

cs-freeze4snap

Manual

Consistent ZFS snapshots for Proxmox VM/LXC guests and ESXi VMs -
freeze if possible, snapshot regardless

Version 1.3.0 · October 13, 2026

Contents

1. Why freeze	3
2. Options	3
3. How it works	4
4. Freeze chains	4
5. Proxmox	5
5.1 The memory step	6
5.2 /etc/pve must be part of the snapshot	6
5.3 Restore (Proxmox)	6
6. ESXi	6
6.1 How the VMs are found	7
6.2 Connection: --cfg	7
6.3 Freeze and thaw as separate steps	8
6.4 Restoring (ESXi)	8
7. Use from napp-it CS jobs	8
8. Howto setup (napp-it 4ai)	8
9. Result format	9
10. Requirements, tests, license	10

1. Why freeze

ZFS snapshots can never corrupt a pool (atomic txg commits) - but a snapshot taken while a guest is writing is, from the guest's own point of view, the same as pulling the power cord: whatever was on disk at that instant is what the guest sees on the next boot.

cs-freeze4snap pauses the guests (or only their writes) right before the snapshot, so the captured state is a clean stopping point instead of an arbitrary mid-write instant. It **never blocks the snapshot**: a guest that cannot be frozen is snapshotted as it is and reported in **warnings**.

It is part of the napp-it 4ai (client-server) cluster tooling. napp-it CS installs and updates it per member with the menu **About > Download cs-tools**. ESXi mode needs v1.3.0 or newer (napp-it CS warns when it finds an older build).

Quick example:

```
cs-freeze4snap snap --dataset rpool/data --name 20260809_1530

{"dataset":"rpool/data","snapshot":"20260809_1530","status":"ok","freeze_ms":6,
 "guests":[
  {"vmid":100,"type":"vm","platform":"proxmox-qemu","frozen":true,"strategy":"qga","thawed":true},
  {"vmid":101,"type":"vm","platform":"proxmox-qemu","frozen":true,"strategy":"qmp-
    pause","thawed":true},
  {"vmid":103,"type":"lxc","platform":"proxmox-
    lxc","frozen":true,"strategy":"cgroup","thawed":true}
 ]}
```

2. Options

```
cs-freeze4snap snap --dataset <ds> --name <snapname> [options]
cs-freeze4snap snap --hypervisor esxi --cfg <file> --dataset <ds> --name <snapname> [options]
cs-freeze4snap discover|freeze|thaw|cleanup --cfg <file> ... (ESXi)
```

Option	Meaning
<code>--dataset string</code>	ZFS dataset to snapshot (required), e.g. <code>rpool/data</code>
<code>--name string</code>	snapshot name (required), e.g. <code>20260809_1530</code>
<code>--recursive</code>	pass <code>-r</code> to <code>zfs snapshot</code> (default true)
<code>--exclude string</code>	comma-separated VMIDs to skip freezing
<code>--include-only string</code>	comma-separated VMIDs; replaces the discovery entirely
<code>--timeout duration</code>	maximum time per guest freeze step (30 s, ESXi 120 s; chain timeouts win)
<code>--forcefreeze</code>	abort WITHOUT snapshotting if a guest could not be frozen
<code>--policy 'chain'</code>	freeze chain, repeatable (see chapter 4), or <code>memkeep=N</code>
<code>--pre-snap-cmd 'cmd'</code>	run after the freeze, right before the ZFS snapshot (a failure is only a warning)

ESXi mode (the tool may run on any machine that reaches the ESXi host):

Option	Meaning		
<code>--hypervis or esxi</code>	hot-snapshot ESXi VMs instead of Proxmox guests		
<code>--cfg file</code>	connection file: host table or key=value (see chapter 6)		
<code>--proto auto\`</code>	ssh\`	soap\`	transport (default auto = soap, ssh if soap is unreachable)
<code>--host, --user</code>	ESXi host and login (default user root; the password is never a command line flag)		

