Профили, пресеты и фильтры синхронизации

Профиль описывает повторяемую файловую задачу: откуда читать, куда писать, какой режим применять и какие фильтры использовать.

Это основной документ про маршруты. Отдельного слоя site/endpoint в проекте больше нет: удалённое хранилище подключается снаружи как обычная буква диска (peer-to-peer mount), после чего удалённый корень для Disk Tools - просто путь. Всё, что раньше выводилось из сетевой топологии, теперь описывается парой путей в `sync_pairs.json`.

Файлы

`config\sync_pairs.json` - пользовательские профили.

`config\sync_pairs.example.json` - bundled examples и fallback для GUI, если рабочий файл пустой.

`config\sync_presets.json` - named groups расширений.

Текущий fallback

Если `sync_pairs.json` содержит пустой список:

```
{
  "pairs": []
}
```

GUI показывает примерные профили из `sync_pairs.example.json`. Это сделано специально: новый пользователь видит структуру профилей и может экспортировать/адаптировать её.

Поля pair

Поле	Что означает
`name`	Уникальное имя профиля.
`source`	Source path для профиля.
`target`	Target path для профиля.
`mode`	`safe`, `quick`, иногда `strict` в CLI.
`mirror`	Legacy boolean. Сейчас policy важнее.
`operation_policy`	`copy_update`, `mirror_safe`, `mirror_hard`.
`requires_masks`	Профиль требует runtime маски. Используется `copy_by_mask`.
`sync_kind`	`sync`, `backup`, `sync2`.
`anchor`	Требовать файл-якорь `.audion-anchor` в обоих корнях перед сканированием. По умолчанию `false`.
`note`	Свободный текст. Показывается в списке профилей GUI и в выводе `main.py pairs`.
`include_presets`	Список include preset names.
`exclude_presets`	Список exclude preset names.
`include_globs`	Прямые include masks.
`exclude_globs`	Прямые exclude masks.
`include_dirs`	Ограничить относительные каталоги.
`exclude_dirs`	Исключить относительные каталоги.
`exclude_hidden`	Исключить dot-prefixed hidden paths.
`exclude_if_size_gt`	Исключить файлы больше N bytes.
`exclude_if_size_lt`	Исключить файлы меньше N bytes.

Текущие bundled профили

`copy_by_mask` - one-way copy. Он требует выбранных масок или групп и не должен восприниматься как backup mirror.

`dev_backup` - mirror для кода и документов, presets `DEV`, `DOCS`, exclude `TEMP`, исключает `.git`, `.venv`, `node_modules`, caches/build dirs.

`docs_backup` - mirror документов, preset `DOCS`, exclude `TEMP`, скрытые пути исключаются.

`graphics_backup` - mirror графики и RAW, preset `GRAPHICS`.

`video_backup` - mirror видео, preset `VIDEO`, исключает proxy/render/cache dirs.

`audio_backup` - mirror audio, preset `AUDIO`.

`arch_backup` - mirror архивов, preset `ARCH`.

`apps_backup` - mirror приложений/installers, preset `APPS`.

Пресеты

Текущие preset groups:

Preset	Содержимое
`DEV`	Код, скрипты, configs, SQL, IaC, web.
`ARCH`	Архивы, образы, split/legacy архивы.
`AUDIO`	Lossless, compressed audio, voice, MIDI/tracker, legacy audio.
`VIDEO`	Современные/legacy containers, transport streams, DVD, subtitles.
`GRAPHICS`	Raster, web graphics, print/vector/design, camera RAW, fonts, assets.
`DOCS`	Office, PDF, text, markdown, CSV, CHM, DjVu, OneNote, diagrams.
`TEMP`	Временные, backup, log, cache-like masks. Обычно exclude.
`APPS`	EXE/MSI/scripts/widgets/binary packages.

Runtime override в GUI

Поля `manual_extensions` и `profile_extensions__<name>` не переписывают JSON-профили. Они добавляют runtime маски только на текущий запуск.

Вкладка `ВКЛЮЧИТЬ` отправляет include/mask globs.

Вкладка `ИСКЛЮЧИТЬ` отправляет exclude globs.

Пустой override означает: оставить профиль как он есть.

