Claude 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 operational companion to the user manual. Where the manual explains the concepts, this guide provides tools you can use straight away: a quick-reference card of the ten habits, formulation techniques with concrete examples, workflows by project type, and ready-made *prompt* templates.

Each section is built to be consulted on its own, mid-session, without having to reread the whole manual.

2 The ten habits — quick reference card

These ten habits sum up the most effective practices. Worth a look at the start of a session, or worth keeping within reach.

2.1 Starting habits

- **Frame it from the start:** give the *context* — who you are, what you are working on — the goal, and the constraints of format, language and length. Those three things in the first message frame the whole *conversation*.
- **Be specific:** “a five-line paragraph in a technical style for a non-specialist audience” is worth infinitely more than “write something on this subject”. The precision of the request directly determines the quality of the answer.
- **Break it down:** for any complex task, start with the outline. Never ask for a ten-page document in one go: ask for the structure, approve it, then take each section one at a time.

2.2 Iteration habits

- **Iterate without waiting for perfection:** the first attempt is rarely the right one, and that is not the aim. Each round of correction sharpens the result. Move from the general to the particular.
- **Correct precisely:** do not say “this isn't good, do it again”. Explain why it is unsatisfying and in which direction to go: substance, structure, tone or length.
- **Tell regenerating from correcting:** regenerating without correcting your request produces a result of the same kind. If it was the *instruction* that was imprecise, write a new corrective message.

2.3 Management habits

- **Use projects:** for any recurring work, work inside a project with the reference files uploaded. A *conversation* outside a project starts from nothing every time.

- **Check critical facts:** errors bear above all on precise data, software versions and recent events. For any content meant to be published or used in production, check against primary sources — including when a web search has been made.
- **Close your sessions:** five minutes of structured closing — summary, open points, suggestions — let the next session start without explaining the *context* again.
- **Keep the project files current:** after every major update, upload the new version and delete the old one. Two versions of the same file produce contradictory rules.

2.4 The rule of three questions

Before sending a message, ask yourself these three questions.

- Is it clear what I want to produce?
- Are the *context* and the audience stated?
- Are the constraints of format, length and style given?

If the answer to any of them is “no” or “maybe”, complete the message before sending it.

3 Advanced formulation techniques

These techniques get markedly better results on complex tasks. They are presented in order of increasing complexity, with concrete examples.

3.1 Show rather than explain

Instead of describing the expected format in words, show one or more examples of what you want. The *alignment* on structure, tone and level of detail is immediate.

- **Principle:** give one to three examples of input and output before the real request.
- **Example:** “Here is the output format I want. For each sensor, extract the name, the type, the location and the last value, separated by vertical bars. Example: Living room *temperature* | *Temperature* | Living room | 21.3 °C. Now process this data file.”

Seeing the example, the format is reproduced exactly, without any need to describe it in detail.

Note: *Particularly effective for repetitive tasks: data extraction, styled rewriting, classification, summaries in a set format.*

3.2 Forcing explicit reasoning

For tasks that call for complex reasoning — analysis, diagnosis, multi-criteria comparison — explicitly ask for step-by-step reasoning before any conclusion. The quality of the final answer improves markedly.

- **Principle:** add “reason step by step” or “think aloud before concluding”.
- **Example:** “My file server has been unreachable from the secondary network since this morning. Here is the configuration and the logs. Analyse the problem step by step: identify the possible causes, rule out those the logs contradict, then give the most likely diagnosis and what to do about it.”

Note: *Useful for technical diagnosis, architecture reviews, risk assessment and choices between several options.*

3.3 Calibrating the role

Assigning a precise role steers the level of expertise, the vocabulary and the style of the answer. This is not play-acting: it is an efficient way of specifying the level and register expected.

- **Principle:** start with “you are”, followed by a precise role and its *context*.
- **Targeted expertise:** “you are an experienced network engineer” yields technical answers, without excessive simplification.
- **Suitable register:** “you are a technical writer documenting for advanced users who are not developers” adjusts the vocabulary.
- **Targeted review:** “you are an experienced home automation architect, review this configuration and identify the problems” steers towards detection rather than explanation.

Note: *Avoid roles that are too generic, of the “you are an expert” kind. Be precise about the field, the level and the intended audience.*

3.4 Saying what you do not want

Specifying what you do not want matters as much as specifying what you do. Negative constraints head off generic answers and the usual stylistic excesses.

- “Don't open with a general introduction, go straight to the point.”
- “Don't use bullet lists, write structured prose.”
- “Don't rephrase my question, answer it directly.”
- “Don't add caveats, I know the limits.”
- “Avoid enthusiastic openers at the start of an answer.”

3.5 The two-stage prompt

For long or complex documents, a two-stage *prompt* produces far better results than a single request.

- **First stage:** ask for the structure. Approve or adjust the outline before any writing.

- **Second stage:** give the go-ahead section by section — “now do section 2.1 in detail”.

💡 *This method applies to generation scripts too. First ask for the list of stylesheet functions to be used, approve it, then ask for the full script.*

3.6 Anchoring on a reference document

When you want *alignment* on an existing style, format or body of content, attach the reference document and ask for it to be analysed before anything is produced.

```
Here is the existing document [ATTACHED]. Analyse its style, its
structure and its level of detail.

Now produce an equivalent document on [NEW SUBJECT], keeping
strictly to the same register and the same organisation.
```

4 Workflows by project type

These workflows are proven sequences for the most frequent kinds of project. Each is built to be followed session by session, with clear deliverables at every stage.

