Knowledge Discovery & Attribution Using Agent Skills + MCP

Ola Hungerford
Bellagio - April 2026

Supporting attribution-first agent infrastructure

- AI agents increasingly rely on specialized human knowledge at inference time
- The infrastructure delivering that knowledge often relies on some combination of agent skills + MCP
- These standards and protocols need attribution baked in, to avoid reinforcing an ‘extraction economy’ status quo set by scraping
- How might MCP and Skills be extended to support knowledge discovery and attribution in AI applications?

What Are Agent Skills?

- Skills are files that get loaded on demand by the model
- AI agents = ‘Brains, books, and tools:’
  - Skills are the books
  - Skills can also include scripts

What Are Agent Skills?

- Skills get invoked on demand by the model, instead of being automatically loaded into context
- Require standard metadata in frontmatter: Name, description
- These are the default parts that the model sees

Minimal example:

SKILL.md

```
---
name: skill-name
description: A description of what this skill does and when to use it.
---
```

What Are Agent Skills?

- Also supports license and optional metadata
- The body of the skill file includes instructions to load when the agent invokes the skill, and references to any other files, if needed

SKILL.md

```
---
name: pdf-processing
description: Extract PDF text, fill forms, merge files. Use when handling PDFs
license: Apache-2.0
metadata:
  author: example-org
  version: "1.0"
---
```

Agent Skills vs Tool Calls

- Skills are designed to use **progressive disclosure**:
  - Agents load files referenced skill progressively, pulling in more detail only as a task calls for it.
- ‘Reading the table of contents’
- Versus tools: by default, all connected tools and their descriptions are usually loaded up front by an MCP host.
  - Bloats context with tools that may not be needed
  - In practice, this is now often being optimized in the host application (using tool search, dynamic loading, etc.)

How to discover and expose Agent Skills to a model?

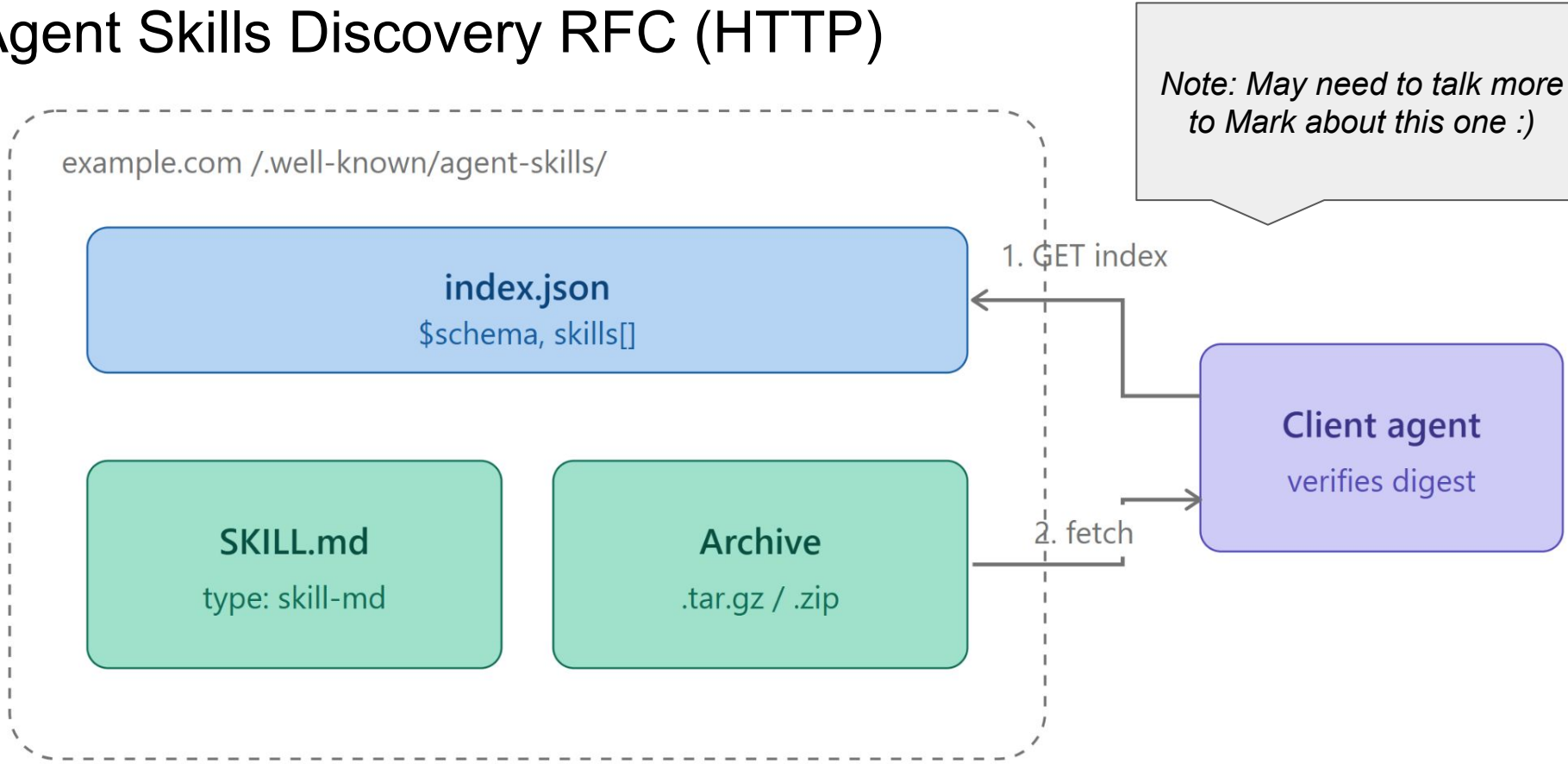
What exists today:

- Centralized catalogs: Vercel skills.sh
- GitHub repository-scoped plugin marketplaces
  a. Public (Example: official Claude plugins)
  b. Internal (Example: enterprise repositories)

New ways:

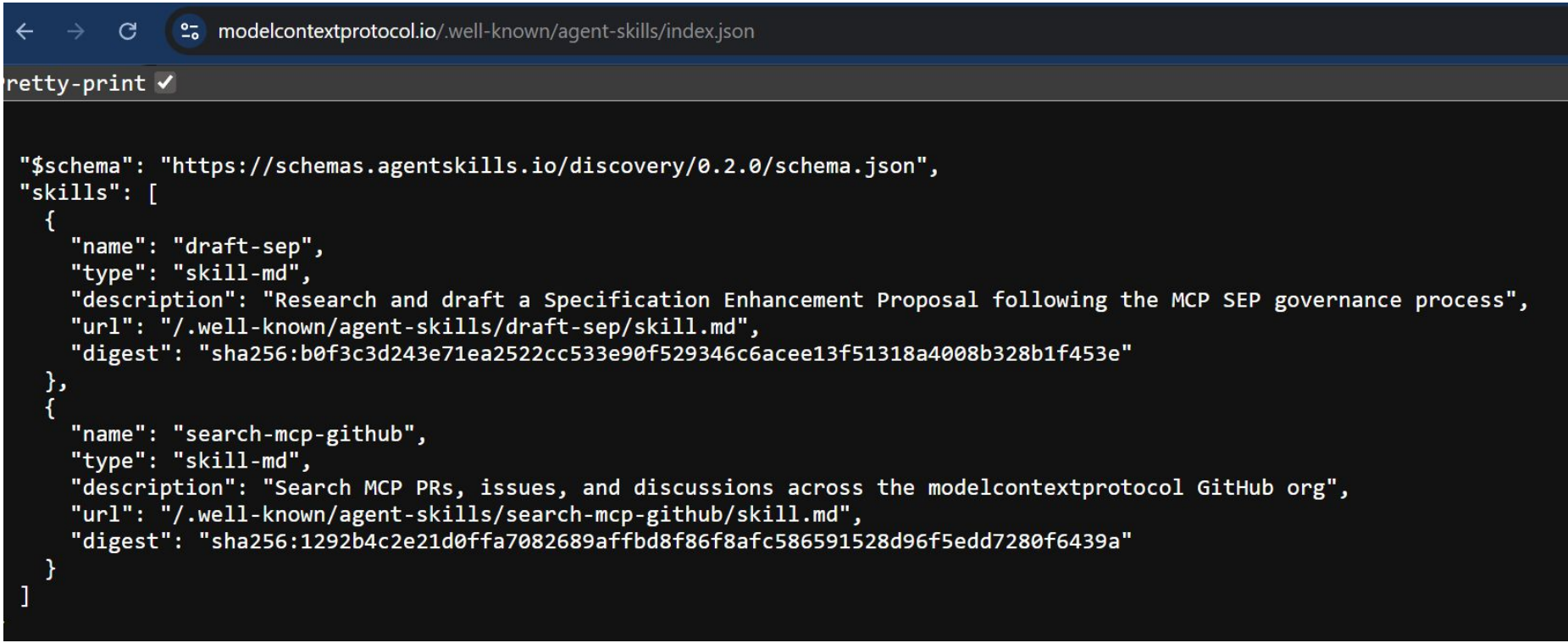
- Agent Skills Discovery RFC (HTTP)
- Skills Over MCP

