Documentation Kit

Project — Naming Convention

1 Introduction

All *project files* follow a single naming convention: Kit files, project-specific files, deliverables produced in session and raw data exports. Consistent naming gives predictable sorting, version traceability and reliable context verification from one session to the next.

The project uses a single naming convention for all files. Each file name consists of a prefix, a category, a subject, a language tag for any document, and a *timestamp* suffix that records the date of creation or last regeneration; the exceptions are named in §2.1.

The convention rests on fundamental principles: spaces instead of underscores, sentence case for subjects, hyphens only in the date segment and as separators between structural segments, and a clear separation between prefix, category, subject and *timestamp*.

Note: *The **Claude AI** project interface replaces spaces with underscores when displaying file names. This is platform behaviour — the canonical name always uses spaces.*

2 Naming pattern

2.1 Pattern

Pattern for a document: Prefix - Category - Subject - [Document language] (YYYY-MM-DD - HHhMM).ext

Pattern for a file without content language — *stylesheet*, *data module*, script: Prefix - Category - Subject (YYYY-MM-DD - HHhMM).ext. Language tag: §2.4.

The *timestamp* is in parentheses and uses a 24-hour clock with the letter h separating hours and minutes. The time is always expressed in Luxembourg local time, CET in winter and CEST in summer.

Note: *No underscores anywhere. Hyphens separate structural segments. The subject uses sentence case with natural spacing.*

This pattern applies to all file types, except the stable executable artifacts documented in §3.3, Unix-style tools of the kit_*.py and kit_*.js type, and the *ad hoc generators* documented in §3.5. *Stylesheet* .js files and all other parsed content follow the pattern. The *timestamp* on .js files makes it possible to verify the correspondence between a *stylesheet* and its Word reference document.

- **No category, no timestamp:** [Prefix] - *Registry.js*, source of the timestamps, and [Prefix] - Settings.js — *Common Structure* §5.4.
- **No category, with timestamp:** the *data modules* Brands, Text Highlights, Document Titles, Colors, Variable Friendly Names and HASS Entities, for example Kit - Brands (TS).js.
- **Name imposed by the tool that reads it:** package.json and package-lock.json, which *npm* reads under that name, and the site icons — favicon.svg, favicon.ico, apple-touch-icon.png, index-icon.svg.
- **No timestamp:** the piece of an annex or a manual, [Prefix] - {Annexe| Manuel} - [Name].pdf — §8 and *HTML Pipeline* §6.3.

- **Linked photo, no timestamp:** assets/png/{slug}/{slug}-{section}-{n}.jpg or .png. The document designates it by that name: a replaced photo keeps it — *General Reference* §13.5, *HTML Reference* §14.7. Its variant in a language, for a translation, carries the same name followed by the language in lower case: {slug}-{section}-{n}-de.jpg.
- **Date first:** the session deliverables — data exports, scripts, inventories — §4.1 to §4.3.
- **Archives:** the site archive and the project archive follow their own pattern — §4.5.
- **Critical rule:** when a file is regenerated, the *timestamp* in the file name must be updated to the current time of regeneration. Never keep the old *timestamp* on a file whose content has changed, save the cover image rendered alone, which takes that of its .docx — *Prompt* §14.

Note: *When several related files are regenerated together, for example a stylesheet .js and its reference .docx, they must receive the same new timestamp to stay synchronised.*

2.2 Segments

- **Prefix:** identifies which scope the file belongs to. The system's prefixes are described in §3.
- **Category:** a functional grouping that controls the alphabetical sort order. Categories are fixed and listed in §7.
- **Subject:** a concise description of the file's content, written in sentence case. Multi-word subjects use natural spacing.
- **Language tag:** two-letter upper-case code identifying the language in which the document is written. Every document carries it, whether it exists in one language or several — see §2.4.
- **Timestamp:** the date and time of creation or last regeneration, in the parenthesised format.
- **Extension:** the file type. Content documents use .docx, stylesheets and scripts use .js, raw data files use .txt, delivery archives use .zip — §4.5.

