Amtiri

REFERENCE MANUAL

# Scheduler

Generated on November 25, 2026

---

A formula is recalculated when a variable it reads changes. Some formulas read nothing that changes: the time, with `now()`, or a logger's history, whose variable is named as text. A SCHEDULER recalculates such formulas on a clock.

| Module | What it does | Typical use |
|---|---|---|
| **Scheduler** ( `SCHEDULER` ) | Makes formula variables recalculate on their schedules | Averages and totals over history, clock variables, values that change with the time of day |

The module also has the settings every module shares: whether it is enabled, its name, its status and message variables and their thresholds. They are described in Settings every module shares.

---

## Scheduler

### Settings

| Key | Setting | Type | Default | Meaning |
|---|---|---|---|---|
| `definitions` | Definitions | list of rows | empty | The variables to recalculate, each with its schedule |

### Definitions

| Key | Setting | Type | Default | Meaning |
|---|---|---|---|---|
| `field` | Variable | variable code | empty | The variable to recalculate. It must have a formula |
| `cron` | CRON | Quartz cron | `0 * * * * ?` | When to recalculate it. A new row defaults to every minute ( `0 * * * * ?` ); the syntax is in Settings every module shares |

One module can hold any number of definitions, and one variable can appear in several, each with its own schedule.

### What a tick does

- **It asks for a recalculation, and nothing else.** A tick has the same effect as a change of a variable the formula reads: the formula runs, and its result is written into the variable. The Scheduler hands the variable no value of its own.
- **Only a formula is recalculated.** A tick on a variable without a formula does nothing: no value, no error, no message.
- **The result is written like any formula's.** It is converted to the variable's declared type and saved when the variable is persistent. A result equal to the value the variable holds changes nothing, and nothing that reads the variable runs; a different one runs everything that does, as any change would.
- **Formulas run one at a time.** The recalculation joins the engine's queue of formulas to recalculate, behind any already waiting. A tick that falls while the variable is still waiting adds nothing.
- **A formula that fails** keeps the variable's value and writes a warning to the log. The Scheduler's own status does not show it: the failure belongs to the variable.

- **Until the first tick** a variable whose formula reads nothing that changes holds its initial value, or no value at all when it has none. A persistent variable holds its saved value instead, and its initial value only when it has nothing saved, as at its first start: see Initial values.
- **Renaming.** When an import renames variables, the definitions follow the rename.
- **Any case.** A definition finds its variable however the code is written: `lit_301_avg` recalculates `LIT_301_AVG`, since the engine keys every variable by its code in capital letters.

### Schedules

Every definition's CRON is checked before any of them is scheduled: one that is empty or cannot be read keeps the whole module from opening, so no definition runs, and its status reads `Invalid schedule '{cron}': {reason}`, or `No schedule given` for an empty one. The module stays in error until the CRON is corrected and the engine started again. Schedules follow the plant's time zone, `application.timezone`, as formulas do (before engine 6.18.9, the Java process's zone), and a time that passed while the engine was stopped is not made up later: see Schedules.

### Messages

A Scheduler that opened is on, and says nothing else. The complete list is in Module messages.

| When | Level | Text |
|---|---|---|
| Opening: a definition has no CRON. The module does not open | `ERROR` | `No schedule given` |
| Opening: a CRON cannot be read. The module does not open | `ERROR` | `Invalid schedule '{cron}': {reason}` |

### Example

Two formula variables recalculated on a clock:

| Variable | Formula | CRON |
|---|---|---|
| `LIT_301_AVG_24H` | `tsAverage(history(PLANT_LOG, "LIT_301", addHours(now(), -24), now()))` | `0 0/15 * * * ?` |
| `CLOCK_UTC` | `now()` | `* * * * * ?` |

`LIT_301_AVG_24H` holds the level's average over the last 24 hours from the LOGGER `PLANT_LOG`, brought up to date every quarter of an hour. `CLOCK_UTC` changes every second, and every dashboard that shows it redraws with it. A third definition for `P101_RUN`, which a Modbus master writes and which has no formula, would do nothing at all.

## Next steps

- Logger
- Settings every module shares
- Module messages
- Module catalogue

This page describes Data Orchester engine 6.18.9.