edays-cli — Installation and Usage Manual

Version: 0.5.0
Date: 2026-10-16
Platform: Windows 10/11 · macOS 13+ · Linux (Ubuntu 22.04+)

Table of Contents

1. Introduction
   - 1.1 How it works
   - 1.2 Using edays-cli with an AI assistant
   - 1.3 Sensitive data protection
2. Prerequisites & System Requirements
   - Installing Python
   - Installing uv
3. Installation
   - Installing the browser
   - If the `edays` command is not found
   - Windows: installing from the Claude desktop app
4. First-Run Configuration
   - 4.1 Set the instance URL
   - 4.2 Sign in
   - 4.3 Choose your absence types
   - 4.4 Add your team roster
   - 4.5 Testing the setup
5. Sessions & Authentication
6. Command Reference
   - Global flags
   - install-browser
   - login
   - whoami
   - entitlement
   - team
   - team sync
   - team find
   - team add
   - team absences
   - absence-types
   - read

- book
- cancel

7. Output Format & Exit Codes
8. Config File Reference
   - `[privacy]` — sensitive absences
9. Troubleshooting
10. Upgrading & Uninstalling

---

1. Introduction

edays-cli is a command-line client for the **eDays** leave and absence management system. It lets you work with absences from the terminal — without opening the eDays web application.

With edays-cli you can:

- **Read** the calendar — booked absences, public holidays — for a month or any date range, for yourself or a team member
- **See who is off**: one command shows the whole team's absences for a day, a week or a range, as JSON or a person × day grid
- **Book** an absence (Holiday, Home Office, or any other type you configure), with a server-side precheck and a confirmation step
- **Cancel** an absence record
- **Check** remaining holiday and yearly entitlement for yourself, a team member or the whole team
- **Check** your profile and whether your session is still valid

All commands output clean JSON by default, or formatted tables with the `--pretty` flag. This makes edays-cli easy to use interactively, in scripts, and from AI agents.

edays-cli is **instance-agnostic**: the URL of your organisation's eDays instance, the absence type IDs and your team roster are kept in your local config file. Nothing organisation-specific is built into the tool.

> **Unofficial tool.** edays-cli is not affiliated with or endorsed by eDays. It uses the same internal API as the eDays web application, which may change without notice.

1.1 How it works

eDays offers a documented public API, but it is usually not enabled for ordinary employee accounts. edays-cli therefore talks to the same internal API that the eDays web application uses in your browser:

1. `edays login` opens a real Chromium window on your eDays instance. You sign in with your organisation's normal single sign-on (e.g. Microsoft Entra ID, Okta) — including MFA if your organisation requires it.
2. Once eDays is loaded, edays-cli captures the session cookies and the anti-forgery token into a local session file.
3. Every other command replays that session over HTTPS. When the session expires (typically after a few hours), commands report an auth error and you run `edays login` again.

edays-cli never sees or stores your password.

1.2 Using edays-cli with an AI assistant

Because every command returns a stable JSON envelope (see Section 7) and supports non-interactive flags (`--yes`, `--dry-run`), edays-cli works well under an AI coding assistant that can run shell commands. If you write instructions (a "skill" or rules file) for your assistant, include at least these rules:

- **Never book or cancel without the user's explicit confirmation.** Show the exact command first; prefer `--dry-run` to preview a booking.
- **The assistant cannot sign in for you.** On exit code `2` (`EdaysAuthError`) it should ask you to run `edays login` yourself.
- **Global flags go before the subcommand** — `edays --pretty read`, not `edays read --pretty`.
- **Parse `data` from the JSON output**; human-readable status goes to stderr.
- **Leave sensitive data alone.** Never pass `--show-sensitive`, never change `[privacy]` in the config, and never try to work out sick leave another way (gaps in the calendar, raw API calls). For sick-leave questions, point the user to the eDays web app. See 1.3.

To make sure an assistant cannot quietly switch the protection off, deny it write access to the config file. In Claude Code, add to `~/.claude/settings.json`:

```
{"permissions": {"deny": ["Edit(~/.edays-cli/config.toml)",
                          "Write(~/.edays-cli/config.toml)"]}}
```

edays-cli itself still updates the file (`login`, `team sync` / `add`, `absence-types --add`), so the assistant needs no direct edits.

1.3 Sensitive data protection

New in 0.5.0. Sick leave is health data. edays-cli is often run by an AI assistant, so by default it **keeps sick leave out of every output** — yours and your team's:

- `read` and `team absences` leave sensitive absences out (or, if you choose, show them only as *Unavailable*, without type or detail). The weekly team overview and any other script built on these commands get the filtered data too.
- `absence-types` marks sensitive types with `sensitive: true` and never reports whether they were *used recently*; `--add` refuses to map them.
- `entitlement` and `book` refuse sensitive types (`SensitiveDataError`) — use the eDays web app for those.
- `whoami` drops sickness figures (e.g. sick days, Bradford factor) from the eDays profile summary.
- eDays error messages that mention a sensitive type are replaced by a neutral text.