Agent Skills Discovery RFC (HTTP)



Matt Silverlock (Cloudflare)
Jonathan Hefner (MCP maintainer, Agent Skills maintainer)

Example: Agent Skills Discovery RFC (HTTP)

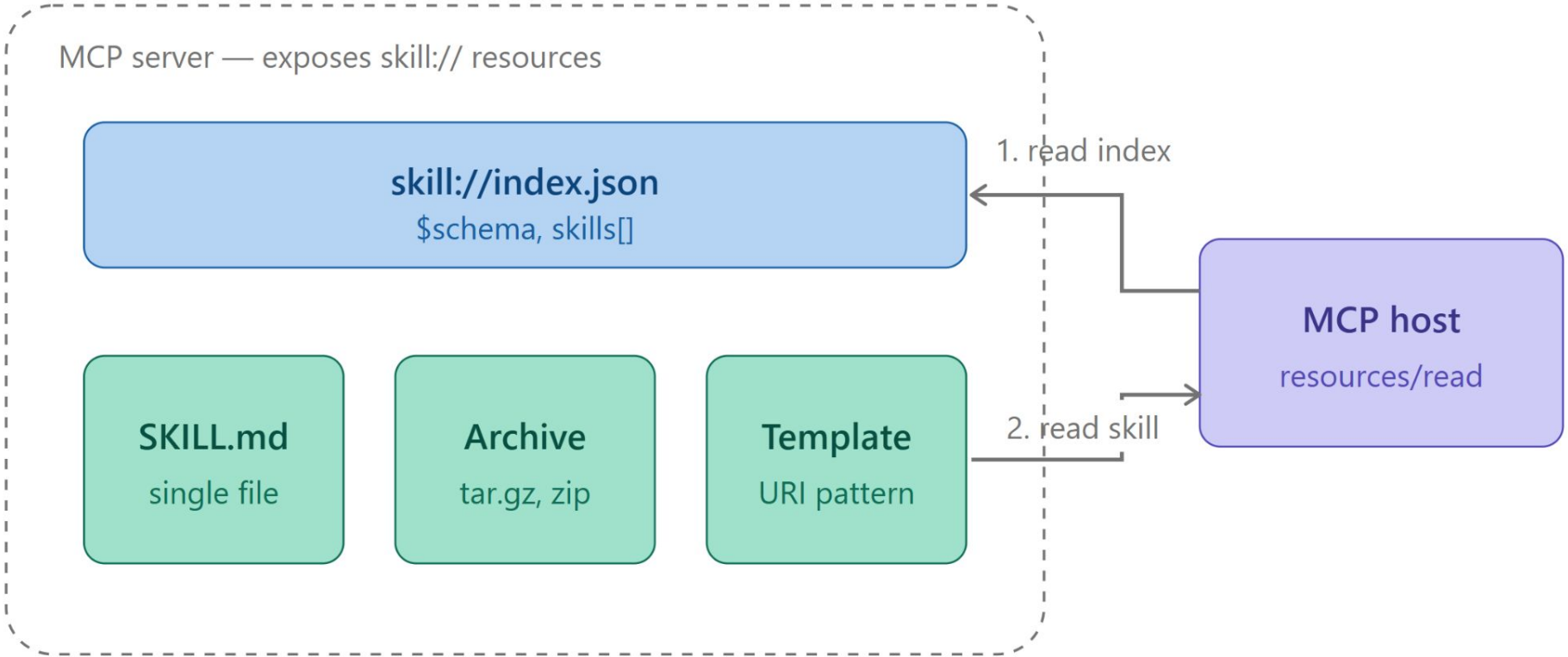


(Note: can be used with Vercel’s npx skills as an install source)

