

Security overview

The architecture, process by process: what the renderer can reach, what the sidecar can reach, and where the network surface would be if there were one. Start here.

Product Oxofolio 0.2.1 — Offline document preparation workbench for audit firms **Supplier** Chadura Tech Private Limited

1. What it is

A desktop application that prepares client documents on the machine it is installed on. It is not a service. There is no account, no server, no tenant and no vendor-side storage, so there is no vendor environment for a customer's data to be in.

2. Architecture

Three processes, all local:

- **Main process** (Electron, Node.js) — owns the filesystem, the dialogs and the application's own storage. The only component with access to either.
- **Renderer** (Chromium, Vue) — the interface. Sandboxed, context-isolated, no Node integration, and reaches the main process only through the 97 named channels listed below.
- **Sidecar** (Python, frozen with PyInstaller) — does the document work. Speaks newline-delimited JSON over stdin and stdout. It opens no sockets and binds no ports; the pipe to its parent is its only means of communication.

Client documents are read by the sidecar and by the renderer's PDF viewer. They are never copied anywhere outside the folder they were found in, except into the output directories named below.

3. Network

The application opens no sockets and makes no network connections. There is no telemetry, no update check, no crash reporting and no licence call-home; licences are verified offline against a public key shipped with the application. No document, filename, folder path or other client data is transmitted anywhere, by any means.

One qualification, stated here rather than left for you to find. The interface carries 2 links to our website:

- feedback — <https://oxofolio.com/feedback/?from=app>
- interviews — <https://oxofolio.com/interviews/?from=app>

Activating one calls `shell.openExternal`, which hands the address to the operating system and opens the user's own browser. **That browser then makes a request, and we can see it** — the visit reaches our web server with the user's IP address, and the `from=app` marker tells us it came from an installation rather than from the website. The application itself still opens no connection, and nothing about the user's documents is included in the address. But a document that said "no network connections of any kind" and

left this out would be answering the wrong question, so it is set out in full.

The addresses are held in the main process and the interface refers to them by name. Code running in the renderer cannot ask for an arbitrary address to be opened; only the ones listed above can be.

Blocking outbound traffic on the host does not affect the application. The links simply fail to load in the browser, and every feature continues to work.

The no-connection property is enforced rather than asserted. `scripts/check-no-network.mjs` fails the build if any of these patterns appears anywhere in the shipped source:

- `/\bfetch\s*\(/`
- `/\bXMLHttpRequest\b/`
- `/\bWebSocket\b/`
- `/from\s+['"]node:(https?|net|dgram|dns)['"]/`
- `/require\(['"](https?|net|dgram|dns)['"]\)/`
- `/\bimport\s+(requests|urllib|httpx|socket)\b/`
- `/^\s*import\s+socket\b/m`

scanned across `electron/**/*.{js,cjs,mjs}`, `src/**/*.{js,vue}`, `python/**/*.{py}`. It runs in `npm run gates`, which every release target depends on.

The renderer additionally runs under a Content Security Policy with `connect-src 'none'`, window opening is denied outright, and navigation is cancelled in the main process.

4. Renderer hardening

SETTING	VALUE
<code>contextIsolation</code>	<code>true</code>
<code>nodeIntegration</code>	<code>false</code>
<code>sandbox</code>	<code>true</code>
<code>webSecurity</code>	<code>true</code>

Content Security Policy in production builds:

- `default-src 'none'`
- `script-src 'self'`
- `style-src 'self' 'unsafe-inline'`
- `img-src 'self' data:`
- `font-src 'self'`
- `connect-src 'none'`

5. The IPC surface

Every channel the interface can invoke. There is no general-purpose bridge and no way to reach the filesystem or run a command except through one of these.

- `app:diagnostics`
- `app:usage`
- `assembly:seal`
- `assembly:state`
- `assembly:verify`
- `checks:duplicates`
- `checks:mixing`
- `clipboard:write`
- `compress:batch`
- `deck:edit`
- `deck:sanitise`
- `dialog:pickFile`
- `dialog:pickFiles`
- `dialog:pickFolder`
- `dialog:savePath`
- `doc:format`
- `doc:personalData`
- `doc:preview`
- `doc:read`
- `doc:readDecrypted`
- `docs:states`
- `encrypt:files`
- `export:pdf`
- `extract:export`
- `extract:fields`
- `files:stats`
- `folders:bookmark`
- `folders:forget`
- `folders:forgetAll`
- `folders:list`
- `folders:open`
- `images:annotate`
- `images:deskew`
- `images:fromPdf`
- `images:preview`

- `images:toPdf`
- `licence:install`
- `licence:state`
- `licence:status`
- `log:write`
- `logs:list`
- `ocr:status`
- `office:thumbs`
- `openers:list`
- `openers:open`
- `pages:merge`
- `pages:preview`
- `pages:reorder`
- `pages:split`
- `pdf:scan`
- `pdf:shape`
- `pdf:toDeck`
- `pdf:toDocument`
- `pdf:toWorkbook`
- `pdf:unlockBatch`
- `pdf:unlockOne`
- `pii:redact`
- `pii:scan`
- `pw:forget`
- `pw:forgetAll`
- `pw:known`
- `pw:status`
- `record:build`
- `record:fields`
- `record:particulars`
- `record:saveFirm`
- `rename:apply`
- `rename:plan`
- `render:covers`
- `render:thumbnails`
- `retention:erase`
- `retention:plan`

- `retention:status`
- `sample:draw`
- `sample:reperform`
- `sample:size`
- `settings:folder`
- `settings:saveFolder`
- `sheet:edit`
- `sheet:layout`
- `sheet:preview`
- `sheet:sanitise`
- `sheet:structure`
- `shell:openLink`
- `shell:revealPath`
- `sig:add`
- `sig:list`
- `sig:place`
- `sig:remove`
- `stamp:batch`
- `stamp:index`
- `vault:clear`
- `vault:loadCsv`
- `vault:loadMaster`
- `vault:status`
- `watch:start`
- `watch:stop`

6. The sidecar surface

Every method the Python process answers to. An unrecognised method is refused.

