

cs-sleeper

Handbuch

Inaktive ZFS-Pool-Platten in den Ruhezustand schicken und bei Bedarf wecken - ein kleiner Daemon ohne Abhängigkeiten

Version 1.1.0-rc9 · 13. Oktober 2026

Inhalt

1. Konzept	3
2. Voraussetzungen	3
3. Installation	3
4. Schnellstart	4
5. Konfiguration	4
5.1 Kommandozeilen-Referenz	6
6. Plattform-Matrix	6
7. Sicherheitsmodell	7
8. Einbindung in napp-it / csweb-gui	7
9. Exit-Codes und Lizenz	8

1. Konzept

cs-sleeper ist ein kleiner Daemon ohne Abhängigkeiten (ein einzelnes statisches Go-Binary), der die I/O-Aktivität jeder Platte beobachtet und Platten, die für eine einstellbare Zeit (`wait`) inaktiv waren, per `smartctl` in den Standby schickt. Er rührt keine Platte an, solange ein `zfs send` oder `zfs receive` läuft, sodass ein laufender Replikations-Job nie unterbrochen wird. Dazu kommen abgesicherte Einzelbefehle, um eine einzelne Platte herunter- oder hochzufahren sowie einen ganzen Pool zu exportieren oder zu importieren.

cs-sleeper gehört zur napp-it / csweb-gui Tool-Familie und läuft auf ZFS-Hosts: Linux, illumos, Solaris, FreeBSD und macOS (sowie Windows als reines `smartctl`-Ziel).

Status: v1.1.0-rc9 ist ein Release Candidate (Vorabversion). Fertige Binaries gibt es auf der [GitHub-Releases-Seite](#).

So arbeitet er - der Daemon misst die I/O jeder Platte einmal je `interval` (Standard 5 s):

1. Aktivität wird erkannt, indem aufeinanderfolgende Zähler verglichen werden (Linux) oder die Rate des Intervalls gelesen wird (illumos, Solaris, FreeBSD, macOS, Windows).
2. Eine Platte ohne I/O sammelt Leerlaufzeit. Ist sie `wait` Sekunden inaktiv gewesen und hat `verify-idle` aufeinanderfolgende Leerlauf-Prüfungen bestanden, schickt der Daemon `smartctl -s standby,now`.
3. Schlafen ist nur **außerhalb** der konfigurierten `activity`-Fenster erlaubt (zum Beispiel "nicht schlafen von 12:00-14:00 und 18:00-06:00", wenn Backups oder Replikation erwartet werden).
4. Vor dem Schlafen und vor jedem `export-now` prüft der Daemon, ob ein `zfs send` oder `zfs receive` läuft; wenn ja, wird die Aktion übersprungen.
5. `sleeppool` schreibt ausstehende Daten weg (`zpool sync`, unter illumos und Solaris POSIX `sync`), bevor die Platten eines Pools heruntergefahren werden, misst dann die I/O erneut und überspringt jede Platte, die währenddessen aktiv wurde. `zpool export` schreibt ohnehin alles weg, der Export-Pfad braucht kein zusätzliches Sync.

Als zusätzliche Absicherung setzt der Daemon beim Start auch den Standby-Timer im Laufwerk (`smartctl -s standby,<standby-min>`), sodass die Platten auch dann von selbst herunterfahren, wenn der Daemon gestoppt wird.

cs-sleeper verwaltet nur die Platten, die in `disks` stehen (plus die aus `pools` ermittelten). Platten in `exclude`, Systemplatten und nicht aufgeführte Platten fasst er nie an.

2. Voraussetzungen

- ein unterstütztes Betriebssystem (siehe Kapitel 6),
- `smartctl` (smartmontools) im `PATH`, mit root- bzw. Administrator-Rechten. Unter macOS und auf jedem Host, auf dem `smartctl` außerhalb des `PATH` liegt, prüft cs-sleeper zusätzlich `/usr/local/sbin`, `/opt/local/sbin`, `/opt/homebrew/sbin`, `/usr/sbin` und `/sbin`,
- `zpool` und `zfs` im `PATH` des verwalteten Hosts (für die Replikations-Sperre und die Import/Export-Befehle),
- keine weiteren Laufzeit-Abhängigkeiten; das Binary ist vollständig statisch (`CGO_ENABLED=0`).

