Kit de documentation

Documentation — Pipeline HTML — FR

1 Introduction

Le site publie une partie des documents du Kit, pas leur totalité: seuls ceux qui sont déclarés au *pipeline* sont convertis en pages. Les autres restent dans le classeur imprimé et n'apparaissent nulle part en ligne, y compris après une mise à jour récente.

La conversion part des fichiers .docx produits par le *stylesheet* des documents en prose, en extrait la structure, et la rend en pages HTML avec leur navigation, leur glossaire et leur page d'accueil. Chaque publication produit une archive prête au transfert.

Le tableau ci-dessous indique ce qui s'applique intégralement dans tout projet et ce qui dépend du projet, à renseigner lors de l'initialisation.

Portée	Section	Valeur projet
INVARIANT	§2.1 Critères de sélection des documents publiés	—
INVARIANT	§2.3 Règles de livraison — outils, logique delta	—
INVARIANT	§3 *Registry.js* — structure, usage, règle de mise à jour	—
INVARIANT	§4 *Déploiement delta* ZIP — logique, nommage, lastDeploy	—
INVARIANT	§5 Structure du site HTML	—
INVARIANT	§6 *Génération* PDF	—
INVARIANT	§7 Règles de robustesse	—
PROJET	§2.2 Rubriques et documents publiés	Liste des documents du projet, slugs HTML, activation PDF. Définir dans *Registry.js* section documents.
PROJET	§4.3 Nommage des ZIP	Remplacer le domaine du Kit par celui du *sous-site* du projet.

2 Périmètre de publication

Tous les documents du Kit ne sont pas publiés sur le site. La sélection est volontairement restreinte aux documents destinés à être lus par un utilisateur actif du Kit. Les fichiers de configuration interne — prompts, stylesheets, constantes *PCL* — sont exclus de la publication directe: ils restent listés à titre indicatif.

2.1 Critères de sélection

- Le document a une valeur de lecture autonome pour un utilisateur du Kit.
- Le document n'expose pas de configuration interne du Kit.
- Le document est stable — pas un brouillon ni un document de travail.

2.2 Rubriques et documents publiés

Le site est organisé en rubriques numérotées. Pour le Kit, le *classeur papier* reprend les mêmes rubriques, avec les mêmes titres et les mêmes documents — voir §2.5. Les feuilles de style n'y figurent pas: leurs documents se consultent dans l'archive téléchargeable.

#	Titre	Documents	Fichier HTML	PDF
01	Avant de commencer	Kit — Documentation — Comment documenter ses projets	comment-documenter-LANG.html	oui
		Kit — Documentation — Reading Guide	reading-guide-LANG.html	oui
02	Travailler avec **Claude AI**	**Claude AI** — Documentation — Manuel	claude-manuel-LANG.html	oui
		Claude AI — Documentation — Guide Pratique	claude-guide-pratique-LANG.html	oui
		Claude AI — Documentation — Environnement	claude-environnement-LANG.html	oui
03	Travailler avec *Grok AI*	*Grok AI* — Documentation — Manuel	grok-manuel-LANG.html	oui
		Grok AI — Documentation — Guide Pratique	grok-guide-pratique-LANG.html	oui
		Grok AI — Documentation — Environnement	grok-environnement-LANG.html	oui

#	Titre	Documents	Fichier HTML	PDF
		Grok AI — Documentation — Prompts de dialogue	grok-prompts-de-dialogue-LANG.html	oui
04	Travailler avec **ChatGPT AI**	**ChatGPT AI** — Documentation — Guide Pratique	chatgpt-guide-pratique-LANG.html	oui
		ChatGPT AI — Documentation — Environnement	chatgpt-environnement-LANG.html	oui
05	Lancer et maintenir un projet	Kit — Projet — Guide d'initialisation	guide-initialisation-LANG.html	oui
		Kit — Projet — Guide de mise à jour	guide-maj-LANG.html	oui
		Kit — Projet — Convention de nommage	convention-nommage.html	oui
06	Diriger l'*intelligence artificielle* et adapter le Kit à vos besoins	Kit — Documentation — Prompts de dialogue	prompts-de-dialogue-LANG.html	oui
		Kit — Documentation — Étendre le Kit	etendre-le-kit-LANG.html	oui
07	Fichiers et structure	Kit — Documentation — Environnement d'exécution	environnement-execution-LANG.html	oui
		Kit — Projet — *Prompt*	kit-prompt-LANG.html	oui
		Kit — Projet — Structure commune	structure-commune-LANG.html	oui

#	Titre	Documents	Fichier HTML	PDF
		Kit — Documentation — *Pipeline* HTML	pipeline-html-LANG.html	oui
		Kit — Documentation — Quality Control	quality-control-LANG.html	oui
		Kit — Glossaire — Termes	glossaire-LANG.html	oui
08	Publier et partager un projet	Kit — Documentation — Publication et reprise	publication-reprise-LANG.html	oui

Note: *LANG désigne le suffixe de langue: comment-documenter-fr.html, comment-documenter-de.html, comment-documenter-en.html. Un document non traduit porte un slug sans suffixe.*

2.3 Règles de livraison

Chaque *génération* produit soit une passe complète — premier déploiement ou lastDeploy nul — soit une *passe delta* pour tous les déploiements suivants. La logique est pilotée par le *Registry*. Voir §4 pour le détail.

