behaviour.
- **Table format** — *narrow* (one row per tag sample) or *wide* (one column per tag), with selectable column mapping.
- **Table mode** — single table, table per day/month, or fixed table with automatic creation.
- **Batch size / flush interval** — tune for the database.

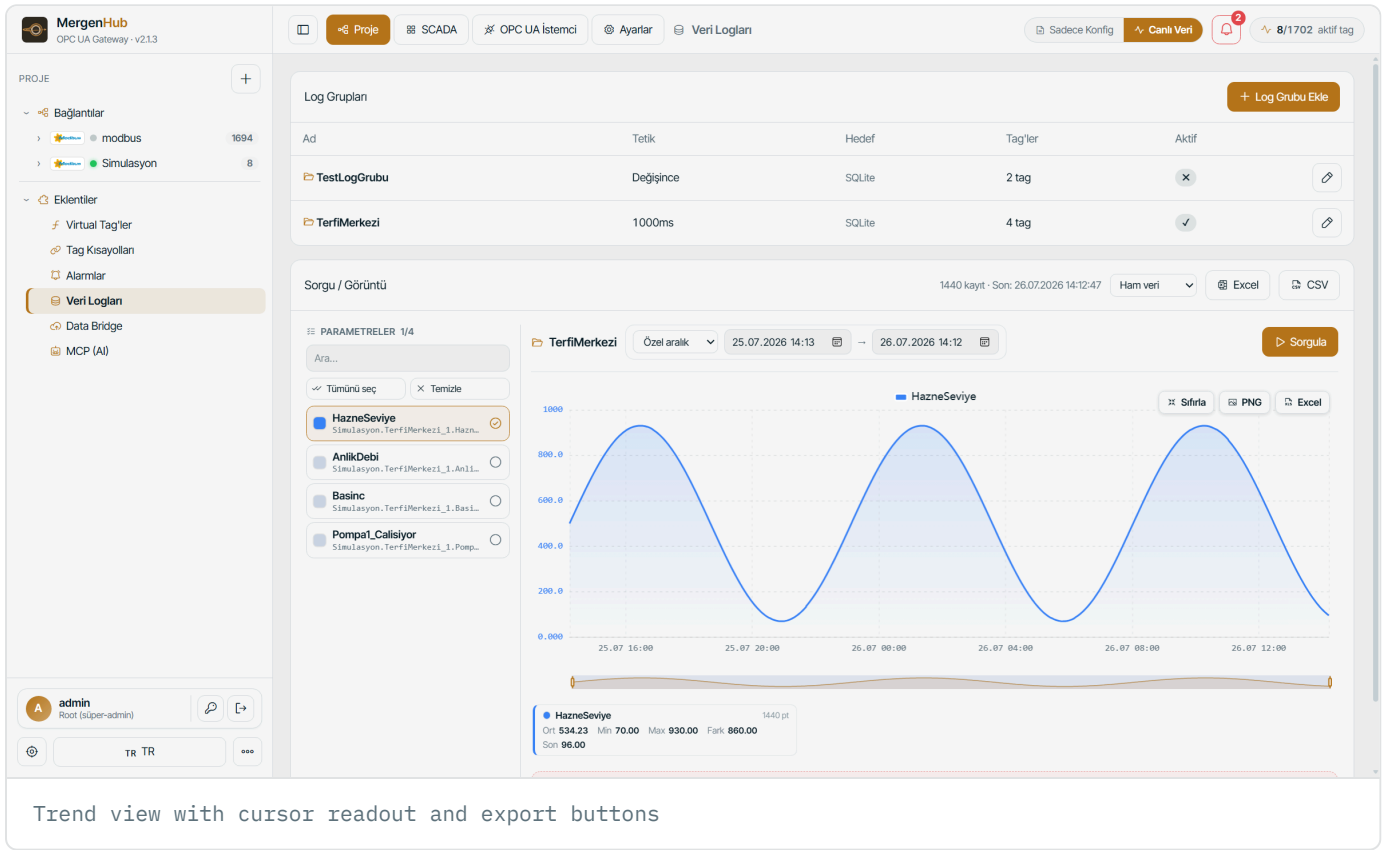
Store-and-forward

When the target is unreachable the historian buffers samples on disk and writes them when the connection returns; nothing is lost during database maintenance or network outages.

Trends and export

Historian > Trend: pick tags and a time range, zoom, pan, compare, then export as **CSV**, **Excel** or **PNG**.

Downsampling (averages/min/max per bucket) keeps long ranges fast.



OPC UA HistoryRead

Tags inside a log group answer HistoryRead requests (raw values and processed aggregates such as average/min/max/interpolative) so UA clients can chart history without direct database access.

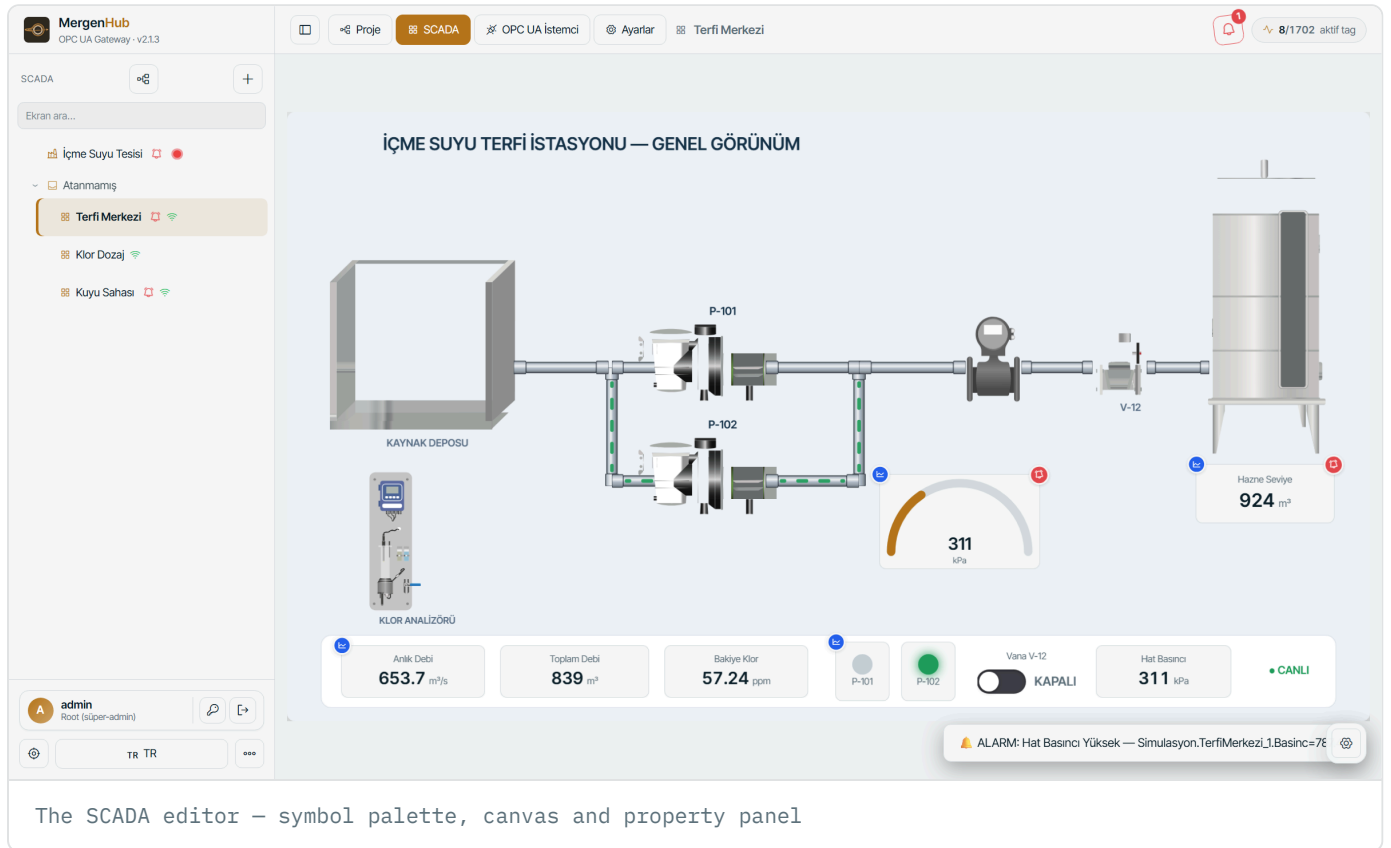
Sizing

Narrow SQLite storage costs about 40–60 bytes per sample. 1,000 tags every 10 s $\approx$ 8.6 M samples/day $\approx$ 0.4–0.5 GB/day. Use on-change logging with deadbands for slowly moving values, and an ODBC database for multi-year retention.

SCADA/HMI

Editor, symbol libraries, pipes and flow animation, bindings, faceplates, operator view, templates.

The SCADA/HMI module is a browser-based screen editor and runtime. Screens are stored on the server (`scada.json`), edited in the browser and viewed by operators on any device — no client installation.



The SCADA editor — symbol palette, canvas and property panel

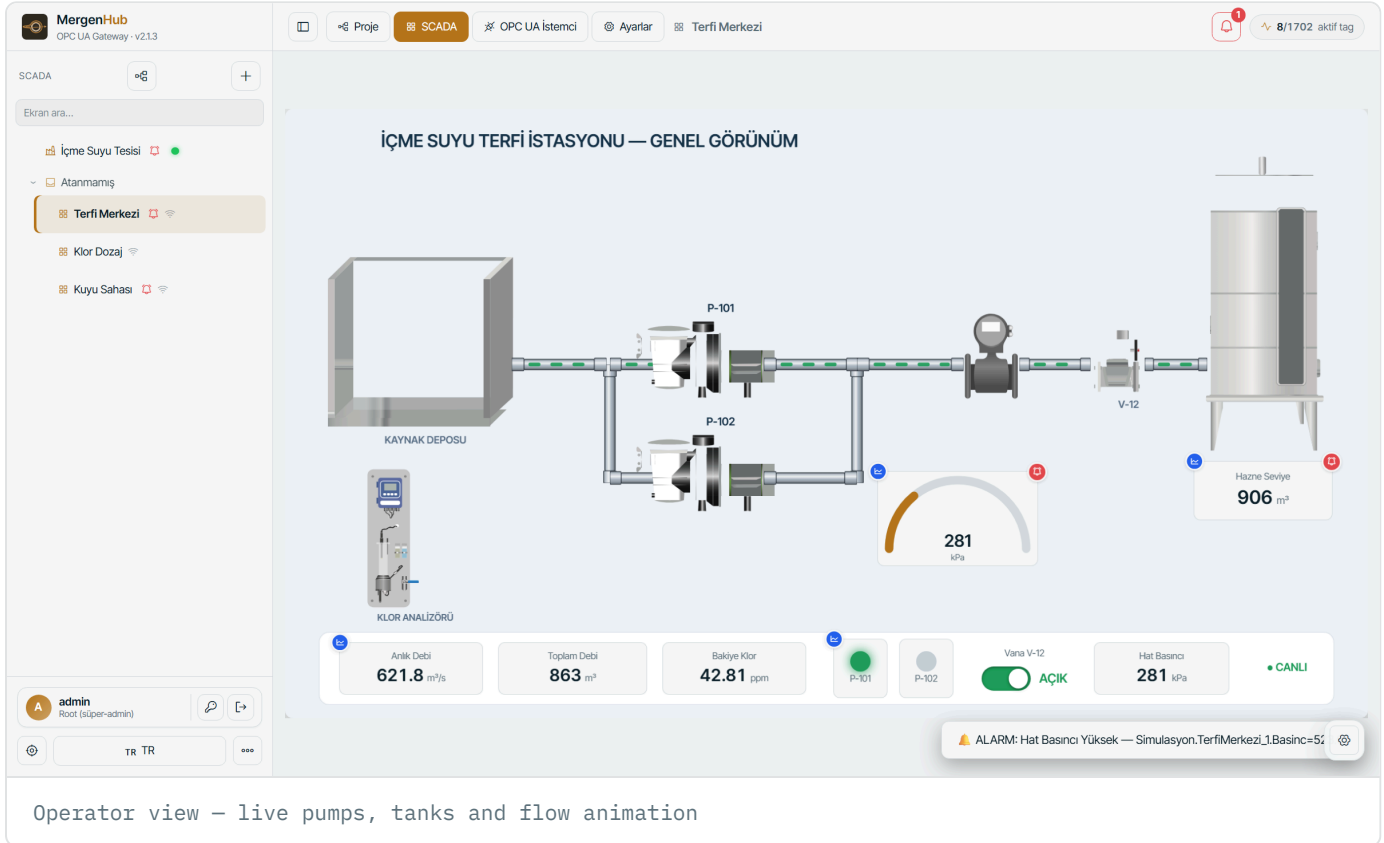
Editor basics

- **Screens and zones** — screens are organised in a zone tree (plant > area > unit); zones also drive navigation and access control.
- **Widgets** — tanks, pumps, valves, motors, gauges, trends, live charts, tables, buttons, text, inputs, images, maps (Web Mercator with GPS-positioned objects), pipes and polyline pipes, symbol library objects (built-in industrial library, WinCC-style dynamic symbols, imported SVG).
- **Bindings** — every property (fill level, colour, visibility, rotation, text, position) can be bound to a tag or an expression.
- **Animations** — rules per object: colour on condition, blink, hide/show, rotate, move, **flow** (animated stripes along pipes and inside symbol shapes, with direction and speed), scale.
- **Groups and isolation** — group objects, double-click to edit inside a group (Illustrator-style), align/distribute, undo/redo, copy/paste between screens.

- **Sub-screens and faceplates** — click actions open another screen, a popup faceplate bound to a device, a trend, or write a value (with confirmation).
- **Templates** — ready-made water/energy screens (well field, pumping station, reservoir, chlorine dosing...) that generate channel, device, tags and a working screen from an IP address.

Runtime (operator view)

Press **Run** or open the screen from the operator menu. Values update live, alarm badges show on bound objects, inputs and buttons write to tags according to the user's permissions, and the view works on tablets and phones with touch pan/zoom.



Symbol libraries

- Built-in library: pumps, valves, tanks, motors, pipes, electrical single-line elements, sensors, containers, 3D-style objects.
- **Import SVG** — bring your own drawings; ids are namespaced automatically so several instances never conflict.
- **My objects** — save any group as a reusable object; the library is server-side and per account.
- WinCC `.svghmi` dynamic symbols can be imported with their bindings.

Permissions

Zones and screens honour the resource ACL: operators see only the screens of zones they may read, and writes need write permission on the tag. Editing screens requires the SCADA edit permission.

Export / import

SCADA > Export project writes screens, zones and the user library as one JSON file; *Import* restores it on another server (bindings keep their tag paths).

Tips

- Use the *flow* animation on pipe widgets with `Flow animation` enabled and bind it to a pump-run tag.
- Keep screen backgrounds light for control rooms; use the theme switch for dark rooms.
- Bind `_System._NoError` to a status lamp per device to make communication loss visible on the screen.

Users, groups and security

Roles and permissions, resource ACL, HTTPS, session policy, audit, hardening checklist.

MergenHub has one user model for the web UI, the REST API, OPC UA and the Data Bridge: **users** belong to **groups**, groups and users have **roles/permissions**, and a **resource ACL** decides which channels, devices and zones each of them may see or write.

Settings > Users – accounts, groups and permissions

Roles and permissions

CAPABILITY	TYPICAL ROLE
Configuration read	everyone who opens the project
Configuration write (channels, devices, tags, alarms, logging)	engineer
Tag write (setpoints, commands)	operator
SCADA edit	engineer
Users, settings, licence	administrator
Root (super-admin)	the built-in <code>admin</code> ; always has everything

Permissions are evaluated live on every request — changing a user's rights takes effect immediately, without re-login.

Resource ACL

Settings › *Users* › *Access* assigns channels/devices/zones to users or groups with *read* or *read+write*. It is **deny by default** for non-root users: a new user sees nothing until something is granted. Ownership is recorded on created objects so the creator keeps access. The ACL applies to the UI, REST, OPC UA (address space is filtered), Data Bridge and MCP.

