

cs-sleeper

Manual

Spin down idle ZFS pool disks and wake them on demand - a small dependency-free daemon

Version 1.1.0-rc9 · October 13, 2026

Contents

1. Concept	3
2. Requirements	3
3. Installation	3
4. Quick start	4
5. Configuration	4
5.1 Command line reference	5
6. Platform matrix	6
7. Safety model	6
8. napp-it / csweb-gui integration	7
9. Exit codes and license	7

1. Concept

cs-sleeper is a small, dependency-free daemon (a single static Go binary) that watches the I/O activity of each disk and puts disks that have been idle for a configurable time (`wait`) into standby via `smartctl`. It refuses to touch a disk while a `zfs send` or `zfs receive` is in flight, so a running replication job is never interrupted. It also ships guarded one-shot commands to spin a single disk down or up and to export or import a whole pool.

cs-sleeper is part of the napp-it / csweb-gui tool family and runs on ZFS hosts: Linux, illumos, Solaris, FreeBSD and macOS (and Windows as a `smartctl`-only target).

Status: v1.1.0-rc9 is a release candidate (pre-release). Pre-built binaries are on the GitHub Releases page.

How it works - the daemon samples the I/O of every disk once per `interval` (default 5 s):

1. Activity is detected by comparing successive counters (Linux) or by reading the interval rate (illumos, Solaris, FreeBSD, macOS, Windows).
2. A disk without I/O accumulates idle time. Once it has been idle for `wait` seconds and has passed `verify-idle` consecutive idle re-checks, the daemon issues `smartctl -s standby,now` to spin it down.
3. Sleep is only allowed **outside** the configured `activity` windows (for example "do not sleep between 12:00-14:00 and 18:00-06:00" when backups or replication are expected).
4. Before sleeping, and before any `export-now`, the daemon checks for a running `zfs send` or `zfs receive`; if one is found, the action is skipped.
5. `sleeppool` flushes pending writes (`zpool sync`, or POSIX `sync` on illumos and Solaris) before it spins the disks of a pool down, then re-samples the I/O and skips every disk that became active during the flush. `zpool export` already flushes, so the export path needs no extra sync.

As a belt-and-braces fallback the daemon also sets the drive-internal standby timer (`smartctl -s standby,<standby-min>`) at startup, so the disks still spin down on their own if the daemon is stopped.

cs-sleeper only manages the disks you list in `disks` (plus the disks resolved from `pools`). It never touches disks in `exclude`, system disks or disks that are not listed.

2. Requirements

- a supported OS (see chapter 6),
- `smartctl` (smartmontools) in the `PATH`, with root or administrator privileges. On macOS, and on any host where `smartctl` lives outside the `PATH`, cs-sleeper also checks `/usr/local/sbin`, `/opt/local/sbin`, `/opt/homebrew/sbin`, `/usr/sbin` and `/sbin`,
- `zpool` and `zfs` in the `PATH` of the managed host (for the replication safety gate and the import/export commands),
- no other runtime dependencies; the binary is fully static (`CGO_ENABLED=0`).

3. Installation

Pre-built binaries are attached to every GitHub release. The layout mirrors the other napp-it tools so that the csweb-gui web interface can download them:

cs-sleeper-mswin.amd64.tar.gz	-> mswin.amd64/cs-sleeper.exe
cs-sleeper-linux.amd64.tar.gz	-> linux.amd64/cs-sleeper
cs-sleeper-linux.arm64.tar.gz	-> linux.arm64/cs-sleeper
cs-sleeper-illumos.amd64.tar.gz	-> illumos.amd64/cs-sleeper
cs-sleeper-solaris.amd64.tar.gz	-> solaris.amd64/cs-sleeper
cs-sleeper-freebsd.amd64.tar.gz	-> freebsd.amd64/cs-sleeper
cs-sleeper-darwin.amd64.tar.gz	-> darwin.amd64/cs-sleeper
cs-sleeper-darwin.arm64.tar.gz	-> darwin.arm64/cs-sleeper

For a manual install copy the binary of your platform to a directory in the `PATH` (for example `/usr/local/bin/cs-sleeper`) and make it executable. Every release also publishes a `checksums.txt`

(SHA-256); verify a download against it:

```
chmod +x /usr/local/bin/cs-sleeper
sha256sum -c checksums.txt --ignore-missing
```

The napp-it backend expects the tool in

`.../data/cs_server/tools/cs-sleeper/<os>.<arch>/cs-sleeper[.exe]`; the menu **About > Download cs-tools** puts it there.