Fichier produit	Outil	Passe complète	Delta
html/*.html — pages documents	**Node.js**, *AST pandoc*, *stylesheet* HTML	Tous	Seulement les .docx modifiés depuis lastDeploy
glossaire.html	Helper Kit kit_gen_glossaire_html.js depuis le module de termes, jamais via *AST*	Toujours	Toujours
index.html et index-{langue}.html — chargeurs	**Node.js**, pageChargeur()	Toujours	Toujours
assets/kit-style.css	**Node.js**, getCSS()	Toujours	Si le *stylesheet* HTML a changé depuis lastDeploy

Fichier produit	Outil	Passe complète	Delta
assets/pdf/*.pdf	*LibreOffice* CLI depuis le .docx fichier source	Si le PDF est activé	Avec le .html correspondant
assets/pieces/* — pièces d'annexe	Copie directe depuis la racine du projet, déclarée par la clé piece	Si déclarée	La fiche produit son propre PDF, comme tout document
assets/png/{slug}/*.png	Extrait par *pandoc*	Avec le .html	Avec le .html correspondant
.htaccess	Non produit — déposé par le site parent	Hors périmètre	Hors périmètre
robots.txt	Non produit — déposé par le site parent	Hors périmètre	Hors périmètre

Note: *Le générateur localise chaque .docx source par correspondance partielle sur le nom de fichier. Le motif doit être non ambigu. Les apostrophes dans les noms de fichiers sont toujours des apostrophes droites, jamais typographiques. Règles détaillées: HTML Reference §14.10.*

Un document publié en plusieurs langues produit une page par variante, chacune avec son slug suffixé et sa propre entrée dans Registry.documents. Le delta compare par slug, donc chaque langue se republie indépendamment des autres. La place du sélecteur de langue suit le mode déclaré au *Registry* sous rendering.languageSelector — *Structure commune* §15.

2.4 Page d'accueil — bloc d'introduction

La *landing page* affiche un bloc de citation au-dessus de la liste des rubriques. Ce texte est la source de vérité officielle — le générateur d'index le reprend verbatim dans sa constante dédiée.

```
Ce site vous aide à vous servir du kit de documentation pour documenter vos projets. Une fois appliqué, le kit produit des documents au format Word, des PDF et des pages web. Il sert de mode opératoire à un moteur d'intelligence artificielle, celui de votre choix.
```

Le texte est passé au générateur de landing par le champ de configuration prévu. Si le champ est vide ou absent, le bloc est automatiquement masqué — pas de barre orange orpheline.

La même règle s'applique aux *projets consommateurs*: chaque *Pipeline HTML* projet définit son propre texte en §2.4, repris verbatim par son générateur de site.

2.5 Cover Sheet papier et landing HTML — deux index indépendants

Les deux index — le §2 du document *Cover Sheet* et la constante de sections du générateur de site — sont déclarés séparément. Aucun script ne lit l'un pour piloter l'autre.

Leur concordance est un choix du projet. Le Kit les tient identiques: mêmes rubriques, mêmes documents, dans le même ordre de lecture, le classeur ne portant que la langue de base. Un projet peut aussi composer un classeur allégé, qui ne reprend qu'une partie de ce que publie le site.

La landing HTML reste l'index web: elle enrichit chaque entrée de métadonnées de publication, et ce qui ne se lit pas en ligne en est absent — feuilles de style, *Registry*, *Cover Sheet*, *modules de données*.

L'optique commune — couleurs des rubriques, esprit, clé de lecture — est conservée dans tous les cas.

Note: *Une divergence entre les deux index n'est pas une dérive à corriger par script. La synchronisation silencieuse par hypothèse de dérive est cataloguée anti-patron dans Quality Control §6; pour le Kit, la concordance se vérifie à la lecture.*

Le générateur de site suit le patron gen-{projet}-site.js, un par projet. Le Kit a le sien, distinct de ceux des *projets consommateurs*. Ce n'est pas un *artefact* Kit stable réutilisable: il est par nature spécifique au projet qu'il publie.

Note: *Le générateur de site du Kit n'expose pas d'API publique. Ses helpers internes et ses constantes sont couplés à la structure du Kit et ne sont pas exportés. Les projets consommateurs écrivent leur propre générateur from scratch — conséquence directe du préfixe gen-, Convention de nommage §3.6.*

3 Registry — source de vérité centrale

Le *Registry* est le point unique de configuration du *pipeline*. Il porte les versions de stylesheets, les horodatages de *génération* de chaque document, les flags de rendu, la mention de compilation et le dernier déploiement. Toute information qui pourrait être dérivée par cohérence est lue ici.

3.1 Structure

Le *Registry* exporte un objet dont les sections sont décrites dans le tableau ci-dessous. Le fichier n'a pas d'*horodatage* dans son nom — il est la source des horodatages. La structure du *Registry* d'un *projet consommateur* est légèrement différente et documentée dans *Structure commune* §5: elle ne porte pas de section stylesheets, les projets déléguant le suivi des versions au *Registry* du Kit.