2.3 Examples

File name	Prefix	Category	Subject
Kit - *Stylesheet* - General - Code (2026-03-05 - 08 h49).js	Kit	*Stylesheet*	General - Code

File name	Prefix	Category	Subject
Kit - *Stylesheet* - General - Reference - FR (2026-03-05 - 08 h49).docx	Kit	*Stylesheet*	General - Reference
Kit - Projet - Prompt - FR (2026-03-05 - 08 h49).docx	Kit	Projet	*Prompt*
Kit - Projet - Convention de nommage - FR (2026-03-05 - 08 h49).docx	Kit	Projet	Convention de nommage
[Prefix] - Project - Prompt - EN (2026-03-05 - 08h49).docx	[Prefix]	Project	*Prompt*
Kit - Documentation - Reading Guide - DE (2026-03-05 - 08h49).docx	Kit	Documentation	Reading Guide, DE variant

2.4 Language tag

The language tag states the language in which a document is written. It also distinguishes the variants of a same document published in several languages, without touching the rest of the name.

It is placed after the subject and before the *timestamp*, in upper case, among the four supported codes: FR, EN, DE, LU. In a pattern it is written [Document language] — [Langue du document] and [Dokumentsprache] in the French and German versions of a Kit document —, and in lower case in a slug: comment-documenter-[document language].html.

Every document carries the tag, whether it exists in a single language or in several. A document translated later thus keeps the name of its original variant: the translation is added alongside, without renaming. The tag can also be read without opening the file, and a reader of another language knows at once in which language the document is written. A file without content language — *stylesheet*, *data module*, *Registry*, script — carries none.

- **Category unchanged:** the category and the subject remain in the project language, defined by project.docLanguage. Only the tag marks the language of the content. The Kit, a French-language project, therefore

names the German variant of its prompt *Kit - Projet - Prompt - DE*, never Kit - Projekt - Prompt - DE.

- **Suffixed HTML slug:** the published page of a translated document carries the same suffix in lower case, reading-guide-de.html. A document published in a single language keeps its bare slug, without suffix, even though its file carries the tag: the slug is a published address, and it only takes its suffix when the document is translated — §3.4. These slugs feed the links of the language selector of the HTML *stylesheet*.
- **One Registry entry per variant:** each language has its own entry in Registry.documents with its *timestamp*. The key of the entry is the slug: that of a document published in a single language stays bare. *Delta deployment* compares by slug, so each language is republished independently of the others.
- **Scope declared in the generator:** the list of published languages of a document is declared in the project's site generator, never in the *Registry*. The translation scope is decided document by document. This rule concerns the document. The scope of the site — the languages it serves, whether a document is translated or not — is a stable flag, and is declared in the *Registry* under project.siteLanguages: *HTML Pipeline* §5.

Source file	Published page	Registry entry
Kit - Documentation - Reading Guide - FR (TS).docx	reading-guide-fr.html	'reading-guide-fr'
Kit - Documentation - Reading Guide - DE (TS).docx	reading-guide-de.html	'reading-guide-de'
Kit - Documentation - Reading Guide - EN (TS).docx	reading-guide-en.html	'reading-guide-en'
[Prefix] - Documentation - [Subject] - EN (TS).docx	[slug].html	'[slug]'

Note: *The glossary is an exception to the pattern: a single terms module feeds several glossary documents, one per language, all with the same timestamp. The tag follows the general rule: kit_gen_glossaire_docx.js sets it on each glossary document, monolingual projects included. The complete contract is in Common Structure §6.1. The Cover Sheet remains a single document, written in the language of the project, whose tag it carries; its .png renderings are produced per published language, and a continuation page inserts " - Page N" before the timestamp — Cover Sheet §4.*

3 Prefixes

Each prefix designates a scope: the Kit and its engines, or the project. Files from different scopes thus coexist in the same view without ambiguity.

3.1 The Kit prefix

The Kit prefix identifies the infrastructure files of the Kit: stylesheets, reference documents, technical prompt, naming convention and initialisation guide. These files are not modified when a new project is deployed. The engine prefixes — **CLAUDE AI**, and that of any engine that joins the Kit — are Kit prefixes: they name its documents specific to an *artificial intelligence* engine, and the governance rule below covers them — slug in §3.4.

- **Governance rule:** the rule and its exception live in *Kit Prompt* §7. In a normal project session, all new files carry the project prefix.

3.2 The project prefix

The project prefix is chosen when the project is initialised. It is short, without special characters, easy to type. All files produced during sessions carry this prefix.

