

# cs-sync

## Handbuch

Echtzeit-Ordnersynchronisation (bidirektional und einseitig) mit ACL-  
Abgleich und ZFS-Replikation - für ZFS-Hosts

Version 3.3.0 · 14. Oktober 2026

# Inhalt

|                                                                           |          |
|---------------------------------------------------------------------------|----------|
| <b>1. Überblick</b>                                                       | <b>3</b> |
| <b>2. Die drei Modi</b>                                                   | <b>3</b> |
| 2.1 sync - bidirektional . . . . .                                        | 3        |
| 2.2 push - einseitig, Sie sind die Quelle . . . . .                       | 4        |
| 2.3 pull - einseitig, Sie sind das Ziel . . . . .                         | 4        |
| <b>3. serve - die Empfängerseite eines push</b>                           | <b>4</b> |
| <b>4. Endpoints und Authentifizierung</b>                                 | <b>4</b> |
| <b>5. ZFS-Replikation (--mode zfs, zfsreceive)</b>                        | <b>4</b> |
| 5.1 Proxmox: VM-/CT-Konfiguration und täglicher Freeze-Snapshot . . . . . | 5        |
| 5.2 Disaster Recovery einer VM (Schritt für Schritt) . . . . .            | 6        |
| <b>6. Installation</b>                                                    | <b>8</b> |
| 6.1 Mit napp-it CS (csweb-gui) . . . . .                                  | 8        |
| 6.2 Eigenständig . . . . .                                                | 8        |
| 6.3 Aus dem Quellcode bauen . . . . .                                     | 8        |
| <b>7. Tests</b>                                                           | <b>8</b> |
| <b>8. Bekannte Einschränkungen</b>                                        | <b>8</b> |
| <b>9. Lizenz und Gewährleistung</b>                                       | <b>9</b> |

# 1. Überblick

cs-sync ist ein Echtzeit-Dienst zur bidirektionalen und einseitigen Ordnersynchronisation mit Ordner-ACL-Abgleich für ZFS-Hosts. Er ist Teil von napp-it 4ai (Client-Server-Edition, csweb-gui) und kann auch vollständig eigenständig genutzt werden.

**Immer verschlüsselt.** Jede Replikation oder Synchronisation zwischen cs-sync-Hosts - ob über LAN, WAN oder Internet - wird Ende-zu-Ende verschlüsselt (ChaCha20-Poly1305), und zwar mit einem einmalig pro Sync-Paar vereinbarten gemeinsamen Transferschlüssel (--key). Es wird nichts im Klartext übertragen; ein Partner mit falschem Schlüssel wird beim Handshake abgewiesen. Ein VPN oder TLS-Tunnel ist nicht nötig.

Jede Beziehung verbindet zwei von drei Endpoint-Arten über eine einheitliche Syntax, die für alle Rollen gleich ist:

```
lokaler Pfad      D:\data, /tank/data, ./data, ../data
cs-sync-Host      host[:port]          (Standardport 9010)
RustFS-Bucket     host[:port]/bucket   (Standardport 9000)
```

Ein einziger String sagt cs-sync, ob ein Ordner überwacht, ein anderer cs-sync-Prozess angewählt oder ein S3-Bucket über rclone angesprochen wird. Version 3.0 war ein inkompatibler Neuentwurf: Es gibt keinen Upgrade-Pfad von 2.x, bestehende Dienste müssen neu angelegt werden.

## 2. Die drei Modi

Immer von links nach rechts: Primary, Secondary, Backup.

| Modus | Primary                                   | Secondary                                               | Backup                                                     |
|-------|-------------------------------------------|---------------------------------------------------------|------------------------------------------------------------|
| sync  | lokaler Pfad (Pflicht)                    | lokaler Pfad oder lokaler RustFS-Bucket (bidirektional) | optional, beliebiger Endpoint (zusätzliche Einweg-Strecke) |
| push  | lokaler Pfad (Pflicht, die Quelle)        | -                                                       | Pflicht, beliebiger Endpoint (das Ziel)                    |
| pull  | Pflicht, beliebiger Endpoint (die Quelle) | -                                                       | Pflicht, lokaler Pfad (das Ziel)                           |

```
sync:  Primary(lok)  <====> Secondary(lok)  [====> Backup(beliebig)]
push:  Primary(lok)  =====> Backup(beliebig)
pull:  Primary(beliebig) =====> Backup(lok)
```

