Kit de documentation

Documentation — Quality Control — FR

1 Introduction

La production d'un document passe par des couches de défense successives. La prévention: les stylesheets rendent certaines erreurs impossibles à écrire. La détection: des *validateurs* examinent le fichier produit avant qu'il soit livré. L'audit: une vérification périodique de ce que la prévention et la détection ne voient pas.

Aucune de ces couches ne suffit seule. Une erreur qu'un *stylesheet* ne peut pas empêcher doit être attrapée en aval; une erreur qu'aucun outil ne sait détecter reste du ressort d'un audit. Les sections qui suivent décrivent chaque couche, les artefacts qui la mettent en œuvre, et l'ordre dans lequel ils s'appellent.

Destiné à **Claude AI** comme référence unique pour toutes les questions d'audit, validation et qualité. Tout renvoi "Audit" ou *Quality Control* dans les autres documents du Kit pointe vers cette source.

1.1 Les couches de défense

Chaque document ou fichier généré passe potentiellement par les niveaux de contrôle suivants, de l'amont vers l'aval:

- **Couche 1 — Prévention runtime:** contrôles intégrés aux fonctions des stylesheets. Bloquent les erreurs avant même que le document soit produit. Exemples: makeImageRun refuse un format invalide, makeHeader refuse un paramètre vide.
- **Couche 2 — Détection session:** scripts externes appelés autour de chaque *génération*. Deux temps distincts. En amont, kit_check_setters.py inspecte le source du générateur avant exécution. En aval, les *validateurs* inspectent le fichier produit avant livraison — kit_validate_docx.js et kit_validate_xlsx.py. S'y ajoutent les prérequis d'environnement, vérifiés avant tout le reste (§3.14).
- **Couche 3 — Audit périodique:** revues transversales faites en session dédiée. Détectent les dérives progressives. Exemples: audit *JSDoc* et Reference, audit version *alignment*, audit *L10N* coverage.

1.2 Principe — fail loud early

Une erreur détectée tard coûte proportionnellement plus. Une erreur silencieuse ne se corrige jamais. Chaque fois qu'une validation peut être déplacée plus tôt dans le *pipeline* (runtime plutôt que session-only, session-only plutôt qu'audit périodique), elle doit l'être.

> 💡 *Si un contrôle peut devenir runtime, c'est toujours mieux. Si pas possible runtime, le mettre en session-only. L'audit périodique est le dernier filet de sécurité, pas la méthode de travail par défaut.*

2 Couche 1 — Contrôles runtime dans les stylesheets

Les contrôles runtime vivent dans les fonctions exportées des stylesheets. Ils lancent une exception *JavaScript* explicite à l'appel incorrect — le générateur échoue immédiatement avec un message identifiant la ligne coupable.

2.1 injectCustomProps — empreinte obligatoire

Stamp les propriétés custom StylesheetVersion et GeneratedAt dans les docx. Appel obligatoire après Packer.toBuffer() dans tout script générateur.

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

Documents sans ces *custom properties* sont rejetés par gen-kit-site.js lors de la conversion HTML.

2.2 makeImageRun — validation dimensions et format

Force la présence du paramètre type ImageRun et valide width/height > 0, format ∈ {png, jpg, gif, bmp, svg}. Lance une erreur explicite au lieu de produire un .docx corrompu silencieusement.

2.3 makeHeader — validation des paramètres

makeHeader(title, subtitle) lève une erreur explicite si title ou subtitle est vide, undefined ou non-string. Aligné sur le principe fail-loud early — chaque générateur doit passer les deux. Voir *General Reference* §10.8.

2.4 Discipline de niveau — le niveau vient du titre

Depuis General v1.70 et HTML v1.43, le niveau de section ne vit plus dans le nom des fonctions. h1, h2 et h3 posent un niveau courant au niveau du module, et toutes les autres familles le lisent: écrire paragraph('texte') sous un h2 produit un paragraphe de niveau 2 sans que le générateur ait à le savoir. L'erreur de concordance entre un suffixe et son titre parent n'est plus une erreur à surveiller, elle est devenue inexprimable.

La validation statique du source qui occupait cette place a donc perdu son objet, et son outil a été retiré du Kit. Le contrôle des niveaux subsiste et il est descendu d'un cran: le check 11 de kit_validate_docx.js lit l'indentation réelle de chaque paragraphe du document produit, sans rien présumer de la façon dont le générateur est écrit. Il est bloquant, et c'est désormais l'unique autorité sur ce point.

2.5 Règles de décision embarquées

Documentées dans les Reference de chaque *stylesheet*, pas enforceables runtime mais clairement explicites:

- **Spread obligatoire sur fonctions retournant Array:** *General Reference* §13.
- **Transitions entre fonctions-Table:** releaseParagraph() obligatoire entre deux blocs-Table adjacents — *General Reference* §13.4.
- **Numérotation:** démarre à §1, jamais §0 — *General Reference* §1.2.
- **prompt:** réservé au dialogue à copier-coller. Pour tout code, yamlStyle.rawBlock ou codeBlock — *General Reference* §8.

3 Couche 2 — Artefacts Kit stables et prérequis de session

Scripts stables qui vivent dans l'arbre du Kit, repris dans chaque projet qui en a besoin. Préfixe kit_ pour signaler leur origine — nomenclature Unix snake_case, exception formelle au patron de *Convention de nommage* §2, nommée en *Convention de nommage* §3.4. Les rôles: *validateurs* (retournent 0 si OK, non-zéro sinon, bloquent la livraison en cas d'échec), renderers (produisent un *artefact* déterministe — même entrée et même *contexte Registry* donnent la même sortie), et helpers (fabriquent des fragments réutilisables sans produire de livrable à eux seuls).

Distinction importante avec les *générateurs ad hoc* (gen-*.js): les artefacts stables ont une logique stable, réutilisable session après session. Ils ne sont PAS régénérés à chaque session — seulement quand leur logique évolue. L'évolution est tracée dans un bloc changelog en tête du fichier lui-même (convention Unix) au même titre que les stylesheets Kit, pas dans un document externe.

Deux règles y portent sur toute la chaîne: §3.13 fixe l'ordre d'appel obligatoire, §3.14 documente les prérequis d'environnement à vérifier avant que quoi que ce soit ne s'exécute.

3.1 kit_validate_docx.js — validateur docx produits par le Kit

Vérifie les docx produits par les stylesheets **Node.js** du Kit. *Artefact* Kit conservé — script **Node.js** autonome sans dépendance *npm* externe (utilise zlib stdlib pour lire le ZIP docx, plus un mini-parseur XML stdlib pour les contrôles structurels *OOXML*). Cette section est la source de vérité unique de son contrat: les autres documents y renvoient sans le redire.

