Audion DevOps Tools — User Guide

Русский · About · Command Reference

Contents

- Quick Start
- The Central Rule
- Risk and Confirmation
- Reading the Window
- Where the Working Data Lives
- AI Transfer
- The Sections
- The Terminal
- Checks
- Technical Reference

How to work with it: launching, the central rule, risk marking, the sections, and what to read about each.

Quick Start

```
Start.exe
```

`Start.exe` asks for administrator rights once — most operations need them: deployment servicing, Linux subsystem features, the `hosts` file, network adapters, default and driver policies, the recovery environment, disk work. Answering "no" opens the window without them: the rights badge shows it, and a click on it restarts the window with the rights.

Without the question about the rights:

```
set AUDION_GUI_NO_ELEVATE=1
Start.exe
```

The Central Rule

Every operation goes through the project's own means:

- the window;
- the console command `devops_cli.cmd run --operation <operation>` — the same one behind the buttons of the window;
- `launcher_*.cmd`;
- `runtime\python.exe system_core\cli_operation.py <operation>`.

This matters most for the Linux subsystem and system operations. When testing the project layer, do not bypass it with manual commands — otherwise you are testing luck in the current console, not the project.

A dangerous operation straight from the command line needs an explicit flag:

```
runtime\python.exe system_core\cli_operation.py <operation> --yes-i-understand
```

Risk and Confirmation

Every operation carries a kind:

kind	meaning
`safe`	reading, diagnostics, or writing only inside the project area
`dangerous`	changes the user, the system, the network, disks, policy, or secrets

And a level that says what changes:

level	what changes
`project_write`	writes into the project
`user_write`	changes the current user

level	what changes
`system_change`	changes Windows policy
`destructive`	may delete data or partitions
`secret_export`	exports secrets, such as SSH keys

Dangerous operations need separate confirmation. Some external wizards also ask you to type a word. Built-in non-interactive operations must carry explicit project flags rather than hidden yes/no prompts.

Reading the Window

Six sections are the large buttons under the header: NETWORK, WSL & VM, DRIVERS, DEFAULT APPS, EXPORT & IMPORT, SYSTEM. Under a section stand a line on what it holds and the row of its pages. A page is cards of operations; the cards of one job share a frame.

The window speaks Russian or English: the RU/EN switch in the header. The visible label is short: its job is to help you decide in a couple of seconds. The tooltip of a button tells what exactly will change, the Windows terms, the risk and the way back; the confirmation of a dangerous operation shows the full text. The id of the operation is in the log — the console command runs it by that id.

What the frames and colours mean

The colouring here is instrument marking, not decoration.

A frame around cards means a pair or a lifecycle: backup and restore, export and import, block and unblock, capture and apply. The way back is always in view.

button	meaning
filled	the main action of its frame
orange	removes or erases
lit switch	the state the machine is in now
plain	picks an option: runs nothing by itself

Colour does not replace the operation kind, the confirmation, or the log — it helps read the character of neighbouring buttons within one frame at a glance.

Stable word pairs

word	meaning
backup	a snapshot of state inside the project
export	take state or artefacts out into a file or folder
import	bring a file or folder back into the system or a profile
restore	apply saved state back
block / unblock	set or lift a restriction
enable / disable	a mode or a component
capture / apply	save a layer and apply it back
cleanup	tidying within a chosen scope; look for a dry run first

Where the Working Data Lives

```
App\          the window and the PowerShell of the program
config\       command definitions, window layout, terminal history, window settings
system_core\  services, PowerShell modules, Linux subsystem assets
tools\        project utilities and wrappers
profiles\     managed profiles, such as the associations baseline
backup\       backups of system and user state
output\       reports and exported artefacts
logs\         operation logs
report\       elevated logs and diagnostics
workspace\    tool working folders
```

`backup\` is a first-class folder alongside input and output: it is created during preparation, opened by its own button, and holds what changes are rolled back from. It is not a cache or scratch space: network snapshots, registry branches, driver store exports, browser bookmarks before and after, certificates, SSH material, virtualisation and `hosts` backups.

AI Transfer

The AI page lists local Claude Code, Codex (including compressed logs) and Grok conversations, Desktop profile availability and data categories. Chats and other data are selected independently. Preview is enabled initially; import reads the selected SOURCE and export uses TARGET. Conflicts are preserved unless explicitly replaced, and imports have recovery journals.

See AI data transfer for supported versions and cloud-data limitations.

