Tiki Scheduler: Automate Maintenance, Search Indexing, and Notifications with a Single Cron Job
Tiki Scheduler: Automate Maintenance, Search Indexing, and Notifications with a Single Cron Job

 Bernard Sfez - 2026-07-02 13:43



Most Tiki Wiki maintenance tasks, from rebuilding the search index to sending notification digests, should never depend on someone remembering to click a button. The built-in Scheduler turns these chores into background jobs: set one cron entry on your server, then manage everything else from the Tiki interface.

The official documentation covers the basics in a few lines; this guide goes further, with the reasoning behind each setting and a set of tasks we actually deploy on client sites.

Why a Scheduler Matters in Tiki

A wiki, groupware platform, or CMS is never really "finished" once it's installed. The search index drifts out of sync as content changes, notification digests need to go out on a schedule, mail queues need flushing, and version checks need to happen quietly in the background. Before Tiki had its own scheduler, each of these needed its own line in the system crontab, calling its own PHP script, on its own schedule; multiply that across a dozen jobs, and every migration to a new server meant re-copying a fragile list of cron lines.

The internal Scheduler feature, introduced in Tiki17, solved this cleanly: only one entry needs to live in the system crontab. That single entry runs console.php scheduler:run, part of Tiki's built-in Console command-line tool, at a fixed interval, and Tiki itself decides which tasks are due, executes them, and logs the result. Everything else, adding a task, changing its schedule, pausing it, reviewing its history, is done from the admin interface, and travels with your Tiki instance across backups and server moves.

For teams that care about digital sovereignty and predictable operations, this matters: your automation logic lives inside your own Tiki database, not scattered across a server you may not fully control.

How the initiate the Scheduler (cron-job)

You'll find it in the Scheduler Admin module under "Settings" in the main application menu, at:

https://example.org/tiki-admin_schedulers.php

From there you can create, list and edit tasks. you will have also access to the logs, see their last run status, etc. Every task has a Run Time expressed in standard cron syntax, minute, hour, day of month, month, day of week, the same format used by GNU/Linux system crontabs:

```
| * | * | * | * | *
| | | | +---- Day of the Week (range: 1-7, 1 = Monday)
| | | +------ Month of the Year (range: 1-12)
| | +-------- Day of the Month (range: 1-31)
| +---------- Hour (range: 0-23)
+------------ Minute (range: 0-59)
```

So */5 * * * * means "every 5 minutes", 0 1 * * * means "every day at 1am". If you're new to cron syntax, crontab.guru is a handy way to check an expression before saving it.

Setting the Master Cron Job

The master cron job's only role is to ask Tiki, frequently, "is anything due?" It should run at least as often as your most frequent task; once a minute is common, once every 5 or 15 minutes is enough for most maintenance-style jobs. This is done by sending a command to the Tiki console that is inside your main Tiki directory.

Start by adding this to your server's crontab (adjust the path to your Tiki root):

To run every 15 minutes
```
*/15 * * * * php /your/tiki/path/console.php scheduler:run
```

To run every hour
```
0 * * * * /your/tiki/path/console.php scheduler:run
```

Process ownership matters. If any of your scheduled tasks write cache files, and most maintenance tasks do, the cron job must run as the same user as your webserver (www-data, apache, nginx...), not as root. Running it as the wrong user is one of the most common causes of "it works when I run it manually, but not from cron."

If you manage several Tiki instances, Tiki Manager can set this up for you automatically (instance:setup-scheduler-cron), which is worth knowing about if you're maintaining more than one or two sites, or if you use our Tiki Wiki support services to keep several client instances in sync.

scheduler:run isn't the only scheduler-related command worth knowing. Three others live in the same scheduler namespace and are handy for troubleshooting from the command line rather than the UI:

- `php console.php scheduler:stats` # table of scheduler task statistics

- `php console.php scheduler:monitor` # monitor scheduler jobs
- `php console.php scheduler:heal` # heal scheduled tasks stuck in a bad state

`scheduler:heal` in particular is worth remembering: if a task ever gets stuck showing "running" after a server restart or a crashed PHP process, this is the command that clears it without having to touch the database directly.

Keeping Your Logs Under Control

This is the detail the official page doesn't mention, and it's the first thing worth adjusting after enabling the Scheduler. Every execution of every task writes a log entry, and by default Tiki is generous, keeping a history that can grow to 10,000 entries per task. On a busy instance with several tasks running every few minutes, that log table fills up fast and adds unnecessary weight to your database for very little practical benefit; nobody reads ten thousand lines of "index:rebuild completed successfully."

Go to tiki-admin.php?page=general#col1 and search for "Number of logs to keep" (or use the search box in the admin panel) and adjust the scheduler_keep_logs preference. For most sites, 300 to 500 entries is more than enough, comfortably a few weeks or months of history depending on how often your tasks run, while keeping the log table light.

Testing the Scheduler: index:rebuild Is Your Best Friend

Once the master cron job is in place, don't wait for a real task's schedule to come around to confirm everything works. The simplest, and arguably most useful, task to test with is rebuilding the unified search index. It is included as template (disable) by default from Tiki29 : "Rebuild index every day". Adjust your master cron and the scheduler task cron to run it in the next 5 minutes or something like that. Watch the task's status in the Scheduler panel switch to "success" after the next master cron run. Once confirmed, set it back to a sensible schedule, once a night is typical for most sites.

