

cs-stream

Manual

Encrypted TCP stream transport for ZFS replication and folder sync -
replaces ssh, mbuffer, netcat and pv

Version 2.1.0 · October 13, 2026

Contents

- 1. Overview** 3
- 2. Usage** 3
 - 2.1 Options 3
 - 2.2 Tunnel mode (rclone over SFTP) 3
- 3. Encryption and security** 4
 - 3.1 Security audit v2.1.0 4
- 4. Installation** 4
 - 4.1 In napp-it CS 4
 - 4.2 Standalone 5
- 5. Build** 5
- 6. Warranty and license** 5

1. Overview

cs-stream is an encrypted TCP stream transport. In ZFS send/receive pipelines it replaces ssh, mbuffer, netcat (nc) and pv with one small tool: an encrypted pipe with a read-ahead buffer, a rate limit and a progress display.

It is part of napp-it CS, where the jobs *job-replicate* (encrypted ZFS replication) and *job-filesync* (rclone-over-SFTP folder sync between members) use it. It has no dependency on a napp-it installation and can be used standalone as a generic encrypted stream transport.

The tool was called *zstream* until v1.3.3. As of v2.0.0 it is called cs-stream (to avoid a clash with the unrelated OpenZFS tool of the same name); wire protocol and command line are unchanged.

Mode	Purpose
<code>listen</code> / <code>send</code>	pipe stdin to stdout like an encrypted netcat - for <code>zfs send</code> / <code>zfs receive</code>
<code>tunnel-listen</code> / <code>tunnel-send</code>	wrap a local TCP service (for example the SFTP server of rclone) in an encrypted tunnel

2. Usage

The receiver is always started first: it opens the port and waits (up to the accept timeout of 120 s). A sender with nothing listening at the other end simply fails to connect.

```
# Receiver (start first) - --allow-ip is REQUIRED as of v2.1.0
cs-stream listen 9000 SESSIONKEY --allow-ip=192.168.1.20 | zfs receive -F tank/backup

# Sender
zfs send tank@snap | cs-stream send 192.168.1.10 9000 SESSIONKEY

# With options (rate limit, progress, log)
zfs send tank@snap | cs-stream send 192.168.1.10 9000 SESSIONKEY --rate=100m --progress
--log=/tmp/cs-stream.log
```

2.1 Options

Option	Default	Description
<code>--buf=SIZE</code>	128m	read-ahead buffer (0 = off), units k/m/g
<code>--rate=SPEED</code>	off	throughput limit per second, units k/m/g (not in tunnel mode)
<code>--progress</code>	off	live progress on stderr (not in tunnel mode)
<code>--log=FILE</code>	off	append a transfer summary to a file
<code>--bind=IP</code>	0.0.0.0	bind the listener to one IP address
<code>--allow-ip=IP</code>	none	required on <code>listen</code> and <code>tunnel-listen</code> : only this source IP may connect

2.2 Tunnel mode (rclone over SFTP)

The tunnel wraps a local child process. `--local=ADDR` is the local address the wrapped process binds to (`tunnel-listen`) or connects to (`tunnel-send`); `--allow-ip=IP` is the only source IP from which `tunnel-listen` accepts sessions. Both are required on `tunnel-listen`.

```
# Receiver (start first)
cs-stream tunnel-listen 9000 KEY --local=127.0.0.1:2222 --allow-ip=192.168.1.10 -- rclone serve sftp
/dst --addr 127.0.0.1:2222

# Sender
cs-stream tunnel-send HOST 9000 KEY --local=127.0.0.1:2222 -- rclone sync /src :sftp: --sftp-
host=127.0.0.1 --sftp-port=2222
```

3. Encryption and security

- AES-256-GCM authenticated encryption (Go standard library, AES-NI accelerated).
- A session key per transfer; in napp-it CS it is generated by job-replicate.pl and job-filesync.pl.
- Every 64 KB chunk is encrypted on its own with a random nonce; the authentication tag detects any tampering or corruption.
- `--allow-ip` restricts which source IP may even attempt a handshake.
- Accept timeout 120 s (the listener never waits forever); read timeout 5 minutes (detects a dead sender or a network failure).
- `--bind=IP` restricts the listener to one interface.