### 2.1 sync - bidirektional

```
cs-sync run --mode sync --primary /tank/share --secondary /tank/mirror
```

Bidirektionale Echtzeit-Synchronisation mit Weitergabe der Ordner-ACLs (NFSv4-ACLs unter illumos/FreeBSD, POSIX-ACLs unter Linux). Beide Seiten müssen lokale Pfade auf demselben Host sein, weil die bidirektionale Synchronisation Änderungserkennung in Echtzeit auf beiden Seiten braucht.

Secondary kann auch ein lokaler RustFS/S3-Bucket sein - der wichtige Fall SMB zu S3:

```
cs-sync run --mode sync \
  --primary /tank/smbshare \
  --secondary localhost:9000/s3share \
  --key s3admin:yourSecretKey
```

Erlaubt ist nur localhost bzw. 127.0.0.1. cs-sync steuert rclone gegen den Bucket (bisync --resync für die erste Basis, danach Kopieren/Löschen je Objekt) und schreibt nie direkt in den Speicherpfad des Buckets.

Eine optionale einseitige Backup-Strecke legt eine dritte Kopie an, die einseitig vom Primary an einen beliebigen Endpoint geschoben wird:

```
cs-sync run --mode sync --primary /tank/share --secondary /tank/mirror \
```

```
--backup 192.168.1.50:9010 --key mySharedKey
```

## 2.2 push - einseitig, Sie sind die Quelle

```
cs-sync run --mode push --primary /tank/share --backup 192.168.1.50:9010 --key mySharedKey
cs-sync run --mode push --primary /tank/share --backup 192.168.1.50/bucket --key access:secret
cs-sync run --mode push --primary /tank/share --backup /mnt/otherdisk
```

Backup kann ein lokaler Pfad sein (einfacher einseitiger Spiegel), ein anderer cs-sync-Host (braucht einen passenden serve-Listener) oder ein RustFS/S3-Bucket, lokal oder entfernt. Nötig sind nur Adresse und Schlüssel.

## 2.3 pull - einseitig, Sie sind das Ziel

```
cs-sync run --mode pull --primary 192.168.1.10/bucket --backup /tank/restore --key access:secret
cs-sync run --mode pull --primary /tank/source --backup /tank/mirror
```

Primary ist die Quelle (ein RustFS-Bucket, lokal oder entfernt, oder ein zweiter lokaler Pfad); Backup ist immer lokal. Direktes Pullen von einem laufenden cs-sync-Host wird nicht unterstützt (siehe Bekannte Einschränkungen).

# 3. serve - die Empfängerseite eines push

Alles auf der Gegenseite eines push braucht einen eigenen Listener:

```
cs-sync serve --dest /tank/incoming --listen 0.0.0.0:9010 \
--key mySharedKey --allow_ip 192.168.1.20
```

--key und --allow\_ip sind beide Pflicht: Ein leerer Schlüssel oder eine fehlende Einzel-IP-Freigabe würde jedem Host im Netz einen Sync-Versuch erlauben. serve nimmt ausschließlich gepushte Daten an.

# 4. Endpoints und Authentifizierung

**--key / --primary\_key / --secondary\_key / --backup\_key.** Ein gemeinsamer Standard (--key) mit Überschreibung je Strecke, interpretiert nach Endpoint-Art:

- cs-sync-Endpoint: ein vorab vereinbarter Transfer- und Verschlüsselungsschlüssel. Das Wire-Protokoll ist nativ verschlüsselt (ChaCha20-Poly1305); der Schlüssel authentifiziert und verschlüsselt und muss zum --key von serve passen.
- RustFS-Endpoint: access:secret (am ersten Doppelpunkt getrennt). cs-sync legt das benannte rclone-Remote selbst nicht-interaktiv an und aktualisiert es - ein manueller rclone-config-Schritt ist nie nötig.

--allow\_ip beschränkt jeden eingehenden Listener eines Prozesses (den Empfänger von serve und das optionale --webhook-listen von run) auf eine einzige Quell-IP.