Which types count as sensitive is configurable. Out of the box any absence whose name contains *sick*, *illness* or *ill health* (case-insensitive) is sensitive; you can add more names or type IDs in `[privacy]`. With the default `hide` mode a person who is on sick leave simply shows as **not absent** — keep that in mind when planning with `team absences`.

To see sensitive absences yourself, use the eDays web app, or add `--show-sensitive` in a terminal (see Global flags). It works only in an interactive terminal after you confirm a prompt, so a script or an AI tool call cannot use it.

---

2. Prerequisites & System Requirements

Requirement	Minimum	How to check
Python	3.11 or newer	`python --version` or `python3 --version`
Package installer	`uv` (recommended) or `pip`	`uv --version` or `pip --version`
eDays account	Access to your organisation's eDays instance in a browser	Open it in your normal browser
Disk space	~300 MB for the Chromium download	—
Display	A desktop session (for the one-time SSO browser window)	—

`edays login` needs a graphical desktop because you complete SSO in a visible browser window. All other commands work in any terminal.

Installing Python

Windows: download from python.org and run the installer.

> **Important:** on the first installer screen, check **"Add Python to PATH"**.
> Alternatively, install Python from the **Microsoft Store**, which adds it to PATH automatically.

macOS:

```
brew install python@3.12
```

Linux (Ubuntu/Debian):

```
sudo apt install python3 python3-pip
```

Verify in a new terminal: `python --version` (Windows) or `python3 --version`.

Installing uv

Windows (PowerShell):

```
powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"
```

macOS / Linux:

```
curl -LsSf https://astral.sh/uv/install.sh | sh
```

Close and reopen your terminal, then verify: `uv --version`. Official guide: https://docs.astral.sh/uv/getting-started/installation/.

---

3. Installation

Option A — via uv (recommended)

```
uv tool install edays-cli
```

uv installs edays-cli into an isolated environment and puts the `edays` command on your PATH.

Option B — via pip

```
pip install edays-cli
```

Option C — from the GitHub repository

```
uv tool install git+https://github.com/xdoko01/edays-cli
```

Installing the browser

`edays login` drives a Chromium browser through Playwright. Download it once:

```
edays install-browser
```

This runs `playwright install chromium` inside edays-cli's own environment, so the browser version always matches the installed Playwright library. If the download fails (for example a DNS error while fetching FFmpeg), just run the command again — these errors are usually transient. The browser is stored in Playwright's shared cache (`%LOCALAPPDATA%\ms-playwright` on Windows, `~/Library/Caches/ms-playwright` on macOS, `~/.cache/ms-playwright` on Linux).

> On Linux you may also need the browser's system libraries: `sudo $(dirname $(readlink -f $(which edays)))/python -m playwright install-deps chromium`.

Verifying the installation

```
edays --version
```

Expected output:

```
edays, version 0.4.0
```

If the `edays` command is not found

The tool's install directory is not on your `PATH` yet. With uv, run:

```
uv tool update-shell
```

then close and reopen the terminal. If that does not help:

- **Windows:** add `%USERPROFILE%\.local\bin` to your User **Path** environment variable (Win → "Environment Variables" → User variables → Path → Edit → New), then reopen the terminal.
- **macOS / Linux:** add `export PATH="$HOME/.local/bin:$PATH"` to `~/.zshrc` or `~/.bashrc` and `source` it.
- **pip on Windows:** find the scripts directory with `python -m site --user-scripts` and add it to your Path.

Windows: installing from the Claude desktop app

The Claude desktop app on Windows runs in an MSIX container that **virtualizes** `AppData`. When Claude runs `uv tool install edays-cli` for you, the tool's environment lands in the app's private copy of `AppData\Roaming\uv\tools`, but the `edays.exe` launcher in `%USERPROFILE%\.local\bin` is shared with the rest of Windows. The two halves then don't match:

- In a normal terminal: `uv trampoline failed to canonicalize script path`.
- After an upgrade run from inside Claude, the other way round: `ModuleNotFoundError: No module named 'edays_cli'`.

Fix — before installing, put uv's tool directory outside `AppData` with a user environment variable, so Claude and normal terminals share one install:

```
setx UV_TOOL_DIR "%USERPROFILE%\.uv-tools"
```

(or Win → "Environment Variables" → User variables → New: `UV_TOOL_DIR` = `%USERPROFILE%\.uv-tools`). Restart the Claude desktop app and open a new terminal so both see the variable.

Recovery if you already hit one of the errors above:

1. Uninstall on **both** sides — once in a normal terminal and once by asking Claude to run it: `uv tool uninstall edays-cli`.
2. Set `UV_TOOL_DIR` as above and restart Claude and the terminal.
3. Install once: `uv tool install edays-cli`, then `edays install-browser`.
4. Check from both a normal terminal and Claude: `edays --version`.

Your config and session in `~/.edays-cli` are not affected.

