Kit de documentation

Project — Initialisation Guide — EN

1 Why the Kit is injected in full

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

This guide is for you, not for the *AI*. It describes how to start a new project from the Kit.

Before the procedure, one important point: the Kit is a coherent system. Its files are not independent bricks to be picked one by one — they depend on each other. Each *stylesheet* has an associated reference document describing how to use it. The *Registry* declares the active versions. The technical *prompt* and the common structure set the normative contract that all your *project files* apply.

Injecting the Kit partially — forgetting a Reference, replacing a *stylesheet* without its *Registry* declaration, leaving an old *prompt* in place — produces silent inconsistencies that only show when a document is generated. The symptom is then a broken layout or, worse, lost content.

Simple rule: the Kit is uploaded in full, as delivered. Every file listed in §3 goes into the *working tree* in one go. No picking and choosing, no file-by-file replacement. This rule holds for initialisation as for later updates.

💡 *An update replaces the Kit's files and leaves yours in place. Your work in progress is preserved as long as you have not modified a Kit file yourself.*

2 The information to prepare

Before opening a session, prepare the following pieces of information.

Information	Example	Note
A short prefix	Arrosage	One word, no special characters. This prefix appears in every file name.
The full project name	Système d'arrosage	The name shown on the cover pages of the documents.
Your name	Norbert Weber	Your name appears in the footer of every document.
The language wanted	FR, EN, DE or LU	French, English, German or Luxembourgish. English by default for *consumer projects*; the Kit stays in French as a documented exception.

3 Creating the project and injecting the Kit

Create a new project. The name is up to you. Then upload into its *context* every Kit file listed below, without exception.

These files will not be modified by your project — they are the same for every project using this Kit. They provide the formatting rules, the permanent instructions and the visual references.

Your project's own files — *Registry*, project *prompt*, glossary, customised *cover sheet*, any *data modules* — are not uploaded at this stage: they are created from scratch during the first session, from the normative skeletons in Common Structure.

Kit file	Role
Kit - *Registry.js*	Source of truth for the active *stylesheet* versions and the timestamps of Kit documents.
Kit - Projet - *Prompt* - LANG (TS).docx	Permanent technical instructions — read at the start of every session.
Kit - Projet - Structure commune - LANG (TS).docx	Normative contract for *project files* and skeletons of the *data modules*.
Kit - Projet - Convention de nommage (TS).docx	Naming rules for *project files*, language tag included.
Kit - Projet - Guide d'initialisation - LANG (TS).docx	This guide.
Kit - Projet - Guide de mise à jour - LANG (TS).docx	Procedure for updating the Kit.
Kit - Documentation - Reading Guide - LANG (TS).docx	Reading guide for the *paper binder*.
Kit - Documentation - *Cover Sheet* (TS).docx	Reference for the *cover page* rendering constants.
Kit - Documentation - *Cover Sheet* - LANG (TS).png	Generated Kit *cover page* — first page of the *binder*.
Kit - Documentation - *Pipeline* HTML - LANG (TS).docx	Site architecture, publication scope, deployment rules.
Kit - Documentation - Quality Control - LANG (TS).docx	Single reference for the quality control mechanisms and session prerequisites.
Kit - Documentation - Prompts de dialogue - LANG (TS).docx	Dialogue commands to copy and paste into the *chat*.
Kit - Documentation - Comment documenter ses projets - LANG (TS).docx	Guidance for the decisions that come before the first line is written.
Kit - Documentation - Étendre le Kit - LANG (TS).docx	How to add your own presentations without changing the Kit’s.

Kit file	Role
Kit - Glossaire - Termes - LANG (TS).docx	Kit glossary — couplet with the terms module.
Kit - Glossary - Terms (TS).js	Terms module of the Kit glossary — renders terms in teal italics in every document.
Kit - Brands (TS).js	Kit brands module — renders brands in bold small caps.
Kit - Text Highlights (TS).js	Module of highlighted fragments — renders entities declared by the project in italics.
Kit - Document Titles (TS).js	Module of display labels — readable name of each document, section titles and category labels, in the reader's language.
Kit - *Stylesheet* - General - Code (TS).js	General *stylesheet* — formatting engine for every document.
Kit - *Stylesheet* - General - Reference (TS).docx	Reference document for the general *stylesheet*.
Kit - *Stylesheet* - Glossary - Code (TS).js	Glossary *stylesheet*.
Kit - *Stylesheet* - Glossary - Reference (TS).docx	Reference document for the glossary *stylesheet*.
Kit - *Stylesheet* - YAML - Code (TS).js	*Stylesheet* for code and YAML blocks.
Kit - *Stylesheet* - YAML - Reference (TS).docx	Reference document for the YAML *stylesheet*.
Kit - *Stylesheet* - HTML - Code (TS).js	Mirror HTML *stylesheet* — produces the sub-site pages.
Kit - *Stylesheet* - HTML - Reference (TS).docx	Reference document for the HTML *stylesheet*.
kit_check_markup.py	Markup check between a source file and the document produced, and agreement check of the note label widths between the artifacts that declare them.
kit_check_ossature.py	Skeleton check between the language variants of a document — block sequence, levels, table dimensions.
kit_extract_map.py	Structure map of an existing document — first step of any regeneration.
kit_gen_document.js	Rebuilds a document from its structure map — *stable artifact*, Quality Control §3.16.