Passwords and sessions

- Password policy and forced change at first login (`admin/admin` default).
- Login rate limiting (429 after repeated failures), audit of successful and failed logins.
- Session policy in *Settings* › *Web & REST API*: idle timeout, single vs multiple sessions per user, secure cookies (HTTPS).
- *Forgot password*: 6-digit code by e-mail or SMS when the user has a contact configured; administrators can reset from the CLI:

```
"C:\Program Files\MergenHub\MergenHub.exe" reset-admin "<new-password>" keep
```

HTTPS

See [Publishing on a domain](#) for certificates and reverse proxies. With HTTPS enabled, HTTP is redirected and cookies are marked secure.

OPC UA security

Endpoints with Sign & Encrypt, anonymous off, trusted-client list, per-user access — see [OPC UA server](#).

API tokens

Settings › *API tokens* issues bearer tokens for integrations; each token inherits the user's permissions and can be revoked. Tokens are shown once.

Audit

Every write, login and configuration change is recorded with user, IP and time (*Settings* › *Audit*), kept 90 days by default. See [Logs](#).

Hardening checklist

- [] Change the `admin` password; create named accounts for people, no shared logins
- [] Give operators tag-write only where needed; engineers no user management
- [] HTTPS on; HTTP redirect; strong session policy
- [] OPC UA: anonymous off, Sign & Encrypt, trusted clients only
- [] Firewall: only required ports, restricted source networks ([Firewall](#))
- [] Review the audit log periodically; keep backups ([Backup](#))

AI assistant (MCP server)

Connecting AI assistants over MCP, OAuth and tokens, the tool set, what it can and cannot do.

MergenHub includes an **MCP server** (Model Context Protocol) so AI assistants — Claude, ChatGPT-compatible agents, internal copilots — can query and configure the plant in natural language with the same permissions as the user who authorised them.

MergenHub
OPC UA Gateway - v3.119

PROJE +

- Bağlantılar
 - Simulation Examples 24
 - Functions 24
- Eklentiler
 - Virtual Tag'ler
 - Tag Kısayolları
 - Alarmlar
 - Veri Logları
 - Data Bridge
 - MCP (AI)**

admin
Root (süper-admin)

TR TR

Eklenitler MCP (AI)

Sadece Konfig Canlı Veri 24/24 aktif tag Deneme 24/500

MCP (Model Context Protocol): AI asistanlarının (Claude, Cursor...) sisteme GÜVENLİ bağlanması için token — yalnız okuma + SCADA ekran yapma + doküman; cihaza/tag'e yazma, config ekle/sil YOK. Token yalnız SENİN yetkin dahilindeki araçları alır.

Dış Erişim — AI sağlayıcıya verilecek adres

http://localhost:8095/mcp Kopyala Keşif testini aç

HTTPS: YOK — web tabanlı AI istemcileri yalnız https:// adres kabul eder. Ayarlar — Web & REST API'den HTTPS'i aç (sertifika + domain).

Erişilebilirlik: bu adres YEREL görünüyor — web AI istemcileri buraya İNTERNETTEN ulaşamaz. Sabit IP ya da domain + port yönlendirme gerekir. (Masaüstü istemciler aynı ağıdaysa yerel adresle çalışır.)

Genel adres (public URL): https://scada.firmanız.com Kaydet

Boş = tarayıcının eriştiği adres kullanılır. Domain yayınında MUTLAKA doldur — OAuth keşif metadata'sı (issuer) bu adresle ilan eder.

DOMAİNE ÇIKARMA ADIMLARI (BİR KEZ)

Sabit IP edin (ISS'den) YA DA dinamik DNS kullan; domain'in A kaydını bu makinenin genel IP'sine yönlendir.
Modem/güvenlik duvarında 443 → bu makine port yönlendirmesi yap (MergenHub HTTPS portu).
Sertifika: ücretsiz Let's Encrypt önerilir — win-acme (wacs.exe) ile domain'e sertifika al (HTTP-01 doğrulaması için 80 portu geçici olarak bu makineye yönlendirilmiş olmalı), PFX olarak dışa aktar ve Ayarlar → REST API → Sertifika'dan yükle; sonra HTTPS'i aç. (Alternatif: self-signed üret — tarayıcı uyar verir, claude.ai kabul etmeyebilir.) Let's Encrypt 90 günlüktür; win-acme görev zamanlayıcıyla yenileme kurar — yenilenen PFX'i tekrar yüklemeyi unutma.
Buradaki Genel adres alanına https://domain yaz ve kaydet.
"Keşif testini aç" ile metadata'nın Dış ağdan da açıldığını doğrula (telefonda mobil veriyle dene).

Yeni MCP Token

Ad Örn: Claude Desktop

Süre Süresiz

İZİN VERİLEN ARAÇLAR — GRUBUN TAMAMINI VER YA DA "ARAÇLAR"DAN TEK TEK SEÇ

☐ Tam Erişim — canlı tag değeri yazma HARIÇ her şey
Tek tıkla: okuma + config/tag TANIMI + alarm + veri kaydı + üretim etiket + SCADA tasarım/önizleme. Canlı tag DEĞERİ yazma HARIÇ (böyle bir araç yok). Her araç yine SENİN yetkinle sınırlı.

☒ Etiketler (oku) 3 araç araçlar

☒ Bağlantı (kanal / cihaz) 4 araç araçlar

☒ Alarmlar 3 araç araçlar

Extensions > MCP (AI) — server status, tokens and connected clients

What the assistant can do

- Read:** list channels/devices/tags, search tags, read live values and quality, device status and diagnostics, active and historical alarms, historian queries and charts, metrics and events, project overview.
- Configure:** create/edit/delete channels, devices, tags, tag groups, virtual tags (with expression validation), aliases, alarms, log groups, Data Bridge agents; import/export projects.
- SCADA:** create screens, add widgets from the real symbol library, draw pipes, set animations, render previews.
- Documentation:** read the built-in manual (`list_docs` / `read_doc`).

What it can **never** do: write a live value to a device (no setpoint/command tool exists) — that stays with humans in the UI.

To talk to the assistant from INSIDE the application, see the [in-app AI assistant](#) (ChatGPT, Claude, OpenRouter, Ollama...) — same tools, same authorization.

Connecting

1. *Extensions* › *MCP (AI)* › enable the server. Endpoint: `https://<host>:<port>/mcp` (JSON-RPC over HTTP).
2. Authentication: **OAuth 2.0 with PKCE and dynamic client registration** (for assistants that support it) or a **bearer token** issued in the same page.
3. Two-tier authorisation: the token's scope × the user's permissions/ACL. A read-only token cannot configure even if the user could.
4. Every tool call is written to the audit log with the user and token name.

Claude Desktop / Claude Code example (`mcp.json`):

```
{ "mcpServers": { "mergenhub": { "url": "https://scada.example.com/mcp", "headers": {  
  "Authorization": "Bearer <token>" } } }
```

Typical prompts

- "Which devices lost communication in the last hour?"
- "Create a Modbus TCP device at 10.0.0.9 with holding registers 40001–40020 as floats named Flow1...Flow10."
- "Add a high alarm on Plant.PLC1.Pressure above 6 bar with a 0.2 deadband and e-mail the operators group."
- "Plot Tank1.Level for the last 24 hours and tell me the minimum."
- "Build a screen with two tanks and a pump connected by pipes, animate flow when the pump runs."

Security notes

- Enable the MCP server only when needed; issue tokens per assistant and revoke them from the same page.
- Put the server behind HTTPS; the endpoint is rate-limited and audited.
- The assistant sees only what the token's user may see (resource ACL).

Automation (workflows)

Drag-and-drop flow editor: schedule/tag/alarm/webhook triggers, conditions and calculations, write-tag, email, SMS and HTTP actions; live run monitoring; building with the assistant.

Automation (workflows)

The Plugins › **Automation** module lets you build "when this happens, do that" rules as a drag-and-drop flow chart: **trigger** → **condition / calculation** → **action**. Flows run live on the server (evaluated every second), every run is recorded step by step and shown live in the editor. The AI assistant can build, validate and run flows for you as well.

Triggers

TRIGGER	FIRES WHEN	KEY SETTINGS
Manual / API	Run button, API or the assistant	— (ideal for testing)
Schedule	Fixed interval or cron (min hour day month weekday)	0 8 * * 1-5 weekdays 08:00 · every 300 s
Tag value	Value changes or crosses a threshold	condition (change / above / below / equals / quality), threshold, deadband, dwel time , fire mode (once / repeat), cooldown
Alarm	Alarm raised / cleared / acknowledged	source filter (Well*), minimum priority
Webhook	An external system calls POST /hooks/{key}	key (generated), secret header X-Hook-Secret ; the body becomes \$payload

The tag trigger never fires on the first read; the deadband removes noise; *dwel time* filters transient excursions; *cooldown* enforces a minimum gap between fires. Use all three together for closed-loop control (pump on/off).

Logic

- **Condition** — continues on the *yes* port when the expression is true, otherwise on *no*.
- **AND** — continues once all connected inputs arrived in the same run; **OR** — the first input passes.
- **Calculate** — evaluates an expression and stores it in \$var.name for later conditions and actions.
- **Set variable, Delay** (seconds).

EXPRESSION SYNTAX

{Channel.Device.Tag} live value · \$value / \$prev new/previous value of the triggering tag · \$var.name · \$payload.field (webhook) · \$hour \$minute \$weekday \$now · AND OR NOT · < <= > >= == != · abs min max round sqrt pow if(cond,a,b) . In texts (email subject, message, URL) the same references go inside {...} : Level {Well.P1.Level} m at {\$now} .

Actions

Write tag (setpoint/command; the value may be an expression) · **Email** · **SMS** · **HTTP request** (JSON, response in `$var.http`) · **Event log** · **Acknowledge alarm** · **Run another flow** · **Note** (canvas annotation).

Permissions and safety

- Reading flows requires *Configuration read*; saving requires *Configuration write*.
- A flow that **writes to a tag** can only be saved by a user with *Tag write + service control*; manual runs need the same permission.
- By default a second trigger while the flow is running is **skipped** (concurrent runs off); timeout is 60 s (flow settings).
- Every create/edit/delete/enable is written to the audit log; flows built by the assistant appear with an "AI tool" entry.
- MCP/assistant tools never write live tag values directly; they only define flows — the engine performs the write at run time.

Editor

Left palette (search; drag or double-click) · canvas (drag empty space to pan, wheel to zoom, drag from a port dot to connect, Delete removes, Ctrl+D duplicates, Ctrl+S saves) · right property panel (Tag Browser for tag fields, insert `{tag}` into expressions, copy the webhook URL, live value) · bottom run panel (runs, steps, error text). During a run nodes light up live (blue running, green done, red failed, amber waiting on AND). The list page offers folders, an enable switch, trigger badges, last-run status and JSON import/export.

Building with the assistant

A sentence such as "When well P1 level drops below 8 m for 30 s, email the operator; at night write to the log instead" is enough. The assistant verifies tag paths, creates the flow **disabled**, shows the validation result, tests it with a manual trigger, opens the editor and asks you before enabling it.

API

GET/POST `/config/v1/workflows` , GET/PUT/DELETE `/config/v1/workflows/{id}` , POST `.../validate` , PATCH `.../{id}/enabled` , POST `/runtime/v1/workflows/{id}/run` , GET `/runtime/v1/workflows/runs` , GET `/runtime/v1/workflows/events` (SSE), POST `/hooks/{key}` (anonymous webhook).

Formula editor and live monitoring

Condition/calculation fields offer click-to-insert symbol chips ($\leq \geq \neq$, AND/OR/NOT, $\times \div$, tag picker, `$value` / `$prev` , flow variables, `$payload` , `$hour` / `$minute` / `$weekday` / `$now` , functions); as you type you see syntax status, whether the tag exists, its live value and a trial result. When a run cannot complete, an error is written to the event log, a warning appears on every page and a badge accumulates on the Automation menu; the run panel shows the step it got stuck on and the error, and the node carries a red note. Saving a flow with an email/SMS action while that service is disabled produces a warning. The Enabled switch in the editor applies immediately without saving.

In-app AI assistant

Choosing a provider (cloud and local LLMs), permissions and consent, network diagnostics, SCADA screen design, attachments, skills, context management.

MergenHub ships an in-app AI assistant that works **with the permissions of the signed-in user**. Open it with the  button in the top bar or Plugins › **AI Assistant**: it docks on the right side of the application and can be popped out into a separate window (a native second window in the desktop app). External assistants connect through the [MCP server](#) — both use the same tool catalogue and the same authorization checks.

Choosing a provider

An administrator (Settings permission) defines one or more model providers under  › **Providers** in the assistant panel:

- **Cloud:** OpenAI (ChatGPT), Anthropic (Claude), OpenRouter, Google Gemini, Mistral, DeepSeek, Groq, xAI, Together, Fireworks, Perplexity, Cerebras, Cohere, Azure OpenAI.
- **Local (no key):** Ollama, LM Studio, vLLM, llama.cpp server (Unsloth GGUF), LocalAI, Jan, text-generation-webui, KoboldCpp, LiteLLM/OpenWebUI proxy. **"Scan local servers"** finds running ones and adds a profile with one click.
- **Custom:** any OpenAI-compatible or Anthropic-compatible endpoint.

Each profile has a base URL, an API key (stored only on the server, returned masked), a default model ("Fetch models" lists them), a context window (manual when the API does not report it) and enabled/default flags. The model chip in the chat header switches profile and model from a **searchable list**; the choice is saved per conversation.

If no provider exists yet, the panel shows the administrator a **3-step wizard** (provider → key → model, test). When the assistant is disabled the menu entry is dimmed and clicking it explains where to enable it.

Behaviour settings

- **Assistant enabled** — when off, nobody can chat.
- **Assistant may make changes** — off: read-only tools; on: configuration/alarm/logging/SCADA write tools **within the user's own permissions**, every write audited as `user (AI assistant)`. There is **never** a tool that writes live values to a device.
- **Reasoning / thinking** (off/low/medium/high) — supported by OpenAI o-series and GPT-5, Claude (extended thinking), Gemini 2.5, OpenRouter, Groq, DeepSeek-R1; the thinking stream appears as a collapsible block.
- Temperature, max output tokens, tool steps per turn, **administrator instructions** (site-specific rules).

What the assistant does

It works like a SCADA/automation engineer: read first, write precisely, then verify.

- **Inventory and diagnostics:** channels/devices/tags, live values and quality, device status, alarms, historian, events, audit log, users and permissions, settings — everything your permissions allow.

- **Network tests:** `ping` , TCP port checks (Modbus 502, S7 102, OPC UA 4840...), DNS resolution, local listening ports. "The device is not communicating" is answered by measurement, not guesswork.
- **Configuration:** create/update/delete channels, devices, tags, virtual tags (expression validated), aliases, alarms (ISA-18.2), log groups, Data Bridge agents.
- **SCADA screens:** real symbol library, pipelines with flow animation, tank/pump scenes, bindings, previews. The built-in "**SCADA screen design**" skill provides the layout grid, ISA-101 colour language, navigation and validation rules.
- **System:** partial settings updates (OPC UA port, HTTPS...), HTTPS certificate (self-signed or a PFX attached to the chat), Windows Firewall ports, service restart. Example: "my domain is X, publish HTTPS on 443" → DNS → is the port free → certificate → settings → firewall → restart → verification.
- **Attachments:** images (PNG/JPG...), PDF, CSV/JSON/TXT, SVG and PFX can be attached; images go to vision-capable models and can be placed on screens with `add_widget type=image` , CSV/JSON content becomes context.

`Channel.Device.Tag` references in answers are clickable and navigate the main application. Tool calls appear as cards with arguments, results and previews; a status line ("Thinking / Creating screen / Testing ports") shows where the work is.

Permissions and consent

Security layers: (1) the **"Use the AI assistant" permission** — granted per user/group, without it the assistant is off; (2) every tool call is limited to the user's **own permissions**; (3) the administrator's **write gate**; (4) **approval** — by default every change waits for Approve/Reject in the chat, the assistant never acts unasked; (5) **audit log** — every tool call (reads included) and chat message is logged; (6) there is no live-value write tool.