- **Check empreinte:** custom.xml présent avec StylesheetVersion et GeneratedAt.
- **Check version:** la StylesheetVersion stampée correspond à la version passée en --version. Absente ou inférieure à cette version: signalée. Le site, lui, ne refuse un document que sans *empreinte* ou sous le plancher minDocumentVersion du *Registry* — *Pipeline HTML* §7.4.
- **Check cohérence:** le nombre de <w:tbl> dans document.xml correspond au nombre de Table blocs attendus depuis l'*AST* de *génération*.
- **Check paragraphes en tables:** si des tables sont présentes, le compte de paragraphes internes doit être supérieur à zéro. Zéro avec des tables présentes signale une perte garantie — piège doc.paragraphs, *Pipeline HTML* §7.1.
- **Check corpus non vide:** un document avec 0 paragraphes ET 0 tables alors que les *custom properties* sont présentes indique presque toujours une corruption silencieuse de docx-js — pattern <rootKey>w:p</rootKey>. Cause établie: deux instances distinctes du module docx dans une même chaîne. Le check 13 est le filet de sécurité post-génération; la cause et sa prévention sont au §3.14.

- **Check extensions word/media/:** chaque extension présente dans word/media/ doit être déclarée comme Default Extension dans [Content_Types].xml. Une extension orpheline — typiquement.undefined produite par un ImageRun sans paramètre type — rend le fichier illisible par Word sans message d'erreur explicite. Voir *General Reference* §13.3.
- **Check ordre canonique <w:rPr>/<w:pPr>:** enfants directs vérifiés contre l'ordre *OOXML* §17.3.1 et §17.3.2. Détecte le bug docx@8.5.0 où rFonts est sérialisé en dernier au lieu de la 2e position. Word 365 applique strictement le schéma XSD, *LibreOffice* tolère.
- **Check conteneurs de table non-vides:** <w:tr> sans aucune <w:tc> directe ou <w:tbl> sans aucune <w:tr> directe. Word 365 refuse à l'ouverture, *LibreOffice* tolère silencieusement.
- **Check namespaces enfants <w:r>:** liste blanche restreinte (w:, mc:, w14:, w15:, w16se:, w16cid:, w16:). Capture les bugs lib docx silencieux comme <type>tab</type> sérialisé par docx@8.5.0 sur new TextRun({ children: [{ type: 'tab' }] }).
- **Check cohérence niveau heading et indentation:** pour chaque paragraphe top-level du body, l'attribut <w:ind w:left="X"/> doit être dans l'ensemble valide pour le niveau du Heading le plus récent. *Modèle* issu des helpers du General *stylesheet*: L1 = {567, 967, 1157}, L2 = {1350, 1750, 1940}, L3 = {2100, 2500, 2690}. Un document de glossaire, reconnu à son nom — Glossaire - Termes, Glossary - Terms, *Stylesheet* - Glossary - Reference —, accepte en plus 1350 et 1550 sous un titre de niveau 1: le nom de terme et la définition de la feuille Glossary. Ailleurs, ces valeurs restent une anomalie.
- **Check warnings typographiques NBSP:** détecte les séquences numériques groupées par tranches de 3 suivies d'une unité monétaire ou typographique connue (€ $ £ ¥ % m² m³ km kg ha ares hectares °C) où les espaces internes sont ordinaires (U+0020) au lieu de *NBSP* (U+00A0). Warnings non bloquants. Helper Kit recommandé: style.formatCurrency(amount, opts).
- **Check 14 — XML strict global:** pour chaque entrée.xml ou.rels du zip, vérifie l'absence de tags au nom invalide (ex. <0/> parasite injecté par oubli du *spread* sur makeTable) et l'équilibre des ouvertures/fermetures via le mini-parseur tokenizeTags.
- **Check 15 — Monotonie numérotation headings:** extrait la séquence des Heading1/2/3 + leur numéro Kit, vérifie que chaque compteur progresse par +1 par parent et démarre à.1. Capture le bug h2(level, text) au lieu de h2(number, text) qui produit un .docx valide mais avec la numérotation cassée.

- **Check 16 — X.1 orphelin:** *anti-pattern* documenté Kit Projet *Prompt* §4.1 + *General Reference* §1.2 — un parent qui n'a qu'un seul sous-titre est suspect, son contenu doit remonter sous le parent. Group by parent, flag les solo.
- **Check 17 — Note/tip isolée après heading:** *anti-pattern* documenté *General* §1.2 — jamais de tip/note après un heading sans paragraphe-mère. Détection par *empreinte* de largeur de la première colonne du <w:tbl> qui suit immédiatement un heading top-level: TIP_BULB_COL_WIDTH pour un conseil, et pour une note l'appartenance à NOTE_LABEL_COL_WIDTHS, la largeur d'étiquette n'étant plus unique depuis General v1.68. Ce check est resté inerte de son ajout au 22 août: son découpage des éléments de premier niveau comparait par préfixe, si bien que la *balise* de propriétés de paragraphe répondait à celle du paragraphe et gonflait la profondeur. Le premier paragraphe du corps avalait tout le reste et la boucle ne voyait plus jamais un titre suivi d'une table.
- **Check 18 — Page de garde:** cohérence entre le drapeau <w:titlePg/> du sectPr et les références d'en-tête ou de pied de page de type first. Sans le drapeau, Word ignore ces références et applique l'en-tête et le pied de page par défaut dès la page 1 — la page de garde perd sa nudité. Cause racine: style.pageProps ne porte pas le drapeau, le patron documenté est properties: {...style.pageProps, titlePage: true } — *General Reference* §10.4. Check symétrique: le drapeau sans référence first est également signalé.
- **Check 19 — Identification du format:** première entrée de l'archive. Packer place le dossier word/ en tête; toute réécriture par une bibliothèque qui ne conserve pas l'ordre le rejette après les métadonnées, et le document est alors servi comme une archive générique — renommé en .zip au téléchargement. Le document reste valide et s'ouvre sans avertissement: ce check est le seul endroit de la chaîne qui voie le défaut avant le destinataire.

Usage: node kit_validate_docx.js 'Mon Document (TS).docx' pour une validation simple, ou node kit_validate_docx.js 'Mon Document (TS).docx' --version 1.67 pour contrôler en plus l'*empreinte*. Exit code 0 = valide, 1 = anomalie.

Note: *Le validateur s'exécute après chaque génération de .docx, avant la copie du fichier vers le dossier de sortie et avant son inclusion dans un ZIP de déploiement. Un exit non-zéro bloque la livraison — jamais de contournement.*

3.2 kit_validate_xlsx.py — validateur xlsx

Vérifie les xlsx produits par les pipelines qui en créent — projet Trading principalement. Les checks ci-dessous constituent son contrat complet.

- **Checks:** structure zip, balance formatCode, références cellule, références feuille dans formules, double-lecture openpyxl.
- **Usage:** *python* kit_validate_xlsx.py document.xlsx [--strict].
- **Artefact conservé:** vit dans l'arbre du Kit et de chaque projet qui produit des xlsx. Non régénéré à chaque session.

