RCLONE В РАЗДЕЛЕ ТРАНСФЕР ДАННЫХ

Про сам перенос — операции, выбор движка, маски, упаковку — написано в TRANSFER_RU.md. Здесь то, что вокруг: установка rclone, подключения, хранилище, конфиг и его шифрование, диагностика.

RClone нужен для native backends: Yandex Disk, Google Drive, SFTP, WebDAV, S3-совместимые хранилища и другие remote, которые он поддерживает. Robocopy не вытеснен — он стал вторым движком того же раздела и берёт на себя папки и шары.

Установка

В GUI установка разделена на две маленькие кнопки в строке `Installer RClone`:

- `System` запускает `install\Install-System-Rclone.cmd` и ставит user-scope binary в `%LOCALAPPDATA%\Programs\rclone`;
- `Portable` запускает `install\Install-Portable-Rclone.cmd` и ставит/обновляет portable binary в `Tools\rclone`.

Оба installer-а:

1. Скачивают latest stable Windows AMD64 ZIP с `https://downloads.rclone.org/rclone-current-windows-amd64.zip`.
2. Распаковывают архив.
3. Находят `rclone.exe`.
4. Проверяют `rclone.exe version`.
5. Не перезаписывают `rclone.conf`.

`System` полезен как обычный инструмент Windows и fallback через user `PATH`. `Portable` полезен для переносимого комплекта проекта. Выбор конфигурации (`Portable`, `System`, `Custom path`) находится не в installer-е, а в строке `Configuration`; подробная механика описана в `RCLONE_PORTABILITY_RU.md`.

Ожидаемый portable путь:

```
Tools\rclone\rclone.exe
```

Поиск rclone.exe

Порядок discovery:

1. `Tools\rclone\rclone.exe`
2. `tools\rclone\rclone.exe`
3. `runtime\rclone\rclone.exe`
4. `runtime\tools\rclone\rclone.exe`
5. `system_core\rclone.exe`
6. `rclone.exe` из `PATH`
7. `rclone` из `PATH`

Если файл не найден, GUI предлагает установить rclone или положить его в `Tools\rclone\rclone.exe`.

Первый запуск

1. `Version` - убедиться, что binary работает.
2. Для Yandex/Drive/OneDrive/Dropbox/Box/Mega выбрать `Провайдеры OAuth`, задать имя remote и provider; GUI откроет отдельное служебное окно RClone для OAuth/провайдерской авторизации.
3. Для сервера выбрать `Create/update SFTP remote`, заполнить `user@host:port`, путь сервера и метод auth.
4. Если нужен backend вне GUI-шаблонов, использовать `Config`.
5. `List remotes` - увидеть configured remote names.
6. `About remote` или `Test remote` - проверить доступ.
7. `Copy endpoint route` или старые `Upload copy` / `Download copy`.
8. `Check endpoint route` или `Check one-way` - проверить соответствие.

То есть пользователь почти не лезет в RClone вручную: GUI строит `remote`, route и preview. Видимая CLI-консоль остаётся там, где это честнее и безопаснее: OAuth wizard, полный `rclone config`, официальный `rclone gui`.

Режимы

Вверху RClone-раздела есть отдельный toggle слоя:

- `Cloud` - cloud remote операции: upload/download/check, list, mkdir, about, OAuth и config;
- `Server` - SFTP/server операции: создание SFTP remote, endpoint route copy/check, диагностика.

В RU-layout эти вкладки показаны как `Облако` и `Сервер`.

Внутри `Cloud` и `Server` остаются компактные вкладки mode-кнопок. На экране остаются только команды выбранного слоя и вкладки, активная команда подсвечивается. Tooltip каждой mode-кнопки объясняет pipeline, например `Workbench Источник -> remote:path -> rclone copy` или `user@host:port -> rclone config create/update`.

`version` запускает `rclone version`.

`config` запускает `rclone config` напрямую в новой видимой консоли. Это нужно для OAuth и interactive setup. Токены не должны попадать в GUI log.

