A Fable Protocol: How to Brief AI Like a Capable Teammate

Larry Todd Wilson · ltwilson.com
6-12-2026

Fable has been out for three days. This guide is a guess, an educated one, but still a guess. I have been working with AI systems long enough to know that the first week of a new model is mostly projection: you bring your habits, your old prompts, your accumulated intuitions, and you see what holds and what needs to be unlearned.

What follows is my best attempt to distill what seems durable: not tricks, not workarounds, but the underlying logic of good delegation. If Fable surprises me, and it probably will, I will revise. Consider this a living draft from someone paying close attention.

Core Idea

A good Fable workflow has four layers:

The goal: What are we trying to accomplish?
The brief: What does Fable need to know to do the work well?
The boundaries: What should it do, avoid, verify, or pause for?
The review loop: How will we know whether the result was actually good?

A prompt is only one part of the system. A better question is:

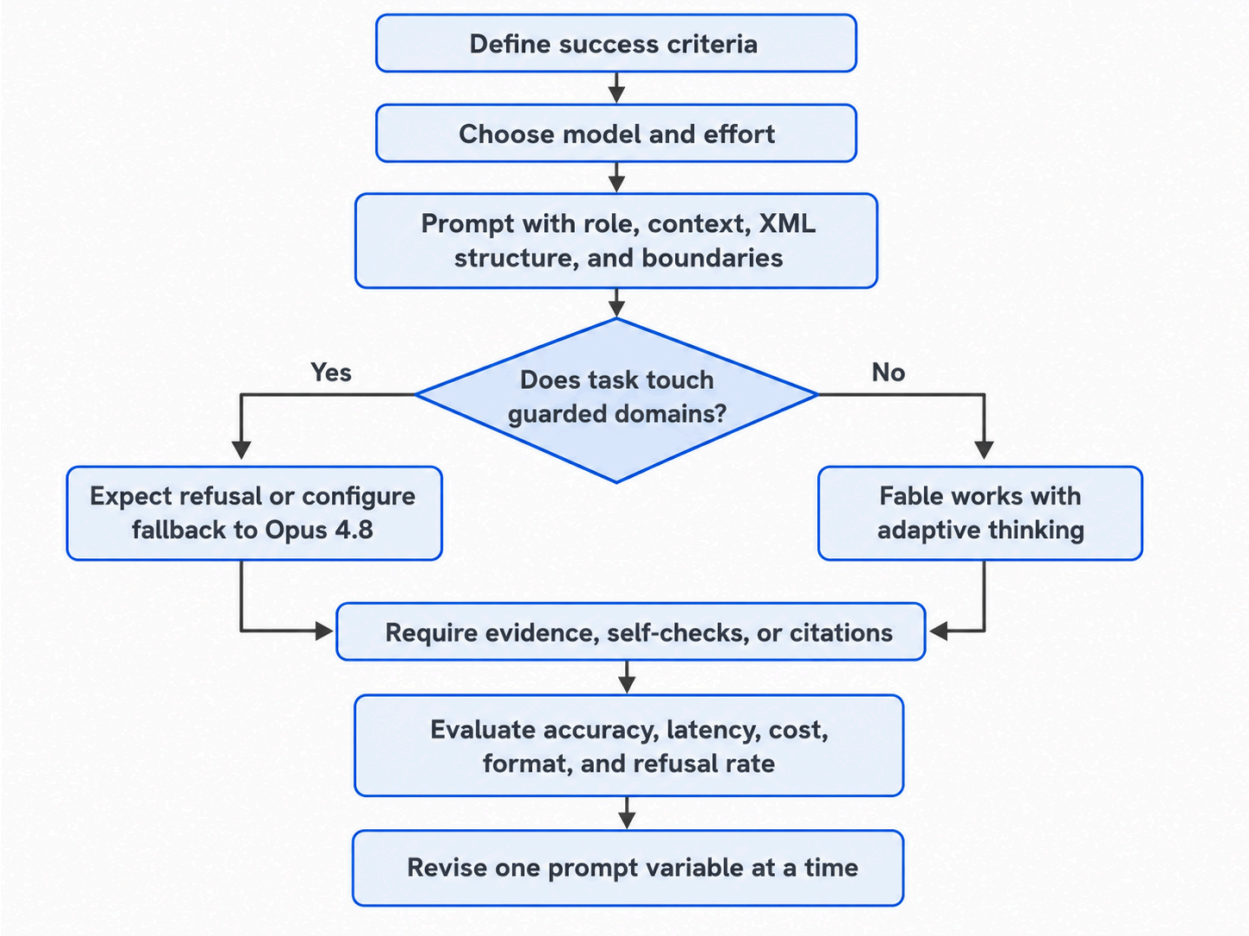
What kind of working relationship am I creating?

