The Purge Job: one schedule for all cleanup

An instance that renders all day accumulates. History records with their PDFs, print tasks that have long been printed, mail tasks whose mails arrived weeks ago, and files on the history disk that no record points at any more. Somebody has to take that out, on a clock, without anybody thinking about it.

The **Purge Job** is that somebody. One schedule, one run, five entities, five retentions.

Screenshot placeholder: `scheduler.purge-form.png`
The purge settings with one retention field per entity and the switch that turns a step off.

The five steps

Step	Deletes
1. Print Tasks	Print tasks past the retention that have reached status `Printed`. One that never printed stays, so a problem does not disappear before anybody looked at it.
2. Mail Tasks	Mail tasks past the retention, whatever their status. The mail itself is long gone, the record is the receipt.
3. History Records	History records past the retention, together with their PDF and their xlsx export.
4. Orphaned Files	Files on the history disk that no history record and no mail task refers to any more, and that are older than the retention.
5. Scheduler Run Log	The run records of the Job Scheduler itself. The run doing the deleting never deletes itself.

> **The order is fixed and not configurable.** Print tasks and mail tasks point at the history records they belong to, so the database refuses to delete a record while one of them still

> references it. Running the steps in this order is what makes the cleanup work, and it is the reason all five belong in one schedule instead of five.

Saying how long

Every step has its own retention in days, and every step can be switched off on its own.

Value	Means
`90`	Delete what is older than 90 days.
`0`	Delete everything older than right now. Useful for a one-off clear-out, dangerous as a standing setting.
switched off	That entity is never purged. The step is named under `skipped` in the run result, so the log shows it was a decision and not an accident.

A sensible starting point keeps print and mail tasks shorter than the history records they belong to, and the orphaned files shortest of all. Something like 30, 30, 90, 7 and 90 days.

Saying when

A purge schedule says when it runs the same way every other schedule does: with a cron expression for the nightly housekeeping, or with a single date and time for a cleanup that belongs to one occasion, such as the clear-out after a migration. Both are on the *Job Scheduler* page.

A purge schedule that is waiting for its single date counts as active cleanup for as long as that date is still ahead, so the overview does not warn about missing housekeeping while one is armed. Once it has run, that cleanup is over; a *0* retention that was meant for exactly that one run does not stay armed for the next night.

Where it comes from

The base setup of an instance creates the schedule under the name **Purge Job**, deactivated and without a cron expression. Its five retentions start from the `PURGE_*_DAYS` keys of the `.env` file, which are read once at that moment. From then on the retentions live on the schedule, and the `.env` keys are not read again.

Nothing is cleaned up until you give the schedule an expression and switch it on. That is deliberate: a fresh instance should not start deleting on a schedule nobody chose.

```
0        3        *        *        *                                          every
0   3   *   *   0              Sunday   nights   only,   for   an   instance   with   little
```

Who may switch a step on

A Purge Job has no owner. It runs without any user at all, which means the usual permission check has nothing to check against. So the check moves to the moment a step is configured: whoever switches a step on, or changes its retention, has to hold the permission to delete what that step deletes.

Step	Needs
Print Tasks	`report-print-task:delete`
Mail Tasks	`report-mail-task:delete`
History Records, Orphaned Files	`report-history-record:delete`
Scheduler Run Log	`scheduled-job:delete`

Steps you leave untouched are not asked about again, so somebody may change the cron expression without holding every delete permission in the instance.

What a run tells you

The run record carries the numbers per step: how many records were eligible, how many were deleted, how many could not be. A step that throws does not stop the ones after it.

Outcome	Status
Every configured step finished	`Success`

Outcome	Status
Some steps finished, one failed or left records behind	`Warning`, with the reason in the result
Every configured step failed	`Error`, and the failure mail goes out

A history record that could not be deleted because something still references it is logged individually, not swallowed into a count.

Doing it by hand

The first four steps are each available as an artisan command as well, for a one-off cleanup or a maintenance window.

```
php artisan report-print-tasks:purge --days=30
php artisan report-mail-tasks:purge --days=30
php artisan report-history-records:purge --days=90
php artisan orphaned-files:purge --days=7
```

Run them in that order for the same reason the schedule does.

If the schedule is gone

The Purge Job is an ordinary schedule and can be deleted like any other. Nothing then cleans up, and the disk keeps filling. The Job Scheduler overview shows a warning for as long as no active purge schedule exists.

There are two ways back, and neither of them is special.

From the frontend. New on the Job Scheduler overview offers **Purge Job** next to Render Job. Pick it, set the retentions, give it a cron expression and switch it on. A purge schedule is created like any other schedule; it needs no owner and no API token, only a name that is still free and the delete permissions of the steps you switch on. The same way you would create a second one, for instance a weekly run with longer retentions next to a nightly one.

From the command line, for an instance you would rather not click through:

```
php                              artisan                              scheduler:create-purge-job
```

It creates the schedule under its original name when it is missing and does nothing when it is already there. It touches neither storage nor users, so it is safe on a live system. Afterwards give it its cron expression and switch it on again.

Revision #4
Created 2026-09-21 08:47:28 UTC by Benjamin Fischer
Updated 2026-09-21 15:56:58 UTC by Benjamin Fischer