Kit de documentation

Dokumentation — Qualitätssicherung — DE

1 Einleitung

Dieses Dokument ist eine sinngemäße Übersetzung. Maßgeblich ist das Originaldokument auf Französisch; Erweiterungen und Änderungen werden stets dort eingepflegt.

Die *Erzeugung* eines Dokuments durchläuft aufeinanderfolgende Schutzschichten. Die Prävention: Die Stylesheets machen bestimmte Fehler unschreibbar. Die Erkennung: *Validatoren* prüfen die erzeugte Datei, bevor sie ausgeliefert wird. Das Audit: eine regelmäßige Prüfung dessen, was Prävention und Erkennung nicht sehen.

Keine dieser Schichten genügt allein. Ein Fehler, den ein *Stylesheet* nicht verhindern kann, muss weiter hinten abgefangen werden; ein Fehler, den kein Werkzeug erkennen kann, bleibt Sache eines Audits. Die folgenden Abschnitte beschreiben jede Schicht, die Artefakte, die sie umsetzen, und die Reihenfolge, in der sie aufgerufen werden.

Für **Claude AI** als einzige Referenz für alle Fragen zu Audit, Validierung und Qualität bestimmt. Jeder Verweis “Audit” oder *Quality Control* in den anderen Dokumenten des Kits zeigt auf diese Quelle.

1.1 Die Schutzschichten

Jedes erzeugte Dokument oder jede erzeugte Datei durchläuft potenziell die folgenden Kontrollebenen, von vorn nach hinten:

- **Schicht 1 — Prävention zur Laufzeit:** in die Funktionen der Stylesheets eingebaute Prüfungen. Sie blockieren Fehler, bevor das Dokument überhaupt erzeugt wird. Beispiele: makeImageRun lehnt ein ungültiges Format ab, makeHeader lehnt einen leeren Parameter ab.
- **Schicht 2 — Erkennung in der Sitzung:** externe Skripte, die rund um jede *Erzeugung* aufgerufen werden. Zwei getrennte Zeitpunkte. Vorher prüft kit_check_setters.py den Quelltext des Generators vor der Ausführung. Nachher prüfen die *Validatoren* die erzeugte Datei vor der Auslieferung — kit_validate_docx.js und kit_validate_xlsx.py. Hinzu kommen die Umgebungsvoraussetzungen, die vor allem anderen geprüft werden (§3.14).
- **Schicht 3 — Regelmäßiges Audit:** übergreifende Überprüfungen in einer eigenen Sitzung. Sie erkennen schleichende Abweichungen. Beispiele: Audit *JSDoc* und Reference, Audit der Versionsabstimmung, Audit der *L10N*-Abdeckung.

1.2 Grundsatz — fail loud early

Ein spät erkannter Fehler kostet entsprechend mehr. Ein stiller Fehler wird nie behoben. Wann immer sich eine Validierung in der *Pipeline* nach vorn verlagern lässt (Laufzeit statt nur Sitzung, nur Sitzung statt regelmäßiges Audit), muss sie verlagert werden.

💡 *Kann eine Prüfung zur Laufzeit erfolgen, ist das immer besser. Geht es nicht zur*

Laufzeit, dann in der Sitzung. Das regelmäßige Audit ist das letzte Sicherheitsnetz, nicht die Standardarbeitsweise.

2 Schicht 1 — Laufzeitprüfungen in den Stylesheets

Die Laufzeitprüfungen leben in den exportierten Funktionen der Stylesheets. Sie werfen bei einem falschen Aufruf eine ausdrückliche *JavaScript*-Ausnahme — der Generator scheitert sofort mit einer Meldung, die die schuldige Zeile benennt.

2.1 injectCustomProps — Pflichtsignatur

Schreibt die benutzerdefinierten Eigenschaften StylesheetVersion und GeneratedAt in die docx. Pflichtaufruf nach Packer.toBuffer() in jedem Generatorskript.

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

Dokumente ohne diese *Custom Properties* werden von gen-kit-site.js bei der HTML-Umwandlung abgewiesen.

2.2 makeImageRun — Prüfung von Abmessungen und Format

Erzwingt den Parameter type von ImageRun und prüft width/height > 0, Format ∈ {png, jpg, gif, bmp, svg}. Wirft einen ausdrücklichen Fehler, statt stillschweigend eine beschädigte Datei (.docx) zu erzeugen.

2.3 makeHeader — Prüfung der Parameter

makeHeader(title, subtitle) wirft einen ausdrücklichen Fehler, wenn title oder subtitle leer, undefined oder kein String ist. Nach dem Grundsatz *fail loud early* — jeder Generator muss beide übergeben. Siehe *General Reference* §10.8.

2.4 Ebenendisziplin — die Ebene kommt vom Titel

Seit General v1.70 und HTML v1.43 lebt die Abschnittsebene nicht mehr im Namen der Funktionen. h1, h2 und h3 setzen eine aktuelle Ebene auf Modulebene, und alle anderen Familien lesen sie: paragraph('Text') unter einem h2 erzeugt einen Absatz der Ebene 2, ohne dass der Generator es wissen muss. Die fehlende Übereinstimmung zwischen einem Suffix und seinem übergeordneten Titel ist kein zu überwachender Fehler mehr, sie ist unausdrückbar geworden.

Die statische Prüfung des Quelltexts, die diesen Platz einnahm, hat damit ihren Gegenstand verloren, und ihr Werkzeug wurde aus dem Kit entfernt. Die Ebenenprüfung besteht fort und ist eine Stufe nach hinten gerückt: Prüfung 11 von kit_validate_docx.js liest die tatsächliche Einrückung jedes Absatzes im erzeugten Dokument, ohne etwas darüber anzunehmen, wie der Generator geschrieben ist. Sie blockiert und ist nun die einzige *Instanz* in dieser Frage.

2.5 Eingebettete Entscheidungsregeln

In den Reference der einzelnen Stylesheets dokumentiert, zur Laufzeit nicht erzwingbar, aber klar ausdrücklich:

- **Spread Pflicht bei Funktionen, die ein Array zurückgeben:** *General Reference* §13.
- **Übergänge zwischen Tabellenfunktionen:** releaseParagraph() ist zwischen zwei aufeinanderfolgenden Tabellenblöcken Pflicht — *General Reference* §13.4.
- **Nummerierung:** beginnt bei §1, nie bei §0 — *General Reference* §1.2.
- **prompt:** dem Dialog zum Kopieren und Einfügen vorbehalten. Für jeden Code yamlStyle.rawBlock oder codeBlock — *General Reference* §8.

3 Schicht 2 — Stabile Kit-Artefakte und Sitzungsvoraussetzungen

Stabile Skripte, die im *Arbeitsbaum* des Kits leben und vom Nutzer in jedes Projekt kopiert werden, das sie braucht. Präfix kit_ als Hinweis auf ihre Herkunft — Unix-Benennung snake_case, eine förmliche Ausnahme vom Muster aus Namenskonvention §2, benannt in Namenskonvention §3.4. Die Rollen: *Validatoren* (geben 0 zurück, wenn alles in Ordnung ist, sonst einen anderen Wert, und blockieren bei Fehlschlag die Auslieferung), *Renderer* (erzeugen ein deterministisches *Artefakt* — gleiche Eingabe und gleicher *Registry-Kontext* ergeben dieselbe Ausgabe) und Helper (fertigen wiederverwendbare Bausteine, ohne selbst einen Liefergegenstand zu erzeugen).

Wichtige Unterscheidung zu den *Ad-hoc-Generatoren* (gen-*.js): Die stabilen Artefakte haben eine stabile Logik, die von Sitzung zu Sitzung wiederverwendet wird. Sie werden NICHT in jeder Sitzung neu erzeugt — nur wenn sich ihre Logik weiterentwickelt. Die Entwicklung wird in einem Changelog-Block am Anfang der Datei selbst festgehalten (Unix-Konvention), wie bei den Kit-Stylesheets, nicht in einem externen Dokument.

Zwei Regeln betreffen darin die ganze Kette: §3.13 legt die verbindliche Aufrufreihenfolge fest, §3.14 dokumentiert die Umgebungsvoraussetzungen, die zu prüfen sind, bevor irgendetwas ausgeführt wird.

3.1 kit_validate_docx.js — Validator für vom Kit erzeugte docx

Prüft die von den **Node.js**-Stylesheets des Kits erzeugten docx. Beibehaltenes Kit-*Artefakt* — eigenständiges **Node.js**-Skript ohne externe *npm*-Abhängigkeit (nutzt zlib aus der Standardbibliothek zum Lesen des docx-ZIP und einen kleinen XML-Parser aus der Standardbibliothek für die strukturellen *OOXML*-Prüfungen). Dieser Abschnitt ist die einzige Wahrheitsquelle seines Vertrags: Die anderen Dokumente verweisen darauf, ohne ihn zu wiederholen.