3.3 kit_render_cover_sheet.py — renderer de la page de couverture

Renderer déterministe qui produit le PNG A4 300 dpi du *Cover Sheet* (Kit ou *projet consommateur*) à partir du couplet.docx *Cover Sheet* correspondant. Toutes les valeurs spécifiques (HEADER_TEXT, rubriques §2, *PCL* surchargeables comme TAB_NUMBERED_MAX) sont lues du .docx via parsing *pandoc AST*. Aucune valeur Kit-spécifique n'est hardcodée dans le code du *renderer* — conforme au contrat documenté *Cover Sheet* §3.

CLI: arguments optionnels combinables. Mode rétrocompat préservé — invocation sans argument résout automatiquement le Kit *Cover Sheet* courant via Registry.documents['cover-sheet'].ts.

- **--cover-sheet PATH:**.docx *Cover Sheet* à rendre. Si absent, résolution automatique du Kit *Cover Sheet* courant via *Registry*.
- **--registry PATH:** *Registry* projet (défaut./Kit - *Registry.js*). Source de project.docLanguage et rendering.coverSheet.
- **--output PATH:** PNG de sortie. Si absent, dérivé du .docx source.
- **--language XX:** override langue (FR | EN | DE | LU). Pilote les chaînes localisées du Kit (subtitle, quote, toc_title).
- **Marqueurs de rubrique numérotée localisés:** FR='numérot', EN='numbered', DE='nummeriert', LU='nummeréiert'. Aligné sur la sémantique du marker FR (racine plus longue que la forme courte du label) — évite le faux positif documenté en session *AI* News où la sous-chaîne 'number' matchait 'no number' dans les cellules Tab des lignes réservées d'un *Cover Sheet* projet EN. v3 introduisait l'*i18n* initiale; v4 durcit les markers EN/DE/LU.
- **Accès défensif au Registry projet:** la lecture de rendering.coverSheet.sectionHMode utilise un fallback gracieux 'FIXED' si la section rendering est absente. Sans ce fallback, KeyError immédiat sur *Registry* projet conforme au squelette *Structure commune* §5.2 historique. Le squelette §5.2 est élargi en parallèle pour documenter la section — défense en profondeur.
- **TAB_NUMBERED_MAX auto-dérivé:** la valeur est calculée à partir du nombre de rubriques extraites de §2 et prime sur le défaut Kit. Le *PCL* §3 TAB_NUMBERED_MAX devient optionnel — utile uniquement pour réserver explicitement des onglets supplémentaires.

Appelé à chaque modification *PCL* d'un *Cover Sheet*. Script *Python* autonome — dépendances *Pillow*, zoneinfo, *pandoc*. Règle *couplet*: le PNG produit porte le même TS que le .docx *Cover Sheet* source.

3.4 kit_check_setters.py — linter discipline des setters

Seul *linter* statique de la *Couche 2* amont. Vérifie qu'un script générateur appelle bien tous les *setters* obligatoires en tête de fichier, selon les flags de Registry.requires. Cible primaire: les générateurs HTML gen-{prefix}-site.js — leur omission de setGlossaryTerms produit silencieusement un site sans hotlinks glossaire.

- **Règle:** chaque flag à true dans Registry.requires impose un appel au *setter* correspondant (glossary appelle setGlossaryTerms, documentTitles setDocumentTitles, brands setBrands, variableNames setVariableNames, hassEntities setHassEntities, textHighlights setTextHighlights). S'y ajoutent setLanguage, toujours, puis selon la cible: setDocumentAuthor et setDocumentSiteBase pour un générateur de .docx; setWebAuthor, setFamilyDomains, et setGlossaryHref quand le projet publie plusieurs glossaires, pour un générateur HTML.
- **Usage:** *python* kit_check_setters.py gen-document.js [--*registry* path] [--html] [--strict]. Le flag --html applique les exigences d'un générateur HTML. Le flag --strict promeut setLanguage de warning à error.
- **Exit codes:** 0 = tous les *setters* requis appelés; 1 = violations; 2 = *Registry.js* introuvable.
- **Couplé avec:** *Pipeline HTML* §7.7, qui documente la liste des *setters* obligatoires côté générateur de site. Le *linter* automatise ce que la doc prescrit.
- **Artefact conservé:** vit dans l'arbre du Kit et de chaque projet qui produit un site HTML. Non régénéré à chaque session.

Note: *Issu du brief Kit - Adaptations souhaitables (item 4) — promotion de la documentation des setters obligatoires en linter automatique.*

Note: *Détection statique conservatrice. Le linter fait du regex matching sur le texte source du .js, sans suivi du flow conditionnel. Conséquence: un setter appelé sous if (Registry.requires.X) {...} avec X=false dans le Registry est rapporté comme "Setter détecté" même si l'appel est dead code à l'exécution. Ce comportement est volontairement imprécis dans le sens conservateur — il accepte trop, refuse trop peu — donc jamais bloquant pour une livraison correcte. Faux positif catalogué en session AI News; correction non prévue (coût d'un parseur JS à flux disproportionné par rapport au bruit).*

3.5 kit_gen_glossaire_html.js — helper générateur glossaire HTML

Helper Kit stable qui génère la page glossaire.html à partir de [Préfixe] - Glossary - Terms.js. Contourne intégralement le *pipeline AST pandoc* — la

structure propriétaire du .docx glossaire (sectionBanner, letterHeader, alias inline) n'est pas lisible par le convertisseur *AST*. Contrat normatif complet dans *HTML Reference* §14.8 et §15.

- **Règle absolue:** le glossaire HTML est TOUJOURS généré par ce helper, JAMAIS réimplémenté dans gen-kit-site.js ni dans les gen-[projet]-site.js des *projets consommateurs*. Centralisation stricte pour éviter les divergences silencieuses.
- **Usage:** const { generateGlossaireHtml } = require('./kit_gen_glossaire_html.js'); generateGlossaireHtml({ termsPath, outputPath, htmlStyle, config }).
- **Artefact conservé:** vit dans l'arbre du Kit et de chaque *projet consommateur* qui publie un glossaire.
- **Dépendances:** aucune *npm* externe. Lit Terms.js via require, utilise les classes.gloss-* du *stylesheet* HTML, le titre et breadcrumb localisés via htmlStyle.getStrings().glossaryTitle (ajout HTML v1.23).

3.6 kit_patch_helpers.py — helpers Python pour patches chirurgicaux .docx

Helper *Python* qui fabrique des fragments XML conformes au *stylesheet* General lors de modifications chirurgicales d'un .docx existant. Couvre les structures complexes (note 1×2, etc.) qui sont source de bugs lorsqu'elles sont produites par clonage de template ou par construction manuelle. Fragments construits FROM SCRATCH depuis les constantes Kit synchronisées — pas par clonage d'un template du document cible.

