Agent Discovery Landscape

draft-jimenez-dawn-discovery-landscape-00

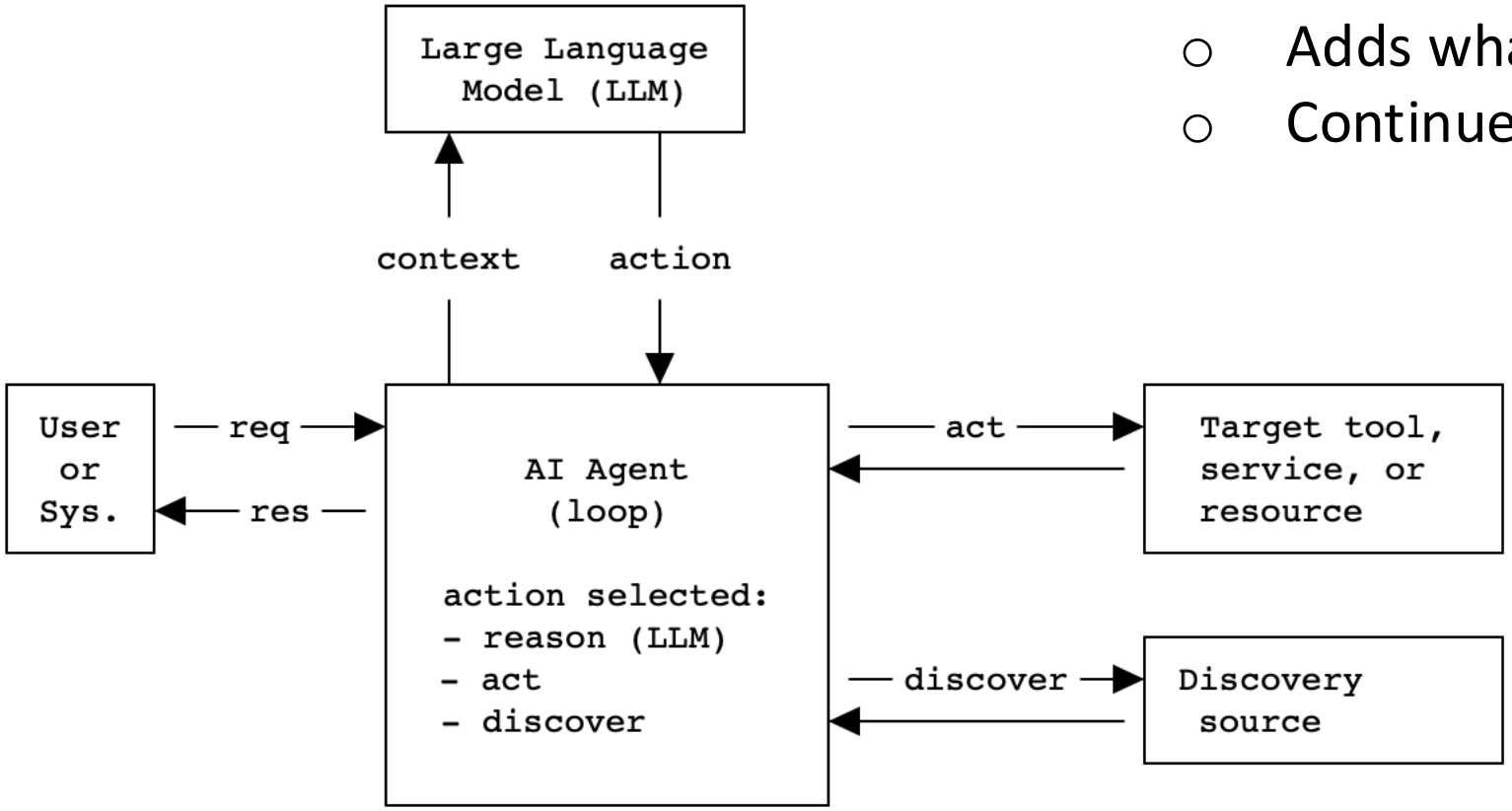
July 20, IETF 126 Vienna, Austria

Jaime Jiménez, Jim Feng, Jari Arkko, Mirja Kühlewind, Rajat Kandoi
Ericsson Research

Agent discovery in context

Discovery in an agent loop:

- Agent reasons about a task
- Realizes its tool set is insufficient
- Queries a discovery source
- Adds what it finds to its working set
- Continues the task, e.g., uses the new tool