Section	Clés	Contenu
project	docLanguage, documentAuthor, webAuthor, documentSiteBase	Identité et langue du Kit. Pilote la sélection *L10N* des stylesheets et les constantes localisées du *renderer Cover Sheet*. Porte aussi l'auteur des documents et des pages, et la racine des renvois entre documents.
requires	coverSheet, readingGuide, glossary, textHighlights, documentTitles, brands, variableNames, hassEntities, colors	Flags de présence des *modules de données* et fichiers optionnels. Lus par les générateurs pour conditionner require() et *setters*.
stylesheets	general, glossary, yaml, html	Version sémantique, *horodatage* du *couplet* .js et .docx, et pour General le plancher minDocumentVersion sous lequel un document publié est obsolète.
familyDomains	tableau de hostnames	Domaines dont les liens restent capturés dans la *WebView* de l'app *Android*. Lu par setFamilyDomains().
documents	un slug par document	*Horodatage* de la dernière *génération*. Une entrée par variante de langue pour les documents traduits.
rendering	homeHref, languageSelector, coverSheet, toc	Flags pilotant la *génération* de livrables: maison de la page d'accueil, les autres pages ramenant au sommaire de leur langue, place du sélecteur de langue, mode de hauteur des rubriques du *Cover Sheet*, *génération* et masquage de la TOC HTML.

Section	Clés	Contenu
compiledWith	objet par langue ou null	Mention de compilation du pied de page HTML. null ou absent: aucune mention. Ne concerne jamais les .docx.
projectZip	version, filename, updated	Release téléchargeable depuis la landing. Numéro de version selon Structure commune §11.
deploy	siteName, siteInfrastructure, lastDeploy	Nom du site publié, qui nomme l'archive; origine du robots.txt et du .htaccess; *horodatage* ISO du dernier déploiement, écrit exclusivement par le générateur de site.

```
module.exports = {
  project:       { docLanguage: 'FR' },
  requires:      { coverSheet: true, readingGuide: true, glossary: true,
                   brands: true, variableNames: false, hassEntities: false },
  stylesheets: {
    general:  { version: '…', ts: 'AAAA-MM-JJ - HHhMM',
                minDocumentVersion: '…' },
    glossary: { version: '…', ts: 'AAAA-MM-JJ - HHhMM' },
    yaml:     { version: '…', ts: 'AAAA-MM-JJ - HHhMM' },
    html:     { version: '…', ts: 'AAAA-MM-JJ - HHhMM' },
  },
  familyDomains: ['sliver.lu', 'hexi.lu'],
  documents: {
    'reading-guide': { ts: 'AAAA-MM-JJ - HHhMM' },
    // un slug par document Kit ; un slug par variante de langue
  },
  rendering: {
    coverSheet: { sectionHMode: 'FIXED' },   // FIXED | DYNAMIC
    toc:        { generate: true, hidden: false },
  },
  compiledWith: null,                        // objet par langue, ou null
  projectZip:       { version: '…', filename: '…', updated: 'AAAA-MM-JJ' },
  deploy:       { lastDeploy: null },        // ISO 8601 UTC
};
```

3.2 Usage dans les générateurs

Chaque générateur qui utilise des références croisées charge le *Registry* au démarrage et lit les valeurs dynamiquement.

```
const registry = require('./Kit - Registry.js');

const v    = registry.stylesheets.general.version;   // version active
```

```
const ts   = registry.documents['reading-guide'].ts; // horodatage document
const mode = registry.rendering.coverSheet.sectionHMode;
const last = registry.deploy.lastDeploy;             // null ou ISO
```

Note: *Les références aux versions dans le texte courant utilisent des gabarits de chaîne. Un changement de version dans le Registry se propage automatiquement à tous les documents générés.*

3.3 Règle de mise à jour

Le *Registry* est mis à jour manuellement, à l'exception des champs que le générateur de site écrit lui-même.

- **Stylesheet changé de version:** mettre à jour la version et l'*horodatage* du *couplet* dans la section stylesheets.
- **Document régénéré:** mettre à jour l'*horodatage* du slug concerné. Un document traduit a une entrée par variante.
- **Flag de rendu ajusté:** mettre à jour la section rendering. Les renderers lisent ce flag au moment de la *génération* — aucune régénération de document n'est nécessaire pour un simple basculement.
- **Mention de compilation:** mettre à jour compiledWith. Relève d'une décision éditoriale, sans impact sur les documents produits.
- **lastDeploy:** mis à jour automatiquement par le générateur de site après chaque *génération* de ZIP. Ne jamais modifier manuellement.
- **Dates de contenu:** contentTs et contentHash, dans la section documents, sont écrits par le générateur de site. À chaque passe, il prend l'*empreinte* du texte nu du document — titres, paragraphes, items, cellules, notes, conseils, blocs de code et de *prompt*, légendes — sans indentation, couleur, italique, *empreinte* de version ni *horodatage*. *Empreinte* identique: rien ne bouge. *Empreinte* différente ou absente: contentTs prend l'*horodatage* du document. La page historique se date sur contentTs, de sorte qu'un bump de *feuille de style*, qui régénère le parc sans en changer le texte, n'y ajoute aucune ligne. Ne jamais modifier à la main. L'écriture conserve les autres clés de l'entrée — le drapeau PDF, par exemple —, et une entrée dont une valeur porte une accolade ou un crochet sort en avertissement avec son nom plutôt que d'être réécrite.
- **Icônes du sous-site:** les quatre icônes du *sous-site* — favicon.svg, favicon.ico, apple-touch-icon.png et index-icon.svg — vivent à la racine du projet, et la passe les copie dans les ressources du site. Le site parent les dessine, le projet les porte: un *sous-site* repris depuis sa seule archive garde son identité. La passe s'arrête en nommant le fichier qui manque; index-icon.svg n'est attendu que si

rendering.indexIcon le déclare. deploy.siteInfrastructure ne couvre plus que robots.txt et .htaccess. Le dessin d'index-icon.svg se centre dans sa zone d'affichage, verticalement et horizontalement, et l'occupe entièrement: le *stylesheet* centre la boîte de 64 par 64, jamais le contenu du fichier, et un dessin décalé se voit tel quel sur la page.

