Audion Hub Manager — User Guide

Русский · About · Decisions · History

Contents

Day-to-day work: register a project, build a mirror, keep history, connect a remote, restore from a copy.

First Run

```
launcher_gui.cmd
```

The window opens in a browser on a local address. If there is no portable runtime, build one:

```
install\Build_Portable_Env.cmd
```

The program opens even with a corrupt project registry: surviving records are kept, and the problem is reported in the output dock. An empty registry yields a temporary project pointing at the program itself — the window opens, and the settings can be fixed from inside.

Registering a Project

A project is not a folder but a set of three paths and the rules around them:

what	why
source	the live project, the single source of truth
mirror	a filtered copy with its own Git
docs	optional layer of notes and descriptions
profile	which files pass into the mirror and into a commit
branch	which branch Git commands work against

The project list at the top switches the whole set at once.

Have a shared folder with twenty projects? Don't make it one project. Run the scan: the program finds the real roots inside, including nested `Project/Project` shapes, and registers each separately. After that projects switch one at a time.

Registry cluttered — records with vanished paths, duplicates, leftover samples? `Support → Clean registry`. Only the list is cleaned: project folders, mirrors, and docs are untouched.

The Mirror

Two steps, and this is the central safety rule.

First look at the plan:

```
launcher_cli.cmd --mirror-preview <project_id> --json
```

Then apply deliberately:

```
launcher_cli.cmd --mirror-apply <project_id> --apply --json
```

Without `--apply` the command stays a preview if the profile says so. That is deliberate: the mirror can delete its own files, and "it applied by accident" is not a story anyone wants to hear once.

Only what the profile allows passes into the mirror. Runtimes, caches, logs, builds, binary payloads, local overrides, and secrets stay outside.

Verification

`Verify mirror` compares source and mirror by checksums, read-only. Manifests are built in memory and the result lands as one dated file in `logs/<project_id>/`. Neither source, mirror, nor docs receive service files.

A separate button checks that hashing uses real BLAKE3 rather than the SHA-256 fallback.

The docs layer is not verified separately: its technical copy is already in the mirror.

History in Git

The program works with ordinary repositories — one for the source, one for the mirror. The first is for live development and the editor, the second is a clean history of the filtered copy.

A broad `git add .` will not do for the mirror. Only what the profile would pass goes into a commit. Otherwise filtering loses its point at the first commit.

A good commit message says what changed:

```
docs(hub): projection v0.1.17 — refreshed the builder menu
```

The version is set by hand. That is deliberate: automatic numbering breeds noise and false importance too easily.

The daily cycle is covered end to end: state, comparison, staging and committing, tags, branches and switching, stashes, history and graph, recovery, maintenance. Rare and dangerous operations remain explicit templates — to be run deliberately, not pressed in passing.

Remotes and Sign-In

The program stores no tokens. Sign-in is delegated to the ordinary tools:

```
gh auth login
glab auth login
ssh -T git@github.com
ssh -T git@gitlab.com
```

HTTPS passwords live in the credential manager, keys in ssh-agent. If you are already signed in elsewhere, push and pull work without the program's knowledge.

A remote address is assembled with buttons: the platform in the top row (GitHub, GitLab, Codeberg, Forgejo, Gitea, your own server), the access type in the second (key or token). The last three add address and port fields — cloud platforms don't need them.

Your Own Forgejo or Gitea

The account connects the ordinary way for those servers: a personal access token from the application settings. The program verifies it against the server and hands it to your credential helper; the token never reaches project files. From then on push picks it up by itself, and the buttons show the account, list repositories, and let you create one.

Known server addresses live in `config/forgejo_hosts.json` — with no tokens.

An Ordinary Day Across Two Machines

```
machine A: commit → tag → push
machine B: fetch with prune → pull fast-forward only
```

Fetching updates knowledge of remote branches without touching your files. A fast-forward pull advances the current branch when you want it to.

Restoring

Source lost, mirror alive:

```
robocopy "<mirror>\<Project>" "<root>\<Project>" ^
  /E /COPY:DAT /DCOPY:DAT /R:1 /W:1 /XJ
```

Then rebuild the runtime or release artefacts if you need them. What comes back is what passed the filter — nothing else was ever in the mirror.

Checking After Changes

```
python -m compileall -q system_core
python -m pytest -q tests
python system_core\ui_nicegui\app.py --smoke
```

Documented baseline: `121 passed`.

Technical Reference

The Window

On the left, opening: project, mirror, docs folder, editor, terminal, Git. The terminal opens in the source; Git opens at the repository root and immediately shows short status.

On the right, tabs in two rows:

tab	about
Quick	local Git commands: init, status, root, short log
Branch	branch state and comparison, switching, tags, merge, revert, cherry-pick
Editor	Markdown and text: open, save, hand over to VS Code
Diff	what changed
Storage	profiles, safety scan
Remote	addresses, push, pull, sign-in
Basket	assembling a readable commit
Reader	viewing documents
History	log and graph
Details	information about the selection

Registry-Wide Actions

Rebuild the list, preview mirrors across all projects, clone a source, safety check across all, verify across all, Git status across all, combined workspace.

After a status run across all projects, a badge summarises it: how many projects, how many clean, how many with changes, where the errors are.

Paths and Files

```
config/projects.json              project registry
config/projection_profiles.json   filtering profiles
config/forgejo_hosts.json         self-hosted server addresses, no tokens
logs/<project_id>/                verification reports
```

Machine-specific settings live in `*.local.json`, `.env`, and other excluded files. They never reach shared config.

Protected Unconditionally

- the source — from any write by the mirror;
- `.git/**` — from scanning, deletion, and maintenance;
- deletion in the mirror — any copy or hashing error skips it, and the setting that once allowed otherwise no longer applies;
- the mirror plan — rules are re-checked before writing, so a stale or edited plan cannot delete anything outside its own area.