Skills + MCP: ‘Prompt Injection As A Service’ (the good kind)

- Skills can work well as a ‘user manual for tools’
- Serving skills through MCP: Same channel as tools, no separate install
- Discover skills through the same interface as the tools they describe
- Opens the door to integrating RBAC, audit logging, version-adaptive content
- Working Group charter

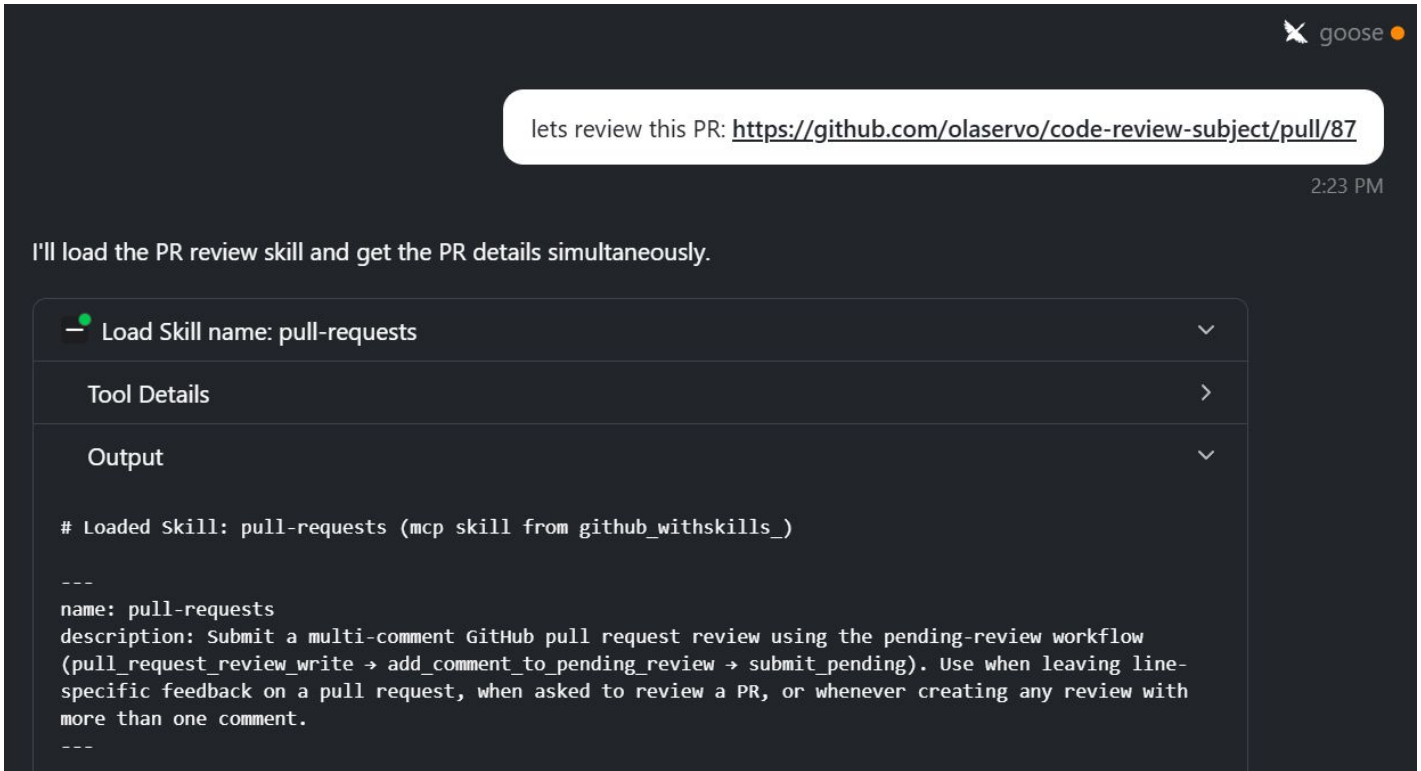
Skills Over MCP (SEP-2640)



