

cs-s3gateway

Manual

Read-only S3 gateway and web page for ZFS snapshots and ordinary folders - without a data format of its own

Version 0.3.1 · October 13, 2026

Contents

1. Overview	3
2. Quick start	3
3. How folders map to buckets	3
4. Credentials and certificates	3
5. Options and config file	4
6. Web page	4
7. Clients	5
7.1 WinSCP (tested)	5
7.2 rclone	5
7.3 aws cli and boto3	5
8. Supported S3 operations	6
9. Security model	6
10. napp-it CS integration	6
11. RustFS data	7
12. Build, test, limits	7
13. License	7

1. Overview

cs-s3gateway exports folders as S3 buckets **without any data format of its own**: the object key is the relative path, and the files stay ordinary files (SMB, `ls` and backup tools keep seeing them as before). Its main use is the snapshot directory of a ZFS dataset: **every snapshot of a dataset becomes a read-only top level folder in an S3 bucket**, reachable from WinSCP, rclone, restic, aws cli, boto3 - or from any browser through the built-in web page.

```
cs-s3gateway --snaps s3-storage-snaps=/pool/dataset --bucket disaster-recovery=/pool/dr
```

`s3-storage-snaps/daily_2026-10-07/Projekte/x.docx` is the file
`/pool/dataset/.zfs/snapshot/daily_2026-10-07/Projekte/x.docx`.

Why a separate tool? RustFS and most S3 servers keep their own object layout and have no read-only bucket or "snapshot as bucket" option. A small separate service that only knows how to *read* is easier to audit than a write-capable server, can be started on demand (with an idle timeout) and has no way to modify data.

- single static binary, Go standard library only, no dependencies
- Windows, Linux (amd64, arm64), macOS (amd64, arm64), FreeBSD, illumos, Solaris
- re-uses the credentials and the TLS certificate of RustFS in napp-it CS - nothing new to configure
- menu **System > Services > cs-s3gateway** in napp-it CS: start, stop, autostart, idle timeout, buckets

2. Quick start

Inside napp-it CS (credentials and PEM files already exist from RustFS):

```
cs-s3gateway --snaps s3-storage-snaps=/tank/s3_storage --listen :9100 --idle-timeout 30m
```

Standalone, without TLS, with explicit credentials (testing):

```
set CS_S3GW_ACCESS_KEY=s3admin
set CS_S3GW_SECRET_KEY=0123456789abcdefghijklmnopqrstuvwxyz
cs-s3gateway --no-tls --listen 127.0.0.1:9100 --bucket docs=D:\share\docs
```

Open <http://127.0.0.1:9100/> in a browser, or point an S3 client to <http://127.0.0.1:9100> (path style).

3. How folders map to buckets

Option	Exported folder	Bucket content
<code>--bucket name=/folder</code>	<code>/folder</code>	the folder itself; key = relative path
<code>--snaps name=/pool/dataset</code>	<code>/pool/dataset/.zfs/snapshot</code>	one top level folder per snapshot

- Both options are repeatable - several buckets on several folders are fine.
- **Bucket names** are 3-63 characters of `a-z 0-9 -` and start and end with a letter or digit. **Object keys** are arbitrary UTF-8 (umlauts, spaces, CJK ...); they are simply the file path with `/` as separator.
- `.zfs` belongs to **one dataset**. Snapshots of child datasets are in their own `.zfs`: give every dataset you want to reach its own `--snaps` bucket. The path is the *mounted* filesystem (`zfs get mountpoint`), on Windows for example `D:/s3_storage`.
- Only regular files and folders are visible. Symlinks, fifos, devices and sockets do not exist for the gateway. Names that cannot be addressed safely (backslash, control characters, invalid UTF-8) are hidden. In `--bucket` a top level `.zfs` is hidden.
- ZFS auto-mounts snapshots on first access, so the first request into a snapshot can be slower.

4. Credentials and certificates

By default they are taken from napp-it CS exactly as the RustFS start script does:

- access key **s3admin**, secret = the first 20 bytes of the first line of `<base>/_cfg/server.auth` (the RustFS login), or an own **session secret**: `--creds-from <file> --access-key s3session` with a file that holds a random secret only this gateway knows. The napp-it CS menu creates it on demand (button *new secret*, setting *Login*), stores it in `_cfg/s3gw/s3gw.secret` and shows it in the login window. A new secret revokes the old one and restarts the service. The service table lists the access key and a *Secret* column for both logins; secrets are never printed in the table, *show* opens them in a popup.
- certificate and key = `<base>/_cfg/s3/pem/rustfs_cert.pem` and `rustfs_key.pem`

So an S3 client that works with RustFS works with the gateway using the same keys - only the port differs (RustFS 9000/9001, gateway 9100).