4. Quick start

1. Show the created config path and the current defaults (creates the file)
`cs-sleeper status`

2. Edit the config and list the disks to manage
`vi /opt/csweb-gui/_cfg/cs-sleeper`
`disks = sda,ada0,c2t1d0` (device names, see the platform matrix)

3. Run once in the foreground to verify the detection
`cs-sleeper daemon --foreground --once`

4. Run as a daemon (background, pid file and state file)
`cs-sleeper daemon`

5. Inspect the state
`cs-sleeper status`

6. Manual control
`cs-sleeper sleepnow --disk sda`
`cs-sleeper wakeupnow --disk sda`
`cs-sleeper sleepnow --pool tank`
`cs-sleeper wakeupnow --pool tank`
`cs-sleeper export-now --pool tank`
`cs-sleeper import-now --pool tank`

5. Configuration

The config file is `/opt/csweb-gui/_cfg/cs-sleeper` (the `_cfg` directory used by the other napp-it tools). It is created with defaults the first time any command runs. Override the location with `--config PATH` or the environment variable `CS_SLEEPER_CONFIG`. Format: one `key = value` per line; `#` or `;` start a comment; empty lines are ignored; booleans accept `yes/no`, `true/false`, `on/off`, `1/0`.

Key	Default	Meaning
<code>enabled</code>	<code>yes</code>	whether the daemon runs at all; <code>no</code> makes <code>daemon</code> exit immediately
<code>disks</code>	<code>empty</code>	comma-separated standalone disks to manage directly (device names, see the platform matrix), in addition to the disks resolved from <code>pools</code> ; <code>hd</code> is still accepted as a legacy alias when reading
<code>exclude</code>	<code>empty</code>	comma-separated disks that are never touched
<code>pools</code>	<code>empty</code>	pools whose member disks are managed (via <code>zpool status</code>)
<code>activity</code>	<code>empty</code>	comma-separated hour ranges <code>H-H</code> (0-23, may cross midnight); sleep is allowed only outside these windows; <code>empty</code> = sleep any time
<code>wait</code>	<code>600</code>	seconds a disk must be idle before it is put to sleep
<code>interval</code>	<code>5</code>	sampling interval in seconds

Key	Default	Meaning
policy	standby	sleep policy; only standby (smartctl spin-down) is implemented
standby-min	10	drive-internal standby timer in minutes, set at daemon start as a fallback
wake	on-access	on-access logs disks that wake on access (shown as last-wake in status); manual disables the tracking (wake with wakeupnow or wakepool)
parallel	4	maximum concurrent sleep/wake operations; 0 = unlimited
verify-idle	5	consecutive idle samples required (after wait) before sleeping
vm-mode	off	off , proxmox_suspend or proxmox_shutdown (VM handling on sleep)
pool-rescan	60	seconds between pool disk re-resolutions in the daemon
state-dir	/var/run/cs-sleeper (Unix), %TEMP%\cs-sleeper (Windows)	directory for the pid file and state.json
log-file	/var/log/cs-sleeper.log (Unix), %TEMP%\cs-sleeper\cs-sleeper.log (Windows)	daemon log file
log-level	info	debug , info , warn or error
pid-file	<state-dir>/cs-sleeper.pid	pid file (prevents duplicate daemons), always derived from state-dir

