Amtiri

USER MANUAL

Managing History

Generated on November 25, 2026

A LOGGER keeps every change of its variables in a file of its own, and nothing trims that file on its own. This guide covers what you can do with a logger's history while the plant runs: delete it, export it, import or merge another history into it, compact it, and put an old log back.

Everything is in the logger's settings. Open the LOGGER module on the diagram and choose its **History** tab, beside the tab of its variables, status and messages. It needs the engine running and the logger open, and the Configuration role.

Operation	Use it to
Delete…	Remove a variable's history, a range of it, or the whole logger's (**Delete all…**)
Export…	Write chosen variables over a chosen range to a file, to download or take to another instance
Import / merge…	Bring in a history file or another logger's history, after checking where they overlap
Compact…	Make an old interval small, keeping each segment's min, max and average
Restore…	Put an old log back over the current one

> **Important**
>
> **These actions apply at once.** They run on the logger's file, outside the editor: **Save** does not apply them, **Undo** does not revert them, and closing the settings does not cancel them. Each dialog shows a preview of exactly what it will do before you start it, and every operation that removes rows keeps them in a file first.

What the section shows

- **A summary**: the file's size, its rows, its oldest and newest times, the logger's sessions, the compacted periods and the free disk. Times are shown in the plant's time zone and stored in UTC.
- **The variables** in the file: whether the logger still logs each one or it is an orphan, its rows, its first and last times, and its compacted periods. **Analyse** adds each variable's duplicate rows and size, which take a read of every row.
- **Files**: the full backups, removed points and exports the logger's operations wrote, with **Restore…**, **Download** and **Delete file**.
- **Jobs**: the one running, with its progress and **Cancel**, and the last 20 with their outcome.

One job runs at a time per logger. A job writes in short batches between the logger's own rows, so logging never stops; a cancelled job keeps what it finished, and the same job started again finishes the rest.

Deleting history

1. Select a variable and choose **Delete…**, or choose **Delete all…** for every variable.
2. Set **From** and **To** to delete a range, or leave them empty for all of each variable's history.

3. Keep **Keep the removed points in a file** ticked: the rows go to a file under Files first, and restoring that file undoes the delete.
4. **Preview** shows the rows to delete. Type the logger's code and **Start**.

- **A compacted segment is kept whole.** A border that falls inside one moves to its edge, and the preview says so.
- **Rows not yet replicated** to a Logger Replicator's receiver are lost there too; the preview counts them.
- **Deleting the newest rows** of a logger that receives replication makes its senders send them again.

Reclaim space gives the freed room back to the disk. It needs free disk of about twice the file, holds up writing while it runs, and replicated batches arriving meanwhile are answered busy and resent.

Exporting

Export… writes chosen variables over a chosen range, or all of their history, to a history file in the logger's `exports` folder. The file carries what it needs to be put back exactly: the value in force at the start of each variable's range, the exact range it speaks for, the logger's sessions over it and its compacted periods. Download it from Files; a file larger than the Site's transfer limit stays on the volume, and the message says where.

Importing and merging

Import / merge… brings history in from a file you upload, a file copied into the logger's `inbox` folder, or another logger of this instance. The file is checked first: it must be a sound SQLite file holding only a logger's own tables, with real timestamps; anything else is refused with the reason.

1. Choose the source, and a **prefix** if the incoming series measured something else than the logger's own.
2. **Check the overlap.** Per variable, the report counts the incoming rows: identical to the log's, conflicting (same time, another value), interleaved (inside the overlap, at times the log does not have) and new.
3. Choose an action per variable:

Action	Writes
Only rows outside the overlap (default)	The rows outside the times the logger was running
Merge	Every row that is not identical; a conflict keeps the logger's row, or the file's
Replace the overlap	The file's rows over the overlap, as a restore would, then the rows outside it
Skip	Nothing

A compacted period in the overlap cannot be merged: interleaved rows would break its average. Many interleaved rows usually mean the two sides measured different things under one code: use a prefix instead.

Compacting

Compaction makes an old interval small. Each segment of it is replaced by its time-weighted average, with a spike to its max and one to its min where they happened: an hour of one-second samples becomes at most five rows, and its min, max and average stay as they were. Nothing compacts on its own: you choose the interval, the variables and the size of the segments.

1. Select a variable and choose **Compact…**.
2. Set **From** and **To**. Segments start at **From**, so choosing it chooses their alignment: start at midnight for daily segments that are calendar days.
3. Choose the **Segment size**: 1 min, 5 min, 15 min, 1 h, 6 h or 1 day; or **Duplicates only**, which removes samples equal to the one before and changes nothing else.

4. Keep **Export the data to be compacted first** ticked. Compaction cannot be undone any other way; restoring that export gives the raw rows back.
5. **Preview**, type the logger's code, **Start**.

- **The newest rows stay raw.** The segment holding a variable's newest row is never compacted.
- **Replication comes first.** Compaction stops before the rows a Logger Replicator's receiver has not acknowledged, and leaves out a variable a replicator reads but has never delivered.
- **Larger segments only.** A compacted interval can be compacted again with larger segments; with the same or smaller ones, its segments are skipped.
- **Text is never averaged**: only Duplicates only applies to it. A boolean keeps the time it spent on and off.
- **Reads keep working.** `history()`, `tsMin`, `tsMax` and `tsAverage` read a compacted period like any other; on windows aligned to the segments, min and max are exact and the average is within a few ten-millionths of the range.

Restoring

Restore… puts an old log back over the current history: a full backup, removed points, an export, a file in the inbox. Choose it under Files, or in the dialog.

1. Leave the variables empty to restore every variable the file holds, or name them.
2. Leave the dates empty to restore the whole interval the file speaks for, or narrow it.
3. Keep **Keep the removed points in a file** ticked, so the restore can itself be undone.
4. **Preview**, type the logger's code, **Start**.

The restore removes the log's rows over the interval and writes the file's, taking care of the borders: the value in force after the interval stays what it was, and a border inside a compacted segment moves to the segment's edge, outward when the file covers the segment and inward otherwise. The preview shows every border it moved.

Example: heating-control

Four things that once took a hand-compiled tool, or could not be done at all:

Task	In the History section
A sensor fault wrote nonsense into `TEMP` for an afternoon	Select `TEMP`, **Delete…** with **From** and **To** around the fault, keep the removed points
Take a month of history to the office	Select `TEMP` and `HEAT`, **Export…** over the month, then **Download** it from Files
A year of one-second samples fills the disk	Select `TEMP`, **Compact…** the year with one-hour segments and Export first; **Reclaim space** afterwards
The wrong interval was compacted	Under Files, select the export the compaction took first and **Restore…** it

Next steps

- Logger
- Logger Replicator
- Moving configuration between installations