- **Prüfung Signatur:** custom.xml vorhanden mit StylesheetVersion und GeneratedAt.
- **Prüfung Version:** die eingetragene StylesheetVersion entspricht der mit --version übergebenen Version. Fehlt sie oder liegt sie darunter: gemeldet. Die Website weist ein Dokument nur ohne *Signatur* oder unter der Untergrenze minDocumentVersion des *Registry* ab — *Pipeline HTML* §7.4.
- **Prüfung Konsistenz:** die Anzahl der <w:tbl> in document.xml entspricht der Anzahl der aus dem Erzeugungs-*AST* erwarteten Tabellenblöcke.
- **Prüfung Absätze in Tabellen:** sind Tabellen vorhanden, muss die Zahl der inneren Absätze größer als null sein. Null bei vorhandenen Tabellen zeigt einen sicheren Verlust an — Falle doc.paragraphs, *Pipeline HTML* §7.1.
- **Prüfung nicht leerer Inhalt:** ein Dokument mit 0 Absätzen UND 0 Tabellen bei vorhandenen *Custom Properties* zeigt fast immer eine stille Beschädigung durch docx-js an — Muster <rootKey>w:p</rootKey>. Nachgewiesene Ursache: zwei verschiedene Instanzen des Moduls docx in derselben Kette. Prüfung 13 ist das Sicherheitsnetz nach der *Erzeugung*; Ursache und Vorbeugung stehen in §3.14.
- **Prüfung Erweiterungen word/media/:** jede in word/media/ vorhandene Erweiterung muss in [Content_Types].xml als Default Extension deklariert sein. Eine verwaiste Erweiterung — typischerweise.undefined, erzeugt von einem ImageRun ohne Parameter type — macht die Datei für Word unlesbar, ohne ausdrückliche *Fehlermeldung*. Siehe *General Reference* §13.3.
- **Prüfung kanonische Reihenfolge <w:rPr>/<w:pPr>:** direkte Kinder werden gegen die *OOXML*-Reihenfolge §17.3.1 und §17.3.2 geprüft. Erkennt den Bug von docx@8.5.0, bei dem rFonts zuletzt statt an 2. Stelle serialisiert wird. Word 365 wendet das XSD-Schema strikt an, *LibreOffice* toleriert.
- **Prüfung nicht leerer Tabellencontainer:** <w:tr> ohne direktes <w:tc> oder <w:tbl> ohne direktes <w:tr>. Word 365 verweigert das Öffnen, *LibreOffice* toleriert stillschweigend.
- **Prüfung Namespaces der Kinder von <w:r>:** eingeschränkte Positivliste (w:, mc:, w14:, w15:, w16se:, w16cid:, w16:). Fängt stille Bugs der Bibliothek docx wie <type>tab</type>, das docx@8.5.0 bei new TextRun({ children: [{ type: 'tab' }] }) serialisiert.
- **Prüfung Übereinstimmung von Überschriftenebene und Einrückung:** für jeden Absatz der obersten Ebene im Body muss das

Attribut <w:ind w:left="X"/> in der gültigen Menge für die Ebene der zuletzt vorangehenden Überschrift liegen. *Modell* aus den Helpern des General-Stylesheets: L1 = {567, 967, 1157}, L2 = {1350, 1750, 1940}, L3 = {2100, 2500, 2690}. Ein Glossardokument, an seinem Namen erkannt — Glossaire - Termes, Glossary - Terms, *Stylesheet* - Glossary - Reference —, lässt unter einer Überschrift der Ebene 1 zusätzlich 1350 und 1550 zu: Begriffsname und Definition des Glossary-Stylesheets. Anderswo bleiben diese Werte eine Anomalie.

- **Prüfung typografische Warnungen NBSP:** erkennt in Dreiergruppen gegliederte Zahlenfolgen, gefolgt von einer bekannten Währungs- oder Maßeinheit (€ $ £ ¥ % m² m³ km kg ha ares hectares °C), deren innere Leerzeichen gewöhnliche (U+0020) statt geschützter Leerzeichen (U+00A0) sind. Nicht blockierende Warnungen. Empfohlener Kit-Helper: style.formatCurrency(amount, opts).
- **Prüfung 14 — Striktes XML global:** prüft für jeden Eintrag.xml oder.rels des ZIP das Fehlen von Tags mit ungültigem Namen (z. B. ein störendes <0/>, eingeschleust durch einen vergessenen *Spread* bei makeTable) und das Gleichgewicht von öffnenden und schließenden Tags mit dem kleinen Parser tokenizeTags.
- **Prüfung 15 — Monotonie der Überschriftennummerierung:** liest die Folge der Heading1/2/3 samt Kit-Nummer aus und prüft, dass jeder Zähler je übergeordnetem Abschnitt um +1 fortschreitet und bei.1 beginnt. Fängt den Bug h2(level, text) statt h2(number, text), der eine gültige Datei mit kaputter Nummerierung erzeugt.
- **Prüfung 16 — Verwaistes X.1:** dokumentiertes *Anti-Pattern* in *Kit Prompt* §4.1 und *General Reference* §1.2 — ein übergeordneter Abschnitt mit nur einem Unterabschnitt ist verdächtig, sein Inhalt gehört unter den übergeordneten Abschnitt. Gruppierung nach übergeordnetem Abschnitt, Einzelfälle werden markiert.
- **Prüfung 17 — Isolierter Hinweis oder Tipp nach einer Überschrift:** dokumentiertes *Anti-Pattern General* §1.2 — nie ein tip oder note direkt nach einer Überschrift ohne vorangehenden Absatz. Erkennung über die Breite der ersten Spalte des <w:tbl>, das unmittelbar auf eine Überschrift der obersten Ebene folgt: TIP_BULB_COL_WIDTH für einen Tipp, und für einen Hinweis die Zugehörigkeit zu NOTE_LABEL_COL_WIDTHS, da die Etikettbreite seit General v1.68 nicht mehr eindeutig ist. Diese Prüfung blieb von ihrer Einführung bis zum 22. August wirkungslos: Ihre Zerlegung der Elemente der obersten Ebene verglich nach Präfix, sodass das Tag der Absatzeigenschaften auf das des Absatzes ansprach und die Tiefe aufblähte. Der erste Absatz des Body verschluckte den ganzen Rest,

und die Schleife sah nie wieder eine Überschrift, auf die eine Tabelle folgt.

- **Prüfung 18 — Deckseite:** Übereinstimmung zwischen dem Flag <w:titlePg/> im sectPr und den Kopf- oder Fußzeilenverweisen vom Typ first. Ohne das Flag ignoriert Word diese Verweise und verwendet ab Seite 1 die Standardkopf- und -fußzeile — die Deckseite verliert ihre Nacktheit. Grundursache: style.pageProps trägt das Flag nicht, das dokumentierte Muster ist properties: {...style.pageProps, titlePage: true } — *General Reference* §10.4. Symmetrische Prüfung: das Flag ohne Verweis first wird ebenfalls gemeldet.
- **Prüfung 19 — Formaterkennung:** erster Eintrag des Archivs. Packer setzt den *Ordner* word/ an den Anfang; jede Neuschreibung durch eine Bibliothek, die die Reihenfolge nicht bewahrt, schiebt ihn hinter die Metadaten, und das Dokument wird dann als gewöhnliches Archiv ausgeliefert — beim Herunterladen in.zip umbenannt. Das Dokument bleibt gültig und öffnet sich ohne Warnung: Diese Prüfung ist die einzige Stelle der Kette, die den Fehler vor dem Empfänger sieht.

Verwendung: node kit_validate_docx.js 'Mon Document (TS).docx' für eine einfache Validierung oder node kit_validate_docx.js 'Mon Document (TS).docx' --version 1.67, um zusätzlich die *Signatur* zu prüfen. Exit-Code 0 = gültig, 1 = Anomalie.

Hinweis: *Der Validator läuft nach jeder Erzeugung einer Datei (.docx), vor dem Kopieren der Datei in den Ausgabeordner und vor ihrer Aufnahme in ein Deployment-ZIP. Ein Exit ungleich null blockiert die Auslieferung — nie umgehen.*

3.2 kit_validate_xlsx.py — Validator für xlsx

Prüft die xlsx, die von den Pipelines erzeugt werden, die solche erstellen — hauptsächlich das Projekt Trading. Die folgenden Prüfungen bilden seinen vollständigen Vertrag.

- **Prüfungen:** ZIP-Struktur, Gleichgewicht von formatCode, Zellbezüge, Blattbezüge in Formeln, Doppellesung mit openpyxl.
- **Verwendung:** *python* kit_validate_xlsx.py document.xlsx [--strict].
- **Beibehaltenes Artefakt:** lebt im *Arbeitsbaum* des Kits und der Projekte, die xlsx erzeugen. Wird nicht in jeder Sitzung neu erzeugt.