**Warum ein entfernter RustFS-Bucket nicht das Secondary von sync sein kann:**

Bidirektionale Synchronisation braucht eine symmetrische Änderungserkennung in Echtzeit auf beiden Seiten. Ein lokaler Bucket bekommt sie indirekt über die Dateisystemüberwachung des Primary; ein entfernter Bucket hat nur Webhooks oder periodische Neuscans, die langsamer und asymmetrisch sind. push und pull decken den Einweg-Fall ab, für den ein entfernter Bucket passt.

# 5. ZFS-Replikation (--mode zfs, zfsreceive)

Die Primary-Seite (run --mode zfs) ist Sender/Master und pusht die Daten; die Backup-Seite (zfsreceive) ist Empfänger/Slave: sie lauscht nur und übernimmt, was ankommt - es wird nichts gepullt. zfspull ist der frühere Name und bleibt als Alias gültig.

Ereignisbasierte Echtzeit-Replikation eines ZFS-Datasets, immer Primary nach Backup. Statt Dateien zu kopieren erkennt cs-sync Änderungen (written@snap) und sendet inkrementelle zfs send | zfs recv-Streams, fortsetzbar mit -s. Offene oder laufende VMs sind kein Problem: Der Snapshot ist

absturzkonsequent. Optional wird cs-freeze4snap für einen anwendungskonsistenten Tages-Snapshot aufgerufen.

# lokal: Primary und Backup auf demselben Host

```
cs-sync run --mode zfs --primary tank/data --backup tank2/data --nr 1 \
--keep hours:24,days:30 --versions y
```

# LAN: Sender am Primary, Empfänger am Backup

```
cs-sync zfsreceive --backup tank2/data --listen :9055 --key <key> --nr 1
```

```
cs-sync run --mode zfs --primary tank/data --backup-host 192.168.2.189:9055 --key <key> --nr 1
```

| Option                                | Bedeutung                                                                                                                                                                                                                                                              |
|---------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| --primary / --backup                  | Dataset-Namen (lokal: beide; LAN: Primary am Sender, Backup an zfsreceive)                                                                                                                                                                                             |
| --backup-host                         | Adresse des zfsreceive-Empfängers (LAN-Modus)                                                                                                                                                                                                                          |
| --nr                                  | Sync-Nummer, Teil der Snapshot-Namen<br>sync_<nr>_<fsguid>_<slot>                                                                                                                                                                                                      |
| --interval                            | Prüfintervall in Sekunden                                                                                                                                                                                                                                              |
| --keep hours:N,days:M                 | Aufbewahrung, von jeder Seite selbst angewendet                                                                                                                                                                                                                        |
| --versions y/n                        | stündliche/tägliche Versionen behalten (n: nur die letzten beiden Snapshots)                                                                                                                                                                                           |
| --rollback ask/auto                   | hat das Backup neuere Snapshots als die gemeinsame Basis: auf Freigabe im Frontend warten (ask) oder zurückrollen (auto)                                                                                                                                               |
| --recursive                           | untergeordnete Datasets und Zvols mit replizieren (-R)                                                                                                                                                                                                                 |
| --freeze, --freeze-bin, --freeze-args | Proxmox-Quelle: erster Snapshot des Tages automatisch über cs-freeze4snap (siehe 5.1), --freeze hh:mm nur für andere Quellen (ESXi); --freeze-bin nur, wenn cs-freeze4snap nicht neben cs-sync, im Tools-Ordner oder im PATH liegt; --freeze-args = Stufenkette/Policy |

Funktionsweise:

- Snapshot-Slots: new, n0, n1 (halten 2), kh<MMDDHH>, kd<MMDD>, kf<MMDD> (täglicher Freeze-Snapshot).
- Die gemeinsame Basis wird über die GUID gefunden. Der Primary fragt nie Backup-Snapshots ab; er erhält die Liste des Backup nur im Handshake.
- Die Absturzwiederherstellung leitet ihren Zustand aus ZFS selbst ab; unterbrochene Übertragungen werden mit send -t <token> fortgesetzt.
- Das Wire-Protokoll nutzt den verschlüsselten Handshake der Dateimodi (neue Frames 15-20, ProtoVersion unverändert).