- **Règle absolue:** pour insérer un bloc note/tip/table dans un .docx existant via *patch chirurgical* (*Prompts de dialogue* §2.4), utiliser obligatoirement les helpers de ce module — jamais cloner manuellement un fragment du document cible.
- **API actuelle:** build_note_block(content, level=1, lang='FR') — retourne un fragment XML <w:tbl>...</w:tbl> représentant une note Kit niveau 1, 2 ou 3.
- **Synchronisation:** constantes Kit hardcoded explicitement synchronisées avec le General Code à chaque évolution. Dépendance documentée — pas une drift silencieuse.
- **Dépendances:** aucune externe. Stdlib *Python* uniquement. Aucune regex en écriture sur fichier structuré.
- **Artefact conservé:** vit dans l'arbre du Kit. Recopié tel quel lors de l'initialisation ou de la mise à jour d'un projet qui en a besoin.

3.7 kit_check_markup.py — contrôle de marquage et d’accord des constantes

Deux contrôles qu'aucun outil du Kit ne faisait, tous deux nés de défauts passés au travers des chaînes existantes. Un diff de texte ne voit pas une perte de couleur: un document régénéré peut porter exactement les mêmes mots, dans le même ordre, et avoir perdu le balisage de plusieurs termes ou de plusieurs marques.

- **Contrôle de marquage:** compte les runs porteurs de chaque marquage — glossaire, marques, noms de variables, entités, gras, hyperliens, images — dans le fichier source et dans le document produit, puis compare. Toute perte est signalée. Le mode --strict signale aussi les gains, pour une régénération qui ne doit rien changer au contenu; --added N déclare le nombre d'étiquettes grasses attendues.
- **Contrôle des largeurs d’étiquette de note:** la valeur vit en trois exemplaires — table localizedStrings de General, NOTE_LABEL_COL_WIDTHS du *validateur*, NOTE_LABEL_COL_BY_LANG des helpers de patch. La duplication est assumée, les deux outils ne pouvant pas importer le *stylesheet*. Le contrôle lit les trois sources à l'exécution et vérifie leur accord.
- **Usage:** *python* kit_check_markup.py SOURCE.docx PRODUIT.docx [--strict] [--added N], ou *python* kit_check_markup.py --widths [--dir DOSSIER]. Exit 0 si accord, 1 si écart, 2 en erreur d'usage.
- **Artefact conservé:** vit dans l’arbre du Kit et de tout projet qui régénère des documents existants.

Note: *Ce contrôle est le seul à voir une perte de marquage. Deux défauts réels lui doivent d'avoir été rattrapés avant livraison: onze termes anglais dépouillés de leur couleur parce que les insécables du texte extrait avaient été neutralisés d'après les surfaces françaises, et cinq marques dépouillées de leurs petites capitales parce qu'un paragraphe ouvrant sur un nom de marque avait été pris pour un paragraphe à introduction grasse. Aucun mot n'avait changé dans ni l'un ni l'autre.*

3.8 kit_check_ossature.py — contrôle d’ossature entre variantes de langue

Compare le squelette d'une traduction à celui de sa source. Le §17 du *Prompt* demande une traduction fidèle au sens, dans un texte qui se lit comme s'il avait été écrit dans sa langue: compter les mots ou comparer phrase à phrase reviendrait à sanctionner exactement ce que la règle demande.

- **Ce qui est mesuré:** la séquence des blocs, leur type et leur niveau, la numérotation des sections, les dimensions de chaque tableau, le nombre d'items de chaque liste.

- **Ce qui ne l'est pas:** le nombre de mots, la longueur des phrases, la correspondance phrase à phrase. L'allemand compose, l'anglais raccourcit, et une traduction qui calquerait la longueur du français serait mauvaise.
- **Ce qui est signalé:** une section, un item, une ligne de tableau ou un encadré présent d'un côté et absent de l'autre. C'est une information perdue ou ajoutée, ce que le §17 interdit.
- **Tolérance:** la note de traduction du §1, que la variante de référence ne porte pas, est reconnue et acceptée. L'option --sans-tolerance la refuse.
- **Usage:** *python* kit_check_ossature.py REFERENCE.docx VARIANTE.docx [AUTRES…]. Exit 0 si les ossatures concordent, 1 sinon.

3.9 kit_extract_map.py — carte de structure d'un document existant

Toute régénération part d'une carte extraite du fichier source. L'outil lit le XML du document et en rend la structure: ordre des blocs, type, niveau, contenu, marquage. Il est en lecture seule et n'emploie aucune expression régulière.

- **Pourquoi il est stable alors que les générateurs ne le sont pas:** le *Prompt* §4.1 impose d'écrire chaque générateur from scratch, parce qu'un générateur est propre à un document. L'extraction ne partage pas cette propriété: elle fait toujours la même chose, et la réécrire à chaque séance ne produit que des variantes d'un même code, chacune avec ses angles morts propres.
- **Ce qu'il reconnaît:** titres, paragraphes et leurs variantes par indentation, items de liste et continuations, encadrés note, tip et *prompt*, blocs de code brut et coloré, tables ordinaires, metaTable du *stylesheet* YAML, termName et definition du *stylesheet* Glossary, images, hyperliens, paragraphes de pont et de libération. Une image, seule ou en cellule, est rangée en fichier dans le dossier <carte>-images et replacée à la régénération; l'ampoule d'un encadré n'est pas relevée, la *feuille de style* la redessine. Une table note si sa première ligne est un en-tête, et se reconstruit sans en-tête sinon; une table d'une colonne est une table ordinaire. Le lien d'une cellule est relevé avec son *ancre* et sa cible, résolue par les relations du document, et reposé par hyperlinkCell.
- **Ce qu'il ne peut pas promettre:** le répertoire est fini. Une construction qui n'y figure pas est dégradée sans que rien dans le fichier produit ne le dise. C'est la raison d'être du §3.10.
- **Usage:** *python* kit_extract_map.py SOURCE.docx CIBLE.json.

3.10 kit_check_fidelite.py — contrôle de fidélité de l'extraction

Extrait, régénère, et compare le XML du corps à l'octet près. Un document qui ne revient pas identique signale une construction que l'extraction ne sait pas rendre. Le 23 août 2026, quatre défauts de ce type ont été trouvés en une journée, tous silencieux et tous invisibles à un diff textuel; ce contrôle les aurait trouvés d'un seul coup.

