

The rail kit:

four Skills + one week

A Skill is a fixed instruction with one input and one output. Each one here is a small folder in the standard agent format (SKILL.md + templates + an example) that loads straight into Claude Code, Cursor or any other agent. No client names, no prices, no meeting content. What you do get: the structure we run on every day.

How to use it: in Claude Code it takes two lines, `/plugin marketplace add yonatangross/rail-kit` and then `/plugin install rail-kit@yonyon`. Or download the ZIP from yonyon.ai/rail and copy the four folders into `~/claude/skills/` (or paste an instruction block as the system prompt in any other tool). Source, tests and examples: github.com/yonatangross/rail-kit, open source. No Wispr Flow yet? Signing up through wisprflow.ai/r?YONATAN1 gets you a free month of Pro (Yonatan's referral link). The instructions are in English because that is how agents read; the drafts come out in your client's language. Nothing is ever sent on its own: the Skills draft, you send.

1 Post-call recap (post-call)

Input A transcript or summary of a client call.

Output A short follow-up message + a one-page scope. Both are drafts.

The instruction (paste as is)

You draft two artifacts after a client call, in the client's language. Input: a transcript or summary.

1. Read the source; prefer the verbatim transcript over the summary.
2. Extract facts first: asks, agreements, open questions, next step + date, any number said aloud with its exact sentence. Anything not in the source is marked [check the transcript].
3. Draft a WhatsApp follow-up: 4 to 6 lines, warm and professional, opens with what you understood from their side, closes with the next step and a date. No prices, no links, no numbered lists.
4. Draft a one-page scope: goal, what gets built (up to 5), what is NOT included, what the client provides, done criteria, relative timeline. One price line only if the price was said explicitly; otherwise "Price: to be agreed".

Rules: a "Speaker 1/2" label is not an identity; if the name is not certain write "named on the call".

Every number and commitment traces to a sentence in the source. Draft only, never send.

Message template

Hi {name},
Thanks for the call. My understanding is that {the core need, in their words}.
We agreed on: {point 1}, {point 2}{, point 3}.
Next step: {what} by {date}.
{a warm closing line}

2 Client context (client-context)

Input A client name + every source you have: call summaries, emails, notes.

Output One page that pulls it all together and flags conflicts between sources.

The instruction (paste as is)

You produce a one-page state of a single client before work starts. Input: the client name
and every available source (if a Wispr Flow MCP is connected, search meetings there first).
Fill exactly these blocks, in the client's language: current state (3 lines, newest source
wins); agreed (each with date + source); open (each with an owner); contradictions
(both
sides, no verdict); next step (one line).
Rules: every line cites its source in parentheses; a line without a source is
dropped.
Never fill a gap from reasoning; a gap is a line under "open". Numbers only with a
transcript
sentence behind them. Read-only.

Page template

```
# Client context: {name}, {date}
## Current state
{up to 3 lines} (source: {..})
## Agreed
{agreement} ({date}, source: {..})
## Open
{question or task}, owned by {who} (source: {..})
## Contradictions
{source A says X} versus {source B says Y}. Not resolved.
## Next step
{one line}
```

3 Call prep (prep-call)

Input Client name, call date, and the goal of the call in one sentence.

Output A one-page prep sheet you can read in the two minutes before the call.

The instruction (paste as is)

You write the sheet read in the two minutes before a client call. Input: client name, date, the goal in one sentence, and the client's history (a client-context page, or Wispr Flow meetings via MCP). Produce, in the client's language, one screen: opener (3 lines you can say out loud); time split (3 to 4 blocks with minutes); three must-ask questions in priority order, each grounded in something the client said or asked before, with the source; three if-then pivots; what NOT to open; the close line and the next step to ask for. Rules: no prices unless the goal of the call is price, then one line with the range agreed beforehand. Plain lines, no tables. Read-only; the only write is the sheet.

Sheet template

```
# Call prep: {name}, {date}, {goal}
## Opener (2 min)
{3 lines}
## Time split
Minutes 0 to 2: opener. Goal: {..}
Minutes 2 to {n}: {segment}. Goal: {..}
Minutes {n} to {end}: close. Goal: {..}
## Three questions you must ask
1. {question} (because: {source})
## If X, then Y
If {scenario}: {response}
## What not to open on this call
{topic}
## Close
{line}. Then ask for: {next step}
```

4 State update (sync-call-state)

Input A call summary you have approved.

Output A proposed update to the client record that waits for your approval before anything is written.

The instruction (paste as is)

You propose, never apply, an update to a client record after an approved call summary.

Input: the approved summary, and the current record if available.

Output, in the client's language: for every field the summary supports changing, a block with

the field, the old value, the new value and the exact sentence that justifies it; then a

three-line record note (date, what happened, what is next); then one question: Write this? yes / no.

Rules: never write to any system; a field with no explicit sentence behind it is not proposed; money fields need the exact words or stay unchanged.

Proposal template

```
# Proposed update: {name}, {date}
## Fields to change
Field: {field}
Old: {old value}
New: {new value}
Because: "{quote from the summary}"
## Record note
{date}: {what happened}. Next: {what}.
Write this? yes / no
```

One page on MCP: your meetings, readable by your agent

The four Skills work on anything you paste. They work far better when the agent reads the meetings itself, and that is exactly what the Wispr Flow MCP does.

1. On Mac: Wispr Flow settings, then MCP (after Connectors).
2. Claude or ChatGPT: one click on Connect, approve in the browser, done.
3. Any other tool (Cursor, VS Code, Claude Code): copy the URL under All other apps and add it as a new MCP server in that tool.
4. What the agent sees: meeting summaries and transcripts, notes, tasks, calendar. Read-only.

Once it is connected, Skill 2 and Skill 3 need nothing pasted: say the client's name and the agent finds the meetings on its own. Requires the Notetaker (beta) and Cloud Sync.

The one-week plan

- **Day 1** Install and set the shortcut. Ten minutes of dictating instead of typing, only for the long prompts.

- **Day 2** Personal dictionary. Internal terms, product names, the jargon that keeps coming up.
- **Day 3** Meetings. Turn meeting summaries on, and put every client meeting on a calendar invite.
- **Day 4** One place where summaries land, and it is the place where you actually work.
- **Day 5** The rule: numbers move only with a transcript. Written down somewhere visible.

If after this week you want the rail to run on its own (summaries that file themselves into a knowledge base, dictation that triggers an agent, the Skills tuned to your business), that is exactly what I help businesses build at Yonyon AI.

A 15-minute call: yonyon.ai/rail

yonyon.ai · [@yonyoniz](https://twitter.com/yonyoniz)