Voraussetzung: OpenZFS (Linux, FreeBSD, illumos, Windows). Unter Windows muss der Prozess mit erhöhten Rechten laufen, weil zfs send | zfs recv-Pipes nicht neu gestartet werden können.

## 5.1 Proxmox: VM-/CT-Konfiguration und täglicher Freeze-Snapshot

Bei Proxmox gibt es zwei Besonderheiten.

**Erstens** liegt die VM-Konfiguration nicht im Dataset (wie bei ESXi), sondern in `/etc/pve`. Soll das in Echtzeit replizierte Dataset einer VM wiederhergestellt werden, ist die VM-Konfiguration nötig. Stellt cs-sync beim Start fest, dass die Quelle ein Proxmox-Host ist, liest es in jedem Zyklus die Konfigurationsdatei jeder VM/jedes Containers, deren Disk-Datasets (`vm-<id>-disk-N`, `subvol-<id>-disk-N`) zum Sync gehören (nicht das ganze `/etc/pve`), und hängt sie als ZFS-User-Property `cs:pveconf` an das Disk-Dataset und an jeden Sync-Snapshot. Die Properties werden über das cs-sync-Protokoll mit zum Backup übertragen (nicht im `zfs send`-Strom); eine reine Config-Änderung geht ohne neuen `zfs send` rüber.

**Zweitens** ist ein normaler Snapshot unter Proxmox nicht in allen Fällen konsistent, sondern wie ein Stromausfall zu sehen, je nach VM unkritisch bis heikel. Deshalb geht auf einer Proxmox-Quelle der erste Snapshot jedes Tages automatisch durch `cs-freeze4snap`, das die VMs des Quell-Datasets vor dem Snapshot einfriert (Guest Agent, Container-Freeze; die Stufenkette pro VM lässt sich mit `--freeze-args` einstellen). `cs-freeze4snap` wird im selben Ordner wie `cs-sync` gesucht (Standalone-Betrieb), sonst im `napp-it-Tools`-Ordner oder im `PATH`; fehlt es, wird der Freeze übersprungen (Warnung im Log) und der Snapshot bleibt absturzkonsistent. Der Snapshot wird mit `cs:consistent=app` (eingefroren) bzw. `paused` (VM nur angehalten, absturzkonsistent) markiert und als `kf<MMTT>` auf beiden Seiten für `--keep days` behalten. An Tagen ohne Datenänderung entsteht kein neuer Snapshot. So gibt es zumindest einmal täglich einen sicher wiederherstellbaren Stand.

Technische Einzelheiten zur Config:

Ein repliziertes Zvol allein ist noch keine wiederherstellbare VM: Hardware, Netzwerk, Boot-Reihenfolge und die Zuordnung der Platten stehen in `/etc/pve/qemu-server/<id>.conf` (Container: `/etc/pve/lxc/<id>.conf`). Stellt `cs-sync` fest, dass der Primary ein Proxmox-Host ist (`/etc/pve` existiert), liest der Sender immer die Config jeder VM/jedes CT, deren Disk-Datasets (`vm-<id>-disk-N`, `subvol-<id>-disk-N`, `base-<id>-disk-N`) zum Sync gehören, und speichert sie als ZFS-User-Property `cs:pveconf` (gzip + base64, bei Bedarf fortgesetzt in `cs:pveconf.1`, `.2` ..., weil ein Property-Wert auf etwa 8 KB begrenzt ist):

- am Disk-Dataset selbst: die **aktuelle** Config, auf beiden Seiten. Ändert sich nur die Config (RAM, NIC, neue Disk), wird nur diese kleine Property gesendet, kein `zfs send`.
- an jedem Sync-Snapshot (`n0`, `n1`, `kh..`, `kd..`, `kf..`): die Config **zum Zeitpunkt dieses Snapshots**. Zu jedem Disk-Stand gibt es die passende Config.

