

AI Agent System Prompt + Full Guide

CTA word: PROMPTS

What most "agent tutorials" skip: tools, memory, and when to stop. This pack is the missing middle - a production-shaped system prompt, three prompts that force the hard decisions, and a short build guide you can paste into Claude, Cursor, or an n8n/LLM node tomorrow.

No fake sample size. No happy-path demo. Steal the structure.

1. The system prompt (paste this)

```
You are an ops agent for {{COMPANY}}. You do not chat. You execute scoped work.

GOAL
{{ONE_SENTENCE_GOAL}}

TOOLS (only these; refuse anything else)
{{TOOL_LIST_WITH_SIDE_EFFECTS}}

MEMORY
- Load: last N decisions, open tickets, and policy snippets for this entity.
- Write: one audit row per action (who/what/why/tool/result).
- Never invent state that is not in tools or memory.

STOP RULES (hard)
1. If the action is irreversible (money, inventory write, customer message send) AND confidence <
{{THRESHOLD}} -> escalate.
2. If required fields are missing after one tool round -> ask one clarifying question, then stop.
3. If the request is outside GOAL -> refuse with a one-line reason.
4. Never chain more than {{MAX_STEPS}} tool calls without a human checkpoint.

OUTPUT
Always return JSON:
{
  "intent": "...",
  "plan": ["..."],
  "actions": [{"tool": "...", "args": {}}],
  "needs_human": true|false,
  "reason": "...",
  "audit": "one line"
}

Tone: blunt. No filler. No apologies.
```

Fill the blanks before you ship. An empty THRESHOLD is how you get a liability.

2. Three prompts that make the build less stupid

Prompt A - Scope the agent (run before you write code)

```
I want an agent that does: {{JOB}}.

Interview me. Ask only the questions that change the design.
Force answers for: tools with side effects, irreversible actions,
who approves those, what "done" means, and what "fail closed" means.
End with a one-page agent brief I can paste into a system prompt.
```

Prompt B - Tool contract (run before you wire APIs)

Here is the agent brief:

{{BRIEF}}

For each tool the agent may call, write a contract:

- name
- inputs / outputs
- side effect (read | write | send | money)
- idempotent? (Y/N)
- failure modes
- what the agent must log

Refuse any tool that has a write/send/money side effect without an approval gate field.

Prompt C - Failure drill (run before go-live)

Here is the system prompt + tool contracts:

{{PROMPT_AND_CONTRACTS}}

Generate 12 adversarial user requests that should:

- escalate
- refuse
- or ask one clarifying question

For each, show the expected JSON output.

Then list the top 3 prompt edits that would have prevented a wrong action.

3. Full build guide (one afternoon shape)

Step 1 - One job

Pick a single recurring ops job (exception triage, inventory check, quote draft). Not "be helpful."

Step 2 - Tools before model

List tools. Mark side effects. Anything irreversible gets `needs\_human: true` by default.

Step 3 - Memory is a table, not vibes

Minimum columns: entity_id, decision, tool, result, actor, ts. If you cannot query it, the agent cannot learn from overrides.

Step 4 - Paste the system prompt

Fill GOAL, TOOL_LIST, THRESHOLD, MAX_STEPS. Leave nothing as "TBD."

Step 5 - Run Prompt C

Fix the three worst failures in the prompt before you give anyone access.

Step 6 - Human gate

Wire approval for money / inventory write / outbound send. Silent success is worse than a loud refuse.

Step 7 - Ship a thin loop

Trigger -> load memory -> plan -> tool (or escalate) -> audit row -> done. No multi-agent theater on day one.

4. Anti-patterns

- Demo videos that never show a refused action

- Agents with open-ended "browse the web" as a tool and no stop rule
- Memory = chat history only
- Escalating everything (or nothing)
- Measuring success by "tasks completed" instead of "wrong irreversible actions = 0"

5. One-week checklist

- [] One-sentence GOAL signed by an owner
- [] Tool contracts written (Prompt B)
- [] System prompt filled (section 1)
- [] 12 failure cases green (Prompt C)
- [] Audit table live
- [] Approval gate live for irreversible classes
- [] Weekly review of overrides -> prompt edits

Pair with GATES (approval) and PLAYBOOK (exceptions) if you are wiring this into a real ops queue.

Team Nebula / teamnebula.ai