Kit file	Role
kit_check_fidelite.py	Extraction fidelity check — round trip compared to the byte.
kit_check_registry.py	Check of the project *Registry*'s scope — called before any *generation*. A key outside the scope is lost at the next update, with nothing to show for it.
kit_check_setters.py	*Linter* of *setter* discipline — called before each *generation*. Without it, a mandatory rule cannot be applied.
kit_check_couplets.py	Check of the shared *timestamp* of a couplet's members. A missing member is reported without failing.
kit_validate_docx.js	*Validator* for the documents produced — called after each *generation*, before delivery.
kit_validate_xlsx.py	*Validator* for the spreadsheet files produced by the pipelines that create them.
kit_render_cover_sheet.py	Deterministic *renderer* of the *cover page*, driven by the rendering constants of the *Cover Sheet* document.
fonts/	*Poppins* fonts for the cover *renderer*, with their OFL licence.
kit_patch_helpers.py	Fragment-building helpers for *surgical patches* — required to insert a note or a box into an existing document.
kit_gen_glossaire_docx.js	Helper that generates the glossary .docx from the terms module. Counterpart of kit_ gen_glossaire_html.js — one source, two renderings.
kit_gen_glossaire_html.js	Helper that generates the site's glossary page from the terms module. Mandatory in every project that publishes a glossary.
package.json	Declaration of the chain's *npm* modules and their minimum version — fresh installation with *npm* install.

Kit file	Role
favicon.svg, favicon.ico, apple-touch-icon.png, index-icon.svg	Icons of the sub-site, drawn by the parent site and carried by the project. The pass copies them into the site's resources; index-icon.svg is expected only if rendering.indexIcon declares it; its drawing centres itself in its display area and fills it, the *stylesheet* centring the box alone.

Note: *This list is authoritative. Update Guide §3 refers to it rather than duplicating it — a file missing here will never be injected anywhere.*

Note: *The npm modules of the chain — docx, and adm-zip for publishing the site — are not Kit files. package.json declares them with their minimum version; they are always installed fresh with npm install in the Kit folder, never by copying a node_modules — Common Structure §11.*

Note: *The documents in the per-engine sections — manual, practical guide, environment report and dialogue commands for each documented assistant — belong to the paper binder and the site. They are not context files for a consumer project: nothing in the generation depends on them. A project may still keep them in its context if its user wants them to hand.*

Note: *If your project needs its own brands beyond the Kit's, the matching module will be created at initialisation. Likewise for variable names and home-automation entities — optional files created only if you enable them.*

4 Starting the first session

Once every Kit file is in the *working tree*, open a *conversation* and paste the text below as the first message.