The napp-it / csweb-gui backend overrides **state-dir**, **log-file** and **pid-file** in the config it writes and points them to **.../csweb-gui/tmp/cs-sleeper/...** on every platform.

```

enabled      = yes
disks        = sda,sdb,ada0,ada1
exclude      =
pools        = tank,backup
activity     = 12-14,18-6
wait         = 600
interval     = 5
policy       = standby
standby-min= 10
wake         = on-access
parallel     = 4
verify-idle= 5
vm-mode      = off
pool-rescan= 60

```

5.1 Command line reference

```

cs-sleeper daemon [--config PATH] [--foreground] [--once]
cs-sleeper status [--config PATH] [--json]
cs-sleeper enable  [--config PATH]
cs-sleeper disable [--config PATH]
cs-sleeper sleepnow --disk NAME [--config PATH]
cs-sleeper wakeupnow --disk NAME [--config PATH]
cs-sleeper sleeppool --pool NAME [--export] [--include-vm] [--at now|HH:MM] [--force]
cs-sleeper wakepool  --pool NAME [--include-vm] [--at now|HH:MM] [--force]
cs-sleeper import-now --pool NAME [--config PATH] [--force]
cs-sleeper export-now --pool NAME [--config PATH] [--force]
cs-sleeper version

```

Command	Description
<code>daemon</code>	idle detection loop; background by default, <code>--foreground</code> logs to stdout, <code>--once</code> runs one sample and exits (diagnostics)
<code>status</code>	one-shot report: config, pools, managed disks, daemon state and a live I/O sample; <code>--json</code> gives machine-readable output
<code>enable</code>	persist <code>enabled=yes</code> and start the daemon
<code>disable</code>	persist <code>enabled=no</code> and stop the daemon
<code>sleepnow</code>	spin one disk down immediately; refused while <code>zfs send/receive</code> runs
<code>wakeupnow</code>	spin one disk up immediately
<code>sleeppool</code>	put a whole pool to sleep (standby, or <code>--export</code> for backup pools); flushes pending writes before standby
<code>wakepool</code>	wake a pool (import if needed) and optionally its VMs
<code>import-now</code>	import one pool (<code>zpool import</code>)
<code>export-now</code>	export one pool (<code>zpool export</code>); refused while replication runs
<code>version</code>	print the version string

The daemon writes the state of each disk to `<state-dir>/state.json` after every sample. `status` reads it and also performs a fresh 1-second I/O sample for a live view.

6. Platform matrix

OS	Device names	I/O source	Sleep / wake
Linux	<code>sda</code> , <code>nvme0n1</code>	<code>/proc/diskstats</code> (cumulative)	<code>smartctl -s standby,now /dev/<name></code>
illumos	<code>c2t1d0</code>	<code>iostat -xn <n> 2</code> (rate)	<code>smartctl ... /dev/rdisk/<name>s2</code>
Solaris	<code>c2t1d0</code>	<code>iostat -xn <n> 2</code> (rate)	<code>smartctl ... /dev/rdisk/<name>s2</code>
FreeBSD	<code>ada0</code> , <code>da0</code>	<code>iostat -x -w <n> -c 2</code> (rate)	<code>smartctl -s standby,now /dev/<name></code>
Windows	<code>PhysicalDisk0</code>	PowerShell <code>Get-Counter</code> (rate)	<code>smartctl ... PhysicalDriveN</code> or OpenZFS <code>/dev/sdN</code>
macOS	<code>disk0</code>	<code>iostat -d -w <n> -c 2</code> (rate)	<code>smartctl -s standby,now /dev/<name></code>

- Use the device names exactly as the OS shows them (`lsblk`, `zpool status`, `iostat -n`, `camcontrol devlist`, `Get-PhysicalDisk`). `/dev/` prefixes are optional and stripped; partition or slice suffixes (`sda1`, `ada0s1`, `c2t1d0s2`) are normalized to the whole disk.
- On illumos and Solaris use the logical names shown by `iostat -n` (for example `c2t1d0`) - the same names `zpool status` uses.
- On Windows the idle detection uses the `PhysicalDisk N` performance counters; `smartctl` needs the matching `PhysicalDriveN` or OpenZFS `/dev/sdN` alias for the actual spin-down.
- On macOS the device names are the whole-disk names of `diskutil list` (for example `disk2`). `smartctl` needs root privileges there; the `smartmontools` installer places it in `/usr/local/sbin`, which `cs-sleeper` locates automatically. On APFS the boot disk is resolved to its physical store, so the real boot device is never spun down.

