Audion Sync Tools — user guide

Русский · Changelog · SYNC ALL

Contents

- What it is
- The program and the data
- Starting on a new machine
- The window
- The tunnel client
- Port forwarding: the desktop and everything else
- From the console
- Where things live
- Sharp edges
- OpenVPN from machine data

What it is

The program that keeps a copy of MAIN VPS on this machine, together with everything without which that copy is useless: the keys, the tunnel client, the access to the home machines. Plus a snapshot of the private Build-stack archives into the R2 bucket.

MAIN VPS takes its own backup — every night at 01:00, by the `audion-backup.timer` unit — and puts it into `r2:audion-backup/daily/YYYY-MM-DD`. Only the machine that holds the data folder can fetch that snapshot onto a local disk: MAIN VPS will not knock on a home computer. That fetching is what this program does.

The program and the data

Deliberately apart.

The program is the `Audion Sync Tools` folder: the window, the scripts, its own `rclone`, its own keys to the R2 bucket, the tunnel client, the toolkit for reaching the machines. Nothing here would be painful to lose — it is all restored from the repository.

The data is the neighbouring folder, named after the machine: `Audion Backup Laptop`, `Audion Backup Work`, `Audion Backup Base`. Snapshots, mirrors of MAIN VPS's code, keys, tokens. Not a single executable file lives there.

The folder's name is the machine's name, and it is visible from outside. There is no separate profile inside: the folder is its own profile. While the data folder was named the same everywhere, profiles had to be told apart by names within it — that is gone.

How the program finds it:

1. The **pointer** `config\profile.json` — the path a person once chose.
2. The **only neighbouring** `Audion Backup*` — on the machine itself there is one.
3. On the **first run**, when there is no pointer yet, the program asks. Even when there is a single folder beside it: that folder holds keys, addresses and the router password, and settling into it on a guess is not acceptable.

At home there may be several — the data folders of the other machines lie there. Then the choice is shown as a list, while console scripts take the first and say so aloud, naming the `-Root` switch.

No drive letter is hard-coded anywhere: on a machine where the tools live on `S:`, nothing needs editing.

The independence is not decorative. The mirror used to take `rclone.exe` and the R2 bucket keys from the publisher, and without that neighbour it did not come up at all. It has to come up first on an empty machine, before everything else.

Starting on a new machine

1. Copy both folders — the program and this machine's data. The keys travel inside the data, in `Tunnel\SSH` (the golden copy) and `Tunnel\Export` (Export Keys collected on the previous machine).
2. Start the program and point at the data folder — it asks on the first run, and downloads there the tools a release comes without.
3. `SETUP` → `IMPORT KEYS`. It lays the keys into the three places the programs look for them: the `%USERPROFILE%\.ssh` profile, the machine data's `config\rclone`, and `core.sshCommand` in the mirror's working copies. It also gives the private copies the permissions ssh insists on, and

repairs the paths in `~\.ssh\config` that lead nowhere. Server pairs missing from Export Keys are taken from `Tunnel\SSH\servers`. Existing private keys are kept and their permissions checked; `-Check` only reports the state.

4. `STORAGE` → `SYNC ALL` — the first filling of the library. After that the same button or `sync-backup.cmd` refreshes it: the program creates no Windows Scheduler tasks, and MAIN VPS takes its nightly snapshot by its own timer.

If the machine already carries someone else's VPN on AmneziaWG — Red Shield, say — no client of our own is needed: the tunnels will run on that one. The `WIREGUARD` button will show this and install nothing.

To confirm it all landed: `Tunnel\check-tunnel-client.cmd` shows the tunnel client and the routes, `sync-backup.cmd -WhatIf` shows the Git catalog and update plan.

The window

Opened by `Start.exe`. It asks for administrator rights once: they are needed to work with the system network connections — the tunnel service, the state of the WireGuard interfaces and their handshakes, which Windows gives only to an administrator. You may refuse: without the rights only the tunnels and `SSHD` do not work, and the title of the window says NO ADMINISTRATOR RIGHTS.

`Start.exe -Root "data folder"` opens the window on the folder named — for this run: the pointer of the program stays as it was. A folder that is not there is not replaced by a guess: the window opens with no data folder, and the log says which one it was given.

The layout: **two file panels are always visible**, and the section strip at the top changes only two things — what the panels show and which row of buttons appears. A button therefore stops being an incantation: in the `SSH KEYS` section the golden copy stands on the left, the profile on the right, and after a button has run you see what changed there.

To the right of the panels is the log, shared by every section. Between the panels runs a narrow strip of buttons, and at the bottom the row of file operations with their keys.

The window needs no PowerShell of the system: its own PowerShell 7 lies next to it, in `App\AudionPs.exe`, and runs every script and wrapper as well.

Captions below are given as the window shows them in English; the `RU / EN` button of the header switches the language.

The first start

A release comes without the tools and without a pointer to the data folder. Both are asked in one window as soon as the main one is up:

- **DATA FOLDER** — the `Audion Backup …` folder found beside the program is offered; the other button opens a browser: the folder may lie anywhere.
- **TOOLS** — what is missing from the program folder and what does not work without it. The button downloads the missing ones, about 100 MB.

There are six tools: `rclone` (without it the panels read neither machines nor R2 buckets, and neither copying nor the mirrors run), `cloudflared` (the route through Cloudflare), Windows Terminal (the terminal to the machines), PeaZip (packing and unpacking), and the editors Microsoft Edit and micro (F4). The window of the first start comes back at every start while a tool is missing: a program that cannot read a machine must say why, not stand with empty panels.

The header

On the left the version and the **DATA** badge — the name of this machine's data folder. A click opens it in Explorer; another one is chosen in `SETTINGS → DATA FOLDER → CHOOSE`.

In the middle is the probe line: which route answered in how long, and which way the window will go. See "Routes to the machines".

On the right:

- **STREAMS** — how many transfers run at once; `−` and `+` halve and double the number. A slow channel is better off with fewer, many small files with more.
- **RU / EN** — the language of the window.
- The sun button — the theme: Audion Vault Dark (the default), Audion Dark, Audion Code Dark, Audion Light.

All of it is remembered between runs.