Token economy: the model receives the core tool groups plus a compact catalogue, not all tools every turn; required groups are loaded during the conversation. Templates: say "make this device type a template" and the assistant saves it as a skill; afterwards one sentence creates channel/device/tags/alarms/screen of that type.

The assistant uses no token: it is the **signed-in user** and acts with that user's permissions. If a tool is refused because of a missing permission (e.g. ConfigWrite) or the write gate, the assistant states the reason and asks; an administrator can open the gate from the card with **"Grant and retry"**. No workarounds, no guessing.

Conversations, context and skills

- Conversations are private to the user; rename, folders, pinning, archive and search are available.
- The **context meter** compares the last turn's tokens with the model's window; at 75 % older turns are summarised automatically (the full history stays on screen). Click the meter to compact manually.
- **Skills** (`DataRoot/ai/skills/*.md`) are site-specific instruction packs. Two built-in expert skills ship enabled (SCADA screen design; alarm/historian/bridge/virtual-tag engineering); administrators edit, disable or add their own. Only name and description enter the system prompt; the body is loaded on demand.
- **External MCP servers:** tools from an ERP, a document store or another MergenHub can be added (Streamable HTTP + optional bearer token).

Data and privacy

Conversations, attachments and skills live in the MergenHub data directory (`ai/`); API keys are stored only in `settings.json` and returned masked. Only the text/images needed for the current turn are sent to the provider;

with a local provider (Ollama, LM Studio...) nothing leaves the machine.

Data processing consent (GDPR / KVKK)

Before a cloud provider (OpenAI, Anthropic, OpenRouter, Google...) can be used, an administrator must give **data processing consent**; without it no chat opens on a cloud profile. The text covers: you are the data controller and the provider is a third party; the data transferred (messages, attachments, tool outputs: project configuration, tag values, alarm/event records, user names — keys and passwords are masked); the legal basis for transfers abroad; retention; informing your users; security; and the record of who accepted and when, with revocation. Local providers (Ollama, LM Studio, vLLM...) keep data on the machine; no consent needed.

Prices, cost and model info

As soon as an API key is entered the model list is fetched automatically; each model shows its context window and price (per 1M tokens in/out). Prices come from the public OpenRouter catalogue (24 h cache), local models count as free; for unknown models only token counts are shown. The estimated cost of each turn and of the conversation appears under the chat and in the conversation list. The model can be changed mid-conversation; the "Thinking" chip selects the reasoning level per conversation.

Approval modes

Three modes for tools that change things: **Ask on every change** (default), **Remember a tool approved once in the conversation**, **Apply without asking** (administrators only; for closed environments). The approval card offers "do not ask again in this conversation" and "always allow"; permanent allowances are listed and removable under Settings › Behaviour. Every tool call is audited in all modes.

Scripts, seeing the screen, and permissions

The assistant can run Python or PowerShell scripts in a **sandbox** for calculations, reports and conversions. The feature is **off by default** and only the **system administrator (Root)** can enable it: Settings › AI › Behaviour › *Script execution*. A risk statement must be read and accepted first; the acceptance (who and when) is written to the immutable compliance ledger.

How scripts are contained:

- They run as a separate **low-privilege Windows account** (MergenHubSandbox) and cannot read the MergenHub data folder.
- **Network access is off by default**. If you enable it you provide an allowlist (IP, CIDR or hostname); no packet can leave the machine for any other address. The restriction is enforced by the operating system firewall.
- Scripts **cannot write live tags**. Field writes only happen through separate approval cards.
- Timeout and output caps apply; the working folder is deleted after each run.
- **The code is shown to you and confirmed on every run**. The "apply without asking" mode and the always-allow list do not apply to this tool. Every run (code digest, exit code, duration) is written to the audit log and the compliance ledger.
- An **emergency stop** button kills all running scripts immediately and locks further runs; only the system administrator unlocks it.

The assistant can also **see your screen** when needed: for questions like "this looks broken" or "read this trend" it asks for permission; if you approve, a screenshot of the current screen is added to that conversation only.

New capabilities (Settings › Users › Permissions):

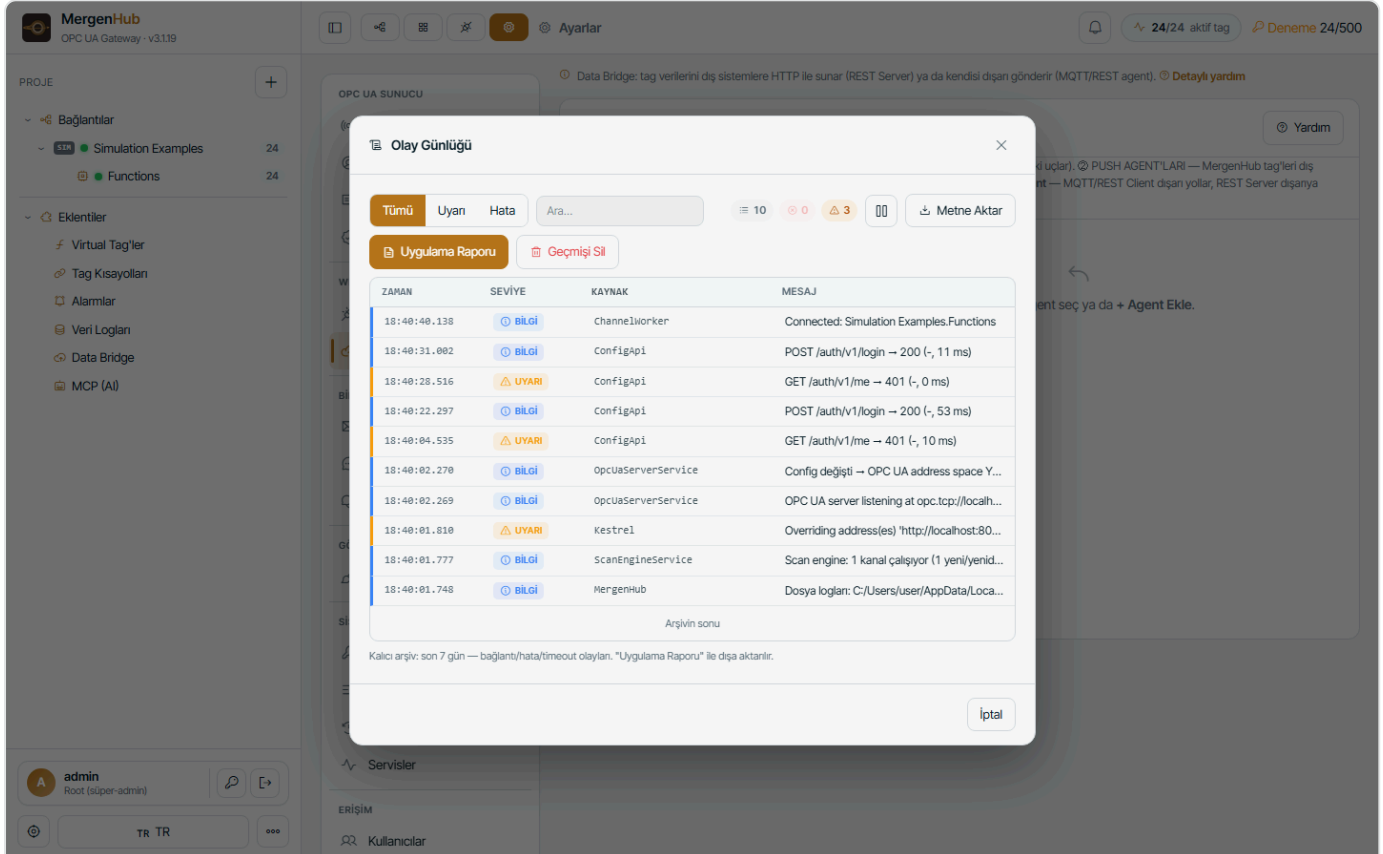
CAPABILITY	MEANING
Use the AI assistant	Chat with the assistant
MCP bridge	Connect with an external client (Claude Desktop, IDE) — separate from the assistant
Assistant may run scripts	Dangerous; never granted by a role default, only Root grants it individually
Assistant/MCP security administration	Script feature, approval mode, permanent allowances, external MCP servers — Root only

The assistant can also read your **license and package status**: how many tags you may still add, perpetual or subscription, when maintenance (the right to new versions) ends, whether a newer version exists and where to renew.

Troubleshooting

Device diagnostics and packet traces, common errors and their fixes, where to look first.

Start with the three places that answer most questions: the **Event log** (... > Event Log), the device's **Diagnostics** panel, and the day's **file log** under `C:\ProgramData\MergenHub\logs`. See [Logs and diagnostics data](#).



Event log with errors filtered – the fastest way to see what went wrong

Device shows Bad quality

CHECK	WHERE
Is the device reachable?	<code>ping</code> , <code>Test-NetConnection <ip> -Port 502</code> from the server
Right unit ID / rack-slot / addresses?	Device settings; the ?address hints
What does the device answer?	Device > Diagnostics : request/response hex, exception codes (Modbus 02 = illegal address, 03 = illegal value)
Too big a block?	Lower <i>max block size</i> on the Modbus device; some RTUs reset above 64 registers
Timeouts on GSM/radio?	Raise request timeout, lower retries, enable demand-driven scanning
Trial window expired?	Top bar shows <i>TRIAL · expired</i> — start a new window or enter a license (Licensing)

OPC UA client cannot connect

- Endpoint URL must match an endpoint in *Settings* › *OPC UA Server* (host name and port).
- Certificate errors: trust the client in *Settings* › *Trusted clients* (or temporarily enable auto-accept); trust the server certificate in the client.
- Anonymous disabled → provide an OPC UA user.
- Port in use → the event log shows *OPC UA server failed to start*; change the port.

Web UI not reachable from other computers

Host is `localhost` (change to `0.0.0.0`), firewall rule missing ([Firewall](#)), or a reverse proxy misconfigured ([Publishing](#)).

Service does not start

`sc query MergenHub` → *STOPPED*; check `logs\mergenhub-YYYYMMDD.log` for the exception at startup (usually a port conflict or a corrupt `settings.json`); Windows Event Viewer › Application › *MergenHub*. Reinstall over the existing installation to repair program files; data is kept.

After an upgrade OPC UA clients say the certificate is invalid

The server certificate was regenerated. Re-trust it in the client, or restore `pki\own` from backup and restart.

Alarms not sending e-mail / SMS

Settings › *E-mail* › *Send test*; check SMTP port/TLS and the user's notification preferences; SMS provider credentials in *Settings* › *SMS*. Failures are logged in the event log with the provider's response.

Historian not logging

The log group must be enabled and the historian service running (*Settings* › *Services*); check the target database connection; with ODBC verify the driver is 64-bit. Quality *Bad* values are stored only if the group allows it.

Performance

- High CPU: too many fast scan rates on serial links or expensive virtual tags; use on-change evaluation.
- High RAM: expected with hundreds of thousands of tags and many OPC UA subscribers; keep 8 GB+ for 100K tags.
- The dashboard's capacity banner and *Settings* › *Services* show CPU/RAM/disk.

Collecting information for support

1. ... › Event Log › **Application Report** (one file with version, settings summary, health and recent events)
2. `logs\mergenhub-<today>.log`
3. Device › Diagnostics trace for the affected device
4. Send via the customer portal (*Support*) — licensed customers with maintenance get priority.

MergenHub Mobile (iOS and Android)

Server mode and phone mode, preparing the server, live values, writes, alarm acknowledge, SCADA screens, push notifications, screen editor, the Add device wizard, security, troubleshooting.

MergenHub Mobile is the iOS and Android app for MergenHub. It works in two modes: in **server mode** it connects to the MergenHub server at the plant to watch live values, write values, acknowledge alarms and open SCADA screens; in **phone mode** it needs no server — the drivers run on the phone itself and talk to PLCs and RTUs directly. The app needs no extra licence: in server mode the server's licence applies, phone mode is free.



Installing

The app is published on the App Store and Google Play as **MergenHub**; store links are on the [MergenHub Mobile](#) page.

Note The app's user interface is currently in Turkish. This guide gives the Turkish label in bold with an English translation in brackets, for example **Sunucu ekle** (Add server). Labels of the MergenHub web interface are given in English.

Permissions the app may ask for:

- **Local network** (iOS) — to reach PLCs on the same network in phone mode.
- **Notifications** — to receive alarm notifications in server mode.
- **Camera** — only when scanning a QR code.
- **Face ID / fingerprint** — only if you turn on the app lock or authentication before writes.

The app sends no personal data to Envest and contains no ads or tracking.

The two modes

	SERVER MODE	PHONE MODE
Where data comes from	MergenHub server (REST API)	The project on the phone and drivers running on the phone
When it runs	24/7 on the server; the phone is only a window	Only while the app is open
Logging, alarms, notifications	Continuous on the server; push notifications to the phone	While the app is open; no notifications
Screens	SCADA screens designed in the web interface	Screens designed on the phone
Permissions	Same as your user on the server	The phone's owner can do everything
Licence	The server's licence	Not required

You choose the mode with the two large buttons on the home screen (**Sunucu modu**, **Telefon modu**); the choice is remembered. Server mode is shown in blue, phone mode in brass. Inside a workspace the header shows the mode (**SUNUCU MODU / TELEFON MODU**), the user and the tab; tapping the header returns to the home screen.

Server mode

PREPARING THE SERVER

On the server (in the web interface, as an administrator):

1. **Network access** — under *Settings > Web, REST API & Mobil* the Host must not be `localhost`. For the local network enter `0.0.0.0` and restart the MergenHub service. The firewall rule added by the installer opens port 8090. See [Publishing on a domain](#) and [Windows Firewall and ports](#).
2. **Mobile access** — on the same tab, in the *Mobile app (iOS / Android)* card, **Allow sign-in from the mobile app** must be on. When it is off, new sign-ins and open mobile sessions are rejected. **Session lifetime (days)** sets how long a phone stays signed in without asking for the password again (default 90).
3. **Per-user access** — in the *Mobile access per user* table the user's **Mobile sign-in** box must be ticked. A user with mobile sign-in disabled cannot connect from the app; web access is unaffected.
4. **Access over the internet** — publish the server on a domain with **HTTPS**. Use HTTP only on a trusted local network; over HTTP the password and data travel unencrypted.

Note Device sessions, push notifications and writing String tags need MergenHub 3.5 or later. The app connects to older servers with HTTP Basic: the password is then kept in the phone's secure store and there is no session list and no push. For server screens to look exactly as in the web interface, the server needs 3.5.1 or later.

ADDING A SERVER AND SIGNING IN

1. On the home screen choose **Sunucu modu** (Server mode) and tap **Sunucu ekle** (Add server).
2. In **Adres** (Address) enter an IP address or a domain. With an IP address the server's port 8090 is tried; with a domain HTTPS is tried first. You can also type the scheme (`http://` or `https://`) and port yourself.
3. Optionally give a **Görünen ad** (Display name), for example "North pump station".
4. Enter **Kullanıcı adı** (User name) and **Parola** (Password) — your account in the web interface.
5. **Bağlantıyı dene** (Test connection) shows the server's name, tag count, status and whether the connection is HTTPS or HTTP. **Kaydet** (Save) signs in.

If the server can only be reached over HTTP, the app shows an **unencrypted connection (HTTP)** warning and asks you to confirm. If the account must change its password at first sign-in, a new-password screen appears when you open the server.

You sign in once: the server issues a **session token** for this phone and the password is not stored on the phone. The token is kept in the phone's encrypted store (Keychain / Keystore).

The server list on the home screen checks each server every ten seconds (**Çevrimiçi online** / **Erişilemiyor** unreachable). Long-press a server to change its address, display name or user; **Sunucuyu kaldır** (Remove server) also closes the device session on the server. You can add several servers and switch between them from the home screen.

PERMISSIONS

What you can do in the app depends on your permissions and the resource access lists (ACL) on the server; channels and devices you cannot access are not shown.

