Kit de documentation

Documentation — Quality Control — EN

1 Introduction

This document is a translation in substance. The reference is the original document in French; extensions and changes are always made there.

Producing a document goes through successive layers of defence. Prevention: the stylesheets make certain errors impossible to write. Detection: *validators* examine the file produced before it is delivered. Audit: a periodic check of what prevention and detection do not see.

None of these layers is enough on its own. An error that a *stylesheet* cannot prevent must be caught downstream; an error that no tool can detect remains a matter for an audit. The sections that follow describe each layer, the artifacts that implement it, and the order in which they are called.

Intended for **Claude AI** as the single reference for all questions of audit, validation and quality. Every "Audit" or "*Quality Control*" reference in the Kit's other documents points to this source.

1.1 The layers of defence

Every generated document or file potentially goes through the following levels of control, from upstream to downstream:

- **Layer 1 — Runtime prevention:** checks built into the *stylesheet* functions. They block errors before the document is even produced. Examples: makeImageRun refuses an invalid format, makeHeader refuses an empty parameter.
- **Layer 2 — Session detection:** external scripts called around each *generation*. Two distinct moments. Upstream, kit_check_setters.py inspects the generator's source before execution. Downstream, the *validators* inspect the file produced before delivery — kit_validate_docx.js and kit_validate_xlsx.py. The environment prerequisites, checked before anything else, are added to these (§3.14).
- **Layer 3 — Periodic audit:** cross-cutting reviews done in a dedicated session. They detect gradual drift. Examples: *JSDoc* and Reference audit, version *alignment* audit, *L10N* coverage audit.

1.2 Principle — fail loud early

An error detected late costs proportionally more. A silent error is never corrected. Whenever a validation can be moved earlier in the *pipeline* (runtime rather than session-only, session-only rather than periodic audit), it must be.

> 💡 *If a check can become runtime, that is always better. If runtime is not possible, make it session-only. The periodic audit is the last safety net, not the default way of working.*

2 Layer 1 — Runtime checks in the stylesheets

Runtime checks live in the exported functions of the stylesheets. They throw an explicit *JavaScript* exception on an incorrect call — the generator fails immediately with a message identifying the offending line.

2.1 injectCustomProps — mandatory fingerprint

Stamps the *custom properties* StylesheetVersion and GeneratedAt into the docx. A mandatory call after Packer.toBuffer() in every generator script.

```
Packer.toBuffer(doc).then(buf => {
  fs.writeFileSync(OUTFILE, style.injectCustomProps(buf, TS));
});
```

Documents without these *custom properties* are rejected by gen-kit-site.js at HTML conversion.

2.2 makeImageRun — dimension and format validation

Enforces the type parameter of ImageRun and validates width/height > 0, format ∈ {png, jpg, gif, bmp, svg}. Throws an explicit error instead of silently producing a corrupt file (.docx).

2.3 makeHeader — parameter validation

makeHeader(title, subtitle) throws an explicit error if title or subtitle is empty, undefined or not a string. In line with the *fail loud early* principle — every generator must pass both. See *General Reference* §10.8.

2.4 Level discipline — the level comes from the heading

Since General v1.70 and HTML v1.43, the section level no longer lives in the function names. h1, h2 and h3 set a current level at module scope, and every other family reads it: writing paragraph('text') under an h2 produces a level 2 paragraph without the generator having to know. A mismatch between a suffix and its parent heading is no longer an error to watch for; it has become inexpressible.

The static validation of the source that used to sit here has therefore lost its purpose, and its tool has been removed from the Kit. Level checking remains and has moved one step down: check 11 of kit_validate_docx.js reads the actual indentation of every paragraph in the document produced, without presuming anything about how the generator is written. It is blocking, and it is now the sole authority on the matter.

2.5 Embedded decision rules

Documented in the Reference of each *stylesheet*, not enforceable at runtime but clearly explicit:

- **Spread mandatory on functions returning an Array:** *General Reference* §13.

- **Transitions between Table functions:** releaseParagraph() is mandatory between two adjacent Table blocks — *General Reference* §13.4.
- **Numbering:** starts at §1, never §0 — *General Reference* §1.2.
- **prompt:** reserved for copy-and-paste dialogue. For any code, yamlStyle.rawBlock or codeBlock — *General Reference* §8.

3 Layer 2 — Stable Kit artifacts and session prerequisites

Stable scripts that live in the Kit's *working tree* and are copied by the user into every project that needs them. The kit_ prefix signals their origin — Unix snake_case naming, a formal exception to the pattern in Naming Convention §2, named in Naming Convention §3.4. The roles: *validators* (return 0 if all is well, non-zero otherwise, and block delivery on failure), renderers (produce a deterministic *artifact* — the same input and the same *Registry context* give the same output), and helpers (build reusable fragments without producing a deliverable on their own).

An important distinction from *ad hoc generators* (gen-*.js): stable artifacts have stable logic, reused session after session. They are NOT regenerated at every session — only when their logic evolves. The evolution is recorded in a changelog block at the top of the file itself (Unix convention), just like the Kit stylesheets, not in an external document.

Two rules in it apply to the whole chain: §3.13 sets the mandatory calling order, §3.14 documents the environment prerequisites to check before anything runs.

3.1 kit_validate_docx.js — validator for docx produced by the Kit

Checks the docx produced by the Kit's **Node.js** stylesheets. A retained Kit *artifact* — a standalone **Node.js** script with no external *npm* dependency (it uses the standard library's zlib to read the docx ZIP, plus a small standard-library XML parser for the structural *OOXML* checks). This section is the single source of truth for its contract: other documents refer to it without restating it.