Do not merely ask for an answer. **Design the work.**

A Fable Operating Loop

This loop is the whole method in miniature. First decide what success means, then choose the right level of effort, brief Fable clearly, handle safety or refusal issues, require evidence, evaluate the output, and revise only one variable at a time.

Try this.



The Protocol

1. Define Success Before You Prompt

Do not begin with the task. Begin with the outcome.

Weak:

Write a proposal.

Better:

I need a one-page proposal for a cautious nonprofit board. The goal is to help them approve a small pilot project without feeling pressured. The proposal should be clear, concrete, and low-hype.

This gives Fable a target. Before asking for work, answer four questions:

Question	Example
What am I trying to accomplish?	Get approval for a pilot project.
Who is this for?	A cautious nonprofit board.
What should the output help them do?	Understand the decision and feel safe saying yes.
What would make the output successful?	Clear, practical, credible, and easy to approve.

A vague prompt asks the model to guess. A good brief removes the guesswork.

2. Choose the Right Level of Effort

Not every task deserves the same depth.

Lighter effort	Higher effort
Clean this up	Strategy
Summarize this	Research
Reformat this	Financial or operational analysis
Draft a simple note	Complex writing
Extract the main points	Technical review
	High-stakes communication
	Work involving ambiguity or tradeoffs

A practical rule:

Match the model's effort to the cost of being wrong.

For simple tasks, ask for speed and restraint. For hard tasks, ask for deeper reasoning, assumptions, tradeoffs, risks, and a recommendation.

Developer / API note: If you are using Fable through an API, treat effort as both a prompt instruction and a runtime decision. Model choice, effort setting, max output, fallback behavior, latency, and cost are not just writing choices. They are part of the system design.

3. Brief Fable Like a Teammate

3. Brief Fable Like a Teammate

Do not dump everything into one paragraph. Structure the prompt so Fable can tell the difference between instructions, source material, examples, and the final task. Use simple tags:

```
<context>
...
</context>
<source_material>
...
</source_material>
<task>
...
</task>
<constraints>
...
</constraints>
<output_format>
...
</output_format>
```

This matters because complex prompts often mix many kinds of information. Tags help Fable keep them separate.

4. Give the Right Context, Not All Context

Give the smallest set of context that could change the answer.

Useful context includes:

- Brand rules and known constraints
- Prior examples and decision history
- Customer notes, research excerpts, product details
- Audience descriptions
- What has already been decided
- What must not be changed

Bad context is everything you happen to have. Good context is what the work actually needs.

5. Show What Good Looks Like

Do not rely on taste words alone. Words like *premium, strategic, warm, clear, strong,* and *polished* are too elastic. They can mean ten different things.

Give an example:

Use this example for structure and tone. Notice the short paragraphs, concrete claims, and calm confidence. Match the clarity, not the wording.

Even better, include a counterexample:

Do not make it sound like this. This version is too salesy, too abstract, and buries the main point.

Examples show the destination. Counterexamples mark the ditch.

6. Control Scope Before Fable Expands the Work

Capable models can over-deliver. That sounds useful until you ask for a simple review and get a full rebrand, a content strategy, a new operating model, and twelve optional next steps.

Use scope rules:

- Do the simplest thing that works well.
- Do not add features, sections, abstractions, or tangents unless they are necessary.
- If I am describing a problem, assess and recommend. Do not implement changes unless I ask.
- If options exist, recommend one clear path.

Good scope rules prevent beautiful wrong turns.

7. Tell Fable When to Ask and When to Act

A weak workflow creates two bad behaviors: the model asks too many questions and stalls, or the model runs too far without permission.

Use this default:

Ask clarifying questions only when the missing information would materially change the result. When you have enough information to act, act. If there are tradeoffs, recommend the best path and briefly explain why.

For autonomous work, add:

Pause only for destructive or irreversible actions, real scope changes, or information only I can provide. Otherwise, continue until the task is complete.

