Audion DevOps Tools — User Guide

Русский · About · Command Reference

Contents

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

Quick Start

```
launcher_gui.cmd
```

The window comes up as administrator — most operations need it: deployment servicing, Linux subsystem features, the `hosts` file, network adapters, default and driver policies, the recovery environment, disk work.

Read-only, without the elevation prompt:

```
set AUDION_GUI_NO_ELEVATE=1
launcher_gui.cmd
```

A different Python:

```
set AUDION_GUI_PYTHON=C:\Path\To\python.exe
launcher_gui.cmd
```

The window opens at `http://127.0.0.1:8092/`.

The Central Rule

Every operation goes through the project's own means:

- 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 Interface

The interface is deliberately hybrid. Command and parameter names stay English: they match the operation id, the log line, the Windows calls, and Microsoft's own documentation. The tooltip explains them in plain language.

A typical trio:

```
button     Default Apps Guard · Enable / repair defaults protection
tooltip    the Windows "Default apps" section, the AppAssociations.xml baseline,
           the DefaultAssociationsConfiguration policy
log        default_apps_apply_policy
```

The short name keeps technical precision; the tooltip explains what will actually change.

What the frames and colours mean

The colouring here is instrument marking, not decoration.

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

colour	meaning
ordinary blue	entering a section — navigation, not an action
green	gentle, read-only, status; or the deliberate undo half of a pair
teal	an ordinary working change
amber	strong intervention, a destructive action, or a secret export

Colour does not replace the operation kind, the confirmation, or the log — it helps you read the character of neighbouring buttons inside one frame.

A command's visible description is deliberately short: its job is to let you decide in a couple of seconds. The full explanation, the Windows terms, the risk, and the rollback live in the tooltip. The confirmation dialog for a dangerous operation shows the full text.

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

```
config\       command definitions, terminal history, window settings
system_core\  services, the window, 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.

The Sections

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

section	about	in detail
Linux subsystem	features and updates, install from network or file, status, backup, clone, move, register disks	`tools\WSL_TOOLKIT_RU.md`
Virtualisation	read-only status, Hyper-V or third-party mode, coexistence, toggles with a boot-config backup	`tools\VIRTUALIZATION_SWITCHER_RU.md`
Networking	diagnostics, backup and restore, proxy	`tools\NETWORK_CONNECTIVITY_RU.md`
Adapters and connections	adapters, sign-in to shared folders, Wi-Fi profiles, quick modes	`tools\NETWORK_CONNECTIVITY_RU.md`
Hosts and Bitrix	address override, endpoint detection, name and port status, byte-exact restore	`tools\BITRIX_HOSTS_RU.md`
Default apps	snapshot and comparison of defaults, policy protection	`tools\DEFAULT_APPS_GUARD_RU.md`
Association defence	Microsoft built-in apps, reinstall blocking, change tracking	`tools\ASSOCIATION_DEFENSE_RU.md`
Hardware and drivers	blocking drivers from updates, graphics restrictions, driver store backup, HDMI audio	`tools\HARDWARE_DRIVER_GUARD_RU.md`
Disks	inventory, recovery environment, SSD wizard	`tools\STORAGE_DISK_PROCEDURES_RU.md`
SSH keys	export and import	`tools\OPENSSH_KEYKIT_RU.md`
Certificates	export and import, working with stores	`tools\CERTIFICATE_KEYKIT_RU.md`

section	about	in detail
Browser bookmarks	building a master copy, merging, restoring	`tools\BROWSER_BOOKMARKS_MASTER_RU.md`
Maintenance	project cleanup, assistant memory backup	`tools\MAINTENANCE_CLEANUP_RU.md`
Migration	gathering every credential into one folder with an inventory	`tools\MIGRATION_RU.md`

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

```
runtime\python.exe -m py_compile system_core\ui_nicegui\app.py
system_core\services\devops_tools.py system_core\core\jobs.py
runtime\python.exe system_core\ui_nicegui\app.py --smoke
runtime\python.exe system_core\doctor.py
```

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_docs_pdf.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

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

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, 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.

Workbench naming

One shared vocabulary across all Audion projects: **Source**, **Add file...**, **Target**, **Reset**, **Delete**, **List**. In Russian: **Источник**, **Добавить файл...**, **Назначение**, **Сбросить**, **Удалить**, **Список**.

`Reset` restores the project folders and deletes no files. `Delete` clears the current source and target only after confirmation. The words `Destination`, `Clear`, «Цель», and «Очистить» are not used for these controls.