- **Fingerprint check:** custom.xml present with StylesheetVersion and GeneratedAt.
- **Version check:** the stamped StylesheetVersion matches the version passed with --version. Absent or lower: reported. The site refuses a document only without a *fingerprint* or below the *Registry*'s minDocumentVersion floor — *Pipeline HTML* §7.4.
- **Consistency check:** the number of <w:tbl> in document.xml matches the number of Table blocks expected from the *generation AST*.
- **Paragraphs-in-tables check:** if tables are present, the count of inner paragraphs must be greater than zero. Zero with tables present signals a guaranteed loss — the doc.paragraphs trap, *Pipeline HTML* §7.1.

- **Non-empty body check:** a document with 0 paragraphs AND 0 tables while the *custom properties* are present almost always indicates a silent docx-js corruption — the <rootKey>w:p</rootKey> pattern. Established cause: two distinct instances of the docx module in the same chain. Check 13 is the post-generation safety net; the cause and its prevention are in §3.14.
- **word/media/ extensions check:** every extension present in word/media/ must be declared as a Default Extension in [Content_Types].xml. An orphaned extension — typically.undefined, produced by an ImageRun without the type parameter — makes the file unreadable by Word with no explicit error message. See *General Reference* §13.3.
- **Canonical <w:rPr>/<w:pPr> order check:** direct children checked against the *OOXML* order §17.3.1 and §17.3.2. Detects the docx@8.5.0 bug where rFonts is serialised last instead of in 2nd position. Word 365 applies the XSD schema strictly; *LibreOffice* tolerates it.
- **Non-empty table containers check:** <w:tr> with no direct <w:tc>, or <w:tbl> with no direct <w:tr>. Word 365 refuses to open; *LibreOffice* tolerates it silently.
- **<w:r> child namespaces check:** restricted allow-list (w:, mc:, w14:, w15:, w16se:, w16cid:, w16:). Catches silent docx library bugs such as <type>*tab*</type>, serialised by docx@8.5.0 on new TextRun({ children: [{ type: '*tab*' }] }).
- **Heading level and indentation consistency check:** for every top-level paragraph of the body, the attribute <w:ind w:left="X"/> must be in the valid set for the level of the most recent Heading. *Model* taken from the General *stylesheet* helpers: L1 = {567, 967, 1157}, L2 = {1350, 1750, 1940}, L3 = {2100, 2500, 2690}. A glossary document, recognised by its name — Glossaire - Termes, Glossary - Terms, *Stylesheet* - Glossary - Reference —, additionally accepts 1350 and 1550 under a level 1 heading: the term name and the definition of the Glossary *stylesheet*. Elsewhere those values remain an anomaly.
- **NBSP typographic warnings check:** detects numeric sequences grouped in threes and followed by a known currency or typographic unit (€ $ £ ¥ % m² m³ km kg ha ares hectares °C) whose internal spaces are ordinary (U+0020) instead of *NBSP* (U+00A0). Non-blocking warnings. Recommended Kit helper: style.formatCurrency(amount, opts).
- **Check 14 — Strict global XML:** for every .xml or.rels entry in the zip, checks that there are no tags with an invalid name (e.g. a stray <0/>

injected by forgetting the *spread* on makeTable) and that openings and closings balance, using the small tokenizeTags parser.

- **Check 15 — Heading numbering monotony:** extracts the sequence of Heading1/2/3 with their Kit number and checks that each counter advances by +1 within its parent and starts at.1. Catches the h2(level, text) bug instead of h2(number, text), which produces a valid file with broken numbering.
- **Check 16 — Orphaned X.1:** *anti-pattern* documented in *Kit Prompt* §4.1 and *General Reference* §1.2 — a parent with a single subheading is suspect; its content should move up under the parent. Grouped by parent, solo cases flagged.
- **Check 17 — Isolated note or tip after a heading:** *anti-pattern* documented in *General* §1.2 — never a tip or note after a heading without a parent paragraph. Detected by the width of the first column of the <w:tbl> immediately following a top-level heading: TIP_BULB_COL_WIDTH for a tip, and for a note membership of NOTE_LABEL_COL_WIDTHS, since the label width has not been unique since General v1.68. This check stayed inert from its addition until 22 August: its splitting of top-level elements compared by prefix, so the paragraph-properties tag answered to the paragraph's own and inflated the depth. The body's first paragraph swallowed everything else and the loop never again saw a heading followed by a table.
- **Check 18 — Cover page:** consistency between the <w:titlePg/> flag of the sectPr and the header or footer references of type first. Without the flag, Word ignores those references and applies the default header and footer from page 1 — the *cover page* loses its bareness. Root cause: style.pageProps does not carry the flag; the documented pattern is properties: {...style.pageProps, titlePage: true } — *General Reference* §10.4. Symmetrical check: the flag without a first reference is also reported.
- **Check 19 — Format identification:** first entry of the archive. Packer puts the word/ folder first; any rewrite by a library that does not preserve the order pushes it behind the metadata, and the document is then served as a generic archive — renamed to.zip on download. The document remains valid and opens without warning: this check is the only point in the chain that sees the defect before the recipient does.

Usage: node kit_validate_docx.js 'Mon Document (TS).docx' for a simple validation, or node kit_validate_docx.js 'Mon Document (TS).docx' --version 1.67 to check the *fingerprint* as well. Exit code 0 = valid, 1 = anomaly.

Note: *The validator runs after every generation of a file (.docx), before the file is copied to the output folder and before it is included in a deployment ZIP. A non-zero exit blocks*

delivery — never bypass it.

3.2 kit_validate_xlsx.py — xlsx validator

Checks the xlsx produced by the pipelines that create them — mainly the Trading project. The checks below make up its complete contract.

- **Checks:** zip structure, formatCode balance, cell references, sheet references in formulas, openpyxl double read.
- **Usage:** *python* kit_validate_xlsx.py document.xlsx [--strict].
- **Retained artifact:** lives in the *working tree* of the Kit and of the projects that produce xlsx. Not regenerated at every session.