Skills Over MCP Working Group: Ola Hungerford (Nordstrom / MCP),
Peter Alexander (Anthropic / MCP Core Maintainer) +50ish other contributors / participants

Examples: Skills Over MCP

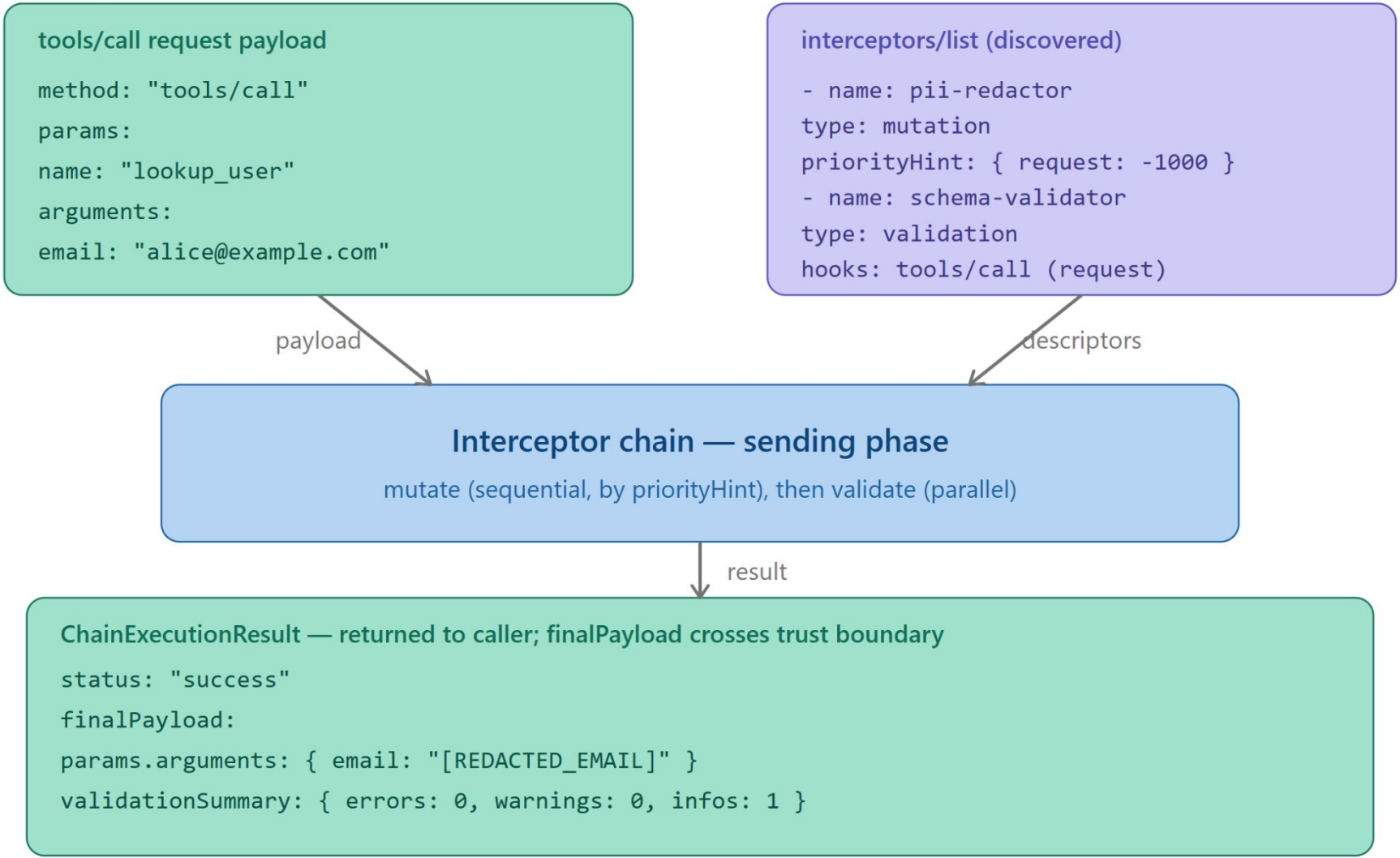
GitHub example: use bundled skills instead of static instructions