3.1 Security audit v2.1.0

A full source audit found and fixed three points:

1. `listen` and `tunnel-listen` accepted connections from any source. `listen` accepts exactly one connection per process lifetime, so an unauthorized host could occupy that single slot and block the real sender (an availability problem only - AES-256-GCM already prevented any read or write without the key). Fix: `--allow-ip` is mandatory; wrong IPs are rejected and closed, the listener keeps waiting, and the accept deadline keeps running so a flood of wrong connections cannot extend the window. This is a breaking change against v2.0.0.
2. `tunnel-listen` had no limit for concurrent sessions: now at most 64 (`maxTunnelSessions`), counted atomically, as defense in depth even for the allowed IP.
3. `--rate` and `--progress` were silently ignored in tunnel mode; a one-time warning is now logged.

Three live tests were run on Windows, Linux, FreeBSD, macOS and OmniOS/illumos, all passed: (A) `listen/tunnel-listen` refuse to start without `--allow-ip`, (B) a connection from a wrong IP is refused and logged while the listener keeps running, (C) a connection from the allowed IP works end to end and decrypts correctly.

Reviewed and not considered actionable: the random 96-bit nonces under a static key (a collision risk only becomes relevant around 2^{32} chunks, about 256 TB under one key - a key is used for one transfer only), and the key derivation by plain SHA-256 (the key material is always a freshly generated high-entropy string from the Perl caller, never a user-chosen password).

4. Installation

Platform	File
macOS x86_64 / ARM64	<code>cs-stream-darwin-amd64</code> , <code>cs-stream-darwin-arm64</code>
Linux x86_64 / ARM64	<code>cs-stream-linux-amd64</code> , <code>cs-stream-linux-arm64</code>
Windows x86_64	<code>cs-stream-windows-amd64.exe</code>
FreeBSD x86_64	<code>cs-stream-freebsd-amd64</code>
illumos x86_64	<code>cs-stream-illumos-amd64</code>
OmniOS / Solaris x86_64	<code>cs-stream-solaris-amd64</code>

The pre-built binaries are attached to the GitHub releases.

4.1 In napp-it CS

The binary lives in `data/cs_server/tools/cs-stream/<os>.amd64/cs-stream` (Linux, BSD, OmniOS, Solaris) or `data\cs_server\tools\cs-stream\mswin.amd64\cs-stream.exe` (Windows). napp-it CS deploys and maintains it on the members through the menu **About > Download cs-tools**; manual placement is only needed to seed the admin host itself with a new version.

4.2 Standalone

cs-stream is a self-contained single-file Go binary with no reference to napp-it at all. No config file, registry entry or environment is needed - only the matching session key on both ends and, as of v2.1.0, `--allow-ip` on the receiving side.

```
git clone https://github.com/guenther-alka/cs-stream.git
cd cs-stream
go build -o cs-stream .

# receiver
cs-stream listen 9000 SESSIONKEY --allow-ip=192.168.1.20 > out.bin
# sender
cs-stream send 192.168.1.10 9000 SESSIONKEY < in.bin
```

5. Build

Go 1.21 or newer, no external dependencies.

```
G00S=linux GOARCH=amd64 go build -o cs-stream-linux-amd64 .
G00S=windows GOARCH=amd64 go build -o cs-stream-windows-amd64.exe .
G00S=freebsd GOARCH=amd64 go build -o cs-stream-freebsd-amd64 .
G00S=solaris GOARCH=amd64 go build -o cs-stream-solaris-amd64 .
```

6. Warranty and license

cs-stream is open source. You may use, analyze or modify it free of charge, "as is", and you bear sole responsibility for its use. This note summarizes the license terms in plain language and does not replace them.

BSD 2-Clause License - Copyright (c) 2026 Guenther Alka / napp-it.org. Redistribution in source or binary form requires retaining the copyright notice; see the file LICENSE.