3.3 kit_render_cover_sheet.py — cover page renderer

A deterministic *renderer* that produces the A4 300 dpi PNG of the *Cover Sheet* (Kit or *consumer project*) from the matching *Cover Sheet* couplet.docx. All specific values (HEADER_TEXT, §2 sections, overridable *PCL* such as TAB_NUMBERED_MAX) are read from the .docx by *pandoc AST* parsing. No Kit-specific value is hard-coded in the *renderer* — in line with the contract documented in *Cover Sheet* §3.

CLI: optional, combinable arguments. Backward-compatible mode preserved — calling it without arguments resolves the current Kit *Cover Sheet* automatically through Registry.documents['cover-sheet'].ts.

- **--cover-sheet PATH:** *Cover Sheet* file (.docx) to render. If absent, the current Kit *Cover Sheet* is resolved automatically through the *Registry*.
- **--registry PATH:** project *Registry* (default./Kit - *Registry.js*). Source of project.docLanguage and rendering.coverSheet.
- **--output PATH:** output PNG. If absent, derived from the source file (.docx).
- **--language XX:** language override (FR | EN | DE | LU). Drives the Kit's localised strings (subtitle, quote, toc_title).
- **Localised numbered-section markers:** FR='numérot', EN='numbered', DE='nummeriert', LU='nummeréiert'. Aligned with the semantics of the FR marker (a root longer than the short form of the label) — avoids the false positive documented in the *AI* News session, where the substring 'number' matched 'no number' in the *Tab* cells of the reserved rows of an English project *Cover Sheet*. v3 introduced the initial *i18n*; v4 hardens the EN/DE/LU markers.
- **Defensive access to the project Registry:** reading rendering.coverSheet.sectionHMode uses a graceful 'FIXED' fallback if the rendering section is absent. Without this fallback, an immediate KeyError on a project *Registry* that follows the historical skeleton of

Common Structure §5.2. The §5.2 skeleton is extended in parallel to document the section — defence in depth.

- **TAB_NUMBERED_MAX derived automatically:** the value is computed from the number of sections extracted from §2 and takes precedence over the Kit default. The *PCL* §3 TAB_NUMBERED_MAX becomes optional — useful only to reserve extra tabs explicitly.

Called on every *PCL* change to a *Cover Sheet*. A standalone *Python* script — dependencies *Pillow*, zoneinfo, *pandoc*. Couplet rule: the PNG produced carries the same TS as the source *Cover Sheet* file (.docx).

3.4 kit_check_setters.py — setter discipline linter

The only static *linter* of upstream *Layer 2*. Checks that a generator script calls every mandatory *setter* at the top of the file, according to the Registry.requires flags. Primary target: the HTML generators gen-{prefix}-site.js — omitting setGlossaryTerms silently produces a site without glossary hotlinks.

- **Rule:** each flag set to true in Registry.requires requires a call to the matching *setter* (glossary calls setGlossaryTerms, documentTitles setDocumentTitles, brands setBrands, variableNames setVariableNames, hassEntities setHassEntities, textHighlights setTextHighlights). Added to these: setLanguage, always, then depending on the target: setDocumentAuthor and setDocumentSiteBase for a .docx generator; setWebAuthor, setFamilyDomains, and setGlossaryHref when the project publishes several glossaries, for an HTML generator.
- **Usage:** *python* kit_check_setters.py gen-document.js [--*registry* path] [--html] [--strict]. The --html flag applies the requirements of an HTML generator. The --strict flag promotes setLanguage from warning to error.
- **Exit codes:** 0 = all required *setters* called; 1 = violations; 2 = *Registry.js* not found.
- **Paired with:** *Pipeline HTML* §7.7, which documents the list of mandatory *setters* on the site generator side. The *linter* automates what the documentation prescribes.
- **Retained artifact:** lives in the *working tree* of the Kit and of every project that produces an HTML site. Not regenerated at every session.

Note: *Arising from the brief Kit - Adaptations souhaitables (item 4) — promoting the documentation of mandatory setters to an automatic linter.*

Note: *Conservative static detection. The linter does regex matching on the source text of the .js, without following conditional flow. Consequence: a setter called under if (Registry.requires.X) {...} with X=false in the Registry is reported as "Setter detected"*

even if the call is dead code at runtime. This behaviour is deliberately imprecise in the conservative direction — it accepts too much and refuses too little — so it never blocks a correct delivery. False positive catalogued in the AI News session; no fix planned (the cost of a flow-aware JS parser is out of proportion to the noise).

3.5 kit_gen_glossaire_html.js — HTML glossary generator helper

A stable Kit helper that generates the glossaire.html page from [Préfixe] - Glossary - Terms.js. It bypasses the *pandoc AST pipeline* entirely — the proprietary structure of the glossary file (.docx) (sectionBanner, letterHeader, inline aliases) cannot be read by the *AST* converter. Full normative contract in *HTML Reference* §14.8 and §15.

- **Absolute rule:** the HTML glossary is ALWAYS generated by this helper, NEVER reimplemented in gen-kit-site.js or in the gen-[projet]-site.js of *consumer projects*. Strict centralisation to avoid silent divergence.
- **Usage:** const { generateGlossaireHtml } = require('./kit_gen_glossaire_html.js'); generateGlossaireHtml({ termsPath, outputPath, htmlStyle, config }).
- **Retained artifact:** lives in the *working tree* of the Kit and of every *consumer project* that publishes a glossary.
- **Dependencies:** no external *npm*. Reads Terms.js via require, uses the .gloss-* classes of the HTML *stylesheet*, and the title and breadcrumb localised through htmlStyle.getStrings().glossaryTitle (added in HTML v1.23).