Sections

Section	In the panels	In the buttons
SETUP — first setup, a button at the top right	the profile, `config\rclone`, the golden copy, Export Keys, the data folder, LOCAL	the steps of setting up: addresses, keys, SSHD, SSH TUN, RESTORE, RCLONE.CONF

Section	In the panels	In the buttons
SSH KEYS — machine keys	the golden copy, Export Keys, the profile, `config\rclone`	LIST, SSH-COPY-ID, REVOKE, EXPORT KEYS, R2 CREDS
TUNNEL — network setup	the machines, MAIN VPS, LOCAL	the probe and DIAGNOSTICS, CLOUDFLARED, OPENVPN, the router, DNS, WIREGUARD, WG BASE-ROUTER
REMOTE — terminal and RDP	the machines, MAIN VPS, LOCAL	terminals of machines and routers, RDP, ROUTER WEB, SOCKS, the choice of a terminal program
STORAGE — snapshots and R2	the nightly snapshots, the data folder, five R2 buckets, MAIN VPS, SECOND VPS, LOCAL	SYNC ALL, SYNC SELECTED, DRY RUN, PRIVATE BACKUP
CLOUDS — cloud drives	the machines, LOCAL and the clouds you connected	connect a cloud, sign in again, rename, disconnect, check, the keys of the clouds
SETTINGS — a button at the top right	the settings in place of the panels	the data folder, tools and versions, the editor of F4, the font, translucency, sizes
ABOUT — a button at the top right	the program in place of the panels	the version, the folders, the documents, the link to the releases

The pin button of the strip between the panels pins a section: at the next start the window opens on it. What does not fit across the width of a row moves to a second line.

The wizards ask inside the window — with a form, a table or a choice of several actions — and the work is done by a script of `Engine\Scripts`: the same one a console calls. Only what holds a conversation of its own opens in a separate console: `SSHD`, `RESTORE`, the installer of WireGuard, raising the tunnel, the talk with the router, OpenVPN, the forwards and the sign-in to a cloud.

Indicators

The buttons that set something up carry an indicator on the left. It has three meanings, and red is not among them:

- **grey** — it is not there: not set up, not installed, no address written;
- **amber** — it is there but not proven in practice: a file in its place does not yet mean it lets you in;
- **green** — proven, or there is nothing to prove.

The proof is a panel that was read. A machine was read — the keys and `RCLONE.CONF` go green; an R2 bucket was read — so does the R2 pair. For a service green means it is running, for `SSH TUN` — that its own ssh holds the channel. The machine buttons of the REMOTE section have four meanings — see "Machines".

The state is read when the window opens, when the queue of jobs ends, when the console of a wizard closes, and by the `STATE` button. The network is not touched by that: the machines are probed by `PORT 22 TCP PING` and once by the window itself at its start.

Setting up — the SETUP and SSH KEYS sections

The setting-up buttons answer one question: what is still missing on this machine. They answer by the data folder, not by what the window remembers: which key is this machine's the program finds out itself — its public half lies in CLIENTS, the private one in the profile.

Section → group	Button	Lit when	Green after
SETUP → STEPS	`ADDRESSES`	BASE and LAPTOP have addresses in `targets.json`	a machine answered the probe
	`SSH-KEYGEN`	this machine's key is in the profile	a machine or VPS panel was read
	`IMPORT KEYS`	the profile holds the pair of a machine of CLIENTS	a machine or VPS panel was read
	`SSH-KEYSCAN`	the host key list exists	a remote panel was read
	`SSHD`	the sshd service is installed	the service is running
	`SSH TUN`	REVERSE SSH TUN is set up	its own ssh holds the channel
SETUP → OTHER	`RCLONE.CONF`	`rclone.conf` has no unfilled placeholders left	a remote panel was read
	`ROUTER PASS`	the router password is stored	at once
	`RELEASE`	the folder is chosen and holds the private archives	at once

Section → group	Button	Lit when	Green after
	`REMOVE TASK`	a daily task left by earlier versions remains — the button shows only then and deletes it	—
SSH KEYS → EXTERNAL	`R2 CREDS`	`r2.credentials` is in place	an R2 bucket was read in a panel
SSH KEYS → FOLDER	`DATA`	there is a golden copy `Tunnel\SSH` or Export Keys in `Tunnel\Export`	at once
TUNNEL → LOCAL	`WIREGUARD`	the machine has a tunnel client	at once
	`WG BASE-ROUTER`	the config of the tunnel lies in the data	the tunnel service is installed

Three buttons of SETUP carry no indicator. `INSTALL` puts `cloudflared.exe` into `Tunnel\Bin` and rewrites its own block in `~\.ssh\config` — needed only where there is no config at all. `RESTORE` restores the machine from the chosen data folder after Windows was reinstalled: the order is in RESTORE_EN.md. `OVPN PATH` chooses the client and the profile of OpenVPN.

In the SSH KEYS section, besides the ones named:

- `LIST` — every key in one list: what the key is, where it lies, whether its private half exists and on which machines it is registered. Below it — the working files of this machine.
- `SSH-COPY-ID` — adds the public keys of CLIENTS to `authorized_keys` of every machine through access that already works.
- `REVOKE` — removes a public key by its name from every machine within reach. It does not take the last key off a machine: nobody could sign in after that.
- `EXPORT KEYS` — collects the keys of this machine into `Tunnel\Export`; `CHECK` shows the layout and touches no file.

`ADDRESSES` opens the machine table. The `SECOND VPS` row is the second server: its address and port go in the row, and three fields below the table take its names — the host in `~\.ssh\config`, the same host over cloudflared and the key name. Older installations have names of their own; the window moves them over from `~\.ssh\config` at start, and the fields let you correct them. After writing, the window checks `~\.ssh\config` and says when a name still has to be written there with `INSTALL`.

The `MAIN VPS` row holds its name for the straight road, and the field `MAIN VPS: NAME FOR CLOUDFLARED` below the table the name cloudflared goes to it by. Both are addresses of the household: they live in `targets.json` of the data folder, the program holds neither. Until the field is filled in, the window and the tunnel client take the names from this machine's `~\.ssh\config` - the hosts `audion-vps-direct` and `audion-vps` - and the field starts from them.

`R2 CREDS` asks for the id and the secret — the Cloudflare panel issues them under R2 - Manage API tokens — and writes them into the file `rclone` reads. Right after writing, the window asks R2 for its bucket list: a key without a check is a promise, not access. The values never reach the log.

`SSH-KEYSCAN` collects host keys into the machine data's `config\rclone\known_hosts`. Checks run in parallel in a separate process, keeping the window responsive. The direct pass waits up to five seconds; unanswered addresses with configured SSH routes get another pass of up to five seconds. `STOP` ends the worker and its children. Unreachable hosts keep their previous keys. Previously stored keys also remain when a change is detected: verify the fingerprint and repeat with Ctrl or `Runs\ssh-keyscan.cmd -Force`. `~\.ssh\known_hosts` is left alone — `ssh`, Windows Terminal and Tabby use it. The scan includes SECOND VPS and its SSH port from `ADDRESSES`. This panel list does not replace confirmation of the first connection through regular SSH.

The colour bars

A thin colour bar sits to the left of each button group. It tells you what kind of thing you are about to change before you even read the captions.

- **Gold** — keys: pairs, authorized_keys, known_hosts, R2 creds
- **Sea Blue** — links: cloudflared, REVERSE SSH TUN, WireGuard, routes, the router
- **Bottle** — machines: terminal, RDP, receiving files
- **Mango** — tools: the tunnel client, rclone, the terminal, updates
- **Teal** — clouds
- **Plum** — storage: snapshots, R2 buckets, RELEASE
- **Grey** — data and setup: folders, addresses, the SETUP steps

Separately: danger buttons (REVOKE, DISCONNECT) are painted **Burn Orange** — they delete or revoke.