3.3 kit_render_cover_sheet.py — Renderer der Deckseite

Deterministischer *Renderer*, der das PNG A4 mit 300 dpi des *Cover Sheet* (Kit oder *Anwenderprojekt*) aus dem zugehörigen Couplet-.docx des *Cover Sheet* erzeugt. Alle spezifischen Werte (HEADER_TEXT, Rubriken §2, überschreibbare *PCL* wie TAB_NUMBERED_MAX) werden über *pandoc-AST* aus der .docx gelesen. Kein Kit-spezifischer Wert ist im Code des Renderers fest verdrahtet — gemäß dem in *Cover Sheet* §3 dokumentierten Vertrag.

CLI: optionale, kombinierbare Argumente. Rückwärtskompatibler Modus erhalten — ein Aufruf ohne Argument löst das aktuelle Kit-Cover-Sheet automatisch über Registry.documents['cover-sheet'].ts auf.

- **--cover-sheet PATH:** zu rendernde Datei (.docx) des *Cover Sheet*. Fehlt sie, automatische Auflösung des aktuellen Kit-Cover-Sheet über das *Registry*.
- **--registry PATH:** Projekt-*Registry* (Standard./Kit - *Registry.js*). Quelle von project.docLanguage und rendering.coverSheet.
- **--output PATH:** Ausgabe-PNG. Fehlt es, wird es von der Quelldatei (.docx) abgeleitet.
- **--language XX:** Sprachüberschreibung (FR | EN | DE | LU). Steuert die lokalisierten Zeichenketten des Kits (subtitle, quote, toc_title).
- **Lokalisierte Marker für nummerierte Rubriken:** FR='numérot', EN='numbered', DE='nummeriert', LU='nummeréiert'. An der Semantik des FR-Markers ausgerichtet (Wurzel länger als die Kurzform der Beschriftung) — vermeidet den in der Sitzung *AI* News dokumentierten Fehlalarm, bei dem die Teilzeichenkette 'number' in den Tab-Zellen der reservierten Zeilen eines englischen Projekt-Cover-Sheet auf 'no number' ansprach. v3 führte die erste *i18n* ein; v4 härtet die Marker EN/DE/LU.
- **Defensiver Zugriff auf das Projekt-Registry:** das Lesen von rendering.coverSheet.sectionHMode verwendet einen sanften Rückfall auf 'FIXED', wenn der Abschnitt rendering fehlt. Ohne diesen Rückfall sofortiger KeyError bei einem Projekt-*Registry*, das dem historischen Gerüst Gemeinsame Struktur §5.2 entspricht. Das Gerüst §5.2 wird parallel erweitert, um den Abschnitt zu dokumentieren — Verteidigung in der Tiefe.
- **TAB_NUMBERED_MAX automatisch abgeleitet:** der Wert wird aus der Zahl der aus §2 extrahierten Rubriken berechnet und hat Vorrang vor dem Kit-Standard. Die *PCL* §3 TAB_NUMBERED_MAX wird optional — nur nützlich, um ausdrücklich zusätzliche *Registerblätter* zu reservieren.

Wird bei jeder *PCL*-Änderung eines *Cover Sheet* aufgerufen. Eigenständiges *Python*-Skript — Abhängigkeiten *Pillow*, zoneinfo, *pandoc*. Couplet-Regel: Das erzeugte PNG trägt denselben TS wie die Quelldatei (.docx) des *Cover Sheet*.

3.4 kit_check_setters.py — Linter für die Setter-Disziplin

Einziger statischer *Linter* der *Schicht 2* vor der Ausführung. Prüft, dass ein Generatorskript am Dateianfang alle Pflicht-*Setter* aufruft, entsprechend den Flags von Registry.requires. Hauptziel: die HTML-Generatoren gen-{prefix}-site.js — lassen sie setGlossaryTerms weg, entsteht stillschweigend eine Website ohne Glossarlinks.

- **Regel:** jedes Flag mit true in Registry.requires verlangt einen Aufruf des zugehörigen Setters (glossary ruft setGlossaryTerms auf, documentTitles setDocumentTitles, brands setBrands, variableNames setVariableNames, hassEntities setHassEntities, textHighlights setTextHighlights). Hinzu kommt setLanguage, immer, dann je nach Ziel: setDocumentAuthor und setDocumentSiteBase für einen.docx-Generator; setWebAuthor, setFamilyDomains und setGlossaryHref, wenn das Projekt mehrere Glossare veröffentlicht, für einen HTML-Generator.
- **Verwendung:** *python* kit_check_setters.py gen-document.js [--*registry* path] [--html] [--strict]. Das Flag --html wendet die Anforderungen an einen HTML-Generator an. Das Flag --strict stuft setLanguage von einer Warnung zu einem Fehler hoch.
- **Exit-Codes:** 0 = alle erforderlichen *Setter* aufgerufen; 1 = Verstöße; 2 = *Registry.js* nicht gefunden.
- **Gekoppelt mit:** *Pipeline HTML* §7.7, das die Liste der Pflicht-*Setter* auf Seiten des Website-Generators dokumentiert. Der *Linter* automatisiert, was die Dokumentation vorschreibt.
- **Beibehaltenes Artefakt:** lebt im *Arbeitsbaum* des Kits und jedes Projekts, das eine HTML-Website erzeugt. Wird nicht in jeder Sitzung neu erzeugt.

Hinweis: *Hervorgegangen aus dem Brief Kit - Adaptations souhaitables (Punkt 4) — Überführung der Dokumentation der Pflicht-Setter in einen automatischen Linter.*

Hinweis: *Konservative statische Erkennung. Der Linter vergleicht den Quelltext der .js mit regulären Ausdrücken, ohne dem bedingten Ablauf zu folgen. Folge: Ein Setter, der unter if (Registry.requires.X) {...} mit X=false im Registry aufgerufen wird, wird als "Setter erkannt" gemeldet, auch wenn der Aufruf zur Laufzeit toter Code ist. Diese Ungenauigkeit ist gewollt und konservativ — sie akzeptiert zu viel und lehnt zu wenig ab —, blockiert also nie eine korrekte Auslieferung. Falsch-Positiv in der Sitzung AI News katalogisiert; keine Korrektur vorgesehen (die Kosten eines flussbewussten JS-Parsers stehen in keinem Verhältnis zum Rauschen).*

3.5 kit_gen_glossaire_html.js — Helper-Generator für das HTML-Glossar

Stabiler Kit-Helper, der die Seite glossaire.html aus [Préfixe] - Glossary - Terms.js erzeugt. Umgeht die *AST-Pipeline* von *pandoc* vollständig — die eigene Struktur der Glossardatei (.docx) (sectionBanner, letterHeader, Inline-Aliasse) ist für den *AST*-Konverter nicht lesbar. Vollständiger normativer Vertrag in *HTML Reference* §14.8 und §15.

- **Absolute Regel:** das HTML-Glossar wird IMMER von diesem Helper erzeugt, NIE in gen-kit-site.js oder in den gen-[projet]-site.js der

Anwenderprojekte nachimplementiert. Strikte Zentralisierung, um stille Abweichungen zu vermeiden.

- **Verwendung:** const { generateGlossaireHtml } = require('./kit_gen_glossaire_html.js'); generateGlossaireHtml({ termsPath, outputPath, htmlStyle, config }).
- **Beibehaltenes Artefakt:** lebt im *Arbeitsbaum* des Kits und jedes Anwenderprojekts, das ein Glossar veröffentlicht.
- **Abhängigkeiten:** keine externen *npm*-Pakete. Liest Terms.js über require, verwendet die Klassen.gloss-* des HTML-Stylesheets sowie Titel und Brotkrumenpfad, lokalisiert über htmlStyle.getStrings().glossaryTitle (hinzugefügt in HTML v1.23).

3.6 kit_patch_helpers.py — Python-Helper für gezielte Patches an .docx

Python-Helper, der bei gezielten Änderungen an einer bestehenden Datei (.docx) XML-Bausteine erzeugt, die dem General-*Stylesheet* entsprechen. Deckt die komplexen Strukturen ab (note 1×2 usw.), die zur Fehlerquelle werden, wenn man sie durch Klonen einer Vorlage oder von Hand baut. Die Bausteine werden VON GRUND AUF aus den synchronisierten Kit-Konstanten gebaut — nicht durch Klonen einer Vorlage des Zieldokuments.