3.6 kit_patch_helpers.py — Python helpers for surgical patches to .docx

A *Python* helper that builds XML fragments conforming to the General *stylesheet* when making surgical changes to an existing file (.docx). It covers the complex structures (1×2 note, etc.) that become a source of bugs when produced by cloning a template or by manual construction. Fragments are built FROM SCRATCH from the synchronised Kit constants — not by cloning a template from the target document.

- **Absolute rule:** to insert a note, tip or table block into an existing file (.docx) by *surgical patch* (Dialogue Commands §2.4), the helpers of this module must be used — never clone a fragment of the target document by hand.
- **Current API:** build_note_block(content, level=1, lang='FR') — returns an XML fragment <w:tbl>...</w:tbl> representing a Kit note at level 1, 2 or 3.
- **Synchronisation:** Kit constants hard-coded and explicitly synchronised with the General Code at each evolution. A documented dependency — not silent drift.

- **Dependencies:** none external. *Python* standard library only. No regex when writing a structured file.
- **Retained artifact:** lives in the Kit's *working tree*. Copied as is when setting up or updating a project that needs it.

3.7 kit_check_markup.py — markup and constants agreement check

Two checks that no Kit tool performed, both born of defects that slipped through the existing chains. A text diff does not see a loss of colour: a regenerated document can carry exactly the same words, in the same order, and have lost the markup of several terms or several brands.

- **Markup check:** counts the runs carrying each kind of markup — glossary, brands, variable names, entities, bold, hyperlinks, images — in the source file and in the document produced, then compares. Any loss is reported. The --strict mode also reports gains, for a regeneration that must not change the content; --added N declares the number of expected bold labels.
- **Note label width check:** the value lives in three copies — the localizedStrings table of General, NOTE_LABEL_COL_WIDTHS of the *validator*, NOTE_LABEL_COL_BY_LANG of the patch helpers. The duplication is deliberate, as neither tool can import the *stylesheet*. The check reads the three sources at run time and verifies that they agree.
- **Usage:** *python* kit_check_markup.py SOURCE.docx PRODUIT.docx [--strict] [--added N], or *python* kit_check_markup.py --widths [--dir DOSSIER]. Exit 0 if they agree, 1 on a discrepancy, 2 on a usage error.
- **Retained artifact:** lives in the *working tree* of the Kit and of any project that regenerates existing documents.

Note: *This check is the only one that sees a loss of markup. Two real defects were caught by it before delivery: eleven English terms stripped of their colour because the non-breaking spaces of the extracted text had been neutralised from the French surfaces, and five brands stripped of their small caps because a paragraph opening on a brand name had been taken for a bold-lead paragraph. Not a single word had changed in either case.*

3.8 kit_check_ossature.py — skeleton check between language variants

Compares the skeleton of a translation with that of its source. §17 of the *Prompt* asks for a translation faithful to the meaning, in a text that reads as if it had been written in its own language: counting words or comparing sentence by sentence would penalise exactly what the rule asks for.

- **What is measured:** the sequence of blocks, their type and level, the section numbering, the dimensions of each table, the number of items in each list.
- **What is not:** the number of words, the length of sentences, sentence-by-sentence correspondence. German compounds, English shortens,

and a translation that copied the length of the French would be a bad one.

- **What is reported:** a section, an item, a table row or a box present on one side and missing on the other. That is information lost or added, which §17 forbids.
- **Tolerance:** the translation note in §1, which the reference variant does not carry, is recognised and accepted. The --sans-tolerance option refuses it.
- **Usage:** *python* kit_check_ossature.py REFERENCE.docx VARIANTE.docx [AUTRES…]. Exit 0 if the skeletons match, 1 otherwise.

3.9 kit_extract_map.py — structure map of an existing document

Every regeneration starts from a map extracted from the binary. The tool reads the document's XML and returns its structure: block order, type, level, content, markup. It is read-only and uses no regular expressions.

- **Why it is stable while generators are not:** *Prompt* §4.1 requires every generator to be written from scratch, because a generator belongs to one document. Extraction does not share that property: it always does the same thing, and rewriting it at every session only produces variants of the same code, each with its own blind spots.
- **What it recognises:** headings, paragraphs and their variants by indentation, list items and continuations, note, tip and *prompt* boxes, raw and coloured code blocks, ordinary tables, metaTable of the YAML *stylesheet*, termName and definition of the Glossary *stylesheet*, images, hyperlinks, bridge and release paragraphs. An image, standing alone or in a cell, is stored as a file in the <map>-images folder and put back at regeneration; the bulb of a tip box is not captured, the *stylesheet* redraws it. A table records whether its first row is a header, and is rebuilt without one otherwise; a single-column table is an ordinary table. The link of a cell is captured with its *anchor* and its target, resolved through the document relationships, and put back by hyperlinkCell.
- **What it cannot promise:** its repertoire is finite. A construction that is not in it is degraded without anything in the produced file saying so. That is the reason for §3.10.
- **Usage:** *python* kit_extract_map.py SOURCE.docx CIBLE.json.

3.10 kit_check_fidelite.py — extraction fidelity check

Extracts, regenerates, and compares the body XML byte for byte. A document that does not come back identical signals a construction that extraction cannot render. On 23 August 2026, four defects of this kind were found in a

single day, all silent and all invisible to a text diff; this check would have found them in one go.