Colours and themes

The sun button of the strip opens the choice of a theme: Audion Vault Dark, Audion Dark, Audion Code Dark and Audion Light. The choice is remembered.

File names are coloured by `App\config\colors\themes.json`: the colours and the order of the rules. What counts as video, a document or an archive is said by `App\config\mask-groups.json` next to it — the same families the mask window offers. Rules are applied top down and the first match wins, so a freshly changed archive is coloured as fresh, not as an archive. Age counts in calendar days, so yesterday's file is "no older than a day" even at thirty hours.

The window's own colours — frames, buttons, list background — live separately, in the themes of the window, and do not touch the colours of the rows.

Panels

Above each panel are the source buttons: what it shows. A local disk, a machine over ssh, an R2 bucket and a cloud are the same thing to this program — one `rclone` reads them all, and copying works between any two sides. The caption of a machine carries the route it is reached by right now: `BASE · SSH SECOND VPS`. The list of sources also opens from the keyboard: Alt+F1 for the left panel, Alt+F2 for the right one.

Below them is the path trail: clicking a segment returns to that level. At the bottom of the panel is its status line: how many objects, and for a disk — how much of it is free.

A click on a column heading changes the order of the rows, a second one turns it over; the arrow shows what they are ordered by. Folders always stay on top.

The frame of the active panel is highlighted. The active one is the source for copying and synchronisation, the other one the destination; a panel becomes active on any touch, and Tab moves the cursor into the other one.

Marks. A marked row gets an orange dot in the narrow first column. Space or Insert marks the row and moves the cursor down, Space on a folder also counts its size (below, "Sizes"), Ctrl+A marks everything — and with everything marked takes the marks off — and `*` of the numeric keypad inverts the marks. The mask window marks by masks. File operations take what is marked, and with no marks — the row under the cursor.

The quick filter. The F key opens a line above the list: what you type narrows the list by name. Esc takes the filter away.

Sizes. The column shows KB, MB and GB or exact bytes — the switch is in SETTINGS. Space, marking a folder, counts its size, as in Total Commander: while it counts the column shows "…", then the number, and it goes into the sum "MARKED" under the panel. "?" — the count had no answer, Space on the folder counts again. Two folders are counted at a time, leaving the folder stops the counts begun; drives in the list of drives are not counted, Insert marks without counting. Shift+Space counts the exact

size of the row under the cursor, a folder too: by the disk on this machine, through `rclone size` on a far side. The answer stands in the status line of the panel.

Rows are coloured by age and by file type — the rules are in `App\config\colors\themes.json`.

The strip between the panels

From top to bottom:

- the arrow up — one level up in the active panel (Backspace);
- the round arrow — read the active panel again (Ctrl+R);
- the asterisk — mark rows by masks;
- the magnifier — the search (Ctrl+F, Alt+F7);
- the two arrows — swap the panels, sources and folders alike (Ctrl+U);
- the pin — pin the section;
- the network sign — routes by machine: each machine gets its own;
- the power button — switch the WG tunnel to the router of BASE on and off. Green while the tunnel is up; amber when its service stands but the tunnel is down; off where no tunnel is installed. With Ctrl held the tunnel is raised in spite of overlapping networks;
- the two lowest buttons put into the clipboard the names of what is marked (Ctrl+Shift+N) and its full addresses as `rclone` understands them (Ctrl+Shift+P), one a line.

Keys

Key	What it does
Tab	go to the other panel
Enter	enter the folder
Backspace	one level up
Alt+← and Alt+→	back and forward through the history of the panel
Home, End	to the start and the end of the list
Space	mark the row and move the cursor down; a folder — and count its size
Insert	mark the row and move the cursor down
Ctrl+A	mark everything; with everything marked — unmark

Key	What it does
`*` of the numeric keypad	invert the marks
Shift+Space	the exact size of the row under the cursor
F	the quick filter by name
Ctrl+R	read the panel again
Ctrl+U	swap the panels
Alt+F1, Alt+F2	the list of sources of the left and the right panel
Ctrl+F, Alt+F7	the search
Ctrl+Enter	a terminal where the panel stands: a folder of this machine or a machine over ssh
F2	rename; a taken name is refused, at the press and right before the job starts
F3	public links to what is marked in an R2 bucket — into the clipboard
F4	open the file in the editor
Shift+F4	a new file — and straight into the editor
F5	copy to the other panel
F6	move
F7	create a folder
F8, Delete	delete (with confirmation); on a disk of this machine — into the Windows Recycle Bin
Shift+F8, Shift+Delete	delete past the Recycle Bin, for good
F11	copy newer files into the other panel
F12	exact copy into the other panel (with confirmation)
Alt+F5	pack into an archive
Alt+F9	unpack archives

Key	What it does
Ctrl+Shift+N	the names of what is marked — into the clipboard
Ctrl+Shift+P	the full addresses of what is marked — into the clipboard
Esc	stop the running operation; in a text field — close the field, the operation goes on; otherwise — take away the filter or the found list, or close SETTINGS

Three buttons understand Ctrl: with it `OPENVPN` checks the paths without connecting, `SSH-KEYSCAN` replaces a host key that changed, and the power button raises the tunnel in spite of overlapping networks.

Masks

The asterisk button opens the `MARK BY MASK` window: it marks rows of the active panel.

- **The mask field.** Masks separated by commas: `*.docx`, `2026-*`. A word without asterisks is a part of a name, `.png` is an extension.
- **OWN SETS.** `SAVE AS…` saves what is typed under a name of its own, and the set becomes a button of this window; `REMOVE SET` takes the button away. The sets lie in the data folder, in `config\mask-sets.json`, and outlive a cleanup of the program.
- **CHANGED.** `ANY TIME`, `LAST HOUR`, `LAST DAY`, `LAST WEEK` — the same limits the colours of age have.
- **GROUPS.** Families of extensions — documents, video, graphics, archives and the rest, each in the colour of its rows. `INCLUDE` adds the marked groups to the field, `EXCLUDE` subtracts them from what is typed: "every document but PDF".

`MARK` takes the previous marks off and sets new ones, `ADD` keeps the previous ones, `CLEAR ALL` removes every mark of the panel, `CLEAR` empties the field, the groups and the time.

The search

Ctrl+F, Alt+F7 or the magnifier of the strip between the panels. It searches from the folder of the active panel downwards — on a disk, on a machine over ssh, in an R2 bucket and in a cloud alike.

- **WHERE** — where to search from; `PANEL FOLDER` puts in the place of the active panel.
- **What to search for** — separated by commas: a word is a part of a name, `.dwg` an extension, `2026-*` a mask, `/expression/` a regular expression. Empty — every file.
- **CHANGED** and **GROUPS** — as in the mask window.