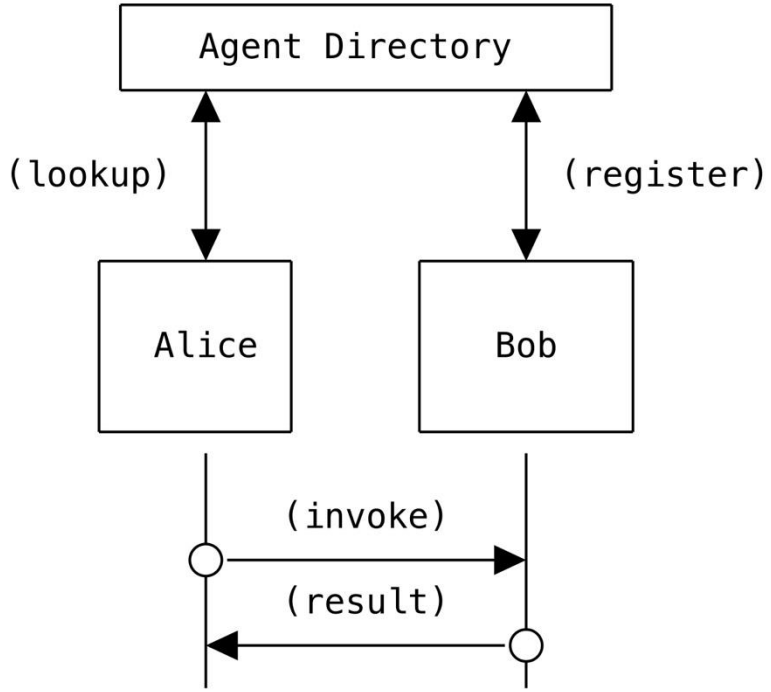


Example agent discovery (demo)

We are showing two agents, Alice and Bob, a set of tools, and an agent directory.

Alice discovers Bob via the directory in 2 hops: fetch /.well-known/ad, then lookup. Bob registers with a TTL and refreshes periodically. Alice caches results in bookmarks with freshness tracking.

Alice probes the AD, gets agent metadata, and invokes tools described in the metadata. Caching & bookmarks.



What this proves

Basic discovery works on existing standards. The parts that need new work are federation, semantic search, and cryptographic identity and interoperability

Discovery at runtime, not just at startup
Probes multiple formats with graceful fallback
Invocation driven by AD metadata

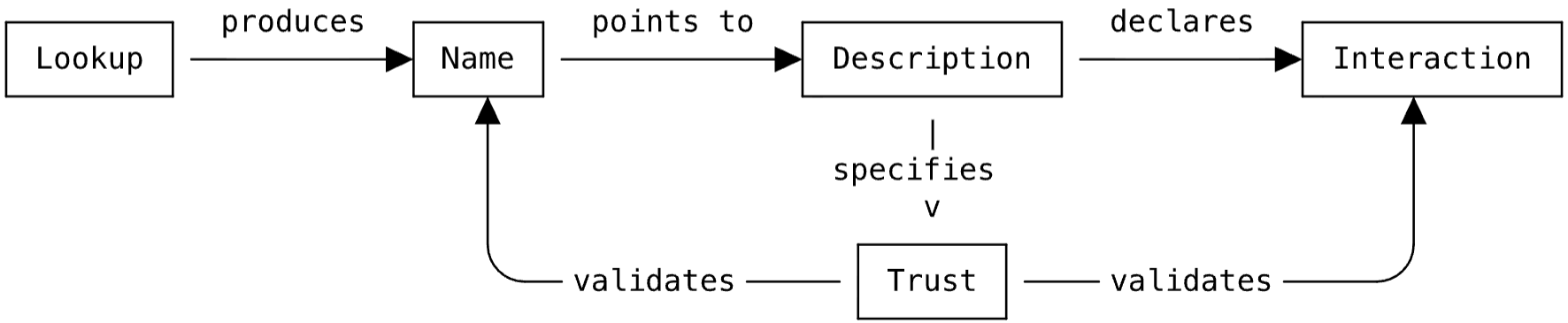
Built in ~300 lines of Python on RFC 9727 (API Catalog) + RFC 9176 (CoAP Resource Directory) patterns + RFC 9111 (HTTP caching)

Use Cases

Scenario	Trust basis	How discovery works
Curated set	Provider picks the tools	Static config
Local context	Local admin controls	DNS-SD, configured domain
Known partner	Domain name as anchor	Fetch well-known URI
Federated consortium	Mutual agreements	Federated registries
Free-form search	Open Internet	Directories, hypermedia traversal

Ordered by increasing trust complexity. Most deployed systems today only handle cases 1-3.

