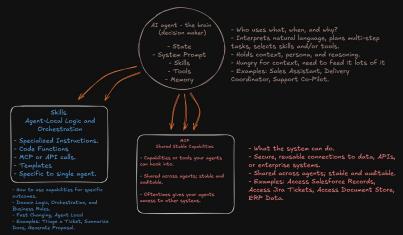
# Agent versus MCP versus Skills

How to think about agents, MCPs, and skills.




## When to separate agents and when to go multi-agent?

### Separate Agents

When do I need to create another agent (separate persona/ runtime)?

Create a new agent when at least one of these is true.

1. Distinct Audience or Interaction Style
   - Exec Briefing vs. Support Chatbot vs. Internal Ops Console
   - If prompts, tone, and context are all different, separate the agent.

2. Distinct Decision Space
   - Agent A should be allowed to create tickets.
   - Agent B should only read knowledge base.
   - If the set of tools it may ever choose is meaningfully different, split.

3. Distinct long-term context.
   - A sales agent that tracks opportunity history versus a delivery agent that tracks project risks.
   - If the context and/or memory you keep would pollute another use case, separate them.

4. Distinct Security and Data Domain
   - Finance vs. HR vs. Customer Data.
   - If you'd write yearly guardrails for one inside the other, it's cleaner to split.

5. Distinct pacing or triggers.
   - A proactive agent that runs on schedules or events versus a conversational agent that reacts to user messages.
   - Different life cycles = different agents.

### Multi-Agent

When to go multi-agent (agents talking to agents)?

Use multi-agent not for fashion, but when you have role separation or task decomposition that's clearer with boundaries.

Good cases:

1. Planner > Specialist
   - Planner Agent interprets the user goal.
   - Specialist Agents (Docs, CRM, Code, Finance) execute.
   - Use this when your skill space is big and you want to avoid giving one agent 40 tools.

2. Guarded or Governed Execution
   - One agent proposes an action ('close opportunity', 'send message').
   - Another agent (or thin reviewer agent) validates or rewrites.
   - That is a multi-agent approval chain.

3. Cross-domain tasks.
   - A Customer 360 experience that needs sales, support, and billing, each with different MCPs and roles.
   - One coordinator agent, several domain agents.

4. Different Reasoning Strengths
   - A fast, cheap agent for retrieval and a slower, better agent for final drafting.
   - Coordinator delegates based on step.

## Anti-patterns to avoid.

1. Stuffing business logic into MCP.
   - Makes the MCP hard to share and version.

2. Calling vendors directly in skills when you already have an MCP.
   - You lose centralization and discoverability.

3. Letting the model see only skills, never raw MCP.
   - Sometimes the model or planner does better if it can see the primitive tools.

Skills should add paths, not hide everything.

## So, Putting it all together. . .

### Mental Model: Agents vs. MCPs vs. Skills vs. Multi-Agent.

Agent: "Given this user/task, what should I do next? How do I feed it as much context as possible?"
MCP: "Here's how to talk to the real world, safely and consistently."
Skill: "Here's a reasonable way to accomplish a domain task with those tools."
Separate Agents: "We have different policies (users, domains, or security) that should not be mashed together."
Multi-Agent Patterns: "We have one policy to pick the right policy."

If you stick with that approach you get:
- MCPs: thin, reusable, secure.
- Skills: thick, fast-changing domain logic.
- Agents: planners/ routers that choose among skills and MCP tools based on as much context as possible.

Quick rule of thumb:
- Capability = MCP.
- Strategy or business workflow = skill.

## TL;DR

MCP = Capabilities → Skills = Behavior → Agents = Decision-Makers → Multi-Agents = Collaboration.

Build from the bottom up, design from the top down.