Interceptors in MCP (SEP-2624)

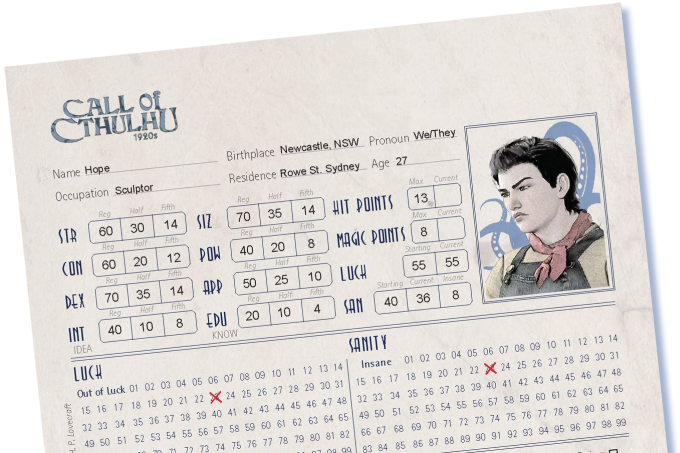
- Today MCP middleware (PII redaction, validation, audit, rate limiting) lives in a sprawl of non-interoperable sidecars, proxies, and gateways.
- Interceptors make middleware itself a discoverable MCP primitive (with interceptors/list + interceptor/invoke).
- Goal: Clients implement the call pattern once, servers expose interceptors once, guardrails can be deployed across everything compatible without per-service code changes.
- Lets you validate and/or mutate MCP requests and responses

Led by: Peder Holdgaard Pedersen (MCP Maintainer/Saxo Bank),
Sambhav Kothari (AAIF TSC/Bloomberg),
Kurt Degiorgio (Bloomberg) + other Interceptors WG members


tools/call request payload
method: "tools/call"
params:
name: "lookup_user"
arguments:
email: "alice@example.com"
interceptors/list (discovered)
- name: pii-redactor
type: mutation
priorityHint: { request: -1000 }
- name: schema-validator
type: validation
hooks: tools/call (request)
payload
descriptors
Interceptor chain — sending phase
mutate (sequential, by priorityHint), then validate (parallel)
result
ChainExecutionResult — returned to caller; finalPayload crosses trust boundary
status: "success"
finalPayload:
params.arguments: { email: "[REDACTED_EMAIL]" }
validationSummary: { errors: 0, warnings: 0, infos: 1 }

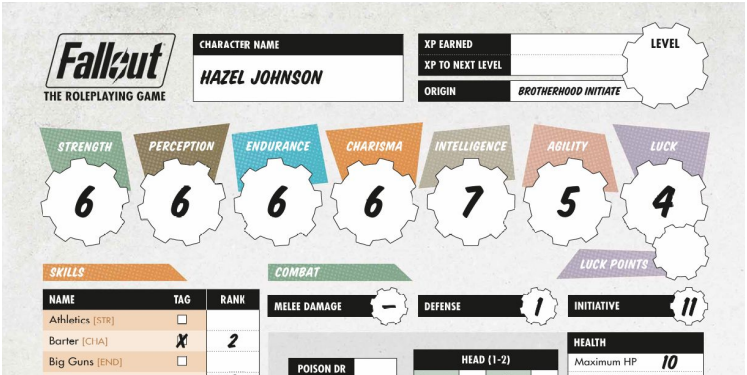
Case Study: Tabletop Role-playing Games

- Johnny: husband/composer/producer/musician
- Johnny's other job is to make sure that we're playing enough games
- Tabletop RPGs like Dungeons & Dragons have a high learning curve
- Sometimes we want an agent to help us learn (or help us run the game)
- Tabletop RPGs rely on specialized knowledge and human creative guidance