ACTION	PERMISSION
Tree, live values, device status	any role (Viewer)
Event log	ConfigRead
Add / edit / delete channels, devices, groups, tags; Add device wizard	ConfigWrite
Write values, buttons and switches	RuntimeControl
See and acknowledge alarms, alarm notifications	Alarms
Trend history (Data Logger)	DataLogging
SCADA screens	Scada
Enter a licence key	Settings
AI assistant	Ai

OVERVIEW (GENEL)



The first page after connecting:

- **Tiles** — tag count, connected / total devices, devices with errors, active and unacknowledged alarms.
- **Aktif alarmlar** (Active alarms) — the three newest; **Tümü** (All) opens the Alarms tab.
- **Cihazlar** (Devices) — the state of each device: **Bağlı** (connected), **Hatalı** (error), **Bağlı değil** (not connected) or **Kapalı** (disabled); the last error is shown under the row. **Ayrıntı** (Details) opens *Device status* with read/write counters.
- **İzleme listesi** (Watch list) — tags you look at often. Add a tag with the star in its detail page; the list is kept per server.

The green **Canlı** (Live) dot in the header means values are flowing; if the connection drops it shows **Bağlantı yok** (No connection). Values refresh once a second by default (*Settings* › *Server mode*).

CHANNELS, DEVICES AND TAGS (KANALLAR)



The **Kanallar** tab is the project tree of the web interface: channels, their devices, each device's tag count and a status dot. Tap a device to open its page.

With the ConfigWrite permission:

- Tap a channel name: **Cihaz ekle** (Add device), **Kanal ayarları** (Channel settings), **Kanalı kopyala** (Copy channel), **Kanalı sil** (Delete channel).
- Tap a device's ... button or long-press it: **Tag'leri aç** (Open tags), **Tag ekle** (Add tag), **Cihaz ayarları** (Device settings), **Cihazı kopyala** (Copy device), **Cihazı sil** (Delete device).
- The **Cihaz ekle** button at the top opens the [Add device wizard](#).

Saving channel or device settings restarts that channel; communication may pause for a few seconds. Fields and setting keys are the same as in the web interface; see [Channels, devices and tags](#).

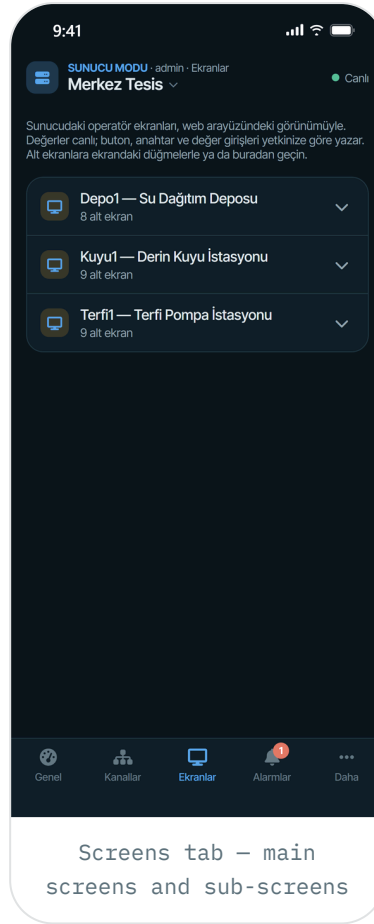


The **device page** lists the device's tags and groups with live values, with read, error and write counters and a search box at the top. With the right permission you add tags and groups and open the device settings here.

The **tag detail** page opens when you tap a tag:

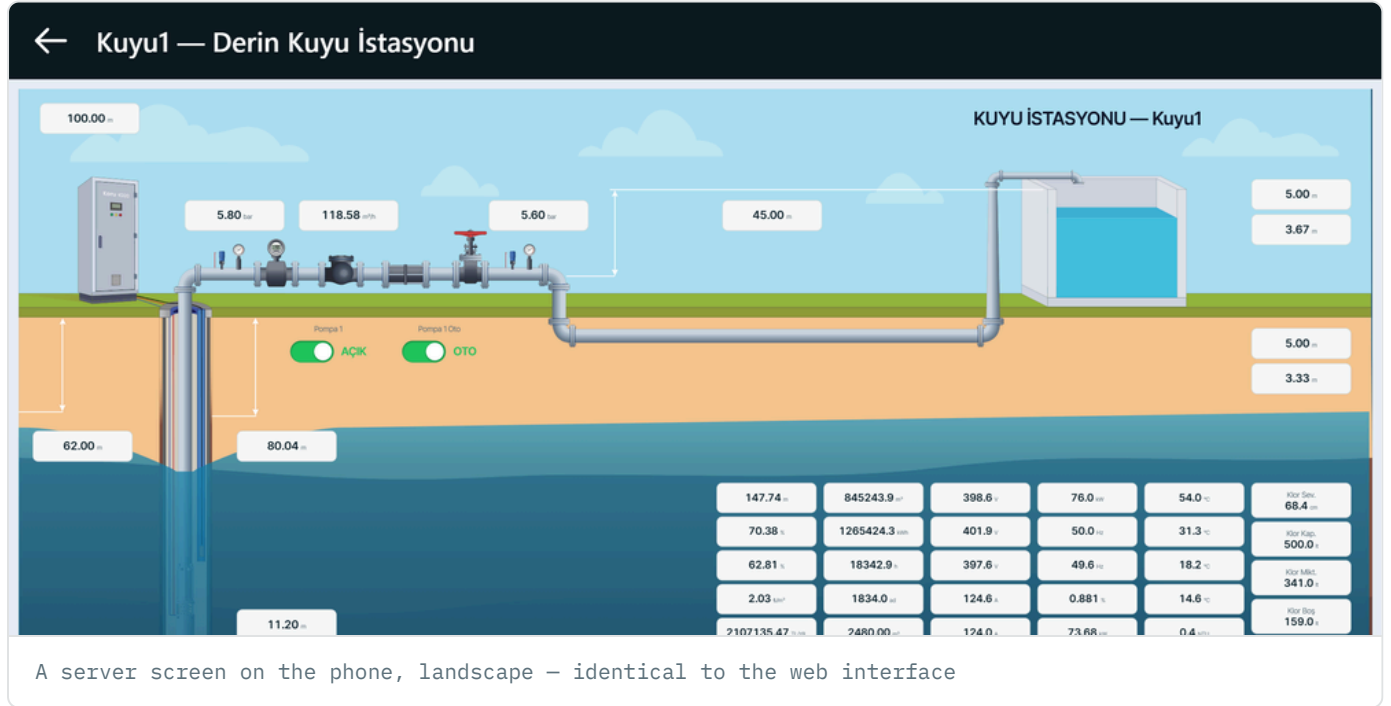
- Value, quality (**Good, Uncertain, Bad**) and timestamp.
- **Değer yaz** (Write value) — **AÇIK / KAPALI** (ON / OFF) buttons for Boolean tags, a value field and **Yaz** (Write) for the rest. Only shown for writable tags and with the RuntimeControl permission.
- **Trend** — 5 min, 15 min, 1 h, 6 h or 24 h. History comes from the Data Logger (DataLogging permission); samples collected while the page is open are added.
- Address, data type, access, scan rate, scaling and description.
- The star at the top right adds the tag to the watch list; the menu has **Düzenle** (Edit), **Kopyala** (Copy), **Sil** (Delete).

SCREENS (EKRANLAR)



Screens tab – main screens and sub-screens

The **Ekranslar** tab lists the server's SCADA screens as in the web interface: main screens at the top, sub-screens (settings screens and so on) in a branch opened with the arrow. The tab needs the Scada permission.



A server screen on the phone, landscape – identical to the web interface

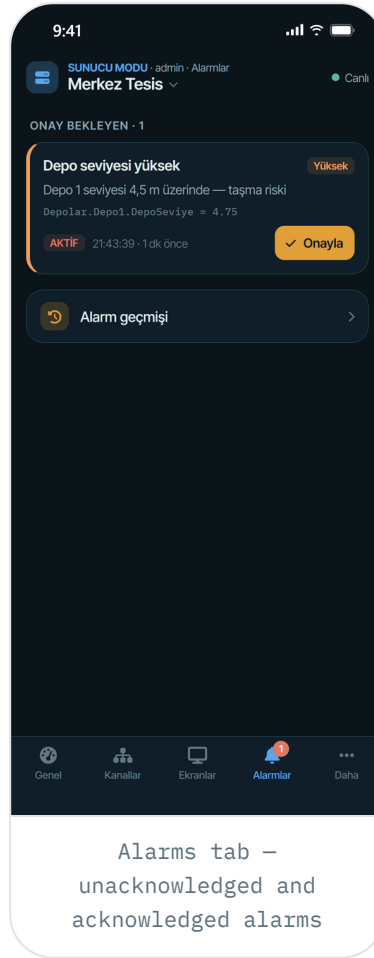
Tapping a screen opens it in the server's own viewer, in operator (run) view: symbols, flow animations, 3D backdrops and sub-screen buttons look exactly as in the web interface. Values are live; buttons, switches and

value inputs write according to your permissions.

Wide station screens are fitted to the width of a phone held upright. The phone icon at the top right turns the screen to **landscape**; pinch to zoom and drag with one finger to pan. On tablets and in landscape the view stays the operator view; screens are designed only in the web interface.

Screens are designed in the web interface ([SCADA / HMI](#)); they are not edited on the phone in server mode. On MergenHub 3.5.0 and older servers screens are drawn in a simplified way: macros, SVG symbol library objects and maps appear as faded boxes.

ALARMS (ALARMLAR)



The red number on the tab icon is the number of unacknowledged alarms. Alarms are listed in two groups: **Onay bekleyen** (Awaiting acknowledgement) and **Onaylanmış, aktif** (Acknowledged, active). Each card shows:

- The alarm name and severity: **Kritik** (critical, 800 and above), **Yüksek** (high, 600–799), **Orta** (medium, 400–599), **Düşük** (low).
- The message, the tag path and its current value — tap the tag line to open the tag detail.
- **AKTİF** (active) or **NORMALE DÖNDÜ** (returned to normal), the time it was raised and who acknowledged it.
- The **Onayla** (Acknowledge) button. The acknowledgement is written to the server's audit log with your user and the source **Mobile**.

Alarm geçmişi (Alarm history) lists alarm events (raised, returned to normal, acknowledged) with time, tag value and user. Alarm definitions are managed in the web interface; see [Alarms and notifications](#).

MORE (DAHA)

In server mode the **Daha** tab contains:

- **Tag ara** (Search tags) — by name, path or address across the server.
- **Cihaz durumu** (Device status) — connection, read/write and error counters per device.
- **Olay günlüğü** (Event log) — the server's event log (ConfigRead permission).
- **AI asistanı** (AI assistant) — only shown with the Ai permission; see below.
- **Cihaz ekle / kütüphane** (Add device / library) — with the ConfigWrite permission; see [Add device wizard](#).
- **Sunucu kaynakları** (Server resources) — CPU, memory, disk, MergenHub's own memory, tag and scanned-tag counts, OPC clients, monitored items, threads and handles, and the state of the services; refreshed every five seconds.
- **Cihazlarım ve bildirimler** (My devices and notifications) — your open mobile sessions and this phone's notification registration.
- **Sunucu lisansı** (Server licence) — customer, key, tag limit, validity, maintenance date and machine fingerprint. A user with the Settings permission can enter and activate a licence key here.
- **Sunucu bağlantısını düzenle** (Edit server connection) and **Bu sunucunun projesini telefona al** (Copy this server's project to the phone).
- **Ayarlar** (Settings) and **Çalışma alanını değiştir** (Switch workspace — back to the home screen).



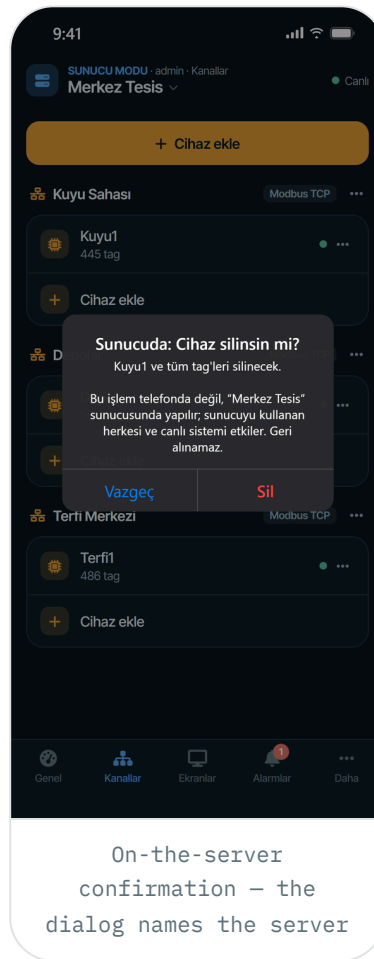
Sessions. *Cihazlarım ve bildirimler* lists your user's open sessions (device name, platform, app version, last seen, IP); the phone you are using is marked **BU CİHAZ** (this device). End the session of a lost phone here with **Kapat**

(Close). **Bu cihazda oturumu kapat** (Sign out on this device) removes the server from the list; you need the password to connect again.

AI assistant. If the in-app AI assistant is enabled on the server and has an active provider, you can chat with it from the phone and choose among the enabled providers and models. When the assistant asks for confirmation or permission to run an action, approve and **Reddet** (Reject) buttons appear under the message. Assistant settings live in the web interface; see [In-app AI assistant](#).

“SUNUCUDA” (ON THE SERVER) CONFIRMATIONS

A change you make in server mode does not happen on the phone — it happens on the server and affects everyone who uses it. The confirmation dialog says so explicitly: its title starts with **Sunucuda:** (On the server:) and the text names the server where the action will run and states that it affects everyone using the server and the live system.



On-the-server
confirmation – the
dialog names the server

The dialog appears for:

- **Deleting** a channel, device, group or tag — "Geri alınamaz." (Cannot be undone.) is added.
- **Creating and saving changes to** a channel, device or tag; creating and renaming a group. For channels and devices it also says that the channel will restart.
- **Copying** a channel, device or tag — the copy starts scanning on the server right away.
- In phone mode, *Proje > Sunucuya gönder* (send to server) — it says on which server how many devices and tags will be created.

- **Installing** in the Add device wizard — the devices to install and what will be created on the server (channel, device, tags, calculated tags, screens) are listed.
- **Writing a value** from the tag detail page, when *Yazmadan önce onay iste* (Ask before writing) is on.

In phone mode the same actions ask for a plain confirmation, without "Sunucuda".

PUSH NOTIFICATIONS

When an alarm is raised the server sends a notification to the phone through the Expo Push service; it arrives even when the phone is locked. All of the following are needed:

1. The server runs **MergenHub 3.5** or later and the phone is connected with a device session.
2. **Allow sign-in from the mobile app** is on (*Settings › Web, REST API & Mobil*).
3. **Send mobile push notifications** is on (*Settings › Mobile push*).
4. **Receive push** is ticked for the user (*Settings › Mobile push › Push per user*).
5. The user has the **Alarms** permission and access to the alarm's tag.
6. **Mobile push** is selected among the notification channels of the alarm definition (on by default).
7. The server has **outbound HTTPS** to the internet (`exp.host` , port 443). The phone does not need to reach the server; the notification arrives via Expo.
8. Notifications for MergenHub are allowed on the phone.

The app shows the phone's registration under *Daha › Cihazlarım ve bildirimler* ("Bu cihaz alarm bildirimlerine kayıtlı" — this device is registered — or the reason it could not register); **Yeniden kaydet** (Register again) repeats it. On *Settings › Mobile push* administrators see which devices receive push, and can send a test notification or an announcement.

There are no notifications in phone mode.

Phone mode

In phone mode the project is kept on the phone and the drivers run on the phone; no server or licence is needed. It is designed for commissioning, address checking and testing in the field.



1. On the home screen choose **Telefon modu** (Phone mode) and tap **Projeye başla** (Start project) — or **Projeyi aç** (Open project) if a project exists.
2. Use **Cihaz ekle** (Add device) to pick a ready-made device from the library or create a channel by hand.
3. The **Durdu** / **Çalışıyor** (Stopped / Running) button at the right of the header starts and stops scanning. Once scanning starts the drivers connect and values flow.