- **The neutralisations without which the test measures something else:** the *timestamp* and the hyperlink relationship identifiers change at every *generation* without the rendering moving. The language comes from the file name suffix — regenerating a German document with the French glossary surfaces produces a discrepancy that says nothing about extraction. And the replayed chain must be the production one, neutralisation of the *pipeline*'s marks included.
- **What it does not promise:** it only sees the constructions present in the document set. A document using tomorrow a helper never used before would slip through. The check reduces the error surface without closing it: what is held to be safe is so in the light of what is known today, and that is the ordinary condition of any quality control. The day an unforeseen case arises, extraction is completed and the document set is checked again.
- **Scope:** a document produced by an earlier *stylesheet* version differs legitimately. The tool reports it separately rather than counting it as a failure.
- **Usage:** *python* kit_check_fidelite.py SOURCE.docx [...] --version {version active}. Exit 0 if everything comes back identical, 1 otherwise.
- **An unchanged document keeps its timestamp.** A fresh *timestamp* is required at every DELIVERY — *Kit Prompt* §4.1 — not at every generator run. A regeneration campaign compares the body produced with that of the source, using the same neutralisation as here; if identical, it discards the product and keeps the source. Without this comparison, the publication delta treats the whole document set as modified, the local *inventory* is rewritten in full, and the site's history loses its meaning: forty-six documents there carry the same minute.
- **The glossary is outside the check.** It is not regenerated by extraction — §3.11 — and the round trip tested here therefore makes no sense for it. It is counted separately in the summary, never as a discrepancy. The Reference of the glossary *stylesheet* follows the same path since 2026-09-17: it carries elements rendered by that *stylesheet* — banner, letter, term, definition — that the map cannot say again, and is regenerated by its session generator.

Note: *The artifacts call kit_extract_map.py by an absolute path, computed from their own location. A relative path tied them to the working directory: kit_check_ossature.py moreover called a name from before its promotion to a stable artifact, extract_map.py, and failed on every call without anything reporting it — the failure only showed at the first*

comparison of variants.

3.11 kit_gen_glossaire_docx.js — .docx glossary generator helper

Produces the glossary file (.docx) from [Préfixe] - Glossary - Terms.js. The counterpart of kit_gen_glossaire_html.js — §3.5: one source, two renderings.

- **The .docx glossary is never regenerated by extraction.** It is the only Kit document whose content lives in a module. Regenerated from its map read in the binary, it freezes: three terms added to Terms.js on 24 August stayed absent from the three variants until a manual check.
- **It calls the pipeline setters.** Without them, glossary terms and brands stay black in the definitions.
- **Outside the skeleton check.** Terms are sorted in the rendered language, so the groupings by letter differ from one variant to another. kit_check_ossature.py excludes it explicitly — §3.8.
- **The two renderings do not have the same structure.** Paper follows the sections declared in glossaryRubriques; the web is a single alphabetical index — §3.5, point 4bis. A *binder* is leafed through and gains entry points; a web page is searched by keyword and needs none. The content is the same: a single source, Terms.js.

3.12 kit_check_couplets.py — shared timestamp of couplets

Checks that every present member of a couplet carries the same *timestamp* — *Kit Prompt* §6.3. A missing member is reported without failing: a project does not necessarily have every *stylesheet*, nor a glossary in every language.

It also reads the version exported by each Code.js and compares it with the *Registry*'s: a couplet agreeing on its *timestamp* may still carry two different versions. A *stylesheet* that does not export STYLESHEET_VERSION is reported as unchecked rather than taken as agreeing.

- **Nothing checked this rule.** The *validator* reads a document in isolation, the fidelity check compares a round trip, the *fingerprint* gate looks at versions. None of them sees two files side by side. The Glossary and YAML couplets were broken on 23 August, the glossary’s on the 24th — all three times, the discrepancy was only found by reading a list of files.
- **What it does not check.** That the content of the members agrees. A .js and its Reference can carry the same minute and contradict each other — that is the job of reading.
- **Grouping is by prefix AND by role.** A directory where several projects coexist holds two distinct Glossary couplets, which have no reason to share a minute. Grouping by role alone declared them broken

by each other — which is what the check did when it was first put into service.

- **A couplet does not always have two members.** The *Cover Sheet*'s brings together a specification file (.docx) and a rendering (.png) per language — *Cover Sheet* §4. The glossary's, the module and one variant per language.

3.13 Absolute calling rule

No file is delivered without validation. A non-negotiable rule. The complete chain has the following links, in this order:

- **Environment prerequisites:** check that require('docx') resolves correctly, and require('*adm-zip*') before a site publication — see §3.14. This link comes before everything else: without it, the generator fails or produces an empty document.
- **Before any generation:** *python* kit_check_registry.py. A *Registry* outside the perimeter produces correct documents and loses its settings at the next update.
- **Before .js generation:** *python* kit_check_setters.py gen-document.js (for HTML generators, add --html).
- **After docx generation:** node kit_validate_docx.js document.docx --version {version active}.
- **After xlsx generation:** *python* kit_validate_xlsx.py document.xlsx.
- **After regenerating an existing document:** *python* kit_check_markup.py source.docx produit.docx. A text diff does not see a loss of markup; this check is the only net for it.
- **After regenerating a language variant:** *python* kit_check_ossature.py reference.docx variante.docx. Checks that no section or item has been lost or added.
- **After a change to extraction:** *python* kit_check_fidelite.py on the document set, with the active version. This link is not run at every delivery but every time the extraction tool changes, and before any regeneration campaign.
- **kit_check_couplets.py.** Before any delivery touching a couplet member. A broken couplet leaves the reader not knowing which member is authoritative.

No final output file is produced until the whole chain has passed. A non-zero exit at any link interrupts delivery — full stop, no workaround.

3.14 Environment prerequisites — session sanity checks

The session environment starts from scratch at every opening. The docx *npm* module, the engine that builds the document, conditions the whole chain. *adm-zip* now serves only the site generator, which assembles the publication archive: it left the document chain in General v1.83. Neither ships in the archive: package.json declares them with their minimum version, and they are always installed fresh with *npm* install — Common Structure §11.

The checks below run before anything else, at the start of the session. They are link zero of the §3.13 chain.