Project	Recommended prefix
[Project name]	[Prefix]

3.3 Stable artifacts — snake_case nomenclature

Stable executable artifacts — Unix-style tools that live permanently in the *working tree*, Kit or project, and whose logic stays stable from session to session — follow a convention distinct from the general pattern. They carry the project prefix in lower case followed by an underscore, kit_ for the base Kit tools and [prefix]_ for those of a *consumer project*, use snake_case for the name, and carry no *timestamp*. Formal exception to the underscore ban stated in §2.

Conforming Kit examples: kit_validate_docx.js, kit_validate_xlsx.py, kit_render_cover_sheet.py, kit_check_setters.py, kit_check_markup.py, kit_check_ossature.py, kit_patch_helpers.py, kit_gen_glossaire_html.js. A project names its own the same way, under its prefix.

Complete operational contract: *Quality Control* §3. Creation recipe for *consumer projects*: *Common Structure* §12. Distinguishing criterion: a stable executable *artifact* follows the snake_case convention, except gen-kit-site.js and gen-glossary-reference.js, which keep their names — §3.5; any other file — *stylesheet*, *data module*, *registry*, document, *ad hoc generator*, source file — follows the pattern of §2 or §3.5 as the case may be.

3.4 Slug of an engine document

A document specific to an *artificial intelligence* engine carries that engine's prefix in its slug, in lower case: claude-manuel-fr, claude-guide-pratique-de, claude-

environnement-en. A bare slug belongs to the Kit itself, whose documents depend on no engine.

The slug is a published address. Renaming it breaks links already shared: the rule applies from the creation of the document, and a renaming is decided as a break, not as a tidy-up. kit_check_registry.py flags an engine document whose slug does not carry its prefix, and a slug prefixed with an engine for a document that is not that engine's.

3.5 Ad hoc generators — gen-* pattern

Ad hoc generators are **NODE.JS** scripts recreated from scratch in each session to produce a .docx deliverable. Unlike the stable artifacts of §3.3, they do not live permanently in the project's *working tree* — they are disposable, their lifetime is that of the current session. An existing document, by contrast, is regenerated from its map by kit_gen_document.js — *Quality Control* §3.16.

Pattern: gen-{slug-document}.js. Examples: gen-cover-sheet.js, gen-pipeline-html.js. No redundant project prefix: each project lives in its own session folder.

Exception: a project's Kit site generator is a durable project file. It is the gen-*-site.js that requires kit_publication.js, one per project; gen-[prefix]-site.js is its recommended name, not a reserved one. A project may also carry another site generator, outside the Kit, that does not require kit_publication.js: the Kit neither reads nor checks it. It is written once per project, lives in the *working tree* and in the archive, and is maintained thereafter: without it, a resumed *subsite* archive does not produce its site — *Publishing and reuse* §8, *Common Structure* §2. The -site suffix distinguishes it from document generators. gen-kit-site.js and gen-glossary-reference.js, for their part, are stable Kit artifacts, delivered with it and never rewritten in session — *Quality Control* §3. They keep their name, under which documents and tools call them.

Note: *No timestamp in a generator's name — its TS is that of the artifact it produces, read via getLuxTimestamp() in the generator itself. Convention aligned with Kit practice and with project consistency: each session starts again from scratch, the history of document generators is not kept.*

Correct examples: gen-cover-sheet.js, gen-pipeline-html.js, gen-structure-commune.js, and for a project's site gen-[prefix]-site.js.

Anti-patterns: redundant project prefix, for example gen-[prefix]-prompt.js where gen-prompt.js suffices. Mixing a short prefix and the full project in the same session — choose one convention and stick to it.

4 Names of session deliverables

In addition to the main *project files*, the *AI* produces session deliverables that use a date-and-version pattern. These names appear in the project file inventory and make it possible to follow the evolution of documents between sessions.

4.1 Data collection files

Pattern: YYYY-MM-DD - Collecte de données - [SOURCE] - [description].txt

The source system appears as the second placeholder, allowing all data files to sort together alphabetically under “Collecte de données”.

Examples: 2026-03-05 - Collecte de données - *pandoc* - version.txt and 2026-03-05 - Collecte de données - *npm* - modules.txt.

4.2 Script files

Pattern: YYYY-MM-DD - Script - *Python* - [name].py

Python scripts used for data collection are stored in the project's *working tree*.

4.3 Project file inventory

Pattern: YYYY-MM-DD - Inventaire fichiers projet - Version 1.xx.txt

The inventory is a plain text file listing every file that must be present in the project's *working tree*. It is one of the end-of-session deliverables, produced on the user's go — *Prompt* §10.2 —, and is self-referencing.