- **Archive du projet:** quand projectShareable vaut true, la passe produit l'archive de l'arbre entier et la pose sous assets/downloads/, sous le nom déclaré à projectZip.filename. Le ZIP de publication la porte, de sorte que le lien de la page d'accueil et projectZip.downloadUrl résolvent la version annoncée, sans dépôt séparé. Le dossier du site et les archives s'excluent de ce qui est empaqueté.

Note: *Le Registry est livré avec le Kit et replacé dans l'arbre du projet à chaque mise à jour. Pas d'horodatage dans le nom du fichier — c'est lui qui en est la source.*

4 Déploiement delta

Le générateur de site produit un ZIP complet ou un ZIP delta selon l'état de lastDeploy. Un déploiement complet inclut tous les fichiers. Un *déploiement delta* n'inclut que les fichiers modifiés depuis le dernier déploiement.

4.1 Logique delta

La comparaison se fait par *horodatage* local du document, lu dans le *Registry*, converti en ISO 8601 UTC puis comparé à lastDeploy. Un document dont l'*horodatage* est postérieur déclenche la régénération de sa page, de son PDF et de ses images. Les autres sont exclus du ZIP delta.

```
// lastDeploy null = passe complète
const lastDeployIso = (mode === 'full')
  ? null
  : (registry.deploy && registry.deploy.lastDeploy);

function kitTsToIsoDateTime(ts) {
  // 'AAAA-MM-JJ - HHhMM' en heure locale Luxembourg vers ISO 8601 UTC.
  // L'offset local (CET +1h, CEST +2h) est recupere via Intl.DateTimeFormat
  // et soustrait pour obtenir l'UTC reel. Sans cette conversion, toutes
  // les comparaisons seraient decalees de une ou deux heures.
  const m = /^(\d{4})-(\d{2})-(\d{2})\s*-\s*(\d{2})h(\d{2})$/.exec(ts || '');
  if (!m) return null;
  const [, y, mo, d, h, mi] = m;
  const naive = new Date(Date.UTC(+y, +mo - 1, +d, +h, +mi, 0));
  const dtf = new Intl.DateTimeFormat('en-US', {
    timeZone: 'Europe/Luxembourg',
    timeZoneName: 'longOffset',
  });
  const tz = dtf.formatToParts(naive).find(p => p.type === 'timeZoneName');
  const m2 = /GMT([+-])(\d{2}):(\d{2})/.exec(tz ? tz.value : '');
  if (!m2) return naive.toISOString();
  const offsetMin = (m2[1] === '+' ? 1 : -1) * (+m2[2] * 60 + +m2[3]);
  return new Date(naive.getTime() - offsetMin * 60 * 1000).toISOString();
```

```
}

function isNewerThanLastDeploy(ts, lastDeployIso) {
  if (!lastDeployIso) return true;          // passe complete
  const iso = kitTsToIsoDateTime(ts);
  if (!iso) return true;                       // prudence : inclure si TS
incompatible
  return iso > lastDeployIso;
}

// Dans la boucle des documents :
const meta = registry.documents && registry.documents[slug];
if (!meta || !meta.ts) continue;
if (mode !== 'full' && !isNewerThanLastDeploy(meta.ts, lastDeployIso)) {
  console.log('- ' + slug + ' (inchange, exclu du delta)');
  continue;
}
```

Note: *La logique delta a longtemps utilisé la date de modification du fichier sur disque. Dans un environnement où le fichier est réimporté à chaque session, cette date est réécrite et n'indique pas la date réelle de génération: le delta finissait par inclure presque tous les fichiers. La comparaison porte désormais sur l'horodatage du Registry, source canonique en heure locale, avec conversion vers UTC.*

4.2 Cas particulier — la feuille de style

Le fichier CSS du site est régénéré si le *stylesheet* HTML a changé de version depuis le dernier déploiement, par comparaison de son *horodatage* dans le *Registry*. Un *Registry* de *projet consommateur* ne portant pas de section stylesheets, le CSS y est régénéré en passe complète uniquement.

4.3 Convention de nommage des ZIP

Patron: {domaine} (AAAA-MM-JJ - HHhMM).zip. Le ZIP du *sous-site* du Kit porte donc le domaine du Kit. Les *projets consommateurs* remplacent le domaine par celui de leur *sous-site*.

4.4 Mise à jour automatique de lastDeploy

Après *génération* réussie d'un ZIP, le générateur écrit dans lastDeploy l'*horodatage* ISO du moment courant. Cette opération réécrit le *Registry* sur disque.

Note: *Ne jamais modifier manuellement lastDeploy. Pour forcer un déploiement complet, mettre la valeur à null et régénérer: le générateur détecte null et bascule en passe complète.*

5 Structure du site

Tous les fichiers HTML de pages documents sont placés dans un sous-dossier dédié. Seule la *landing page* reste à la racine du *sous-site*. Cette séparation rend la hiérarchie lisible et les chemins relatifs uniformes depuis tous les documents.