This creates bounded momentum. Fable should not be helpless. It also should not be reckless.

8. Require Evidence, Not Just Confidence

A persistent danger is polished uncertainty. An answer may sound smart, look professional, and feel convincing. That does not mean it is true.

Use an evidence rule:

Before making factual claims, check them against the supplied context, source material, tool results, or reliable references. If something is verified, say so. If it is an assumption, label it. If it is unclear, say what is missing. If it is unsupported, do not present it as fact.

For important work, ask Fable to separate:

Category	Meaning
Verified	Supported by the provided material or a reliable source.
Assumption	Reasonable, but not directly proven.
Interpretation	A judgment or synthesis based on evidence.
Gap	Something needed but missing.
Risk	Something that could make the answer wrong or incomplete.

Do not reward confidence. Reward accountability.

9. Use Self-Checks Carefully

9. Use Self-Checks Carefully

For serious work, prefer stronger verification:

- Source quotes
- Citations
- Calculations
- Test results
- Comparison against provided files
- A separate reviewer
- A second model or second pass
- A checklist tied to the success criteria

Use this rule:

Fable may review its own work, but important claims need evidence.

10. Plan for Refusals and Guarded Domains

Some tasks touch areas where Fable may refuse or narrow the answer, especially if the request involves harmful domains or attempts to extract internal reasoning.

Build that possibility into the workflow:

- Ask for safe, bounded help.
- Narrow the task if needed.
- Request high-level explanation rather than operationally harmful detail.
- Use fallback behavior when appropriate in API systems.
- Track refusal rate if you are using this in production.

This is not about bypassing a safety system. It is about designing work that stays useful, safe, and predictable.

Source-confidence note: Treat model-specific claims in this guide as provisional. Fable is new, and early behavior may change quickly. The durable lesson is not a trick for one model. It is the discipline of clear delegation: define success, give the right context, set boundaries, require evidence, and review the result.

11. Do Not Ask for Internal Reasoning

Do not ask the model to reveal its internal chain-of-thought. Ask for useful accountability instead.

Good requests	Poor request
Give a concise rationale	Show your full internal reasoning.
List the assumptions	
Show the evidence	
Give the decision factors	
Explain the tradeoffs	
Cite the source	
Provide a verification step	

You do not need internal reasoning to evaluate the work. You need evidence, assumptions, and a clear decision trail.

12. Make the Final Report Easy to Read

After serious work, do not accept a maze. Ask for the outcome first.

A good final report includes:

#	Element	What it covers
1	Outcome	What happened or what was found.
2	Deliverable	The actual answer, draft, analysis, or recommendation.
3	Evidence	What supports it.
4	Risks / Gaps	What remains uncertain.
5	Next step	What should happen now.

Clear beats clever. Complete beats flashy. Useful beats impressive.

A Strong Default Prompt

Use this when the work matters.

You are helping me as [role].

<context>
I am working on [larger goal] for [audience].
They need [what the final output should help them understand, decide, create, improve, or do].
Success means [define what good looks like].
</context>

<source_material>
[Paste the relevant context, notes, files, excerpts, links, data, examples, or decision history.]
</source_material>

<task>
[State the exact task.]
</task>

<constraints>
- Scope: [what is included and excluded]
- Tone: [plain, executive, technical, warm, concise, direct, etc.]
- Length: [target length]
- Must include: [required elements]
- Must avoid: [things to avoid]
- Do not add extra sections, abstractions, or tangents unless necessary.
</constraints>

<examples>
[Optional: paste one good example.]
[Optional: paste one counterexample and explain what is wrong with it.]
</examples>

<how_to_work>
Ask clarifying questions only if the missing information would materially change the result.
When you have enough information to act, act.
If there are tradeoffs, recommend the best path and briefly explain why.
Pause only for destructive or irreversible actions, real scope changes, or information only I can provide.

</how_to_work>

<evidence_standard>
Before making factual claims, check them against the provided context,
source material, tool results, or reliable references.
Separate verified facts from assumptions, interpretations, gaps, and risks.
If something is unsupported, say so clearly.
Do not present guesses as facts.
</evidence_standard>

<output_format>
Open with the outcome.
Then provide:

1. Final deliverable
2. Key decisions or rationale
3. Evidence, assumptions, risks, or gaps
4. Recommended next step

</output_format>

Short Version for Everyday Use

Use this when the task is smaller.

