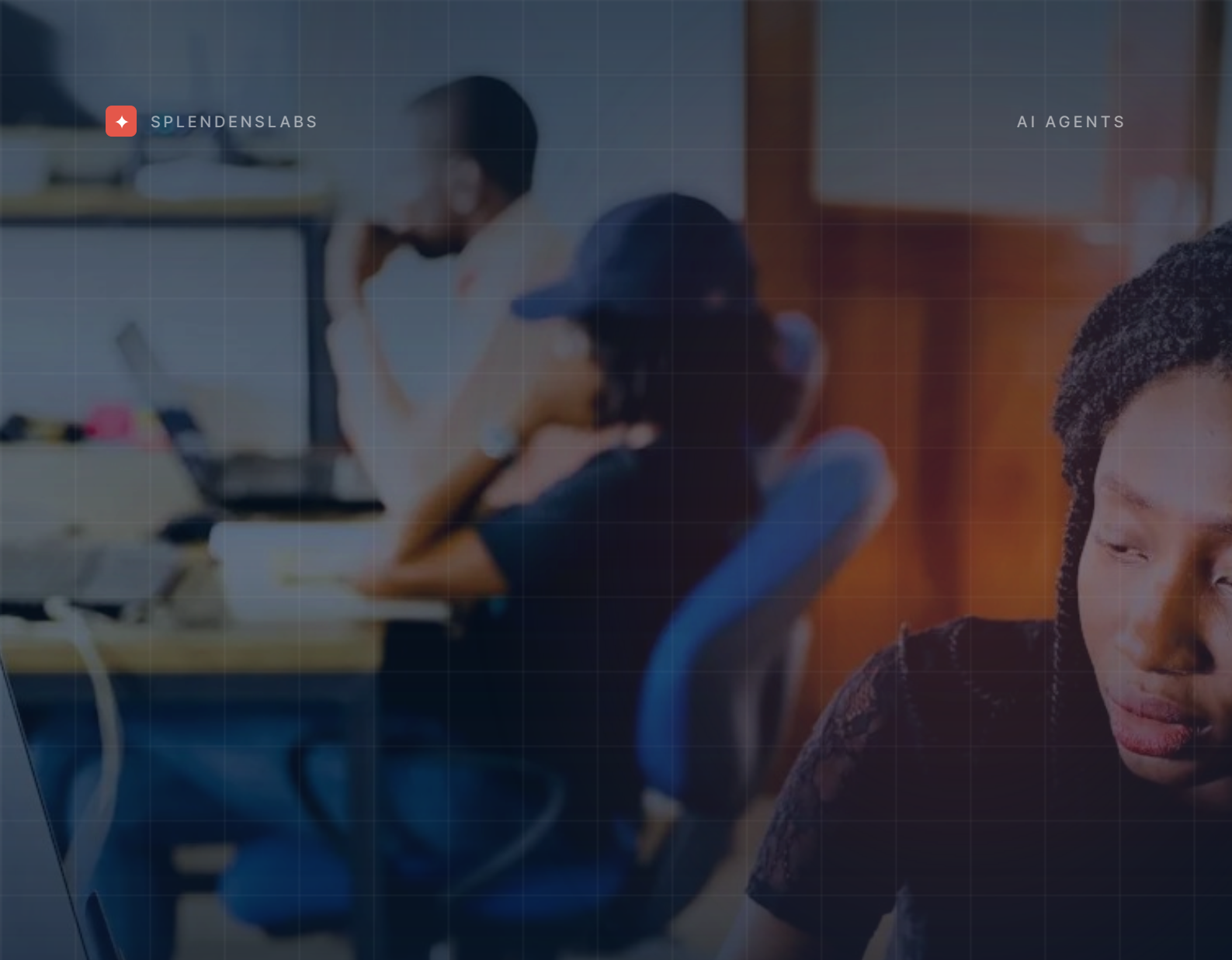


✦ COMPARISON · AI AGENTS

MCP vs Traditional APIs

How the Model Context Protocol compares with wiring REST APIs into agents through direct function calling — and how to use them together securely.

