17 — The Protocol Layer

🕒 state of the art 2026-07 · revised 2026-07-31 · 📖 ~9 min read

Learning objectives

By the end of this chapter, you should be able to:

1. **Explain** why the protocol layer is what turns a market of silos into an ecosystem — and why each protocol standardizes a different *boundary* of the harness;
2. **Distinguish** the boundaries covered by MCP (Model Context Protocol), A2A (Agent-to-Agent) and ACP (Agent Client Protocol), agentskills.io and AGENTS.md — including the two classic confusions (the two "ACP"s; MCP × A2A as vertical × horizontal);
3. **Analyze** the adoption matrix measured in code and locate a real harness in it;
4. **Evaluate** a protocol's health by measured adoption and governance (neutral foundation × single vendor), rather than by marketing;
5. **Decide** which protocols a new harness needs to speak so as not to be left out of everyone else's composition architectures.

The problem

Chapters 02–16 deal with what happens *inside* a harness. This chapter deals with what happens *between* them — and between harnesses and the rest of the world. Without shared protocols, each harness is a silo: its tools, its project instructions, its subagents and its skills only work inside it. The protocol layer is what turns that market of silos into an ecosystem: each protocol standardizes a different boundary of the harness — agent ↔ tool, agent ↔ agent, agent ↔ editor, agent ↔ user, plus the cross-cutting formats for procedural knowledge (SKILL.md) and project instructions (AGENTS.md).

The practical consequence: in a market that *composes* harnesses, not speaking the protocols is not missing a feature — it is being left out of everyone else's architectures.

The state of the art

The map: one protocol per boundary

The map, organized by the boundary each one solves:

Protocol	Boundary	Origin / governance	Status (2026)
MCP (Model Context Protocol)	agent ↔ **tools/data**	Anthropic → universal adoption (OpenAI, Google, Microsoft)	mature; ~97M downloads
A2A (Agent-to-Agent)	agent ↔ **agent** (delegation across organizations)	Google → **Linux Foundation** (v1.0 in 2026)	consolidating; absorbed IBM's ACP (Agent Communication Protocol)
ACP (Agent Client Protocol)	agent ↔ **editor/client**	Zed	rapid adoption among coding harnesses
agentskills.io (Agent Skills / SKILL.md)	portable **procedural knowledge**	Anthropic (open spec, Dec 2025)	~40 compatible products in 6 months
AGENTS.md	portable **project instructions**	community → **Agentic AI Foundation** (Linux Foundation)	60,000+ repositories; 20+ tools read it natively
AG-UI	agent ↔ **user interface**	community (CopilotKit)	emerging
ACP-IBM (Agent Communication Protocol)	agent ↔ agent	IBM	**discontinued** — merged into A2A (Aug 2025)

Two confusions to clear up: (1) "ACP" names two distinct protocols — IBM's (agent-agent communication, discontinued in favor of A2A) and Zed's (agent-editor, alive and expanding); in this book, ACP = Zed. (2) MCP and A2A do not compete: MCP is the *vertical* connection (agent→tool), A2A is the *horizontal* one (agent→peer agent) — a real system uses both.

The adoption matrix — measured in code, not in marketing

This chapter's differentiator: we cross the protocols with the **11 harness evaluations from the benchmark** (plus the 4 frameworks from the frameworks-1 round) (evidence per file, see `benchmark/avaliacoes/`). No external comparison has this column of truth:

Harness	MCP client	MCP server	ACP	A2A	SKILL.md / agentskills	AGENTS.md (or equiv.)
opencode	✅	—	✅ (Zed)	—	partial	✅ AGENTS.md
gemini-cli	✅	—	✅	✅ **client+server**	✅	GEMINI.md
OpenHarness	✅	—	—	—	✅ (Claude format)	CLAUDE.md
Codex CLI	✅	✅	—	—	✅	✅ AGENTS.md
Goose	✅	✅ (`goose mcp`)	✅ (entire desktop)	—	✅	✅ AGENTS.md + .goosehints
Aider	❌	❌	—	—	—	✅ (reads it)
OpenHands	✅	✅ (FastMCP)	✅ (profiles)	—	✅ (org repos)	microagents
OpenClaw	✅	✅	✅ (orchestrates third parties)	—	✅ (52 bundled)	✅ AGENTS.md + SOUL.md
Hermes	✅	✅	✅	—	✅ (core of its learning)	✅ AGENTS.md + SOUL.md
IronClaw	✅	—	—	—	✅ (OpenClaw compat)	identity files
n8n	✅	✅ (Trigger)	—	—	—	—
frameworks:						
LangGraph	❌	❌ (paid server only)	❌	❌	❌	—
OpenAI Agents SDK (Software Development Kit)	✅	—	❌	—	partial	sandbox agents only
CrewAI	✅ (mandatory)	—	✅ **client+server**	—	✅	✅ **auto-generated**
software-agent-sdk	✅ (OAuth)	—	❌	✅ (uses harnesses as engine)	✅ (spec)	✅

Readings of the matrix:

1. **MCP has won, in fact**: 10 out of 11 (the exception, Aider, is a philosophical choice). And between rounds 1 and 2, the pattern migrated from "client" to "client+server" — the harness as a consumable service.