Option	Meaning		
<code>--storage nfs</code>	select the VMs by storage (only nfs)		
<code>--nfs-path path</code>	NFS export path as ESXi mounts it (default: the mountpoint of <code>--dataset</code>)		
<code>--nfs-server host</code>	NFS server as ESXi knows it (needed if the path is exported twice)		
<code>`--vms all`</code>	<code>id,name,...`</code>	which VMs (default all; <code>--snap</code> is an alias)	
<code>`--mode freeze`</code>	<code>mem`</code>	<code>plain`</code>	one chain for every VM: <code>freeze,plain,zfs / memory,plain,zfs / plain,zfs</code>
<code>--allow-mixed</code>	also snapshot VMs that have disks on other datastores		
<code>--include-off</code>	also snapshot powered-off or suspended VMs		
<code>--thaw-timeout dur</code>	maximum time to remove one VM snapshot (2 m)		
<code>--zfs-cmd 'command'</code>	take the ZFS snapshot remotely, e.g. <code>ssh nas zfs snapshot -r {fullname}</code>		
<code>--state file</code>	freeze/thaw: file that carries the VM snapshots		

Subcommands (ESXi): `discover` lists the NFS datastores and VMs that would be snapshotted (JSON) and changes nothing; `freeze` takes the VM snapshots, writes `--state` and leaves them in place; `thaw` removes the VM snapshots listed in `--state`; `cleanup` removes leftover VM snapshots of this tool (tag `cs-freeze4snap`) on the NFS.

Guest discovery is automatic: `zfs list -r <dataset>` plus the Proxmox naming convention (`vm-<id>-disk-*`, `subvol-<id>-disk-*`) - there is no VMID list to maintain. `--exclude` and `--include-only` cover the exceptional case. The result is one JSON object on stdout, meant to be parsed by the caller (see chapter 9).

3. How it works

All guests of the dataset are discovered and frozen **in parallel**, then **one** `zfs snapshot -r` is taken (covering all of them at the same instant), then everything is thawed in parallel. The slowest guest determines how long the freeze phase lasts, not the sum of all guests - so guests that share state pause at approximately the same moment.

```
discover guests on the dataset
    |
    v
freeze VM-A --+
freeze VM-B --+--> wait for all --> ONE zfs snapshot -r --> thaw all
freeze LXC-C --+      (parallel)      (atomic)      (parallel)
```

4. Freeze chains

What "freezing" a guest means is a **chain** of steps, tried from left to right - the first step that works wins, then the ZFS snapshot is taken.

Step	ESXi	Proxmox VM	Proxmox LXC
freeze (alias quiesce)	VM snapshot, guest filesystem quiesced by VMware Tools	QEMU guest agent fsfreeze	fsfreeze , else cgroup freeze
memory (alias mem)	VM snapshot including the RAM (hot snapshot, slower; a restore resumes the running state)	qm snapshot --vmstate	-
plain	VM snapshot of the disks only (crash-consistent)	-	-
pause	-	QMP stop/cont (the whole VM pauses)	-
zfs	give up freezing, take the ZFS snapshot as it is	same	same

Steps that do not exist on a platform are skipped, so one chain can serve ESXi and Proxmox.

```
[freeze,memory,zfs,30]           # global chain: freeze, else memory, else ZFS; 30 s per step
192.168.2.48:*,freeze,plain,zfs  # every VM of this ESXi host
192.168.2.48:vm100,memory,zfs    # one VM (vm100, ct100 or 100)
192.168.2.203:vm101,freeze,memory # Proxmox member; no zfs at the end -> strict
```

Precedence (strongest first): **--mode**, **--policy**, a **host:vmid** line, a **host:*** line, the global [...] line, the built-in default (**freeze,memory,zfs** for ESXi and Proxmox VMs, **freeze,zfs** for LXC). **--policy** is repeatable and takes the same lines without the host prefix. A number is a timeout in seconds (**memory:300** for one step, a bare number for every step without its own). The chain that applied is reported in `guests[].chain`.

freeze,memory,zfs gives a filesystem-consistent point when the guest tools work, and a running-state point (RAM included) when they do not. **plain** and **pause** are crash-consistent and have to be asked for. A chain of only **zfs** never touches the guest (**zfs-only**).

Best effort or strict. A chain that ends with **zfs** is best effort: if every step fails, the guest is reported as not frozen and the ZFS snapshot is taken anyway. A chain **without** **zfs** is strict for its guests: if no step works, the run aborts before the ZFS snapshot (exit 1, **status: error**) - like **--forcefreeze**, but per guest. A Proxmox VM that is not running counts as consistent (**stopped**, no snapshot) and does not break a strict chain.