7. Safety model

`cs-sleeper` is deliberately conservative:

1. **Replication gate** - before sleeping a disk or exporting a pool it checks for a running `zfs send` or `zfs receive` (via `pgrep -f`). If one is found the action is skipped for this tick and logged.

2. **Idle detection** - a disk is slept only after `wait` seconds without I/O plus `verify-idle` consecutive idle samples, which avoids false triggers from transient activity.
3. **Activity windows** - sleep is refused inside the configured `activity` windows, so maintenance and replication slots are always respected.
4. **Explicit allow-list** - only disks listed in `disks` or `pools` (and not in `exclude`) are ever managed.
5. **No forced pool actions by default** - `export-now` and `import-now` pass `-f` to `zpool` only with `--force`, and refuse while replication runs.
6. **Never-sleep set** - the OS boot disk, the disks of the boot pool and SLOG, L2ARC, special and dedup flash devices are never spun down, in addition to everything in `exclude`. This is enforced for every command that can put a disk to standby or export a pool (`sleepnow`, `sleeppool`, `export-now` and the daemon loop); `sleeppool` and `export-now` also refuse outright if the pool is the boot pool.
7. **Flush before standby** - `sleeppool` runs `zpool sync <pool>` (POSIX `sync` on illumos and Solaris), then re-samples the I/O and skips any disk that became active. The daemon's continuous per-disk path needs no explicit flush: a disk sleeps only after minutes of inactivity, far longer than the transaction-group commit of about 5 s.
8. **Hardened external commands** - `smartctl`, `zpool`, `qm`, `iostat` and the other helpers are resolved via well-known absolute install paths first and a `$PATH` search only as a fallback (cs-sleeper normally runs as root or Administrator); every external call runs under a timeout so an unresponsive device cannot wedge the daemon; disk and pool names are validated before they are passed on as arguments.

Caveat: `smartctl` spin-down on some controllers needs extra `-d` options (for example USB bridges or specific HBAs). If `smartctl` fails, the error is logged and the disk is left alone: configure the drive so that `smartctl` sees it, or use the native tool of the host.

8. napp-it / csweb-gui integration

- Config: `/opt/csweb-gui/_cfg/cs-sleeper` (created on first use). The csweb-gui settings form writes `state-dir`, `log-file` and `pid-file` under `.../csweb-gui/tmp/cs-sleeper/...`
- Binary: `.../data/cs_server/tools/cs-sleeper/<os>.<arch>/cs-sleeper[.exe]`.
- Autostart: the csweb-gui backend starts the daemon from `server_boot_tasks.pl` (idempotent via the pid file and the `enabled` flag).
- Menu: **System > Services > Sleeper** offers `status`, `enable`, `disable`, `sleepnow`, `wakeupnow`, `sleeppool`, `wakepool`, `import-now` and `export-now`. `pools` is presented as two categorized fields, "active pools" and "backup pools" - purely a UI aid stored in a separate `_cfg/cs-sleeper.pooltype` file that cs-sleeper itself never reads. Every pool supports the full sleep/wake/export/import action set regardless of the category, and the `activity` window stays one global schedule for all managed disks.

The one-shot commands (`sleepnow`, `wakeupnow`, `import-now`, `export-now`) do not need autostart; only `daemon` does. `enable` and `disable` toggle the autostart by persisting the `enabled` flag and starting or stopping the daemon.

9. Exit codes and license

Code	Meaning
0	success (or daemon disabled)
1	error (config, smartctl, zpool, refused action)
2	usage error (missing required flag)

BSD 2-Clause - see the file LICENSE.