---

4. First-Run Configuration

4.1 Set the instance URL

Run any command — for example `edays read`. On the first run edays-cli writes an empty config file and reports that `base_url` is not set, including the file's path:

```
{"ok": false, "command": "read", "error": {"type": "ConfigError",
 "message": "base_url is not set. Edit C:\\Users\\you\\.edays-cli\\config.toml and set [edays] base
```

The config file lives in your home folder:

```
C:\Users\<you>\.edays-cli\config.toml
```

> **macOS / Linux:** the config file is at `~/.edays-cli/config.toml`.

Open the file and set `base_url` to the address you use for eDays in your browser, without a trailing path. The generated file explains each section in comments; feel free to add your own — edays-cli keeps comments when it updates the file:

```
[edays]
base_url = "https://yourcompany.e-days.com"
my_user_id = ""    # filled in by `edays login`
session_path = ""  # blank = ~/.edays-cli/edays_state.json

[absence_types]

[roster]
```

4.2 Sign in

```
edays login
```

A Chromium window opens on your eDays instance. Complete your organisation's sign-in. When eDays has loaded, the window closes by itself and the command prints who you are. On the first login edays-cli also fills in `my_user_id` in the config file for you.

4.3 Choose your absence types

Each eDays instance has its own numeric IDs for absence types (Holiday, Home Office, Sick, ...). `edays book` uses short names — *kinds* — that you map to those IDs in `[absence_types]`. List the types you can book:

```
edays --pretty absence-types
```

```
 id   Name                       Configured kind
──────────────────────────────────────────────────
123   Holiday
456   Home Office
789   Sickness
 …
```

Then map the kinds you want, by ID or by name:

```
edays absence-types --add holiday=123 --add "home-office=Home Office"
```

This writes the mappings to the config:

```
[absence_types]
holiday = 123
home-office = 456
```

These keys become the `KIND` argument of `edays book` (`edays book holiday …`). Use lowercase names without spaces for convenience.

> **Name matching:** an exact (case-insensitive) name wins; otherwise the text must match exactly one type name. With similar names such as "Holiday" and "Holiday (CZ)", give the full name or the ID.

Which one is the right type? eDays may list ~70 types, including duplicates (two "Home Office" entries) and types you are not actually permitted to book. Check them:

```
edays --pretty absence-types --usable
```

```
 id   Name           Configured kind   Usable          Unit    Used recently
──────────────────────────────────────────────────────────────────────────────
456   Home Office    home-office       yes             days    yes
457   Home Office                      not permitted
123   Holiday (CZ)   holiday           yes             hours   yes
```

Pick the type that is **usable** and, ideally, **used recently** (you or your colleagues already book it). `--only-usable` hides the rest. When an `--add` name is ambiguous, the error lists the same hints for each candidate.

4.4 Add your team roster

The roster lets you use `--for NAME` to read or book on behalf of team members (subject to your permissions in eDays — typically managers and authorisers). It maps a display name to that person's eDays user ID (a GUID). You don't need to look the IDs up yourself:

```
edays --pretty team sync            # add all your direct reports
edays --pretty team find novak      # search people visible to you
edays team add "Jan Novak"          # add one person
edays team add novak --as Jan       #  ... under a shorter roster name
```

The result in the config:

```
[roster]
"Alice Smith" = "1b2c3d4e-0000-0000-0000-000000000001"
"Bob Brown"   = "1b2c3d4e-0000-0000-0000-000000000002"
```

eDays decides who is visible to you — usually your reportees if you are a manager. Ordinary employees may see nobody, which is fine: the roster is optional, and without it every command works for yourself (`--for me` , the default). You can also edit the `[roster]` section by hand, e.g. to rename people.

If the commands above find nobody you need, look the ID up manually: open the person's calendar or profile in the eDays web app with the browser developer tools (**F12**) → **Network** tab open, and find the GUID in request paths such as `/api30/calendar/<userId>/<month>/<year>`.

4.5 Testing the setup

```
edays --pretty whoami
edays --pretty read
edays book holiday 2026-12-23 --dry-run
```

`--dry-run` runs the eDays precheck without booking anything. If all three succeed, you are ready to go.

---

5. Sessions & Authentication

Item	Location (default)
Session file	`~/.edays-cli/edays_state.json` (Windows: `C:\Users\<you>\.edays-cli\edays_state.json`)
Browser profile	`~/.edays-cli/browser-profile/` (next to the session file)

- The **session file** contains the eDays session cookies and anti-forgery token. Treat it like a password: anyone who has it can act as you in eDays until it expires. Do not commit, share or sync it.
- The **browser profile** is persistent, so your identity provider can remember the device. Subsequent logins are often quicker and may not ask for MFA again.
- Sessions **expire after a few hours** (set by your eDays instance). An expired session makes commands exit with code `2` and an `EdaysAuthError`; run `edays login` again.
- To use a different session location, set `session_path` in the config.
- To sign out completely, delete the session file and the `browser-profile` folder.