Timeouts. A step that runs out of time counts as failed and the next step starts. With soap the vSphere task is cancelled first; with ssh a running **vim-cmd** cannot be cancelled from outside (a snapshot the host may still create is removed), so prefer soap when you rely on step timeouts. For Proxmox the timeout bounds each command of a step.

Never a hard failure unless a chain is strict or **--forcefreeze** is set: a guest that cannot be frozen gets **frozen: false** plus a warning, an unreachable ESXi host or a wrong password is reported the same way, and the ZFS snapshot is taken in any case.

5. Proxmox

The tool runs **on the Proxmox host** (not inside a guest): it shells out to **zfs** and **qm/pct** and talks to the QGA/QMP sockets that only exist there.

- Guests are found per dataset by the naming convention - no VMID list.
- A VM with disks on more than one pool is reported in **warnings: zfs snapshot -r** never crosses pool boundaries, so only the disks under the targeted dataset are in the snapshot (**vmid 101 (vm) has disk(s) outside rpool/data that will NOT be included in this snapshot**).
- **freeze** needs **qemu-guest-agent** in the guest. Without it the chain falls back to **memory** (VM snapshot with the RAM) or, if you ask for it, to **pause** (QMP **stop/cont**).
- LXC: **fsfreeze** is applied on the host to the mountpoint of the container (this also works for unprivileged containers). If the filesystem does not support **FIFREEZE** - OpenZFS often does not, check with **fsfreeze -f <mountpoint>** - the tool syncs and freezes the container via cgroup v2.

5.1 The memory step

Without a guest agent the default chain `freeze,memory,zfs` falls back to `memory`. The tool runs

```
qm snapshot <vmid> cs4s_<snapshot> --vmstate 1 --description "cs-freeze4snap <snapshot>"
```

which needs no guest tools: `qm` stops the VM for a moment (seconds, depending on the RAM size), writes the RAM into a `vm-<id>-state-...` volume and snapshots the disks at that same point. A restore gives a running VM in exactly that state.

- The VM snapshot is **not** removed after the ZFS snapshot: on ZFS the consistent state is the pair `vm-<id>-disk-N@cs4s_<snapshot>` plus the vmstate volume. The recursive `pool@<snapshot>` is taken a moment later and holds disks slightly newer than the RAM state.
- Such a VM snapshot lives as long as the ZFS snapshot named in its description: every run removes the `cs4s_` snapshots of a VM whose ZFS snapshot is gone (10 minutes grace for parallel jobs). Only snapshots named `cs4s_*` with the description `cs-freeze4snap ...` are ever touched, and a snapshot that a running restore still needs is kept.
- `memkeep=N` (global line, `host:memkeep=N` or `--policy 'memkeep=N'`) caps the memory snapshots per VM (0 = no cap). Proxmox reserves the vmstate volume thick (about 2.2 x RAM), so for RAM-heavy VMs with a long retention set for example `memkeep=3`.
- A failed or timed-out `qm snapshot` is removed again; the default timeout of the step is 120 s.
- The result of such a VM has `strategy: "qm-mem"`, `frozen: true` and a warning that names a VM snapshot that failed before.

5.2 /etc/pve must be part of the snapshot

The memory snapshot is a section of `/etc/pve/qemu-server/<id>.conf`, and `qm rollback` needs it after a restore, so the config has to travel with the snapshot. napp-it CS adds `/etc/pve` to `include=` automatically for `freeze=proxmox` and syncs it again right before the ZFS snapshot. When calling the tool yourself:

```
cs-freeze4snap snap --dataset tank/vm --name s1 \  
  --pre-snap-cmd 'rsync -a --delete /etc/pve/ /tank/vm/_include/_etc_pve'
```

The same combination is worth it for plain snapshot jobs: napp-it CS `include=<folders>` syncs regular folders into `<dataset>/_include` before the snapshot, so one **recursive** snapshot holds the guest configuration and the frozen disk data together:

1. include -> /etc/pve synced into `<dataset>/_include/_etc_pve`
2. freeze -> the chain pauses the guests
3. snapshot -> one atomic recursive zfs snapshot -r captures both

5.3 Restore (Proxmox)

VM stopped, disks and config back from the ZFS snapshot (or from the replication target):

```
zfs rollback -r tank/vm/vm-100-disk-0@cs4s_<snapshot>    # every disk of the VM  
qm rollback 100 cs4s_<snapshot>                        # RAM state, the VM comes up running
```

`zfs rollback -r` first is required: Proxmox refuses `qm rollback` while a newer ZFS snapshot exists on the disk. If the config is gone (new host), copy it back first from `<dataset>/_include/_etc_pve/qemu-server/<id>.conf`.