`SEARCH` starts it, `STOP` or Esc stops it — what was found stays in the table. The first 50,000 found are shown; when there are more, narrow the request.

Two things can be done with what was found. `GO TO FILE` opens the folder of the file in the active panel and puts the cursor on it. `TO PANEL` shows everything found in the active panel as one flat list: from there F5 and F6 take it to the other panel, F8 deletes, and Esc brings the folder itself back.

Links — F3

F3 puts into the clipboard the public links to what is marked, one a line, and repeats them in the log. It works in a panel that stands in an R2 bucket with a site of its own: `audion-share`, `audion-media`, `audion-releases`, `audion-vault`. Nothing is uploaded or changed: a link is built from the place of the object.

Where a site serves only the paths that were opened, a link to a closed path is marked as closed: it is built, but will work once the path is opened. Links to `audion-vault` stand behind Access and open only for the owner.

The editor — F4

F4 opens the file under the cursor in the editor; Shift+F4 asks for a name, creates an empty file and opens it. The editor is chosen in SETTINGS: Microsoft Edit and micro travel with the program and open in its terminal, Notepad is always there in Windows, `OWN…` is any program. The editor works with files on the disks of this machine: a file of a far side is copied over first.

Archives

Alt+F5 packs what is marked (or the row under the cursor) into the other panel, Alt+F9 unpacks the marked archives — into the other panel or in place, all together or each into a folder of its own. PeaZip does the work, the one the program carries, and only on the disks of this machine.

The formats: ZIP, 7Z, self-extracting SFX, TAR, TAR.GZ, TAR.ZSTD, TAR.LZ4. The packing dialog offers compression, encryption with a password, one archive for everything or an archive per object, a prefix and a suffix of the name, and a check of the archive after packing. The password is never written to disk.

Folder synchronisation

The three buttons to the right of the file operations work not with folder names but with what is open in the panels: the two directories from the breadcrumb trails, including everything nested at any depth.

Direction comes from the active panel — the source is the one holding the cursor, and the arrow on the button reads "from the active panel into the other one".

Button	What it does	Deletes
`F11 COPY →`	tops up what is missing, refreshes what is older	no
`F12 SYNC →`	the destination becomes an exact copy of the source	yes, with confirmation and a 200-file safety catch
`BISYNC ⇄`	changes travel both ways	as needed, safety catch at 50

Marks set the scope. Nothing marked — the whole open folder is levelled. Something marked — only those folders and files, however many: the names become `rclone` filters, and filters apply to deletion as well, so neighbouring folders in the destination are left alone. For example: `S:\Audion` on the left, `E:\Backup` on the right, `Docs` and `Engine` marked on the left — the mirror brings `E:\Backup\Docs` and `E:\Backup\Engine` in line with the source and leaves everything else in `E:\Backup` as it was.

A highlighted row is not a scope — only a mark is. F5 accepts a highlighted row because copying spoils nothing; a mirror deletes, so the choice has to be deliberate.

`BISYNC` is the one that refuses marks: it remembers the state of a folder pair, and changing the scope would require a fresh baseline. On the first run of a pair it asks which side is authoritative and stores the pair in the data folder: in `config\bisync\pairs.json`, and beside it, in `config\bisync\state`, lies what `rclone` compares the sides with. When that state is lost, the pair is asked about again. The route to a machine does not matter to a pair: made at home, it is the same pair from afar and is not asked about again.

A folder inside a folder. F5, F6 and all three syncs refuse before the question when the target lies inside the source, and `F12 SYNC` and `BISYNC` also when the source lies inside the target: such a mirror would delete its own source as an extra folder. It is the real place of the folders that is compared — through a junction too, and when one machine is open twice: by its disk and over ssh, whatever route leads to it. By a far route the window asks the machine itself whether it is this one — and only when the paths at both ends match; no answer, and the operation is not begun.

All three go into the same queue as F5 and F6.

The queue

F5 and F6 do not start an operation at once — they **put the marked items into a queue**. The destination is remembered at the moment of adding, so you can collect files in one folder, walk into another, switch the source and add from there: everything travels to wherever the opposite panel pointed when you pressed.

The same holds for the other operations that ask — F2, F7, F8, F3: they go where their question said, even if the panel went elsewhere while you were answering — sent by the search window, or opened on WORK after OPENVPN.

Tasks run one at a time: parallel transfers would share one channel and mix their output. The title shows the position — `[3 of 10]`; adding more while the queue runs grows the counter with it.

`STOP` stops the whole batch: it kills the running transfer, clears the rest of the queue and terminates any `rclone` left over from earlier runs. Esc does the same while an operation runs — except Esc in a text field: there it closes the field.

A transfer does not outlive the window. Closing it while a job runs means aborting the job, and the window asks first: "Close the window and abort the transfer?". `ABORT` kills the job with its processes and clears the rest of the queue; a refusal keeps both the window and the transfer.

What you see while it works

A progress bar at the bottom, the state in the title of the window (visible in the taskbar without restoring the window) and the status line: operation, speed, percentage, remaining time. Every five seconds an `rclone` line lands in the log — so much of so much, speed, ETA.

The panels are read again after every operation. By hand — Ctrl+R or the round arrow of the strip between the panels; `STATE` reads both again.

Where the traffic goes

Through the machine running the window: `rclone` reads from the source and writes to the destination. Copying from this computer to BASE goes over the home network through the router — neither SECOND VPS nor the tunnel takes part. On a 2.5-gigabit network that gives about 180 MB/s one way and 200 the other: some two thirds of what SMB does, in exchange for resume, retry on failure and verification.

From someone else's network the traffic takes a longer way: to MAIN VPS or SECOND VPS and from there home through its tunnel. Measured over a mobile channel — 1.65 MB/s, and that is the limit of the channel itself, not of the program: both far routes gave the same figure.

Routes to the machines

Four routes lead to the home machines, and the window takes the first one available.

`ROUTE` button	Where it goes
`LAN`	straight to the address of the machine in its own network — from home, or with the way home raised
`SSH MAIN VPS`	through the MAIN VPS jump host by its direct name
`SSH MAIN VPS CF`	the same MAIN VPS over cloudflared
`SSH SECOND VPS`	through the SECOND VPS jump host

`AUTO` listens to the probe; the others keep what was chosen even when the probe disagrees — a person knows more about a network than a knock on a port. The route stands in the caption of the source above the panel: `BASE · SSH MAIN VPS CF`. A choice made with the buttons of the row holds until the program is closed.

Home always wins. From afar `AUTO` takes the VPS that answers the probe faster: MAIN VPS by its name or SECOND VPS — each timed to the server itself. Less than a tenth apart (or 5 ms) is no difference, and MAIN VPS comes first then: the route does not jump from probe to probe. The route over cloudflared has no time of its own to MAIN VPS — the probe only reaches the edge of Cloudflare — so it comes after the timed ones, yet it passes where nothing else does: MAIN VPS has no open port at all. Sensitive data never travels through SECOND VPS: pin `SSH MAIN VPS` with a button of the row for it.