3. Installation

Fertige Binaries hängen an jedem GitHub-Release. Das Layout entspricht den anderen napp-it Tools, damit die csweb-gui Oberfläche sie herunterladen kann:

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
```

Für eine manuelle Installation das Binary der eigenen Plattform in ein Verzeichnis im **PATH** kopieren (zum Beispiel `/usr/local/bin/cs-sleeper`) und ausführbar machen. Jedes Release enthält auch eine `checksums.txt` (SHA-256); ein Download sollte damit geprüft werden:

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

Das napp-it Backend erwartet das Tool in

`.../data/cs_server/tools/cs-sleeper/<os>.<arch>/cs-sleeper[.exe]`; das Menü **About > Download cs-tools** legt es dort ab.

4. Schnellstart

1. Pfad der erzeugten Konfiguration und aktuelle Standardwerte anzeigen (legt die Datei an)
`cs-sleeper status`

2. Konfiguration bearbeiten und die zu verwaltenden Platten eintragen
`vi /opt/csweb-gui/_cfg/cs-sleeper`
`disks = sda,ada0,c2t1d0` (Gerätenamen, siehe Plattform-Matrix)

3. Einmal im Vordergrund laufen lassen, um die Erkennung zu prüfen
`cs-sleeper daemon --foreground --once`

4. Als Daemon starten (Hintergrund, Pid-Datei und Status-Datei)
`cs-sleeper daemon`

5. Zustand anzeigen
`cs-sleeper status`

6. Manuelle Steuerung
`cs-sleeper sleepnow --disk sda`
`cs-sleeper wakeupnow --disk sda`
`cs-sleeper sleeppool --pool tank`
`cs-sleeper wakepool --pool tank`
`cs-sleeper export-now --pool tank`
`cs-sleeper import-now --pool tank`

5. Konfiguration

Die Konfigurationsdatei ist `/opt/csweb-gui/_cfg/cs-sleeper` (das `_cfg`-Verzeichnis der anderen napp-it Tools). Sie wird beim ersten Aufruf eines beliebigen Befehls mit Standardwerten angelegt. Der Ort lässt sich mit `--config PFAD` oder der Umgebungsvariable `CS_SLEEPER_CONFIG` ändern. Format: eine Zeile `schlüssel = wert; #` oder `;` beginnen einen Kommentar; leere Zeilen werden ignoriert; boolesche Werte akzeptieren `yes/no`, `true/false`, `on/off`, `1/0`.

Schlüssel	Standard	Bedeutung
<code>enabled</code>	<code>yes</code>	ob der Daemon überhaupt läuft; bei <code>no</code> beendet sich <code>daemon</code> sofort
<code>disks</code>	leer	kommagetrennte einzelne Platten, die direkt verwaltet werden (Gerätenamen, siehe Plattform-Matrix), zusätzlich zu den aus <code>pools</code> ermittelten; <code>hd</code> wird beim Lesen weiter als alter Alias akzeptiert
<code>exclude</code>	leer	kommagetrennte Platten, die nie angefasst werden
<code>pools</code>	leer	Pools, deren Mitgliedsplatten verwaltet werden (über <code>zpool status</code>)

Schlüssel	Standard	Bedeutung
activity	leer	kommagetrennte Stundenbereiche H-H (0-23, dürfen über Mitternacht gehen); Schlafen ist nur außerhalb dieser Fenster erlaubt; leer = jederzeit schlafen
wait	600	Sekunden, die eine Platte inaktiv sein muss, bevor sie schlafen gelegt wird
interval	5	Messintervall in Sekunden
policy	standby	Schlaf-Richtlinie; nur standby (smartctl-Spin-down) ist umgesetzt
standby-min	10	Standby-Timer im Laufwerk in Minuten, beim Start des Daemons als Rückfall gesetzt
wake	on-access	on-access protokolliert Platten, die bei Zugriff aufwachen (in status als last-wake); manual schaltet die Verfolgung ab (Wecken mit wakeupnow oder wakepool)
parallel	4	maximale Zahl gleichzeitiger Schlaf-/Wecken-Vorgänge; 0 = unbegrenzt
verify-idle	5	aufeinanderfolgende Leerlauf-Messungen, die (nach wait) vor dem Schlafen nötig sind
vm-mode	off	off , proxmox_suspend oder proxmox_shutdown (Umgang mit VMs beim Schlafen)
pool-rescan	60	Sekunden zwischen erneuten Ermittlungen der Pool-Platten im Daemon
state-dir	/var/run/cs-sleeper (Unix), %TEMP%\cs-sleeper (Windows)	Verzeichnis für Pid-Datei und state.json
log-file	/var/log/cs-sleeper.log (Unix), %TEMP%\cs-sleeper\cs-sleeper.log (Windows)	Logdatei des Daemons
log-level	info	debug , info , warn oder error
pid-file	<state-dir>/cs-sleeper.pid	Pid-Datei (verhindert doppelte Daemons), immer aus state-dir abgeleitet

Das napp-it / csweb-gui Backend überschreibt **state-dir**, **log-file** und **pid-file** in der Konfiguration, die es schreibt, und setzt sie auf allen Plattformen auf **.../csweb-gui/tmp/cs-sleeper/...**

