USER MANUAL (IN-APP, COMPLETE)

Protocols (Drivers)

Online: mergenhub.envest.com.tr/docs/manual-protocols

ON THIS PAGE

Addressing — General
Modbus (TCP / RTU / ASCII)
Siemens S7
OMRON (FINS / NJ CIP)
Allen-Bradley Logix (EtherNet/IP)
Fatek FBs
KNX (KNXnet/IP)
BACnet/IP
Bridging KNX and BACnet
DNP3
IEC 60870-5-104
IEC 61850 (MMS)
MQTT Client
Simulator

Addressing — General

Each driver has its own address format. A tag with an invalid address is rejected (same rules as the reference server). JSON exports from real OPC servers can be imported; driver names and address formats are preserved exactly.

Modbus (TCP / RTU / ASCII)

Drivers: **Modbus TCP/IP Ethernet**, **Modbus RTU Serial**, **Modbus ASCII Serial**.

0xxxx	Coil (read/write bit) — F01/F05/F15
1xxxx	Discrete Input (read-only bit) — F02
3xxxx	Input Register (read-only word) — F04
4xxxx	Holding Register (read/write word) — F03/F06/F16

Address = area prefix + offset (5–7 digits; with 6-digit addressing, 7 digits such as `4000011` are accepted). Bit within a word (`40001.3`) and string (`40001.10H` = 10 characters, H=HiLo/L=LoHi) are supported. Register bit writes use **FC22 Mask-Write** (atomic). Byte order (First Word/DWord Low, Modicon bit order, Treat Longs as Decimals), block size (64/125 registers), IP Protocol (TCP/UDP), Close Socket on Timeout and deactivating the tag on Illegal Address can be configured. A bare IP device ID and `<IP>` without a unit are supported.

> **Tip** Block size resets some devices: using **64** instead of a 125-register block (the common industry default) improves compatibility. If a device reports "Bad", reduce the block size to 64.

> **Limit** Modbus TCP 1024 channels · 8192 devices per channel; RTU 1024 · 255; ASCII 100 · 247. Details: Channel and Device Limits.

See also Timing Scaling Common Problems

Siemens S7

Driver: **Siemens S7 TCP/IP** (S7-300/400/1200/1500, ISO-on-TCP / RFC1006). The device ID is the IP + **Rack/Slot** (typically Rack 0, Slot 1 for 1200/1500).

`DBn,type offset`	Data block. E.g. `DB1,DBW20` (word), `DB1,DBD8` (dword/real), `DB1,DBX0.3` (bit)
`M / MB / MW / MD`	Merker (bit/byte/word/dword). E.g. `M10.3`, `MW14`
`I / E`, `Q / A`	Input / output. E.g. `I0.0`, `Q2.1`
`T / C`	Timer / Counter

```
{ "common.ALLTYPES_NAME":"Sicaklik", "servermain.TAG_ADDRESS":"DB1,DBD8",
  "servermain.TAG_DATA_TYPE":"Float" }
```

> **Limit** 1024 channels · 1024 devices per channel. Details: Channel and Device Limits.

OMRON (FINS / NJ CIP)

Two drivers:

OMRON FINS Ethernet	CS/CJ/CP series. Areas: `CIO, W, H, D, A`. E.g. `D100`, `CIO0.05` (bit), `W20`. FINS SA1/DA1 and FirstWordLow settings are on the channel/device.
OMRON NJ Ethernet	NJ/NX series. **EtherNet/IP CIP symbolic** — access by the **symbol name** in the PLC (UTF-8, Turkish names supported). E.g. address = `Pompa1.Debi`.

> **Note** In FINS the word/bit order may vary by device; if a value comes in reversed or shifted, change the **FirstWordLow** setting. Verified live with real NJ/CJ PLCs.

> **Limit** FINS Ethernet 512 channels · 32 devices per channel; NJ Ethernet 256 · 1024. Details: Channel and Device Limits.

Allen-Bradley Logix (EtherNet/IP)

Driver: **Allen-Bradley ControlLogix Ethernet**. Connects to ControlLogix, CompactLogix, GuardLogix and Micro800 PLCs over EtherNet/IP (CIP) **by tag name**. No extra software such as RSLinx is needed.