- **Les neutralisations sans lesquelles le test mesure autre chose:** l'*horodatage* et les identifiants de relation des hyperliens changent à chaque *génération* sans que le rendu bouge. La langue vient du suffixe du nom de fichier — régénérer un document allemand avec les surfaces françaises du glossaire produit un écart qui ne dit rien de l'extraction. Et la chaîne rejouée doit être celle de production, neutralisation des marques du *pipeline* comprise.
- **Ce qu'il ne promet pas:** il ne voit que les constructions présentes dans le parc. Un document employant demain un helper jamais utilisé passerait au travers. Le contrôle réduit la surface d'erreur sans la fermer: ce qui est tenu pour sûr l'est au regard de ce qu'on sait aujourd'hui, et c'est la condition ordinaire de tout contrôle qualité. Le jour où un cas non prévu se présente, l'extraction se complète et le parc se recontrôle.
- **Portée:** un document produit par une version de *stylesheet* antérieure diffère légitimement. L'outil le signale à part plutôt que de le compter comme un échec.
- **Usage:** *python* kit_check_fidelite.py SOURCE.docx [...] --version {version active}. Exit 0 si tout revient identique, 1 sinon.
- **Un document inchangé garde son horodatage.** L'*horodatage* frais est exigé à chaque LIVRAISON — *Kit Prompt* §4.1 — pas à chaque passage de générateur. Une campagne de régénération compare le corps produit à celui de la source, par la même neutralisation qu'ici; identique, elle jette le produit et conserve la source. Sans cette comparaison, le delta de publication traite tout le parc comme modifié, l'*inventaire* local se réécrit en entier, et l'historique du site perd son sens: quarante-six documents y portent la même minute.
- **Le glossaire est hors contrôle.** Il ne se régénère pas par extraction — §3.11 — et l'aller-retour testé ici n'a donc pas de sens pour lui. Il est compté à part dans la synthèse, jamais en écart. La Reference de la feuille glossaire suit la même voie, depuis le 2026-09-17: elle porte des éléments rendus par cette feuille — bannière, lettre, terme, définition — que la carte ne sait pas redire, et se régénère par son générateur de séance.

Note: *Les artefacts appellent kit_extract_map.py par chemin absolu, calculé depuis leur propre*

emplacement. Un chemin relatif les liait au répertoire de travail: kit_check_ossature.py invoquait de surcroît un nom d'avant la promotion en artefact stable, extract_map.py, et échouait à chaque appel sans que rien ne le signale — l'échec ne se voyait qu'à la première comparaison de variantes.

3.11 kit_gen_glossaire_docx.js — helper générateur glossaire .docx

Produit le .docx du glossaire à partir de [Préfixe] - Glossary - Terms.js. Pendant de kit_gen_glossaire_html.js — §3.5: une source, deux rendus.

- **Le glossaire .docx ne se régénère jamais par extraction.** C'est le seul document du Kit dont le contenu vit dans un module. Régénéré depuis sa carte lue dans le fichier source, il se fige: trois termes ajoutés à Terms.js le 24 août sont restés absents des trois variantes jusqu'à un contrôle à la main.
- **Appelle les setters du pipeline.** Sans eux, les termes de glossaire et les marques restent en noir dans les définitions.
- **Hors contrôle d'ossature.** Les termes se trient dans la langue rendue, donc les regroupements par lettre diffèrent d'une variante à l'autre. kit_check_ossature.py l'exclut explicitement — §3.8.
- **Les deux rendus n'ont pas la même structure.** Le papier suit les rubriques déclarées dans glossaryRubriques; le web est un seul index alphabétique — §3.5, point 4bis. Un classeur se feuillette et gagne des points d'entrée; une page web se cherche par mot-clé et n'en a pas besoin. Le contenu, lui, est le même: une source unique, Terms.js.

3.12 kit_check_couplets.py — horodatage partagé des couplets

Vérifie que tous les membres présents d'un *couplet* portent le même *horodatage* — *Kit Prompt* §6.3. Un membre absent est signalé sans faire échouer: un projet n'a pas forcément toutes les feuilles de style, ni un glossaire dans toutes les langues.

Il lit aussi la version exportée par chaque Code.js et la compare à celle du *Registry*: un *couplet* accordé en *horodatage* peut porter deux versions différentes. Une feuille qui n'exporte pas STYLESHEET_VERSION est déclarée non contrôlée plutôt que tenue pour accordée.

- **Rien ne vérifiait cette règle.** Le *validateur* lit un document isolé, le contrôle de fidélité compare un aller-retour, la porte d'*empreinte* regarde les versions. Aucun ne voit deux fichiers côte à côte. Les *couplets* Glossary et YAML ont été rompus le 23 août, celui du glossaire le 24 — les trois fois, l'écart n'a été trouvé qu'à la lecture d'une liste de fichiers.

- **Ce qu'il ne vérifie pas.** Que le contenu des membres concorde. Un .js et sa Reference peuvent porter la même minute et se contredire — c'est le rôle de la lecture.
- **Le groupement se fait par préfixe ET par rôle.** Un répertoire où plusieurs projets cohabitent porte deux *couplets* Glossaire distincts, qui n'ont aucune raison de partager une minute. Grouper sur le seul rôle les déclarait rompus l'un par l'autre — c'est ce que le contrôle a fait à sa première mise en service.
- **Un couplet n'a pas toujours deux membres.** Celui du *Cover Sheet* réunit un .docx de spécification et un rendu.png par langue — *Cover Sheet* §4. Celui du glossaire, le module et une variante par langue.

3.13 Règle d'appel absolue

Aucun fichier n'est livré sans validation. Règle non-négociable. La chaîne complète comporte les maillons suivants, dans cet ordre:

- **Prérequis d'environnement:** vérifier que require('docx') se résout correctement, et require('*adm-zip*') avant une publication du site — voir §3.14. Ce maillon précède tout le reste: sans lui, le générateur échoue ou produit un document vide.
- **Avant toute génération:** *python* kit_check_registry.py. Un *Registry* hors périmètre produit des documents corrects et perd ses réglages à la mise à jour suivante.
- **Avant génération .js:** *python* kit_check_setters.py gen-document.js (pour les générateurs HTML, ajouter --html).
- **Après génération docx:** node kit_validate_docx.js document.docx --version {version active}.
- **Après génération xlsx:** *python* kit_validate_xlsx.py document.xlsx.
- **Après régénération d'un document existant:** *python* kit_check_markup.py source.docx produit.docx. Un diff de texte ne voit pas une perte de marquage; celle-ci en est le seul filet.
- **Après régénération d'une variante de langue:** *python* kit_check_ossature.py reference.docx variante.docx. Vérifie qu'aucune section ni aucun item n'a été perdu ni ajouté.
- **Après une évolution de l'extraction:** *python* kit_check_fidelite.py sur le parc, avec la version active. Ce maillon ne se joue pas à chaque livraison mais à chaque fois que l'outil d'extraction change, et avant toute campagne de régénération.

- **kit_check_couplets.py.** Avant toute livraison touchant un membre de *couplet*. Un *couplet* rompu laisse le lecteur sans savoir lequel des deux fait foi.

Aucun fichier de sortie finale n'est produit tant que la chaîne n'a pas entièrement passé. Un exit non-zéro à n'importe quel maillon interrompt la livraison — arrêt complet, pas de contournement.

3.14 Prérequis d'environnement — sanity checks de session

L'environnement de session repart de zéro à chaque ouverture. Le module *npm* docx, moteur de construction du document, conditionne toute la chaîne. *adm-zip* ne sert plus qu'au générateur de site, qui assemble l'archive de publication: il a quitté la chaîne des documents en General v1.83. Aucun n'est livré dans l'archive: package.json les déclare avec leur version minimale, et ils s'installent toujours à neuf par *npm* install — *Structure commune* §11.

