Prompt Engineering / AI Agents

Force the agent to review its own actions, identify mistakes, and plan corrections before next step.
Difficulty: Advanced
Model: GPT-4 / Claude / Gemini
Use Case: Self-Correcting Agents, Autonomous Workflows, Quality Control
Updated: May 2026
Why This Prompt Exists
Agents act, make mistakes, and keep going — because no one told them to check their work. A reflection step catches errors before they compound.

You get:

- agents that confidently proceed with wrong information
- mistakes that compound (error in step 2 ruins steps 3-10)
- no opportunity for self-correction
- agent outputs that require human verification anyway (defeating the purpose)
- no learning from past mistakes within the same task

But reflection changes behavior:

- action review: what did I just do?
- outcome assessment: did it work as expected?
- error identification: what went wrong?
- correction planning: what should I do differently?
- re-execution: try again with corrections

Without reflection, agents don’t learn from their own mistakes.

This prompt adds a structured reflection loop to any agent workflow.

The Prompt

```
Assume the role of an agent reflection architect who adds self-checking loops.

Your task is to design a reflection prompt that forces agents to review their actions.

Generate:

1. REFLECTION TRIGGERS
   - After every action (expensive but thorough)
   - After task completion (lightweight, catches final errors)
   - When confidence is low (adaptive)
   - When output seems suspicious (anomaly detection)

2. REFLECTION DIMENSIONS
   - Did the action produce the expected result?
   - Were there any errors or warnings?
   - Is the output internally consistent?
   - Does the output match any known constraints?
   - What assumptions was I making that might be wrong?

3. REFLECTION PROMPT STRUCTURE
   - Context reminder (what the agent was trying to do)
   - Action summary (what the agent did)
   - Self-assessment (what the agent thinks)
```

   - Confidence score (1-10)
   - Correction plan (if needed)
   - Re-execution or proceed

4. SAMPLE REFLECTION PROMPT
   - Ready-to-use copy-paste reflection block

5. INTEGRATION POINTS
   - Where to insert reflection in the agent loop
   - How to handle reflection findings (retry, escalate, fail)

6. TRADE-OFFS
   - Accuracy improvement vs. latency increase
   - When to skip reflection (simple tasks)
   - When reflection is mandatory (high-stakes tasks)

INPUTS:

Agent task type:
[RESEARCH / CODING / CUSTOMER SUPPORT / DATA PROCESSING / OTHER]

Error tolerance:
[LOW (must be correct) / MEDIUM / HIGH (fast is better)]

Current agent architecture:
[REACT / PLAN-EXECUTE / MULTI-AGENT / OTHER]

Reflection budget (extra tokens/time allowed):
[E.G., "Up to 20% overhead"]

```
RULES:
- Reflection should catch errors before they propagate, not just at the end
- Keep reflection prompts concise (reflection shouldn't double task time)
- Flag tasks that can't benefit from reflection (purely creative)
- Include a "proceed anyway" escape for low-confidence but correct outputs
- Test reflection on known failure cases first
```

How To Use It

- Add reflection to any agent working on high-stakes tasks (code generation, customer support, data analysis).
- Use “after every action” for tasks where errors compound (e.g., multi-step planning).
- Use “after task completion” for tasks where only final output matters (e.g., summarization).
- Monitor reflection overhead — if it adds >30% latency, consider lighter reflection.
- Save reflection outputs to train better agent prompts (the errors reveal prompt weaknesses).

Example Input

Agent task type:
“Code generation and testing”

Error tolerance:
“LOW — code must run without errors”

Current agent architecture:
“REACT”

Reflection budget:
“Up to 20% overhead”

Why It Works
Most agents execute blindly — they don’t check their work because the prompt doesn’t ask them to.

This framework improves outcomes by forcing:

- reflection triggers (when to check)
- reflection dimensions (what to check)
- structured self-assessment (not just “looks good”)
- correction planning (how to fix errors)
- integration guidance (where to insert reflection)

Great agent reflection doesn’t eliminate mistakes — it catches them before they matter.

Build Better AI Systems

Subscribe for advanced prompt engineering, AI coding tools, debugging frameworks, and practical strategies for developers and engineers.

Carefully engineered prompts for people doing real work.

Share this:

- Share on Facebook (Opens in new window) Facebook
- Share on X (Opens in new window) X

See also Multi-Agent Handoff Designer