```

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 Kommandozeilen-Referenz

```
cs-sleeper daemon [--config PFAD] [--foreground] [--once]
cs-sleeper status [--config PFAD] [--json]
cs-sleeper enable [--config PFAD]
cs-sleeper disable [--config PFAD]
cs-sleeper sleepnow --disk NAME [--config PFAD]
cs-sleeper wakeupnow --disk NAME [--config PFAD]
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 PFAD] [--force]
cs-sleeper export-now --pool NAME [--config PFAD] [--force]
cs-sleeper version
```

Befehl	Beschreibung
daemon	Leerlauf-Erkennungsschleife; standardmäßig im Hintergrund, --foreground loggt auf stdout, --once macht eine Messung und endet (Diagnose)
status	Einmal-Bericht: Konfiguration, Pools, verwaltete Platten, Daemon-Zustand und eine aktuelle I/O-Messung; --json liefert maschinenlesbare Ausgabe
enable	enabled=yes speichern und den Daemon starten
disable	enabled=no speichern und den Daemon stoppen
sleepnow	eine Platte sofort herunterfahren; wird abgelehnt, solange zfs send/receive läuft
wakeupnow	eine Platte sofort hochfahren
sleeppool	einen ganzen Pool schlafen legen (Standby, oder --export für Backup-Pools); schreibt vor dem Standby ausstehende Daten weg
wakepool	einen Pool wecken (bei Bedarf importieren) und optional seine VMs
import-now	einen Pool importieren (zpool import)
export-now	einen Pool exportieren (zpool export); wird abgelehnt, solange eine Replikation läuft
version	Versionsstring ausgeben

Der Daemon schreibt den Zustand jeder Platte nach jeder Messung in `<state-dir>/state.json`. **status** liest ihn und macht zusätzlich eine frische 1-Sekunden-Messung für eine Live-Ansicht.

6. Plattform-Matrix

OS	Gerätenamen	I/O-Quelle	Schlafen / Wecken
Linux	sda, nvme0n1	/proc/diskstats (kumulativ)	smartctl -s standby,now /dev/<name>
illumos	c2t1d0	iostat -xn <n> 2 (Rate)	smartctl ... /dev/rdisk/<name>s2
Solaris	c2t1d0	iostat -xn <n> 2 (Rate)	smartctl ... /dev/rdisk/<name>s2
FreeBSD	ada0, da0	iostat -x -w <n> -c 2 (Rate)	smartctl -s standby,now /dev/<name>
Windows	PhysicalDisk0	PowerShell Get-Counter (Rate)	smartctl ... PhysicalDriveN oder OpenZFS /dev/sdN
macOS	disk0	iostat -d -w <n> -c 2 (Rate)	smartctl -s standby,now /dev/<name>

- Die Gerätenamen genau so verwenden, wie das Betriebssystem sie zeigt (**lsblk**, **zpool status**, **iostat -n**, **camcontrol devlist**, **Get-PhysicalDisk**). Das Präfix **/dev/** ist optional und wird entfernt; Partitions- oder Slice-Endungen (**sda1**, **ada0s1**, **c2t1d0s2**) werden auf die ganze Platte normalisiert.
- Unter illumos und Solaris die logischen Namen von **iostat -n** verwenden (zum Beispiel **c2t1d0**) - es sind dieselben Namen wie bei **zpool status**.

- Unter Windows nutzt die Leerlauf-Erkennung die `PhysicalDisk N` Performance-Zähler; smartctl braucht für das eigentliche Herunterfahren den passenden Alias `PhysicalDriveN` oder OpenZFS `/dev/sdN`.
- Unter macOS sind die Gerätenamen die Namen ganzer Platten aus `diskutil list` (zum Beispiel `disk2`). smartctl braucht dort root-Rechte; der smartmontools-Installer legt es in `/usr/local/sbin` ab, das cs-sleeper automatisch findet. Unter APFS wird die Bootplatte auf ihren physischen Speicher aufgelöst, sodass das echte Boot-Gerät nie heruntergefahren wird.