4.1 Technical documentation

Suited to installation guides, configuration manuals, system documentation, procedures.

Step	Action	Deliverable	Session
1 — Framing	Define scope, intended audience, level of detail	Approved framing note	1
2 — *Inventory*	List every section and subsection	Approved detailed outline	1
3 — Content	Write section by section, in iterations	Approved sections	2 to N
4 — Script	Generate the production script through the Kit	Script ready	N
5 — *Generation*	Run the script and pass the checks	Timestamped, validated document	N
6 — Revision	Proofreading, corrections, regeneration if needed	Finished document	N+1

Note: *For documents of ten sections or more, plan one session per group of three sections. Do*

not try to produce everything at once: the context window fills up and the first decisions survive only in summary form.

4.2 Analysis and diagnosis

Suited to log analysis, configuration diagnosis, architecture review, audits and comparison of solutions.

- Attach every relevant file from the start.
- Ask for a reasoned, step-by-step analysis.
- Approve the diagnosis before asking for recommendations.
- Ask for the corrective actions in order of priority.
- Ask for a synthesis at the end of the session, for the record.

💡 *For complex analyses, break it down: first the inventory of problems, then the prioritisation, then the solutions. Do not ask for everything at once.*

4.3 Learning a technical subject

Suited to discovering a technology, a protocol, a programming interface or a tool.

- Overview: “give me eight key points on this subject, for someone who already knows such and such a field”.
- Deepening: identify the two or three least clear points and ask for targeted explanations.
- Anchoring: ask for a concrete example in your real *context*.
- Application: work through a real practical case with the new knowledge.
- Consolidation: ask for a structured summary to keep as a reference.

Note: *Ask to be questioned in return. This dialogue mode speeds up learning and reveals the gaps you would not have spotted on your own.*

4.4 Generating scripts and automations

Suited to document *generation* scripts, home automation routines and data processing.

- Specify the inputs and the outputs precisely before a line is written.
- Ask for the structure of the script first, and approve it before the code.
- Generate the code section by section for long scripts.
- Test each section immediately before moving to the next.
- Ask for a commented version, to make future maintenance easier.

💡 *For Kit scripts, attach the active stylesheet to the conversation. The calls will then be adjusted to the functions actually available rather than to those assumed.*

5 Prompt templates

These templates are proven models for the most frequent situations. Copy, adapt the parts in brackets, send.

5.1 Opening a session

To be used at the start of any working session in a Kit project.

```
Hello. New session on the project [PROJECT NAME].

Goal for this session: [PRECISE GOAL — for example, write sections
3 and 4 of the network guide].

First confirm that you have read the project files, in particular
[KEY FILES — for example, the project prompt and the active
stylesheet]. State the version of the stylesheet you have read.
```

5.2 Closing a session

To be used at the end of every session. Can be sent as it stands.

```
Before we close this session:

1. Summarise the files produced, with their exact names and
   their timestamps.
2. List the important decisions taken during the session.
3. Set out the open points — what is left to do or to check.
4. Offer two or three suggestions for the next session that
   you think would be useful but that we did not discuss.
5. Tell me which files I need to upload to the project before
   the next session.
```

5.3 Generating a document

To be used once the content is approved and you are ready to produce the document.

```
The content of sections [LIST] is approved. Now generate the full
script to produce the document.

Stylesheet   : [FILE + VERSION]
Output name  : [PREFIX] - [CATEGORY] - [SUBJECT], timestamp
               obtained from the Kit function, never typed
Author       : [AUTHOR NAME]
Structure    : [level 1 for the main sections, level 2 for the
               subsections, and so on — every element at the
               level of its parent heading]

Call the setters required by the project registry, then run the
full check chain before delivering anything to me.
```

5.4 Targeted correction

To be used when a section produced does not meet expectations.

```
The section [NAME] does not work, for the following reasons:

- [PROBLEM 1 — for example, too technical for the intended audience]
```

```
- [PROBLEM 2 — for example, points 2 and 3 overlap]
- [PROBLEM 3 — for example, the tone is too formal]

Rewrite this section with: [DIRECTION — for example, accessible
language, points 2 and 3 merged, a direct tone].

Keep: [WHAT TO PRESERVE — for example, the concrete examples and
the four-point structure].
```

5.5 Analysing a technical file

To be used to analyse logs, configurations or data exports.

```
Here is [FILE TYPE — for example, a data export, application
logs, a configuration file]. [ATTACHED FILE]

Analyse this file and produce:

1. A structured inventory of the [ITEMS — for example, devices,
   automations, errors].
2. The anomalies or points of concern you identify.
3. [OPTIONAL — for example, recommendations, or a comparison
   with the previous configuration].

Reason step by step before concluding on the anomalies.
```

5.6 Resuming a session

To be used when picking a project back up after a break, or in a new *conversation*.

```
Hello. I'm picking the project [PROJECT NAME] back up after
[HOW LONG or CONTEXT].

Read the project files, in particular [KEY FILES]. Summarise in
five points what you understand of the current state.

We are going to work on: [GOAL FOR THIS SESSION].
```

Note: *These templates are starting points. Adapt them to your context: the precision of your own wording always beats a generic template followed word for word.*