Die Backup-Seite kann ein beliebiges Betriebssystem sein (OmniOS, FreeBSD, Linux); gelesen wird mit `zfs get` und `cs-sync pveconf`. Mitgenommen wird nur die Config der VM, die zur Disk gehört, nie das ganze `/etc/pve` (storage.cfg, Benutzer, Firewall und Cluster-Dateien sind Host-Einstellungen und nicht Teil der Replikation). Die ID steckt im Wert, das Backup-Dataset darf deshalb beliebig heißen.

| Befehl                                                                                                                                   | Bedeutung                                                                                                                                                                                                                                                                                                            |
|------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>cs-sync pveconf list --dataset &lt;ds&gt;</code>                                                                                   | alle VM/CT-Configs unterhalb des Datasets (ID, Typ, Größe) und die Snapshots, die eine tragen                                                                                                                                                                                                                        |
| <code>cs-sync pveconf show --dataset &lt;disk-ds&gt; [--snapshot n0, kf1013 oder &lt;name&gt;]</code>                                    | Config ausgeben (aktuell oder aus einem Snapshot)                                                                                                                                                                                                                                                                    |
| <code>cs-sync pveconf restore --dataset &lt;disk-ds&gt; [--snapshot ..] --to &lt;Ordner&gt; [--map alt=neu] [--newid N] [--force]</code> | schreibt <code>&lt;id&gt;.conf</code> in <code>&lt;Ordner&gt;</code> ; <code>--map nvme=local-zfs</code> benennt den Storage um; <code>--newid 905</code> benennt auch die Disks <code>vm-105-disk-N</code> in <code>vm-905-disk-N</code> um; ohne <code>--force</code> wird eine vorhandene Datei nie überschrieben |

`--recursive y` mit einem Parent-Dataset behandelt alle Disks darunter. `cs-sync` schreibt nie von selbst nach `/etc/pve`; der Restore geht immer in einen Ordner, den Sie angeben.

## 5.2 Disaster Recovery einer VM (Schritt für Schritt)

Dieser Ablauf wurde live ausgeführt: Backup auf OmniOS (ohne Proxmox), VM-Daten zurück auf einen Proxmox-Node gesendet, Config mit neuer ID wiederhergestellt, VM gestartet, Guest Agent antwortete. Namen in spitzen Klammern sind Platzhalter.

**Ausgangslage.** Der Primary ist weg oder defekt. `<B>` ist das Backup-Dataset (z. B. `tank/dr/vm-105-disk-0`) mit den Snapshots `sync_<nr>_<guid>_n0 / n1 / kh<MMDDHH> / kd<MMDD> / kf<MMDD>`. `<T>` ist der Proxmox-Node, auf dem die VM wieder laufen soll, `<S>` sein ZFS-Storage (siehe `/etc/pve/storage.cfg`), `<P>` der Pool hinter `<S>`.

**1. Backup einfrieren.** Den Empfänger-Dienst dieser VM auf dem Backup-Host stoppen (Services > Realtime Sync), damit während der Wiederherstellung kein Snapshot rotiert wird. Am Backup-Dataset nie `zfs rollback`, `zfs destroy` von `sync_`-Snapshots oder `zfs set readonly=off`

ausführen; die Wiederherstellung arbeitet immer auf einer Kopie.

**2. VM finden.** `cs-sync pveconf list --dataset <B-oder-Parent>` zeigt `id=<id>`, Typ (`qemu` oder `lxc`) und die Snapshots mit Config. Keine Ausgabe ("no cs:pveconf found"): die Quelle war kein Proxmox-Host (kein `/etc/pve`); die Daten lassen sich trotzdem wiederherstellen (Schritt 4), die Config muss dann von Hand neu angelegt werden.

**3. Snapshot wählen.** `zfs list -t snapshot -d 1 -o name,creation,cs:consistent <B>. kf<MMDD>` mit `cs:consistent=app` ist der anwendungskonsistente Tages-Snapshot (Guest-Agent-Freeze); `n0` ist der neueste Stand (absturzkonsistent, wie nach einem Stromausfall). Ohne besonderen Grund `n0`; startet der Gast nicht sauber, mit `kf` wiederholen.