Смонтированный удалённый корень и якорь

Смонтированный удалённый корень, чей peer уснул или отключился, выглядит как существующая пустая папка. `resolve_existing_dir()` такой корень принимает, сканирование возвращает ноль файлов, и план читается как "удалить в приёмнике всё". Два пути проходят мимо существующих защит: `mirror_hard` на цели меньше `DELETE_COUNT_FLOOR` (50 файлов) и `mirror_safe`, который ratio-guard не проверяет вообще - ничего не удаляется, но всё дерево приёмника уезжает в `_audion_quarantine` по сети.

Защита - файл-якорь `.audion-anchor`, по той же схеме и по той же причине, что `.stfolder` у Syncthing.

```
{
  "name": "projects_to_dacha",
  "source": "N:\\Projects",
  "target": "D:\\Backup\\Projects",
  "mode": "safe",
  "mirror": true,
  "operation_policy": "mirror_safe",
  "anchor": true,
  "note": "N: is a mounted remote root",
  "sync_kind": "backup"
}
```

Как это работает:

1. Якорь создаётся один раз и только явной командой. Во время прогона он никогда не создаётся сам.

```
runtime\python.exe system_core\main.py anchor N:\Projects
```

2. При `"anchor": true` план проверяет якорь в **обоих** корнях до сканирования. Если якоря нет, поднимается `SourceUnreachable` - отдельная ошибка, не `FileNotFoundError`. Разница между *недоступен* и *пуст* делается до того, как построен хоть какой-то план.
3. Перед разрушительной фазой `apply` проверяет якорь ещё раз: между preview и запуском монтирование могло отвалиться.
4. Сам якорь исключён из хеширования и из diff так же, как служебный манифест, поэтому он не копируется и не удаляется.
5. `anchor` по умолчанию `false`, поэтому существующие профили работают без изменений. Для разовых запусков без профиля есть флаг `--anchor`.

Якорь нужен там, где корень приходит из сети. Для двух локальных дисков он лишний.

Проверка перед запуском

Над списком полей команды GUI показывает карточку `ПРОВЕРКА ПЕРЕД ЗАПУСКОМ`. Это три проверки рядом с кнопкой `ЗАПУСТИТЬ`:

1. **Якорь** - есть ли `.audion-anchor` в обоих корнях, или для этой пары он отключён.
2. **Место** - сколько байт запланировано записать против свободного места на приёмнике.
3. **Удаления** - сколько файлов будет удалено или отправлено в карантин, абсолютным числом и долей от приёмника. Это те же цифры, на которых срабатывает ratio-guard, показанные *до* исключения, а не только внутри него.

Якорь и свободное место считаются вживую при каждом показе карточки, а плановые цифры берутся из последнего `Compare` или `Dry run` по этому же маршруту. Пока плана нет, карточка так и говорит.

Строка `PREVIEW` появляется для зеркальных политик и показывает токен подтверждения вида `DELETE 53289F2276BC`. Он привязывает запуск к тому просмотру, который оператор видел: если между preview и apply маршрут или цифры изменились, `apply` откажется работать с `PreviewConfirmationMismatch`. В CLI то же самое делает `--expect-confirmation`.

Политики apply

`copy_update` подходит для обычного copy/sync.

`mirror_safe` подходит для ситуаций, где нужно убрать target-only из рабочего вида, но сохранить их в `_audion_quarantine`.

`mirror_hard` подходит для настоящего backup mirror, где target-only нужно удалить.

Для filtered hard mirror CLI может требовать `--yes-filtered-hard`. Для большого delete ratio может требоваться `--yes-delete-ratio`.

Рекомендуемый цикл настройки профиля

1. Скопировать один из example-профилей.
2. Изменить `name`, `source`, `target`.
3. Выбрать `sync_kind`.
4. Выбрать `operation_policy`.
5. Добавить include/exclude presets.
6. Добавить `exclude_dirs` для caches/build/temp.
7. Если хотя бы один корень смонтирован по сети - поставить `"anchor": true` и создать якорь командой `main.py anchor`.
8. Запустить `pairs`.
9. Запустить compare или dry-run.
10. Проверить карточку `ПРОВЕРКА ПЕРЕД ЗАПУСКОМ`.
11. Только после этого запускать apply.