The kebab `⋮` at the end of the row opens the routes per machine: each machine its own, with the time to the nearest node. `APPLY` — until the program is closed, `MAKE DEFAULT` — for the starts to come as well, `RESET` — `AUTO` for every machine with no defaults, as in a fresh program. A tangerine dot on the kebab means the routes now are not the defaults.

The probe decides — at startup and on the `PORT 22 TCP PING` button (TUNNEL → NET). Next to it stands `DIAGNOSTICS`: it walks every route to BASE and writes which ones live, how fast they answer and how many folders they read; the report goes into `logs`. In an unfamiliar network it is the place to start: it answers "what do I work through right now".

Routes by machine. One choice for all does not always fit: the home network may reach BASE and not a machine in another network. The kebab at the end of the `ROUTE` row and the button with the network sign in the strip between the panels give each machine a route of its own.

The line in the middle of the header says what is reachable and where the window will go:

```
probe 18:03 · BASE 1 ms · MAIN VPS 90 ms · MAIN VPS CF 57 ms · SECOND VPS 45 ms → LAN
```

The word `none` in place of a time means the route is silent. After the arrow is the way of the window; when the probe did not choose it, the brackets say who did: `chosen` — with a `ROUTE` button, `by machine` — each has its own. The whole line is in the tooltip.

When nobody answers — the program started before the network was up, or a VPN is still connecting — the probe is tried again in 20 seconds, up to three times in a row. When the network changes (a VPN came up, another Wi-Fi) the window waits for it to settle and probes again.

An answer means the port is open, not that the protocol will pass. Filtered networks cut ssh after the connection is made, and the probe does not see that. The choice by hand is there for such cases.

MAIN VPS in the panels

MAIN VPS stands in the panels next to the machines and the R2 buckets: an R2 bucket on the left, MAIN VPS on the right, and the file moves between them directly — no `scp`, no console. The route is rclone's own `sftp` with the MAIN VPS key; the spare one, over cloudflared, steps in only if the direct one goes quiet.

MAIN VPS has no home address, so its entry in `targets.json` carries an `sshHost` field — the name of a host already described in `~\.ssh\config`. The MAIN VPS button in the machine row opens a console by that same name.

The log

The right third of the window. Both the window and the scripts write here — line by line, coloured: step headings, successes, warnings. Red is left only for what really broke. Every line with its time also goes into the file of the day, `logs\gui-YYYYMMDD.log` in the program folder: a machine far away is taken apart by that file.

`COPY` puts what the panel holds into the clipboard — its last three thousand lines; `CLEAR` empties the panel. The file of the day keeps everything.

An error the handler of a button did not catch does not close the window: it goes whole into `logs\gui-crash-<time>.log`, and one line with the name of that file stays in the log. When errors come one after another — twenty within a minute — the program closes, and the last file tells why.

Two buttons stand below the log. `STOP` or Esc interrupts the running operation and clears the queue. `STATE` reads both panels and the indicators again, probes the network and writes a summary — the same one appears a few seconds after the window opens:

```
===== STATE =====
snapshot: 05.10 04:00, 76 MB, age 11 h
R2 keys: own, in place
cloudflared: 2026.9.3
sshd: running · WG BASE-ROUTER: installed: router-base-wg

===== TUNNELS =====
clients: WG, RS AWG · installing on RS AWG
router-base-wg   up, 34 s ago, 172.16.6.1 answers in 12 ms
redlinkabcdefgh  up, not ours
```

TUNNELS lists every tunnel service of the machine, the ones of another VPN too: they share one driver, and the one that took it first decides for the rest. A tunnel is ours when its config lies in `Tunnel\WG` of the data folder; for such a tunnel the last handshake and the answer of the far end are shown. "The client is gone" means the program the service is started by was removed or moved: the tunnel is installed again with the wizard. The "schedule" line appears only where a task of earlier versions is left.

The same facts come from a console through `Engine\Scripts\Get-StateSummary.ps1`.

SETTINGS

The `SETTINGS` button stands at the top right, beside `SETUP`, `ABOUT`, the language and the theme. The settings stand in place of the panels; the log stays beside them. Esc, the cross or the same button leaves them. A tangerine dot on the button means a newer version of a tool is out; the hint of the button names which.

- **DATA FOLDER** — where it is now; `CHOOSE` another, `OPEN` it in Explorer. With another folder the window changes in one step: the clouds of the old one leave together with its config, the panels open anew, the clouds of the new one are attached. The folder is not changed while an operation runs, counts or asks its question: the log says "an operation is running: the data folder is changed after it". A folder that has no machine addresses yet gets their empty frame from the sample of the program — the `ADDRESSES` button takes it from there.
- **TOOLS AND VERSIONS** — what is installed and what came out newer. `CHECK VERSIONS` downloads nothing, it only asks; `UPDATE` updates all six tools and installs the missing ones. Settings, keys and configs stay.
- **EDITOR OF F4** — Microsoft Edit, micro, Notepad or one of your own.

- **FONT** — `SMALLER`, `NORMAL`, `LARGER`, and the fonts themselves: one for the program, another for the panels and the log.
- **TRANSLUCENCY** — the effects of Windows 11: `ACRYLIC`, `MICA` or a solid window, and three densities of the background; by default — `ACRYLIC` in the file panels only. With the effects switched off in the system the window stays solid.
- **SIZES** — `SHORT` (KB, MB, GB) or `IN BYTES`: a few bytes of difference between two folders show another version.
- **BOTTOM ROW OF BUTTONS** — with captions or without.

The settings of the window live in `config\gui-settings.json` of the program and do not go into the data folder. The sets of masks and the pairs of BISYNC are not settings of the window but what was made in it: they lie in the data folder.

ABOUT

The `ABOUT` button at the top right shows, in place of the panels, the version of the program, its folder and the data folder. From here open the user guide, the changelog, the restore instructions and the README — in the language of the window — the folder of licenses and the page of releases on GitHub.

Clouds

The `CLOUDS` section. Connected clouds stand as panel sources next to the machines, and everything the window can do it can do with them: copy, move, delete, rename, marking by mask, the search, mirror and sync.

The buttons of the section: `CONNECT` adds a cloud — the sign-in happens in a browser on the side of the cloud itself, the program sees neither the login nor the password; `SIGN IN AGAIN` replaces only the token, the settings stay; `RENAME` gives a cloud another name; `DISCONNECT` takes it off the list — the account and its files are not touched; `VIA MAIN VPS` moves data from the left panel to the right one by the hands of MAIN VPS rather than of this machine. Before it starts the window counts how much will go and asks: the work goes on MAIN VPS by itself and `STOP` will not stop it — `clouds-send.cmd -Watch` shows how it goes. One transfer runs at a time: a second one does not start until the first has ended.

