Grok AI

Documentation — Manual — EN

1 Introduction

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

This manual introduces *Grok* and how it is used with the Documentation Kit. It is written for anyone who wants to produce documents — guides, procedures, HTML sites — without being tied to a single *artificial intelligence* provider.

The Kit is provider-neutral: each assistant has its own section. This one describes what *Grok* can do inside an isolated project, how it leans on the artifacts, and how the production chain keeps deliverables consistent.

Note: *Technical terms are defined in the Kit glossary. When in doubt, look there before going on.*

2 Identity

2.1 Product and publisher

Grok is a *large language model* developed by *xAI*. It reads and writes text in French, English, German and other languages. Inside a project environment it also runs tools: reading and writing files, executing code, searching the web, generating documents.

xAI publishes several models side by side. The names change; the tiers stay: fast for simple volume, balanced for everyday work, more powerful for extended reasoning. For the Kit, stay on the default working tier unless a task resists.

2.2 What Grok does

- **Structure.** Outlines, sections, tables, production flows.
- **Write.** Clear technical prose, in the project language, at the level asked for.
- **Execute. Node.js** and *Python* scripts, validation, placing files in the artifacts.
- **Check.** *Stylesheet*, levels, naming and the .docx *fingerprint*.
- **Resume.** Re-read the *project files* and pick up where the last session stopped.

2.3 What Grok is not

Grok is not a search engine. A fluent answer is not evidence. It has no access to another project's artifacts. It does not remember an earlier *conversation*, except through the files placed there and, where enabled, *persistent memory*.

It can hallucinate — above all on software versions, recent dates and figures. For anything meant to be published or used in production, check the critical facts against primary sources.

💡 *Asking where a piece of information came from is always fair, including after a web*

search.

3 Working environment

3.1 Isolated projects

A *Grok* project gathers the conversations and files of one subject area. Each project is isolated: *Grok* cannot see another project's artifacts. That is deliberate. A professional *context* should not bleed into a personal one, and a Kit project should not mix its rules with those of a *consumer project*.

Open a *conversation* in the right project. A *conversation* outside a project has neither the reference files nor the conventions of that area.

3.2 Artifacts — the source of truth

The files in the current project's artifacts are the source of truth: *Registry*, *Prompt*, stylesheets, documents, glossary, generators. The session's *working memory* disappears. The artifacts remain.

Item	Role	Who maintains it
Artifacts	Source of truth between sessions	You, after each delivery
Project *prompt*	Standing rules of work	You and *Grok*, on approval
Registry	Versions, timestamps, language, flags	Updated at every delivery
Working memory	History of the running *chat*	Platform — not reliable on its own

Note: *After a major update, place the new version and remove the old one. Two versions of the same file yield contradictory rules.*

3.3 Context window

Everything written in a *conversation* takes up the *context window*: messages, replies, attached files. As the limit nears, the detail of the earliest instructions fades. For long work, several short sessions with a summary and up-to-date artifacts beat one endless *conversation*.

💡 *If you find yourself restating a rule already set in the same session, it is time to close cleanly and start afresh.*

4 Grok and the Kit

4.1 Its role in the chain

The Kit is not a Word template to fill in by hand. It is a chain: approved content, generator script, *stylesheet*, Luxembourg *timestamp*, validation, delivery. *Grok* runs that chain inside the project. It does not invent the formatting — it calls the functions of the active *stylesheet*.

#	Step	Deliverable
1	Framing — subject, audience, structure	Approved outline
2	Content — section by section	Approved text
3	Script — *stylesheet* calls and *setters*	Generator ready
4	*Generation* — Packer + injectCustomProps	Timestamped document
5	Validation — kit_validate_docx and checks	Exit 0
6	Delivery — one file at a time	*Artifact* up to date

The checks before and after are binding. A failure halts the chain: nothing is delivered until it has passed in full.

4.2 Capabilities that matter for documentation

- **Read in full.** *Prompt*, *stylesheet* Reference, *Registry* — before any *generation*.
- **Strict levels.** p2 under an h2, note3 under an h3. No hand formatting outside the helpers.
- **Naming.** Prefix - Category - Subject (YYYY-MM-DD - HHhMM).ext, *timestamp* through getLuxTimestamp().
- **Publication.** Once the .docx files pass, the site generator produces HTML, CSS, PDF and the glossary.
- **Resumption.** Re-read the artifacts, summarise the state, announce the plan, wait for the go-ahead.

4.3 What the Kit requires of Grok

Announce before acting. Wait for confirmation before altering an existing document. Preserve the source content. Deliver one file at a time. Never rebuild a document from session *memory* alone.

> 💡 *If a rule seems forgotten midway through a long session, a short reminder is enough. It is also a sign that the context window is filling up.*

5 Other documentation projects

5.1 Bring the Kit in whole, do not tinker

For a new area — home automation, network, a collection, a household procedure — the whole Kit is copied into the *consumer project*. No cherry-picking. Stylesheets, *Registry*, *validators* and conventions travel together.

Grok then sets up the prefix, the language, the author and the *inventory* following the initialisation guide. The result: the same habits, the same output, a site of its own.

5.2 What changes from one project to the next

- **The prefix.** Never “Kit”, unless adapting the Kit itself.
- **The language.** project.docLanguage in the *Registry*. The Kit stands on FR by exception; *consumer projects* start on EN.
- **The data modules.** Glossary, brands, possibly variables and entities — according to Registry.requires.
- **The content.** The only place where the project speaks for itself. The formatting engine stays the Kit's.

Note: *A stylesheet evolution happens in the Kit project, then goes across to the consumer projects in one block. See the update guide.*

6 Limits and checks

6.1 Facts, versions, dates

The errors bear above all on precise data. A web search lowers the risk; it does not remove it. Two sources can contradict each other. For a procedure, a software version or a published figure, go back to the official documentation.

6.2 Files and formatting

Attaching a .docx to the artifacts does not preserve its formatting: the platform extracts the plain text. To modify an existing document, place the binary in the running *conversation* and regenerate through the *stylesheet* — never patch the XML blind when a Kit helper exists.

Note: *Do not count on continuity between projects. What was established in one must be brought into the other again, through files or by stating it explicitly.*

7 What comes next

This manual sets the frame. The *Grok* practical guide covers the habits of a session, phrasing techniques, the flows for each kind of work and templates ready to send.

The Kit evolves. The principles of this document hold: artifacts as the source of truth, *generation* through the *stylesheet*, validation before delivery, isolation between projects.