4.4 Session summary

Pattern: [Prefix] - Projet - Résumé de session - [Document language] (YYYY-MM-DD - HHhMM).docx

The session summary is a Word document formatted according to the general *stylesheet*, produced at the end of the session on the user's go — *Prompt* §10.2; its structure is that of *Prompt* §10.3. Unlike the other session deliverables, it follows the main pattern of §2.1.

4.5 Delivery archives

A session delivers its work in archives, never file by file — *Common Structure* §11. Their names follow distinct rules: the archive of the published site carries the site's name and a *timestamp*, whatever the extent of the pass, since it always contains the entire site; the project archive carries a version number.

Archive	Pattern	Origin of the name
Published site	{siteName} (TS).zip	Registry.deploy.siteName
Project archive	{name}-{version}.zip, without *timestamp*	Registry.projectZip.filename, ending in -{version}.zip, version being projectZip.version

A deployment archive carries the name of the domain it goes to, not that of the project that produces it: it is recognised by its destination. A project that publishes several *subsites* produces as many, distinguished by this name.

Note: *The project archive is the only one without a timestamp. Every project produces it: it is the archive the user keeps and gives to the next thread. When shared, its name is also an address — the home page link points to it — and an address that changes with every pass is a dead address. It therefore carries a version number, which goes up with each delivery. An archive*

already produced is never redone under the same number: if it is missing at the start of a pass, it is requested — Common Structure §11.

Note: *The fixed segments of the patterns stay the same in every language: only {siteName}, {name}, {version} and (TS) are substituted. An archive is a container of files. Its name serves the sorting of a backup, where the chronological continuity of a same family matters more than language: a project that switches its documentation language must not see its archives change name and lose their history. §5 therefore does not apply here.*

5 Localised names

File-name categories and subjects are localised according to the project language, defined by the project.docLanguage field in Kit - *Registry.js* for the Kit, or [Prefix] - *Registry.js* for a *consumer project*. The supported languages are EN, default for *consumer projects*, FR, historical language of the Kit, DE and LU.

Consistency rule: within a same project, the categories and subjects of documents follow the declared language, without mixing. Files without content language — stylesheets, *data modules*, *Registry* — keep their English technical name in all languages: generators and tools look for them under that name. The Kit itself stays in FR by documented exception.

This rule concerns the segments of the name, not the language in which the document is written. A French-language project that translates a document into German keeps its French categories and marks the content language with the tag of §2.4.

5.1 Localised categories

Reference table of the fixed categories translated into each language of the Kit.

Category (EN)	FR	DE	LU
Annex	Annexe	Anhang	Annex
Documentation	Documentation	Dokumentation	Dokumentatioun
Glossary	Glossaire	Glossar	Glossär
Manual	Manuel	Handbuch	Handbuch
Project	Projet	Projekt	Projet
Stylesheet	*Stylesheet*	*Stylesheet*	*Stylesheet*

Note: *The Stylesheet category remains identical in every language — an English technical term of universal value, aligned with CSS and Word terminology.*

5.2 Subject and label of Kit documents

The localised labels of Kit documents have their single source in Kit - Document Titles (TS).js, whose titleEntries carries the label of each document in each language — *Common Structure* §6.8. The subject in a file name and the label in a text are kept apart. In a file name, the subject stays that of the project language in every variant — §2.4 —, and stable technical names, such as *Cover Sheet*,

Pipeline HTML, *Quality Control* or Reading Guide, stay invariant there. In a text, a page or a cross-reference, a document is named by its label in the language of the text, as Document Titles carries it. Archive names are invariant — §4.5.

5.3 Examples of full names by language

Project language	Example
EN	[Prefix] - Project - Prompt - EN (2026-04-23 - 14h30).docx
FR	Kit - Projet - Prompt - FR (2026-04-23 - 14h30).docx
DE	[Prefix] - Projekt - Prompt - DE (2026-04-23 - 14h30).docx
LU	[Prefix] - Projet - Prompt - LU (2026-04-23 - 14h30).docx

These examples illustrate the localisation of the name segments. The segments follow the project language, the tag of §2.4 that of the content: the Kit, an FR-language project, names the German variant of its prompt Kit - Projet - Prompt - DE (2026-04-23 - 14h30).docx.