5.1 Hiérarchie des fichiers

Chemin	Contenu
[*sous-site*]/	Racine du *sous-site*
index.html, index-{langue}.html	Chargeurs vers les sommaires. Seuls fichiers que le Kit écrit à la racine; ils ne portent aucun contenu.
html/	Dossier de toutes les pages documents et des sommaires par langue
html/[slug].html	Page document — un fichier par document publié
html/[slug]-[lang].html	Variante de langue d'un document traduit — Convention de nommage §2.4
html/glossaire.html	Glossaire HTML — généré par le helper Kit depuis le module de termes, jamais via *AST*
assets/	Ressources partagées
assets/kit-style.css	*Feuille de style* HTML, générée depuis le *stylesheet*
assets/favicon.svg	Icône principale. Portée par le projet — Kit - Projet - *Prompt* §16
assets/favicon.ico	Repli multi-tailles. Porté par le projet — Kit - Projet - *Prompt* §16
assets/apple-touch-icon.png	Icône d'écran d'accueil *iOS*. Portée par le projet — Kit - Projet - *Prompt* §16
assets/index-icon.svg	Illustration de la page d'accueil. Portée par le projet — Kit - Projet - *Prompt* §16
assets/downloads/	Archive du projet offerte au téléchargement depuis la page d'accueil. Dossier présent quand projectShareable vaut true au *Registry*: la passe y produit l'archive, sous le nom déclaré à projectZip.filename, et le ZIP de publication la porte.
assets/pieces/	Pièces d'annexe déposées par le projet, sous leur nom d'origine — jamais le slug, jamais un fichier produit. *Pipeline* HTML §6.3
assets/pdf/	PDF générés depuis les .docx — un par document autorisé

Chemin	Contenu
assets/png/{slug}/	Images extraites par *pandoc* — un dossier par document
robots.txt	Directives d'exploration. Déposé par le site parent — Kit - Projet - *Prompt* §16
.htaccess	Configuration *Apache* du *sous-site*. Déposé par le site parent — Kit - Projet - *Prompt* §16

5.2 Chemins relatifs

Tous les fichiers HTML du sous-dossier utilisent des chemins relatifs remontant d'un niveau: la *landing page*, la *feuille de style*, les images et les PDF sont tous atteints en remontant au parent.

Note: *Ne jamais utiliser un chemin relatif au dossier courant pour atteindre la landing page depuis une page document: il pointerait vers un fichier inexistant dans le sous-dossier. Toujours remonter d'un niveau.*

5.3 Structure du ZIP de déploiement

Le ZIP reflète exactement la hiérarchie ci-dessus. L'extraction sur le serveur reconstitue la structure attendue sans manipulation manuelle.

```
kit.sliver.lu (AAAA-MM-JJ - HHhMM).zip
├── index.html
├── index-fr.html
├── index-de.html
├── html/
│   ├── kit-index-fr.html
│   ├── kit-index-de.html
│   ├── reading-guide.html
│   ├── manuel.html
│   ├── glossaire.html
│   └── ...
└── assets/
    ├── kit-style.css
    ├── pdf/
    │   └── reading-guide.pdf
    └── png/
        └── {slug}/
            └── {slug}
```

Un site publie un sommaire par langue, dans html/, nommé {préfixe}-index-{langue}.html — le préfixe vient de deploy.siteName. La langue de base porte son suffixe comme les autres: aucune n'a deux adresses possibles.

Note: *La racine appartient au projet. Le Kit n'y écrit que des chargeurs: un par langue publiée, index-fr.html et ses pairs, plus index.html qui sert la langue déclarée au Registry sous project.docLanguage — jamais celle du navigateur, sans quoi un lien partagé ne mènerait pas au même endroit selon le destinataire. Ces noms sont ceux qu'attend le .htaccess du site parent, dont le bloc de langue teste et sert des chemins absolus*

depuis la racine: c'est à cette condition qu'il reste identique d'un sous-site à l'autre. Un chargeur ne porte aucun contenu et emploie location.replace: un href ou une balise de rafraîchissement écrivent une entrée d'historique, et le lecteur qui revient en arrière depuis le sommaire y serait renvoyé aussitôt.

Note: *Le générateur refuse d'écraser un fichier de la racine qu'il n'a pas produit, et s'arrête en le nommant. Un projet dont la production est un site y met sa propre page d'accueil: elle ne peut pas disparaître sans message. Les fichiers écrits par le Kit portent une marque en tête, qui les distingue de ceux du projet.*

- **Contenu localisé.** Titres de rubrique, libellés de documents, sous-titre, texte d'accueil et titre du sommaire suivent la langue de la page. Les libellés viennent de la table des titres — *Structure commune* §6.8; les titres de rubrique sont déclarés dans le générateur de site.
- **Liens vers la bonne variante.** L'index allemand pointe les pages allemandes. Un document non traduit y figure sous son libellé allemand et pointe la variante existante.
- **Sélecteur de langue.** Chaque index renvoie vers les autres par le sélecteur de langue, placé selon rendering.languageSelector — *Structure commune* §15. La page d'historique porte un nom par langue: historique, verlauf, history.

6 Génération PDF

La *génération* d'un PDF depuis un document Word est possible directement en session via *LibreOffice* en ligne de commande, disponible dans l'environnement. Si le fichier source est dans l'*arbre de travail*, le PDF peut être produit sans intervention côté utilisateur.