FORMAT	LEVEL	READ TIME	UPDATED
Comparison	Intermediate	10 min	Oct 2026

OVERVIEW

What this comparison covers

The Model Context Protocol (MCP) is an open standard, introduced by Anthropic in November 2024, for connecting AI applications to tools and data. Instead of writing a custom function-calling integration for every API in every application, a system exposes an MCP server once and any MCP-capable client can use it. MCP does not replace your REST APIs — servers usually wrap them. This comparison explains what MCP adds, where direct integration is still the better choice, how to combine the two, and the security questions both raise.

KEY TAKEAWAYS

- MCP standardizes how AI applications discover and call tools, read resources and use prompt templates.
- Direct function calling against your APIs is simpler when one application needs a few tools you control.
- MCP pays off when many clients need the same integrations, or you want to use third-party servers.
- MCP servers typically wrap existing APIs; it is a layer for AI clients, not a replacement for your API design.
- Treat every tool — MCP or not — as untrusted input and a privileged action: least privilege, approvals and logging.

WHO IT'S FOR

Engineers building agents and AI assistants

Platform and API teams

Architects defining integration standards

Security reviewers

HOW TO USE IT

Read section 01 for an accurate picture of MCP, compare in section 02, then use sections 03–05 to choose an approach and section 06 as the security review checklist.