`gui` запускает `rclone gui` напрямую в новой видимой консоли. Проект не добавляет `--no-auth`.

`list_remotes` запускает `rclone listremotes`.

`about_remote` запускает `rclone about <remote>:<path>`.

`list_remote_path` запускает `rclone lsf <remote>:<path>`.

`mkdir_remote` запускает `rclone mkdir <remote>:<path>`.

`remote_test` запускает `rclone lsf <remote>:<path> --max-depth 1`.

`remote_auth_wizard` запускает целевой мастер:

```
rclone config create <remote> <backend>
```

Поддерживаемые GUI-backend presets: `yandex`, `drive`, `onedrive`, `dropbox`, `box`, `mega`. Для редких backend-ов остаётся `Config`.

`remote_create_sftp` создаёт или обновляет SFTP remote из GUI-полей:

```
rclone config create server sftp host 203.0.113.10 user user port 22 ...
```

Если remote уже существует, GUI использует `config update`, чтобы не плодить одноимённые настройки. Auth-методы: `ssh-agent`, `password`, `key_file`. Пароль не сохраняется в cache/history: перед записью в rclone config он проходит через `rclone obscure -` по stdin.

Перенос — один режим `transfer_run`, а не пять кнопок под каждое направление. Команда следует из выбранной операции, а движок — из пары:

```
rclone copy   <источник> <назначение>     Односторонняя
rclone sync   <источник> <назначение>     Зеркало
rclone move   <источник> <назначение>     Перенести
rclone bisync <источник> <назначение>     Двусторонняя
rclone check  <источник> <назначение>     Сверить
```

Источником и назначением может быть папка на этой машине, шара, сохранённое подключение или адрес `http(s)://`. Назначений может быть несколько сразу. Если пара — две локальные папки и хеши не нужны, поедет Robocopy: на дереве папок он быстрее и переносит NTFS-атрибуты, которые rclone на Windows теряет.

Подробно про операции, движки, маски и упаковку — в TRANSFER_RU.md.

Параметры remote

`rclone_remote_name` вводится без `:`. Дефолт для OAuth/обычных cloud-сценариев: `drive`.

`rclone_remote_path` вводится без leading `:`. Backslash нормализуется в slash. Пустой path означает root remote.

Финальный spec строится как:

```
remote:path
```

Endpoint route builder

`SOURCE ENDPOINT` и `TARGET ENDPOINT` - парные toggle-панели внутри RClone-раздела. По умолчанию это remote -> remote маршрут `drive:Backup -> server:/home/user/backups`.

Для каждой стороны выбирается:

- `Workbench Источник` - текущий локальный Source;

- `Workbench Назначение` - текущий локальный Target;
- `Saved remote` - сохранённый `remote:path`.

Preview показывает и человекочитаемую подпись, и финальный spec. Например:

```
SOURCE: drive:Backup
TARGET: server:/home/user/backups
COMMAND: rclone copy drive:Backup server:/home/user/backups ...
```

Если выбран Workbench endpoint, локальный путь передаётся RClone как отдельный subprocess-аргумент. Preview кавычит Windows-пути только для визуального контроля.

Authorization builder

`Remote authorization` - отдельная toggle-панель для создания/обновления remote.

SFTP fields:

Поле	Пример	Назначение
`Remote name`	`server`	Имя в `rclone.conf`, потом используется как `server:/path`.
`user`	`user`	SSH/SFTP пользователь.
`host`	`203.0.113.10`	IP или DNS host.
`port`	`22`	TCP port.
`Remote path`	`/home/user/backups`	Человекочитаемый preview и подсказка для route target.
`Auth`	`ssh-agent`, `password`, `key_file`	Метод авторизации.
`known_hosts`	`%USERPROFILE%\.ssh\known_hosts`	Проверка host key; особенно полезно для password-auth.

Для password-auth пароль вводится в masked field и не попадает в `config\rclone_command_cache.json`. В логах и preview используется redaction.