Setting	Default	Override
napp-it CS folder	<code>/opt/csweb-gui</code> (C:\opt\csweb-gui on Windows)	<code>--base</code>
secret file	<code><base>/_cfg/server.auth</code>	<code>--creds-from file</code>
access key	<code>s3admin</code>	<code>--access-key</code> , env <code>CS_S3GW_ACCESS_KEY</code>
secret	from the secret file	env <code>CS_S3GW_SECRET_KEY</code> (keeps it out of the process list)
certificate / key	<code><base>/_cfg/s3/pem/...</code>	<code>--cert</code> , <code>--key</code>

5. Options and config file

Option	Meaning
<code>--bucket name=folder</code>	export a folder, repeatable
<code>--snaps name=mountpoint</code>	export <code><mountpoint>/_zfs/snapshot</code> , repeatable
<code>--rust name=folder, --rustsnaps name=mountpoint</code>	RustFS data folders, see chapter 11
<code>--listen :9100</code>	listen address (default <code>:9100</code>)
<code>--idle-timeout 30m</code>	exit after that time without requests (0 = run until stopped)
<code>--log file</code>	append log lines to a file as well as stdout
<code>--max-keys 1000</code>	maximum keys per list response
<code>--no-tls</code>	plain HTTP (testing, or behind a TLS proxy)
<code>--base, --creds-from, --access-key, --cert, --key</code>	credentials and certificate, see chapter 4
<code>--conf file</code>	read options from a file
<code>--version</code>	print the version

`--conf file` reads `flag=value` lines (`#` starts a comment); options on the command line win over the file. napp-it CS writes this file from its menu. The value of `snaps=` and `bucket=` is `name=path`, split at the first `=`. Secrets are never stored in this file.

```
# s3gw.conf
listen=:9100
idle-timeout=30m
snaps=s3-storage-snaps=/tank/s3_storage
bucket=disaster-recovery=/tank/dr
```

6. Web page

The gateway has a small built-in read-only web page for use "on the road" from any browser or phone. Open the gateway address (`https://host:9100/`, it redirects to `/_browse/`) and log in with the same access key and secret. You get the bucket list, folder listings (500 entries per page) and for each file:

- a click on the name shows `png jpeg gif webp pdf txt` directly in the browser, everything else is a download;
- the **Download** link always forces a download (`?dl=1, Content-Disposition: attachment`).

HTML, SVG and all other active content types are **never rendered**; every file response carries `Content-Security-Policy: sandbox` and `X-Content-Type-Options: nosniff`, and pages are `no-store`. The page uses HTTP Basic authentication, so **use it over HTTPS** (the default); the failed-login limiter of the S3 side applies as well. The namespace `/_browse` cannot collide with a bucket, because bucket names cannot contain `_`.

RustFS' own web console cannot show a foreign S3 endpoint, which is why the gateway brings its own page.

7. Clients

Only path style addressing is supported (`https://host:9100/bucket/key`), region `us-east-1`. For a self-signed certificate import it or disable verification in the client.

7.1 WinSCP (tested)

File protocol *Amazon S3*, host name and port 9100, access key `s3admin` plus secret. Under Advanced > Environment > S3 set URL style to **Path** (required: WinSCP defaults to virtual-host style, which the gateway does not support). For a self-signed certificate untick *Verify TLS certificate*.

7.2 rclone

```
[snaps]
type = s3
provider = Other
endpoint = https://host:9100
access_key_id = s3admin
secret_access_key = <secret>
force_path_style = true
region = us-east-1
no_check_bucket = true

rclone ls snaps:s3-storage-snaps/daily_2026-10-07
rclone copy snaps:s3-storage-snaps/daily_2026-10-07/Projekte /restore
```

7.3 aws cli and boto3

```
aws s3 ls s3://s3-storage-snaps/ --endpoint-url https://host:9100 --no-verify-ssl
aws s3 cp s3://s3-storage-snaps/daily_2026-10-07/x.docx . --endpoint-url https://host:9100 --no-verify-ssl

import boto3
from botocore.config import Config
s3 = boto3.client("s3", endpoint_url="https://host:9100", aws_access_key_id="s3admin",
                  aws_secret_access_key="...", region_name="us-east-1", verify=False,
                  config=Config(s3={"addressing_style": "path"}))
for o in s3.list_objects_v2(Bucket="s3-storage-snaps", Delimiter="/").get("CommonPrefixes", []):
    print(o["Prefix"])
```

Presigned URLs (`aws s3 presign, generate_presigned_url`) work for GET and HEAD.

8. Supported S3 operations

Operation	Status
ListBuckets	yes
ListObjects (v1) / ListObjectsV2	yes - prefix, delimiter, max-keys, continuation, <code>encoding-type=url</code>
GetObject / HeadObject	yes - <code>Range</code> and conditional headers (<code>If-Modified-Since</code> , ...)
GetBucketLocation, GetBucketVersioning	yes (always <code>us-east-1</code> , versioning not enabled)
presigned GET/HEAD (SigV4)	yes
other bucket sub-resources (ACL, policy, tagging, ...)	<code>501 NotImplemented</code>
PUT, POST, DELETE, multipart upload, copy	<code>AccessDenied</code> - "cs-s3gateway is read-only"

ETags are derived from size and modification time, not from an MD5 of the content. That is sufficient for caching and for sync tools that compare size and time; tools that insist on MD5 ETags can use their `--size-only` style options.