5.4 Renaming on a change of language

When project.docLanguage changes for an active project, all *project files* are renamed according to the categories and subjects of the new language, and all documents are regenerated in that language, with a fresh *timestamp*. The change is made in a single batch, by the *AI*, at the user's request — *Update Guide* §6.

Note: *The Kit itself does not change language: it is written in French, and project.docLanguage is FR in Kit - Registry.js — Kit Prompt §17. Consumer projects start at EN by default.*

6 Version numbering

What carries a version in its name or in the *Registry* — stylesheets, a project's archive, the Kit's archive, the project file inventory, §4.3 — uses the pattern x.yy where x is the major version and yy the minor version. The minor version is incremented with each update; that of an archive once per archive delivered, not with each file changed — *Common Structure* §11. The major version is incremented for a structural overhaul, and after x.99 with no overhaul: x+1.00 then follows. The kit_ tools and a project's own, [prefix]_, are kept through the dated changelog at the head of the file — *Quality Control* §3, *Common Structure* §12; their new entries carry only their date.

7 Categories

Categories are fixed labels that form the second segment of the file name. They control the alphabetical sort order in the project folder and group related files. The table below uses the English names — equivalents in the other languages in §5.1.

Category	Prefix	Function
Annex	Project	Reference PDF documents added manually — reports, technical data sheets, third-party docs. Each PDF is accompanied by a short Kit .docx summarising its content
Documentation	Kit or project	Cross-cutting reference documents: Reading Guide, *Cover Sheet*, HTML *Pipeline*, Quality Control, Dialogue Commands
Glossary	Kit or project	Glossary document, term list, formatting modules
Manual	Project	Original PDF documents from the manufacturer or producer — equipment manuals, datasheets. Each PDF is accompanied by a short Kit .docx describing the hardware and its use
Project	Kit or project	Project management files: Prompt, Naming Convention, Initialisation Guide, Update Guide, Common Structure, session summaries, inventory
Stylesheet	Kit or project	*Stylesheet* .js modules and their Word .docx references

Note: *Adding a new category is a project decision that must be recorded in the project prompt.*

Note: *For Stylesheet files, the subject consists of two parts separated by a hyphen: the qualifier — General, Glossary, YAML, HTML — and the type, Code for .js files and Reference for .docx files. The Reference, a document, carries the language tag; the Code does not.*

8 Quick reference

Context	Pattern	Example
Kit document	Kit - Category - Subject - [Document language] (YYYY-MM-DD - HHhMM).ext	Kit - Projet - Prompt - FR (2026-03-05 - 08h49).docx
Project document	[Prefix] - Category - Subject - [Document language] (YYYY-MM-DD - HHhMM).ext	[Prefix] - Project - Prompt - EN (2026-03-05 - 08h49).docx

Context	Pattern	Example
Translated document	[Prefix] - Category - Subject - [Document language] (YYYY-MM-DD - HHhMM).ext	Kit - Documentation - Reading Guide - DE (2026-08-17 - 13h55).docx
File without language	[Prefix] - Category - Subject (YYYY-MM-DD - HHhMM).ext	Kit - *Stylesheet* - General - Code (2026-03-05 - 08h49).js
Annex or manual sheet and piece	[Prefix] - {Annexe|Manuel} - [Name] - [Document language] (TS).docx and [Prefix] - {Annexe|Manuel} - [Name].pdf	[Prefix] - Annexe - [Name] - FR (2026-05-06 - 14h00).docx
Session summary	[Prefix] - Projet - Résumé de session - [Document language] (YYYY-MM-DD - HHhMM).docx	[Prefix] - Project - Session summary - EN (2026-03-05 - 21h45).docx
Data export	YYYY-MM-DD - Collecte de données - [SOURCE] - [description].txt	2026-03-05 - Collecte de données - *pandoc* - version.txt
Script	YYYY-MM-DD - Script - *Python* - [name].py	2026-03-05 - Script - *Python* - collect.py
Inventory	YYYY-MM-DD - Inventaire fichiers projet - Version 1.xx.txt	2026-03-05 - Inventaire fichiers projet - Version 1.01.txt
Kit *artifact*	kit_subject.ext	kit_check_markup.py
Project *artifact*	[prefix]_subject.ext	[prefix]_validate_[subject].py
Ad hoc generator	gen-{slug-document}.js	gen-pipeline-html.js, gen-cover-sheet.js
HTML site generator	gen-[prefix]-site.js	gen-[prefix]-site.js