Why Games Need Skills + MCP

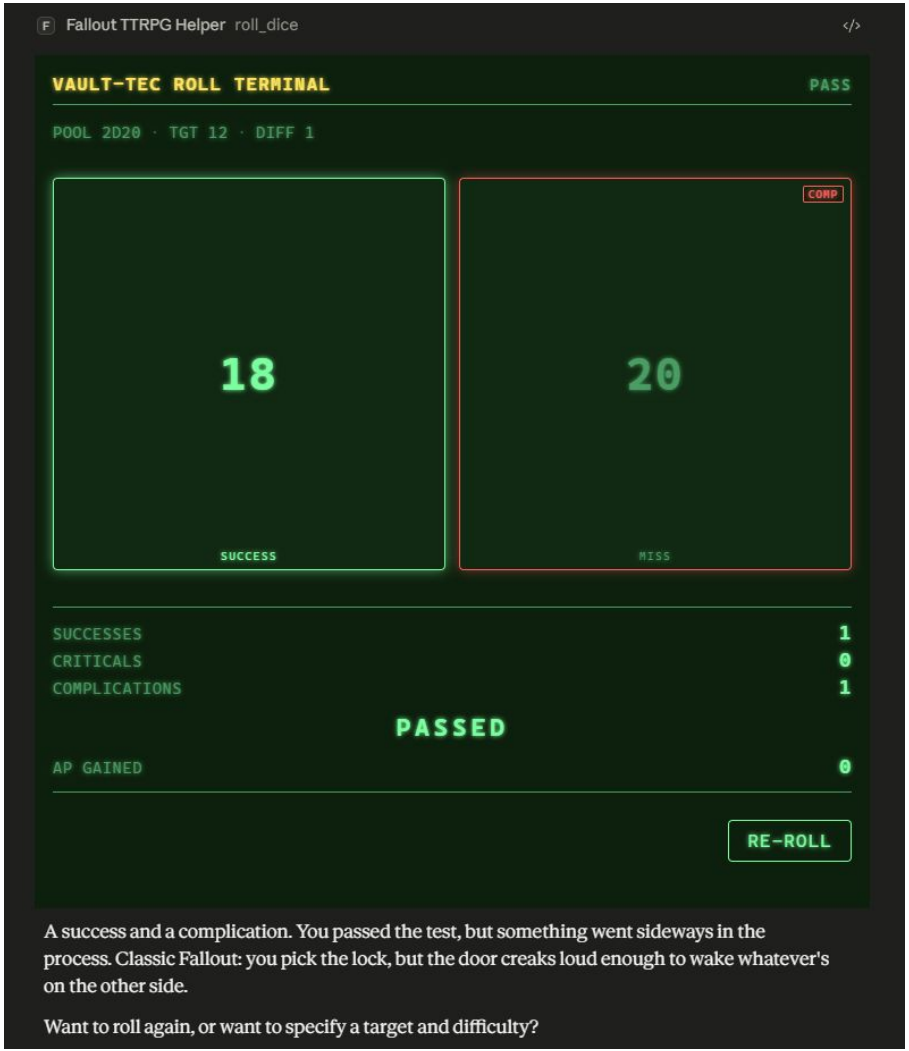
- For popular, older tabletop games like Dungeons & Dragons: models have been trained on enough information that they can get it mostly right
- Fallout is an example of one of the many newer, evolving games that models don't fully understand without external content
- Rely on combo of numerical calculations, story specifics, character stats



Bootstrapping What's Missing

- ‘Specialized books + calculations’ are a good fit to translate to ‘skills + tools’
- There aren’t any official skills or tools to help us learn and play the game
- …but Claude was happy to help create them from the source materials!
- *(see next slide for examples)*

- Rule book, character sheets, and story -> Skills
- Dice tools, Character Sheet rendering -> MCP Apps
- Model used skills to create the apps and tools to match the game