6. ESXi

`cs-freeze4snap` can snapshot an **NFS export that is used as an ESXi datastore** in a VM-consistent way (v1.3.0 or newer): it finds the VMs on that NFS itself, takes a VM snapshot of each powered-on VM (the "freeze"), takes the ZFS snapshot and removes the VM snapshots again (the "thaw"). The ZFS snapshot then holds every VM with an embedded ESXi snapshot: after a restore, revert the VM to that snapshot to get a filesystem-consistent (`freeze`) or running-state (`mem`) image instead of a crash image.

```
cs-freeze4snap snap --hypervisor esxi --cfg /path/esxi.cfg \
--dataset tank/nfs --name auto_20260919_2100 --storage nfs --vms all
```

The ESXi host is reached over the network (ssh or soap), so the tool does **not** have to run on the ZFS server - any machine that can reach the ESXi host works (a frontend, a jump host). The ZFS snapshot is taken by the local `zfs`, or on a remote machine by `--zfs-cmd`:

```
cs-freeze4snap snap --hypervisor esxi --cfg esxi.cfg --dataset tank/nfs --name s1 \
--zfs-cmd 'ssh root@nas zfs snapshot -r {fullname}'
```

`{dataset}`, `{snapshot}` and `{fullname}` are replaced; the names are checked against `[A-Za-z0-9_.:/-]` first. The command runs via `sh -c (cmd /C on Windows)`.

6.1 How the VMs are found

1. The NFS export path comes from `--nfs-path`, or from the mountpoint of `--dataset` (without a local `zfs` it assumes `/<dataset>` and warns).
2. The NFS datastores of the host whose export matches are selected, plus the exports of child datasets with `--recursive` (the default). If the same path is exported by more than one NFS server, name it with `--nfs-server`.
3. Every VM with files on those datastores is a candidate; snapshotted are the powered-on VMs (`--vms all`, or a comma list of ids and names). Skipped, with the reason in `skipped`: powered-off or suspended VMs (they are consistent anyway; `--include-off` snapshots them too), VMs that also have disks on another datastore (a snapshot of this NFS would hold only part of them; `--allow-mixed` snapshots them anyway, with a warning) and VMs not in the `--vms` list.

`discover` shows exactly that without changing anything:

```
cs-freeze4snap discover --cfg esxi.cfg --nfs-path /tank/nfs
```

6.2 Connection: --cfg

Host table - one file for all servers, one server per line, `#` comments:

```
# host,user,password
192.168.2.48,root,secret
192.168.2.49,root,pass,with,commas # the password is everything after the 2nd comma
192.168.2.50,cert # ssh key of the running user (~/.ssh/id_ed25519, id_ecdsa,
id_rsa), user root
192.168.2.51,ops,cert # same key, other user
192.168.2.52,root,cert,/etc/keys/esx # explicit OpenSSH private key
```

`--host <ip>` selects the line (with only one line in the file it can be left out). `cert` entries work with ssh only (OpenSSH keys, no PuTTY `.ppk`); soap needs a password. Options that belong to one server are extra lines `<host>:<option>=<value>` in the same file: `proto`, `port`, `hostkey`, `tls_sha256`, `timeout`, `useragent`, for example `192.168.2.48:proto=ssh` or `192.168.2.48:hostkey=SHA256:abcd...`

Single-server key=value file - when a server needs more than login data:

```
host=192.168.2.48
user=root
password=... # or CS_ESXI_PASSWORD in the environment - never a command line flag
proto=auto # auto (soap, ssh if soap cannot be reached), ssh (vim-cmd) or soap (port
443)
hostkey=SHA256:... # ssh: pin the host key (the first run prints it in "warnings")
tls_sha256=... # soap: pin the certificate (sha256 hex)
key=/path/id_ed25519 # ssh: OpenSSH private key instead of a password
timeout=15 # connect timeout in seconds
```

The format is detected from the first entry line (`host,...` is a table, `key=...` a key=value file).