- **Absolute Regel:** um einen Block note, tip oder table per gezieltem *Patch* in eine bestehende Datei (.docx) einzufügen (Gesprächsbefehle §2.4), sind zwingend die Helper dieses Moduls zu verwenden — nie einen Baustein des Zieldokuments von Hand klonen.
- **Aktuelle API:** build_note_block(content, level=1, lang='FR') — gibt einen XML-Baustein <w:tbl>...</w:tbl> zurück, der einen Kit-Hinweis der Ebene 1, 2 oder 3 darstellt.
- **Synchronisierung:** fest verdrahtete Kit-Konstanten, ausdrücklich mit dem General-Code bei jeder Weiterentwicklung synchronisiert. Dokumentierte Abhängigkeit — keine stille Drift.
- **Abhängigkeiten:** keine externen. Nur *Python*-Standardbibliothek. Keine regulären Ausdrücke beim Schreiben strukturierter Dateien.
- **Beibehaltenes Artefakt:** lebt im *Arbeitsbaum* des Kits. Wird bei der Einrichtung oder Aktualisierung eines Projekts, das es braucht, unverändert kopiert.

3.7 kit_check_markup.py — Prüfung der Auszeichnung und der Übereinstimmung der Konstanten

Zwei Prüfungen, die kein Werkzeug des Kits leistete, beide aus Fehlern entstanden, die durch die bestehenden Ketten gerutscht waren. Ein Textvergleich sieht keinen Farbverlust: Ein neu erzeugtes Dokument kann genau dieselben Wörter in derselben Reihenfolge tragen und dennoch die Auszeichnung mehrerer Begriffe oder mehrerer Marken verloren haben.

- **Prüfung der Auszeichnung:** zählt die Runs, die jede Auszeichnung tragen — Glossar, Marken, Variablennamen, Entitäten, Fettdruck, Hyperlinks, Bilder — in der Quelldatei und im erzeugten Dokument und vergleicht. Jeder Verlust wird gemeldet. Der Modus --strict meldet auch Zugewinne, für eine Neuerzeugung, die am Inhalt nichts ändern darf; --added N gibt die Zahl der erwarteten fetten Etiketten an.
- **Prüfung der Etikettbreiten von Hinweisen:** der Wert lebt in drei Exemplaren — Tabelle localizedStrings von General, NOTE_LABEL_COL_WIDTHS des Validators, NOTE_LABEL_COL_BY_LANG der *Patch*-Helper. Die Verdopplung ist gewollt, da die beiden Werkzeuge das *Stylesheet* nicht importieren können. Die Prüfung liest die drei Quellen zur Laufzeit und prüft ihre Übereinstimmung.
- **Verwendung:** *python* kit_check_markup.py SOURCE.docx PRODUIT.docx [--strict] [--added N] oder *python* kit_check_markup.py --widths [--dir DOSSIER]. Exit 0 bei Übereinstimmung, 1 bei Abweichung, 2 bei Bedienungsfehler.
- **Beibehaltenes Artefakt:** lebt im *Arbeitsbaum* des Kits und jedes Projekts, das bestehende Dokumente neu erzeugt.

Hinweis: *Diese Prüfung ist die einzige, die einen Auszeichnungsverlust sieht. Zwei echte Fehler wurden durch sie vor der Auslieferung abgefangen: elf englische Begriffe, die ihre Farbe verloren hatten, weil die geschützten Leerzeichen des extrahierten Texts nach den französischen Formen neutralisiert worden waren, und fünf Marken, die ihre Kapitälchen verloren hatten, weil ein Absatz, der mit einem Markennamen begann, für einen Absatz mit fettem Einstieg gehalten worden war. In keinem der beiden Fälle hatte sich ein Wort geändert.*

3.8 kit_check_ossature.py — Prüfung des Gerüsts zwischen Sprachvarianten

Vergleicht das Gerüst einer Übersetzung mit dem ihrer Quelle. §17 des *Prompt* verlangt eine sinngetreue Übersetzung, in einem Text, der sich liest, als wäre er in seiner Sprache geschrieben: Wörter zu zählen oder Satz für Satz zu vergleichen, hieße genau das zu bestrafen, was die Regel verlangt.

- **Was gemessen wird:** die Folge der Blöcke, ihr Typ und ihre Ebene, die Nummerierung der Abschnitte, die Abmessungen jeder Tabelle, die Zahl der Einträge jeder Liste.
- **Was nicht gemessen wird:** die Zahl der Wörter, die Länge der Sätze, die Entsprechung Satz für Satz. Das Deutsche setzt zusammen, das Englische kürzt, und eine Übersetzung, die die Länge des Französischen nachbildete, wäre schlecht.
- **Was gemeldet wird:** ein Abschnitt, ein Eintrag, eine Tabellenzeile oder ein Kasten, der auf einer Seite vorhanden ist und auf der anderen

fehlt. Das ist eine verlorene oder hinzugefügte Information, was §17 verbietet.

- **Toleranz:** die Übersetzungsnotiz in §1, die die Referenzvariante nicht trägt, wird erkannt und akzeptiert. Die Option --sans-tolerance lehnt sie ab.
- **Verwendung:** *python* kit_check_ossature.py REFERENCE.docx VARIANTE.docx [AUTRES…]. Exit 0, wenn die Gerüste übereinstimmen, sonst 1.

3.9 kit_extract_map.py — Strukturkarte eines bestehenden Dokuments

Jede Neuerzeugung geht von einer aus der Quelldatei extrahierten Karte aus. Das Werkzeug liest das XML des Dokuments und gibt dessen Struktur wieder: Reihenfolge der Blöcke, Typ, Ebene, Inhalt, Auszeichnung. Es arbeitet nur lesend und verwendet keinen regulären Ausdruck.

- **Warum es stabil ist, während die Generatoren es nicht sind:** der *Prompt* §4.1 verlangt, jeden Generator von Grund auf zu schreiben, weil ein Generator zu einem bestimmten Dokument gehört. Die Extraktion teilt diese Eigenschaft nicht: Sie tut immer dasselbe, und sie in jeder Sitzung neu zu schreiben, erzeugt nur Varianten desselben Codes, jede mit ihren eigenen blinden Flecken.
- **Was es erkennt:** Titel, Absätze und ihre Varianten nach Einrückung, Listeneinträge und Fortsetzungen, Kästen note, tip und *prompt*, rohe und eingefärbte Codeblöcke, gewöhnliche Tabellen, metaTable des YAML-Stylesheets, termName und definition des Glossary-Stylesheets, Bilder, Hyperlinks, Brücken- und Freigabeabsätze. Ein Bild, allein oder in einer Zelle, wird als Datei im *Ordner* <Karte>-images abgelegt und bei der Neuerzeugung wieder eingesetzt; die Glühbirne eines Kastens wird nicht erfasst, das *Stylesheet* zeichnet sie neu. Eine Tabelle vermerkt, ob ihre erste Zeile eine Kopfzeile ist, und wird sonst ohne Kopfzeile neu aufgebaut; eine einspaltige Tabelle ist eine gewöhnliche Tabelle. Der Link einer Zelle wird mit seinem Ankertext und seinem Ziel erfasst, aufgelöst über die Beziehungen des Dokuments, und von hyperlinkCell wieder gesetzt.
- **Was es nicht versprechen kann:** das Repertoire ist endlich. Eine Konstruktion, die darin nicht vorkommt, wird verschlechtert, ohne dass etwas in der erzeugten Datei darauf hinweist. Das ist der Grund für §3.10.
- **Verwendung:** *python* kit_extract_map.py SOURCE.docx CIBLE.json.

3.10 kit_check_fidelite.py — Prüfung der Treue der Extraktion

Extrahiert, erzeugt neu und vergleicht das XML des Body bytegenau. Ein Dokument, das nicht identisch zurückkommt, zeigt eine Konstruktion an, die die Extraktion nicht wiedergeben kann. Am 23. August 2026 wurden an

einem Tag vier solche Fehler gefunden, alle still und alle für einen Textvergleich unsichtbar; diese Prüfung hätte sie auf einen Schlag gefunden.

