

Project File Map

[Русский](#) · [Reference](#) · [Install](#)

Where everything lives. This document is for whoever maintains the project; a user does not need it.

Working folders (`input`, `output`, `logs`, `report`, `release`, `workspace`, `runtime`, `wheelhouse`) are described by role rather than file by file — their contents are produced by runs and rebuilt.

Root

File	What it is
<code>Start.exe</code>	Compiled window launcher: opens <code>App\AudionAddressProcessor.exe</code>
<code>address_cli.cmd</code>	The console command: the same operations without a window; the window itself stands on it
<code>builder_main.cmd</code>	Portable environment build
<code>cleanup_project.cmd</code>	Cleanup of generated artifacts
<code>pytest.ini</code>	Test settings
<code>release_manifest.json</code>	Manifest of the last release build
<code>README.md</code>	Public project description

CMD files must stay UTF-8 without BOM with CRLF line endings. After editing, run `install\Check-CmdEncoding.cmd`.

App and Source — The Window

Path	What it is
Source/	The window sources: Avalonia 12 / .NET 10. <code>Core/</code> — running the command, the manifest, field values, the journal, settings; <code>Views/</code> — the form, panels, dialogs; <code>MainWindow.*.cs</code> — sections, working area, running, OKTMO; <code>Ps/</code> — source of the program's own PowerShell; <code>Tools/</code> — icons and the check without a mouse
Source/config/i18n.json	The English pairs of every Russian text of the window
App/	The built window (<code>AudionAddressProcessor.exe</code>) and the own PowerShell <code>AudionPs.exe</code> for the build scripts; built by <code>install\Build-App.cmd</code> , travels with a release as it is

config — Settings

File	What it is
project.yaml	Engine settings: city, reference column, target columns, linked columns, safety limits
window_layout.yaml	The layout of the window: sections, the operations in them, the buttons of the working area and of the result panel, icons
tool_manifest.yaml	Every operation and field of the window; the interface is built from it
version.json	Version and the language of the root README
path_history.json	History and pins of source and target paths
window_settings.json	The window's settings: size, language, theme, font (made by the window itself)
oktmo_current_keys.txt	Current OKTMO search keys
oktmo_region_pins.json	Pinned regions
oktmo_municipality_pins.json	Pinned municipalities

File	What it is
<code>oktmo_pins_bundle.json</code>	The combined pin set

system_core — Code

Top Level

File	What it does
<code>Ultimate_GT_Aligner.py</code>	The matching engine: address parsing, passes, linked column transfer
<code>safe_table_join.py</code>	Comparison of two workbooks and column transfer through Excel
<code>excel_layout.py</code>	Result workbook formatting: column widths, text wrap, fill
<code>remove_empty_rows.py</code>	Removal of entirely empty rows
<code>Parser_Diagnostic.py</code>	Address parsing diagnostics
<code>inspect_md_tables.py</code>	Markdown table inspection
<code>doctor.py</code>	Environment and import checks
<code>main.py</code>	Direct matching run without the window
<code>cli.py</code>	The console command: <code>describe</code> , <code>run</code> , <code>options</code> , <code>oktmo</code> , <code>serve</code> , <code>stop-office</code> , <code>version</code>
<code>office_session.py</code>	Keeps track of an operation's own Excel and Word, so a stop closes only them

address_engine — The Address Engine

File	What it does
<code>audion_address_core.py</code>	Address decomposition, recognition of address-like text
<code>address_slots.py</code>	Slots: definition, filling, address line assembly
<code>slot_fill.py</code>	Filling slots from evidence
<code>models.py</code>	Data structures: observation, address parts, fingerprint, cluster

File	What it does
<code>source_intake.py</code>	Reading sources in every supported format
<code>source_table_assembly.py</code>	Collecting addresses from sources into a workbook
<code>source_cleanup.py</code>	Filtering out Office temporary files
<code>reference_builder.py</code>	Building the reference directory; slot and column definitions
<code>reference_processor.py</code>	Sorting and decomposing the directory by slots
<code>oktmo_lookup.py</code>	OKTMO index, disk cache, region / municipality / settlement lookup
<code>oktmo_user_keys.py</code>	The project key file and pins
<code>rosstat_update.py</code>	Downloading and updating the Rosstat registry
<code>region_reference.py</code>	Region reference
<code>russian_morphology.py</code>	Grammatical forms of names
<code>universal_address.py</code>	The common address normalizer
<code>normalizer_runtime.py</code>	Configuring the normalizer per run
<code>address_diagnostics.py</code>	Diagnostic extracts
<code>legacy_office_converter.py</code>	DOC/XLS conversion

services — The Service Layer

`address_aligner_service.py` is the only file. Every window command points to a function here: the service takes the parameters, calls the engine, and writes the log, the report, and the artifact list.