3.14.1 Sanity test

On a healthy environment, the docx module typically exposes between 270 and 310 keys depending on the minor version. A zero value indicates a known degraded state. *adm-zip* either resolves or it does not — there is no intermediate state.

```
node -e "console.log(Object.keys(require('docx')).length)"
# Sortie attendue sur environnement sain : nombre > 200
# Sortie pathologique : 0 (require résolu sur un module sans exports
CJS)

node -e "require('adm-zip'); console.log('adm-zip OK')"
# Sortie attendue : adm-zip OK
# Sortie pathologique : MODULE_NOT_FOUND
```

Decision criterion. If the docx key count exceeds 200, no action; *adm-zip* must also load before a site publication. A zero count signals a missing module or a broken installation: reinstall, never patch. A normal count guarantees nothing about the consistency of resolution — §3.14.2.

Note: *Case met in the 2026-08-20 session: adm-zip missing. The generator built the document without error, then failed on the call to injectCustomProps, after Packer.toBuffer() but before writing the file — so without corrupting anything. A generator that wrapped this call in a try/catch would, on the other hand, produce a valid file (.docx) that opens, but has no fingerprint, a defect that would only show when publishing the site. Since General v1.83 the fingerprint no longer depends on adm-zip, and this case can no longer occur at generation.*

3.14.2 Inconsistent resolution — the cause of the empty body

The docx package exposes two builds of the same code: dist/index.cjs and dist/index.umd.cjs. Node’s ordinary resolution picks the first; a require with a directory path picks the second. Both load without error, expose the same number of keys, and produce distinct classes: instanceof is false from one to the other.

A document whose paragraphs come from one *instance* and whose serialiser comes from the other comes out with a body full of rootKey tags. No error is thrown. The file opens; it is empty.

Note: *The rule that prevents this case is unique and fits in one sentence: every file in the same chain resolves docx the same way. Since General v1.80, that way is requireExterne — ordinary resolution, then KIT_NODE_MODULES, then npm root -g. No path is written in advance. General Reference §1.3.*

The remedy is therefore to align resolutions, never to modify the installed module. The procedure for patching the docx package.json, described here until 2026-09-01, is withdrawn: it treated a symptom, touched a file belonging to a third party, and did not survive a reinstallation.

3.14.3 Persistence and blind spot

Reinstalling docx during a session changes nothing in the resolution rule, which lives in the Kit's files and not in the module. The diagnosis in §3.14.1 remains useful after the fact: it detects the plain absence of a module, the case met on 2026-08-20 for *adm-zip*.

Blind spot. The key count does not see an inconsistent resolution: both instances expose as many. Only check 13, downstream, and the fidelity round trip, ahead of a campaign, catch it. It was the latter that established the cause.

3.15 kit_check_registry.py — Registry perimeter

- **What it checks.** The top-level keys of the project *Registry*, and those alone. A key missing from the Kit's list is an error: it will be lost at the next update, and its loss will not show.
- **Contrôle ajouté:** It also checks the slugs: a document belonging to an engine carries that engine's prefix — Naming convention §3.5 —, and a slug prefixed with an engine belongs to that engine. Exit 1 on a violation.
- **Usage.** *python* kit_check_registry.py, or --*registry* to name a file. Exit 1 if a key is outside the perimeter, 2 if the *Registry* cannot be found or read.

Note: *The file is loaded by Node, as the generators load it: a home-made parser would be fooled by a comment or a string containing a brace, and the check would bear on something other than what is actually read.*

Note: *What belongs to the project lives in [Préfixe] - Settings.js — Common Structure §5.4. The content of the Kit's blocks is checked elsewhere: kit_check_setters.py for requires, the site generator for deploy and rendering.*

3.16 kit_gen_document.js — rebuilding from a map

- **What it does.** Rebuilds a document from the map extracted from its binary by kit_extract_map.py. It knows no particular document: it is a *stable artifact*, not a session generator. It is part of the Kit's download archive, so that a project has everything it needs to work from the start. Like every *stable artifact*, it changes only when its logic changes — the rule of the generator written from scratch, *Kit Prompt* §4.1, targets *ad hoc generators* gen-*.js.
- **Usage.** node kit_gen_document.js carte.json “Nom sans TS” LANG titre sous-titre tagline --ts TS. The three style modules are declared through environment variables.

Note: *It neutralises the non-breaking spaces set by the pipeline passes before reinjecting the text: reinjected as they are, they would stop a glossary term or a highlighted fragment from being detected again, and the rendering would be lost without any check seeing it.*

4 Layer 3 — Periodic cross-cutting audits

Reviews done in a dedicated session. They deal with drift that cannot (yet) be detected at runtime or in session. The aim is often to identify a rule that deserves to move up into *Layer 1* or 2.

4.1 JSDoc and Reference audit

Confronts every exported function of every *stylesheet* with its line in the matching Reference. Detects divergences in return type, signatures, behaviour. Historical motivation: Bug #1 had revealed that the *JSDoc* and the Reference of makeTable said two different things. The rule is now prescribed in *Kit Prompt* §4.1 — the audit checks continued compliance, not an unknown drift.

- **Scope:** General, Glossary, YAML, HTML — all stylesheets together.
- **Deliverable:** a bump of each *stylesheet* couplet with *JSDoc* corrections and an aligned Reference.
- **Tracking:** Bug #8 in Todos.

4.2 Version alignment audit

Checks the consistency of version numbers across the following locations. Any inconsistency indicates a document not regenerated after a bump, or a partially updated file.

- **Registry:** source of truth for the active versions.
- **.js header:** the “Version: x.xx” comment in the first lines. It is the location that Update Guide §4.2 prescribes to the user for a manual check.

- **Exported constant:** STYLESHEET_VERSION, the value actually stamped into documents. A gap with the header is invisible at runtime but distorts any human check.
- **.docx fingerprint:** the StylesheetVersion *custom property* of each document produced.