9. Security model

- **Read only.** Only GET, HEAD and list are implemented; every other request is answered with `AccessDenied`. No code path writes to the exported folders, and ZFS snapshots are immutable on top of that.
- **Folder jail.** All file access goes through Go's `os.Root`: neither `..` nor a symlink can leave the exported folder. Only regular files and folders are opened, so a fifo or device can never block the server.
- **AWS Signature V4** for header and presigned requests, constant time comparison, 15 minutes clock skew allowed.
- **Brute force limiter.** 10 failed logins within a minute block the client address for a minute (S3 and web page).
- **TLS** (minimum 1.2) with the RustFS certificate; plain HTTP only with `--no-tls`.
- **Web page hardening.** Basic auth over TLS, strict CSP, `nosniff`, `sandbox` on file responses, HTML-escaped names, active content types download-only, request paths quoted in log lines.
- **Least privilege.** Run the service as an unprivileged user that can read the exported folders and the `_cfg` files and nothing else. The secret is not logged and not stored in the config file.
- **Audited.** Findings, fixes and known limitations are in SECURITY.md. Buckets that would expose the napp-it `_cfg` folder (server.auth, TLS key) are refused; `.zfs` stays hidden under every alias on Windows and macOS.
- **Idle timeout.** `--idle-timeout` makes the service end itself, so an on-demand service is not an open port all day.
- **Exposure.** The gateway is meant for a trusted LAN or VPN, or for use behind a reverse proxy. If you publish it to the internet, use HTTPS and a firewall or IP restriction as for any admin interface - it uses the `admin` credentials of RustFS (no per-user accounts yet).

10. napp-it CS integration

napp-it CS contains the menu **System > Services > cs-s3gateway**. It

- starts and stops the service and switches **autostart** on and off (started again with `server.pl` and at boot),
- sets the port, the **idle timeout** (never, 15 min ... 1 day) and HTTPS yes/no,
- manages the bucket list (`snap:name=/pool/dataset`, `dir:name=/folder`; an empty list is pre-filled with the snapshots of the RustFS `s3_storage` filesystem),
- shows the endpoint, a link to the web page and a **login** popup with access key and secret,

- writes `_cfg/s3gw/s3gw.conf` and keeps the pid file and the log next to it.

The binary is expected in `_my/tools/cs-s3gateway/<os>.amd64/`; it is distributed with the menu **About** > **Download cs-tools**.

11. RustFS data

RustFS keeps its objects in its own format (a folder per object with `xl.meta` plus part files), not as plain files. Two bucket types read this format directly and present the objects as normal files:

```
cs-s3gateway --rust name=/s3_storage           # RustFS data folder: <rustfs-bucket>/<object>
cs-s3gateway --rustsnaps name=/tank/s3_storage # <dataset>/zfs/snapshot/<snapshot>/<rustfs-bucket>/<object>
```

In the napp-it CS menu use the bucket line types `rust:name=path` and `rustsnap:name=path`, for example `rustsnap:s3-storage-snaps=d:/s3_storage`.

Supported: single disk RustFS (`xl.meta` v2, inline data, multipart objects), nested keys, the latest version of versioned objects (delete markers hide the object), real ETag and content type from the metadata. The internal folders (`.rustfs.sys`, `xl.meta`, part files) are never visible or addressable.

Not supported (listed, but GET answers 501 with a reason): compressed or encrypted objects, tiered objects, and erasure coded data spread over several disks. The bitrot checksums of the part files are skipped, not verified. Older versions of an object are not served.

12. Build, test, limits

Go 1.26 is used (the folder jail needs `os.Root`, Go 1.24 or newer).

```
go vet ./... && go test ./...
powershell -NoProfile -File build-all.ps1 -Test
```

`build-all.ps1` produces 8 static variants (`CGO_ENABLED=0, -trimpath -s -w`) in `dist\release` plus `checksums.sha256`. The tests cover the documented AWS Signature V4 examples (header and presigned), a complete list/get/range/presign round trip including Unicode names, paging and the folder jail, the failed-login limiter, bucket name validation and the web page (login, HTML escaping, download headers, traversal attempts).

Limits: read only by design - no uploads, no delete, no multipart, no versioning, no per-user accounts or bucket policies; one set of credentials (the admin keys of RustFS); path style addressing only (no `bucket.host` virtual hosts). Tested with the Go test suite, boto3, WinSCP (URL style Path) and the browser page on Windows; symlink handling and the ZFS snapshot auto-mount behavior on Linux and illumos still need field tests.

13. License

BSD 2-Clause - see the file LICENSE; changes are listed in CHANGELOG.md.