2. **agentskills.io is the fastest standardization we have ever measured**: a spec from December 2025, 8 of our 11 compatible by July 2026. Chapter 12's prediction ("an MCP of extensibility is taking shape") came true — and with a structural detail: skills are portable markdown, so the same skill runs on Claude Code, on Hermes and on IronClaw. Self-improving learning (ch. 16) writes in *that* format — the knowledge one agent learns is, in theory, transferable to another.
3. **ACP is the cohort's most important silent protocol**: 6 out of 11 speak it, and three harnesses (OpenClaw, OpenHands, Goose) use it to **orchestrate other harnesses** as subagents — Claude Code, Codex, Gemini CLI and opencode become interchangeable parts. What used to be "agent ↔ editor" has become, in practice, the composition bus between harnesses.
4. **A2A has left "one player's bet" territory** *(updated in the frameworks-1 round)*: gemini-cli was the only harness to implement it, but **CrewAI** came in with native client AND server (full AgentCard, JWS, gRPC/REST) — the second measured implementer, and the first framework. Governance at the Linux Foundation and the absorption of ACP-IBM keep pointing to A2A as the candidate for the inter-organizational boundary; in product harnesses, however, that boundary still barely exists.
5. **AGENTS.md has consolidated as the neutral standard**: the AGENTS/CLAUDE/GEMINI.md fragmentation of ch. 03 is resolving itself — Codex, Goose, opencode, OpenClaw and Hermes have already converged on AGENTS.md (now under the Agentic AI Foundation), with the proprietary files becoming aliases.

The stack: how the protocols compose

A complete agentic system in 2026 uses the whole stack, one layer per boundary:

```
[user]
   │  AG-UI / chat channels / TUI            (interface)
[harness A]
   │  ACP                                    (composition: A drives B as a subagent)
[harness B]
   │  A2A                                    (delegation to another organization's agent)
[remote agent]
   │  MCP                                    (each agent reaches its tools)
[tools/data]

cross-cutting: AGENTS.md (per-project instructions) · SKILL.md (portable procedures)
```

Implications for harness engineering

1. **Protocol is a survival dimension, not a feature dimension**: Aider, a technical reference in three dimensions, is outside the entire composition ecosystem for not speaking MCP/ACP. In a market that composes harnesses, not speaking the protocols means being left out of everyone else's architectures.
2. **The expiration clause does not apply here** (ch. 14): protocols are the boundary with the world — the scaffolding that *remains* when models improve. Investing in protocol is the harness investment with the longest half-life.
3. **For the benchmark**: the matrix above becomes a permanent section of the comparative, updated each round. Protocols do not receive a 0–3 score like harnesses — they are evaluated by **measured adoption**

(the matrix) and **governance health** (neutral foundation > single vendor).

Addendum (2026-07-31): the MCP **2026-07-28** spec (announcement) reinforces this chapter's thesis from another angle: a stateless core, an extension framework and the **first formal deprecation policy** (12 months) are the typical behavior of a protocol leaving adolescence and entering its infrastructure phase — disciplined versioning matters more than features. The cohort's adoption of the new version enters the matrix next round. And the same-day confirmation on the other boundary (spec 065): the A2A specification confirms the **stable v1.0 under the Linux Foundation**, organized in three layers (data model in Protobuf/JSON Schema, abstract operations, JSON-RPC/gRPC/REST bindings), with **v1.0.1 already bringing a formal extension mechanism** — the two boundary winners reached, in the same quarter, the same stage: formal extensions instead of features in the core.

> **EXECUTIVE SUMMARY**
>
> The protocol layer already has one winner per boundary: MCP on the vertical (agent→tool, near-total adoption), ACP as the composition bus between harnesses, agentskills.io as the portable format for procedural knowledge and AGENTS.md as the neutral standard for project instructions — while A2A remains the consolidating bet for the inter-organizational boundary, sustained more by governance (Linux Foundation, absorption of ACP-IBM) than by measured adoption in product harnesses. The engineering decision is asymmetric: protocols are the harness component with the longest half-life, immune to the expiration clause, and the adoption matrix — not marketing — is the instrument for re-evaluating them each benchmark round.

Industry sources

- ecosystem map 2026
- Zylos: MCP/A2A/ACP convergence
- Zuplo: where ACP ended up
- Agent Skills: format and adoption
- AGENTS.md guide 2026
- Zed ACP

Adoption matrix: the benchmark's own evidence (`benchmark/avaliacoes/`).

- **See also**: the living collection Awesome Harness Engineering — Skills & MCP gathers more consultable resources for this dimension (patterns, articles and implementations), curated by problem.

Check your understanding

1. A colleague claims that "A2A will replace MCP". Why does the claim confuse the boundaries, and how does the stack show that a real system uses both? (Re-read "The map" and the diagram.)

2. "ACP" appears twice in the protocol table, with opposite statuses ("rapid adoption" and "discontinued"). Explain the difference between the two protocols — and which of them this book calls ACP.
3. You are designing a new harness. Based on the matrix readings and the implications, which protocols are mandatory today, which one is still a bet, and what does the Aider exception teach about the cost of speaking none?
4. Why does the expiration clause (ch. 14) not apply to the protocol layer, when it applies to almost everything else in the harness?