sweeppea CLI Cheat Sheet

Run your whole sweepstakes platform from the terminal.

INSTALL
`npx @sweeppea/cli`

PLATFORMS
macOS · Linux · Windows

DOCS
apidocs.sweeppea.com

Getting started

```
npx @sweeppea/cli auth login     # store your API key (hidden prompt, validated first)
npm install -g @sweeppea/cli     # or install the `sweeppea` command globally
sweeppea account health          # verify credentials and connectivity
sweeppea <category> --help       # every command documents itself
```

GLOBAL FLAGS

`--json`	Raw JSON instead of tables. Works on every command.
`-y, --yes`	Skip the confirmation on destructive actions.
`--api-base <URL>`	Point at another environment (testing).
`-h, --help`	Help for any command or subcommand.

WHERE YOUR API KEY LIVES

`SWEEPPEA_API_KEY`	Env var. Wins over everything — use it in CI.
`OS keyring`	Keychain on macOS, Credential Manager on Windows.
`File, mode 0600`	Fallback on Linux. Directory is left at 0700.

The key is never echoed, never logged, and redacts itself in errors.

The core

SWEEPSTAKES

`list`	List them. Archived hidden unless --all.
`create`	New sweepstakes. Needs --name --type --handler and the date/time range.
`update <token>`	Change only the fields you pass.
`clone <handler>`	Copy into a new one with --new-handler.
`pause / unpause <token>`	Stop and resume entries.
`delete <token>`	Permanent. Confirms first.

`--type` is `sms`, `email` or `social`. `--handler` is A-Z, 0-9 and underscore, max 20 — validated locally before anything hits the network.

WINNERS

`draw <token>`	Draw now with --winners N. Irreversible; confirms.
`list <token>`	Winners already drawn. --all walks every page.
`schedule <token>`	Future drawing: --date --time --timezone --winners.
`scheduled <token>`	Pending scheduled drawings.
`unschedule <tok> <sch>`	Cancel a pending drawing. Confirms.

`draw` also takes `--group`, `--completed-entries`, `--include-opted-out` and `--exclude-spam`.

PARTICIPANTS

`add <token>`	--email or --phone, plus --field "K=V" (repeatable) and --bonus.
`get <token>`	One entrant by --token, --email or --phone.
`list <token>`	20 per page. --all fetches every page.
`count <token>`	Totals split into entrants, AMOE and opt-outs.
`bonus <tok> <prt> <n>`	Overwrite bonus entries.
`delete <tok> <prt>`	Permanent. Confirms first.

GROUPS

`list <token>`	The prize groups of a sweepstakes.
`create <tok> <name>`	New group. The name must be unique.
`update <tok> <grp> <name>`	Rename it.
`delete <tok> <grp>`	Permanent. Confirms first.

ACCOUNT

`health`	Credentials and connectivity.
`profile`	Your user profile.
`business`	Business information.
`plan`	Active subscription plan.

AUTH

`login`	Hidden prompt. Validated before it is stored.
`logout`	Remove the stored credential.
`status`	Are you logged in, and is the key still valid.

Everything else

FILES — ACCOUNT DRIVE

`list`	Files and storage usage.
`upload <path>`	Max 10 MB. Private by default; --public to share.
`url <file>`	Short-lived presigned URL. --mode, --expires.
`send <file> <email>`	Email it as an attachment (max 5 MB).
`delete <file>`	Permanent. Confirms first.

RULES — OFFICIAL RULES

`list <token>`	The rules documents of a sweepstakes.
`create <token>`	--title plus --content or --content-file.
`update <tok> <rul>`	Also --abbreviated-shopify.
`delete <tok> <rul>`	Permanent. Confirms first.

NOTES — ACCOUNT LEVEL

`list / get <note>`	All notes, or one by token.
`create`	--title --note, and --pinned to pin it.
`update <note>`	--unpin to unpin.
`delete <note>`	Permanent. Confirms first.

CALENDAR

`list / get <event>`	All events, or one by token.
`create`	--title --start-date --end-date; also --location --color --all-day.
`update <event>`	Only the fields you pass change.
`delete <event>`	Permanent. Confirms first.

BILLING — READ ONLY

`wallet`	Wallet transactions.
`transactions`	Billing transactions.
`consumptions`	Monthly and yearly totals.
`datatransfer <token>`	Data-transfer records for a sweepstakes.