Les vérifications ci-dessous s'exécutent avant tout le reste, en tête de session. Elles sont le maillon zéro de la chaîne §3.13.

3.14.1 Test de sanity

Sur un environnement sain, le module docx expose typiquement entre 270 et 310 clés selon la version mineure. Une valeur nulle indique un état dégradé connu. *adm-zip*, lui, se résout ou ne se résout pas — il n'y a pas d'état intermédiaire.

```
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
```

Critère de décision. Si le compte de clés docx dépasse 200, aucune action; *adm-zip* doit en outre se charger avant une publication du site. Un compte nul signale un module absent ou une installation cassée: réinstaller, jamais patcher. Un compte normal ne garantit rien sur la cohérence de résolution — §3.14.2.

Note: *Cas rencontré en session 2026-08-20: adm-zip absent. Le générateur a construit le document sans erreur puis a échoué à l'appel de injectCustomProps, après Packer.toBuffer() mais avant l'écriture du fichier — donc sans rien corrompre. Un générateur qui envelopperait cet appel dans un try/catch produirait en revanche un .docx valide et ouvrable, mais dépourvu d'empreinte, dont le défaut n'apparaîtrait qu'au moment de publier le site. Depuis General v1.83, l'empreinte ne dépend plus d'adm-zip et ce cas ne peut plus se produire à la génération.*

3.14.2 Résolution incohérente — la cause du corps vide

Le paquet docx expose deux constructions du même code: dist/index.cjs et dist/index.umd.cjs. La résolution ordinaire de Node retient la première; un require portant un chemin de répertoire retient la seconde. Les deux se chargent sans erreur, exposent le même nombre de clés, et produisent des classes distinctes: instanceof est faux de l'une à l'autre.

Un document dont les paragraphes viennent d'une *instance* et le sérialiseur de l'autre sort avec un corps rempli de balises rootKey. Aucune erreur n'est levée. Le fichier s'ouvre, il est vide.

Note: *La règle qui prévient ce cas est unique et tient en une phrase: tous les fichiers d'une même chaîne résolvent docx de la même façon. Depuis General v1.80, cette façon est requireExterne — résolution ordinaire, puis KIT_NODE_MODULES, puis npm root -g. Aucun chemin n'est écrit d'avance. General Reference §1.3.*

Le remède est donc l'*alignement* des résolutions, jamais une modification du module installé. La procédure de patch du package.json de docx, décrite ici jusqu'au 2026-09-01, est retirée: elle traitait un symptôme, portait sur un fichier appartenant à un tiers, et ne survivait pas à une réinstallation.

3.14.3 Persistance et point aveugle

Une réinstallation de docx en cours de session ne change rien à la règle de résolution, qui vit dans les fichiers du Kit et non dans le module. Le diagnostic du §3.14.1 reste utile après coup: il détecte l'absence pure et simple d'un module, cas rencontré le 2026-08-20 pour *adm-zip*.

Point aveugle. Le compte de clés ne voit pas une incohérence de résolution: les deux instances en exposent autant. Seuls le check 13, en aval, et l'aller-retour de fidélité, en amont d'une campagne, l'attrapent. C'est ce dernier qui a établi la cause.

3.15 kit_check_registry.py — périmètre du Registry

- **Ce qu'il vérifie.** Les clés de premier niveau du *Registry* projet, et elles seules. Une clé absente de la liste du Kit est une erreur: elle sera perdue à la prochaine mise à jour, et sa perte ne se verra pas.
- **Contrôle ajouté:** Il contrôle aussi les slugs: un document propre à un moteur porte le préfixe de ce moteur — *Convention de nommage* §3.5 —, et un slug préfixé d'un moteur appartient à ce moteur. Exit 1 sur une faute.
- **Usage.** *python* kit_check_registry.py, ou --*registry* pour désigner un fichier. Exit 1 si une clé est hors périmètre, 2 si le *Registry* est introuvable ou illisible.

Note: *Le fichier est chargé par Node, comme le chargent les générateurs: un parseur maison se tromperait sur un commentaire ou une chaîne contenant une accolade, et le contrôle porterait sur autre chose que ce qui sera réellement lu.*

Note: *Ce qui appartient au projet vit dans [Préfixe] - Settings.js — Structure commune §5.4. Le contenu des blocs du Kit est vérifié ailleurs: kit_check_setters.py pour requires, le générateur de site pour deploy et rendering.*

3.16 kit_gen_document.js — reconstruction depuis une carte

- **Ce qu'il fait.** Reconstruit un document depuis la carte extraite de son fichier source par kit_extract_map.py. Il ne connaît aucun document en particulier: c'est un *artefact stable*, non un générateur de session. Il fait partie de l'archive de téléchargement du Kit, afin qu'un projet dispose dès le départ de tout ce qu'il faut pour travailler. Comme tout *artefact stable*, il n'évolue que lorsque sa logique change — la règle du générateur écrit from scratch, *Kit Prompt* §4.1, vise les *générateurs ad hoc* gen-*.js.
- **Usage.** node kit_gen_document.js carte.json "Nom sans TS" LANG titre sous-titre tagline --ts TS. Les trois modules de style se déclarent par variables d'environnement.

Note: *Il neutralise les insécables posées par les passes du pipeline avant de réinjecter le texte: réinjectées telles quelles, elles empêcheraient un terme de glossaire ou un fragment de mise en évidence d'être redétecté, et le rendu se perdrait sans qu'un contrôle le voie.*

4 Couche 3 — Audits périodiques transversaux

Revues faites en session dédiée. Traitent des dérives qui ne peuvent pas (encore) être détectées runtime ou session. Le but est souvent d'identifier une règle qui mériterait de remonter en *Couche 1* ou 2.

4.1 Audit JSDoc et Reference

Confronte chaque fonction exportée de chaque *stylesheet* avec sa ligne dans la Reference correspondante. Détecte les divergences type de retour, signatures, comportements. Motivation historique: Bug #1 avait révélé que *JSDoc* et Reference de makeTable disaient deux choses différentes. Règle désormais prescrite en Kit Projet *Prompt* §4.1 — l'audit vérifie la conformité continue, pas une dérive inconnue.

- **Portée:** General, Glossary, YAML, HTML — tous les stylesheets ensemble.
- **Livrable:** bump de chaque *couplet stylesheet* avec corrections *JSDoc* + Reference alignée.
- **Tracking:** Bug #8 dans Todos.

4.2 Audit version alignment

Vérifie la cohérence des numéros de version à travers les emplacements suivants. Toute incohérence indique un document non régénéré après bump, ou un fichier partiellement mis à jour.

