Kit de documentation

Documentation — Extending the Kit — EN

1 Purpose

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

The Kit covers a finite set of presentations and document types. Sooner or later a project meets a need that is not among them: a missing layout, a particular kind of block, a rule specific to its field.

Two routes exist. Modifying the Kit's files meets the need at once, and forces a choice at the first update between your adaptation and the new version. Adding your own files alongside costs a little more work up front and lets both coexist.

This document describes the second route: where your files belong, how to name them, how a generator combines them with those of the Kit, and what breaks when the Kit evolves.

2 What belongs to the Kit and what belongs to you

The boundary is a prefix. Every file whose name starts with “Kit -” or “kit_” comes from the Kit and will be replaced at the next update. Your files carry one of two other prefixes, according to their reach.

- **Kit-X —, for your extensions of the Kit:** an additional *stylesheet*, a colour palette, a tool of your own. These files apply to all your projects and follow you from one to the next without being renamed. The X stands for extension.
- **[Prefix] —, for what applies to one project only:** its documents, its *Registry*, its *data modules*, its executable artefacts. The full list is in Common Structure §2.

Note: *Your Kit-X files exist in a single copy. Back them up regularly.*

- **Kit files, replaced wholesale:** the four stylesheets and their Reference, the executable artefacts kit_*.py and kit_*.js, Kit - *Registry.js*, and the Kit's own documents. The replacement procedure is described in the Update Guide.
- **Project files, never touched:** your *Registry*, your *Prompt*, your documents, your *data modules*, your executable artefacts — Common Structure §12 — and the stylesheets this document teaches you to write.

Note: *A Kit file modified in place loses its modification at the next update, without warning: replacement is an overwrite. That is why extension goes through addition.*

3 Replacing the colours

The colours of the Kit's stylesheets are defaults, and three levels stack. Kit-X - Colors.js carries those that follow you from one project to the next. [Prefix] - Colors.js carries those that apply to one project only, and takes precedence over the former. Each file declares only the keys it changes; the rest keep the value of the level above.

The project file is driven by the requires.colors flag of its *Registry*: set it to true, otherwise the generator will not load it. The Kit-X one is driven by no flag — it is there or it is not. Each file exports colorEntries, an object with two tables: docx for the

documents, html for the web pages. Separating them lets you set the contrast on printed paper independently of the screen. The key names and their defaults are tabulated in the Reference of both stylesheets, and the file's contract in Common Structure §6.5.

The Kit checks neither contrast nor harmony: an unreadable colour produces a valid document that is tiresome to read. A value that is not a colour, on the other hand, stops *generation*, naming the offending key.

```
// Kit-X - Colors (2026-08-23 - 17h00).js
const colorEntries = {
  docx: { HEADING_COLOR: '8B0000' },
  html: { HEADING_COLOR: '#8B0000', CODE_BG: '#FAFAFA' },
};
module.exports = { colorEntries };
```

The generator loads both levels and applies them in order. The *stylesheet* imports neither the *Registry* nor these files: the generator bridges the two, as it does for the language and the glossary.

```
// setColors fusionne : deux appels successifs cascadent.
if (kitX)     style.setColors(kitX.docx);
if (couleurs) style.setColors(couleurs.docx);
```

4 Writing your own stylesheet

Your own *stylesheet* exports functions that produce blocks, exactly like those of the Kit. It complements the General *stylesheet* rather than replacing it: your generator imports both and calls one or the other depending on the block.

4.1 Naming and pairing

The name follows the convention of the Kit's stylesheets, with the prefix matching the file's reach: “Kit-X - *Stylesheet* - [Subject] - Code.js” for a *stylesheet* that follows you from project to project, “[Prefix] - *Stylesheet* - [Subject] - Code.js” for one specific to a single project. Kit-X suits most cases: a layout useful once is generally useful elsewhere.

The Reference.docx carries the same prefix, the same subject and the same *timestamp* as its .js. The two are delivered together — the pairing rule, *Prompt* §6, applies to your stylesheets as to the Kit's.

The Reference records what the code alone does not say: the formatting values, the return contracts, the mandatory sequences. It is regenerated at each change to the .js, never rebuilt from the code alone.

4.2 What your stylesheet imports

The Kit's geometry constants — indents, table widths, fonts, colours — belong to the General *stylesheet* and are read from it. Copying them into your own file creates a divergence that shows only at the first change of geometry.

```
const style = require('./Kit - Stylesheet - General - Code.js');
```

```
// La geometrie vient de General, jamais d'une copie locale.
const { INDENT_L1, INDENT_L2, TABLE_WIDTH_L1, FONT, BODY_TEXT_COLOR } = style;
```

Note: *The Kit's YAML stylesheet is the exception and imports nothing: it must remain usable on its own. Its geometry values are therefore copied and documented as such. That hand-maintained dependency let a level-3 divergence sleep for several months — importing is the default choice.*

4.3 What your stylesheet exports

Each function returns either a Paragraph or an array of elements, like the Kit's functions. The return contract is documented in your Reference: what the function returns, what must precede it, what must follow it.

