Prompt Engineering / AI Agents

Design protocols for when Agent A should pass a task to Agent B — triggers, information to pass, fallbacks.
Difficulty: Advanced
Model: GPT-4 / Claude / Gemini
Use Case: Multi-Agent Systems, Orchestration, Task Routing
Updated: May 2026
Why This Prompt Exists
Multi-agent systems fail not because individual agents are weak, but because handoffs are messy — dropped context, unclear triggers, no fallbacks.

You get:

- agents that don’t know when to hand off (keep doing tasks they’re bad at)
- handoffs that lose context (next agent starts from scratch)
- dropped tasks (no one is responsible after handoff)
- handoff loops (A passes to B, B passes back to A, repeat)
- no standard handoff protocol (every agent implements differently)

But good handoffs have structure:

- triggers: when should handoff happen? (task type, confidence, error, completion)
- information package: what must be passed? (context, partial results, constraints)
- protocol: synchronous (wait for response) vs. asynchronous (fire and forget)
- fallbacks: what if target agent fails or is unavailable?
- escalation: when should a human be involved?

Without handoff design, multi-agent systems collapse.

This prompt designs reliable handoff protocols between agents.

The Prompt

```
Assume the role of a multi-agent orchestration architect who designs
handoff protocols.

Your task is to specify when and how agents should transfer tasks to
each other.

Generate:

1. AGENT ROLES
   - Agent A: [description, strengths, weaknesses]
   - Agent B: [description, strengths, weaknesses]
   - (Additional agents as needed)

2. HANDOFF TRIGGERS
   - Task type boundary (e.g., "research → writing")
   - Confidence threshold (e.g., "Agent A confidence < 0.6 → handoff")
   - Error condition (e.g., "Agent A fails twice → handoff")
   - Task completion (e.g., "Agent A finishes subtask → handoff for
next subtask")
   - Escalation (e.g., "Agent A unsure → handoff to human")

3. HANDOFF PROTOCOL
   - Information to pass:
     * Task description
     * Completed work
```

```
     * Context (conversation history, constraints)
     * Partial results
     * Known failures
   - Format: [Structured JSON / Natural language / Hybrid]

4. COMMUNICATION PATTERN
   - Synchronous (A waits for B's response before continuing)
   - Asynchronous (A hands off and moves to next task)
   - Supervised (Orchestrator mediates handoff)

5. FALLBACK & ERROR HANDLING
   - What if B is unavailable? (retry, queue, escalate)
   - What if B fails? (return to A, escalate, terminate)
   - What if A and B disagree? (tie-breaker agent or human)

6. READY-TO-USE HANDOFF PROMPT
   - A prompt template for agents to execute handoffs

INPUTS:

Agent A description (source agent):
[E.G., "Researcher agent — finds information, cites sources"]

Agent B description (target agent):
[E.G., "Writer agent — turns research into prose"]

Task flow:
[E.G., "Research → Write → Review"]
```

```
Reliability requirement:
[STANDARD / HIGH (no failures tolerated) / EXPERIMENTAL]

RULES:
- Handoff should preserve all relevant context (no information loss)
- Include confidence scores in handoff (so next agent knows
uncertainty)
- Design for failure — assume handoffs will fail sometimes
- Avoid handoff loops (detect if A → B → A and break)
- Log all handoffs for debugging (critical for multi-agent systems)
```

How To Use It

- Design handoffs before implementing multi-agent systems — don't discover problems in production.
- Include confidence scores in handoff packages — they help the next agent calibrate trust.
- Always include a fallback (what if the target agent is down?).
- Test handoffs by forcing failures (simulate agent unavailability).
- Monitor handoff logs to detect loops or dropped tasks.

Example Input

Agent A description:
"Research agent — retrieves and summarizes information from the web. Strength: thorough. Weakness: verbose."

Agent B description:
"Writer agent — turns research into concise, readable content. Strength: clear writing. Weakness: can't verify facts."

Task flow:
"Research (Agent A) → Write (Agent B)"

Reliability requirement:
"HIGH — customer-facing content"

Why It Works
Most multi-agent systems are built ad hoc — handoffs happen implicitly, which means they fail unpredictably.

This framework improves outcomes by forcing:

- explicit handoff triggers (when to transfer)
- information packaging (what to pass)
- communication pattern selection (synchronous vs. asynchronous)
- fallback planning (what if handoff fails)
- loop prevention (detect and break cycles)

Great handoff design doesn't just connect agents — it ensures tasks don't fall through the cracks.

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 Agent Reflection Prompt