SETUP: STEP BY STEP

1	New channel → driver **Allen-Bradley ControlLogix Ethernet**.
2	New device → pick the **Model**. **ID**: `IP,1,slot` , e.g. `192.168.1.10,1,0` (backplane 1, CPU slot 0). On CompactLogix the slot is usually 0. On Micro800 write the IP only.
3	List the controller-scoped tags with **Create Tags from Device**, or add tags by hand.

ADDRESSING

`Temperature`	Controller-scoped tag.
`Program:MainProgram.Counter`	Program-scoped tag.
`Motor.Speed` · `Tank[3]`	Structure (UDT) member · array element.
`Status.5`	Bit inside a DINT (write: read-modify-write).
`RecipeName`	STRING (up to 82 characters), data type String.

Types: BOOL → Boolean, SINT/INT/DINT/LINT → Char/Short/Long/LLong, REAL/LREAL → Float/Double.

> **Note** The driver first tries a large CIP connection (1996 bytes), then a small one, then unconnected messaging; it does this on its own. If older CPUs are slow, set **CIP Connection Size** to 504.

> **Limits** A whole structure (UDT) is not read as one tag; make its members separate tags. Use DINT bits instead of BOOL arrays.

Fatek FBs

Driver: **Fatek FBs**. Connects to Fatek FBs, FBe, B1 and HB1 PLCs with Fatek's own protocol: Ethernet (FBs-CM25E, CBE or the built-in port) or a serial port (RS-232/485).

SETUP: STEP BY STEP

1	New channel → driver **Fatek FBs**. **Physical Medium**: Ethernet or COM Port. On a serial line the Fatek default is **9600, 7 data bits, Even, 1 stop**.
2	New device → **ID** = PLC station number (default `1`). On Ethernet enter the **PLC IP Address** and **Port** (default **500**); protocol TCP or UDP.
3	Add tags (address examples below) and check the values.

ADDRESSING

`X0` · `Y4` · `M10` · `S2` · `T0` · `C3`	Bit areas (Boolean). Numbers are decimal.
`R100` · `D20` · `F0`	16-bit register. Float / Long / DWord read two consecutive registers (low word first).
`WM16` · `WX0` · `RT0` · `RC5`	Bit group word · timer and counter value.
`DR10` · `DD0` · `DRC200`	Native 32-bit access (32-bit counters are RC200 and up).

`R100.3`	Bit inside a register (write: read-modify-write).

> **Note** Consecutive addresses are read in one request (up to 128 bits or 64 registers). If the PLC rejects an address (illegal address), only that block's tags go Bad; the connection stays up.

KNX (KNXnet/IP)

Driver: **KNXnet/IP**. Connects to a KNX building automation line through a **KNX IP interface** (tunneling) or a **KNX IP router** (routing, multicast). No ETS or other software is needed.

SETUP: STEP BY STEP

1. New channel → driver **KNXnet/IP**. **Connection Type**: **Tunneling (UDP)** (every interface, default), **Tunneling (TCP)** (KNXnet/IP 2 interfaces) or **Routing (multicast)**.
2. **Interface / Router IP**: the interface's IP address (`224.0.23.12` for routing). **Port** defaults to **3671**. With several network cards pick the one on the KNX network in **Network Adapter**.
3. New device → a name only (floor, panel, room...). **No device ID is needed.** Devices on one channel share one tunnel.
4. Add tags by hand (`1/2/3;DPT=9.001`) or with **Create Tags from Device** — from the ETS export (CSV / XML) set in the device's **ETS Group Address Export File**, otherwise by listening to the bus.

> **ETS file** The file must be **a local file on the server**: a full path (e.g. `C:\KNX\group-addresses.csv`), extension `.csv` , `.xml` or `.txt` , at most **20 MB**. Network paths (`\\server\share\…`) and relative paths are not accepted; copy the file to the server first.

ADDRESSING: GROUP ADDRESS + DPT

`1/2/3;DPT=9.001`	3-level group address (main 0-31 / middle 0-7 / sub 0-255) + datapoint type.
`1/234;DPT=5.001`	2-level group address (main 0-31 / sub 0-2047).
`0/0/1`	Without a DPT it follows the data type: Boolean → 1.001, Byte → 5.010, Word → 7.001, Short → 8.001, Long → 13.001, DWord → 12.001, Float → 9.001, String → 16.000.
`DPST-9-1`	ETS notation is accepted too (`1/0/1;DPT=DPST-9-1`).