Warning Phone mode runs only while the app is open and in the foreground. When the phone sleeps or the app goes to the background, scanning stops, alarms are not monitored and the alarm history is lost when the app closes. For continuous monitoring, logging and notifications run the project on a MergenHub server. *Settings > Tarama sürerken ekranı açık tut* (Keep the screen on while scanning, on by default) stops the screen from turning off.

DRIVERS

All of MergenHub's field drivers and the simulator run on the phone. Driver names, device ID formats, address syntax and setting keys are the same as on the server, so a project moves between server and phone unchanged. *Daha > Desteklenen protokoller* (Supported protocols) shows each driver's address formats with examples.

DRIVER	NOTES
Simulator	Sine, ramp, square wave, random, counter, writable register
Modbus TCP/IP Ethernet	Holding/input registers, coils, discrete inputs, register bits, strings
Modbus RTU Serial / Modbus ASCII Serial	Through a serial server or gateway (Ethernet encapsulation)

DRIVER	NOTES
Siemens S7 TCP/IP	S7-200/300/400/1200/1500, DB, M, I, Q areas
OMRON FINS Ethernet	FINS/TCP on the phone; a device set to UDP connects over TCP
OMRON NJ Ethernet	NJ/NX, EtherNet/IP CIP symbolic variables
MQTT Client	MQTT 3.1.1 over TCP or WebSocket; JSON, CSV, XML, binary, regex extraction
DNP Client Ethernet	Integrity and Class 1/2/3 polls, CROB and analog output
IEC 60870-5-104 Client	General interrogation, spontaneous events, commands
IEC 61850 MMS Client	Read, write and control (Oper)
OPC UA Client	Unsecured (None) endpoints only; anonymous or user name

The phone must be on the same network (Wi-Fi) as the devices. The store version includes raw TCP support: under *Settings* › *İstemci modu* (Client mode) the **Ham TCP** (Raw TCP) badge shows **NATIVE** and nothing else is needed. The **TCP köprüsü** (TCP bridge) field in the same section is only for development builds; leave it empty in the store app.

EDITING CHANNELS, DEVICES AND TAGS

The Kanallar tab and its menus are the same as in server mode. Specific to phone mode:

- A new channel asks for a driver; after it is created the first device is added straight away.
- On a new device, **Örnek tag'ler ekle** (Add example tags) adds the driver's sample tags.
- The tag form: name, address, data type, **Yazılabilir** (Writable), scan rate, **Açıklama / birim** (Description / unit — shown as a unit if 6 characters or fewer, e.g. `bar`, `m3/h`) and **Ölçekleme** (Scaling: raw and scaled low/high, clamping, negation). **Kaydedince yenisini ekle** (Add another after saving) speeds up entering many tags.
- The device menu has **Şablon olarak kaydet / paylaş** (Save / share as template); see [Templates and sharing](#).

Writes to scaled tags apply the scaling in reverse.

ALARMS AND CALCULATED TAGS

Alarm tanımları (Alarm definitions — *Daha* › *Alarm tanımları* or the bottom of the Alarms tab) defines alarms on the phone:

- **Condition** — very high, high, low, very low, deviation +/-, rate of rise, rate of fall.
- **Limit** — plus a **setpoint** for deviation alarms; for rate alarms the limit is units per second.
- **Deadband** — none, absolute or percent (with range low/high), plus **on** and **off delay** (s).
- **Severity** (low, medium, high, critical), **message**, **area**, **enabled**.

Gelişmiş tag'ler (Advanced tags — *Daha* › *Gelişmiş tag'ler*) are tags calculated from other tags; their path is

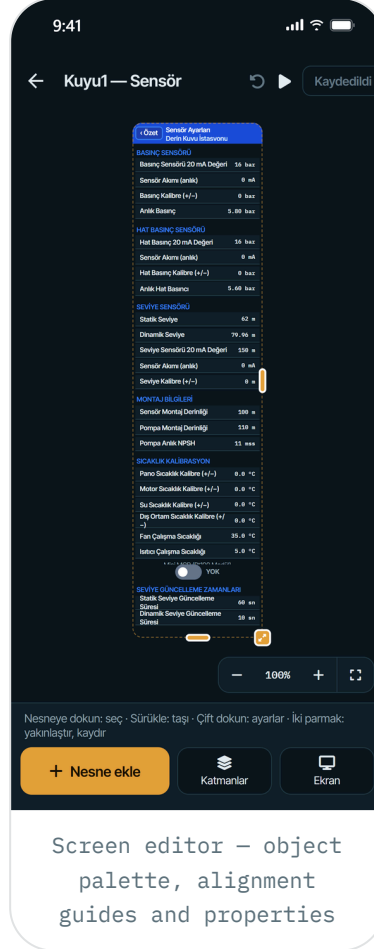
`_AdvancedTags.<name>` :

- **Expression** — `{Channel.Device.Tag}` references, arithmetic, comparisons and functions such as `ROUND`, `MIN`, `MAX`, `IF`, `POLY`. The function list is the same as on the server; see [Advanced \(virtual\) tags](#) and [Creating device templates](#).
- **Totalizer** — sums value × seconds (for an m³/h input divide the result by 3600).

- **Average, Maximum, Minimum** — over the last **sample count** values of the source tag.

SCREENS AND THE SCREEN EDITOR

In phone mode the **Ekranlar** tab lists this project's operator screens. Tap a screen to run it; **Düzenle** (Edit) opens the editor; the menu has **Çalıştır** (Run), **Düzenle**, **Kopyala** (Copy) and **Sil** (Delete). **Ekran ekle** (Add screen) opens a new screen as a draft; it is added to the project only when you tap **Kaydet** (Save).



In the editor:

- The **Nesne ekle** (Add object) palette: value display, lamp, button, switch, value input, setting row, gauge, tank and level bar, pump, valve, motor, fan, pipe, text, panel, image and more than 260 industrial symbols.
- Tap an object and drag it; alignment guides appear. Corner and edge handles resize it freely, the round handle on top rotates it. On empty space one finger pans and two fingers zoom.
- For the selected object: tag binding, label, unit, decimals, colours; **Durum** (State — when the object counts as "on"), **Kumanda** (Control — on/off values), **Seviye (dolum)** (Level fill — minimum and maximum); **Koşullu kurallar** (Conditional rules: change colour, blink, hide; conditions $>$, $\geq$, $=$, $\neq$, $<$, $\leq$ or between); **Dokununca** (On tap) — go to another screen.
- **Katmanlar** (Layers), **Öne / Arkaya** (Bring forward / Send backward), **Çoğalt** (Duplicate), **Sil** (Delete) and undo.
- **Ekran ayarları** (Screen settings): name, size (phone portrait 400×700, phone tall, phone landscape, tablet, wide screen) and background colour.

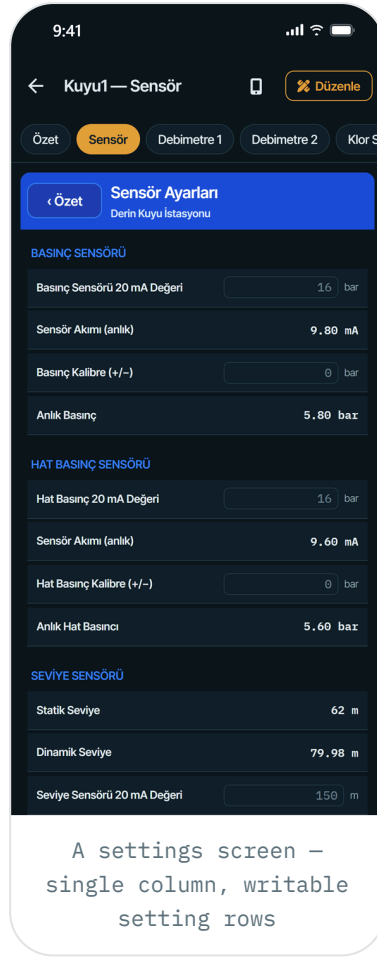
Screens are stored in the same structure as the server's SCADA screens; when a project moves to the server as a package, its screens go too.

SUMMARY AND TABS FOR LIBRARY DEVICES

A device installed from the Device Library in phone mode comes with a screen set made for the phone: a portrait **Özet** (Summary) screen with measurements, controls and buttons for the settings screens, plus single-column settings screens.



When the Summary is open a tab strip appears at the top: **Özet** and the device's settings screens (for example sensor, PID, communication). Tap a tab, or a settings button on the Summary, to switch.



On setting rows tap the value to write a new one; depending on your settings the write asks for confirmation and authentication.

MOVING A PROJECT BETWEEN PHONE AND SERVER

From the phone to the server:

- *Daha > Proje > Sunucuya gönder* (Send to server) — pick a saved server and a channel; the channel is created on the server as a new channel with its devices and tags. Screens, alarms and advanced tags are not sent this way; a channel with the same name must not exist on the server.
- Create a MergenHub package (`.mhpkg.json`) with *Proje > Projeyi dışa aktar* (Export project) and install it in the web interface with *Device Library > Import from file*. This also moves screens, alarms and calculated tags.
- *Proje > Kepware/MergenHub JSON olarak paylaş* (Share as Kepware/MergenHub JSON) — the classic project JSON, loaded in the web interface with *Project > Import* (channels, devices and tags only).

From the server to the phone: in phone mode *Daha > Sunucudan proje al* (Get project from server), or in server mode *Daha > Bu sunucunun projelerini telefona al*. Select the channels; if **Aynı adlı kanalın yerine geç** (Replace a channel with the same name) is off, the channel is added under a new name. Channels, devices and tags are copied.

Add device wizard

The wizard works the same way in both modes. In server mode the device is installed **on the server** (ConfigWrite permission required); in phone mode **into the phone's project**. Open it with the **Cihaz ekle** button on the Kanallar tab, the **Cihaz ekle** card of an empty project or *Daha > Cihaz ekle / kütüphane*.

CHOOSING A SOURCE

- **Cihaz kütüphanesi** (Device Library) — device packages prepared and tested by Envest, by brand, model and firmware version, with tag list, calculated tags and screens. You can select several devices.
- **Telefondaki şablonlarım** (My templates on this phone) — templates you saved or received.
- **Dosya, pano, QR ya da adres** (File, clipboard, QR or address) — a MergenHub package from a file, pasted from the clipboard (for example text from an AI assistant), from a QR code or from an `https://.../device.mhpkg.json` address.
- **Elle oluştur** (Create manually) — choose the driver and define the channel and device yourself; add tags later.

CHOOSING FROM THE LIBRARY



The library list is downloaded from the website. The search box searches brand, model, firmware version and protocol; the category buttons narrow the list. Each card shows the protocol, tag count and the number of screens that will be installed. The box on the right of a card adds the device to the selection; **Seçilenleri kur** (Install selected) installs them all together.

Without internet the last downloaded list is shown ("Çevrimdışı", offline); installing needs internet to download the package. In server mode the phone downloads the package and installs it on the server, so the server itself needs no internet access.



Tap a card for details: photo, category, **Ne yapar?** (What it does), **Nasıl çalışır?** (How it works), **Paketin içeriği** (Package contents), **Kurulumda sorulanlar** (Asked at installation) and the product page. **Bu cihazı kur** (Install this device) installs only this one; **Seçime ekle** (Add to selection) is for installing several.

If your device is not there, the **Ekleme isteği gönderin** (Request it) link under the list opens the customer panel on the website; see [Creating device templates](#).

INSTALLATION DETAILS

Installation details –
channel and device name,
connection parameters

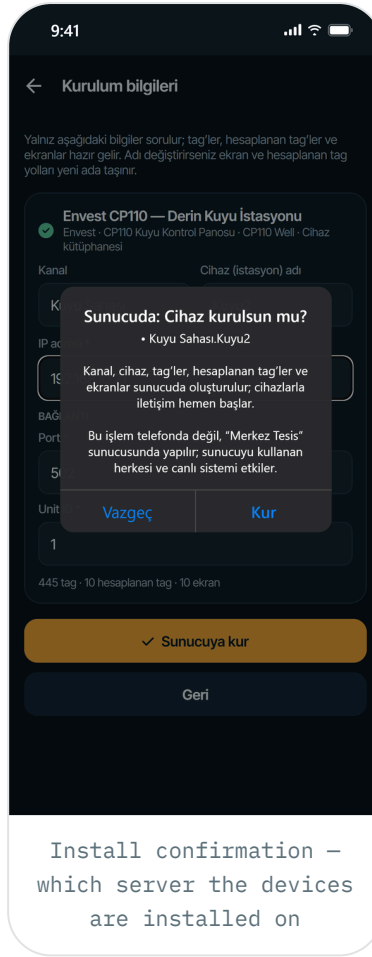
For each device only the package's questions are asked; tags, calculated tags and screens come ready:

- **Kanal** (Channel) and **Cihaz (istasyon) adı** (Device / station name) — the names in the package are suggested; if a name is taken the first free number is used (Well1 → Well2). When you change the name, tag paths on screens, alarms, calculated tag names and expressions, and the station name in screen names move to the new name.
- **Package parameters** — IP address, port, unit ID, FINS nodes or station parameters; fields marked ***** are required.

Under each card the app shows what will be installed (tags, calculated tags, screens, alarms) or the problems to fix. Installation is enabled only when no problem remains. Checked:

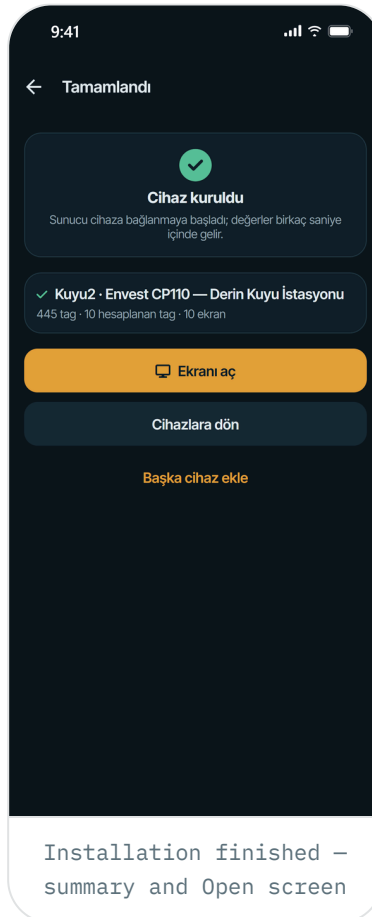
- A required parameter is empty or a placeholder is not filled (such as `{{ip}}`).
- The channel name is used on the server (or in the project) with another driver — choose another channel name. If it is used with the same driver, the device is added to that channel.
- A device with the same name exists in the channel, or two devices in the list have the same name.
- A calculated tag name already exists. Calculated tag names are unique across the server (or project); in library packages they start with the device name, so a different device name resolves the clash.
- The package's addresses and driver settings are validated against the driver.

Sunucuya kur (Install on server — **Projeye kur**, Install into project, in phone mode) starts the installation; in server mode the **Sunucuda:** confirmation comes first and lists the devices to be installed.



Install confirmation –
which server the devices
are installed on

INSTALLATION AND RESULT



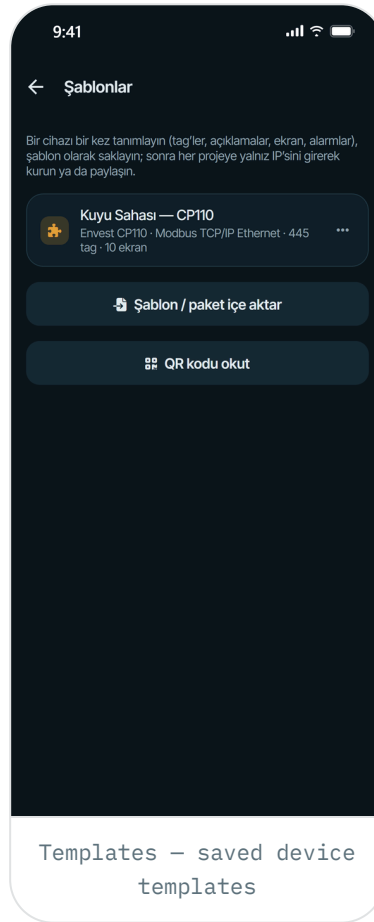
Installation finished –
summary and Open screen

In server mode the order is: channel (created if missing), the device with all its tags, calculated tags, runtime reload, alarms, screens. Devices with hundreds of tags take a few seconds; stay on the page. The result card shows, per device, the number of tags, calculated tags, screens and alarms installed, and any parts that could not be installed. **Ekrani aç** (Open screen) opens the installed main screen.