- **Die Neutralisierungen, ohne die der Test etwas anderes misst:** *Zeitstempel* und Beziehungskennungen der Hyperlinks ändern sich bei jeder *Erzeugung*, ohne dass sich die Darstellung bewegt. Die Sprache ergibt sich aus dem Kürzel im Dateinamen — ein deutsches Dokument mit den französischen Glossarformen neu zu erzeugen, führt zu einer Abweichung, die nichts über die Extraktion aussagt. Und die nachgespielte Kette muss die der Produktion sein, einschließlich der Neutralisierung der Markierungen der *Pipeline*.
- **Was es nicht verspricht:** es sieht nur die im Bestand vorhandenen Konstruktionen. Ein Dokument, das morgen einen nie verwendeten Helper nutzt, rutschte durch. Die Prüfung verkleinert die Fehlerfläche, ohne sie zu schließen: Was als sicher gilt, gilt es im Licht dessen, was man heute weiß, und das ist die gewöhnliche Bedingung jeder Qualitätssicherung. Tritt ein unvorhergesehener Fall auf, wird die Extraktion ergänzt und der Bestand erneut geprüft.
- **Geltungsbereich:** ein Dokument, das mit einer älteren *Stylesheet*-Version erzeugt wurde, weicht berechtigt ab. Das Werkzeug meldet es gesondert, statt es als Fehlschlag zu zählen.
- **Verwendung:** *python* kit_check_fidelite.py SOURCE.docx […] --version {version active}. Exit 0, wenn alles identisch zurückkommt, sonst 1.
- **Ein unverändertes Dokument behält seinen Zeitstempel.** Der frische *Zeitstempel* ist bei jeder AUSLIEFERUNG verlangt — *Kit Prompt* §4.1 —, nicht bei jedem Generatorlauf. Eine Neuerzeugungskampagne vergleicht den erzeugten Body mit dem der Quelle, mit derselben Neutralisierung wie hier; ist er identisch, verwirft sie das Erzeugnis und behält die Quelle. Ohne diesen Vergleich behandelt das Veröffentlichungsdelta den ganzen Bestand als geändert, das lokale Inventar wird vollständig neu geschrieben, und der Verlauf der Website verliert seinen Sinn: Sechsundvierzig Dokumente tragen dort dieselbe Minute.
- **Das Glossar ist von der Prüfung ausgenommen.** Es wird nicht durch Extraktion neu erzeugt — §3.11 —, und der hier geprüfte Hin- und Rückweg hat für es keinen Sinn. Es wird in der Übersicht gesondert gezählt, nie als Abweichung. Die Reference des Glossar-Stylesheets folgt seit dem 2026-09-17 demselben Weg: Sie trägt von diesem *Stylesheet* gerenderte Elemente — Banner, Buchstabe, Begriff, Definition —, die die Karte nicht wiedergeben kann, und wird von ihrem Sitzungsgenerator neu erzeugt.

Hinweis: *Die Artefakte rufen kit_extract_map.py über einen absoluten Pfad auf, der aus ihrem*

eigenen Speicherort berechnet wird. Ein relativer Pfad band sie an das Arbeitsverzeichnis: kit_check_ossature.py rief zudem einen Namen aus der Zeit vor der Überführung in ein stabiles Artefakt auf, extract_map.py, und scheiterte bei jedem Aufruf, ohne dass etwas darauf hinwies — das Scheitern zeigte sich erst beim ersten Vergleich von Varianten.

3.11 kit_gen_glossaire_docx.js — Helper-Generator für das Glossar als .docx

Erzeugt die Glossardatei (.docx) aus [Préfixe] - Glossary - Terms.js. Gegenstück zu kit_gen_glossaire_html.js — §3.5: eine Quelle, zwei Darstellungen.

- **Das Glossar als .docx wird nie durch Extraktion neu erzeugt.** Es ist das einzige Dokument des Kits, dessen Inhalt in einem Modul lebt. Aus seiner in der Quelldatei gelesenen Karte neu erzeugt, erstarrt es: Drei am 24. August zu Terms.js hinzugefügte Begriffe fehlten in den drei Varianten bis zu einer Prüfung von Hand.
- **Ruft die Setter der Pipeline auf.** Ohne sie bleiben Glossarbegriffe und Marken in den Definitionen schwarz.
- **Von der Gerüstprüfung ausgenommen.** Die Begriffe werden in der dargestellten Sprache sortiert, daher unterscheiden sich die Gruppierungen nach Buchstaben von einer Variante zur anderen. kit_check_ossature.py schließt es ausdrücklich aus — §3.8.
- **Die beiden Darstellungen haben nicht dieselbe Struktur.** Das Papier folgt den in glossaryRubriques deklarierten Rubriken; das Web ist ein einziger alphabetischer Index — §3.5, Punkt 4bis. Ein *Ordner* wird durchgeblättert und gewinnt Einstiegspunkte; eine Webseite wird per Stichwort durchsucht und braucht keine. Der Inhalt ist derselbe: eine einzige Quelle, Terms.js.

3.12 kit_check_couplets.py — gemeinsamer Zeitstempel der Couplets

Prüft, dass alle vorhandenen Mitglieder eines Couplets denselben *Zeitstempel* tragen — *Kit Prompt* §6.3. Ein fehlendes Mitglied wird gemeldet, ohne die Prüfung scheitern zu lassen: Ein Projekt hat nicht unbedingt alle Stylesheets oder ein Glossar in allen Sprachen.

Es liest außerdem die von jeder Code.js exportierte Version und vergleicht sie mit der des *Registry*: Ein im *Zeitstempel* übereinstimmendes Couplet kann zwei verschiedene Versionen tragen. Ein *Stylesheet* ohne STYLESHEET_VERSION wird als ungeprüft ausgewiesen, statt als übereinstimmend zu gelten.

- **Nichts prüfte diese Regel.** Der *Validator* liest ein einzelnes Dokument, die Treueprüfung vergleicht einen Rundlauf, die

Signaturschranke betrachtet die Versionen. Keiner sieht zwei Dateien nebeneinander. Die Couplets Glossary und YAML wurden am 23. August gebrochen, das des Glossars am 24. — alle drei Male wurde die Abweichung erst beim Lesen einer Dateiliste gefunden.

- **Was es nicht prüft.** Dass der Inhalt der Mitglieder übereinstimmt. Eine .js und ihre Reference können dieselbe Minute tragen und einander widersprechen — das ist Aufgabe der Lektüre.
- **Die Gruppierung erfolgt nach Präfix UND nach Rolle.** Ein Verzeichnis, in dem mehrere Projekte nebeneinander bestehen, enthält zwei getrennte Glossar-Couplets, die keinen Grund haben, eine Minute zu teilen. Die Gruppierung allein nach Rolle erklärte sie gegenseitig für gebrochen — genau das tat die Prüfung bei ihrer ersten Inbetriebnahme.
- **Ein Couplet hat nicht immer zwei Mitglieder.** Das des *Cover Sheet* vereint eine Spezifikationsdatei (.docx) und eine Darstellung (.png) je Sprache — *Cover Sheet* §4. Das des Glossars das Modul und eine Variante je Sprache.

3.13 Absolute Aufrufregel

Keine Datei wird ohne Validierung ausgeliefert. Nicht verhandelbare Regel. Die vollständige Kette umfasst die folgenden Glieder in dieser Reihenfolge:

- **Umgebungsvoraussetzungen:** prüfen, dass require('docx') korrekt aufgelöst wird, und require('*adm-zip*') vor einer Veröffentlichung der Website — siehe §3.14. Dieses Glied geht allem anderen voraus: Ohne es scheitert der Generator oder erzeugt ein leeres Dokument.
- **Vor jeder Erzeugung:** *python* kit_check_registry.py. Ein *Registry* außerhalb des Geltungsbereichs erzeugt korrekte Dokumente und verliert seine Einstellungen bei der nächsten Aktualisierung.
- **Vor der Erzeugung .js:** *python* kit_check_setters.py gen-document.js (bei HTML-Generatoren --html hinzufügen).
- **Nach der Erzeugung docx:** node kit_validate_docx.js document.docx --version {version active}.
- **Nach der Erzeugung xlsx:** *python* kit_validate_xlsx.py document.xlsx.
- **Nach der Neuerzeugung eines bestehenden Dokuments:** *python* kit_check_markup.py source.docx produit.docx. Ein Textvergleich sieht keinen Auszeichnungsverlust; diese Prüfung ist das einzige Netz dafür.
- **Nach der Neuerzeugung einer Sprachvariante:** *python* kit_check_ossature.py reference.docx variante.docx. Prüft, dass kein Abschnitt und kein Eintrag verloren oder hinzugekommen ist.

- **Nach einer Weiterentwicklung der Extraktion:** *python* kit_check_fidelite.py auf dem Bestand, mit der aktiven Version. Dieses Glied läuft nicht bei jeder Auslieferung, sondern jedes Mal, wenn sich das Extraktionswerkzeug ändert, und vor jeder Neuerzeugungskampagne.
- **kit_check_couplets.py.** Vor jeder Auslieferung, die ein Mitglied eines Couplets berührt. Ein gebrochenes Couplet lässt den Leser im Unklaren, welches der Mitglieder maßgeblich ist.

Keine endgültige Ausgabedatei wird erzeugt, solange die Kette nicht vollständig durchlaufen ist. Ein Exit ungleich null an einem beliebigen Glied unterbricht die Auslieferung — vollständiger Abbruch, keine Umgehung.