TICKETS — SUPPORT

`list`	Open by default. --closed, --search, --priority, --platform.
`get <case>`	Full detail by case number.
`create`	--title --description --priority.
`update <case>`	Only open tickets.
`resolve <case>`	Close it.
`delete <case>`	Permanent. Confirms first.

TODOS — ADMIN ONLY

`list`	--status, --priority, --resource, --pinned.
`create`	--title --priority; also --deadline and --pin.
`update <todo>`	--complete, --incomplete, --completion 0-100.
`delete <todo>`	Permanent. Confirms first.

A sweepstakes, start to finish

```
# 1. Create it. Tokens come back as bare 36-character UUIDs, never a prefix.
SWEEP=$(sweeppea sweepstakes create --name "Summer Giveaway" --type email \
  --handler SUMMER2026 --start-date 2026-08-01 --end-date 2026-08-31 \
  --start-time 09:00 --end-time 17:00 --json | jq -r '.Data.SweepstakesToken')

# 2. Add an entrant. The API requires both contact fields, not just one.
sweeppea participants add $SWEEP --email amelia@example.com --phone 3055550142 \
  --field "First Name=Amelia" --field "City=Miami" --bonus 3

# 3. One prize group per prize. Keep the token: --group needs it, not the name.
GROUP=$(sweeppea groups create $SWEEP "Grand Prize" --json | jq -r '.Data.GroupToken')

# 4. Draw, then keep the result. `draw` confirms first; --yes skips that in CI.
sweeppea winners draw $SWEEP --winners 3 --group $GROUP
sweeppea winners list $SWEEP --json > winners.json
```

Worth remembering

`YYYY-MM-DD`	Every date flag. Times are HH:MM, 24-hour.
`--timezone <n>`	A number, not a name. Only on winners schedule.
`<token> vs <handler>`	Almost everything takes the token; clone takes the handler.
`--all`	Walks every page. Without it: 20 entrants, 10 winners.
`10 MB`	Upload limit. Emailed attachments cap at 5 MB.
`--expires <secs>`	Presigned URLs: 60 to 3600 seconds.
`429`	Retried with backoff automatically. Do not parallelise.
`502`	Re-checked before the operation is reported as failed.

Destructive commands

ELEVEN COMMANDS CANNOT BE UNDONE

`winners draw` . `sweepstakes delete` . `participants delete` . `winners unschedule` . `groups delete` . `notes delete` . `calendar delete` . `rules delete` . `files delete` . `tickets delete` . `todos delete`

All of them behave the same way. With a terminal in front of them they print what will happen and wait for `y` — a stray Enter aborts. With no terminal and no `--yes` they refuse outright, so a script never hangs waiting for an answer and never deletes on a newline lost in a pipe. In CI, pass `--yes` when you actually mean it.

Automation

Every command accepts `--json`, which is what turns the CLI into a building block rather than a tool you sit in front of. In CI there is no login step: export `SWEEPPEA_API_KEY` as a repository secret and every command picks it up.

```
# Export the winners of a sweepstakes to CSV
sweeppea winners list $SWEEP --json \
  | jq -r '.Winners[] | [.KeyEmail, .GroupName] | @csv' > winners.csv

# Count entrants across every active sweepstakes
sweeppea sweepstakes list --json \
  | jq -r '.Data[].SweepstakesToken' \
  | xargs -n1 sweeppea participants count

# In CI: no login step, just the secret
export SWEEPPEA_API_KEY="$SWEEPPEA_KEY"
sweeppea winners draw "$SWEEP" --winners 1 --yes
```

WHY A CLI

Sweeppea already had two doors onto its API: the dashboard, for doing things by hand, and the MCP server, for asking an agent to do them in conversation. Both need somebody present at every step.

But what a campaign actually demands is repetitive and scheduled — load the entrants from an activation, publish the official rules, draw at an exact time, export for legal. That scales neither by clicking nor by chatting.

The CLI is the door that runs unattended. A cron job or a CI pipeline does exactly what is written, identically every time, with nobody watching, and returns JSON and exit codes so it chains into what you already use. It also gives agents a floor to stand on: one that drives the CLI stops guessing at API calls and runs real commands, with human confirmation on anything irreversible.

BUILT TO BE SAFE

Zero unsafe code, enforced by the compiler. HTTPS only, with the URL validated by parsing rather than string splitting. Local validation before anything hits the network. Automatic retry with backoff, and a re-check after a gateway error before reporting failure.

npmjs.com/package/@sweeppea/cli · apidocs.sweeppea.com · mcpdocs.sweeppea.com Every command documents itself with `--help`