What was checked against rclone itself:

Cloud	rclone key	Sign-in	Own application key	Worth knowing
Google Drive	`drive`	OAuth in a browser	yes	without your own key the quota is shared by all rclone users
Dropbox	`dropbox`	OAuth in a browser	yes	same
OneDrive	`onedrive`	OAuth in a browser	yes	the password setting is for shared links, not for signing in
Yandex Disk	`yandex`	OAuth in a browser	yes	the application is registered in the Yandex console
pCloud	`pcloud`	OAuth in a browser	yes	—
Box	`box`	OAuth in a browser	yes	—
MEGA	`mega`	login and password	no	the password goes into the encrypted config
Proton Drive	`protondrive`	login and password	no	asks for the code with two-factor on, and for the second password in two-password mode

The list is deliberately short: rclone knows four times as many backends, and the wizard will pass any other key through — type it instead of the offered one.

Your own application key is not always needed. The shared rclone key is divided among all of its users: at peak hours that means refusals and lower speed. Your own is registered in the cloud's own console in a few minutes; the wizard asks about it and skips the question on Enter.

Where the access keys live. Not in the shared `rclone.conf` — that file lies open and travels between machines. Clouds live in `config\rclone\clouds.conf`, encrypted by rclone itself, and its password sits in `clouds.key`, sealed by Windows to this user and this machine: a copy on another machine will not open. To see paths and state use `clouds-keys.cmd` or the `WHERE KEPT` button; neither prints any value.

The golden copy of the cloud config lies on MAIN VPS (the `TO MAIN VPS` button), and its password lies in the data folder, `Tunnel\clouds.key`, as plain text; only the owner, administrators and the system may read the file. With it the `FROM MAIN VPS` button brings the clouds to a new machine without signing in again. EXPORT KEYS and IMPORT KEYS do not carry it: the file is put onto a new machine by hand.

How much space is left is reported by `clouds-check.cmd` and the `CHECK` button: per cloud, and in one line across all of them. Where a cloud has no notion of a quota, nothing is shown.

Machines

In the REMOTE section the machine buttons of the `TERMINAL` and `ROUTERS` groups open a terminal on the machine. The route picks itself from the last probe: the home address answers — go straight; it is silent — through the MAIN VPS jump host, and if that is silent too, over cloudflared or through SECOND VPS.

The set of buttons comes from `targets.json`, not from the markup: what is in the file is in the row, and a key prefixed `router-` lands in the router group. A machine whose address is not written yet stands with a grey indicator, and pressing it says so. The address is written with `ADDRESSES` of SETUP; no code changes — the button, the probe and the terminal take the address from one file.

`TERMINAL` chooses the program and shows the choice in its caption. Without a choice Windows Terminal from `Tools\terminal` is used — it travels with the program; without that, the system `ssh`. `TABBY` finds Tabby by itself: it understands an `ssh://user@address` link and opens a tab straight on the host. Ctrl+Enter in a panel opens a terminal where the panel stands.

The indicator of a machine has four meanings: green — it answers directly; amber — silent directly, a detour is taken; Burn Orange — silent, and there is no detour; grey — no address written. The hint of the indicator names the address and the time of the answer. The network is probed by `PORT 22 TCP PING` and once when the window opens.

`SSHD`, `SSH TUN`, `WIREGUARD` and `WG BASE-ROUTER` set up **this** machine. What has to be asked they ask in the window, and the work itself runs in a console of its own: it is interactive. All of them ask for administrator rights except `SSH TUN`: its channel is held by an ordinary process, not a service. `SSHD` puts the public keys of machines from CLIENTS into `C:\ProgramData\ssh\administrators_authorized_keys`: for an account in the Administrators group Windows sshd reads keys only there and ignores `.ssh\authorized_keys`. A line of a machine from CLIENTS that carries another key is replaced with the key from CLIENTS, and missing keys are added. Lines of machines CLIENTS does not know stay and are named by their label: every machine has its own CLIENTS, and `SSH-COPY-ID` on the neighbours brings them.

`SSH TUN` of SETUP is REVERSE SSH TUN: it lets you into this machine from afar through a reverse SSH tunnel. The machine itself holds one outgoing connection, and MAIN VPS or SECOND VPS opens a port on its own localhost: whoever is there gets here through that port. No adapter, no service, no routes. The wizard has four actions, each with its own wrapper:

- `RAISE THROUGH SECOND VPS` — `reverse-ssh-second-vps.cmd`: the SECOND VPS host, a shorter way that needs no VPN; if it does not answer, the channel goes through MAIN VPS;
- `RAISE THROUGH MAIN VPS` — `reverse-ssh-main-vps.cmd`: MAIN VPS only, for what must not be shown;
- `INSPECT` — `reverse-ssh-check.cmd`: the key, the port on MAIN VPS or SECOND VPS, the process and whether the port is listening now;
- `DELETE` — `reverse-ssh-remove.cmd`: the channel stops, the tunnel key is deleted from MAIN VPS or SECOND VPS, whichever it was registered on, and stays in the data.

On the first connection to a VPS, setup offers to confirm its SSH fingerprint in the separate OpenSSH window. Compare the displayed SHA256 with `ssh-keygen -lf /etc/ssh/ssh_host_ed25519_key.pub` in the provider's server console and confirm only a matching fingerprint. Setup then retries the connection without a dialog and continues. A changed fingerprint is not replaced; inspection and removal do not accept new hosts. Confirmation is unavailable with redirected input: run setup in its separate window.

Setup progress and each route's errors are saved under the program's `logs` folder as `ssh-beacon-setup-<date>-<time>-<pid>.log`. A fallback error no longer hides why the selected VPS failed. A missing `IdentityFile` is reported as a private key path problem.

The first press of `SSH TUN` configures a persistent tunnel. The next press disables it and its Windows sign-in startup; another press reuses the saved key, port and server. Ctrl opens VPS selection, inspection and removal. Persistent workers run independently of the GUI and start through a shell:startup shortcut. They reconnect after SSH interruptions. GUI closure and file-operation STOP do not stop them. The channel is unavailable before sign-in, during sleep or shutdown. LAPTOP supports explicit persistent setup; old remnants stay disabled. CLI: `reverse-ssh-toggle.cmd`, `Setup-SshBeacon.ps1 -Resume -Persistent`, or `-Stop`, with an explicit `-Root`.

Startup errors are captured in `logs\ssh-beacon-<name>-worker-*.err.log`; OpenSSH connection progress is in `logs\ssh-beacon-<name>-ssh.log`. The green indicator requires the server to acknowledge the remote forward. This does not yet test local SSHD on the machine. For new workers, process age alone does not prove connectivity.

The tunnel key lives on MAIN VPS and SECOND VPS in an account of its own, `tunnel`: no shell, a locked password, and nothing in its home but the key file. This machine's key is kept on the machine itself — in someone else's network, on a trip, behind a foreign firewall — so a stolen one runs into an empty account rather than the working one. The account is created on the server once, by hand; what it is allowed ships as the sample `Tunnel\RemoteDesktop\config\sshd-tunnel.sample.conf`. Until it exists, `RAISE` changes nothing and says so.