`CS_ESXI_HOST`, `CS_ESXI_USER` and `CS_ESXI_PASSWORD` override the file; `--host`, `--user` and `--proto` override both. Keep the file readable by the job user only. The freeze chains can live in the same file (napp-it CS: `_cfg/freeze4snap/freeze4snap.cfg`, described in `freeze4snap.readme` next to it).

	ssh	soap
free ESXi license	works (<code>vim-cmd</code>)	works: the free license accepts write calls only from a client whose User-Agent starts with "VMware" - the tool sends <code>VMware VI Client/4.0.0</code> (<code>useragent=</code> overrides it)
speed (16 VMs, measured)	about 12 s to list the VMs, about 3 s per snapshot	about 0.2 s to list the VMs, about 2 s per snapshot
needs	ssh service on the host	port 443

`--proto auto` (the default) tries soap first and falls back to ssh when the vSphere API cannot be reached (a warning says so). A failed soap *login* is final and not retried over ssh: the same credentials would fail again, and every failed login counts towards the account lockout of the ESXi host (5 by default). Entries with `cert` use ssh directly. The ESXi clock is not used (the tool names the snapshots itself), so a wrong host clock does no harm.

6.3 Freeze and thaw as separate steps

For jobs where the snapshot is taken by something else (a replication step, a script on the ZFS server):

```
cs-freeze4snap freeze --cfg esxi.cfg --dataset tank/nfs --name s1 --state /var/tmp/s1.json
zfs snapshot -r tank/nfs@s1 # by any means
cs-freeze4snap thaw --cfg esxi.cfg --state /var/tmp/s1.json
```

`freeze` writes the VM snapshot handles to the state file (mode 0600) and leaves the snapshots in place; `thaw` removes them and deletes the state file (it keeps the file and exits non-zero if a snapshot could not be removed; a snapshot that is already gone counts as removed). **Do not leave VM snapshots standing** - they grow. After a crash run `cleanup`:

```
cs-freeze4snap cleanup --cfg esxi.cfg --nfs-path /tank/nfs
```

6.4 Restoring (ESXi)

Restore the VM files from the ZFS snapshot (for example clone the snapshot and register or copy the VM folder, or `zfs rollback`), then revert the VM to the embedded snapshot on the ESXi host: `vim-cmd vmsvc/snapshot.revert <vmid> <snapid> 0`, or in the UI Snapshots > `cs4s-<snapshot>-<vmid>` > Restore.

7. Use from napp-it CS jobs

napp-it CS (snapshot and replication jobs) has one job setting **Freeze VM prior snap = off | proxmox | esxi**. `proxmox` runs the tool on the ZFS member (the Proxmox host); `esxi` runs it on the napp-it CS **frontend** and takes the ZFS snapshot on the member between freeze and thaw. ESXi hosts, logins, protocol and the freeze chains come from `_cfg/freeze4snap/freeze4snap.cfg` on the frontend (`freeze4snap.readme` in the same folder describes the file). A failed freeze never blocks the job unless a chain is strict; a missing tool or cfg is a hard error. For `proxmox` the jobs also make sure that `/etc/pve` is part of the `include=` of the job (added when missing) and pass `--pre-snap-cmd` and `memkeep=` to the tool.

The event-based real-time ZFS replication of cs-sync (Primary to Backup) calls cs-freeze4snap the same way for its daily application-consistent snapshot.

8. Howto setup (napp-it 4ai)

- Download the newest napp-it 4ai (menu **About > Frontend Update**).
- Get a key for 3 servers - free for non-commercial and home use (Evaluate & Home use on www.napp-it.org).
- Add the servers that take part to the frontend (server group, then select the server).

Proxmox: cs-freeze4snap must be present **on the Proxmox host** itself (**About > Download cs-tools** deploys it per member). `/etc/pve` becomes part of the replication snapshot automatically:

with *Freeze VM prior snap = proxmox* and a recursive job the folder is added to the include of the job if it is missing and re-synced right after the freeze, so the VM configs travel in the same snapshot as the consistent disk data. In contrast to the Proxmox backup tools the job works **per dataset**: the VMs that use the dataset are found automatically, but their disks have to be completely on it (a VM with a disk on another dataset is skipped and reported).

ESXi: cs-freeze4snap must be present on the **frontend machine (About > Download cs-tools)**.

