Grok AI

Documentation — Practical Guide — 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 guide is the session tool. It does not repeat the presentation of *Grok*: it says what to do, in what order, with which messages. Each section stands on its own and can be consulted mid-work.

Audience: users of the Kit and of *consumer projects* who work with *Grok*. Tone: direct.

2 Session card

#	Habit	Effect
1	One goal per session	Keeps conversations from sprawling
2	Outline first, content after	Fixing a structure costs less
3	Artifacts are the source of truth	*Chat memory* does not survive on its own
4	Announce, then wait for the go-ahead	You keep control of the files
5	Strict *stylesheet* levels	Readability and validation
6	Validate before delivering	kit_validate_docx exit 0
7	Correct in a targeted way	Problem, direction, what to keep
8	Say what you refuse as well	Less filler
9	Close formally	Summary, *inventory*, open points
10	Full Kit injection	No partial update

3 Running a session

3.1 Opening

The first message sets the frame: project, goal for the day, files to reread, constraints of language or format. Ask *Grok* to confirm what it has read — the *stylesheet* version in particular — before anything is produced.

```
New session on [PROJECT NAME].
Goal: [PRECISE GOAL].
Read the artifacts, in particular [KEY FILES].
Confirm the stylesheet version you have read.
Announce the plan. Run nothing until I have said go.
```

3.2 During the work

- **One section at a time.** Approve it before moving to the next.
- **Files in the project.** Point to the artifacts rather than pasting long extracts again.

- **A short reminder.** If a rule drifts, restate it in one sentence. Do not explain everything again.
- **Open points.** Note what comes up and is not being dealt with now.

3.3 Closing

Five minutes of closing save you explaining everything again in the next session. The Kit's short formula is enough.

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

Expected: files produced with their exact names and timestamps, decisions, open points, suggestions for what follows, files to place in the artifacts.

4 Wording

4.1 Three questions before sending

- **What.** Is the deliverable named (outline, section, script, document)?
- **For whom.** Are the audience and the level stated?
- **Under what constraints.** Language, length, format, what not to do?

If any of the three answers is vague, complete the message before sending it.

4.2 Show rather than describe

One example of input and output aligns the format faster than a description. Useful for extractions, rewrites and repetitive tables.

```
Format wanted, one line per item:
Name | Type | Location | Value
Example: Living room temperature | Temperature |
         Living room | 21.3 °C
Now process the attached file.
```

4.3 Correcting without breaking everything

Regenerating without changing the *instruction* produces a result of the same kind. If the *prompt* was imprecise, write a corrective message.

```
The section [NAME] does not work:
- [PROBLEM 1]
- [PROBLEM 2]
Rewrite it with: [DIRECTION].
Keep: [WHAT TO PRESERVE].
```

Note: *Avoid “this isn't good, do it again”. Say whether it is the substance, the structure or the tone, and give the direction.*

5 Standard workflows

5.1 A professional document

#	Action	Stop while
1	Framing: audience, scope, language	The framing is not approved
2	Detailed outline of the sections	The outline is not approved
3	Writing section by section	The current section is not approved
4	*Stylesheet* script and *setters*	The upstream checks fail
5	*Generation* and kit_validate_docx	The exit code is not 0
6	Delivery into the artifacts	The name or the *timestamp* is wrong

💡 *Beyond a dozen sections, one session per group of three. The context window is not a binder.*

5.2 Analysis and diagnosis

- **Attach every useful file with the first message.**
- **Ask for step-by-step reasoning.** *Inventory*, causes ruled out, diagnosis, then actions.
- **Approve the diagnosis before the recommendations.**
- **Close with a synthesis worth filing.**

5.3 A script or an automation

Specify the inputs and the outputs before a line is written. Ask first for the list of *stylesheet* functions to call. Attach the active *stylesheet*. Test each block before moving on.

> For a Kit script: the Reference read within the current session, contracts respected, no call outside a helper.

6 Further templates

6.1 Picking up after a break

```
I'm picking [PROJECT NAME] back up after [HOW LONG OR CONTEXT].
Read [KEY FILES].
Summarise the current state in five points.
Goal for this session: [GOAL].
```

6.2 Generation once the content is approved

```
The content of sections [LIST] is approved.
Generate the full script.
Stylesheet: [FILE + VERSION]
```

```
Output name: [PREFIX] - [CATEGORY] - [SUBJECT], timestamp
             via getLuxTimestamp()
Author: [NAME]
Call the Registry setters, then the full check chain before
any delivery.
```

6.3 Analysing a technical file

```
Here is [FILE TYPE]. [ATTACHED FILE]
Produce:
1. A structured inventory of the [ITEMS].
2. Anomalies or points of concern.
3. [OPTIONAL: recommendations].
Reason step by step before concluding.
```

Note: *These templates are starting points. Precise wording beats a template followed word for word.*

7 Costly mistakes

Do	Avoid
One clear goal per session	Mixing everything into one long *conversation*
Approve the outline before the content	Asking for a complete document in one go
Keep the artifacts current	Relying on *chat memory* alone
***Stylesheet* helpers at the right levels**	Manual formatting outside the styles
Announce, then go	Letting *generation* start without an approved plan
Validate the .docx before delivery	Delivering an unchecked file
A note or a tip after the useful content	A note or a tip right under a heading
Full Kit injection	Copying three files and ignoring the rest