---

6. Command Reference

Global flags

Global flags go immediately after `edays`, **before** the subcommand:

```
edays --pretty read --month 2026-11      # correct
edays read --month 2026-11 --pretty      # error: No such option '--pretty'
```

Flag	Short	Meaning
`--version`		Show the installed version and exit
`--pretty`	`-p`	Human-readable tables/text on stdout instead of JSON
`--quiet`	`-q`	Suppress informational messages on stderr
`--config PATH`		Use a different config file instead of the default
`--show-sensitive`		Include sensitive absences such as sick leave in this one call. Interactive terminal only; asks for confirmation (1.3)
`--help`		Show help (also works after any subcommand)

`--show-sensitive` fails with `SensitiveDataError` when stdin or stderr is not a terminal (a pipe, a script, an AI tool call) or when you answer *no*. Note that running it from inside an AI assistant's chat, e.g. with Claude Code's `!` prefix, still puts the output into the conversation.

`edays install-browser`

Downloads the Chromium build that Playwright needs for `edays login`. Run once after installing, and again if `login` reports that Chromium is missing (e.g. after an upgrade that bumps the Playwright version).

```
edays install-browser
```

`edays login`

Opens a browser window on your eDays instance, waits until you complete SSO, captures the session and verifies it.

```
edays login
edays login --timeout 600
```

Option	Default	Meaning
`--timeout SECONDS`	`300`	How long to wait for the sign-in to complete

If the browser window does not appear in front, use Alt+Tab. On success the output includes your user record and the session file path.

`edays whoami`

Shows your eDays profile and your remaining holiday. Because it calls the API, it doubles as a **session check** — exit code `0` means the session is valid.

```
edays --pretty whoami
```

```
Jane Doe
userId: 1b2c3d4e-…
holiday: 12.5 days (100 h) remaining of 25 days, year ends 2026-12-31
```

The holiday line needs a `holiday` kind in `[absence_types]` (see 4.3); without it `whoami` prints a hint on stderr and `entitlement` is `null`.

JSON `data`: `{"user": {…}, "summary": {…}, "entitlement": {"kind", "typeId", "remainingDays", "remainingHours", "entitlementDays", "entitlementHours", "yearEnd", "potName", "other"}}` — the same fields as `edays entitlement`.
If only the entitlement lookup fails, `entitlement` holds an `error` object and the command still succeeds. Fields of the eDays profile summary that describe sickness (e.g. sick days, Bradford factor) are removed from `summary` (1.3).

`edays entitlement`

Shows remaining and yearly entitlement for an absence kind (default `holiday`), in days and hours, for you, a team member or the whole team.

```
edays --pretty entitlement                  # yourself
edays --pretty entitlement --for alice      # a roster member
edays --pretty entitlement --team           # you + everyone in the roster
edays entitlement --kind home-office
```

Option	Default	Meaning
`--for NAME`	`me`	`me`, or a roster name (case-insensitive substring)
`--team`	off	You plus every roster member (cannot be combined with `--for`)
`--kind KIND`	`holiday`	Configured kind from `[absence_types]`

```
                Entitlement — holiday
Name          Remaining days   Remaining h   Entitlement days   Year end
──────────────────────────────────────────────────────────────────────────
me                      12.5           100                 25   2026-12-31
Alice Smith                5            40                 25   2026-12-31
```

- **Remaining** already deducts holiday that is booked but not yet taken — the same number the eDays web app shows.
- For other people you need the eDays rights to read their entitlement — in practice, being their manager. eDays answers HTTP 404 "*You don't have permission*" otherwise; that person's row shows **no permission** (JSON: `error` with `status: 404`) and the others are still listed. Note that you may well see someone's calendar (`team absences`) but not their entitlement.
- eDays may report extra entitlement "pots" (e.g. carry-over); ones edays-cli doesn't recognise are passed through raw in `other`.
- A sensitive `--kind` (by kind name, type ID or the type's name in eDays) is refused with `SensitiveDataError`, for example a sick-leave balance.

JSON `data`:


```
{
  "kind": "holiday", "typeId": 123,
  "people": [{"name": "me", "userId": "…", "remainingDays": 12.5, "remainingHours": 100,
              "entitlementDays": 25, "entitlementHours": 200, "yearEnd": "2026-12-31",
              "potName": "Holiday", "other": [], "error": null}]
}
```

`edays team`

Lists the roster from the config file (name and user ID). Reads only the local config; no API call.

```
edays --pretty team
```

The subcommands below fill the roster from eDays. They can only see the people eDays shows your account (see 4.4). People are matched by user ID, so existing entries — including ones you renamed — are never duplicated or overwritten.

`edays team sync`

Adds all your direct reports in eDays to the roster. People already in the roster are skipped. If a new person has the same name as an existing entry, the first 8 characters of their user ID are appended to the name.

```
edays --pretty team sync --dry-run   # preview
edays --pretty team sync
```

Option	Default	Meaning
`--dry-run`	off	Show what would be added; do not save

JSON `data`: `{"added": [{name, userId}], "skipped": [{name, userId}], "dry_run": false}`.

`edays team find [QUERY]`

Searches people visible to you in eDays whose name contains `QUERY` and shows their user IDs. Without `QUERY` it lists everyone visible. Changes nothing.

```
edays --pretty team find novak
edays --pretty team find --reportees
```

Option	Default	Meaning
`--reportees`	off	Only your direct reports

JSON `data`: `[{"name", "userId", "inRoster"}]` — `inRoster` is the roster name the person is stored under, or `null`.