Create the server file `/opt/csweb-gui/_cfg/freeze4snap/freeze4snap.cfg` (Windows frontend:

`C:\opt\csweb-gui\_cfg\freeze4snap\freeze4snap.cfg`), for example:

```
# ESXi servers, one per line: host,user,password or host,cert[,keyfile]
192.168.1.48,root,1234
```

Replication job: create a replication job for the filesystem that holds the VMs: source and destination are free (any to any), and the transfer is encrypted (cs-stream, the default of "Replication type"). Set *Freeze VM prior snap = proxmox* (tool on the Proxmox source member) or **esxi** (tool on this frontend, freezes the VMs on the source NFS filesystem). The built-in default chain is `freeze,memory,zfs`: try the freeze first (VMware Tools, QEMU guest agent); if that is not possible because the guest tools are missing, take the hot snapshot including the RAM, then the ZFS snapshot in any case - a chain that ends in `zfs` never blocks the job. Start the replication with any keep/hold retention you like, for example `hold 4s` (always keep the last 4 replications; `hold 8` = hold 8 days) or `keep hours:24,days:32,months:12,years:2` (with hourly snapshots that keeps the last 24, then the last 32 days, 12 months and 2 years).

9. Result format

```
{
  "dataset": "rpool/data",
  "snapshot": "20260809_1530",
  "status": "ok",
  "error": "...",
  "note": "...",
  "warnings": [...],
  "freeze_ms": 6,
  "guests": [
    {
      "vmid": 101,
      "type": "vm",
      "platform": "proxmox-qemu",
      "frozen": true,
      "strategy": "qmp-pause",
      "thawed": true,
      "warning": "...",
    }
  ]
}
```

ESXi mode adds `hypervisor` (`"esxi"`), `transport`, `nfs_path`, `datastores` (the matched ESXi datastore names), `skipped` (`vmid`, `name`, `reason`), `state` (freeze) and `removed` (cleanup); each `guests[]` entry has `platform`: `"esxi"`, `name`, `strategy` (`esxi-quiesce`, `esxi-mem`, `esxi-snap`) and `snap_id`. `CS_ESXI_DEBUG=1` prints every ssh command and soap call with its duration to stderr.

The exit code is 0 for `status: "ok"` and non-zero otherwise (also when `--forcefreeze` aborts the job). Only a failing `zfs snapshot` itself (permissions, name collision, pool full, ...) produces `status: "error"` - freeze problems never do.

10. Requirements, tests, license

- **ESXi mode** runs on any machine that reaches the ESXi host over ssh (port 22) or soap (port 443); the ZFS snapshot is taken locally or with `--zfs-cmd`. Static builds (`CGO_ENABLED=0`) exist for Linux (amd64, arm64), macOS (amd64, arm64), Windows, FreeBSD, illumos and Solaris (amd64).
- **Proxmox mode** runs on the Proxmox host (not inside a guest).
- Go 1.21 or newer to build; no runtime dependencies beyond golang.org/x/sys (for the `FIFREEZE` and `FITHAW` ioctls).

Tested: unit tests with in-process fakes (a fake ssh server, a fake vSphere SOAP endpoint that refuses write calls unless the User-Agent starts with "VMware", fake QGA and QMP), `go test ./... -race`. ESXi 8 with the free license and NFS datastores of a ZFS server: `discover`, `freeze/thaw`, `snap`, `cleanup` over ssh and soap, the `freeze`, `mem` and `plain` modes, wrong password and unreachable host (best effort and `--forcefreeze`); from napp-it CS replication jobs (Windows frontend, OmniOS ZFS server) two running VMs took 9-12 s for freeze, ZFS snapshot and thaw. Proxmox 9 with running and stopped QEMU VMs and an LXC container: `qga`, `qmp-pause`, `cgroup`, `stopped` and the error paths, a guest with disks on two pools flagged in `warnings`, `--forcefreeze` aborting without a snapshot, the `include=/etc/pve` combination verified inside the snapshot. Not tested: NFS 4.1 datastores, ESXi 6.x and 7.x, restoring a hot snapshot.

Note: ZFS does not support `FIFREEZE` on the tested kernel and OpenZFS (check with `fsfreeze -f <mountpoint>`), so the cgroup v2 fallback is the normal LXC path in practice. Do not assume either outcome on your own system without checking.

More: [_cfg/freeze4snap/freeze4snap.readme](#) in the frontend and howto.ai/cs-freeze4snap.info.
License: BSD 2-Clause (see LICENSE), the same as cs-sync and cs-stream.