- `assembly_seal`
- `assembly_verify`
- `build_record`
- `check_mixing`
- `compress_batch`
- `deck_edit`
- `deck_sanitise`
- `decrypt_to_bytes`

- diagnostics
- doc_format
- doc_personal_data
- doc_preview
- document_states
- encrypt_files
- export_transactions
- extract_fields
- file_stats
- find_duplicates
- image_annotate
- image_deskew
- image_preview
- images_to_pdf
- logs_list
- ocr_status
- office_thumbnails
- pages_merge
- pages_preview
- pages_reorder
- pages_split
- pdf_shape
- pdf_to_deck
- pdf_to_document
- pdf_to_images
- pdf_to_workbook
- pii_scan
- redact_files
- rename_apply
- rename_plan
- render_covers
- render_thumbnails
- retention_erase
- retention_plan
- retention_status
- sample_draw
- sample_reperform

- sample_size
- scan
- sheet_edit
- sheet_layout
- sheet_preview
- sheet_sanitise
- sheet_structure
- sign_place
- stamp_batch
- stamp_index
- unlock_batch
- unlock_one
- vault_clear
- vault_load_csv
- vault_load_master
- vault_status
- write_log

7. Where data is written

Inside a client folder, the application only ever creates these directories, and writes its outputs inside them:

- _assembly/
- _bundles/
- _compressed/
- _corpus_out/
- _edited/
- _encrypted/
- _erasure/
- _extracted/
- _marked/
- _ocr/
- _record/
- _redacted/
- _renamed/
- _sample/
- _signed/
- _split/

- `_stamped/`
- `_unlocked/`

Every document operation writes a new file and leaves the source untouched — unlock, compress, combine, convert, redact, encrypt, split, stamp, OCR, image export and image mark-up all work this way. There is no app-managed document store and there will not be one: a second copy of client statements outside the firm's backup and access controls is a security regression rather than a convenience.

One feature writes in place, deliberately: the spreadsheet viewer's light edit mode. A reviewer correcting three cells should not have to round-trip through Excel to do it. Because that is a change to a client's file, it is constrained in three ways that are worth checking rather than taking on trust:

- **It is off by default.** Editing is a mode the user turns on; no keystroke begins altering a document that is merely being read.
- **The file as it arrived is preserved first.** Before the first write it is copied to `_originals/`, hashed with SHA-256, and never refreshed afterwards — so a later edit cannot quietly become "the original".
- **The change is recorded.** Source and output hashes, which cells changed, and whether any formula was replaced, in the same processing log every other operation writes.

Edits are applied to the file's own XML rather than by re-saving the workbook, so formulas, cached results, formatting, charts, pivot tables and macros are carried across untouched. Structural changes that would move a formula's references are refused outright rather than approximated.

In the application's own storage (`userData`, outside any client folder):

FILE	CONTENTS
<code>folder-settings.json</code>	Per-folder preferences, the recent and pinned folder list, retention basis, audit-file seals and the engagement fingerprint used by the client-mixing check. Includes client folder paths.
<code>firm.json</code>	The firm's own Article 30 particulars: name, address, privacy contact, data protection officer.
<code>passwords.json</code>	Document passwords the operator chose to remember, encrypted with the operating system keystore. Written only when the OS can encrypt; never in plain text.
<code>signatures/</code>	Signature and stamp images the operator added, as PNG. Capped at three.
<code>licence.key</code>	The installed licence and its signature.
<code>install-time.json</code>	First-run date, so the trial can be counted; the date last seen, so a clock moved backwards can be detected; and the machine identifier described under Licensing below. No document, folder or client information.

8. Licensing

Licences are files, signed with Ed25519 and verified against a public key compiled into the application. Verification is arithmetic performed on this machine. Nothing is transmitted, there is no activation step, and an installation that never touches a network works indefinitely.

A licence is issued for one computer, so the application derives an identifier for the machine it is running on and compares it with the one in the licence. That identifier is a SHA-256 hash of two values read from the operating system: an installation identifier, which changes when the operating system is reinstalled, and a hardware serial number. Because it is a hash, the values underneath cannot be recovered from it.

What this means in practice, stated plainly because it is the part a reviewer will want to be sure of:

- The identifier is derived on demand, held in memory, and written only to the application's own storage on this machine. It is not transmitted anywhere.
- It identifies a computer, not a person. It contains no user name, no network address, and nothing about the documents processed.
- It is displayed in Settings. Obtaining a licence means the operator reading it and sending it to us deliberately -- there is no step in which the application discloses anything on its own.
- Reinstalling the operating system or replacing the computer produces a new identifier, and the existing licence stops matching. The application then warns for 7 days before it stops processing, so a failed disk does not halt work while a replacement licence is arranged.
- Where the identifier cannot be read in full -- a hardened or virtualised host is the usual reason -- the check is skipped rather than failed. No one is blocked because a probe did not answer.