`edays team add QUERY`

Finds one person by name and adds them to the roster. `QUERY` must match exactly one visible person, or exactly one person's full name. Otherwise the command fails with a `RosterError` listing the candidates.

```
edays team add "Jan Novak"
edays team add novak --as Jan
```

Option	Default	Meaning
`--as NAME`	eDays name	Roster name to store the person under

JSON `data`: `{"added": [...], "skipped": [...]}` (skipped if already in the roster).

`edays team absences`

Who is off: the absences of you and everyone in the roster over a day, a week or a date range, in one command. People are read in parallel; if one person can't be read, their row carries an `error` and the rest still come back.

```
edays --pretty team absences                          # this week
edays --pretty team absences --week 2026-W44
edays --pretty team absences --date 2026-12-23
edays --pretty team absences --from 2026-12-21 --to 2027-01-01
edays team absences --month 2026-11 --names "alice,bob"
```

Option	Default	Meaning
`--week [YYYY-Www]`	this week	ISO week; a bare `--week` is the current week
`--date YYYY-MM-DD`	—	A single day
`--month YYYY-MM`	—	A whole month
`--from YYYY-MM-DD` / `--to YYYY-MM-DD`	—	A range (`--to` defaults to `--from`), up to 186 days
`--names a,b`	you + roster	Comma-separated roster names (or `me`) instead

Use only one of `--week`, `--date`, `--month` or `--from/--to`.

With `--pretty` you get a person × working-day grid (weekends hidden):

```
                Team absences 2026-10-26 – 2026-11-01
                 Mo   Tu   We   Th   Fr
  Name           26   27   28   29   30
 ──────────────────────────────────────────
  me             ·    HO   ½H   ·    ·
  Alice Smith    ·    ·    PH   H    H
Public holidays: 2026-10-28 Statehood Day
```

Code	Meaning
`H` / `½H`	Holiday, full / part day (half day, or a time range under 8 hours)
`HO` / `½HO`	Home Office, full / half day
`BT`	Business Trip
`PH`	Public holiday (no absence booked)
other letters	Initials of any other absence type
`OFF`	A sensitive absence, in `redact` mode only (in the default `hide` mode it is not shown)
`?` / `(!)`	That person could not be read — the error is printed under the table

JSON `data`:


```
{
  "from": "2026-10-26", "to": "2026-11-01",
  "holidays": [{"date": "2026-10-28T00:00:00", "name": "Statehood Day"}],
  "people": [{"name": "me", "userId": "…", "absences": [ … as in read … ],
              "holidays": [ … ], "error": null}],
  "privacy": {"filtering": true, "mode": "hide"}
}
```


Sensitive absences are filtered exactly as in `edays read`.

Public holidays come from each person's own calendar (`people[].holidays`); the top-level `holidays` are those of the first person read successfully (you, by default).

`edays absence-types`

Lists the absence types you can book, with their IDs and the kind each one is mapped to in your config. With `--add` it also saves new kind mappings for `edays book`.

```
edays --pretty absence-types
edays absence-types --add holiday=123
edays absence-types --add "home-office=Home Office" --add unpaid=789
edays --pretty absence-types --for alice      # types available to a team member
edays --pretty absence-types --usable         # which ones can actually be booked
edays absence-types --only-usable
```

Option	Default	Meaning
`--add KIND=ID_OR_NAME`	—	Map `KIND` to an absence type by ID or name. Repeatable
`--for NAME`	`me`	Whose bookable types to list (types can differ per country)
`--usable`	off	Check every type: usable or not permitted, days- or hours-based, used recently
`--only-usable`	off	Like `--usable`, and hide the types that are not usable

A name resolves when it matches one type name exactly (case-insensitive) or is contained in exactly one type name. Otherwise the command fails with a `ConfigError` listing the candidates — each with a hint such as `usable, used recently` or `not permitted` — and none of the `--add` mappings are saved. An existing kind is overwritten.

About `--usable`:

- eDays lists — and even describes — types you are not allowed to book; only a booking **precheck** reveals it. `--usable` therefore reads each type's definition (for days/hours) and runs a precheck for a weekday about three months ahead (two requests per type, a few in parallel — under 10 seconds for ~70 types). A precheck validates only; **nothing is booked**. eDays answers "*You don't have permission to make this request.*" (HTTP 404) → **not permitted**. A precheck that merely fails validation (e.g. "not scheduled to work") still counts as usable. Any other failure shows as `?` with the reason (`usable: null`).
- **Used recently** means an absence with that type name is in the person's calendar within 90 days before or after today. The calendar only carries names, so two types with the same name are both marked. For sensitive types it is always empty (`recentlyUsed: null`), because it would tell whether the person was on sick leave.
- Sensitive types are still listed (the list of types is not personal data) and marked `sensitive: true`. `--add` refuses to map one (`SensitiveDataError`).