Two Use Cases, One Attribution Problem

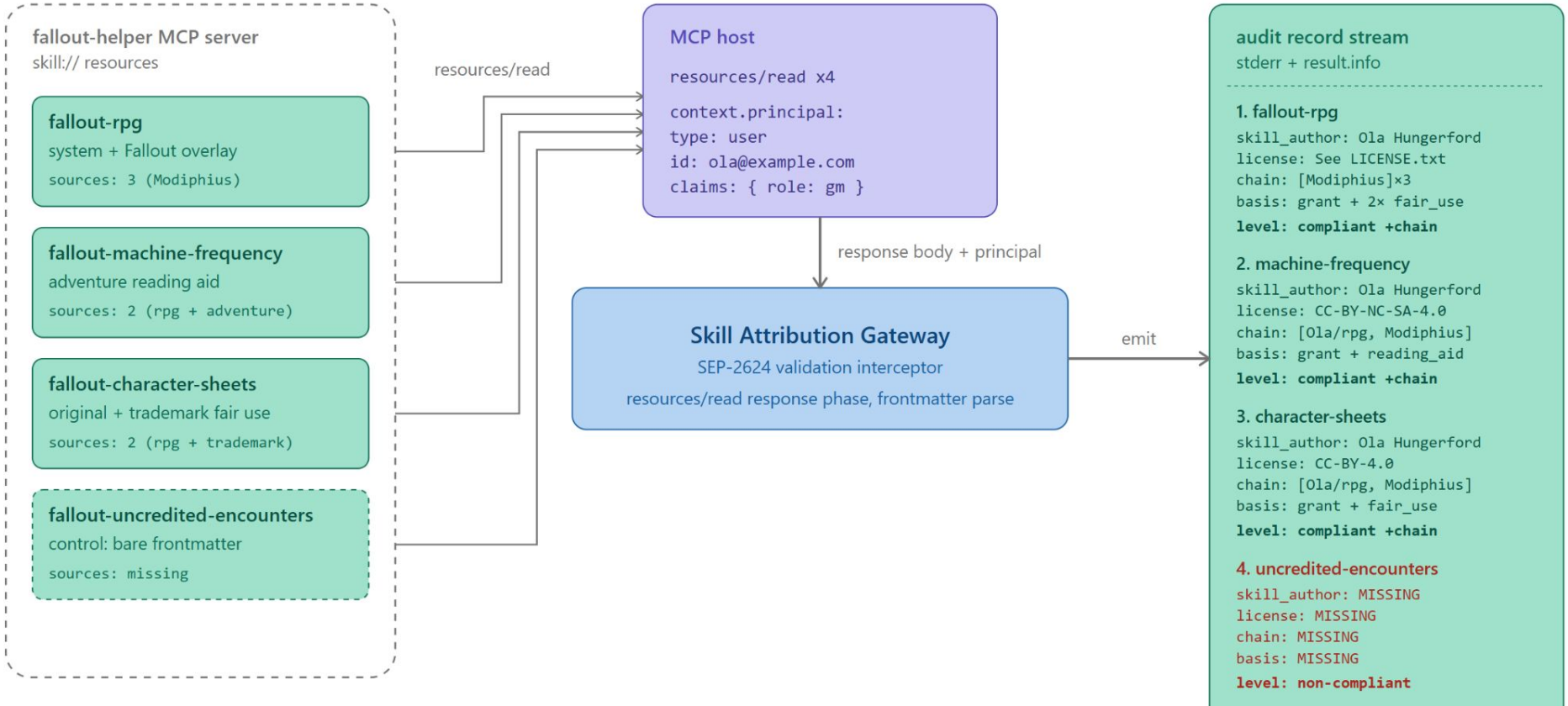
- Outside of the ‘game helper’ use case: these types of games also encourage customization, extension, etc. by the players (including creating new characters and stories)
- Games like Fallout are based on copyrighted game engines and IP
- For both use cases: attribution is complex but needed for both game creators and for users as creators

Beyond this example: this pattern at scale

- AI agents will keep needing specialized human knowledge at inference time. Games like this are one example.
- Today, no shared infrastructure attributes or compensates that contribution.
- Two groups can shape the alternative: specialized content publishers, and users-as-creators who extend what publishers provide. The standards under design right now (skills, MCP, interceptors, etc.) are where attribution either gets built in or doesn't.

(See next slide for one example of how skills over MCP and Interceptors could work together to support license validation, attribution, etc. in AI applications)

Example: Attribution/Validation/Injection w/ MCP Interceptors



How and where to get involved in these topics

- MCP Working Group charters in Community section of the MCP docs include more information on participating in related groups
- Please email me olahungerford@gmail.com if you'd like access to the example skills and code in this presentation!