SUPPORTED DPTS

1.x	Switch, binary state (Boolean)
5.001 · 5.003 · 5.010	Percent 0-100 · angle 0-360 · 0-255
6.x · 7.x · 8.x	1-byte signed · 2-byte unsigned · 2-byte signed (8.010 percent)
9.x	2-byte float: temperature, lux, speed, pressure...
12.x · 13.x · 14.x	4-byte unsigned · signed (energy Wh) · IEEE float (power, current...)

16.000 · 16.001	14-character text (ASCII · Latin-1)
10.001	Time (String): `08:30:00` ; if the device also sends the day of the week, `Pzt 08:30:00` (Pzt Sal Çar Per Cum Cmt Paz = Mon … Sun: the app writes Turkish day abbreviations; this is the value format and is not translated)
11.001	Date (String): `2026-11-18` — years 1990-2089 only
19.001	Date-time (String): `2026-11-18T08:30:00` — years 1900-2155
232.600	RGB colour (DWord): a 0xRRGGBB number (red = 16711680); `#FF0000` also works when writing
2.x · 3.x · 17.x · 18.x · 20.x · 29.x	Control bits, scene, 1-byte enum (HVAC mode), 8-byte counter — raw value

TIME AND DATE VALUES

Tags of 10.001, 11.001 and 19.001 must have the data type **String**; they are not saved with a numeric data type. The value read arrives in the format above; use the same format when writing. When writing, these are accepted too: time without seconds (`08:30`), the full Turkish day name or the English one (`Pazartesi 08:30` , `Mon 08:30`), a day number (1 = Monday … 7 = Sunday), a date as `18.11.2026` , a space between date and time. An invalid value (e.g. `25:00` , `2026-02-30`) is not sent and returns an error.

19.001 date only	`2026-11-18` : arrives from the device with the "time invalid" flag and is written the same way.
19.001 time only	`08:30:00` : with the "date invalid" flag.
19.001 without year	`--11-18T08:30:00` : with the "year invalid" flag.
19.001 fault	If the clock sends the "fault" (F) flag, the tag goes **Bad**.

> **Writing 19.001** The day of the week is calculated from the date. Working day is sent as "unknown", the summer time flag off (winter / standard time) and the clock source as "no external sync". When reading, these flags do not change the value.

CREATING TAGS BY LISTENING TO THE BUS: ESTIMATED TYPES

Without an ETS file the type is estimated from the telegram length. 3- and 4-byte data can fit more than one type; the description of these tags gets **Tahmini DPT — doğrulayın** ("estimated DPT — verify"; the driver always writes this text in Turkish). 4 bytes: 14.x (power, current…) if the value is plausible as a float, otherwise 13.x (counter). 3 bytes: 10.001 if the value is valid as a time, 11.001 if it is valid as a date, otherwise 232.600 (RGB). Compare the type of these tags with the type in ETS; if needed, correct the DPT in the address and the data type.

READING AND WRITING

The driver listens to every group telegram on the bus; the last value of each group address (GroupValue_Write / Response) is cached and tags read from it. For an address not seen yet a **GroupValue_Read** is sent once; the tag stays **Uncertain** until the answer. Writing uses **GroupValue_Write**; an out-of-range value (e.g. 150 to 5.001) is not sent and returns an error.

> **Note** The object must have the **R (read) flag** set in ETS; otherwise the first value arrives only when the device sends it. Cyclic reading (device setting) loads the bus; KNX devices already send on change.

> **Connection** If the interface does not answer the liveness query (every 60 s by default) 3 times or closes the connection, tags go **Bad** and it reconnects. Interfaces allow few tunnels (mostly 1-8): if ETS uses the same interface there may be no room left (`E_NO_MORE_CONNECTIONS`).

BACnet/IP

Driver: **BACnet/IP**. Connects to building automation controllers (HVAC, air handling units, lighting, energy meters) over BACnet/IP on UDP port **47808**. Reads object properties (ReadProperty / ReadPropertyMultiple) and writes them (WriteProperty).

