Audion Hub Manager — History

Русский · About · User Guide · Decisions

Contents

This history exists not to count versions but to avoid fixing the same thing twice. What changed is here — and why.

1.10.0

A corrupt registry no longer blocks startup. A single stray comma in `config/projects.json` used to abort the program with a traceback before the window opened: the registry was read with a raising JSON loader on import. Startup config reads now fall back to defaults, keep the surviving records, and report the problem in the output dock. An empty or unreadable registry yields a temporary project pointing at the program itself — the window opens, and the config can be fixed from inside it.

Mirror plans are re-validated at apply time. Applying trusted the roots stored in the plan file, so a stale or edited plan could delete under a path overlapping the source. The profile and the non-overlapping-roots rule are now re-checked before anything is written or removed.

Behaviour change: a copy error blocks deletion unconditionally. Previously `delete_after_successful_copy: false` let deletion proceed after failures — exactly the case

where a mirror can drop files that exist nowhere else. The setting remains in the config but can no longer disable that protection.

Commands run through the engine, not through a shell string. Bundle creation and fast-forward pull composed a command line in the GUI, interpolating a filesystem path into it: a folder name containing `&` or a quote would have broken it. Both now run as argument lists. The engine functions existed all along and were simply never called.

Taskbar identity. A name that already begins with the brand no longer gets a second prefix: the app is `Audion.Hub.Manager` rather than `Audion.Tools.Audion.Hub.Manager`. Windows groups taskbar windows by this id.

A new icon — a white three-armed hub mark on a near-black rounded tile, rendered at 16 through 256 pixels, with a simplified variant below 32 where the three nodes collapse into a blur. The previous icon is kept.

Two server helpers that nothing used were removed.

1.9.0

The token works immediately. Every button used to read the token back from the credential store, so one typed into the field did nothing until it was stored — saving was a precondition for anything working, rather than a convenience. The typed token is now used at once, and "remember token" does what its name promises: keeps it across restarts. The status line names which token the buttons are about to use.

Forgetting clears both places at once — the field and the remembered copy in the credential store.

1.8.0

Cloning arrived in the Remote pane, which had no cloning action at all. It takes the address from the field, clones next to the active project folder, refuses to clone onto an existing folder, and runs `git` as an argument list rather than through a shell.

Cloning no longer hangs. It runs with terminal prompts disabled: a credential prompt has no way to reach you from the window, so instead of waiting into the void the program warns that Git reads the store, not the form.

The Remote pane remembers your choice — platform and address type across sessions — and opens where you left it instead of resetting to GitHub.

1.7.3

Token scopes follow the route, not the object. Per-operation testing against a live Forgejo 15 showed that creating a repository sits under `/api/v1/user/*` and therefore needs a user scope, not a repository one. Git over HTTPS is judged separately and needs a repository scope regardless — one set of scopes cannot cover both the API and git.

Documented what is wrong with a credential-bearing address: Git writes that token in the clear into `.git/config` of the working copy, where it survives every backup and mirror of that folder. The same family of mistake as a secret in a URL query parameter, and avoided entirely by the credential-helper path.

1.7.2

A command-injection path was closed. URL parsing keeps `;`, `&&`, and `$(…)` inside a hostname, and that hostname was interpolated into an `ssh -T` probe the GUI runs through a shell. A `forgejo_hosts.json` arriving with a cloned project could therefore run a command on the first probe. Hostname, SSH user, and port are now validated where every path converges, hostile records are dropped instead of loaded, and addresses embedding credentials are refused outright.

Addresses that make Git execute a command are refused (`ext::`, `fd::`), along with addresses starting with a dash — Git would read them as an option — and addresses carrying control characters. Validation runs both when saving from the pane and when applying a config file, so one from another project cannot smuggle such an address in.

A warning when a host is reached over plain HTTP before a token is sent: legitimate for a LAN instance, but the token crosses the wire unencrypted.

1.7.1

A server typed into the form is no longer dropped. The config write went through a function that returns early when the login already matches, so the host was never written to file and had to be retyped on the next start.

Config parsing became fault-tolerant. A hand-edited file with a syntax error used to raise straight into an event handler, because the pane reads it on every platform switch and every keystroke in the server field.

The repository list is no longer silently truncated. Paging stops at a thousand entries, and reaching that ceiling is now reported rather than presented as a complete list.

An unusable SSH port is rejected instead of quietly substituting 22, which built a plausible-looking address that failed later at push time.

An address with an embedded login or token is not saved into `config/remotes.json` — a committed file, where the secret would have been published with the next commit.

Erasing a token requires a named account: credential helpers key entries by host *and* username, so an empty login matched nothing while reporting success.

1.7.0

Support for self-hosted Forgejo and Gitea. `config/forgejo_hosts.json` describes servers — address, SSH user and port, preferred address type, cached login — and holds no secrets.

The ordinary account contract was adopted: a personal access token, verified against the server and handed to the credential helper. Nothing is written into project config, and plain HTTPS push picks the same credential up.

OAuth2 sign-in was evaluated and rejected for now: Forgejo has not implemented OAuth2 scopes, so such tokens carry administrative account rights, expire quickly, and cannot serve push unaided.

403 is separated from 401 in every call. An insufficient scope now reports "the token is valid but was created without the scope this call needs", names the scope the server asks for, and suggests the working set — instead of a bare status code alongside a token that is in fact correct.

Server error text is redacted before display. Forgejo quotes an unknown token back in its message, so a revoked or expired token would have appeared verbatim in the output dock, and from there in screenshots and copied output.

The 413 case is documented for instances behind a proxy or tunnel with a request-body cap: it is the initial history push that hits it, and pushing over SSH — including through a local port forward — is the way around it.

Earlier versions

`0.2.x` — planning: mirror safety contracts, layer separation, filtering profiles. `v0.1.0` — the application skeleton.

What carried over from that period are the rules set out in Decisions: collected from real Disk Auditor failures, and not worth learning twice.