- **Registry:** source de vérité des versions actives.
- **En-tête du .js:** le commentaire “Version: x.xx” des premières lignes. C'est l'emplacement que *Guide de mise à jour* §4.2 prescrit à l'utilisateur pour sa vérification manuelle.
- **Constante exportée:** STYLESHEET_VERSION, la valeur réellement stampée dans les documents. Un écart avec l'en-tête est invisible au runtime mais fausse toute vérification humaine.
- **Empreinte du .docx:** *custom property* StylesheetVersion de chaque document produit.

Cet audit est naturel en fin de session de modifications *stylesheet*. L'écart entre en-tête et constante a été constaté en session 2026-08-20 sur deux stylesheets bumpés — d'où son ajout explicite à la liste.

4.3 Audit de la couverture multilingue

Scanne les stylesheets à la recherche de strings hardcodées en français qui devraient passer par le pattern *L10N*. État actuel:

- **Glossary stylesheet:** *L10N* complet, alsoLabel seule clé.
- **HTML stylesheet:** *L10N* complet. La clé compiledWithClaudeAI est supprimée au profit de config.compiledWith, et les clés du sélecteur de langue sont ajoutées.
- **General stylesheet:** *L10N* complet (noteLabel, noteLabelCol, sessionTitles). La largeur de la colonne d'étiquette des encadrés note y a rejoint l'étiquette en v1.68: une seule valeur ne pouvait pas décrire des langues dont les étiquettes n'ont pas la même chasse. La clé compiledWithClaudeAI a été supprimée en v1.67 — aucun document.docx ne porte plus de mention de compilation.
- **YAML stylesheet:** rien de notable.

5 Empreintes et traçabilité

Chaque fichier du Kit porte une *empreinte* qui permet de l'identifier et de vérifier sa provenance.

5.1 Custom properties dans les docx

Injectées via style.injectCustomProps(buf, TS). Champs:

- **StylesheetVersion:** version du *stylesheet* utilisé — lue depuis style.STYLESHEET_VERSION.
- **GeneratedAt:** *horodatage* TS du document (format AAAA-MM-JJ - HHhMM). Doit être identique au TS du nom de fichier — un écart signale

un générateur qui passe deux valeurs différentes à titlePage et à injectCustomProps.

5.2 Convention de timestamp couplet

Les paires stylesheet.js + Reference.docx partagent exactement le même TS. Un bump de l'un entraîne la régénération de l'autre avec TS identique. Sans cette contrainte, rien ne dit lequel des deux fait foi. *Inventaire* complet des *couplets*: Kit Projet *Prompt* §6.

- **Couplet glossaire Kit:** Kit - Glossary - Terms.js et les documents *Kit - Glossaire - Termes*, un par langue.
- **Couplet Cover Sheet Kit:** Kit - Documentation - Cover Sheet.docx et ses PNG, un par langue.
- **Couplet Cover Sheet projet:** [Préfixe] - Documentation - Cover Sheet.docx et son PNG — présent si Registry.requires.coverSheet vaut true.

5.3 Registry comme source unique

Kit - *Registry.js* tient les versions, TS et flags de tous les fichiers du Kit. Toute référence de version dans un autre document doit pouvoir être retrouvée dans *Registry*. En cas de divergence, *Registry* fait foi. La structure détaillée du *Registry* Kit est documentée dans *Pipeline HTML* §3.1; celle des *Registry* projet dans *Structure commune* §5.

6 Anti-patterns catalogués

Erreurs fréquentes, identifiées par expérience, à prévenir par la *Couche 1* ou la *Couche 2* dès que possible.

Anti-patron	Conséquence	Défense
Niveau de paragraphe incohérent avec son titre parent	Rendu mal indenté, incohérence visuelle	Inexprimable depuis General v1.70 — le niveau vient du titre, pas du nom de la fonction. Contrôlé en aval par kit_validate_docx.js check 11 (§3.1)
Note directement après heading	Effet paternaliste — “trop instructif”	*Couche 2* — kit_validate_docx.js check 17 (§3.1)
§0 comme numéro de démarrage	Numérotation non-standard, héritage LaTeX	Règle explicite General §1.2, futur check statique
Deux fonctions-Table adjacentes sans releaseParagraph	Fusion visuelle des deux tables en Word	Documenté General §13.4 — discipline générateur
ImageRun sans paramètre type	Fichier word/media/.undefined, donc .docx corrompu	Corrigé par style.makeImageRun obligatoire

Anti-patron	Conséquence	Défense
require('docx') sans chemin global	Conflit copie locale vs global. Écart constaté sur un *stylesheet* importé depuis un environnement tiers en 2026-08-20.	Règle : requireExterne — résolution ordinaire, puis KIT_NODE_MODULES, puis *npm* root -g; aucun chemin écrit d'avance. General Reference §1.3, §3.14.2.
***prompt* pour du code non-dialogue**	Confusion sémantique — *prompt* réservé au *chat*	Règle : yamlStyle.rawBlock pour tout code (JS, bash, SVG, HTML, JSON)
disclaimer() comme style final	Ton paternaliste, déprécié éthiquement	Éliminé — style.tip() à la place
Édition XML docx directe (ElementTree)	Corruption des namespaces, docx illisible	Règle : rewrite from scratch en **Node.js** générateur
***Custom properties* manquantes**	gen-kit-site.js bloque le document	Runtime — injectCustomProps obligatoire
Extracteur JSON générique sans classification sémantique	Tables 1×1 toutes rendues en *prompt*, blocs de code mal catégorisés	Règle : chaque bloc 1×1 classifié manuellement — rawBlock (code), codeBlock (YAML), ou *prompt* (dialogue uniquement)
Largeurs égales gardées sur des colonnes très inégales	Gaspillage pour colonnes numériques ou symboles courts, texte principal compressé	Règle: l'équirépartition par défaut vaut pour des colonnes comparables, et la colonne # est étroite d'office — General §9. Sinon, passer colWidths selon le contenu — symboles ≈ 5-10 %, texte long ≈ 30-40 %.
Regex en écriture sur fichier structuré	Regex ne connaît pas la grammaire du format — peut matcher un cas protégé et corrompre le fichier	Règle Kit Projet *Prompt* §4.1 : toute modification sur fichier structuré passe par le parseur natif. Regex autorisé en lecture pure uniquement.
Changelog inline dans documents Kit-publiés	Logs parasites qui polluent la lecture d'un document de référence	Règle rédactionnelle : aucune section "Historique du document" / changelog inline. Exception unique : Kit - Projet - Todos.