SETUP: STEP BY STEP

1	New channel → driver **BACnet/IP**. In **Network Adapter** choose the interface facing the BACnet network. If the devices are on another IP subnet, enter the BBMD of that network in **BBMD Address** (foreign device registration).
2	New device → on the first step press **Search network**: devices answering Who-Is are listed. Pick one; the **ID** (Device Instance, e.g. `1001`) and the **IP Address** are filled in. If it is not listed, type the ID and IP by hand; with an empty IP the device is searched with Who-Is when connecting.
3	Right-click the device → **Create Tags from Device**: the device Object_List is read and every object carrying a value becomes a tag named after its Object_Name. Or add tags by hand.
4	Check the value; write to writable objects (AO, AV, BO, BV, MSO, MSV). The default write priority is **16**.

ADDRESSING

`AI:1` · `AV:3` · `BO:2` · `MSV:5`	Object type:instance; without a property, Present_Value.
`AV:3.PresentValue` · `AI:1.StatusFlags` · `AV:3.85`	Property name or number.
`AO:1@8`	This tag writes with priority 8 (1..16).
`AO:1.PriorityArray[8]`	Array element.
`DEV:1001.ObjectName`	Property of the device object.

Types: AI/AO/AV → Float, BI/BO/BV → Boolean, MSI/MSO/MSV → Word (1..n), IV → Long, PIV/ACC → DWord, LAV → Double, CSV and name / description → String.

QUALITY

Good	Value read, Status_Flags clear.
Uncertain	The object's Status_Flags has **Fault** or **Out_Of_Service** (the value arrives but is not reliable).
Bad	The device does not answer, or the object / property does not exist on the device (then the tag is taken out of the scan).

CHANGE OF VALUE (COV) SUBSCRIPTION

By default the device is asked on every scan (polling). If **Use COV Subscription** in the **Change of Value (COV)** group of the device properties is turned on, the driver subscribes with **SubscribeCOV** to the objects whose Present_Value or Status_Flags is read (AI, AO, AV, BI, BO, BV, MSI, MSO, MSV, IV, PIV, LAV ...). The device sends a notification whenever the value changes; the tag is updated from it, and the scan sends no read to the device for that object. On devices with many points this clearly reduces network load.

Subscription Lifetime (s)	Default **300** s (60..3600). Renewed at half of the lifetime. Re-subscribed when the connection drops and comes back; subscriptions are cancelled when the channel / device closes.
Confirmed COV Notifications	When on, the device waits for an acknowledgement of every notification; when off (default), unconfirmed notifications are used.
Device / object without support	If the device does not know SubscribeCOV or refuses the subscription of an object, those objects keep being read by polling; nothing else needs to be done.

A notification is accepted only when it comes from the address of the subscribed device with the process identifier MergenHub gave for that subscription; a forged notification from another address cannot change the tag. While values come from notifications, the device is asked once if it has not been heard for more than 15 s: if the device is down, the tags go **Bad**.

LARGE RESPONSES (SEGMENTATION)

The device may send responses that do not fit in one packet (the Object_List of a device with many objects, a large ReadPropertyMultiple) **segmented**; the driver collects the segments in order and joins them. A lost or out-of-order segment is requested again. To protect memory a response can be at most **4 MB** / 4096 segments; if this is exceeded, or the device does not support segmentation, the driver makes the request smaller (reads Object_List index by index). No setting is needed.

> **Note** Nearby instances of the same type are read in one ReadPropertyMultiple request (20 properties per request by default). If the device does not know ReadPropertyMultiple, the driver switches to single reads by itself. Writing `null` to a String tag relinquishes that priority.

> **Port** If another BACnet program on this machine holds 47808, the driver moves to a random port: reads and writes work, but broadcast I-Am replies may not arrive; then type the device IP by hand.

Bridging KNX and BACnet

With the KNX and BACnet drivers in one project MergenHub becomes a **translator** between the two networks: it writes a KNX value to BACnet and a BACnet value to KNX. MergenHub connects to both networks as a **client**; the bridge is built in Advanced Tags with the **Link (write-back)** type.