This audit comes naturally at the end of a session of *stylesheet* changes. The gap between header and constant was found in the 2026-08-20 session on two bumped stylesheets — hence its explicit addition to the list.

4.3 Multilingual coverage audit

Scans the stylesheets for strings hard-coded in French that should go through the *L10N* pattern. Current state:

- **Glossary stylesheet:** *L10N* complete, alsoLabel the only key.
- **HTML stylesheet:** *L10N* complete. The compiledWithClaudeAI key is removed in favour of config.compiledWith, and the language selector keys are added.
- **General stylesheet:** *L10N* complete (noteLabel, noteLabelCol, sessionTitles). The width of the label column of note boxes joined the label in v1.68: a single value could not describe languages whose labels do not have the same set width. The compiledWithClaudeAI key was removed in v1.67 — no file (.docx) carries a compilation notice any more.
- **YAML stylesheet:** nothing notable.

5 Fingerprints and traceability

Every file of the Kit carries a *fingerprint* that makes it possible to identify it and verify its origin.

5.1 Custom properties in the docx

Injected through style.injectCustomProps(buf, TS). Fields:

- **StylesheetVersion:** version of the *stylesheet* used — read from style.STYLESHEET_VERSION.
- **GeneratedAt:** the document's TS *timestamp* (format AAAA-MM-JJ - HHhMM). Must be identical to the TS of the file name — a gap signals a generator passing two different values to titlePage and injectCustomProps.

5.2 Couplet timestamp convention

The stylesheet.js + Reference.docx pairs share exactly the same TS. A bump of one entails regenerating the other with an identical TS. Without this constraint, nothing tells which of the two is authoritative. Complete *inventory* of couplets: *Kit Prompt* §6.

- **Kit glossary couplet:** Kit - Glossary - Terms.js and the documents *Kit - Glossaire - Termes*, one per language.
- **Kit Cover Sheet couplet:** Kit - Documentation - Cover Sheet.docx and its PNGs, one per language.
- **Project Cover Sheet couplet:** [Préfixe] - Documentation - Cover Sheet.docx and its PNG — present if Registry.requires.coverSheet is true.

5.3 Registry as the single source

Kit - *Registry.js* holds the versions, TS and flags of all the Kit's files. Every version reference in another document must be traceable to the *Registry*. In case of divergence, the *Registry* prevails. The detailed structure of the Kit *Registry* is documented in *Pipeline HTML* §3.1; that of project Registries in Common Structure §5.

6 Catalogued anti-patterns

Frequent errors, identified through experience, to be prevented by *Layer 1* or *Layer 2* as early as possible.

Anti-pattern	Consequence	Defence
Paragraph level inconsistent with its parent heading	Badly indented rendering, visual inconsistency	Inexpressible since General v1.70 — the level comes from the heading, not from the function name. Checked downstream by kit_validate_docx.js check 11 (§3.1)
Note directly after a heading	Patronising effect — "too instructive"	*Layer 2* — kit_validate_docx.js check 17 (§3.1)
§0 as the starting number	Non-standard numbering, a LaTeX legacy	Explicit rule General §1.2, future static check
Two adjacent Table functions without releaseParagraph	The two tables merge visually in Word	Documented in General §13.4 — generator discipline
ImageRun without the type parameter	File word/media/.undefined, hence a corrupt .docx	Fixed by the mandatory style.makeImageRun
require('docx') without a global path	Conflict between local and global copies. Discrepancy found on a *stylesheet* imported from a third-party environment, 2026-08-20.	Rule: requireExterne — ordinary resolution, then KIT_NODE_MODULES, then *npm* root -g; no path written in advance. General Reference §1.3, §3.14.2.

Anti-pattern	Consequence	Defence
***prompt* for non-dialogue code**	Semantic confusion — *prompt* is reserved for the *chat*	Rule: yamlStyle.rawBlock for all code (JS, bash, SVG, HTML, JSON)
disclaimer() as a final style	Patronising tone, ethically deprecated	Removed — style.tip() instead
Direct editing of the docx XML (ElementTree)	Namespace corruption, unreadable docx	Rule: rewrite from scratch as a **Node.js** generator
Missing *custom properties*	gen-kit-site.js blocks the document	Runtime — injectCustomProps is mandatory
Generic JSON extractor without semantic classification	Every 1×1 table rendered as *prompt*, code blocks miscategorised	Rule: each 1×1 block classified by hand — rawBlock (code), codeBlock (YAML) or *prompt* (dialogue only)
Equal widths kept on very unequal columns	Wasted space for numeric or short-symbol columns, main text squeezed	Rule: equal distribution by default suits comparable columns, and the # column is narrow automatically — General §9. Otherwise pass colWidths by content — symbols ≈ 5-10 %, long text ≈ 30-40 %.
Regex when writing a structured file	A regex does not know the grammar of the format — it can hit a protected case and corrupt the file	Rule Kit *Prompt* §4.1: every change to a structured file goes through the native parser. Regex allowed for pure reading only.
Inline changelog in Kit-published documents	Stray logs that clutter the reading of a reference document	Editorial rule: no “Document history” section / no inline changelog. Sole exception: Kit - Projet - Todos.
ASCII separators of variable length	Inconsistent rendering, poorer readability	Rule Kit *Prompt* §1 checklist: separators unified at 30 characters, = major, - minor.
Footer wrapped in new Footer({ children: [...] })	docx-js serialises the child Footer as “<options/>” under <w:ftr>. Word refuses to open the file (Text Recovery Converter).	*Layer 1* — explicit *JSDoc* on makeFooter / makeCoverFooter. *Layer 2* — kit_validate_docx.js check 7.