6.1 Commande de référence

```
libreoffice --headless --convert-to pdf document.docx --outdir ./

# Exemple avec chemin complet :
libreoffice --headless \
  --convert-to pdf \
  'Kit - Documentation - Pipeline HTML (AAAA-MM-JJ - HHhMM).docx' \
  --outdir /mnt/user-data/outputs/
```

6.2 Intégration dans le générateur de site

Si le PDF est activé pour un document dans le *Registry*, le générateur le produit via *LibreOffice* et l'inclut dans le ZIP sous le dossier des PDF.

Note: *LibreOffice est disponible en session. Ne jamais demander à l'utilisateur de générer les PDF lui-même dès lors que le fichier source du document est dans l'arbre de travail. Cette capacité est toujours disponible, sans condition.*

6.3 Annexes et manuels — PDF externe

Les documents des catégories Annexe et Manuel d'un *projet consommateur* portent un PDF source qui n'est jamais produit par *LibreOffice*. Le PDF est déposé par l'utilisateur dans l'*arbre du projet* avec le même nom de base que le document couplé. Le générateur le copie tel quel — aucune conversion n'est appelée pour ces documents.

Règle d'appariement par nom de base: pour chaque document de ces catégories, le générateur cherche un PDF dont le nom est identique au document privé de son segment d'*horodatage* et de son extension. Correspondance exacte, sans tolérance.

- **Couplet:** le document porte un *horodatage*, le PDF n'en porte jamais — c'est un fichier source figé, déposé tel quel.
- **Slug HTML:** préfixé par la catégorie pour éviter toute collision si deux documents homonymes existent dans deux catégories différentes. Le PDF correspondant suit la même règle.
- **Comportement du pipeline:** copie directe du PDF vers le dossier des PDF; le document suit la chaîne normale de *génération* HTML et de validation.
- **Échec bruyant sur absence:** si le PDF attendu est absent de l'arbre au moment de la *génération*, le générateur s'arrête et nomme le fichier manquant. Pas de repli silencieux sur un PDF généré, pas de page sans icône.
- **Rubriques de la landing:** les rubriques permanentes Annexes et Manuels apparaissent en queue de la landing du *sous-site*, après les rubriques numérotées. Chacune est masquée si elle ne contient aucun document. Ordre interne alphabétique.
- **Genre du document couplé:** le document qui accompagne la pièce est un document de constat — *Kit Prompt* §12. Il rend compte de la pièce sans la recopier, signale nommément ce qui cloche, et ferme sur la section des points relevés. Le §6.3 règle la mécanique; le genre du texte se lit au *Prompt*.
- **La fiche se publie, le bouton sert la pièce:** la fiche d'annexe se publie comme toute autre page, et son bouton de téléchargement rend la pièce d'origine, jamais le PDF tiré d'elle. La pièce se déclare au *Registry*, par la clé piece de l'entrée du document, et vit à la racine du projet; la passe la copie sous assets/pieces/, sous son nom d'origine et jamais sous le slug, et s'arrête en la nommant si elle manque. Le dossier sépare ce qui est déposé de ce qui est produit: une pièce et le PDF d'une fiche ne peuvent pas se heurter.
- **Le PDF de la fiche reste produit:** il entre dans le ZIP du site et dans l'archive du projet, pour le classeur, et aucune page n'y renvoie: le

lecteur n’a ainsi jamais à choisir entre deux PDF dont un seul fait foi. Le bouton ne change ni d’icône ni d’étiquette selon la page — c’est la note en tête de la fiche qui dit ce qu’il rend.

Note: *Le drapeau PDF vaut pour une fiche d’annexe comme pour tout autre document: son PDF se produit depuis son .docx et reste dans assets/pdf, sans qu’aucune page y renvoie. La pièce, elle, est servie à part, depuis assets/pieces.*

Patron de référence pour le générateur de site: détection par catégorie, appariement par nom de base, copie du PDF source.

```
// gen-{prefix}-site.js — extrait pour une fiche d'annexe
const fs   = require('fs');
const path = require('path');

// La piece se declare au Registry, par la cle piece de l'entree du
// document ; elle vit a la racine du projet. Aucun appariement par nom
// de base : le Registry dit le fichier, et lui seul.
function pieceDeclaree(slug, registry, projectDir) {
  const meta = (registry.documents || {})[slug] || {};
  if (!meta.piece) return null;
  const src = path.join(projectDir, meta.piece);
  if (!fs.existsSync(src)) {
    throw new Error('Piece d\'annexe manquante : ' + meta.piece);
  }
  return src;
}

// Dans la boucle de generation de chaque document :
const src = pieceDeclaree(slug, registry, PROJECT_DIR);
if (src) {
  fs.mkdirSync(path.join(OUTPUT_DIR, 'assets/pieces'), { recursive: true });
  fs.copyFileSync(src,         path.join(OUTPUT_DIR,         'assets/pieces',
path.basename(src)));
  // pdfHref de renderDocument pointe la piece, jamais le PDF de la fiche
}
// Le .docx suit la chaine ordinaire : page HTML, et PDF si le drapeau
// du document l'autorise. Ce PDF reste dans assets/pdf, sans lien.
```

7 Règles de robustesse

Cette section documente les pièges les plus dangereux du *pipeline* et les règles impératives qui en découlent. Chaque règle est issue d'une erreur réelle ayant provoqué une perte de contenu ou un rendu corrompu.

7.1 Piège doc.paragraphs — perte silencieuse des tables

Si un script de mise à jour lit un document existant avec une bibliothèque *Python* et itère sur ses paragraphes pour le reconstruire, toutes les tables disparaissent silencieusement. Le document produit est syntaxiquement valide mais amputé de son contenu tabulaire.