Anti-patron	Conséquence	Défense
Séparateurs ASCII de longueur variable	Rendu visuel incohérent, lisibilité dégradée	Règle Kit Projet *Prompt* §1 checklist : séparateurs unifiés 30 caractères, = majeur, - mineur.
Footer enveloppé dans new Footer({ children: [...] })	docx-js sérialise le Footer enfant en “<options/>” sous <w:ftr>. Word refuse d'ouvrir le fichier (Text Recovery Converter).	*Couche 1* — *JSDoc* explicite makeFooter / makeCoverFooter. *Couche 2* — kit_validate_docx.js check 7.
Synchronisation silencieuse entre *Cover Sheet* et landing HTML par hypothèse de drift	Perte ou ajout silencieux: les deux index sont indépendants — un classeur peut être allégé, la landing omet ce qui ne se lit pas en ligne.	Les deux index sont déclarés séparément (*Pipeline* HTML §2.5). Aucun script ne lit l'un pour piloter l'autre.
Hardcoded Kit-spécifique dans *artefact stable*	Empêche l'usage par les *projets consommateurs*	Lecture des données spécifiques depuis source canonique : .docx via *pandoc AST*, *Registry* projet, fichiers .js de données.
require('docx') retourne objet vide (cache *npm* corrompu)	TypeError immédiat sur tout new Document(), Packer.toBuffer()	Prérequis d’environnement — §3.14
require('docx') vue ESM dégradée — docx silencieusement vide	<rootKey>w:p</rootKey> au lieu de vrais éléments *OOXML* — Word affiche un document vide. Sanity §3.14.1 PASS.	*Couche 2* — kit_validate_docx.js check 13; prévention par l’*alignement* des résolutions, §3.14.2.
<0/> parasite injecté par oubli du *spread* sur makeTable	Tag XML invalide. Word peut tolérer mais *pandoc* HTML échoue silencieusement sur la table cassée.	*Couche 2* — kit_validate_docx.js check 14 (XML strict global). Bloque la livraison.
Numérotation cassée par h2(level, text) au lieu de h2(number, text)	Tous les §2.X s'affichent comme “2”. Document valide mais numérotation incohérente.	*Couche 2* — kit_validate_docx.js check 15 (monotonie heading). Bloque la livraison.
X.1 sans X.2 — sous-titre orphelin	*Anti-pattern* documenté General §1.2. Sous-section solo dont le contenu doit remonter sous le parent.	*Couche 2* — kit_validate_docx.js check 16 (X.1 orphelin). Bloque la livraison.

Anti-patron	Conséquence	Défense
Note/tip isolée immédiatement après heading	*Anti-pattern* documenté General §1.2. Tip/note utilisée comme contenu principal au lieu de souligner un paragraphe précédent.	*Couche 2* — kit_validate_docx.js check 17 (note/tip isolée). Bloque la livraison.
***Setter pipeline* omis en tête de gen-{prefix}-site.js**	Régression silencieuse — site HTML sans hotlinks glossaire, sans small-caps brands, sans routage famille/externe. Le générateur tourne sans erreur.	*Couche 2* — kit_check_setters.py (§3.4). *Pipeline* HTML §7.9 documente la règle. *Linter* exit 1 si *setter* requis manquant.
Références first sans <w:titlePg/> dans le sectPr	Word ignore headers.first et footers.first : la page de garde reçoit l'en-tête navy et le pied de page par défaut. Défaut silencieux, invisible à la *génération*.	*Couche 2* — kit_validate_docx.js check 18. Patron correct : properties: { ...style.pageProps, titlePage: true } — General Reference §10.4.
Réutilisation des helpers de gen-kit-site.js dans un *projet consommateur*	gen-kit-site.js est Kit-only par Convention de nommage §3.6 ; ses helpers sont couplés à la constante SECTIONS Kit. Réutilisation = duplication implicite ou crash sur registry.stylesheets absent du *Registry* projet.	*Couche 1* — gen-kit-site.js module.exports limité à { main } depuis 2026-05-02. Les *projets consommateurs* écrivent leur propre gen-{prefix}-site.js from scratch (*Pipeline* HTML §2.5).
Comparaison de *balise* XML par préfixe	“w:pPr” répond à “w:p” et gonfle la profondeur, que “/w:pPr” ne décrémente pas. L'élément avale tout le reste du corps et le contrôle devient inerte sans rien signaler.	Règle : exiger que le nom de *balise* se termine par un chevron, une espace ou une barre. Helpers isTagStart et findTagStart de kit_validate_docx.js.
Exit zéro d’un contrôle qui n’a rien vérifié	Un *linter* qui ne trouve pas de quoi travailler rend zéro, ce qui se lit comme un contrôle passé. Constaté sur le *linter* de niveaux avant son retrait, dont les cinq générateurs pilotés par carte ne déclenchaient aucun contrôle.	Règle : distinguer “rien à contrôler” de “contrôle passé”. Un outil qui n’a pas trouvé de quoi travailler doit le dire et sortir en erreur, jamais rendre zéro

Anti-patron	Conséquence	Défense
Marque prise pour une introduction grasse	Le *pipeline* rend les marques en petites capitales grasses. Un paragraphe ouvrant sur un nom de marque a donc un premier run gras : extrait comme boldText, il perd ses petites capitales, le boldText étant exclu du *pipeline*.	Règle : un run gras ET en petites capitales est une marque, jamais un lead. Détection par kit_check_markup.py (§3.7).
Insécables du glossaire neutralisés d'après la mauvaise langue	Le texte extrait d'un fichier source porte les insécables posés par la passe glossaire. Les neutraliser d'après les surfaces d'une autre langue laisse le terme non apparié : il perd sa couleur sans qu'aucun mot ne change.	Règle : lire les surfaces via buildSearchTerms(langue du document). Détection par kit_check_markup.py (§3.7).

7 Feedback loop projet vers Kit

Les couches §1.1 sont les défenses internes du Kit — prévention runtime, *validateurs* session, audits transversaux. Une autre source de signal existe: le retour des *projets consommateurs* en usage réel. C'est par ce canal que sont arrivées historiquement les corrections suivantes: brief Sorso (sectionBanner Glossary v1.18, gen-kit-site *setters* HTML — entrée Todos #34), brief Sliver (require ESM dégradé — check 13 ajouté, entrée Todos #27), brief Immobilier (cellule note narrow — kit_patch_helpers et schéma XML normatif *General* §6/§7, entrées Todos #25 et #26), brief *AI* News (markers *i18n* du *renderer Cover Sheet* — v4, entrée Todos #31). Aucun de ces bugs n'aurait été capturé par les Couches 1-3 — ils émergent d'un usage concret hors du périmètre Kit.

À partir de 2026-05-11, ce canal informel est formalisé via un fichier projet dédié: [Préfixe] - Project - Kit *bug report* (TS).docx. Patron de nomenclature standard Convention §2.1, catégorie Project ouverte aux deux préfixes §7. Le *projet consommateur* produit le *bug report* from scratch via gen-kit-bug-report.js, valide avec kit_validate_docx, et livre par dépôt du fichier source dans le *chat* du projet Kit accompagné des artefacts nécessaires (gen-*.js incriminé,.docx résultat foireux, captures éventuelles).

La spécification complète — déclencheurs, nomenclature, structure du document, workflow de remontée, non-permanence du fichier — est dans *Structure commune* §13. Le présent §7 ne duplique pas le contrat: il pointe vers la source de vérité projet et marque la transition canal informel vers canal structuré. Les briefs historiques restent valides comme références dans Todos §4 (entrées #25, #26, #27, #31, #34).