That text is an enabler: it forces every source of truth to be read in the right order, an explicit reading report, confirmation of your configuration, and a wait for your go-ahead before a single file is created. The result is coherent because no step can be skipped.

```
SETTING UP A NEW PROJECT — FULL KIT INJECTION
=============================

PROJECT CONFIGURATION :
-----------------------------

Project prefix : [YOUR PREFIX]
Full name      : [PROJECT NAME]
My name        : [YOUR NAME]
Language       : [FR / EN / DE / LU]

MANDATORY READING — STRICT ORDER :
-----------------------------

Before creating any project file, read in this order :

1. Kit - Projet - Prompt (TS).docx
```

```
   — permanent technical instructions.
2. Kit - Stylesheet - General - Code.js + Reference.docx
   — formatting engine for all documents.
3. Kit - Stylesheet - Glossary - Code.js + Reference.docx
   — stylesheet of the project glossary.
4. Kit - Stylesheet - YAML - Code.js + Reference.docx
   — code and YAML blocks in the documentation.
5. Kit - Stylesheet - HTML - Code.js + Reference.docx
   — mirror HTML stylesheet.
6. Kit - Projet - Structure commune (TS).docx
   — normative contract of the project files and skeletons.
7. Kit - Projet - Convention de nommage (TS).docx
   — naming rules.
8. Kit - Documentation - Quality Control (TS).docx
   — mandatory validation chain and session prerequisites.
9. Kit - Projet - Guide d'initialisation (TS).docx
   — this guide.

READING REPORT — ANSWER BEFORE GOING ON :
------------------------------

For each document in the list above :
- read in full : yes / no
- version or timestamp recorded

Then confirm :
- the four configuration items
- the list of optional data modules wanted
  (brands, variable names, home-automation entities,
  Reading Guide, project cover sheet) and their value

STOP CONDITIONS :
------------------------------

If any of these is met, full stop and explicit report
before any creation :

- a Kit file is missing from the working tree
- a version number declared in the Registry does not
  match the actual version of the file
- two files of the same pair carry different timestamps
- a configuration item is missing or ambiguous

AFTER REPORT AND CONFIRMATIONS — WAIT FOR THE GO :
------------------------------

Create no project file before an explicit « go » has been
received in the chat. Announce the full plan first : order
of the files to produce, intended content, sequential
delivery per Kit - Projet - Prompt §9.
```

After the reading report and the confirmation of your optional modules, simply answer “go” in the *chat* to trigger the creation of the *project files*.

5 What happens next

All the technical work is taken care of. You have nothing technical to decide beyond the four pieces of information in §2 and the optional modules confirmed at step 2.

#	What is done	What you do
1	Reads every Kit file in order and confirms its understanding: prefix, name, language.	Check that the configuration was read correctly.
2	Asks which optional modules to enable and whether you want the Reading Guide and the project *cover sheet*.	Answer according to your need. With no view, the defaults document a standard project.
3	Creates the *project files* from scratch out of the Structure commune skeletons: project *Registry*, project *prompt*, glossary, customised *cover sheet*, enabled modules. If the *cover sheet* is wanted, you are asked interactively for the project-specific values — the two header lines, the list of sections, the documents of each section.	Wait.
4	Delivers the files one by one following the *sequential delivery* protocol.	Download each delivered file.
5	Confirms the project is ready and lists the files to upload into the *context*.	Upload the new *project files*.

Note: *The number of cover-sheet sections is not capped at six. The TAB_NUMBERED_MAX value is derived automatically from the number of declared sections; the only limit is the number of tabs, twelve. See Common Structure §8.2.*

6 Working day to day

6.1 Start of session

Each *conversation* stands alone — nothing is retained from the previous session. Continuity comes from the project's files. At the start of each session the *AI* checks that everything is in order before beginning.

If a file is missing, it will tell you. Upload it and carry on.

6.2 End of session

To end a session, simply say:

```
And now to the end of session, please.
```

You will automatically receive a session summary, a *file inventory* and, if needed, an updated project *prompt*. The list of files to upload before the next session is given as well.

💡 *The session summary holds the open questions and the recommendations — it is*

your thread for picking the work up again next session.

6.3 Keeping files current

The most important rule: after each session, upload the updated files delivered to you into the project. Without that step, the next session starts from old files.

Depending on the flags enabled in your *Registry*, not every file in the table below is necessarily present in your project. Upload only those that actually exist.

File	When to upload it
Project *Registry*	As soon as a document *timestamp* or a flag changes.
Updated project *prompt*	As soon as a new convention has been decided in session.
File inventory	After every end of session.
Glossary terms module	As soon as new terms have been added.
Brands module	As soon as a new brand name has been identified — if the matching flag is on.
Variables module	As soon as a new variable name has been identified — if the matching flag is on.
Home-automation entities module	As soon as a new entity has been identified — if the matching flag is on.
Content documents	After each session in which they were modified.

💡 *If in doubt about which files to upload, reread the end of the session summary: the list is explicit there.*

7 The cover page in your documents

Every document this Kit generates begins with a *cover page* carrying the title, subtitle and date. The header and footer are automatically suppressed on that first page for a clean result — they appear only from page 2.

The Kit handles this automatically, with no action from you.

💡 *If a document you receive shows the header on its cover page, report it: that is an error in the generator script, not in Word. The Kit's validator now catches this defect before delivery.*