JSON `data`: `{"types": [{"id", "name", "kind", "sensitive"}], "added": {"kind": id}}`. With `--usable` / `--only-usable` each type also has `usable` (`true` / `false` / `null`), `timeUnit` (`"days"` / `"hours"` / `null`), `reason` and `recentlyUsed`.

`edays read`

Reads the eDays calendar for a person — booked absences and public holidays — for one month or any date range.

```
edays read                                      # current month, yourself
edays read --month 2026-11
edays read --from 2026-10-25 --to 2026-11-05    # spans two months
edays --pretty read --for alice                 # roster member
```

Option	Default	Meaning
`--month YYYY-MM`	current month	Month to read
`--from YYYY-MM-DD`	—	Range start, instead of `--month`
`--to YYYY-MM-DD`	`--from`	Range end (inclusive); a range may span up to 186 days
`--for NAME`	`me`	`me`, or a roster name (case-insensitive substring)

A range is fetched month by month and trimmed to the exact days. For the whole team at once, use `edays team absences`.

JSON `data` shape:

```
{
  "userId": "…",
  "from": "2026-11-01", "to": "2026-11-30",
  "month": "2026-11",
  "absences": [{"date": "2026-11-03T00:00:00", "name": "Home Office",
                "description": "8am-12pm", "dayPart": 2, "durationInMins": 240,
                "recordId": ""}],
  "holidays": [{"date": "2026-11-17T00:00:00", "name": "Public holiday"}],
  "privacy": {"filtering": true, "mode": "hide"}
}
```

- Sensitive absences (sick leave by default) are not in `absences` (1.3). `privacy` says whether filtering was on and in which mode. In `hide` mode there is deliberately no count of what was left out. In `redact` mode such a day appears as `{"date", "name": "Unavailable", "description": null, "dayPart": null, "durationInMins": null, "recordId": "", "private": true}`.
- `month` is present only when you read a month (`--month` or the default).
- `dayPart` is eDays' day part: `1` = full day, `2` = morning half day, `3` = afternoon half day (day-based types such as Home Office), `4` = a time range (hours-based types such as Holiday; `description` shows the times, e.g. `8am-4pm` or `12pm-4pm`, and `durationInMins` the length).
- `recordId` is **usually empty**: the eDays calendar does not return record IDs. A dedicated `records` command that lists cancellable IDs is planned.

`edays book KIND FROM_DATE`

Books an absence. `KIND` is a key from `[absence_types]` in your config. The command first runs the eDays **precheck** (validation, entitlement impact, authoriser) and prints its result to stderr, then asks for confirmation.

```
edays book holiday 2026-12-22 --to 2026-12-31
edays book holiday 2026-11-14 --from-time 13:00 --to-time 17:00
edays book home-office 2026-11-05
edays book home-office 2026-11-05 --day-part am
edays book home-office 2026-11-05 --for alice --details "client visit"
edays book holiday 2026-12-22 --dry-run
edays book holiday 2026-12-22 --yes
```

Option	Default	Meaning
`--to YYYY-MM-DD`	`FROM_DATE`	Last day of the absence (inclusive)
`--for NAME`	`me`	Book on behalf of a roster member (needs eDays rights)
`--from-time HH:MM`	—	Start time, for **hours-based** absence types only
`--to-time HH:MM`	—	End time, for **hours-based** absence types only
`--day-part full|am|pm`	`full`	Half day for **day-based** types: `am` = morning, `pm` = afternoon. Single day only
`--details TEXT`	—	Free-text note attached to the absence
`--dry-run`	off	Run the precheck only; book nothing
`--yes`	off	Skip the confirmation prompt

Notes:

- Whether a type is hours-based or day-based is decided by your eDays instance (Holiday is usually hours-based, Home Office day-based). When you pass `--from-time` / `--to-time` or `--day-part`, `book` reads the type from eDays first and stops with a `BookingOptionError` (exit code 1, nothing sent) when the option does not fit:
- times on a day-based type. eDays would ignore them and book the full day, so use `--day-part am|pm` for half a day;
- `--day-part am|pm` on an hours-based type. Use the times instead;
- `--day-part am|pm` on a type that eDays limits to full-day bookings;
- `--day-part` together with times, or with a `--to` date range.

- If eDays does not say how a type is measured, `book` prints a warning on stderr and continues. Check the result with `edays read`. A half day shows up there with a description such as `8am–12pm` or `12pm–4pm`.
- The precheck summary shows the duration, e.g. `0.5 day(s)` or `4:00 hours`. eDays' precheck message (on some instances literally "*Absence booked successfully*") is shown as `precheck: OK — nothing booked yet` / `… (dry run)`: the precheck never books anything.
- The confirmation prompt is written to stderr, so stdout stays pure JSON even in interactive use.
- An unknown `KIND` fails with a `ConfigError` that lists your configured kinds.
- A sensitive `KIND` (e.g. sick leave) is refused with `SensitiveDataError` before anything is sent to eDays. Book those in the eDays web app.
- JSON `data` always says whether something was booked:
- booked: `{"booked": true, "record": {…the created record as returned by eDays…}}`
- `--dry-run`: `{"dry_run": true, "booked": false, "precheck": {…}}`
- declined at the prompt: `{"booked": false, "aborted": true}`

Changed in 0.4.0: a successful booking used to return the eDays record directly as `data`; it is now under `data.record`.