SETUP: STEP BY STEP

1	Add the KNX channel, device and tag. Example: channel `KNX`, device `Kat1`, tag `Sicaklik` = `1/0/1;DPT=9.001` (Float).
2	Add the BACnet channel, device and tag. Example: channel `BAC`, device `AHU`, tag `OdaAyar` = `AV:1` (Float).
3	**Advanced Tags** → new tag, type **Link**. **Input** = `KNX.Kat1.Sicaklik`, **Output** = `BAC.AHU.OdaAyar`, **Link Mode** = **onChange**. Save.
4	For the other direction add a second Link: Input a BACnet tag (e.g. `BAC.AHU.BmsAyar` = `AV:2`), Output a KNX tag (e.g. `KNX.Kat1.AyarNoktasi` = `1/1/1;DPT=9.001`).
5	Change the value on the wall thermostat or write it from the BMS; watch the tag on the other side change too. The Link tag's own value equals its input.

CONDITIONAL BRIDGE: AUTOMATION

If the value should be transferred only under a condition, build an Automation flow. Example: trigger **KNX presence sensor became 0** → condition **after 19:00** → action **write 18 to BACnet AV:1**. If a calculation or unit conversion is needed, first create a computed (Derived) tag and use it as the Link's input.

TYPE MAPPING

You decide which KNX point goes to which BACnet object; MergenHub does not map them by itself. Common pairs:

`DPT 1.001` (switch)	BV / BO (Boolean)
`DPT 5.001` (percent)	AV / AO (Float, 0-100)
`DPT 9.001` (temperature) · `9.x`	AI / AV / AO (Float)
`DPT 13.x` · `14.x` (energy, power)	AI / AV (Float) or ACC
`DPT 20.102` (HVAC mode)	MSV: the values start differently (KNX at 0, MSV at 1); do the translation with a computed tag.

Limits

• MergenHub does **not appear** on the network as a BACnet or KNX device itself: a BMS cannot find MergenHub with Who-Is, and it is not listed as a device in ETS. That would need a separate BACnet server mode.

• **Latency:** the KNX side listens to the bus and sees a change at once. The BACnet side is read at the tag's scan rate. The bridge may lag from under a second up to the scan time; do not use it for life or machine safety loops.

• **BACnet priority:** MergenHub writes at priority 16 by default. If the BMS writes the same object at a higher priority (lower number), MergenHub's value does not take effect. Change the priority in the address if needed (e.g. `AV:1@8`).

• **Loop:** do not link the same two points to each other with two Links. The value bounces back and forth and, because DPT 9 rounds, may never settle. Use separate points for set point and feedback.

Tip On a real site try a single point first; check the value on the other side and which priority wins in BACnet (`AV:1.PriorityArray`), then add the other points.

DNP3

Driver: **DNP Client Ethernet**. Address format `grup.varyasyon.index` (`g.v.i`).

`1.x.i` · `BIi`	Binary Input (i = index)
`10.x.i` · `BOi`	Binary Output (writable, control)
`30.x.i` · `AIi`	Analog Input
`40.x.i` · `AOi`	Analog Output (writable)
`DS0.1` · `DA250`	Data Set member · Device Attribute (g0)

The short mnemonic (`AI0`) and the DNP3 `g.v.i` formats are accepted together. The device ID is the **IP** only; the master/outstation address + port are device/channel properties.

Wired industry-parity features: Operate Mode (Direct / SBO / No-Ack, with operate STATUS verification — no false Good), class polling (Class 0/1/2/3) + integrity, unsolicited (mode 1/2/3), time synchronization (UTC/DST/Honor/LAN·Serial), Event Playback (event buffer + playback), Data Sets (g85–88), Device Attributes (g0), File Control (g70), Secure Authentication (IEC 62351-5 / SAv5) and path redundancy (Redundancy). CRC/frame validation is built in.

> **Limit** 1024 channels · 1024 devices per channel. Details: Channel and Device Limits.

IEC 60870-5-104

BETA This driver has not yet been through full field testing; use it with care in production.

Driver: **IEC 60870-5-104 Client**. Address format `TYPE.IOA.attr` — TypeID + Information Object Address.