I am working on [goal] for [audience]. They need [desired outcome].

Task:
[What I want.]

Context:
[Relevant context.]

Reference:
[Optional example of what good looks like.]

This is a [routine / hard / very hard] task. Scope your effort accordingly.

Ask questions only if the missing information would materially change the result. When you have enough information, act.

Do the simplest thing that works well. Avoid extra features, tangents, or unnecessary complexity.

Verify factual claims against the context or reliable sources.
If something is uncertain, say so.

Final output:

1. Start with the outcome.
2. Provide the deliverable.
3. Include assumptions, risks, or gaps.
4. End with the recommended next step.

Developer / API Notes

Most people can ignore this section. It matters if you are putting Fable into a product, tool, internal workflow, or automated system.

Before running the prompt, decide these outside the prompt:

Runtime choice	What to decide
Model	Is Fable the right model, or is a faster or cheaper model enough?
Effort	Does the task need routine, hard, or very hard work?
Output budget	Is there enough room for the answer, evidence, and final report?
Fallback	What should happen if the request is refused or narrowed?
Monitoring	Will you track accuracy, latency, cost, refusal rate, and format compliance?
Evaluation	What test cases prove the prompt is working?

Do not bury these choices inside a clever prompt. They are part of the operating system around the prompt.

A Five-Minute Prompt Upgrade

Before sending a prompt, improve it with these five questions:

What does good mean here?
Who is the audience?
What context would change the answer?
What should Fable not do?
How will I know whether the result is true, useful, and complete?

A better prompt is not longer by default. A better prompt is clearer.

A One-Variable Rule

When a prompt fails, do not rewrite everything at once. Change one variable:

- Add a better example.
- Remove irrelevant context.
- Tighten the scope.
- Clarify the audience.
- Change the output format.
- Raise or lower effort.
- Add an evidence requirement.
- Add a stop rule.
- Add a counterexample.

If you change everything, you learn nothing.

An Evaluation Checklist

After Fable answers, score the result:

Question	Pass?
Did it answer the actual question?	
Did it use the provided context?	
Did it stay in scope?	
Did it avoid unnecessary complexity?	
Did it separate facts from assumptions?	
Did it cite or support important claims?	
Did it ask only necessary questions?	
Did it give a clear next step?	
Was the output worth the time and cost?	

Prompting improves when evaluation becomes visible.

Privacy and Source-Confidence Note

Do not paste confidential, proprietary, client, legal, medical, financial, or personal material into any AI system unless you understand the account settings, data-retention policy, permissions, and risk.

Source-confidence note: Treat model-specific claims in this guide as provisional. Fable is new, and early behavior may change quickly. The durable lesson is not a trick for one model. It is the discipline of clear delegation: define success, give the right context, set boundaries, require evidence, and review the result.

A Real Shift

Prompting is no longer just asking better questions. It is delegation.

Good delegation has a pattern:

- Define the goal.
- Give the context.
- Set the boundaries.
- Require evidence.
- Review the result.
- Improve one thing at a time.

The best AI users will not be the people with the fanciest prompts. They will be the people who know how to brief clearly, review honestly, and decide what work is worth handing over in the first place.

Sources Consulted

Anthropic Prompt Engineering Overview: This supported the point that prompt work should begin with success criteria, evals, and a first-draft prompt. [Link]
Anthropic Define Success Criteria and Build Evaluations: This supported the operating loop: define success, build evals, test, refine, and validate. [Link]
Anthropic Prompting Best Practices: This supported the use of clear instructions, structured prompts, examples, XML tags, roles, and format control. [Link]
Anthropic Prompting Claude Fable 5: This was the key Fable-specific source for effort, autonomy, guarded-domain refusals, fallback, scope, memory, and progress-claim grounding. [Link]
Anthropic Models Overview: This supported the runtime layer: model choice, model capabilities, output limits, context window, and cost tradeoffs. [Link]
Anthropic Building with Extended Thinking: This supported the distinction between useful rationale/evidence and asking for hidden reasoning. [Link]
The Prompt Report: This gave a broad research foundation for treating prompting as a structured family of techniques rather than prompt folklore. [Link]
Chain-of-Thought Prompting: This supported the idea that structured reasoning can help complex tasks, while the final guide translated that into visible evidence, assumptions, and verification instead of hidden reasoning. [Link]

© 2026 Larry Todd Wilson, ltwilson.com