REVERSE SSH TUN is for a machine that cannot be reached from outside. A machine that goes out on its own does not need one: an extra channel is an extra key on MAIN VPS or SECOND VPS and an extra port. The indicator is green while the tunnel's own ssh holds the channel — started by its process, with its key and port; amber when the tunnel is set up but there is no channel yet or it is still coming up; grey when it is not set up. Whether the port on MAIN VPS or SECOND VPS listens is answered by `INSPECT`.

`NDMS API` talks to the home router through its own API: it shows the model, the firmware and the peers of the WG server with their handshakes. It only reads. `ROUTER WG PEER` issues this machine a WireGuard config — it registers a peer in the router and builds the file; the key pair is born here, the router gets only the public key. The router password is entered with `ROUTER PASS`. NDMS is the firmware of the router whose API these buttons speak: the field hides what is typed, the value is stored portably in the data set under the name of the router, and right after saving the program tries to sign in and says whether it worked. The file survives a reinstall and travels with the set, yet is useless by itself — without access to the router nothing can be done with it; the value never gets into correspondence or into a repository.

`WIREGUARD` installs the program itself — a new machine starts with it. The indicator is green when it is already there; the button then does nothing and says so. There is no portable build: the tunnel is raised by a Windows service and its network adapter is a driver, so an installer and the rights are needed.

The config of a client need not be written by hand: `new-router-peer.cmd` registers a peer in the router itself and builds the file. Look at the preview first:

```
new-router-peer.cmd -WhatIf          what would be done
new-router-peer.cmd -Name NAME         issue one for another machine
new-router-peer.cmd -Name NAME -Remove remove a peer
```

The router password is asked once; with `-Remember` it is stored in the data folder under the name of the router and is not asked again. `check-router.cmd` shows that the router answers and lists the peers.

`WG BASE-ROUTER` raises the way home from the config the router issues: a client is added in the settings of its WG server, and the router shows a QR code and a file. The script picks a fresh `.conf` from Downloads by itself, or the path is given with `-From`. The golden copy of the config is kept in the data, next to the keys: `Tunnel\WG\router-base-wg.conf`. Which networks go into the tunnel the wizard asks in the window: by default the home one alone, everything else goes past the tunnel. The list of networks lies beside it, in `Tunnel\WG\routes.txt`.

The tunnel client

Tunnels are held by a Windows service, and the service is created by a client — either plain WireGuard or AmneziaWG. The system holds one kernel driver for all, and whoever claims it first decides for the rest: the Amnezia build speaks both obfuscation and plain WireGuard, the plain one only plain.

Hence the rule: **the program installs the tunnel on the client the machine already has.** If someone else's VPN on AmneziaWG is there — Red Shield, for instance — the tunnel goes on it. Not because it is better, but so as not to take away someone's own VPN: a second client installed beside it would claim the driver, and the other VPN would stop connecting.

The program installs no client of its own and touches nobody else's. The `WIREGUARD` button looks first at what is there and installs only when there is nothing at all.

The service is named after whoever created it: `WireGuardTunnel$name` or `AmneziaWGTunnel$name`. The program looks under both — otherwise, on a machine with AmneziaWG, its own tunnel would look as if it had never been set up.

Raising and automatic start. The tunnel wizard offers two raise actions:

- `RAISE` — set it up and leave manual start. After a reboot the tunnel waits for the button. That is right for a machine travelling through other people's networks: you switch the computer on at work and the tunnel does not disturb the VPN there until it is wanted.
- `RAISE ALWAYS` — the same, but the service is switched to automatic start. For a machine out of arm's reach: it rebooted, and it is back in touch.

The mode toggles with the same press; the tunnel need not be removed.

The power button of the strip between the panels switches an already installed tunnel on and off. The indicator of `WG BASE-ROUTER` says whether it is installed; the power button, whether it is up right now. These are different questions: the service can be running while the adapter is disabled.

Port forwarding: the desktop and everything else

The window carries files, the terminal runs commands. That leaves the third thing - a desktop, a router's web page, a network share, a camera. All of it travels over the ssh you already have, with no tunnel.

It works like this: ssh brings someone else's port to your machine. The address you name is resolved **on the far side**, not here. So through one machine you reach everything it can see:

```
Tunnel\RemoteDesktop\desktop-base.cmd     the desktop of BASE
Tunnel\RemoteDesktop\web-router.cmd       the home router's web page
Tunnel\RemoteDesktop\socks-base.cmd       a proxy into the whole home network
```

In the window these are the `RDP`, `ROUTER WEB` and `SOCKS` buttons of the `REMOTE` section.

What happens to the system: nothing. No adapter, no routes, no DNS of its own; another VPN has nothing to argue with. Close the desktop window and the forward is gone, leaving no traces.

An arbitrary port is given by switches:

```
Open-Forward.ps1 base port -RemoteHost 192.168.1.15 -RemotePort 445
Open-Forward.ps1 base web  -RemoteHost 192.168.1.50
Open-Forward.ps1 base-cf desktop
```

The first parameter is a host from `~\.ssh\config`: `base` goes over the home network, `base-direct` goes through MAIN VPS with ProxyJump, `base-cf` uses the same MAIN VPS over cloudflared. Name the far route explicitly when the direct one is gone.

The proxy is different: it carries no single port but opens the far side's whole network to a browser. The script prints the address and port to put into the browser as SOCKS5.

When a tunnel is still needed: when the home network is needed whole and at once

- not one device but everything, including what you did not name in advance. For a single port a forward is cheaper in every way.

All of this works because the keys are in place and the host fingerprints are verified: without trust on both sides every such forward would be a hole, not a route.

From the console

The buttons do no work themselves — each calls the same script a person would call from a console. That is why the window and the wrappers never drift apart: one script gets fixed, not two places.

Wrapper	What it does
`sync-backup.cmd`	MAIN VPS mirror; `-WhatIf`, `-Only 'Folder','Folder'`
`import-keys.cmd`	lay the keys out; `-Check`, `-Force`, `-Source Store`
`export-keys.cmd`	collect Export Keys into `Tunnel\Export`; `-Check`, `-Force`
`ssh-keyscan.cmd`	refresh host keys in machine data; `-Root`, `-Force`, `-TimeoutSeconds`
`backup.cmd`	Build-stack snapshot; `-ArchiveRoot`, `-Publish`, `-ProjectName`
`Tunnel\check-tunnel-client.cmd`	state of the tunnel client
`Tunnel\install-tunnel-client.cmd`	install the client and write the routes
`Tunnel\update-cloudflared.cmd`	update `cloudflared`

Each accepts `-Root` — the path to the data folder when it is not the neighbouring one.

