

edays-cli — Installation and Usage Manual

Version: 0.3.0

Date: 2026-10-06

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](#)
 2. [Prerequisites & System Requirements](#)
 - [Installing Python](#)
 - [Installing uv](#)
 3. [Installation](#)
 - [Installing the browser](#)
 - [If the edays command is not found](#)
 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](#)
 - [team](#)
 - [team sync](#)
 - [team find](#)
 - [team add](#)
 - [absence-types](#)
 - [read](#)
 - [book](#)
 - [cancel](#)
 7. [Output Format & Exit Codes](#)
 8. [Config File Reference](#)
 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** a month's calendar — booked absences, public holidays — for yourself or a team member
- **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** your profile, entitlement, 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.

2. Prerequisites & System Requirements

Requirement	Minimum	How to check
Python	3.11 or newer	<code>python --version</code> or <code>python3 --version</code>
Package installer	<code>uv</code> (recommended) or <code>pip</code>	<code>uv --version</code> or <code>pip --version</code>
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. 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 0.3.0
```

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.

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_url to your eDays instance, e.g. https://<company>.e-days.com"}}
```

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:

```
[edays]  
base_url = "https://yourcompany.e-days.com"  
my_user_id = ""  
session_path = ""  
  
[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.

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	<code>~/.edays-cli/edays_state.json</code> (Windows: <code>C:\Users\<you>\.edays-cli\edays_state.json</code>)
Browser profile	<code>~/.edays-cli/browser-profile/</code> (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
<code>--version</code>		Show the installed version and exit
<code>--pretty</code>	<code>-p</code>	Human-readable tables/text on stdout instead of JSON
<code>--quiet</code>	<code>-q</code>	Suppress informational messages on stderr
<code>--config PATH</code>		Use a different config file instead of the default
<code>--help</code>		Show help (also works after any subcommand)

`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
<code>--timeout SECONDS</code>	<code>300</code>	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 entitlement summary. Because it calls the API, it doubles as a **session check** — exit code `0` means the session is valid.

```
edays --pretty whoami
```

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
<code>--dry-run</code>	<code>off</code>	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
<code>--reportees</code>	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
<code>--as NAME</code>	eDays name	Roster name to store the person under

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

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 sick=789
edays --pretty absence-types --for alice      # types available to a team member
```

Option	Default	Meaning
<code>--add KIND=ID_OR_NAME</code>	—	Map <code>KIND</code> to an absence type by ID or name. Repeatable
<code>--for NAME</code>	me	Whose bookable types to list (types can differ per country)

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, and none of the `--add` mappings are saved. An existing kind is overwritten.

JSON data:{"types": [{"id", "name", "kind"}], "added": {"kind": id}}.

edays read

Reads one month of the eDays calendar for a person: booked absences (with their recordId, needed for cancel) and public holidays.

```
edays read                                # current month, yourself
edays read --month 2026-11
edays --pretty read --for alice          # roster member
```

Option	Default	Meaning
--month YYYY-MM	current month	Month to read
--for NAME	me	me, or a roster name (case-insensitive substring)

JSON data shape:

```
{
  "userId": "...",
  "month": "2026-11",
  "absences": [{"date": "2026-11-03", "name": "Home Office",
                  "description": "Full day", "recordId": "..."}],
  "holidays": [{"date": "2026-11-17", "name": "Public holiday"}]
}
```

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)

Option	Default	Meaning
<code>--from-time HH:MM</code>	—	Start time, for hours-based absence types only
<code>--to-time HH:MM</code>	—	End time, for hours-based absence types only
<code>--day-part</code> <code>full am pm</code>	<code>full</code>	Half day for day-based types: <code>am</code> = morning, <code>pm</code> = afternoon. Single day only
<code>--details TEXT</code>	—	Free-text note attached to the absence
<code>--dry-run</code>	<code>off</code>	Run the precheck only; book nothing
<code>--yes</code>	<code>off</code>	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`.
- 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.
- The JSON `data` of a successful booking is the created record as returned by eDays.

`edays cancel RECORD_ID`

Cancels an absence record. Get the `RECORD_ID` from `edays read` (field `recordId`) or from the `book` output.

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

Option	Default	Meaning
<code>--for NAME</code>	<code>me</code>	Owner of the record (roster name or <code>me</code>)
<code>--yes</code>	<code>off</code>	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 <code>--dry-run</code>)
1	Error: API error, config error, unknown/ambiguous name, refused cancel
2	Authentication error — run <code>edays login</code> (also used by Click for invalid usage)
Error type	Typical cause
<code>EdaysAuthError</code>	No session yet, or the session expired
<code>EdaysApiError</code>	eDays rejected the request (validation, permissions, refused cancel)
<code>ConfigError</code>	Missing <code>base_url</code> , unknown absence kind, unresolvable <code>absence-types --add</code> , malformed config file
<code>RosterError</code>	<code>--for</code> or <code>team add</code> name matches nobody or several people; <code>my_user_id</code> not set
<code>LoginTimeout</code>	SSO not completed within <code>--timeout</code>
<code>BrowserNotInstalled</code>	Chromium missing — run <code>edays install-browser</code>

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`.

```
# — 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"
```

Key rules

Rule	Detail
<code>base_url</code> is required	Every command except bare <code>team</code> and <code>install-browser</code> needs it
Absence type IDs are integers	A non-numeric value is a <code>ConfigError</code>
<code>--for</code> matching	<code>me</code> → <code>my_user_id</code> ; otherwise exactly one roster name must contain the text
No secrets in the config	The session lives in a separate file (see Section 5)
Rewrites drop comments	<code>edays login</code> , <code>team sync/add</code> and <code>absence-types --add</code> rewrite the file; TOML comments are not preserved

9. Troubleshooting

Symptom	Fix
No such option '--pretty'	Put global flags before the subcommand: <code>edays --pretty read</code>
<code>ConfigError: base_url is not set</code>	Edit the config file named in the message (4.1)
<code>EdaysAuthError</code> / exit code 2	Run <code>edays login</code>
<code>BrowserNotInstalled</code>	Run <code>edays install-browser</code>
<code>LoginTimeout</code>	Re-run <code>edays login</code> , finish SSO faster or use <code>--timeout 600</code>

Symptom	Fix
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: <code>edays absence-types --add KIND=ID</code> (4.3)
RosterError: ... is ambiguous	Use a longer part of the name
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 <code>--day-part am pm</code> ; hours-based types (Holiday) take <code>--from-time/--to-time</code> . See <code>edays book</code>
EdaysApiError (400) on book	Read the message — usually an overlap, a non-working day, or missing times for an hours-based type. Try <code>--dry-run</code> to see the precheck

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.

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 v0.3.0