7. Sicherheitsmodell

cs-sleeper ist bewusst vorsichtig:

1. **Replikations-Sperre** - vor dem Schlafen einer Platte oder dem Export eines Pools wird geprüft, ob ein `zfs send` oder `zfs receive` läuft (per `pgrep -f`). Wird eines gefunden, wird die Aktion für diesen Durchlauf übersprungen und protokolliert.
2. **Leerlauf-Erkennung** - eine Platte schläft erst nach `wait` Sekunden ohne I/O plus `verify-idle` aufeinanderfolgenden Leerlauf-Messungen; das vermeidet Fehlauflösungen durch kurze Aktivität.
3. **Aktivitätsfenster** - Schlafen wird innerhalb der konfigurierten `activity`-Fenster verweigert, Wartungs- und Replikationszeiten werden also immer respektiert.
4. **Explizite Positivliste** - verwaltet werden nur Platten aus `disks` oder `pools` (und nicht in `exclude`).
5. **Standardmäßig keine erzwungenen Pool-Aktionen** - `export-now` und `import-now` geben `-f` nur mit `--force` an `zpool` weiter und verweigern, solange eine Replikation läuft.
6. **Nie-schlafen-Menge** - die Boot-Platte des Betriebssystems, die Platten des Boot-Pools sowie SLOG-, L2ARC-, Special- und Dedup-Flash-Geräte werden nie heruntergefahren, zusätzlich zu allem in `exclude`. Das gilt für jeden Befehl, der eine Platte in den Standby schicken oder einen Pool exportieren kann (`sleepnow`, `sleeppool`, `export-now` und die Daemon-Schleife); `sleeppool` und `export-now` verweigern außerdem, wenn der Pool der Boot-Pool ist.
7. **Wegschreiben vor dem Standby** - `sleeppool` führt `zpool sync <pool>` aus (unter illumos und Solaris POSIX `sync`), misst dann die I/O erneut und überspringt jede Platte, die aktiv wurde. Der fortlaufende Einzelplatten-Pfad des Daemons braucht kein explizites Wegschreiben: Eine Platte schläft erst nach Minuten ohne Aktivität, viel länger als der Transaction-Group-Commit von etwa 5 s.
8. **Abgesicherte externe Befehle** - `smartctl`, `zpool`, `qm`, `iostat` und die anderen Helfer werden zuerst über bekannte absolute Installationspfade aufgelöst und nur als Rückfall über den `$PATH` (cs-sleeper läuft normalerweise als root oder Administrator); jeder externe Aufruf läuft mit Timeout, sodass ein nicht antwortendes Gerät den Daemon nicht festhalten kann; Platten- und Pool-Namen werden geprüft, bevor sie als Argumente weitergegeben werden.

Hinweis: Das smartctl-Herunterfahren braucht bei manchen Controllern zusätzliche `-d`-Optionen (zum Beispiel USB-Bridges oder bestimmte HBAs). Schlägt smartctl fehl, wird der Fehler protokolliert und die Platte in Ruhe gelassen: das Laufwerk so konfigurieren, dass smartctl es sieht, oder das native Werkzeug des Hosts verwenden.

8. Einbindung in napp-it / csweb-gui

- Konfiguration: `/opt/csweb-gui/_cfg/cs-sleeper` (beim ersten Gebrauch angelegt). Das Einstellungsformular von csweb-gui schreibt `state-dir`, `log-file` und `pid-file` unter `.../csweb-gui/tmp/cs-sleeper/...`.
- Binary: `.../data/cs_server/tools/cs-sleeper/<os>.<arch>/cs-sleeper[.exe]`.
- Autostart: Das csweb-gui Backend startet den Daemon aus `server_boot_tasks.pl` (idempotent über die Pid-Datei und das Flag `enabled`).

- Menü: **System > Services > Sleeper** bietet `status`, `enable`, `disable`, `sleepnow`, `wakeupnow`, `sleeppool`, `wakepool`, `import-now` und `export-now`. `pools` wird als zwei Felder dargestellt, "active pools" und "backup pools" - reine Bedienungshilfe, in einer eigenen Datei `_cfg/cs-sleeper.pooltype` abgelegt, die cs-sleeper selbst nie liest. Jeder Pool unterstützt unabhängig von der Kategorie alle Aktionen Schlafen, Wecken, Export und Import, und das `activity`-Fenster bleibt ein globaler Zeitplan für alle verwalteten Platten.

Die Einzelbefehle (`sleepnow`, `wakeupnow`, `import-now`, `export-now`) brauchen keinen Autostart; nur `daemon` braucht ihn. `enable` und `disable` schalten den Autostart um, indem sie das Flag `enabled` speichern und den Daemon starten oder stoppen.

9. Exit-Codes und Lizenz

Code	Bedeutung
0	Erfolg (oder Daemon deaktiviert)
1	Fehler (Konfiguration, smartctl, zpool, abgelehnte Aktion)
2	Aufruffehler (fehlende Pflicht-Option)

BSD 2-Clause - siehe Datei LICENSE.