`M_SP.IOA`	Single-point information (telesignal)
`M_ME.IOA`	Measured value (telemetry)
`C_SC.IOA`	Single command (telecontrol, write)

Device ID: `IP:2404,commonAddress` . The ASDU/APCI layer, k/w window and t1-t3 timers are built in. All points are read at startup and periodically with a **General Interrogation (GI)**. **Command/set-point writes** (C_SC/C_DC/C_SE) run the **ACT → ACT-CON** confirmation handshake: if the device returns a negative confirmation (COT P/N), the tag becomes Bad and no false Good is written.

> **Limit** 1024 channels · 1024 stations per channel (stations on the same IP:port share a single socket; common addresses must differ). Details: Channel and Device Limits.

IEC 61850 (MMS)

Driver: **IEC 61850 MMS Client**. Full ISO stack (MMS/ACSE/Presentation/Session/COTP/TPKT). Address = object reference path:

```
LD/LN.DO.DA   →  ör. RELAY/MMXU1.TotW.mag.f  (aktif güç)
                   RELAY/XCBR1.Pos.stVal      (kesici durumu)
```

Logical device (LD) → logical node (LN) → data object (DO) → data attribute (DA). Device ID: `IP:102` . Reading is supported at the LN/DO/DA levels. **Control writes** run the **select→operate** handshake with the full control model (orCat / orIdent / ctlNum / Test / Check) and verify the IED's **AddCause** response — if rejected, the tag returns Bad (no false Good is written). SCL/ICD import and AP-Title/TSEL selector authentication are supported.

> **Limit** 256 channels · 256 devices (IEDs) per channel. Details: Channel and Device Limits.

MQTT Client

Driver: **MQTT Client** (3.1.1 / 5.0). It subscribes to an MQTT broker and reads incoming topics/payloads as tags ("use MQTT like Modbus"). **New model (104 parity):** the broker connection (host/port, TLS 1883/8883, CA + client certificate, QoS, user/password, Client ID) is at **CHANNEL** level; devices on the same channel use a single shared broker socket. A **device** is only a set of topics — the Device ID is optional (empty or a label). The device also carries liveness (**Arrival Watchdog**: expected period × 1.5 → Uncertain, × 3 → Bad; reconnect grace) and tag discovery settings.

Address = **topic + payload parsing**. In the tag property editor a **wizard** appears instead of the raw address (Read + separate Write fields); it builds the address string in the background:

Bare value	`values/debi` — the payload is directly a number/text (45 / 45,7 / ON)
JSON field	`telem:/flow/max` — descend into sub-fields with `/` ; array: `/data/0/x`
Key-value	`raw;key=debimetre` — extracts from the text "debimetre: 45"
Delimited (CSV)	`csv;idx=2;sep=;` — "10;20;30" → 30 · sep= a SINGLE character: `:` `#` `tab` `space`
XML / Regex / Binary	`xml=data/flow` · `re=debi=([0-9.]+)` · `bin=f32le@0`
Wildcard	`sens/+/temp` (single level), `#` (multi level)
Write	`;w=cmd/sp` (different topic), `;wp=/set` (embed in JSON), `;wr=1` (retain), `;wbool=1` (true/false)

Smart discovery: "Generate Tags from Device" breaks down the incoming payload (JSON/XML/CSV/key-value) and suggests tags with JSON pointers; with deep/huge payloads it can produce 500+ leaf tags.

> **Note** MQTT also exists **northbound** as an agent in the Data Bridge — there it *publishes* tags to the broker; here it *reads* from the broker.

> **Limit** 60 channels (broker connection) · 1024 devices per channel. Details: Channel and Device Limits.

Simulator

Driver: **Simulator**. Synthetic data without hardware — for testing, demos and SCADA/alarm verification. It uses no network/socket; the device ID is a plain number (1–999), not an IP.

`SINE:10:0:100`	Sine — period 10 s, between 0 and 100
`RAMP:5:0:100`	Sawtooth/ramp (0..100)
`SQUARE:2:0:1`	Square wave (Boolean)
`RANDOM:0:100`	Random 0..100
`COUNTER`	Continuously increasing counter

Address format **FUNCTION:period:min:max**. Compatible with Simulator import (R/K/B/S registers).

Limit 1024 channels · 999 devices per channel. Details: Channel and Device Limits.