This single test tells you three things at once: the master cron job is firing, Tiki correctly picks up due tasks, and the process ownership is correct, since an index rebuild will fail loudly if the cache/index directories aren't writable by the cron user.

Useful Tasks to Schedule

Once the mechanism is proven, the Scheduler becomes the natural home for anything you'd otherwise run by hand. A few we set up routinely, all using the ConsoleCommandTask with a console.php command:

- Keep your XML sitemap fresh, so search engines always see current content: `sitemap:generate`
- Auto-import files dropped into a folder on the server into a File Gallery, handy for automated exports from another system, scanners, or scripts that deposit files via SFTP: `php console.php files:batchupload 1 --subdirToSubgal --createSubgals` (Here 1 is the target gallery ID; add --fileUser www-data --fileGroup www-data if permissions need adjusting after import, and --filesPath /var/www/other/uploads if the drop folder lives outside the default batch directory.)
- Optimize the search index without a full rebuild, lighter than `index:rebuild`, useful between full rebuilds on larger sites: `index:optimize`

- Clear expired login tokens and stale caches on a nightly basis: `tokens:clear`, `cache:clear`
- Take a scheduled files backup (`backup:files` requires a destination path argument): `php console.php backup:files /home/destination/path/backups/files`

This is a good habit in general with console.php: run any new command by hand first, once, before you ever put it in a scheduled task. It's the fastest way to catch a missing argument or a permissions issue while you're watching, rather than a week later in a silent log line.

Each of these is a one-line addition once the master cron job exists. As covered above, Tiki can also email an admin automatically if a scheduled job fails, and again once it heals, configurable via the scheduler_users_to_notify_on_stalled and scheduler_users_to_notify_on_healed preferences.

Going Further: (Almost) No Limit to What You Can Automate

The examples above all use ConsoleCommandTask, but it isn't the only Task type the Scheduler supports, TikiCheckerCommandTask (used by the default "Tiki Check" preset task) is another built-in one, and the Scheduler is designed to be extensible beyond console.php commands. In particular, under the advanced Security preferences there is an option that lets a scheduled task run a raw shell command directly, not wrapped in console.php at all. Check tiki-admin.php?page=security on your own instance (filter the preferences for "scheduler" or "shell") to confirm the exact label and default state for your version, since this is exactly the kind of setting that should stay off unless you specifically need it. With it enabled, a scheduled task effectively becomes a cron job that lives inside Tiki's own interface, with logging and a UI, instead of a raw line in /etc/crontab.

We'd rather be conservative here: enable shell-command tasks only on instances you fully administer, keep the list of who can create scheduler tasks (tiki_p_admin_schedulers permission) tight, and prefer the standard console.php commands whenever one already does the job. The full, current list of available commands is always worth checking in the Console documentation, since new ones are added with almost every release.

To give a sense of scale, here is a non-exhaustive list of what's realistic to automate in a stock Tiki instance, no custom code required, built directly from the current console.php command list:

Task	How to set it (console.php command)	Remarks
Rebuild the search index	index:rebuild	Heaviest but most reliable; good nightly default
Optimize the search index	index:optimize	Lighter alternative between full rebuilds
Generate the XML sitemap	sitemap:generate	Keeps search engines in sync with new content
Send notification digests	notification:digest	Requires users to be subscribed to digest mode

Task	How to set it (console.php command)	Remarks
Send daily user reports	daily-report:send	Tiki's Daily Reports feature
Flush the mail queue	mail-queue:send	Only relevant if outgoing mail is set to "Queue"
Batch-import dropped files	files:batchupload 1 --subdirToSubgal --createSubgals	Great for automated exports landing via SFTP
Clear Tiki caches	cache:clear	Safe to run daily; regenerates on next page load
Remove expired tokens	tokens:clear	Housekeeping, no visible effect for users
Remove expired login cookies	users:remove-cookies	Complements tokens:clear
Recalculate tracker math fields	tracker:recalc	Only needed if trackers use computed fields
Refresh incoming RSS feeds	rss:refresh	Useful if you aggregate external feeds
Delete old system logs	log:delete	Pairs well with a defined retention policy
Sync calendar subscriptions	calendar:sync	For calendars pulling external ICS feeds
Update Tiki via version control	vcs:update	Best reserved for staging, not production, unattended
Back up instance files	backup:files /path/to/backup	Needs a writable destination path, see above
Back up the database	database:backup /path/to/backup	Pair with backup:files for a full nightly snapshot
Check installed packages	package:update	Review before enabling unattended on production

As with backup:files, a few of these (database:backup, files:batchupload) expect a path or gallery ID as an argument rather than running bare; the pattern from above applies: test the exact command by hand once, then wrap it in a scheduled task with that same, working syntax.

Wrapping Up

The Scheduler is one of those Tiki features that quietly pays for itself: a few minutes of setup, and your search index, notifications, sitemap, and housekeeping tasks all run themselves, reliably and with a visible history you can check at a glance. We hope this gives you a more complete picture than the short reference page, and a few ready-to-use commands to start with.

If you'd rather have this configured, monitored, and kept up to date for you, that's exactly what our Tiki Wiki support services are for. We're always happy to help.

Related Reading

- Tiki documentation: Scheduler, Cron, Console, Cron Job to Rebuild Search Index, Manager
- Profile: Scheduler_presets_20