

MODULES

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.

Online: mergenhub.envest.com.tr/docs/automation

ON THIS PAGE

[Automation \(workflows\)](#)[Triggers](#)[Logic](#)[Actions](#)[Permissions and safety](#)[Editor](#)[Building with the assistant](#)[API](#)[Formula editor and live monitoring](#)

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 (<code>min hour day month weekday</code>)	<code>0 8 * * 1-5</code> 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 (<code>we11*</code>), minimum priority
Webhook	An external system calls <code>POST /hooks/{key}</code>	key (generated), secret header <code>X-Hook-Secret</code> ; the body becomes <code>\$payload</code>

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.