`edays cancel RECORD_ID`

Cancels an absence record. The `RECORD_ID` is the `id` from the `book` output (`data.record.id`).

> **Known issue:** eDays may refuse that ID, and `edays read` does not return record IDs. Reliable record IDs and a check of the allowed actions before cancelling are planned for a later release. Until then, cancel in the eDays web app if `edays cancel` is refused.

```
edays cancel 3f1e…
edays cancel 3f1e… --for alice --yes
```

Option	Default	Meaning
`--for NAME`	`me`	Owner of the record (roster name or `me`)
`--yes`	off	Skip the confirmation prompt

> **Limitation:** eDays decides which records can be cancelled. Some absence types — often Home Office — have no cancel action, and eDays refuses the request; the command then fails with a message saying so. Ask your eDays administrator to remove such records.

7. Output Format & Exit Codes

Without `--pretty`, every command prints exactly one JSON object on stdout.

Success:

```
{"ok": true, "command": "read", "data": { … }}
```

Failure:

```
{"ok": false, "command": "book", "error": {"type": "EdaysAuthError", "message": "…"}}
```

With `--pretty`, successful output becomes a table or text; failures still print the JSON envelope on stdout and additionally a one-line `Error (…)` on stderr.

Status messages, the precheck summary and confirmation prompts always go to **stderr** (silence status messages with `--quiet`).

Exit code	Meaning
`0`	Success (including an aborted confirmation and `--dry-run`)
`1`	Error: API error, config error, unknown/ambiguous name, refused cancel, sensitive type refused
`2`	Authentication error — run `edays login` (also used by Click for invalid usage)

Error type	Typical cause
`EdaysAuthError`	No session yet, or the session expired
`EdaysApiError`	eDays rejected the request (validation, permissions, refused cancel)
`ConfigError`	Missing `base_url`, unknown absence kind, unresolvable `absence-types --add`, malformed config file (incl. an invalid `[privacy]` table)
`SensitiveDataError`	`entitlement`, `book` or `absence-types --add` on a sensitive type; `--show-sensitive` outside an interactive terminal or not confirmed
`RosterError`	`--for` or `team add` name matches nobody or several people; `my_user_id` not set
`LoginTimeout`	SSO not completed within `--timeout`
`BrowserNotInstalled`	Chromium missing — run `edays install-browser`

8. Config File Reference

Location: `C:\Users\<you>\.edays-cli\config.toml` (Windows) or `~/.edays-cli/config.toml` (macOS / Linux). Override it with `--config PATH`.

```toml
# ── Instance & identity ──────────────────────────────────────
[edays]
base_url     = "https://yourcompany.e-days.com"  # required — your eDays instance
my_user_id   = "1b2c3d4e-…"                       # filled in by `edays login`
session_path = ""                                 # blank = default location

# ── Absence types ────────────────────────────────────────────
# KIND name used by `edays book` = eDays absence type ID (integer)
[absence_types]
holiday     = 123
home-office = 456

# ── Team roster ──────────────────────────────────────────────
# Display name = eDays user ID. Used by --for (case-insensitive substring).
[roster]
"Alice Smith" = "1b2c3d4e-0000-0000-0000-000000000001"
"Bob Brown"   = "1b2c3d4e-0000-0000-0000-000000000002"

# ── Privacy ──────────────────────────────────────────────────
# Optional — without it the built-in sick-leave patterns apply.
[privacy]
sensitive_types = ["*medical*", "*maternity*", 999]
mode = "hide"
```

Key rules

Rule	Detail
`base_url` is required	Every command except bare `team` and `install-browser` needs it
Absence type IDs are integers	A non-numeric value is a `ConfigError`
`--for` matching	`me` → `my_user_id`; otherwise exactly one roster name must contain the text
No secrets in the config	The session lives in a separate file (see Section 5)
Updates keep your edits	`edays login`, `team sync` / `add` and `absence-types --add` change only the keys they set; comments, order and any other keys stay as you wrote them
`[privacy]` is yours	edays-cli reads it but never writes it

`[privacy]` — sensitive absences

Controls which absences are treated as sensitive and what happens to them (see 1.3). Every key is optional; a missing table means the built-in defaults.