3.14 Umgebungsvoraussetzungen — Plausibilitätsprüfungen der Sitzung

Die Sitzungsumgebung beginnt bei jedem Öffnen bei null. Das *npm*-Modul docx, der Motor für den Aufbau des Dokuments, bedingt die ganze Kette. *adm-zip* dient nur noch dem Website-Generator, der das Veröffentlichungsarchiv zusammenstellt: Es hat die Kette der Dokumente mit General v1.83 verlassen. Keines von beiden liegt dem Archiv bei: package.json erklärt sie mit ihrer Mindestversion, und sie werden stets neu mit *npm* install installiert — Gemeinsame Struktur §11.

Die folgenden Prüfungen laufen vor allem anderen, am Anfang der Sitzung. Sie sind das Glied null der Kette §3.13.

3.14.1 Plausibilitätstest

In einer gesunden Umgebung stellt das Modul docx typischerweise zwischen 270 und 310 Schlüssel bereit, je nach Nebenversion. Ein Wert von null zeigt einen bekannten beschädigten Zustand an. *adm-zip* wird aufgelöst oder nicht — einen Zwischenzustand gibt es nicht.

```
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
```

Entscheidungskriterium. Liegt die Zahl der docx-Schlüssel über 200, keine Maßnahme; *adm-zip* muss sich außerdem vor einer Veröffentlichung der Website laden lassen. Eine Zahl von null zeigt ein fehlendes Modul oder eine kaputte Installation an: neu installieren, nie patchen. Eine normale Zahl garantiert nichts über die Stimmigkeit der Auflösung — §3.14.2.

Hinweis: *In der Sitzung 2026-08-20 aufgetretener Fall: adm-zip fehlte. Der Generator baute das Dokument fehlerfrei auf und scheiterte dann beim Aufruf von*

injectCustomProps, nach Packer.toBuffer(), aber vor dem Schreiben der Datei — also ohne etwas zu beschädigen. Ein Generator, der diesen Aufruf in ein try/catch einschlösse, erzeugte dagegen eine gültige, zu öffnende Datei (.docx), aber ohne Signatur, deren Mangel erst bei der Veröffentlichung der Website auffiele. Seit General v1.83 hängt die Signatur nicht mehr von adm-zip ab, und dieser Fall kann bei der Erzeugung nicht mehr auftreten.

3.14.2 Uneinheitliche Auflösung — die Ursache des leeren Body

Das Paket docx stellt zwei Builds desselben Codes bereit: dist/index.cjs und dist/index.umd.cjs. Die gewöhnliche Auflösung von Node wählt den ersten; ein require mit einem Verzeichnispfad wählt den zweiten. Beide laden fehlerfrei, stellen gleich viele Schlüssel bereit und erzeugen verschiedene Klassen: instanceof ist von einem zum anderen falsch.

Ein Dokument, dessen Absätze aus der einen *Instanz* und dessen Serialisierer aus der anderen stammen, kommt mit einem Body voller rootKey-Tags heraus. Es wird kein Fehler geworfen. Die Datei öffnet sich, sie ist leer.

Hinweis: *Die Regel, die diesem Fall vorbeugt, ist einzig und passt in einen Satz: Alle Dateien derselben Kette lösen docx auf dieselbe Weise auf. Seit General v1.80 ist diese Weise requireExterne — gewöhnliche Auflösung, dann KIT_NODE_MODULES, dann npm root -g. Kein Pfad wird im Voraus festgeschrieben. General Reference §1.3.*

Die Abhilfe ist daher die Angleichung der Auflösungen, nie eine Änderung des installierten Moduls. Das Verfahren zum Patchen der package.json von docx, das hier bis zum 2026-09-01 beschrieben war, ist zurückgezogen: Es behandelte ein Symptom, betraf eine Datei, die einem Dritten gehört, und überstand keine Neuinstallation.

3.14.3 Beständigkeit und blinder Fleck

Eine Neuinstallation von docx im Laufe der Sitzung ändert nichts an der Auflösungsregel, die in den Dateien des Kits lebt und nicht im Modul. Die Diagnose aus §3.14.1 bleibt nachträglich nützlich: Sie erkennt das schlichte Fehlen eines Moduls, den am 2026-08-20 bei *adm-zip* aufgetretenen Fall.

Blinder Fleck. Die Schlüsselzahl sieht keine uneinheitliche Auflösung: Beide Instanzen stellen gleich viele bereit. Nur Prüfung 13 hinterher und der Treue-Rundlauf vor einer Kampagne fangen sie ab. Letzterer hat die Ursache ermittelt.

3.15 kit_check_registry.py — Geltungsbereich des Registry

– **Was es prüft.** Die Schlüssel der obersten Ebene des Projekt-*Registry*, und nur sie. Ein Schlüssel, der nicht in der Liste des Kits steht, ist ein

Fehler: Er geht bei der nächsten Aktualisierung verloren, und sein Verlust wird nicht sichtbar.

- **Contrôle ajouté:** Es prüft außerdem die Slugs: Ein einem *Modell* eigenes Dokument trägt dessen Präfix — Namenskonvention §3.5 —, und ein mit einem *Modell* präfigierter Slug gehört diesem *Modell*. Exit 1 bei einem Verstoß.
- **Verwendung.** *python* kit_check_registry.py oder --*registry*, um eine Datei anzugeben. Exit 1, wenn ein Schlüssel außerhalb des Geltungsbereichs liegt, 2, wenn das *Registry* nicht gefunden oder nicht gelesen werden kann.

Hinweis: *Die Datei wird von Node geladen, so wie die Generatoren sie laden: Ein eigener Parser würde sich bei einem Kommentar oder einer Zeichenkette mit einer geschweiften Klammer irren, und die Prüfung beträfe etwas anderes als das, was tatsächlich gelesen wird.*

Hinweis: *Was dem Projekt gehört, lebt in [Préfixe] - Settings.js — Gemeinsame Struktur §5.4. Der Inhalt der Blöcke des Kits wird anderswo geprüft: kit_check_setters.py für requires, der Website-Generator für deploy und rendering.*

3.16 kit_gen_document.js — Wiederaufbau aus einer Karte

- **Was es tut.** Baut ein Dokument aus der Karte wieder auf, die kit_extract_map.py aus seiner Quelldatei extrahiert hat. Es kennt kein bestimmtes Dokument: Es ist ein *stabiles Artefakt*, kein Sitzungsgenerator. Es ist Teil des Download-Archivs des Kits, damit ein Projekt von Anfang an alles hat, was es zum Arbeiten braucht. Wie jedes stabile *Artefakt* ändert es sich nur, wenn sich seine Logik ändert — die Regel des von Grund auf geschriebenen Generators, *Kit Prompt* §4.1, betrifft die *Ad-hoc-Generatoren* gen-*.js.
- **Verwendung.** node kit_gen_document.js carte.json “Nom sans TS” LANG titre sous-titre tagline --ts TS. Die drei Stilmodule werden über Umgebungsvariablen angegeben.

Hinweis: *Es neutralisiert die von den Durchläufen der Pipeline gesetzten geschützten Leerzeichen, bevor es den Text wieder einspeist: Unverändert wieder eingespeist, verhinderten sie, dass ein Glossarbegriff oder ein hervorgehobenes Fragment erneut erkannt wird, und die Darstellung ginge verloren, ohne dass eine Prüfung es sähe.*

4 Schicht 3 — Regelmäßige übergreifende Audits

Überprüfungen in einer eigenen Sitzung. Sie behandeln Abweichungen, die (noch) nicht zur Laufzeit oder in der Sitzung erkannt werden können. Oft geht es darum, eine Regel zu finden, die es verdiente, in *Schicht 1* oder 2 aufzusteigen.

4.1 Audit JSDoc und Reference

Stellt jede exportierte Funktion jedes Stylesheets ihrer Zeile in der zugehörigen Reference gegenüber. Erkennt Abweichungen bei Rückgabetyp,

Signaturen, Verhalten. Historischer Anlass: Bug #1 hatte gezeigt, dass *JSDoc* und Reference von makeTable zwei verschiedene Dinge sagten. Die Regel ist nun in *Kit Prompt* §4.1 vorgeschrieben — das Audit prüft die fortlaufende Einhaltung, keine unbekannte Abweichung.

- **Geltungsbereich:** General, Glossary, YAML, HTML — alle Stylesheets zusammen.
- **Liefergegenstand:** Versionssprung jedes *Stylesheet*-Couplets mit *JSDoc*-Korrekturen und angeglichener Reference.
- **Nachverfolgung:** Bug #8 in Todos.

4.2 Audit der Versionsabstimmung

Prüft die Stimmigkeit der Versionsnummern über die folgenden Stellen hinweg. Jede Unstimmigkeit zeigt ein nach dem Versionssprung nicht neu erzeugtes Dokument oder eine teilweise aktualisierte Datei an.