S3-совместимые хранилища: Cloudflare R2, Wasabi, Selectel

Отдельный режим, потому что OAuth здесь нет вовсе. R2 выдаёт пару ключей, и каждый запрос подписывается ими по схеме AWS Signature v4 — ни браузера, ни редиректа, ни обновляемого токена.

Поле	Пример	Назначение
`Имя remote`	`r2`	Имя в `rclone.conf`, потом `r2:bucket/path`.
`Хранилище`	`Cloudflare R2`	Пять кнопок; определяет `provider` и регион по умолчанию.
`Адрес хранилища`	`https://<account_id>.r2.cloudflarestorage.com`	Обязателен для всех, кроме Amazon.
`Файл с ключами`	`%USERPROFILE%\.aws\credentials`	Путь к файлу на этой машине. Пусто — rclone ищет там, где ищет всегда.
`Секция в файле`	`r2-releases`	Имя в квадратных скобках внутри файла. Пусто — `default`.

Ключи в программу не попадают и в проекте не хранятся. В `rclone.conf` уходит `env_auth true` плюс путь к файлу и имя секции — ссылка, а не сам секрет. Причина не в общей осторожности: проекты переносные и ездят на съёмном диске, а диск, который ключей никогда не носил, не может их потерять.

Формат файла — как у AWS, одна секция на хранилище:

```
[r2-releases]
aws_access_key_id = <Access Key ID>
aws_secret_access_key = <Secret Access Key>
```

Где взять ключи. Прямая ссылка, минуя меню:

```
https://dash.cloudflare.com/?to=/:account/r2/overview
```

`:account` Cloudflare подставляет сам. Дальше: Account Details → рядом с API Tokens кнопка **Manage** → **Create Account API token**.

Права выдаются четырьмя уровнями. Формулировки Cloudflare:

Уровень	Что даёт
`Admin Read & Write`	create, list, and delete buckets, edit bucket configuration, read, write, and list objects
`Admin Read only`	list buckets and view bucket configuration, read and list objects
`Object Read & Write`	read, write, and list objects in specific buckets
`Object Read only`	read and list objects in specific buckets

Берите `Object Read & Write` и сужайте до нужных бакетов.

Admin добавляет создание и удаление бакетов, изменение их настроек (включая публичный доступ) и просмотр списка всех бакетов аккаунта. В ежедневной выкладке не нужно ничего из этого, а ключ живёт на ноутбуке. Утёкший объектный ключ даёт писать в один бакет — плохо, но конечно, и лечится выпуском нового. Утёкший админский даёт удалить бакет целиком одним вызовом и открыть приватное наружу. Разница не в вероятности утечки, а в размере последствия.

Бакеты и объекты

Два слова, оба стандартные, и их стоит развести один раз.

Бакет — ёмкость, ведёт себя как диск. Их несколько на аккаунт, вкладывать друг в друга нельзя, заводится однажды: сделали `releases` — больше не возвращаетесь. Selectel в своей документации зовёт то же самое контейнером (наследство OpenStack Swift), у Cloudflare, Яндекса и VK — бакет.

Объект — то, что внутри. Каждый ваш файл там официально объект, отсюда и название всего класса: объектное хранилище.

Папок нет вовсе. У объекта есть только имя, а косые черты в нём — обычные символы. `releases/3.0.1/setup.exe` — это одно длинное имя, а не три уровня. Панель Cloudflare рисует дерево для глаз, внутри плоский список.

Отсюда главное: **выкладка нового релиза ничего не создаёт**. Пишете объект с именем `releases/3.0.1/setup.exe` — он там. Ни папку `3.0.1`, ни что-либо ещё заводить не нужно, потому что заводить нечего.

Поэтому узкий ключ ничего не ломает. Следствия ровно два, оба безобидные:

- адресуйте `r2:releases/путь`, а не просто `r2:` — списка бакетов узкому ключу не видно;
- новый бакет заводится в панели Cloudflare, а не из окна. Раз в год, если вообще.