Mesure sur un document réel: 300 paragraphes dans le XML, dont 84 seulement visibles par l'*itération* — 28 pour cent. Les 216 autres vivaient

dans les tables, et les 18 tables ont toutes été perdues sans le moindre avertissement.

Note: *Jamais de mise à jour d'un document existant par itération sur ses paragraphes. Toujours recréer le script générateur from scratch depuis le fichier source. Le fichier source vit dans l'arbre de travail pour réécrire le script, jamais pour éditer le XML directement.*

7.2 Échappement sur les chemins

La fonction d'échappement du *stylesheet* HTML protège les caractères spéciaux pour le rendu du texte courant. Elle ne doit jamais être appliquée sur un attribut de chemin.

- **Interdit:** échapper une source d'image, une cible de lien ou un chemin de fichier — cela corrompt le chemin.
- **Autorisé:** échapper un texte de cellule ou un contenu de paragraphe — texte courant uniquement.

Note: *Échapper un chemin d'image produit un attribut corrompu: l'image devient inaccessible sans message d'erreur explicite dans le HTML.*

7.3 Lectures préalables propres au pipeline HTML

La règle générale de lecture obligatoire des Reference avant *génération* est énoncée dans *Prompt* §4.1 et §5.1. Elle s'applique ici sans exception. Les lectures suivantes sont spécifiquement requises avant d'écrire la première ligne d'un convertisseur *AST* ou d'un générateur de site.

- **HTML Reference:** intégralement, en particulier les contrats de fonctions et le *pipeline* de conversion *AST*.
- **General Reference:** §1.2 pour la structure de base d'un document et §13 pour les contrats de retour — le convertisseur produit du HTML mais raisonne sur des structures issues du *modèle* .docx.

Règle d'arrêt strict: si l'un de ces documents n'a pas été lu dans la session courante, arrêt complet. Ne pas poser de question, ne pas supposer, ne pas continuer.

7.4 Validation du document source avant conversion

Un document ne rentre dans le *pipeline* HTML qu'après avoir passé la chaîne de validation. Le contrat détaillé du *validateur* — liste des contrôles, usage, codes de retour — est dans *Quality Control* §3.1. Les conditions ci-dessous sont propres au *pipeline* et vérifiées à l'entrée. L'*empreinte* et la version ne posent pas la même question: l'*empreinte* dit si le document a été produit par le Kit, le plancher s'il rend encore fidèlement.

Condition	Attendu	Conséquence si absent
Empreinte **présente**	StylesheetVersion et GeneratedAt dans les propriétés du document	Absente: le document est rejeté à la conversion
Version à jour	StylesheetVersion supérieure ou égale à minDocumentVersion, le plancher déclaré au *Registry*	Inférieure: le document est obsolète et bloqué
Validateur **passé**	kit_validate_docx.js exit 0 sur le .docx source	Un document non validé ne doit jamais entrer dans le *pipeline*

Note: *Le générateur de site refuse tout document dépourvu d'empreinte ou dont la version est inférieure au plancher. Un document publié a donc traversé un moteur de formatage dont le rendu est encore tenu pour fidèle. Bumper le stylesheet ne périme plus rien: le parc peut porter plusieurs versions à la fois, homogène en rendu sans l'être en numéro. Déplacer le plancher est l'acte délibéré qui impose la régénération générale, et ce jugement n'est rattrapé par aucun contrôle.*

7.5 Le niveau d’un titre vient du document

La conversion rend un bloc de titre portant son niveau, et le générateur le lit tel quel. Cela tient à la chaîne d’héritage des styles, rétablie en General v1.94 par la déclaration du style Normal: sans elle, un convertisseur qui résout l’héritage ne reconnaissait aucun titre, et tout ce qui suivait restait au niveau du titre précédent.

Le texte d’un titre ne décide de rien. Un paragraphe qui commence par un numéro reste un paragraphe; un titre dont le texte commence par un chiffre reste un titre, ce que le *Kit Prompt* §12 proscrit par ailleurs pour la lecture.

Note: *Un document régénéré avec une feuille antérieure à General v1.94 ne déclare pas le style Normal: ses titres retombent en paragraphes à la conversion, et le niveau de tout ce qui suit tombe avec eux. Le parc se régénère donc avant sa prochaine publication.*

7.6 Paragraphe à introduction grasse — le séparateur reste

Quand le convertisseur *AST* rencontre un paragraphe commençant par du gras, il détecte une introduction grasse et appelle le helper correspondant avec le texte gras et le reste de la phrase.

Le reste part tel quel. Le séparateur qui suit l’introduction — deux-points, tiret cadratin, tiret simple — appartient au document: il se lit sur le papier et doit se lire sur la page. Le *stylesheet* HTML n’en pose aucun de son côté, donc rien ne fait double emploi.

Règle: aucun retrait en tête du reste de la phrase, ni séparateur ni espace. Un retrait coûte un caractère que la garde de conservation du texte réclame ensuite, et la publication s’arrête sur une perte venue du générateur et non du document. Cette règle s’applique à tout convertisseur *AST* recréé from scratch.

7.7 Hyperliens famille et externes

Le générateur HTML doit distinguer les liens vers les domaines de la famille et leurs sous-domaines, qui restent capturés dans la *WebView* de l'application mobile, des liens externes qui doivent ouvrir dans le navigateur système. Cette distinction est portée par un helper interne du *stylesheet* HTML, qui ajoute les attributs d'ouverture externe uniquement sur les seconds.