- **Registry:** Wahrheitsquelle der aktiven Versionen.
- **Kopf der .js:** der Kommentar “Version: x.xx” in den ersten Zeilen. Das ist die Stelle, die Aktualisierungsleitfaden §4.2 dem Nutzer für seine Prüfung von Hand vorschreibt.
- **Exportierte Konstante:** STYLESHEET_VERSION, der tatsächlich in die Dokumente eingetragene Wert. Eine Abweichung vom Kopf ist zur Laufzeit unsichtbar, verfälscht aber jede menschliche Prüfung.
- **Signatur der .docx:** *Custom Property* StylesheetVersion jedes erzeugten Dokuments.

Dieses Audit bietet sich am Ende einer Sitzung mit *Stylesheet*-Änderungen an. Die Abweichung zwischen Kopf und Konstante wurde in der Sitzung 2026-08-20 bei zwei Stylesheets nach einem Versionssprung festgestellt — daher ihre ausdrückliche Aufnahme in die Liste.

4.3 Audit der mehrsprachigen Abdeckung

Durchsucht die Stylesheets nach fest auf Französisch codierten Zeichenketten, die über das *L10N*-Muster laufen sollten. Aktueller Stand:

- **Glossary-Stylesheet:** *L10N* vollständig, alsoLabel einziger Schlüssel.
- **HTML-Stylesheet:** *L10N* vollständig. Der Schlüssel compiledWithClaudeAI ist zugunsten von config.compiledWith entfernt, und die Schlüssel des Sprachwählers sind hinzugefügt.
- **General-Stylesheet:** *L10N* vollständig (noteLabel, noteLabelCol, sessionTitles). Die Breite der Etikettspalte der Hinweiskästen ist in v1.68 zum Etikett hinzugekommen: Ein einziger Wert konnte keine Sprachen beschreiben, deren Etiketten nicht dieselbe Laufweite haben. Der Schlüssel compiledWithClaudeAI wurde in v1.67 entfernt — keine Datei (.docx) trägt mehr einen Kompilierungsvermerk.

- **YAML-Stylesheet:** nichts Bemerkenswertes.

5 Signaturen und Rückverfolgbarkeit

Jede Datei des Kits trägt eine *Signatur*, mit der sie sich identifizieren und ihre Herkunft prüfen lässt.

5.1 Custom Properties in den docx

Eingetragen über style.injectCustomProps(buf, TS). Felder:

- **StylesheetVersion:** Version des verwendeten Stylesheets — gelesen aus style.STYLESHEET_VERSION.
- **GeneratedAt:** *Zeitstempel* TS des Dokuments (Format AAAA-MM-JJ - HHhMM). Muss mit dem TS des Dateinamens identisch sein — eine Abweichung zeigt einen Generator an, der titlePage und injectCustomProps zwei verschiedene Werte übergibt.

5.2 Zeitstempelkonvention der Couplets

Die Paare stylesheet.js + Reference.docx teilen genau denselben TS. Ein Versionssprung des einen zieht die Neuerzeugung des anderen mit identischem TS nach sich. Ohne diese Bedingung sagt nichts, welches der beiden maßgeblich ist. Vollständiges Inventar der Couplets: *Kit Prompt* §6.

- **Glossar-Couplet des Kits:** Kit - Glossary - Terms.js und die Dokumente *Kit - Glossaire - Termes*, eines je Sprache.
- **Cover-Sheet-Couplet des Kits:** Kit - Documentation - Cover Sheet.docx und seine PNG, eines je Sprache.
- **Cover-Sheet-Couplet des Projekts:** [Préfixe] - Documentation - Cover Sheet.docx und sein PNG — vorhanden, wenn Registry.requires.coverSheet true ist.

5.3 Registry als einzige Quelle

Kit - *Registry.js* hält die Versionen, TS und Flags aller Dateien des Kits. Jede Versionsangabe in einem anderen Dokument muss sich im *Registry* wiederfinden lassen. Bei Abweichung ist das *Registry* maßgeblich. Die ausführliche Struktur des Kit-*Registry* ist in *Pipeline HTML* §3.1 dokumentiert; die der Projekt-*Registry* in Gemeinsame Struktur §5.

6 Katalogisierte Anti-Patterns

Häufige, aus Erfahrung erkannte Fehler, denen so früh wie möglich durch *Schicht 1* oder *Schicht 2* vorzubeugen ist.

Anti-Pattern	Folge	Abwehr
Absatzebene passt nicht zum übergeordneten Titel	Falsch eingerückte Darstellung, optische Unstimmigkeit	Seit General v1.70 unausdrückbar — die Ebene kommt vom Titel, nicht vom Funktionsnamen. Nachgelagert geprüft durch kit_validate_docx.js Prüfung 11 (§3.1)
Hinweis direkt nach einer Überschrift	Bevormundende Wirkung — "zu belehrend"	*Schicht 2* — kit_validate_docx.js Prüfung 17 (§3.1)
§0 als Startnummer	Nicht standardmäßige Nummerierung, LaTeX-Erbe	Ausdrückliche Regel General §1.2, künftige statische Prüfung
Zwei aufeinanderfolgende Tabellenfunktionen ohne releaseParagraph	Optisches Verschmelzen der beiden Tabellen in Word	Dokumentiert in General §13.4 — Disziplin des Generators
ImageRun ohne Parameter type	Datei word/media/.undefined, also beschädigte .docx	Behoben durch verpflichtendes style.makeImageRun
require('docx') ohne globalen Pfad	Konflikt zwischen lokaler und globaler Kopie. Abweichung festgestellt bei einem aus einer fremden Umgebung importierten *Stylesheet*, 2026-08-20.	Regel: requireExterne — gewöhnliche Auflösung, dann KIT_NODE_MODULES, dann *npm* root -g; kein Pfad wird im Voraus festgeschrieben. General Reference §1.3, §3.14.2.
***prompt* für Code außerhalb eines Dialogs**	Semantische Verwechslung — *prompt* ist dem *Chat* vorbehalten	Regel: yamlStyle.rawBlock für jeden Code (JS, bash, SVG, HTML, JSON)
disclaimer() als letzter Stil	Bevormundender Ton, ethisch überholt	Entfernt — stattdessen style.tip()
Direkte Bearbeitung des docx-XML (ElementTree)	Beschädigte Namespaces, unlesbare docx	Regel: Neuschreiben von Grund auf als **Node.js**-Generator
Fehlende *Custom Properties*	gen-kit-site.js blockiert das Dokument	Laufzeit — injectCustomProps verpflichtend
Generischer JSON-Extraktor ohne semantische Klassifizierung	Alle 1×1-Tabellen als *prompt* dargestellt, Codeblöcke falsch eingeordnet	Regel: jeder 1×1-Block wird von Hand klassifiziert — rawBlock (Code), codeBlock (YAML) oder *prompt* (nur Dialog)

Anti-Pattern	Folge	Abwehr
Gleiche Breiten bei sehr ungleichen Spalten beibehalten	Platzverschwendung bei Zahlen- oder kurzen Symbolspalten, gestauchter Haupttext	Regel: Die Gleichverteilung als Standard gilt für vergleichbare Spalten, und die Spalte # ist von selbst schmal — General §9. Sonst colWidths nach Inhalt übergeben — Symbole ≈ 5-10 %, langer Text ≈ 30-40 %.
Regex beim Schreiben einer strukturierten Datei	Ein Regex kennt die Grammatik des Formats nicht — er kann einen geschützten Fall treffen und die Datei beschädigen	Regel Kit *Prompt* §4.1: jede Änderung an einer strukturierten Datei läuft über den nativen Parser. Regex nur zum reinen Lesen erlaubt.
Inline-Changelog in vom Kit veröffentlichten Dokumenten	Störende Protokolle, die das Lesen eines Referenzdokuments beeinträchtigen	Redaktionelle Regel: kein Abschnitt “Dokumentverlauf” / kein Inline-Changelog. Einzige Ausnahme: Kit - Projet - Todos.
ASCII-Trennlinien variabler Länge	Uneinheitliche Darstellung, schlechtere Lesbarkeit	Regel Kit *Prompt* §1, Checkliste: einheitliche Trennlinien mit 30 Zeichen, = für die Hauptebene, - für die Nebenebene.
Footer in new Footer({ children: [...] }) eingeschlossen	docx-js serialisiert den inneren Footer als “<options/>” unter <w:ftr>. Word verweigert das Öffnen der Datei (Text Recovery Converter).	*Schicht 1* — ausdrückliches *JSDoc* zu makeFooter / makeCoverFooter. *Schicht 2* — kit_validate_docx.js Prüfung 7.
Stille Synchronisierung zwischen *Cover Sheet* und HTML-*Landing Page* aufgrund vermuteter Drift	Stiller Verlust oder Zusatz: Die beiden Verzeichnisse sind unabhängig — ein *Ordner* kann verschlankt sein, die *Landing Page* lässt weg, was sich nicht online liest.	Die beiden Verzeichnisse werden getrennt deklariert (*Pipeline* HTML §2.5). Kein Skript liest das eine, um das andere zu steuern.
Fest verdrahtete Kit-spezifische Werte in einem stabilen *Artefakt*	Verhindert die Nutzung durch *Anwenderprojekte*	Spezifische Daten aus der maßgeblichen Quelle lesen: .docx über *pandoc-AST*, Projekt-*Registry*, .js-Datendateien.

