

Audion Disk Tools

[Русский](#) · [User Guide](#) · [Command Reference](#)

Contents

- [Why It Exists](#)
- [Principles](#)
- [What It Can Do](#)
- [Next](#)
- [Technical Reference](#)
 - [Running](#)
 - [Modes and Where to Start](#)
 - [Where Things Live](#)
 - [Limits of the File Engine](#)
 - [Workbench Naming](#)

A portable toolkit for disks and files on Windows: compare, update, mirror, filter by type, archive, transfer across the network or to cloud storage.

Why It Exists

File synchronisation is the task where a mistake costs most. A program that "just copies" will one day delete what is no longer in the source — and you find out when the source is gone too. A program that "decides intelligently" does it silently.

Here it is the other way round: **direction, deletion, and the comparison method are all declared explicitly**, and before any serious operation there is a step that changes nothing.

The safety ladder:

1. compare what actually differs – nothing written to disk
2. preview what exactly will be copied, updated, deleted
3. run what the preview showed

For everyday mirroring keep the safe comparison; switch to strict when files of equal size and equal timestamps matter.

Principles

No two fixed folders. Source and target are chosen freely: an external drive, a network path, a large directory. Terabytes are not copied into the project just to work with them.

The comparison method is named aloud.

method	how it compares
quick	size and timestamp
safe	plus a checksum where sizes match and timestamps differ
strict	a checksum for every same-size file, even when timestamps match

Safe does not catch "same size, same time, different content" — that is what strict is for.

Deletion is a separate policy, not a side effect. A one-way update deletes nothing. A hard mirror deletes what is absent from the source — and is called hard. There is a middle option: the surplus goes to quarantine instead of being erased.

Cloud is separated from files. Ordinary paths — local and external drives, network folders, mounted remotes — go through the file engine. Real cloud storage lives in its own section on its own engine.

Access tokens and the contents of its configuration never reach project files.

A vanished drive is not an empty folder. Windows presents a dropped network mount as an existing, empty directory — and a mirror would read that as "delete everything in the target". So profiles with a network root carry an anchor: an `.audion-anchor` file must be present in both roots before the walk begins. It is created once by a separate command and **never creates itself during a run** — otherwise the guard would protect against everything except the real case.

Before the run you are shown what you are risking. A card above the command names three things: whether the anchor is in place, whether there is free space for the planned write, and how many

files will be deleted — as a count and as a share of the target. For mirror policies it also issues a confirmation bound to the plan you just read: what runs is exactly what was shown.

Verified on real folders, not on samples. In the last run: a mask copy of `*.docx` moved 29 documents, a checksum manifest was built over 2478 entries, and verification of that manifest passed. The only interactive branch left is the system folder picker, which needs a real click.

What It Can Do

section	about
Compare	what differs between two folders, with no changes on disk
One-way update	new and changed files go to the target, nothing is deleted
Mirror	an exact copy of the source, including deleting the surplus in the target
Mirror with quarantine	the same, but the surplus goes to quarantine
Two-way sync	exchange of missing and newer files, with no automatic deletions
Manifest and diff	a list of paths with checksums, applied as an update or a mirror
Mask copy	a one-off selection: documents only, media only, your own pattern
Archiving	ZIP, 7Z, SFX, TAR and its compressed variants; encryption, self-extraction
Data transfer	folder, network share, server, and cloud as equals; the engine per pair is chosen automatically

The type filter is not a list of extensions in a field but a browser: groups for documents, development, media, archives, applications, temporary files, plus your own masks.

Next

- [User Guide](#) — step by step, modes, profiles.
- [Command Reference](#) — every command and parameter.
- `tools\SYNC_PROFILES_RU.md` — profiles, presets, filters.
- `tools\ARCHIVE_OPERATIONS_RU.md` — archives, self-extraction, encryption.
- `tools\TRANSFER_RU.md` — data transfer: operations, engines, packaging.

- `tools\RCLONE_OPERATIONS_RU.md` — cloud connections, diagnostics.
- `tools\RCLONE_PORTABILITY_RU.md` — installation, portable configuration.
- `tools\MANIFEST_DIFF_RU.md` — manifests and diffs.
- `tools\WORKBENCH_AND_GUI_RU.md` — workbench, terminal, tooltips.

Section documents are Russian only.

Technical Reference

Running

`launcher_project.cmd`

the main one

`launcher_profiles.cmd`

saved synchronisation profiles

Both work through the quick picker and through a plain menu.

Modes and Where to Start

scenario	command	direction	deletes	start with
folder check	<code>audit</code>	one folder	no	straight away
compare	<code>compare</code>	source → target	no	before anything serious
one-way, dry	<code>sync --dry-run</code>	source → target	no	after compare
one-way	<code>sync</code>	source → target	no	after the dry run
mirror, preview	<code>backup --preview</code>	source → target	no	optional
mirror	<code>backup</code>	source → target	yes	when needed
mirror with quarantine	<code>backup --operation-policy mirror_safe</code>	source → target	to quarantine	optional

scenario	command	direction	deletes	start with
two-way, dry	<code>sync2 --dry-run</code>	both ways	no	before running
two-way	<code>sync2</code>	both ways	no	after the dry run
mask copy	<code>copy_by_mask --mask-globs</code>	source → target	no	a dry run
saved profile	<code>--pair</code>	per configuration	per policy	compare → preview → run

The legacy `dry_run_default` key is ignored with a deprecation warning — direction and deletion come from an explicit operation policy.

Where Things Live

```
config\sync_presets.json    filter presets
config\sync_pairs.json     saved folder pairs and policies
config\rclone\rclone.conf  portable cloud configuration
```

Limits of the File Engine

The main engine works with anything Windows and Python see as an ordinary directory: local and external drives, UNC network paths, mounted network folders, local folders of cloud clients, mounted remotes.

Real cloud storage is the data transfer section: it uses its own engines, by default through the portable configuration inside the project, and stores neither tokens nor that configuration's contents in project files.

Workbench Naming

One shared vocabulary across all Audion projects: **Source**, **Add file...**, **Target**, **Reset**, **Delete**, **List**. Russian: **Источник**, **Добавить файл...**, **Назначение**, **Сбросить**, **Удалить**, **Список**.

Reset restores the project folders and deletes no files. **Delete** clears the current source and target only after confirmation. The words **Destination**, **Clear**, «Цель», and «Очистить» are not used for these controls.