Документация: https://developers.cloudflare.com/r2/api/tokens/

Три вещи, на которых спотыкаются:

- Общая страница API-токенов Cloudflare — **не та**. Там сотня галочек, и ни одна не нужна: тот API управляет аккаунтом, а не данными. Ключи для данных выдаются только внутри раздела R2.
- Cloudflare называет выданное то **Access Key ID / Secret Access Key**, то **Client ID / Client Secret**. Это одно и то же.
- **Secret Access Key показывается один раз.** Не записали — придётся выпускать новый.

Несуществующий файл ключей отвергается сразу, при сборке команды. Иначе remote создался бы, выглядел здоровым и упал при первой заливке с ошибкой авторизации, которая о пропавшем файле не говорит ничего.

Для OAuth/provider auth GUI открывает отдельное окно RClone. Пользователь видит CLI, но назначение окна однозначное: пройти авторизацию выбранного provider-а и вернуть управление Workbench.

В режиме `Провайдеры OAuth` дефолтный provider - Google Drive, дефолтное `Имя remote` - `drive`. Смена provider-а автоматически меняет `Имя remote`, если оно ещё дефолтное (`drive`, `yandex`, `dropbox`, `server` и т.п.). Custom name сохраняется и не перетирается.

SFTP `user` и `host/IP` имеют history dropdown последних успешных операций. Кэш: `config\rclone_endpoint_history.json`. Он хранит только `users` и `hosts`, без паролей, портов и путей. Очищается кнопкой очистки в RClone auth-панели.

Tooltip contract для auth wizard

Подсказки в RClone-слое не должны быть общими фразами. Они объясняют, что произойдёт после `ЗАПУСТИТЬ` и что вводить.

`Remote name`:

- вводится только имя без `:`;
- пример для сервера: `server`;

- пример для Google Drive: `drive`;
- после создания используется как `server:/path` или `drive:Folder`.

`Backend` / `ЗАПУСТИТЬ` для OAuth:

- откроется отдельное видимое окно RClone с командой вида `rclone config create drive drive`;
- remote name уже задан в GUI, повторно вводить его не нужно;
- если RClone спрашивает `Client ID` и `Client Secret`, для обычного сценария оставлять пусто и нажимать Enter;
- advanced config - оставить default/No;
- auto config/browser - выбрать Yes/default;
- в браузере войти в provider и разрешить доступ;
- после success использовать remote в route как `drive:<путь>`.

SFTP tooltips:

- `user` - только SSH user без `@`, например `user`;
- `host` - IP/DNS без `ssh://` и без пути, например `203.0.113.10`;
- `port` - TCP port, обычно `22`;
- `Remote path` - абсолютный путь на сервере, например `/home/user/backups`;
- `Auth=password` - пароль вводится в masked field, отправляется в `rclone obscure -` через stdin и не сохраняется в cache/history;
- `Auth=ssh-agent` - пароль в GUI не вводится;
- `Auth=key_file` - вводится путь к private key;
- `known_hosts` - рекомендуемый путь для проверки host key.

В SFTP auth-панели есть маленькие кнопки переноса remote в `SOURCE ENDPOINT` или `TARGET ENDPOINT`. Они записывают, например, `/home/user/backups` как `server:/home/user/backups` в route builder.

Flag profiles

`safe_cloud_upload` (`safe_yandex_upload` в command/cache id):

```
--transfers 1 --checkers 1 --retries 10 --low-level-retries 50 --retries-sleep 10s --timeout 10m --contimeout 30s --stats 1s --progress
```

`unstable_mobile`:

```
--transfers 1 --checkers 1 --retries 20 --low-level-retries 80 --retries-sleep 15s --timeout 15m --contimeout 45s --stats 1s --progress
```

`balanced`:

```
--transfers 2 --checkers 4 --retries 5 --low-level-retries 20 --retries-sleep 10s --timeout 10m --contimeout 30s --stats 1s --progress
```

`fast_lan_to_cloud`:

```
--transfers 4 --checkers 8 --retries 3 --low-level-retries 10 --timeout 10m --contimeout 30s --stats 1s --progress
```

`rclone_bwlimit` добавляет контролируемый `--bwlimit`, например `10M`. В GUI это full-width speed-builder под profile/log flags: `Без лимита`, `Выбор скорости` (`1..9`, `10..90`, `100`, `150`, `200`, `300`, `500`, `900`) или `Своя скорость` со спиннерами. Произвольное поле extra flags намеренно не добавлено.