The scripts of `Engine\Scripts` run under the program's own PowerShell: `App\AudionPs.exe -NoProfile -File Engine\Scripts\<script>.ps1 -Root "data folder"`.

Where things live

In the program:

```
Start.exe                       opens the window
Runs\                           wrappers: mirror, keys, snapshot, updates
App\AudionSyncTools.exe         the window
App\AudionPs.exe                own PowerShell 7: runs the scripts and the wrappers
Engine\BackupEngine.psm1        the engine: targets, keys, tunnels, the data pointer
Engine\BackupRclone.psm1        own rclone: lookup, config, calls
Engine\DataRoot.ps1             where the data folder is - one answer for all
Engine\Scripts\                 the scripts behind the wrappers
Engine\Launcher\                Start.cs, exe and icon builders
Tunnel\                         tunnel client and Bin\cloudflared.exe
Tunnel\RemoteDesktop\           access to the machines
Tools\rclone\rclone.exe         own binary
Tools\terminal\                 Windows Terminal: the terminal to the machines and for
the editors
Tools\PeaZip\                   the archiver
Tools\edit\, Tools\micro\       the editors of F4
App\config\                     rows of buttons, the dictionary, row colours, mask
groups
logs\                           the log of the window by day and the logs of the
wizards
Assets\SyncTools.ico, .png, .svg   the icon
config\version.json             version
config\profile.json             pointer: where this machine's data folder is
config\rclone.sample.conf       the rclone.conf template, no values
config\gui-settings.json        what the window remembered
```

In the data:

```
Snapshots\                        the latest MAIN VPS snapshot, whole
MAIN VPS Backup\                  MAIN VPS's units, scripts and documents
Forgejo Mirror\                   bare copies of the repositories
Audion Portal\                    the portal
Audion Vault Portal\              the notes vault worker
Planning Signals\                 signal collection
Audion Share\                     the storefront worker
Tunnel\SSH\clients\               CLIENTS: the keys of the fleet's machines
Tunnel\SSH\servers\               pairs to MAIN VPS and SECOND VPS, the golden copy
Tunnel\SSH\beacon\                this machine's REVERSE SSH TUN key
Tunnel\Export\SSH\                Export Keys: a cast of this machine's keys
Tunnel\WG\                        the config of the tunnel to the BASE router
Tunnel\clouds.key                 the password of the cloud config
Tunnel\Tokens\                    GitHub, Cloudflare and Telegram tokens
Tunnel\ACCESS.md                  the access register
config\rclone\                    rclone.conf, R2 keys, host fingerprints, clouds
config\mirror.json                the mirror plan
Tunnel\RemoteDesktop\config\targets.json   machine and SECOND VPS addresses
```

The machine's profile lives there too — everything that sets it apart from the same program on someone else's computer:

```
config\rclone\                    rclone.conf and keys to the R2 bucket
config\mirror.json                the SYNC ALL plan: MAIN VPS, the R2 bucket,
repositories and folders
Tunnel\RemoteDesktop\config\      machine addresses and the router password
```

Window settings — `config\gui-settings.json` — are not part of it: they belong to the program itself and live in its folder.

The key layout rests on one rule: **a machine's private key stays with that machine.** It lies in the `%USERPROFILE%\.ssh` profile, where ssh looks for it, in CLIENTS next to its `.pub`, and in `Tunnel\Export\SSH`, from where it returns after a system reinstall. Only the public key goes to other machines.

Sharp edges

A key with shared permissions. `ssh` and `ssh-keygen` refuse to read a private key that someone other than its owner can reach — including accounts that no longer exist on the machine. The import fixes it, narrowing permissions both on the copies and on the files in the source.

Moving the folder. The paths to the key and to `cloudflared` are written in two places: `~\.ssh\config` and the `.git\config` of every mirror. After a move git quietly stops fetching — "Identity file not accessible". The import repairs both, but only the paths that lead nowhere: working lines are left alone.

Golden copy and Export Keys. `Tunnel\SSH` is the golden copy, kept for the day an export comes out wrong: EXPORT KEYS never writes into it. The copy of the routes stored there, `ssh_config.copy`, is not updated automatically — when a new host appears in `~\.ssh\config`, that copy has to be refreshed. IMPORT KEYS prefers Export Keys and supplements missing server pairs from `Tunnel\SSH\servers`, preserving the golden copy's contents. Private file permissions may be repaired so OpenSSH accepts them after moving to another Windows installation. If REVERSE SSH TUN setup gets `bad permissions`, it repairs the selected key's permissions within machine data or the profile and retries login. Repeat setup after updating the program.

One snapshot. `Snapshots` keeps the latest and removes the previous ones: the version history is in the R2 bucket anyway, and two snapshots on disk mean a hundred and ten megabytes plus the question of which one is fresh.

The data folder is not empty. `Audion Backup` must not be deleted: it holds the keys, the tokens and the only local copy of MAIN VPS. The program knows how to fill it, but keeps no data inside itself.

OpenVPN from machine data

TUNNEL → NET → OPENVPN offers connect/open WORK or disconnect the selected profile. Keep the portable client in `Data folder\OpenVPN Portable`, profiles in its `data\config\*.ovpn`, and the `.rdp` file beside `openvpn-portable.exe`. The owner supplies these files; they are not included in the Sync Tools release.

Installed OpenVPN GUI is detected automatically; otherwise the portable copy in the selected data folder is used. **SETUP → OTHER → OVPN PATH** saves an explicit client and profile selection, which takes priority. A single profile is selected automatically. Client paths inside data are relative and survive moving the data folder to another drive.

Install the appropriate OpenVPN driver before the first connection on a new Windows machine; Sync Tools does not install it. **Ctrl+OPENVPN** validates paths and the running GUI without connecting. An existing GUI using another config folder is left alone. Close it yourself when disconnecting its current VPN is acceptable, then press OPENVPN.

A launch request does not prove a connection. Check status and errors in the OpenVPN tray; logs are in its `data\log`. The external client starts through UAC. Audion waits up to 90 seconds for a real

rclone listing of `work:/`, then opens WORK in the original active panel. Changing DATA, section or panel source while waiting cancels that automatic opening. OpenVPN provides the route; rclone uses SSH/SFTP and the selected DATA's `[work]` configuration to access the disks. On a new machine configure IMPORT KEYS, RCLONE.CONF and trusted WORK host keys first. Server keys are never accepted automatically. A failed listing reports an error and leaves the VPN running. OPENVPN can disconnect the selected profile; other profiles are not stopped. STOP remains a file-operation control. CLI actions: `openvpn.cmd -Connect` or `openvpn.cmd -Disconnect`, with `-Root`. Copies of one client certificate may displace each other on different machines; simultaneous use needs client profiles approved by the VPN server administrator. Open the RDP file separately after the connection is established.

CLI equivalents: `Tunnel\RemoteDesktop\openvpn.cmd -Root "data folder"` and `openvpn-path.cmd -Root "data folder"`; use `openvpn.cmd -Check -NoPause` to check.