core — Core

File	What it does
<code>jobs.py</code>	Running operations, cancellation, progress, the live line of a long stage
<code>operation_model.py</code>	The rules that assemble the parameters of a run: defaults, conditions for showing fields, slot order; the manifest as the window takes it

File	What it does
<code>window_layout.py</code>	Reading <code>window_layout.yaml</code> and checking that every operation stands in the window
<code>manifest.py</code>	Parsing <code>tool_manifest.yaml</code>
<code>paths.py</code>	Managed project folders
<code>config.py</code>	Reading settings
<code>ansi_terminal.py</code>	Stripping the colour control codes from program output
<code>cmd_encoding.py</code>	CMD encoding checks
<code>output_decode.py</code>	Decoding subprocess output
<code>logging_utils.py</code>	Logging

Other

Path	What it is
<code>icons/</code>	The program icon
<code>start_launcher/</code>	Source of <code>Start.exe</code>

install — Install and Build

File	What it does
<code>Build_Portable_Env.cmd</code> / <code>.ps1</code>	Portable environment build
<code>install_portable_offline.cmd</code>	Install from the local <code>wheelhouse</code>
<code>verify_portable_env.cmd</code>	Verify the built environment
<code>init_folders.cmd</code>	Create the working folders
<code>Check-CmdEncoding.cmd</code>	Check CMD encodings
<code>Clean-Install-Cache.cmd</code> / <code>.ps1</code>	Clean the build cache

File	What it does
<code>Update-Requirements-Lock.cmd</code>	Update the dependency lock file
<code>Build-StartLauncher.cmd</code> / <code>.ps1</code>	Build <code>Start.exe</code>
<code>Build-App.cmd</code>	Build the window <code>App\</code> from <code>Source\</code>
<code>requirements_full.in</code>	Dependency list
<code>README_INSTALL.md</code>	Install notes

data — Data

Path	What it is
<code>regions_reference.yaml</code>	Region reference
<code>rosstat/data-*.csv</code>	The OKTMO registry, downloaded by a button in the window
<code>rosstat/oktmo_lookup_cache.json</code>	The parsed OKTMO index

docs — Documentation

File	What it is
<code>README_RU.md</code> , <code>README_EN.md</code>	Project description, the entry point
<code>USER_GUIDE_RU.md</code> , <code>USER_GUIDE_EN.md</code>	User guide
<code>OKTMO_RU.md</code> , <code>OKTMO_EN.md</code>	The OKTMO registry, keys, and pins
<code>REFERENCE_RU.md</code> , <code>REFERENCE_EN.md</code>	Technical reference
<code>INSTALL_RU.md</code> , <code>INSTALL_EN.md</code>	Install and launch
<code>tools/PROJECT_FILES_GUIDE_*.md</code>	This document

File	What it is
<code>tools/WINDOW_INVENTORY.md</code>	The inventory of the earlier window (NiceGUI) and its check against the new one: what is covered where (in Russian)
<code>PDF/</code>	Rendered PDFs of every document in two themes

PDFs are built by a separate tool from every `.md` in `docs`, nested folders included. The folder structure is preserved.

tests — Tests

File	What it covers
<code>test_address_aligner_regression.py</code>	The matching passes
<code>test_safe_table_join.py</code>	Moving columns by address
<code>test_excel_layout.py</code>	Result workbook formatting
<code>test_console_output.py</code>	Stripping colour codes from output, the encodings of subprocess output
<code>test_cli_door.py</code>	The console command: the same run parameters as the earlier window, stopping, Excel and Word
<code>test_window_avalonia.py</code>	The window and the command read the same keys; the English pairs; the self-check of the built window
<code>former_window_parameters.json</code>	The parameters of the 14 operations recorded from the earlier window on 11.10.2026: the command is checked against them

The window is checked not by a page in a browser but by itself: `App\AudionAddressProcessor.exe --self-check` builds every section and compares them with the manifest.

licenses — Licenses

License texts of third-party components, `THIRD_PARTY_NOTICES.txt`, `COMPONENTS.json`, the scan report, and the pack manifest. Produced by the Audion Build Licenses tool from the finished contents

of the project, not edited by hand.