Прогресс, ретраи и проценты

Все transfer-профили включают `--stats 1s --progress`.

GUI парсит строки rclone и обновляет:

- процент из `Transferred`;
- текущий transferred text;
- speed;
- ETA;
- `Checks`;
- `Retries`;
- `Errors`;
- статус операции;
- progress bar.

Это важно для медленных каналов: даже если файл большой и копируется долго, видно, что поток живой, сколько retry накопилось и как меняется ETA.

Если процент неизвестен, терминал всё равно остаётся главным источником правды: rclone пишет stats каждую секунду.

Логи и redaction

Каждый запуск получает log file:

```
logs\YYYYMMDD_HHMMSS_rclone_<mode>.log
```

GUI добавляет:

```
--log-file <path>
--log-level INFO
```

`DEBUG` включается только выбором `rclone_log_level=DEBUG`. Этот режим используйте только локально и временно: raw rclone log может содержать чувствительные детали remote/request.

Display/log redaction скрывает очевидные token/password/client_secret поля. GUI не вызывает `rclone config show`.

Portable/System config

В `ОБЛАЧНЫЕ ОПЕРАЦИИ` эти компактные линейки находятся во вкладке `Настройки` в режиме `Настройка`:

`Installer RClone`: `System` ставит user-scope RClone в `%LOCALAPPDATA%\Programs\rclone` и добавляет его в user PATH, `Portable` обновляет только `Tools\rclone`. Оба installer-а не перезаписывают `rclone.conf`.

`Configuration`: содержит scope-переключатель `Portable`, `System`, `Custom path`, поле custom config path и действия `Import`, `Export`, `Doctor`. `Custom path` - это явный путь к `rclone.conf`, не установка RClone.

`Import` копирует `%APPDATA%\rclone\rclone.conf` в `config\rclone\rclone.conf` с backup, `Export` копирует portable config обратно в system config с backup, `Doctor` read-only проверяет

active config, remotes и абсолютные пути внутри config. Подробная механика: RCLONE_PORTABILITY_RU.md.

В `Portable` scope каждая команда строится с:

```
--config <ROOT>\config\rclone\rclone.conf
--cache-dir <ROOT>\tmp\rclone-cache
```

В `System` scope GUI не добавляет `--config`, и RClone использует обычный `%APPDATA%\rclone\rclone.conf`.

Конструктор команд

RClone preview показывает:

- `EXE`;
- `ROUTE`;
- `CONFIG`;
- `CACHE`;
- `SOURCE` / `TARGET`, если выбран route-mode;
- `LOG`;
- команду целиком в Windows-quoted виде.

Кнопки:

- `Pin` - закрепить текущую конструкцию.
- `Unpin` - снять закрепление.
- `Delete` - удалить из cache.
- Dropdown - выбрать сохранённую конструкцию.

Файл cache: `config\rclone_command_cache.json`.

Что ограничено специально

Разрушительный `rclone sync` не входит в обычный список профилей передачи. Он доступен только как отдельное `Защищённое зеркало` под предпросмотром маршрута: GUI показывает фактические `SOURCE` и `TARGET`, строит `rclone sync <SOURCE> <TARGET>`, предупреждает об

удалении файлов, существующих только в приёмнике, и требует вручную ввести `SYNC` в диалоге подтверждения.

Так точное зеркало остаётся доступным для осознанных сценариев, но его нельзя случайно выбрать вместо безопасного `rclone copy`.

Нет arbitrary extra flags поля. Флаги идут через controlled profiles.

Нет project-stored OAuth tokens.