Key	Default	Meaning
`sensitive_types`	`[]`	Extra sensitive types, **added** to the built-in ones. A string is a case-insensitive name pattern (`*` = any text, `?` = one character; without wildcards the whole name must match). An integer is an eDays absence type ID
`mode`	`"hide"`	`"hide"` leaves sensitive absences out entirely. `"redact"` keeps the day but shows only *Unavailable* (grid code `OFF`), without type, time or detail
`include_defaults`	`true`	Whether the built-in patterns `"*sick*"`, `"*illness*"`, `"*ill health*"` apply. Set to `false` only if they match a type that is not sensitive on your instance — and then list the real sensitive types in `sensitive_types`

Notes:

- **Name patterns are the main mechanism.** The eDays calendar returns absence names, not type IDs, so `read` and `team absences` can only match by name. Type IDs help where the type is known (`absence-types`, `entitlement`, `book`). Calendar names can differ from the names in `edays absence-types` (e.g. a country suffix); a pattern such as `"*medical*"` covers both.
- Find the names on your instance with `edays --pretty absence-types`, and check one known day in the eDays web app. Sickness is often an unplanned record type that does not appear among the bookable types at all.
- `hide` vs `redact`: `hide` reveals the least — a sick colleague just shows as not absent. `redact` keeps "who is off" accurate, but anyone reading the output can guess that *Unavailable* means something private.
- If `include_defaults = false` and `sensitive_types` is empty, filtering is off and every command prints a warning on stderr.
- `book` checks the type behind a kind when it reads the type from eDays (with `--day-part` or times). A kind that you mapped to a sensitive type by hand-editing the config is caught by its ID only if that ID is in `sensitive_types`; mappings made with `absence-types --add` are checked by name.

```toml
[privacy]
# Example: a local-language sick-leave name (Czech), medical appointments, one type by ID
sensitive_types = ["*nemoc*", "*medical*", 999]
mode = "redact"
```

---

9. Troubleshooting

Symptom	Fix
`No such option '--pretty'`	Put global flags before the subcommand: `edays --pretty read`
`ConfigError: base_url is not set`	Edit the config file named in the message (4.1)
`EdaysAuthError` / exit code `2`	Run `edays login`
`BrowserNotInstalled`	Run `edays install-browser`
`install-browser` fails while downloading (DNS, FFmpeg)	Usually transient — run `edays install-browser` again
Windows: `uv trampoline failed to canonicalize script path`, or `ModuleNotFoundError: No module named 'edays_cli'`	Installed from the Claude desktop app; set `UV_TOOL_DIR` and reinstall (Windows: installing from the Claude desktop app)
A person shows `(!)` or an `error` in `team absences` / `entitlement --team`	Read the error under the table; usually you have no eDays rights to see that person
`Unknown absence kind 'holiday'` on `entitlement`	Map it first: `edays absence-types --add holiday=ID`
`LoginTimeout`	Re-run `edays login`, finish SSO faster or use `--timeout 600`
Login window opens but never closes	Make sure you reached the eDays home page; some SSO flows need one extra click
`Unknown absence kind`	Map it: `edays absence-types --add KIND=ID` (4.3)
`RosterError: ... is ambiguous`	Use a longer part of the name
`absence-types --add` says a name is ambiguous	Use the ID of the candidate marked `usable` (see `edays --pretty absence-types --usable`)
`team sync` / `team find` finds nobody	eDays shows you no staff (usual for non-managers); add IDs manually (4.4)
`eDays refused to cancel`	The record type is not cancellable on your instance; ask your eDays admin
`BookingOptionError` on book	The option does not fit the absence type. Day-based types (Home Office) take `--day-part`

Symptom	Fix
	`am|pm`; hours-based types (Holiday) take `--from-time` / `--to-time`. See `edays book`
`EdaysApiError (400)` on book	Read the message — usually an overlap, a non-working day, or missing times for an hours-based type. Try `--dry-run` to see the precheck
`SensitiveDataError`	The absence type is sensitive (sick leave by default); use the eDays web app. For `--show-sensitive`: run it yourself in a terminal and answer the prompt (1.3)
A colleague on sick leave shows as available	Expected in the default `hide` mode; set `[privacy] mode = "redact"` to show such days as *Unavailable*
An error says "details hidden"	The eDays error mentioned a sensitive type; see it in the eDays web app
Sick leave still visible	Its name does not match the patterns; add it to `[privacy] sensitive_types` (8)

---

10. Upgrading & Uninstalling

Upgrading

```
uv tool upgrade edays-cli        # if installed via uv
pip install --upgrade edays-cli  # if installed via pip
edays --version
edays install-browser            # in case the Playwright version changed
```

Your config and session files are not affected by upgrades. On Windows, upgrade from the same place you installed from, or set `UV_TOOL_DIR` first (see 3).

Upgrading to 0.4.0: scripts that read the JSON of `edays book` must take the created record from `data.record` instead of `data`.

Upgrading to 0.5.0: sick leave is no longer shown by default (1.3). Check that the built-in patterns match the sick-leave names on your instance, and add other sensitive types to `[privacy]`. Your existing config keeps working; the `[privacy]` table is optional. An existing mapping

of a sensitive kind (e.g. `sick = 789`) stays in the config, but `book` and `entitlement` now refuse it.

Uninstalling

```
uv tool uninstall edays-cli      # if installed via uv
pip uninstall edays-cli          # if installed via pip
```

Then optionally remove the `~/.edays-cli` folder (config, session and browser profile) and the Playwright browser cache listed in Installing the browser.

Windows:

```
Remove-Item "$env:USERPROFILE\.edays-cli" -Recurse
```

macOS / Linux:

```
rm -rf ~/.edays-cli
```

End of manual — edays-cli v0.5.0