

cs-send

Manual

Mail (SMTP/TLS) and chat alerts (Discord, Slack, Telegram, ntfy, Gotify) from the command line - one static binary

Version 0.3.0 · October 13, 2026

Contents

1. Overview	3
2. Installation	3
3. Sending mail	3
3.1 TLS mode follows the port	3
3.2 Gmail, Outlook and other providers	3
3.3 Inline images and attachments	4
3.4 Log file	4
4. Sending chat alerts	4
4.1 Not included	4
5. Use in napp-it CS	5
6. Build from source	5
7. Tests	5
8. License and warranty	5

1. Overview

cs-send sends email (SMTP over TLS, HTML body with inline images and regular attachments) and chat alerts (Discord, Telegram, Slack, ntfy, Gotify) from the command line. It is one static binary built from the Go standard library only - there are no external dependencies at all.

cs-send is part of the napp-it CS toolset, where it is wired into the report job infrastructure to deliver status and error messages. It is fully usable standalone, for example from cron or from your own scripts.

KISS: one binary, two sub-commands (**mail** and **chat**), no configuration file. Everything is given as command line options.

2. Installation

Prebuilt binaries for every platform are attached to each GitHub release

(**cs-send-*<os>.<arch>.tar.gz***):

Platform	Archive
Windows	mswin.amd64
Linux	linux.amd64, linux.arm64
illumos / OmniOS	illumos.amd64
Solaris	solaris.amd64
FreeBSD	freebsd.amd64
macOS	darwin.amd64, darwin.arm64

Extract and run - there is no install step and no runtime dependency beyond the TLS/CA trust store of the operating system.

In napp-it CS the binary is distributed to the members with the menu **About > Download cs-tools**; it is not downloaded automatically.

3. Sending mail

```
cs-send mail \  
  --to alice@example.com,bob@example.com \  
  --subject "Backup finished" \  
  --html-file report.html \  
  --attach chart.png \  
  --smtp-host smtp.gmail.com --smtp-port 587 \  
  --user you@gmail.com --key <app-password>
```

3.1 TLS mode follows the port

The TLS mode is derived from the port, there is no separate flag: port **465** means implicit TLS from the first byte (the "SSL" port of most providers). Any other port connects in plain text and then attempts STARTTLS; **587** is the near-universal STARTTLS submission port.

3.2 Gmail, Outlook and other providers

"Less secure app access" no longer exists at Google. You need a Google account with 2-Step Verification enabled and an App Password generated for it (<https://myaccount.google.com/apppasswords>). Use the app password as **--key**, not your normal account password. Outlook/Office365 and generic SMTP relays work the same way, with whatever password or app-password scheme the provider requires.

3.3 Inline images and attachments

Inline images are embedded in the HTML body (``) instead of being a downloadable attachment:

```
cs-send mail --to a@b.com --subject "Status" \
--html '<html><body><h1>All good</h1></body></html>' \
--inline chart.png=chart \
--smtp-host smtp.example.com --smtp-port 587 --user u --key p
```

`--inline path=cid` can be repeated or comma separated (`--inline a.png=img1,b.png=img2`). `--attach path` adds regular, downloadable attachments the same way. `cs-send mail` without further arguments prints the complete flag reference.

3.4 Log file

`--logfile <path>` (for `mail` and `chat`) appends one timestamped result line per run. This is useful when `cs-send` is started by cron or by a napp-it CS job, where stdout and stderr are not kept.

The log never contains credentials: SMTP passwords (`--key`), chat webhook URLs and bot tokens are redacted. A webhook URL grants send access to everybody who knows it, so it is treated like a password.

4. Sending chat alerts

A chat alert is one outbound HTTPS request per provider - no OAuth flow, no app review, only a webhook URL or a bot token you already have.

```
cs-send chat --provider discord --url <webhook-url> --text "Deploy finished"
cs-send chat --provider slack --url <webhook-url> --text "Deploy finished"
cs-send chat --provider ntfy --url https://ntfy.sh/my-topic --text "Disk 90% full" --priority 5
cs-send chat --provider gotify --url https://gotify.example.com --token <app-token> --text "Scrub
done"
cs-send chat --provider telegram --token <bot-token> --chat-id <id> --text "Backup failed"
```

Provider	What you need	Setup
Discord	a webhook URL	Server Settings > Integrations > Webhooks > New Webhook > copy URL
Slack	a webhook URL	classic Incoming Webhook (still works for existing workspaces)
ntfy	a topic URL	pick any topic name on ntfy.sh or self-host; no account needed for a public topic
Gotify	a server URL and an app token	self-hosted; create an app in the Gotify web UI to get the token
Telegram	a bot token and a chat ID	message @BotFather, <code>/newbot</code> , copy the token; message @userinfobot to find your chat ID

`--title` and `--priority` are honored by ntfy and Gotify, which have a native concept of both. The other three providers have no separate title field, so a `--title` is prepended to `--text` as a bold line.

4.1 Not included

Deliberately left out for now: WhatsApp (the Business API is gated behind Meta verification - disproportionate for an alert tool), Signal (no official webhook API, a separate `signal-cli` daemon would be needed), Microsoft Teams (classic incoming webhooks are being retired in favor of Power Automate workflows - much more setup) and Matrix (doable with an access token and a REST PUT, but more setup than the five providers above).

5. Use in napp-it CS

The report jobs of napp-it CS call cs-send to deliver status and error messages by mail or chat. The SMTP and chat settings are entered in the job menus; cs-send itself keeps no state and no configuration.

For long-lived credentials prefer environment variables or a wrapper script over a bare command line option - this keeps secrets out of the shell history of a shared machine.

6. Build from source

```
git clone https://github.com/guenther-alka/cs-send.git
cd cs-send
go build -o cs-send .
GOOS=linux GOARCH=amd64 go build -o cs-send .
```

7. Tests

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

The unit tests cover the MIME message builder (plain text only, text plus HTML alternative ordering, inline-image `Content-ID` wiring, regular attachments and the invariant that Bcc never appears in the headers - verified by parsing the built message back with `net/mail` and `mime/multipart`) and every chat provider against a local `httptest.Server`, including the error body on a non-2xx response. The live SMTP conversation of `mail.Send` is not covered by an automated test (it needs a real server); it was exercised manually against Gmail and a local relay.

8. License and warranty

BSD 2-Clause - see the file LICENSE in the repository.

There is no warranty. Verify recipients and attachments before you send anything that matters, and treat SMTP credentials and chat tokens like any other secret: do not commit them and do not pass them on the command line of a shared machine if that is a concern.