**4. Daten nach <T> bringen**, als Dataset mit genau dem Namen aus der Config (`vm-<id>-disk-N`, bzw. der neuen ID):

- Backup-Pool liegt auf <T> selbst: `zfs send <B>@<snap> | zfs recv -u <P>/vm-<id>-disk-0`
- anderer Host mit ssh: `zfs send <B>@<snap> | ssh <T> zfs recv -u <P>/vm-<id>-disk-0`
- kein ssh zwischen den Hosts: cs-sync rückwärts. Auf <T>: `cs-sync zfsreceive --backup <P>/vm-<id>-disk-0 --listen :9070 --key <k> --allow_ip <Backup-Host> --nr 9`; auf dem Backup-Host: `cs-sync run --mode zfs --primary <B> --backup-host <T>:9070 --key <k> --nr 9`. Zeigt das Log `ok`, beide stoppen.

Für jede Disk der VM wiederholen (die Config nennt sie: `scsi0`, `virtio0`, `efidisk0`, `tpmstate0`, `rootfs`). Danach auf <T>: `zfs set readonly=off <P>/vm-<id>-disk-N` und, beim cs-sync-Weg, `zfs inherit cs:sync <P>/vm-<id>-disk-N`.

**5. Config wiederherstellen** auf dem Host, der die Properties hat (Backup-Host): `cs-sync pveconf restore --dataset <B> --snapshot <slot> --to /root/restore --map <alter-storage>=<S> [--newid <N>]` (Container genauso, die Datei gehört nach `/etc/pve/lxc`). `--newid` verwenden, wenn die alte ID auf <T> oder im Cluster noch existiert. Datei nach <T> kopieren: `/etc/pve/qemu-server/<id>.conf`. Existiert `<id>.conf` schon, verweigert der Befehl (kein `--force`, außer die vorhandene VM soll wirklich ersetzt werden).

**6. Config anpassen.** `ide2: ...:vm-<id>-cloudinit` wird nicht repliziert (eigenes Dataset): mit `qm set <id> --delete ide2` entfernen oder mit `qm set <id> --ide2 <S>:cloudinit` neu anlegen. Läuft das Original eventuell noch oder wurde die ID geändert: neue MAC (`qm set <id> --net0 virtio=<neue-mac>,bridge=vmbro0`) und zuerst `link_down=1`, um doppelte IPs zu vermeiden. Disks auf anderen Storages als dem synchronisierten Dataset sind nicht im Backup und müssen getrennt wiederhergestellt werden.

**7. Vor dem Start prüfen.** `qm config <id>` muss ohne Fehler ausgeben; `pvesm path <S>:vm-<id>-disk-0` muss ein Device liefern; `qm status <id>` meldet `stopped`.

**8. Starten und prüfen.** `qm start <id>`, dann `qm agent <id> ping` (antwortet, wenn der Guest Agent läuft) und `qm guest exec <id> -- uptime`. Anwendungsdaten prüfen. Ein absturzkonsistenter Snapshot löst beim ersten Boot `fsck/journal-Replay` aus.

**9. Zurück in den Normalbetrieb.** `link_down` entfernen, die VM unter einen neuen Sync stellen (neuer Primary -> neues Backup, neue `--nr`, erster Vollsend). Das alte Backup-Dataset erst löschen, wenn der neue Sync durchgelaufen ist.

Container: Schritte 1 bis 5 wie oben (`--to /etc/pve/lxc`, Dataset `subvol-<id>-disk-0`), Prüfung mit `pct config <id>` und `pct start <id>`; die Subvol-Datasets müssen dieselben `mountpoint/canmount`-Werte haben wie ein gesundes Subvol auf <T> (Vergleich mit `zfs get canmount,mountpoint`). Der Container-Weg wurde für die Config (Properties und Restore) getestet, aber nicht für einen kompletten Neustart.

**Regeln für Automatisierung (KI oder Skript):** am Backup nie etwas löschen oder zurückrollen; jeder Schritt hat eine Prüfung (Ausgabe von `list`, `zfs list`, `qm config`); beim ersten Fehler anhalten und ihn melden; nie `--force` für eine ID, die in Benutzung ist; eine wiederhergestellte Kopie nie mit aktivem Netzwerk starten, solange das Original noch laufen könnte.