Two of the Kit's rules apply to your functions. Naming is explicit, with no abbreviation and no acronym to decode — *Prompt* §12. And every displayed string goes through a localisation table, even if your project speaks only one language today — *Prompt* §4.1.

5 Registering your stylesheet

A Kit-X *stylesheet* follows you from project to project; its version therefore belongs in Kit-X - *Registry.js*, your own register, and not in the *Registry* of any one project. Create that file when you write your first *stylesheet*.

```
// Kit-X - Registry.js
module.exports = {
  stylesheets: {
    planches: { version: '1.00', ts: '2026-08-23 - 14h00' },
  },
};
```

Your generators read the active version there rather than carrying it hard-coded. Three registers then coexist, each with its own domain: the Kit's for the Kit's stylesheets, Kit-X's for yours, the project's for its documents and timestamps.

6 Writing a generator that combines the two

A generator imports the Kit's *stylesheet* and yours, then calls each according to the block to produce. The order of calls determines the order of the document, as in any generator.

```
const style    = require('./Kit - Stylesheet - General - Code.js');
const planches = require('./Kit-X - Stylesheet - Planches - Code.js');
const registry = require('./Monprojet - Registry.js');

style.setLanguage(registry.project.docLanguage);
planches.setLanguage(registry.project.docLanguage);

const corps = [
  style.h1('1', 'Montage'),
  style.paragraph('Le montage se fait en trois passes.'),
  planches.schema(image, 'Vue eclatee'),   // votre fonction a vous
  style.releaseParagraph(),
];
```

The section level is held by the General *stylesheet*, set by h1, h2 and h3 and read by the other functions. A *stylesheet* of your own that needs the current level reads it through style.currentLevel() and never writes it: only three functions set it, and adding a fourth reopens the matching error this arrangement closes.

7 What validation covers

The Kit's checking tools apply to your documents without adaptation. They read the produced file, not the code that produced it.

- **kit_validate_docx.js:** checks the structure of the produced.docx — levels, sequences, version *fingerprint*. It expects the *fingerprint* of the General *stylesheet*, which your documents carry since it builds them.
- **kit_check_setters.py:** the one exception to the sentence above: it reads the generator, not the document. It checks that the *setters* required by the *Registry* flags are called up front — an omitted *setter* silently disables the matching rendering. It therefore applies to the generator this document teaches you to write.
- **kit_check_markup.py:** compares the markup between a source binary and the regenerated document. Useful as soon as you take up an existing document.
- **kit_check_ossature.py:** compares the skeleton of two language variants.
- **kit_extract_map.py and kit_check_fidelite.py:** extraction recognises the Kit's blocks. A block produced by your *stylesheet* is unknown to it and will come back degraded. The fidelity check will tell you — that is its purpose. Two answers are possible: extend the extraction in your own working copy, or report the need to the Kit.

Note: *Extraction is part of the Kit and follows its updates. A local extension will therefore be overwritten. A block type meant to last is reported to the Kit rather than maintained alongside — Common Structure §13 describes the route.*

8 What breaks when the Kit evolves

The Kit offers no compatibility, forward or backward. An evolution may rename a function, change a signature or remove a helper. Your *stylesheet*, which calls the Kit's, bears the consequences.

- **A function name changes:** your calls fail loudly at the first run. That is the most comfortable case — the error is immediate and localised.
- **A geometry value changes:** your blocks shift without anything failing, if you copied the value instead of importing it. That is the reason for the rule in §4.2.
- **A sequence contract changes:** a *successor* that has become mandatory produces a malformed document that the *validator* flags. Read the *stylesheet*'s Reference after each update.

The session in which an update is absorbed — described in the Update Guide — is when these gaps are dealt with. Your stylesheets are reread there alongside your generators.

9 Creating templates outside the Kit

A project whose subject demands it — a recipe, a field sheet, a page format foreign to documentation — may create templates of its own, outside the Kit's forms. The right is declared in the *Registry* by allowNonKitTemplates, false by default. It is a written permission, not a lock: no check enforces it, and without it a project does not create *templates outside the Kit*.

The right bears on forms, never on rules. The project keeps the same reflexes: source file first, *Registry* as the authority, fresh *timestamp*, version number raised as soon as a file changes, generator rewritten from scratch, couplets in agreement, validation chain before any delivery, no loss of content, no formatting from *memory*, decisions written into the documents at the same time as they are applied to the code. Anything that would change a common rule goes back to the Kit — Common structure §13 — rather than being fixed on the spot.

What the Kit no longer guarantees: it no longer answers for the rendering produced by a *template outside the Kit*, nor for its survival when a *stylesheet* evolves. It still requires the reflexes, the validation chain and the identity of the sub-site.

10 When to report the need to the Kit

A local extension suits a need specific to your field. A need other projects would also meet belongs in the Kit, where it will be maintained and checked.

- The need can be stated without naming your field: it is general.
- You have written the same function twice for two different documents.
- Your *stylesheet* copies a geometry value from the Kit because no function exposes it.
- Extraction does not recognise your block and you maintain a local copy of the tool.

La voie de remontée est décrite dans Common Structure §13. Un besoin remonté et accepté devient une fonction du Kit, et votre extension locale disparaît au profit de celle-ci.