Configuration: la liste des domaines famille est déclarée dans le *Registry*, et chaque générateur de site appelle le *setter* correspondant en tête. Si le *setter* n'est jamais appelé, tous les liens absolus sont traités comme externes.

Mécanique de correspondance. Les URL relatives et les ancres sont famille par définition. Les schémas non-http — courriel, téléphone — sont famille par définition, la gestion revenant au système. Les URL absolues font l'objet d'une extraction du nom d'hôte et d'une comparaison avec la liste: correspondance exacte ou sous-domaine. Les liens croisés entre sites de la famille restent donc dans la *WebView*.

Une classe CSS est posée comme rail sur les liens externes, déclarée vide volontairement. Elle permet à un projet de styler les externes différemment sans toucher au *stylesheet*. Le helper est appelé à la détection des URL dans le texte courant et pour les cellules de tableau porteuses d'un lien.

Note: *Dépendance à l'application mobile. Les attributs d'ouverture externe ne sont efficaces que si l'application implémente le routage correspondant dans son composant WebView. Sans cela, ils sont silencieusement ignorés et les liens externes deviennent morts.*

7.8 Blocs d’une colonne à la conversion

Un bloc de code ou de dialogue peut ressortir de la conversion avec sa première ligne dans l’en-tête de sa table. Le générateur lit donc toutes les lignes pour ces blocs, et prend la première de cet ensemble pour la note et le conseil.

La séparation en-tête et corps ne vaut plus que pour la table ordinaire. Lue sur le seul corps, la première ligne d’un bloc se perdait, et la garde de conservation arrêtait la passe en nommant un fragment introuvable.

Note: *Dépendance à l'application mobile. Les attributs d'ouverture externe ne sont efficaces que si l'application implémente le routage correspondant dans son composant WebView. Sans cela, ils sont silencieusement ignorés et les liens externes deviennent morts.*

7.9 Setters obligatoires en tête du générateur de site

Tout générateur de site HTML doit appeler une série de *setters* en tête, avant la première construction d'élément. L'absence d'un *setter* requis désactive silencieusement le *pipeline* correspondant: le générateur tourne sans erreur mais produit un site sémantiquement amputé.

#	Setter	Source	Effet si absent
1	setLanguage	Registry.project.docLanguage	Défaut EN — toutes les chaînes localisées en anglais
2	setFamilyDomains	Registry.familyDomains	Tous les liens absolus traités comme externes
3	setGlossaryTerms	module de termes du projet	Aucun lien glossaire dans aucune page — site sémantiquement amputé
4	setBrands	module de marques du projet	Aucune marque rendue en petites capitales
5	setVariableNames	module de variables du projet	Aucun nom de variable rendu en italique vert
6	setHassEntities	module d’entités du projet	Aucune entité rendue en italique violet
7	setGlossaryHref	slug du glossaire de la langue rendue	Tous les liens de terme visent la même page — un lecteur non francophone reçoit les définitions de la langue du projet
8	setDocumentTitles	[Préfixe] - Document Titles (TS).js — titleEntries et categoryLabels	Libellés figés dans la langue du projet, aucun renvoi de document transformé en hyperlien, sous-titres de page de garde non composés
9	setLanguageSelector	Registry.rendering.languageSelector	Défaut “document”: icônes au niveau du document au lieu du menu de barre

#	Setter	Source	Effet si absent
10	setTextHighlights	Module Text Highlights du projet	Aucun fragment déclaré mis en évidence

Les *setters* de données sont conditionnés par les flags du *Registry* — un projet sans glossaire n'appelle pas le *setter* correspondant, le module n'existant pas. setLanguage et setFamilyDomains sont universels et toujours requis. setGlossaryHref ne concerne que les projets publiant leur glossaire en plusieurs langues: sa valeur par défaut convient à un projet monolingue, et il s'appelle une fois par document rendu puisque la cible change avec la langue du document.

Le générateur transmet également au *stylesheet* la mention de compilation lue dans le *Registry*, et pour chaque document traduit la liste de ses langues publiées. Cette liste se déclare dans le générateur, jamais dans le *Registry*: le périmètre de traduction est une décision éditoriale par document, pas un drapeau stable de projet.

```
// Patron canonique en tete de gen-{prefix}-site.js — ordre recommande
const registry = require('./[Prefixe] - Registry.js');
const style    = require('./Kit - Stylesheet - HTML - Code (TS).js');

// 1. Langue active — depuis Registry
style.setLanguage(registry.project.docLanguage || 'EN');

// 2. Domaines famille — toujours appele, meme si la liste est vide
style.setFamilyDomains(registry.familyDomains || []);

// 3 a 6. Setters conditionnels selon Registry.requires
if (registry.requires && registry.requires.glossary) {
  const { glossarySearchTerms } = require('./[Prefixe] - Glossary -
Terms.js');
  style.setGlossaryTerms(glossarySearchTerms);
}
if (registry.requires && registry.requires.brands) {
  const { brandEntries } = require('./[Prefixe] - Brands.js');
  style.setBrands(brandEntries);
}
// idem variableNames et hassEntities
```

Note: *Le linter kit_check_setters.py vérifie statiquement que tous les setters requis sont effectivement appelés. Il s'exécute avant chaque génération de site. L'absence d'un setter étant silencieuse à l'exécution, ce linter est le seul filet automatique — voir Quality Control §3.4.*