## 6. Installation

Fertige Binaries hängen an jedem GitHub-Release als `cs-sync-<os>.<arch>.tar.gz`: `mswin.amd64`, `linux.amd64`, `linux.arm64`, `illumos.amd64`, `solaris.amd64`, `freebsd.amd64`, `darwin.amd64`, `darwin.arm64`.

### 6.1 Mit napp-it CS (csweb-gui)

csweb-gui verteilt und aktualisiert das Binary je Mitglied unter About > Download cs-tools. Das Menü Realtime Sync (03\_System > 02\_Services > 25\_Realtime\_Sync) erwartet es an einem festen Pfad je Plattform:

```
data/cs_server/tools/cs-sync/<osdir>.<arch>/cs-sync[.exe]
```

`<osdir>` ist `mswin`, `linux`, `illumos`, `solaris`, `freebsd` oder `darwin`. Auf dem Windows-Frontend ist das `C:\opt\csweb-gui\data\cs_server\tools\cs-sync\mswin.amd64\cs-sync.exe`; danach angelegte Dienste finden es ohne Neustart.

### 6.2 Eigenständig

Das Archiv der Plattform entpacken, `cs-sync` an einen beliebigen Ort legen und wie oben gezeigt starten. Für jeden RustFS-Endpoint muss `rcclone` installiert sein und im `PATH` liegen; `cs-sync` ruft es als Programm auf und bringt keinen eigenen S3-Client mit.

### 6.3 Aus dem Quellcode bauen

Benötigt Go 1.25 oder neuer.

```
git clone https://github.com/guenther-alka/cs-sync.git
cd cs-sync
go build -o cs-sync .
go build -o cs-sync.exe -ldflags "-X main.version=dev" .
GOOS=illumos GOARCH=amd64 go build -o cs-sync .
```

In PowerShell `$env:GOOS` und `$env:GOARCH` als getrennte Zuweisungen vor der Build-Zeile setzen.

## 7. Tests

```
go build ./...
go vet ./...
go test ./...
```

Die Unit-Tests decken den Endpoint-Parser (u.a. Windows-Laufwerkspfade gegenüber `host:port`), das RustFS-Paket (Webhook-Ereignisse, `--allow_ip`-Filter, `xl.meta`-Parsing) und den Remote-Empfänger ab (Wire-Pfade, die aus `--dest` ausbrechen würden, werden abgewiesen). Eine mitgelieferte Integrationssuite gegen ein echtes RustFS oder einen zweiten `cs-sync`-Host gibt es nicht; diese Fälle wurden manuell geprüft.

## 8. Bekannte Einschränkungen

- Pullen von einem laufenden `cs-sync`-Host wird nicht unterstützt: Das Wire-Protokoll hat keinen Frame "schick mir deinen Baum". `run --mode pull` weist einen `cs-sync`-Primary mit klarer Fehlermeldung ab. Stattdessen einen `serve`-Listener und ein `push` verwenden (das `napp-it-CS`-Menü Realtime Sync macht das automatisch).
- Die Backup-Strecke von `sync` kann kein lokaler Pfad sein; dafür eine zweite, unabhängige `push`-Beziehung anlegen.
- Keine ACL-Weitergabe zu oder von einem RustFS-Endpoint; S3 kennt keine Dateisystem-ACLs.
- Kein rekursives Löschen nicht leerer Ordner auf der RustFS-Seite (dokumentiertes No-Op), weil Ordner dort nur eine Namenskonvention über Objektschlüsseln sind.

- RustFS mit selbstsigniertem Zertifikat: cs-sync übergibt rclone immer `--no-check-certificate` (nicht konfigurierbar), weil rclone sonst beim TLS-Handshake hängt. Ungültige oder abgelaufene Zertifikate werden dadurch ebenfalls akzeptiert - cs-sync ist für vertrauenswürdige LANs gedacht, nicht für feindliche Netze.

## 9. Lizenz und Gewährleistung

BSD 2-Clause. Es gibt keine Gewährleistung. Dies ist Infrastruktursoftware, die im Normalbetrieb Dateien löscht und überschreibt. Die Lösch-Budget-Schutzoptionen (`--max-delete-count`, `--max-delete-percent`) lesen, zuerst mit Nicht-Produktivdaten testen und unabhängige Backups behalten.