Anti-Pattern	Folge	Abwehr
require('docx') gibt ein leeres Objekt zurück (beschädigter *npm*-Cache)	Sofortiger TypeError bei jedem new Document(), Packer.toBuffer()	Umgebungsvoraussetzungen — §3.14
require('docx') verschlechterte ESM-Sicht — docx stillschweigend leer	<rootKey>w:p</rootKey> statt echter *OOXML*-Elemente — Word zeigt ein leeres Dokument. Plausibilitätstest §3.14.1 bestanden.	*Schicht 2* — kit_validate_docx.js Prüfung 13; Vorbeugung durch Angleichung der Auflösungen, §3.14.2.
Störendes <0/>, eingeschleust durch einen vergessenen *Spread* bei makeTable	Ungültiges XML-Tag. Word kann es tolerieren, aber *pandoc* HTML scheitert still an der kaputten Tabelle.	*Schicht 2* — kit_validate_docx.js Prüfung 14 (striktes XML global). Blockiert die Auslieferung.
Kaputte Nummerierung durch h2(level, text) statt h2(number, text)	Alle §2.X erscheinen als “2”. Gültiges Dokument, aber unstimmige Nummerierung.	*Schicht 2* — kit_validate_docx.js Prüfung 15 (Monotonie der Überschriften). Blockiert die Auslieferung.
X.1 ohne X.2 — verwaister Unterabschnitt	Dokumentiertes *Anti-Pattern* General §1.2. Einzelner Unterabschnitt, dessen Inhalt unter den übergeordneten Abschnitt gehört.	*Schicht 2* — kit_validate_docx.js Prüfung 16 (verwaistes X.1). Blockiert die Auslieferung.
Isolierter Hinweis oder Tipp unmittelbar nach einer Überschrift	Dokumentiertes *Anti-Pattern* General §1.2. Tipp oder Hinweis als Hauptinhalt verwendet, statt einen vorangehenden Absatz zu unterstreichen.	*Schicht 2* — kit_validate_docx.js Prüfung 17 (isolierter Hinweis oder Tipp). Blockiert die Auslieferung.
***Pipeline*-*Setter* am Anfang von gen-{prefix}-site.js vergessen**	Stille Regression — HTML-Website ohne Glossarlinks, ohne Kapitälchen für Marken, ohne Familien- und externes Routing. Der Generator läuft fehlerfrei.	*Schicht 2* — kit_check_setters.py (§3.4). *Pipeline* HTML §7.9 dokumentiert die Regel. *Linter*-Exit 1, wenn ein erforderlicher *Setter* fehlt.
Verweise first ohne <w:titlePg/> im sectPr	Word ignoriert headers.first und footers.first: Die Deckseite erhält die marineblaue Kopfzeile und die Standardfußzeile. Stiller Fehler, bei der *Erzeugung* unsichtbar.	*Schicht 2* — kit_validate_docx.js Prüfung 18. Korrektes Muster: properties: { ...style.pageProps, titlePage: true } — General Reference §10.4.

Anti-Pattern	Folge	Abwehr
Wiederverwendung der Helper von gen-kit-site.js in einem *Anwenderprojekt*	gen-kit-site.js ist laut Namenskonvention §3.6 dem Kit vorbehalten; seine Helper sind an die Kit-Konstante SECTIONS gebunden. Wiederverwendung = stillschweigende Verdopplung oder Absturz, weil registry.stylesheets im Projekt-*Registry* fehlt.	*Schicht 1* — module.exports von gen-kit-site.js seit 2026-05-02 auf { main } beschränkt. *Anwenderprojekte* schreiben ihren eigenen gen-{prefix}-site.js von Grund auf (*Pipeline* HTML §2.5).
Vergleich von XML-Tags nach Präfix	“w:pPr” spricht auf “w:p” an und bläht die Tiefe auf, die “/w:pPr” nicht verringert. Das Element verschluckt den ganzen Rest des Body, und die Prüfung wird wirkungslos, ohne etwas zu melden.	Regel: verlangen, dass der Tag-Name mit einer spitzen Klammer, einem Leerzeichen oder einem Schrägstrich endet. Helper isTagStart und findTagStart von kit_validate_docx.js.
Exit null einer Prüfung, die nichts geprüft hat	Ein *Linter*, der nichts zu tun findet, gibt null zurück, was sich wie eine bestandene Prüfung liest. Festgestellt beim Ebenen-*Linter* vor seiner Entfernung, dessen fünf kartengesteuerte Generatoren keine Prüfung auslösten.	Regel: “nichts zu prüfen” von “Prüfung bestanden” unterscheiden. Ein Werkzeug, das nichts zu tun gefunden hat, muss es sagen und mit Fehler enden, nie null zurückgeben
Marke für einen fetten Einstieg gehalten	Die *Pipeline* stellt Marken in fetten Kapitälchen dar. Ein Absatz, der mit einem Markennamen beginnt, hat daher einen ersten fetten Run: als boldText extrahiert, verliert er seine Kapitälchen, da boldText von der *Pipeline* ausgenommen ist.	Regel: Ein Run, der fett UND in Kapitälchen ist, ist eine Marke, nie ein Einstieg. Erkennung durch kit_check_markup.py (§3.7).

Anti-Pattern	Folge	Abwehr
Geschützte Leerzeichen des Glossars nach der falschen Sprache neutralisiert	Der aus einer Quelldatei extrahierte Text trägt die vom Glossardurchlauf gesetzten geschützten Leerzeichen. Werden sie nach den Formen einer anderen Sprache neutralisiert, bleibt der Begriff unzugeordnet: Er verliert seine Farbe, ohne dass sich ein Wort ändert.	Regel: die Formen über buildSearchTerms(Sprache des Dokuments) lesen. Erkennung durch kit_check_markup.py (§3.7).

7 Rückmeldeschleife vom Projekt zum Kit

Die Schichten §1.1 sind die internen Verteidigungen des Kits — Prävention zur Laufzeit, *Validatoren* in der Sitzung, übergreifende Audits. Eine weitere Signalquelle gibt es: die Rückmeldung der *Anwenderprojekte* im tatsächlichen Einsatz. Über diesen Kanal kamen historisch die folgenden Korrekturen: Brief Sorso (sectionBanner Glossary v1.18, HTML-*Setter* von gen-kit-site — Eintrag Todos #34), Brief Sliver (verschlechtertes ESM-require — Prüfung 13 hinzugefügt, Eintrag Todos #27), Brief Immobilier (schmale Hinweiszelle — kit_patch_helpers und normatives XML-Schema *General* §6/§7, Einträge Todos #25 und #26), Brief *AI* News (*i18n*-Marker des Cover-Sheet-Renderers — v4, Eintrag Todos #31). Keiner dieser Bugs wäre von den Schichten 1-3 erfasst worden — sie entstehen aus einem konkreten Einsatz außerhalb des Kit-Umfangs.

Ab 2026-05-11 wird dieser informelle Kanal über eine eigene Projektdatei formalisiert: [Préfixe] - Project - Kit *bug report* (TS).docx. Standard-Benennungsmuster Namenskonvention §2.1, Kategorie Project für beide Präfixe offen, §7. Das *Anwenderprojekt* erzeugt den *Bug Report* von Grund auf mit gen-kit-bug-report.js, validiert mit kit_validate_docx und liefert durch Ablage der Quelldatei im *Arbeitsbaum* des Kit-Projekts, begleitet von den nötigen Artefakten (betroffenes gen-*.js, fehlerhaftes Ergebnis (.docx), gegebenenfalls Bildschirmfotos).

Die vollständige Spezifikation — Auslöser, Benennung, Struktur des Dokuments, Ablauf der Meldung, Nicht-Dauerhaftigkeit der Datei — steht in Gemeinsame Struktur §13. Der vorliegende §7 dupliziert den Vertrag nicht: Er verweist auf die Wahrheitsquelle des Projekts und markiert den Übergang vom informellen zum strukturierten Kanal. Die historischen Briefs bleiben als Referenzen in Todos §4 gültig (Einträge #25, #26, #27, #31, #34).