The Sections

Each subsystem has its own document under `tools\` — the order of operations, Windows specifics, and pitfalls.

section of the window	pages	details
NETWORK	link, Wi-Fi, DNS, repair, hosts	`tools\NETWORK_CONNECTIVITY_RU.md`, `tools\DNS_DOCTOR_RU.md`, `tools\BITRIX_HOSTS_RU.md`
WSL & VM	WSL, setup, distribution, Ubuntu, Hyper-V	`tools\WSL_TOOLKIT_RU.md`, `tools\VIRTUALIZATION_SWITCHER_RU.md`
DRIVERS	Windows Update, HWID lock, NVIDIA, Driver Store, audit	`tools\HARDWARE_DRIVER_GUARD_RU.md`
DEFAULT APPS	Microsoft, Edge, policy, watch, snapshot, groups, report	`tools\ASSOCIATION_DEFENSE_RU.md`, `tools\DEFAULT_APPS_GUARD_RU.md`
EXPORT & IMPORT	bundle, SSH, certificates, access, fonts & shell, bookmarks, AI CLI	`tools\MIGRATION_RU.md`, `tools\OPENSSH_KEYKIT_RU.md`, `tools\CERTIFICATE_KEYKIT_RU.md`, `tools\BROWSER_BOOKMARKS_MASTER_RU.md`
SYSTEM	environment, disks, cleanup, program	`tools\STORAGE_DISK_PROCEDURES_RU.md`, `tools\MAINTENANCE_CLEANUP_RU.md`

The subsystem documents are in Russian.

Two short sequences are worth knowing by heart.

Defaults after a clean Windows install. Check protection → overwrite the baseline with current defaults → leave "remove suggested" on → enable protection → sign out and in, or reboot → check protection again.

A Bitrix address override. Detect the endpoint → check name and port status → enable the override → do the work → disable it. Disabling restores `hosts` byte-for-byte from the backup named in the managed comment.

Section documents are currently Russian only.

The Terminal

Output appears live, with Windows and Linux subsystem encodings decoded — no mojibake. The final status lives below the terminal and stays until the next run: grey for idle, a blue pulse while running, green after success, red after an error. A pop-up notification is not the source of truth: if the window was inactive, you may never see it.

There is a line for running a command by hand, with history and a cache of frequent ones.

Checks

```
install\verify_portable_env.cmd
runtime\python.exe -m pytest -q
```

What is actually run before a release: the checklist (Russian).

Technical Reference

Section launchers

```
cli\launcher_wsl.cmd
cli\launcher_bitrix.cmd
cli\launcher_default_apps.cmd
cli\launcher_association_defense.cmd
cli\launcher_hardware.cmd
cli\launcher_codex_nuke.cmd
cli\launcher_python_nuke.cmd
```

The first ones go through the same manifest and service layer as the window. The last two wrap the built-in cleanup tools with elevation and typed confirmation.

Operations from the command line

```
devops_cli.cmd describe
devops_cli.cmd run --operation <operation> --set <field>=<value>
devops_cli.cmd check --operation <operation>
runtime\python.exe system_core\cli_operation.py <operation> --yes-i-understand
```

`devops_cli.cmd` is the command behind the buttons of the window: `check` tells whether a form may run and what to ask before it, `run` runs it. The `cli\` menus go through `system_core\cli_operation.py`; a dangerous operation there needs an explicit flag.

The full list of operations and their fields is the command reference. It is generated from `config\tool_manifest.yaml` and includes tooltips, risk classification, inherited fields, defaults, and choices.

Maintenance

```
init_folders.cmd                create missing working folders
cleanup_project.cmd             careful cleanup
cleanup_project.cmd /DRYRUN /Y  show the plan, delete nothing
cleanup_project.cmd /BACKUP     the backup folder only, with its own prompt
```

Cleanup keeps scripts, configuration, documentation, licences, the built window, and the folder structure. It removes what was generated or downloaded: the runtime, the package store, builds, downloads, bundled binaries, logs, reports, working-folder contents, and caches. Neighbouring tools are untouched.

Working folders

Two rows sit above the terminal. **SOURCE** is where imports read from: keys, certificates, Wi-Fi profiles, copies; until another is chosen it is the `input` folder. **TARGET** is where exports, backups and reports go; until another is chosen it is the `output` folder. Both rows remember earlier and pinned folders; the source row takes a network folder `\\SERVER\Share\Folder`. An export of keys and passwords is a secret: keep it as a key.