A package may carry two screen sets. **On the server** the server set is installed: one main screen with the settings screens under it. **On the phone** the phone set is installed: the Summary and its settings screens as tabs. Screen IDs are renewed at installation; installing the same package again does not overwrite existing screens.

Templates and sharing

Define a device once (tags, descriptions, screens, alarms), keep it as a template and later install it into other projects by entering only its IP address — or share it. Templates and packages use the same format (`.mhpkg.json`); see [Creating device templates](#).



Saving a template (phone mode): on the Kanallar tab open the device's ... menu and choose **Şablon olarak kaydet / paylaş**. Enter **Ad** (Name), **Açıklama** (Description), **Üretici** (Vendor), **Model** and **Hazırlayan** (Author); with **Ekranları da al** (Include screens) on, screens bound to this device go into the package. IP addresses in the device ID and settings are turned into the `{{ip}}` parameter automatically. Then:

- **Dosya olarak paylaş** (Share as file) — through the phone's share sheet (e-mail, messaging, files).
- **Panoya kopyala** (Copy to clipboard) — as text; handy for giving to an AI assistant.
- **Şablonlarıma kaydet** (Save to my templates) — into the template list on the phone.

To package the whole project use *Daha › Proje › Projeyi dışa aktar*.

The **templates page** (in phone mode, *Daha › Şablonlar ve paketler*): tap a template to install it; its menu has **Projeye kur** (Install into project), **Dosya olarak paylaş**, **Panoya kopyala**, **Sil. Şablon / paket içe aktar** (Import template / package) takes a package from a file, the clipboard, a QR code or an address; for project packages choose **Projeye ekle** (Add to project — devices with the same name are updated) or **Projeyi bununla değiştir** (Replace the project).

Sharing by QR code. The app scans QR codes but does not create them. Put the package at a reachable address and make a QR code for that address with any QR generator. QR contents the app understands: the package's `https:// address, mergenhub://paket?url=<address>` , or — for small packages — the JSON text itself.

Settings and security



Ayarlar (Settings — the gear on the home screen or *Daha › Ayarlar*):

- **Görünüm** (Appearance) — Mergen Gece (night), Mergen Gündüz (day) or Otomatik (follows the phone).
- **Sunucu modu › Canlı değer yenileme aralığı** (Live value refresh interval) — 0.5 / 1 / 2 / 5 s. The server keeps scanning a watched device for 8 s after the last request, so the interval stays below that.

- **İstemci modu** (Client mode) — raw TCP status and the TCP bridge for development builds.
- **Güvenlik › Uygulama kilidi** (Security › App lock) — asks for Face ID, Touch ID, fingerprint or the phone passcode when the app opens and after it has been in the background for a while. Re-lock after: immediately, 1, 5 or 15 minutes.
- **Güvenlik › Değer yazarken doğrula** (Authenticate when writing) — asks for authentication before writing a value or sending a command to a device.
- **Tarama sürerken ekranı açık tut** (Keep the screen on while scanning) — so scanning does not stop in phone mode.
- **Yazmadan önce onay iste** (Ask before writing) — on by default; a confirmation showing the value before every write.

When you turn on one of the security settings the app authenticates you once; if the phone has no screen lock the setting stays off.

Security principles:

- **The password is not stored on the phone.** The phone keeps only the session token the server issued for that device, in the encrypted store.
- **Administrators see every session.** *Settings › Web, REST API & Mobil* shows active users (web, mobile, API), mobile device sessions and per-user mobile access; a lost phone's session is closed with one click, or the user's mobile access is turned off. Web access is not affected.
- **The audit log** shows the source of every action; actions from the app appear as **Mobile**.
- **The app cannot change the server's mobile settings;** they are managed only in the web interface.
- **Permissions are the same as on the server;** the app never widens them.

Limitations

- Phone mode runs only while the app is in the foreground; it sends no notifications and does not keep alarm history. Trends are limited to the last few hundred samples (at most 1 hour).
- In phone mode OPC UA connects only to unsecured (None) endpoints; MQTT TLS is available only over WebSocket (`wss://`); the other TCP drivers have no TLS; OMRON FINS connects over TCP only.
- In server mode screens are not edited on the phone; design happens in the web interface.
- Servers older than MergenHub 3.5 have no device sessions, session list, push notifications or String tag writes; on 3.5.0 server screens are drawn in a simplified way.
- The app's user interface is in Turkish.

Troubleshooting

"Sunucuya ulaşılamadı" (server unreachable) or "Erişilemiyor" in the list. Is the phone on the same network as the server? If the server's Host is `localhost` , only that computer can reach it; set `0.0.0.0` and restart the service. Check that port 8090 is open in the firewall. If `http://<server-ip>:8090` opens in the phone's browser, the app can connect too. **Bağlantıyı dene** lists the addresses it tried and the error for each. Over the internet use a domain and HTTPS.

"Bu sunucuda mobil erişim kapalı" (mobile access is disabled on this server). An administrator must turn on **Allow sign-in from the mobile app** under *Settings › Web, REST API & Mobil*. If it affects only one user, that user's **Mobile sign-in** box is off.

The session dropped and the app asks for the password. An administrator closed the session, turned off the user's mobile access, or the session lifetime expired. Long-press the server and enter the **Parola** again.

No notifications. Go through the list in [Push notifications](#). The most common causes: notifications not allowed on the phone, no outbound internet from the server to `exp.host`, Mobile push off in the alarm definition, no access to the tag. The test button on the administrator's *Settings* > *Mobile push* page checks the chain end to end.

Values show "Bad" or "—". In server mode the problem is between the device and the server: look at the last error under *Daha* > *Cihaz durumu* and open the device's *Diagnostics* in the web interface ([Troubleshooting](#)). In phone mode: is scanning running (**Çalışıyor** in the header)? Is the phone on the same network as the device? On iOS, does MergenHub have **Local Network** permission (phone *Settings* > *Privacy & Security* > *Local Network*)? Are the device ID (IP, unit ID, rack/slot) and port right?

A value cannot be written. You need the RuntimeControl permission, the tag must be writable and you need write access to it (ACL). If the write button is missing, one of these is missing.

"SCADA yetkisi yok" on the Screens tab. Your user needs the Scada permission on the server.

A server screen looks simplified. The server runs MergenHub 3.5.0 or older; update to 3.5.1 or later.

The library does not open. The library is downloaded from the internet; offline the last list is shown but nothing can be installed. Download the package from the website and install it with **Dosya, pano, QR ya da adres**.

"Bu adla hesaplanan tag zaten var" (a calculated tag with this name already exists). The same device package was already installed with the same device name. Choose a different **Cihaz (istasyon) adı**.

Scanning stops by itself in phone mode. Scanning stops when the app goes to the background or the phone sleeps. Keep *Tarama sürerken ekranı açık tut* on and leave the app in the foreground.

Creating device templates (packages)

The .mhpkg.json format: parameters, channels-devices-tags, calculated tags and expression functions, alarms, server and phone screen sets, validation, a complete example, the AI skill, getting a device into the library.

A **device template** (MergenHub package) is a single JSON file that carries a device in full: channel and driver settings, the device, its register map (tags), calculated tags, alarms and operator screens. At installation only site-specific values (IP address, port, unit ID...) are asked; everything else comes ready. The file extension is `.mhpkg.json`.

This page explains how to build a package, what its fields mean and what happens at installation.

Where packages are used

- **MergenHub web interface** — the *Device Library* dialog (the books icon in the project tree header, the **Device Library** button in the *Connectivity* view, or the card in the SCADA new-screen chooser). It installs from the library, and **Import from file** installs a `.mhpkg.json` file you have.
- **MergenHub Mobile** — the *Add device* wizard (library, my templates, file, clipboard, QR, address), the *Templates* page and *Import package / template*. See [MergenHub Mobile](#).
- **Device Library** — packages prepared and tested by Envest are published [on the website](#), and the apps read them from a public API: `GET /api/v1/library` (catalogue), `GET /api/v1/library/{slug}` (one device), `GET /api/v1/library/{slug}/package` (the package file).

Cihaz Kütüphanesi
×

Marka, model, yazılım sürümü, protokol...

Dosyadan içe aktar

Tümü · 11

Su Yönetimi · 11

SONDAJ KUYUSU
Envest CP110 — Derin Kuyu İstasyonu
 Envest · CP110 Kuyu Kontrol Panosu · CP110 Well
 Modbus TCP/IP Ethernet · 445 tag · 10 hesaplanan · 10 ekran

☐ Seç

SU DEPOSU
Envest CP110 — Su Dağıtım Deposu
 Envest · CP110 Depo Kontrol Panosu · CP110 Store
 Modbus TCP/IP Ethernet · 560 tag · 4 hesaplanan · 9 ekran

☐ Seç

TERFİ İSTASYONU
Envest CP110 — Terfi Pompa İstasyonu
 Envest · CP110 Terfi Kontrol Panosu · CP110 Riser
 Modbus TCP/IP Ethernet · 486 tag · 7 hesaplanan · 10 ekran

☐ Seç

SONDAJ KUYUSU
AQUA CNT 100+ — Kuyu Pompa Kontrolü
 Envest · AQUA CNT 100+ · AquaCnt V2 — Kuyu
 Modbus TCP/IP Ethernet · 159 tag · 11 hesaplanan · 10 ekran

☐ Seç

SU DEPOSU
AQUA CNT 100+ — Depo İzleme
 Envest · AQUA CNT 100+ · AquaCnt V2 — Depo
 Modbus TCP/IP Ethernet · 159 tag · 5 hesaplanan · 9 ekran

☐ Seç

TERFİ İSTASYONU
AQUA CNT 100+ — Terfi Pompa Kontrolü

☐ Seç

Kaynak: mergenhub.envest.com.tr/devices · Cihazınız yok mu? Ekleme isteği gönderin

İptal

Seçilenleri kur

The Device Library dialog in the MergenHub web interface

On a server without internet access, download the package from the website and install it with **Import from file**.



Kurulum bilgileri

Yalnız aşağıdakiler sorulur; tag'ler, hesaplanan tag'ler ve ekranlar hazır gelir. Adı değiştirirseniz ekran ve hesaplanan tag yolları yeni ada taşınır.

Envest CP110 — Su Dağıtım Deposu
Envest · CP110 Depo Kontrol Panosu · CP110 Store · envest-cp110-depo.mhpkg.json

Kanal

Cihaz (istasyon) adı

Korubin-Depo

Depo2

IP adresi *

192.168.1.50

BAĞLANTI

Port *

Unit ID *

502

1

• IP adresi: gerekli

İptal
Kur

Import from file — installation details and check result

Quick start

The easiest way is to start from a working device:

1. Open MergenHub Mobile in **phone mode**, define the device with its channel, device and a few tags, and check that values arrive.
2. On the Kanallar (Channels) tab open the device's ... menu and choose **Şablon olarak kaydet / paylaş** (Save / share as template). Enter name, description, vendor and model; leave **Ekranları da al** (Include screens) on if screens should come along.
3. Get the package with **Dosya olarak paylaş** (Share as file) or **Panoya kopyala** (Copy to clipboard). IP addresses in the device ID and settings have been turned into the `{{ip}}` parameter automatically.

4. Edit the file in a text editor or with an AI assistant (add tags, write descriptions, add alarms and screens). The `guide` section of an exported file contains the format rules, the setting keys of the drivers used, address examples and the symbol list.
5. Bring the package back: on the phone *Cihaz ekle* › *Dosya, pano, QR ya da adres* (Add device › File, clipboard, QR or address), on the server *Device Library* › *Import from file*.



To package a whole project use *Daha* › *Proje* › *Projeyi dışa aktar* (More › Project › Export project) on the phone (`kind: "project"`).

File structure

A package is a single JSON object:

```
{
  "format": "mergenhub-package",
  "formatVersion": 1,
  "kind": "template",
  "meta": { "name": "..." },
  "parameters": [],
  "channels": [],
  "derived": [],
  "alarms": [],
  "screens": [],
  "notes": "..."
}
```

FIELD	REQUIRED	DESCRIPTION
format	yes	Always "mergenhub-package"
formatVersion	recommended	Currently 1 ; a package with a higher version than the app knows is rejected
kind	yes	"template" (device template, parameters asked at installation) or "project" (a whole project)
meta	yes	Name and descriptive information
parameters	no	Values asked at installation
channels	yes	Channel → device → tag tree (at least one channel)
derived	no	Calculated (advanced) tags
alarms	no	Alarm definitions
screens	no	Operator screens
notes	no	Free text (Markdown) describing what the device does
guide	no	Explanatory section written on export; ignored on import

A device template contains one channel with one device. Only then are the channel and device names asked at installation, and paths move when the name changes.

META

FIELD	DESCRIPTION
name	Display name of the package (required), e.g. "Example EA-100 Energy Meter"
description	A one- or two-sentence summary
vendor , model , version	Manufacturer, model and device firmware version; shown in lists and search
author	Who prepared it
keywords	Search keywords (array)
homepage	Product page address
createdAt , createdWith	Written automatically on export

Parameters and placeholders

parameters are the fields the user is asked at installation. Wherever `{{key}}` appears in any **string** value of the package, it is replaced with the value the user entered.

```
"parameters": [
  { "key": "ip", "label": "IP address", "type": "text", "required": true, "placeholder":
    "192.168.1.50" },
  { "key": "port", "label": "Port", "type": "number", "default": 502, "group": "Connection" },
  { "key": "unit", "label": "Unit ID", "type": "number", "default": 1, "group": "Connection" }
]
```

FIELD	DESCRIPTION
key	Starts with a letter or <code>_</code> ; contains letters, digits and <code>_</code> (e.g. <code>ip</code> , <code>port</code> , <code>unit</code> , <code>fins_node</code>)
label	Name shown in the form
type	<code>text</code> (default), <code>number</code> or <code>select</code>
default	Default value
required	If <code>true</code> the field cannot be left empty
placeholder , hint	In-field example and help text below
group	Heading in the form; fields of the same group are shown together
options	For <code>select</code> : <code>[{ "value": 1, "label": "TCP" }, ...]</code>

Rules:

- Placeholders work only inside string values. To bind a numeric setting to a parameter, write it as a string: `"modbus_ethernet.DEVICE_ETHERNET_PORT_NUMBER": "{{port}}"` . Numeric driver settings are converted to numbers at installation.
- The device ID is typically built from parameters: `"id": "<{{ip}}>.{unit}"` .
- If any `{{...}}` is left unfilled after applying the values, installation is blocked.
- Leave the IP address default empty and set `required: true` ; put an example address in `placeholder` . That way nobody can install without entering the address.