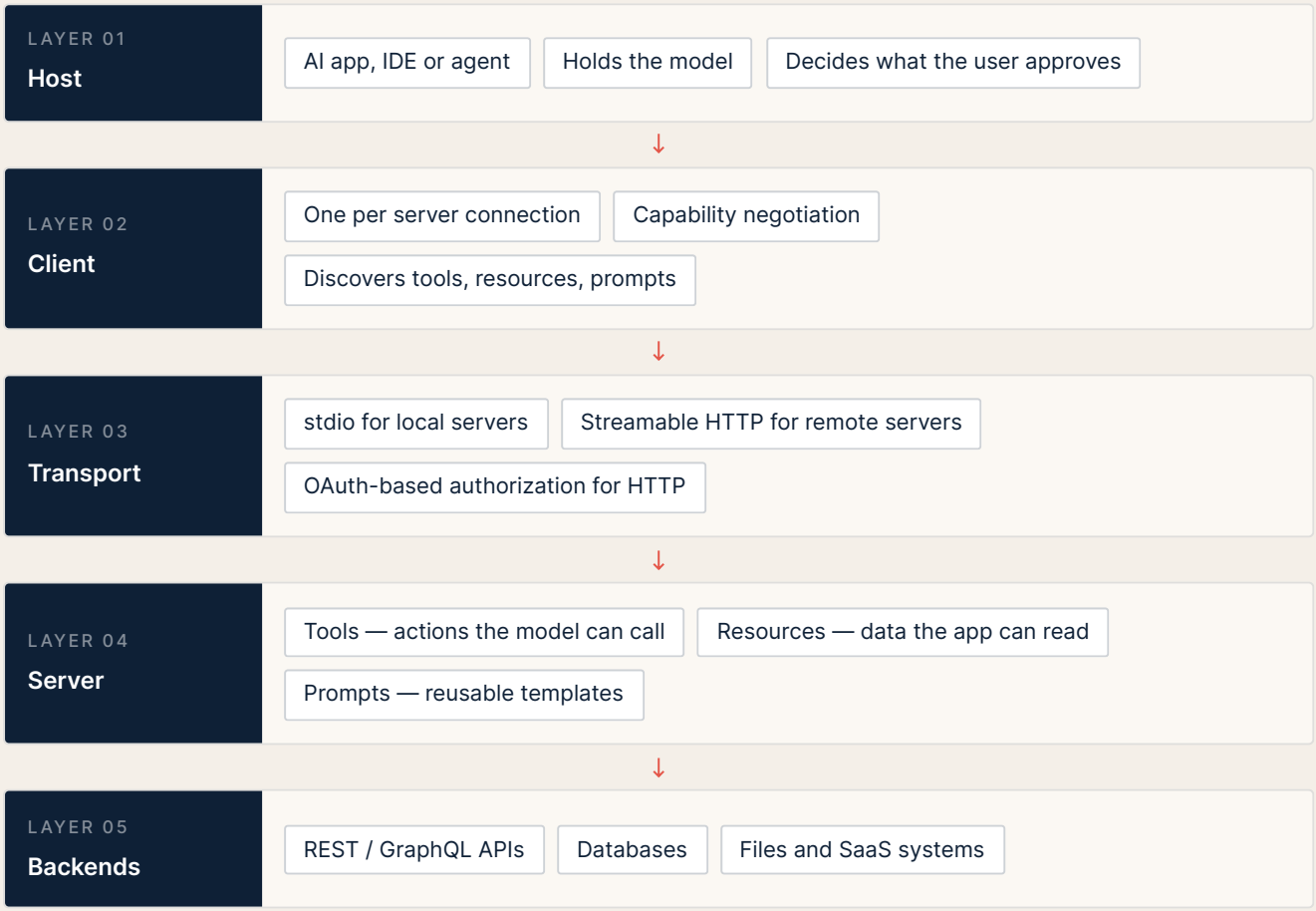
CONTENTS

01 What MCP actually is
02 MCP vs direct API integration
03 When to choose which
04 Using them together
05 Security considerations
06 Decision checklist

01 / BASICS

What MCP actually is

MCP defines a client–server protocol over JSON-RPC 2.0. A host application (an IDE, chat app or agent) runs one MCP client per server it connects to; each server exposes capabilities in a standard shape.



SERVER PRIMITIVE

Tools

Named functions with a JSON Schema for inputs that the model can choose to invoke.

SERVER PRIMITIVE

Resources

Addressable data such as files or records that the application can read into context.

SERVER PRIMITIVE

Prompts

Parameterized templates a server offers, typically surfaced for users to pick.

Clients can also offer capabilities back to servers, such as sampling (a server asking the host's model for a completion) and elicitation (a server asking the user for input). The specification is versioned by date; check which version your client and server support.

> MCP standardizes how tools are connected. It does not decide which tools an agent should have — that is still your job.
>
> SPLENDENSLABS ENGINEERING PRINCIPLE

✦ 02 / SIDE BY SIDE

MCP vs direct API integration

“Traditional” here means the application defines function-calling tools itself and calls REST APIs directly from its own code.

DIMENSION	DIRECT API + FUNCTION CALLING	MCP
Integration unit	Per application, per API	One server, reusable by any MCP client
Tool discovery	Hard-coded in the app	Listed by the server at run time
Schema definition	Written in each app’s tool format	JSON Schema served by the server
Model and vendor coupling	Tied to one SDK’s tool format	Protocol is model-agnostic
Deployment	Library code inside the app	Separate local or remote process
Auth	Your existing API auth	OAuth-based for HTTP servers; local servers inherit the environment
Ecosystem	Only what you build	Third-party servers for many common systems
Moving parts	Fewer	Extra process, transport and protocol version

DIRECT API INTEGRATION	MCP
Simplest path for a few tools in one app	Build once, use from many clients
Full control of calls and errors	Dynamic discovery of tools and data
No extra runtime component	Growing ecosystem of servers
Duplicated work across apps and models	Another component to secure and version

✦ 03 / DECISION

When to choose which

The question is not which is better, but how many clients need the integration and who controls both ends.

CHOOSE DIRECT

One app, a few tools

A single product calling a handful of your own APIs gains little from an extra protocol layer.

CHOOSE DIRECT

Tight latency or control needs

Hot paths where you want no extra hop and full control of retries and errors.

CHOOSE DIRECT

Fixed, audited workflows

Deterministic steps where the model should not discover new tools at run time.

CHOOSE MCP

Many clients, same systems

Several assistants, IDEs or agents need the same internal integrations.