Anti-pattern	Consequence	Defence
Silent synchronisation between *Cover Sheet* and HTML *landing page* on the assumption of drift	Silent loss or addition: the two indexes are independent — a *binder* may be lighter, the *landing page* leaves out what is not read online.	The two indexes are declared separately (*Pipeline* HTML §2.5). No script reads one to drive the other.
Hard-coded Kit-specific values in a *stable artifact*	Prevents use by *consumer projects*	Read specific data from the canonical source: .docx via the *pandoc AST*, project *Registry*, .js data files.
require('docx') returns an empty object (corrupt *npm* cache)	Immediate TypeError on every new Document(), Packer.toBuffer()	Environment prerequisites — §3.14
require('docx') degraded ESM view — docx silently empty	<rootKey>w:p</rootKey> instead of real *OOXML* elements — Word shows an empty document. Sanity §3.14.1 PASS.	*Layer 2* — kit_validate_docx.js check 13; prevention by aligning resolutions, §3.14.2.
Stray <0/> injected by a forgotten *spread* on makeTable	Invalid XML tag. Word may tolerate it, but *pandoc* HTML fails silently on the broken table.	*Layer 2* — kit_validate_docx.js check 14 (strict global XML). Blocks delivery.
Numbering broken by h2(level, text) instead of h2(number, text)	Every §2.X displays as “2”. Valid document but inconsistent numbering.	*Layer 2* — kit_validate_docx.js check 15 (heading monotony). Blocks delivery.
X.1 without X.2 — orphaned subsection	*Anti-pattern* documented in General §1.2. A lone subsection whose content belongs under the parent.	*Layer 2* — kit_validate_docx.js check 16 (orphaned X.1). Blocks delivery.
Isolated note or tip immediately after a heading	*Anti-pattern* documented in General §1.2. Tip or note used as the main content instead of underlining a preceding paragraph.	*Layer 2* — kit_validate_docx.js check 17 (isolated note or tip). Blocks delivery.
***Pipeline setter* omitted at the top of gen-{prefix}-site.js**	Silent regression — HTML site without glossary hotlinks, without small-caps brands, without family and external routing. The generator runs without error.	*Layer 2* — kit_check_setters.py (§3.4). *Pipeline* HTML §7.9 documents the rule. *Linter* exit 1 if a required *setter* is missing.

Anti-pattern	Consequence	Defence
First references without <w:titlePg/> in the sectPr	Word ignores headers.first and footers.first: the *cover page* receives the navy header and the default footer. Silent defect, invisible at *generation*.	*Layer 2* — kit_validate_docx.js check 18. Correct pattern: properties: { ...style.pageProps, titlePage: true } — General Reference §10.4.
Reusing the helpers of gen-kit-site.js in a *consumer project*	gen-kit-site.js is Kit-only under Naming Convention §3.6; its helpers are coupled to the Kit SECTIONS constant. Reuse = implicit duplication or a crash on registry.stylesheets, absent from the project *Registry*.	*Layer 1* — module.exports of gen-kit-site.js limited to { main } since 2026-05-02. *Consumer projects* write their own gen-{prefix}-site.js from scratch (*Pipeline* HTML §2.5).
Comparing XML tags by prefix	“w:pPr” answers to “w:p” and inflates the depth, which “/w:pPr” does not decrement. The element swallows the rest of the body and the check goes inert without reporting anything.	Rule: require the tag name to end with an angle bracket, a space or a slash. Helpers isTagStart and findTagStart of kit_validate_docx.js.
Exit zero from a check that verified nothing	A *linter* that finds nothing to work on returns zero, which reads as a passed check. Found on the level *linter* before its removal, whose five map-driven generators triggered no check.	Rule: distinguish “nothing to check” from “check passed”. A tool that found nothing to work on must say so and exit with an error, never return zero
Brand taken for a bold lead	The *pipeline* renders brands in bold small caps. A paragraph opening on a brand name therefore has a bold first run: extracted as boldText, it loses its small caps, boldText being excluded from the *pipeline*.	Rule: a run that is bold AND in small caps is a brand, never a lead. Detected by kit_check_markup.py (§3.7).

Anti-pattern	Consequence	Defence
Glossary non-breaking spaces neutralised from the wrong language	Text extracted from a binary carries the non-breaking spaces set by the glossary pass. Neutralising them from another language's surfaces leaves the term unmatched: it loses its colour without a single word changing.	Rule: read the surfaces through buildSearchTerms(document language). Detected by kit_check_markup.py (§3.7).

7 Feedback loop from project to Kit

The §1.1 layers are the Kit's internal defences — runtime prevention, session *validators*, cross-cutting audits. Another source of signal exists: feedback from *consumer projects* in real use. It is through this channel that the following corrections historically arrived: the Sorso brief (sectionBanner Glossary v1.18, gen-kit-site HTML *setters* — Todos entry #34), the Sliver brief (degraded ESM require — check 13 added, Todos entry #27), the Immobilier brief (narrow note cell — kit_patch_helpers and normative XML schema *General* §6/§7, Todos entries #25 and #26), the *AI* News brief (*i18n* markers of the *Cover Sheet renderer* — v4, Todos entry #31). None of these bugs would have been caught by Layers 1-3 — they emerge from concrete use outside the Kit's perimeter.

From 2026-05-11, this informal channel is formalised through a dedicated project file: [Préfixe] - Project - *Kit bug report* (TS).docx. Standard naming pattern Naming Convention §2.1, Project category open to both prefixes, §7. The *consumer project* produces the *bug report* from scratch with gen-kit-bug-report.js, validates it with kit_validate_docx, and delivers it by dropping the binary into the Kit project's *chat* along with the necessary artifacts (the offending gen-*.js, the faulty result (.docx), screenshots if any).

The complete specification — triggers, naming, document structure, reporting workflow, non-permanence of the file — is in Common Structure §13. This §7 does not duplicate the contract: it points to the project's source of truth and marks the transition from informal to structured channel. The historical briefs remain valid as references in Todos §4 (entries #25, #26, #27, #31, #34).