Channels, devices and tags

```
"channels": [
  {
    "name": "Energy",
    "driver": "Modbus TCP/IP Ethernet",
    "settings": {},
    "devices": [
      {
        "name": "Meter1",
        "id": "<{{ip}}>.{unit}",
        "scanMs": 1000,
        "timeouts": { "connectS": 3, "requestMs": 1000, "retries": 3 },
        "settings": { "modbus_ethernet.DEVICE_ETHERNET_PORT_NUMBER": "{{port}}" },
        "tags": [
          { "name": "VoltageL1", "address": "30001", "type": "Float", "access": "read", "scanMs":
1000, "units": "V" }
        ],
        "groups": []
      }
    ]
  }
]
```

Channel

FIELD	DESCRIPTION
name	Channel name (required, no dots)

FIELD	DESCRIPTION
<code>driver</code>	Full driver name (table below)
<code>settings</code>	Channel-level driver settings (Kepware-compatible keys)
<code>description</code>	Description
<code>devices</code>	Array of devices (required)

Device

FIELD	DEFAULT	DESCRIPTION
<code>name</code>	—	Device name (required, no dots)
<code>id</code>	per driver	Device ID; the format depends on the driver
<code>description</code>	—	Description
<code>enabled</code>	<code>true</code>	If <code>false</code> the device is not scanned
<code>scanMs</code>	1000	Device scan rate (ms)
<code>scanMode</code>	4	0 respect client, 1 no faster than, 2 fixed rate, 3 demand poll, 4 respect tag rate
<code>timeouts</code>	3 / 1000 / 3	<code>connectS</code> connect timeout (s), <code>requestMs</code> request timeout (ms), <code>retries</code> retry attempts
<code>settings</code>	driver defaults	Device-level driver settings
<code>tags</code> , <code>groups</code>	—	Tags and tag groups

Tag

FIELD	DEFAULT	DESCRIPTION
<code>name</code>	—	Tag name (required, no dots, unique within the device or group)
<code>address</code>	—	Address in the driver's syntax (required)
<code>type</code>	<code>Default</code>	Data type (below)
<code>access</code>	<code>read</code>	<code>read</code> or <code>readwrite</code>
<code>scanMs</code>	1000	Tag scan rate (ms)
<code>description</code>	—	Description; shown as a unit in the app if 6 characters or fewer
<code>units</code>	—	Unit; used as the description when <code>description</code> is empty
<code>scaling</code>	—	Scaling (below)

Data types: `Default` , `Boolean` , `Char` , `Byte` , `Short` , `Word` , `Long` , `DWord` , `Float` , `Double` , `String` , `BCD` , `LBCD` , `Date` , `LLong` , `QWord` . `Default` is the type the driver chooses for the address.

Scaling converts the raw value to engineering units; writes apply it in reverse:

```
"scaling": { "type": "linear", "rawLow": 0, "rawHigh": 10, "low": 0, "high": 1, "resultType": "Double", "units": "kWh" }
```

type is linear or sqrt (square root); rawLow / rawHigh the raw range, low / high the scaled range; resultType the scaled data type (default Double); clampLow , clampHigh clamp at the limits, negate inverts the sign.

Groups organise tags in folders inside the device and can be nested. A tag in a group has the path Channel.Device.Group.Tag .

```
"groups": [ { "name": "Energy", "description": "Counters", "tags": [ ... ], "groups": [ ... ] } ]
```

DRIVERS AND ADDRESS SYNTAX

driver takes the driver's full name. Address syntax and setting keys are the same in the MergenHub server and MergenHub Mobile; for details see [Protocol drivers](#).

DRIVER (DRIVER)	DEVICE ID (ID)	ADDRESS EXAMPLES
Simulator	device number, e.g. 1	SINE:10:0:100 , RAMP:30:0:500 , RANDOM:20:80 , REG:1
Modbus TCP/IP Ethernet	<IP>.Unit , e.g. <192.168.1.10>.1	40001 holding, 30001 input register, 00001 coil, 10001 discrete input, 40001.3 bit, 40010.16H string
Modbus RTU Serial , Modbus ASCII Serial	slave address 1 – 247	same as Modbus TCP
Siemens S7 TCP/IP	<IP>.Rack.Slot , e.g. <192.168.0.1>.0.1	DB1.DBD4 , DB1.DBW2 , DB1.DBB0 , DB1.DBX0.3 , MW10 , IO.0 , Q0.0
OMRON FINS Ethernet	<IP>.Node , e.g. <192.168.1.10>.1	D100 , D0.05 , CI010 , W5.00 , H2 , A3 , E100
OMRON NJ Ethernet	<IP>	PumpState , Line1.Valve[3] , Matrix[1,2] , Counter.5
MQTT Client	broker host:port (ignored if the channel sets a broker)	sensors/temp , sensors/pump1:/flow , plant/csv;idx=2;sep=;
DNP Client Ethernet	outstation IP	AI0 , BI5 , CTR2 , A01 , B03 , 30.1.51.Value
IEC 60870-5-104 Client	RTU IP	100 , IOA100 , M_ME_NC.100.VALUE , C_SC_NA.3000.DIRECTVALUE
IEC 61850 MMS Client	IED IP	LD0/MMXU1.MX.TotW.mag.f , LD0/XCBR1.ST.Pos.stVal
OPC UA Client	not used (endpoint is a channel setting)	ns=2;s=Channel.Device.Tag , ns=2;i=1001

Frequently used setting keys:

- **Modbus TCP** (device): modbus_ethernet.DEVICE_ETHERNET_PORT_NUMBER (502), modbus_ethernet.DEVICE_ZERO_BASED_ADDRESSING , modbus_ethernet.DEVICE_FIRST_WORD_LOW (word order for 32-bit values), modbus_ethernet.DEVICE_HOLDING_REGISTERS (block size, 1–125).
- **Modbus RTU/ASCII** (device): servermain.DEVICE_ETHERNET_COMMUNICATIONS_IP and servermain.DEVICE_ETHERNET_COMMUNICATIONS_PORT (serial server), plus modbus.DEVICE_... keys.

- **Siemens S7** (device): `siemens_tcpip_ethernet.DEVICE_COMMUNICATIONS_PORT_NUMBER` (102), `siemens_tcpip_ethernet.DEVICE_S7_COMMUNICATIONS_CPU_RACK`, `..._CPU_SLOT`, and for the model `servermain.DEVICE_MODEL` (0 S7-200, 1 S7-300, 2 S7-400, 3 S7-1200, 4 S7-1500).
- **OMRON FINS**: on the channel `omron_fins_ethernet.CHANNEL_PORT_NUMBER` (9600); on the device `omron_fins_ethernet.DEVICE_PROTOCOL` (0 UDP, 1 TCP) and the node numbers.
- **MQTT** (channel): `mqtt.BROKER_HOST`, `mqtt.PORT`, `mqtt.USERNAME`, `mqtt.PASSWORD`, `mqtt.QOS`, `mqtt.TRANSPORT`.
- **OPC UA client** (channel): `opcuaclient.CHANNEL_UA_SERVER_ENDPOINT_URL`.

A setting you leave out takes the driver default. All setting keys of a driver, with explanations, are in the `guide.drivers` section of any package exported with that driver.

Calculated tags

The `derived` array defines tags calculated from other tags. Once installed their path is `_AdvancedTags`. `<name>`; screens and alarms use that path.

```
"derived": [
  {
    "name": "Meter1_ApparentPower",
    "type": "Derived",
    "expression": "ROUND({Energy.Meter1.VoltageL1} * {Energy.Meter1.CurrentL1} / 1000, 2)",
    "scanRateMs": 1000,
    "description": "Apparent power (kVA)"
  }
]
```

FIELD	DESCRIPTION
<code>name</code>	Name; unique across the server — start it with the device name (<code><Device>_...</code>)
<code>type</code>	<code>Derived</code> (expression), <code>Totalizer</code> , <code>Average</code> , <code>Maximum</code> , <code>Minimum</code>
<code>expression</code>	For <code>Derived</code> the expression; for the others the source tag path (without braces)
<code>sampleSize</code>	Sample count for average / maximum / minimum (default 10)
<code>scanRateMs</code>	Update rate (ms, default 1000)
<code>enabled</code>	Default <code>true</code>
<code>description</code>	Description
<code>dataType</code>	Result data type code on the server (default 9 = Double); ignored on the phone

A totalizer sums value × seconds; for an m³/h input divide the result by 3600.

FUNCTION REFERENCE

In an expression a tag reference is written `{Channel.Device.Tag}` (with a group, `{Channel.Device.Group.Tag}`). Operators: `+`, `-`, `*`, `/`, `%`, comparisons `<`, `<=`, `>`, `>=`, `==`, `!=` (1 if true, 0 if false), logic `&&`, `||`, `!` and parentheses. The decimal separator is always a dot (`1.5`). Function names are case-insensitive; functions without arguments still need parentheses (`PI()`). An expression can be at most 8000 characters. The function list is the same on the server and the phone.

FUNCTION	DESCRIPTION
<code>ABS(x)</code>	Absolute value
<code>SQRT(x)</code>	Square root (0 for negative input)
<code>POW(x, y)</code>	x to the power y
<code>EXP(x)</code> , <code>LN(x)</code>	e to the power x, natural logarithm
<code>LOG(x)</code> , <code>LOG(x, base)</code>	Base-10 or given-base logarithm
<code>ROUND(x)</code> , <code>ROUND(x, digits)</code>	Rounding (half away from zero; 0–15 digits)
<code>FLOOR(x)</code> , <code>CEIL(x)</code>	Round down / up
<code>MOD(a, b)</code>	Remainder (0 if b = 0)
<code>SIGN(x)</code>	Sign: -1, 0, 1
<code>MIN(a, b, ...)</code> , <code>MAX(a, b, ...)</code>	Smallest / largest
<code>AVG(a, b, ...)</code> , <code>SUM(a, b, ...)</code>	Average / sum
<code>IF(cond, a, b)</code>	a if cond is non-zero, otherwise b
<code>CLAMP(x, low, high)</code>	Limit to a range
<code>MAP(x, inLow, inHigh, outLow, outHigh)</code>	Linear scaling
<code>LERP(a, b, t)</code>	Linear interpolation between a and b
<code>POLY(x, c0, c1, c2, ...)</code>	Polynomial: $c0 + c1 \cdot x + c2 \cdot x^2 + \dots$
<code>HYPOT(a, b)</code>	$\sqrt{a^2 + b^2}$
<code>SIN</code> , <code>COS</code> , <code>TAN</code> , <code>ASIN</code> , <code>ACOS</code> , <code>ATAN</code>	Trigonometry (radians)
<code>ATAN2(y, x)</code>	Two-argument arctangent
<code>SINH</code> , <code>COSH</code> , <code>TANH</code>	Hyperbolic functions
<code>DEG(x)</code> , <code>RAD(x)</code>	Radians to degrees, degrees to radians
<code>PI()</code> , <code>E()</code>	Constants
<code>RANDOM()</code> , <code>RANDOM(high)</code> , <code>RANDOM(low, high)</code> , <code>RND(...)</code>	Random number; for testing only

Examples: `({C.D.A} + {C.D.B}) / 2` , `IF({C.D.Pressure} > 6, 1, 0)` , `MAX({C.D.Level} - {C.D.Static}, 0)` , `MAP({C.D.Raw}, 4, 20, 0, 10)` .

Alarms

```
"alarms": [
  { "name": "High voltage L1", "tag": "Energy.Meter1.VoltageL1", "conditionType": "Hi", "limit":
250,
    "deadband": 2, "severity": 700, "message": "Phase 1 voltage above 250 V", "enabled": true }
]
```

FIELD	DEFAULT	DESCRIPTION
<code>name</code>	—	Alarm name

FIELD	DEFAULT	DESCRIPTION
tag	—	Full path of a tag in the package, or <code>_AdvancedTags.<name></code> (required, must exist in the package)
conditionType	—	HiHi , Hi , Lo , LoLo , DevHi , DevLo (deviation), RocHi , RocLo (rate of change, units/s)
limit	—	Limit
setpoint	0	Setpoint for deviation alarms
deadbandType , deadband	Absolute , 0	Deadband: None , Absolute , Percent
rangeLow , rangeHigh	0, 100	Range for a percent deadband
onDelaySec , offDelaySec	0	On and off delay (s)
severity	500	1–1000; 800 and above is critical
message , area	—	Message and area
enabled	true	Enabled

Screens

Screens have the same structure as the server's SCADA screens. A screen consists of objects (`widgets`) placed in a coordinate space (`width` × `height`).

FIELD	DESCRIPTION
id	Unique within the package (required); renewed at installation, links are moved
name	Screen name; if it contains the device name, it is renamed at installation
target	"server" (installed only on the server), "phone" (only in phone mode) or absent (both)
parentId	id of the parent screen; empty for a main screen
order	Order among sub-screens
width , height	Coordinate space; phone portrait 400 × 700 (taller for long screens), server e.g. 1200 × 700
background	Background colour (<code>#eef2f7</code>); on phone screens leave it empty to follow the theme
group , zoneId	Screen group and zone on the server; leave empty in templates
widgets	Objects

TWO SCREEN SETS

A good device package carries two sets:

- **Server set** (`"target": "server"`) — **one main screen** (empty `parentId`) with the settings screens under it (`parentId` = the main screen's `id`). In the web interface the main screen appears in the list and the settings screens in its branch. Put buttons (`nav`) from the main screen to the settings screens and back.

- **Phone set** ("target": "phone") — a portrait **Summary** screen (measurements, controls and buttons to the settings screens) with single-column settings screens as its sub-screens. In phone mode a tab strip with the Summary and the sub-screens appears on top. Name the screens `<Device> – Summary` , `<Device> – Settings` ; the tabs show the part after "—".

At installation only the set matching the target is installed. A screen whose parent belongs to the other set becomes a main screen; `nav` links into the other set are removed. A screen without `target` is installed in both places.

OBJECTS

```
{ "id": "w2", "type": "display", "x": 24, "y": 80, "w": 220, "h": 80,
  "tag": "Energy.Meter1.VoltageL1", "props": { "label": "Voltage L1", "units": "V", "decimals": 1
} }
```

FIELD	DESCRIPTION
id	Unique within the package (e.g. <code>w1</code> , <code>w2</code> ...)
type	Object type (below)
x , y , w , h	Position and size (in screen coordinates)
rot	Rotation (degrees)
tag	Bound tag path: <code>Channel.Device.Tag</code> or <code>_AdvancedTags.<name></code>
props	Properties depending on the type
anim	Conditional rules
onClick , nav	To go to another screen on tap: <code>"onClick": "nav"</code> and <code>"nav": "<screen id>"</code>
popupScreen	Screen shown as a popup (server)

TYPE	WHAT IT DOES	KEY PROPS
display	Value display	label , units , decimals , color , fontSize
label	Static text (no tag)	text , fontSize , color , align
lamp	Lamp	label , onColor , offColor , onText , offText , blinkOn , onOp / onA / onB
button	Writes <code>onValue</code> / <code>offValue</code> when pressed	label , onValue , offValue , onText , offText , onColor , offColor
switch	Switch (writes like a button)	label , onValue , offValue , onText , offText , onColor
radio	Option button	label , onValue , offValue
input	Value input (writable tag)	label , decimals
sfield	Setting row: label + value (+ unit)	label , units , decimals , editable , step
gauge	Dial gauge	label , min , max , units , decimals , fillColor
tank , levelbar	Tank level, level bar	label , min , max , units , decimals , fillColor

TYPE	WHAT IT DOES	KEY PROPS
<code>pump</code> , <code>motor</code> , <code>fan</code> , <code>valve</code>	Machine icons (spin / change colour when running)	<code>label</code> , <code>onColor</code> , <code>offColor</code> , <code>onText</code> , <code>offText</code> , <code>spin</code> , <code>blinkOn</code> , <code>onOp</code> / <code>onA</code> / <code>onB</code>
<code>pipe</code>	Pipe; coloured when flowing	<code>thickness</code> , <code>flowColor</code> , <code>onOp</code> / <code>onA</code>
<code>rect</code>	Panel / rectangle (no tag)	<code>fill</code> , <code>stroke</code> , <code>radius</code>
<code>image</code>	Image (no tag)	<code>src</code> (data URI or address)
<code>svg</code>	Object from the symbol library	<code>lib2</code> (symbol id), <code>label</code> , <code>onOp</code> / <code>onA</code> / <code>onB</code> , <code>onColor</code> , <code>offColor</code>

On state. For lamps, machines and symbols, when the object counts as "on" is set with `onOp` (`gt`, `ge`, `eq`, `ne`, `lt`, `le`, `between`), `onA` and `onB` ; without them the object is on when the value is greater than 0.5.

Conditional rules (`animations`): { "kind": "color", "tag": "", "op": "gt", "a": 80, "b": null, "p2": "#ef4444" } . `kind` : `color` (sets the colour to `p2`), `blink`, `visible` (**hides** the object while the condition is true), `level` (fill: `a` minimum, `b` maximum, `p2` colour, `op` always). An empty `tag` means the object's own tag.

Symbols. The `lib2` field of an `svg` object is an id from the built-in symbol library (e.g. `en-pumps/en_pump_submersible_1`, `en-meters/en_flowmeter_1`, `pumps_centrifugal_pump_4`). The full list of ids is in the `guide.symbols` section of exported packages.