CHOOSE MCP

Using off-the-shelf servers

You want to plug into existing MCP servers rather than build each connector.

CHOOSE MCP

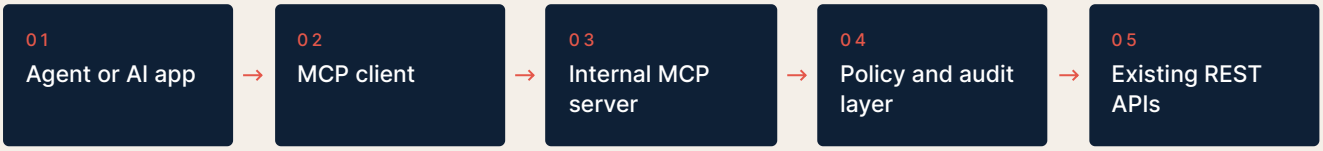
Platform teams serving AI apps

A central team wants to publish governed tool access for internal AI applications.

04 / HYBRID

Using them together

In practice MCP sits on top of your APIs, not instead of them. The API stays the source of truth for business rules; the MCP server is an AI-friendly façade.



01 **Keep business logic in the API**

Validation, authorization and side effects stay in the service; the MCP server translates.

02 **Design tools for models, not endpoints**

Expose task-level tools with clear descriptions instead of mirroring every endpoint one to one.

03 **Return compact, relevant results**

Shape responses to fit a context window — filtered fields and pagination, not raw payloads.

04 **Use direct calls where they fit**

Deterministic workflow steps can still call the API directly; the agentic parts use MCP.

05 / SECURITY

Security considerations

Giving a model tools gives it the ability to act. These risks apply to any tool integration, and some are specific to MCP's dynamic, third-party ecosystem.

RISK	WHAT HAPPENS	MITIGATION
Prompt injection via tool results	Content returned by a tool carries instructions the model follows	Treat tool output as data; require approval for sensitive follow-up actions
Malicious or poisoned tool descriptions	A server's tool metadata manipulates the model	Vet and pin server versions; review descriptions; allow-list servers

RISK	WHAT HAPPENS	MITIGATION
Over-broad permissions	A tool can do far more than the task needs	Least-privilege credentials and narrowly scoped tools
Token passthrough / confused deputy	A server forwards or misuses credentials meant for another audience	Validate token audience; follow the spec's authorization guidance
Untrusted local servers	A stdio server runs with the user's local access	Install only reviewed servers; sandbox where possible

✦ DESIGN RULE

The host decides, not the server

Consent and approval for consequential tool calls belong in the host application, enforced in code and shown to the user — never delegated to the server or the model.

✦ 06 / CHECKLIST

Decision checklist

Use this before adopting MCP or adding a new direct integration.

FIT

- [ ] Number of AI clients needing this integration is known
- [ ] Existing MCP servers for the system have been evaluated
- [ ] Latency and control requirements are stated

DESIGN

- [ ] Tools are task-level with clear descriptions and schemas
- [ ] Business rules remain in the underlying API
- [ ] Supported MCP specification version is recorded

SECURITY

- [ ] Servers are allow-listed and versions pinned
- [ ] Credentials are least-privilege and audience-checked
- [ ] Consequential actions require approval in the host
- [ ] Tool output is treated as untrusted input

OPERATIONS

- [ ] Every tool call is logged with inputs and caller
- [ ] Servers are monitored and owned by a named team
- [ ] Evals cover tool selection and misuse cases

✦ GET IN TOUCH

Let's Talk

We design agent tool layers — MCP servers, direct integrations or both — with the permissions, approvals and audit trails production needs.

- ✦ Integration inventory and MCP fit review
- ✦ MCP server design over your existing APIs
- ✦ Tool security and permission review
- ✦ Agent pilot on your real systems

BOOK AN AI STRATEGY SESSION →

EMAIL
hello@splendenslabs.com

WEB
splendenslabs.com

OFFICE
Sona Incubation Foundation, Salem