

Audion Disk Tools - архитектура проекта

Документ фиксирует текущую карту проекта: что является источником истины, где лежат конфиги, какие папки считаются управляемыми и как GUI связан с CLI.

Имя проекта

Публичное имя: `Audion Disk Tools`.

Исторические идентификаторы в коде и схемах могут оставаться `Audion Disk Auditor` и `audion_disk_auditor`. Их не нужно переименовывать без отдельной миграции.

Источник истины

CLI и файловое ядро остаются источником истины:

- `system_core\main.py` - CLI subcommands.
- `system_core\auditor_core.py` - сравнение, фильтры, планы, apply, отчёты.
- `system_core\services\disk_auditor_service.py` - сервисный слой GUI к CLI.
- `system_core\services\rclone_service.py` - построение и безопасность rclone-команд.
- `system_core\ui_nicegui\app.py` - GUI, Workbench, терминал, архивы, network/RClone wrappers.

GUI не должен реализовывать свой отдельный алгоритм синхронизации. Он собирает параметры, запускает сервис/CLI и показывает вывод.

Папки

`input` и `output` - управляемые проектные fallback-папки. Они удобны для тестов, но больше не являются обязательным центром работы.

logs - plain UTF-8 логи процессов. RClone пишет отдельный

logs\YYYYMMDD_HHMMSS_rcclone_<mode>.log.

report - JSON/TXT/HTML/MD отчёты, списки файлов, summaries.

workspace - временные рабочие данные, local stage для части network/archive процедур.

config - настройки GUI, профили, пресеты, кэши.

Tools - переносимые внешние утилиты, сейчас PeaZip и rclone.

runtime и **wheelhouse** - воспроизводимый Python payload.

install - build/install/update скрипты.

licenses - third-party notices.

Основные конфиги

config\tool_manifest.yaml описывает CLI/GUI операции, поля и captions. Часть ROOT-команд создаётся в **app.py**, потому что они зависят от runtime-состояния GUI.

config\gui_settings.yaml хранит язык, тему, текущий source/target и GUI-only настройки.

config\ui_colors.yaml хранит темы. Дефолтная тема - **code_dark**.

config\path_history.json хранит историю source/target и pinned маршруты.

config\terminal_commands.json хранит историю, pinned commands, shell и CWD правого терминала. Команды с очевидными маркерами секретов автоматически исключаются из history/pins.

config\mask_cache.json хранит наборы масок.

config\rclone_command_cache.json хранит историю конструктора RClone, pinned/unpinned/deleted варианты.

config\rclone_endpoint_history.json хранит локальную историю SFTP **user** и **host/IP** из успешных RClone-операций. Пароли, порты и пути туда не записываются.

config\sync_pairs.json - пользовательские профили. Сейчас файл может быть пустым.

config\sync_pairs.example.json - bundled fallback-профили и шаблон экспорта.

config\sync_presets.json - именованные группы расширений.

Управляемые очистки

GUI и сервисные команды очистки должны ограничиваться управляемыми папками проекта. Очистка `input/output` или `workspace` не должна затрагивать произвольные source/target, внешние диски или UNC-пути.

`cleanup_project.cmd` - отдельная source/release cleanup процедура. Это не обычная install-cache очистка.

Внешние инструменты

PeaZip используется для архивных и network archive сценариев.

RClone используется только в отдельном cloud/remotes слое. Robocopy и SMB остаются Windows/network слоем.

OAuth-токены, app passwords и содержимое `rclone.conf` не должны попадать в проектные JSON/YAML. Portable `rclone.conf` хранится отдельным ignored-файлом в `config\rclone\rclone.conf`.

Для release-сборки локальные runtime-state файлы вроде `config\gui_settings.yaml`, `config\path_history.json`, `config\workspace_paths.json`, `config\terminal_commands.json`, `config\mask_cache.json`, `config\rclone_command_cache.json`, `config\rclone_endpoint_history.json` и `config\rclone\rclone.conf` должны оставаться вне публичного архива или репозитория.