Server screens may also use the other object types of the web SCADA editor; the phone shows them as faded boxes and the check gives a warning. For screens that must work in both places, use the types in the table.

Naming and renaming at installation

- Channel, device, group and tag names **contain no dots** (`.`); the dot is the path separator. Names must be unique at their level (case-insensitive).
- A **tag path** is `Channel.Device.Tag` (in a group `Channel.Device.Group.Tag`). Screens, alarms and expressions always use the full path.
- A **device name** is unique only within its channel; the same package can be installed into the same channel many times with different device names.
- Calculated tag names are unique across the server.** Start every calculated tag's name with the device name: `Meter1_ApparentPower` . When the device name changes at installation the prefix changes too (`Meter2_ApparentPower`), so the same package installs a second time without a clash.

If the user changes the channel and device name at installation, the package is updated as follows:

- All paths starting with `Channel.Device` (object `tag` s on screens, rules, alarm `tag` s, `{...}` references in expressions) move to the new name.
- Calculated tags whose name starts with `<OldDevice>_` are renamed to `<NewDevice>_`, and `_AdvancedTags....` references move with them.
- The old device name in screen names and in the `text` of `label` objects (as a whole word, if at least 3 characters long) is replaced with the new one.

Clashes checked at installation:

- If the channel name is used on the target with another driver, installation is blocked; if it is used with the same driver, the device is added to the existing channel.
- If a device with the same name exists in the channel, installation is blocked; the app suggests the first free numbered name (Meter1 → Meter2).
- If a calculated tag name already exists on the target, installation is blocked; a different device name resolves it.

Validation and common errors

MergenHub Mobile checks a package before installing it; problems are listed on the installation card. The web interface's **Import from file** checks parameters, names and clashes but does not validate addresses against the driver, so it is good practice to try a new package on the phone first.

Errors (installation blocked):

- `Bu bir MergenHub paketi değil` (not a MergenHub package) — the `format` field is missing or wrong.
- `JSON hatası (satır ..., sütun ...)` (JSON error at line, column) — a syntax error; the position and surrounding text are shown.
- `Bilinmeyen sürücü "..."` (unknown driver) — write the driver's full name in `driver` (e.g. `Modbus TCP/IP Ethernet`).
- `Adres geçersiz (...)` (invalid address) — the address does not match the driver's syntax; the message says why (e.g. for Modbus the area digit must be 0, 1, 3 or 4).
- `Bilinmeyen veri tipi "..."` (unknown data type) — type names are written as in the list above (`Float` , not `float32`).
- `access "read" ya da "readwrite" olmalı` (access must be read or readwrite).
- `... adı tekrar ediyor` (duplicate name), `... adında nokta olamaz` (name contains a dot), `... adı boş` (empty name).
- `Alarm ...: Tag bulunamadı` (tag not found) — the alarm's tag is not in the package.
- `Geçersiz parametre anahtarı` (invalid parameter key) — the key contains characters other than letters, digits and `_`.
- `{{...}}` için değer girilmedi (no value for a placeholder) — a placeholder has no parameter or the value is empty.

Warnings (installation proceeds, but have a look):

- `Paket içinde olmayan tag'e bağlı` (bound to a tag not in the package) — a screen object, rule or expression uses a path that is not in the package; usually a typo.
- `Gidilecek ekran pakette yok` (target screen not in the package) — `nav` does not match any screen `id`.
- `Bu uygulamanın tanımadığı nesne tipi` (object type unknown to this app) — the phone cannot draw it; it may work on the server.
- `Sembol kütüphanede yok` (symbol not in the library) — wrong `lib2` id.

Expressions are compiled by the server at installation; a broken expression (unknown function, wrong argument count, a comma decimal such as `1,5`) is listed as "not installed" in the result, and the other parts are installed.

The app tolerates `//` and `/* */` comments and trailing commas, but the web interface does not accept comments. Write packages as plain, valid JSON without comments.

Complete example

A single Modbus TCP energy meter: six tags (one scaled, two writable), one calculated tag, one alarm, a main screen and a settings screen in the server set, a Summary and a settings screen in the phone set. This package passes the MergenHub Mobile check with no errors and no warnings.

```

{
  "format": "mergenhub-package",
  "formatVersion": 1,
  "kind": "template",
  "meta": {
    "name": "Example EA-100 Energy Meter",
    "description": "Single-phase energy metering, relay output and CT ratio setting.",
    "vendor": "Example",
    "model": "EA-100",
    "version": "1.0",
    "author": "Envest",
    "keywords": ["energy", "meter", "Modbus TCP"]
  },
  "parameters": [
    { "key": "ip", "label": "IP address", "type": "text", "required": true, "placeholder": "192.168.1.50", "hint": "Address of the meter on the network" },
    { "key": "port", "label": "Port", "type": "number", "default": 502, "group": "Connection" },
    { "key": "unit", "label": "Unit ID", "type": "number", "default": 1, "group": "Connection" }
  ],
  "channels": [
    {
      "name": "Energy",
      "driver": "Modbus TCP/IP Ethernet",
      "devices": [
        {
          "name": "Meter1",
          "id": "<{{ip}}>{{unit}}",
          "scanMs": 1000,
          "timeouts": { "connectS": 3, "requestMs": 1000, "retries": 3 },
          "settings": { "modbus_ethernet.DEVICE_ETHERNET_PORT_NUMBER": "{{port}}" },
          "tags": [
            { "name": "VoltageL1", "address": "30001", "type": "Float", "access": "read", "scanMs": 1000, "units": "V" },
            { "name": "CurrentL1", "address": "30003", "type": "Float", "access": "read", "scanMs": 1000, "units": "A" },
            { "name": "ActivePower", "address": "30005", "type": "Float", "access": "read", "scanMs": 1000, "units": "kW" },
            { "name": "TotalEnergy", "address": "30007", "type": "DWord", "access": "read", "scanMs": 5000, "scaling": { "type": "linear", "rawLow": 0, "rawHigh": 10, "low": 0, "high": 1, "resultType": "Double", "units": "kWh" } },
            { "name": "RelayOutput", "address": "00001", "type": "Boolean", "access": "readwrite", "scanMs": 1000, "description": "Relay output (1 = energised)" },
            { "name": "CtRatio", "address": "40001", "type": "Word", "access": "readwrite", "scanMs": 5000, "description": "Current transformer ratio (e.g. 100 = 100/5 A)" }
          ]
        }
      ]
    }
  ],
  "derived": [
    {
      "name": "Meter1_ApparentPower",
      "type": "Derived",
      "expression": "ROUND({Energy.Meter1.VoltageL1} * {Energy.Meter1.CurrentL1} / 1000, 2)",
      "scanRateMs": 1000,
      "description": "Apparent power (kVA) = voltage × current / 1000"
    }
  ],
  "alarms": [
  ]
}

```

```

        "name": "High voltage L1",
        "tag": "Energy.Meter1.VoltageL1",
        "conditionType": "Hi",
        "limit": 250,
        "deadband": 2,
        "severity": 700,
        "message": "Phase 1 voltage above 250 V",
        "enabled": true
    }
},
"screens": [
    {
        "id": "ea100-main", "name": "Meter1 – Energy Meter", "target": "server", "parentId": "",
"order": 0,
        "width": 1200, "height": 700, "background": "#eef2f7",
        "widgets": [
            { "id": "w1", "type": "label", "x": 24, "y": 16, "w": 700, "h": 40, "props": { "text":
"ENERGY METER – Meter1", "fontSize": 24, "color": "#0f172a" } },
            { "id": "w2", "type": "display", "x": 24, "y": 80, "w": 220, "h": 80, "tag":
"Energy.Meter1.VoltageL1", "props": { "label": "Voltage L1", "units": "V", "decimals": 1 } },
            { "id": "w3", "type": "display", "x": 264, "y": 80, "w": 220, "h": 80, "tag":
"Energy.Meter1.CurrentL1", "props": { "label": "Current L1", "units": "A", "decimals": 2 } },
            { "id": "w4", "type": "display", "x": 504, "y": 80, "w": 220, "h": 80, "tag":
"Energy.Meter1.ActivePower", "props": { "label": "Active power", "units": "kW", "decimals": 2 } },
            { "id": "w5", "type": "display", "x": 744, "y": 80, "w": 220, "h": 80, "tag":
"Energy.Meter1.TotalEnergy", "props": { "label": "Total energy", "units": "kWh", "decimals": 1 }
        },
            { "id": "w6", "type": "gauge", "x": 24, "y": 180, "w": 240, "h": 220, "tag":
"_AdvancedTags.Meter1_ApparentPower", "props": { "label": "Apparent power", "min": 0, "max": 50,
"units": "kVA", "decimals": 2 } },
            { "id": "w7", "type": "switch", "x": 284, "y": 180, "w": 200, "h": 70, "tag":
"Energy.Meter1.RelayOutput", "props": { "label": "Relay", "onValue": 1, "offValue": 0, "onText":
"ON", "offText": "OFF", "onColor": "#22c55e" } },
            { "id": "w8", "type": "rect", "x": 24, "y": 620, "w": 200, "h": 56, "props": { "fill":
"#1d4ed8", "stroke": "#1d4ed8", "radius": 10 }, "onClick": "nav", "nav": "ea100-settings" },
            { "id": "w9", "type": "label", "x": 24, "y": 620, "w": 200, "h": 56, "props": { "text":
"Settings", "fontSize": 16, "color": "ffffff" }, "onClick": "nav", "nav": "ea100-settings" }
        ]
    },
    {
        "id": "ea100-settings", "name": "Meter1 – Settings", "target": "server", "parentId": "ea100-
main", "order": 1,
        "width": 1200, "height": 700, "background": "#eef2f7",
        "widgets": [
            { "id": "w10", "type": "label", "x": 24, "y": 16, "w": 700, "h": 40, "props": { "text":
"SETTINGS – Meter1", "fontSize": 24, "color": "#0f172a" } },
            { "id": "w11", "type": "sfield", "x": 24, "y": 80, "w": 520, "h": 44, "tag":
"Energy.Meter1.CtRatio", "props": { "label": "CT ratio", "decimals": 0, "editable": true, "step":
5 } },
            { "id": "w12", "type": "rect", "x": 24, "y": 620, "w": 200, "h": 56, "props": { "fill":
"#1d4ed8", "stroke": "#1d4ed8", "radius": 10 }, "onClick": "nav", "nav": "ea100-main" },
            { "id": "w13", "type": "label", "x": 24, "y": 620, "w": 200, "h": 56, "props": { "text":
"< Main screen", "fontSize": 16, "color": "ffffff" }, "onClick": "nav", "nav": "ea100-main" }
        ]
    },
    {
        "id": "ea100-phone", "name": "Meter1 – Summary", "target": "phone", "parentId": "", "order":
0,
        "width": 400, "height": 700, "background": "",
        "widgets": [
            { "id": "w14", "type": "rect", "x": 0, "y": 0, "w": 400, "h": 64, "props": { "fill":
"#1d4ed8", "stroke": "#1d4ed8", "radius": 0 } },

```

```

    { "id": "w15", "type": "label", "x": 14, "y": 8, "w": 372, "h": 48, "props": { "text":
"Energy Meter", "fontSize": 18, "color": "#ffffff", "align": "left" } },
    { "id": "w16", "type": "display", "x": 8, "y": 76, "w": 188, "h": 62, "tag":
"Energy.Meter1.VoltageL1", "props": { "label": "Voltage L1", "units": "V", "decimals": 1 } },
    { "id": "w17", "type": "display", "x": 204, "y": 76, "w": 188, "h": 62, "tag":
"Energy.Meter1.CurrentL1", "props": { "label": "Current L1", "units": "A", "decimals": 2 } },
    { "id": "w18", "type": "display", "x": 8, "y": 146, "w": 188, "h": 62, "tag":
"Energy.Meter1.ActivePower", "props": { "label": "Active power", "units": "kW", "decimals": 2 } },
    { "id": "w19", "type": "display", "x": 204, "y": 146, "w": 188, "h": 62, "tag":
"Energy.Meter1.TotalEnergy", "props": { "label": "Total energy", "units": "kWh", "decimals": 1 }
},
    { "id": "w20", "type": "gauge", "x": 106, "y": 220, "w": 188, "h": 178, "tag":
"_AdvancedTags.Meter1_ApparentPower", "props": { "label": "Apparent power", "min": 0, "max": 50,
"units": "kVA", "decimals": 2 } },
    { "id": "w21", "type": "switch", "x": 8, "y": 410, "w": 384, "h": 58, "tag":
"Energy.Meter1.RelayOutput", "props": { "label": "Relay", "onValue": 1, "offValue": 0, "onText":
"ON", "offText": "OFF", "onColor": "#22c55e" } },
    { "id": "w22", "type": "rect", "x": 8, "y": 480, "w": 188, "h": 48, "props": { "fill":
"#1d4ed8", "stroke": "#1d4ed8", "radius": 10 }, "onClick": "nav", "nav": "ea100-phone-settings" },
    { "id": "w23", "type": "label", "x": 8, "y": 480, "w": 188, "h": 48, "props": { "text":
"Settings", "fontSize": 14, "color": "#ffffff" }, "onClick": "nav", "nav": "ea100-phone-settings"
}
  ]
},
{
  {
    "id": "ea100-phone-settings", "name": "Meter1 – Settings", "target": "phone", "parentId":
"ea100-phone", "order": 1,
    "width": 400, "height": 700, "background": "",
    "widgets": [
      { "id": "w24", "type": "label", "x": 12, "y": 12, "w": 376, "h": 30, "props": { "text":
"CURRENT TRANSFORMER", "fontSize": 13, "color": "#3b82f6", "align": "left" } },
      { "id": "w25", "type": "sfield", "x": 8, "y": 48, "w": 384, "h": 44, "tag":
"Energy.Meter1.CtRatio", "props": { "label": "CT ratio", "decimals": 0, "editable": true, "step":
5 } }
    ]
  }
},
{
  "notes": "# Example EA-100\n\nReads voltage, current, active power and total energy; the relay
output and the CT ratio are writable. The energy counter has 0.1 kWh resolution (raw value / 10)."
}
}

```

If you install this package with the device name `Meter2`, tag paths become `Energy.Meter2. ...`, the calculated tag `Meter2_ApparentPower` and the screen names `Meter2 – ...`.

Working with an AI assistant

The package format is designed to be read and written by AI assistants. You can give a device's register map, manual or datasheet to an assistant and have it prepare the package.

[Download the MergenHub device template skill](#) (zip). It contains:

- `SKILL.md` — the workflow and hard rules the assistant follows.
- `reference/format.md` — a compact reference of every field.
- `reference/drivers.md` — driver names, device ID formats and address syntax.
- `reference/expressions.md` — expression functions.
- `examples/minimal-modbus.mhpkg.json` — a complete example that passes the check.

Upload the zip as a skill in the Claude app, or unpack it into `~/ .claude/skills/` for Claude Code; with an assistant that does not support skills, attach the files to the conversation. Then ask, for example: "Build a MergenHub template for this energy meter from the attached register table; ask for IP, port and unit ID at installation; add an alarm at 250 V on the voltage."

Open the assistant's file in MergenHub Mobile first with *Cihaz ekle › Dosya, pano, QR ya da adres › Panodan yapıştır* (Paste from clipboard) and look at the check result; then install it on the server.

Getting a device into the official library

To have a device added to the Device Library, send a request from the customer panel on the website: *Panel › Device requests › New request*. The **Ekleme isteği gönderin** link under the library list in MergenHub Mobile and the library dialog in the web interface open the same page.

The form asks for brand, model, firmware version, category, communication protocol, product page, what the device does and which data you want to monitor, and connection details. You can attach up to five files (10 MB each): photo, register map, manual — and a `.mhpkg.json` package you prepared.

Requests are **never published automatically**. The Envest team verifies the device and documents, may ask you for details, prepares and tests the package (tags, screens) and adds it to the library itself. The request's status (received, in review, approved — in preparation, published or not accepted) and the team's note are shown in the panel; the decision is also sent by e-mail. A company can have at most ten pending requests at a time.