Discovery Components



5 functional components. Each mechanism bundles a different subset:

- DNS-AID covers Lookup + Name
- A2A Agent Cards cover Lookup + Description
- ANP spans all 5

Mechanisms Surveyed

DNS-Based

- DNS-AID — `_agents` + SVCB
- DN-ANR — 3-layer, FQDN + SVCB
- AID — TXT at `_agent.<domain>`
- AgentDNS — semantic DNS (expired)

Host-Level

- api-catalog (RFC 9727)
- A2A Agent Cards
- MCP
- ANP
- AIDIP

Registries and Directories

- 3GPP CAPIF — operator registry
- IoA — semantic (Tsinghua)
- ANP ADSP — crawl + DID
- AGNTCY ADS — DHT + OASF
- ARDP — federated, JWS
- Agent Directory — soft-state (Ericsson)

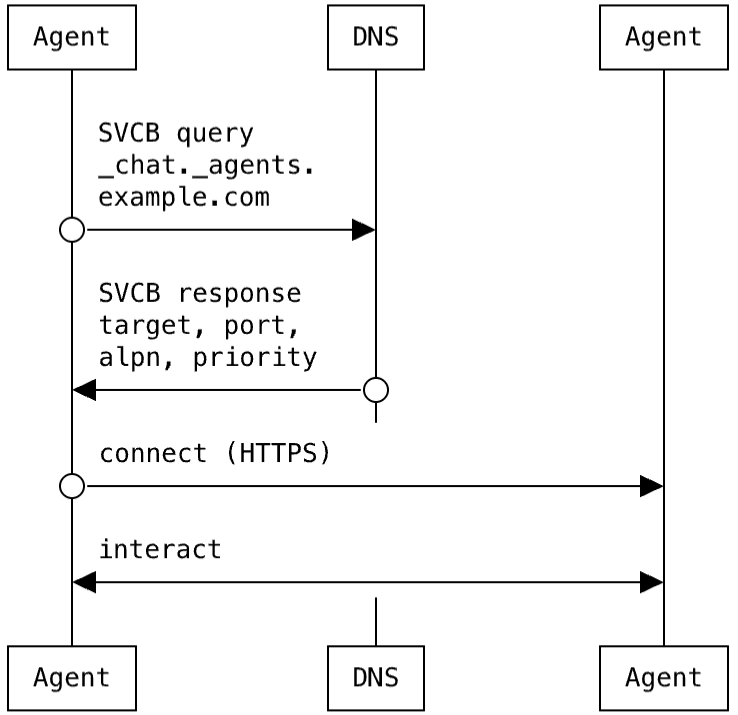
Other

- A2A Registry — curated, no API

DNS-Based Resolution

Proposal	Approach
DNS-AID	`_agents` subdomain + SVCB
DN-ANR	3-layer, FQDN + SVCB/HTTPS
AID	TXT at `_agent.<domain>`
AgentDNS	Root domain naming (expired)

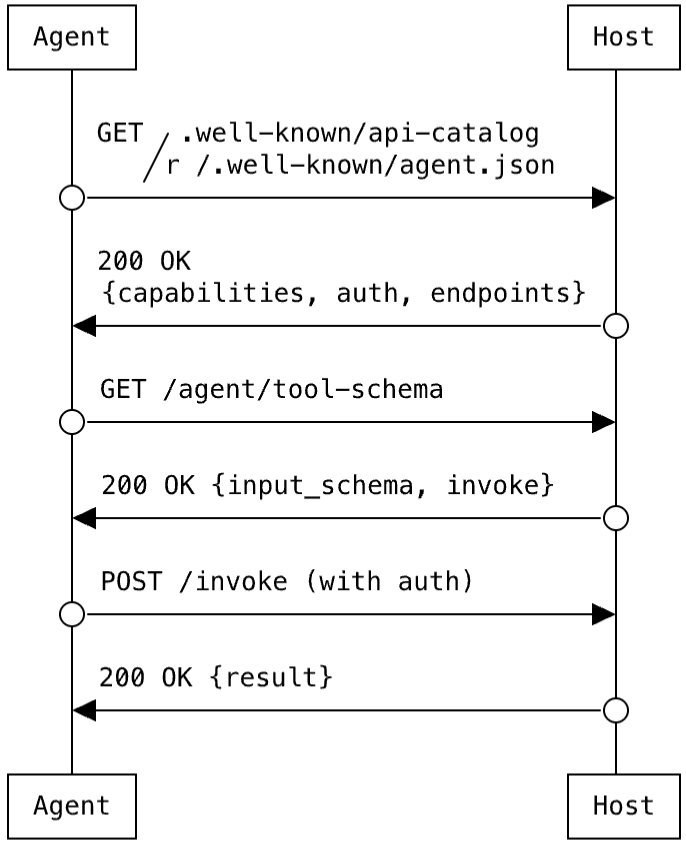
No new protocol elements needed. DNSSEC provides trust. Per-domain scope only.



Host-Level Self-Description

Approach	Endpoint
api-catalog	`/.well-known/api-catalog`
A2A	`/.well-known/agent.json`
MCP	Local config → JSON-RPC
ANP	`/.well-known/agent-descriptions`
AIDIP	Semantic resolution (ASR)

Pull-based, static. You must know the host. Most widely deployed pattern today.

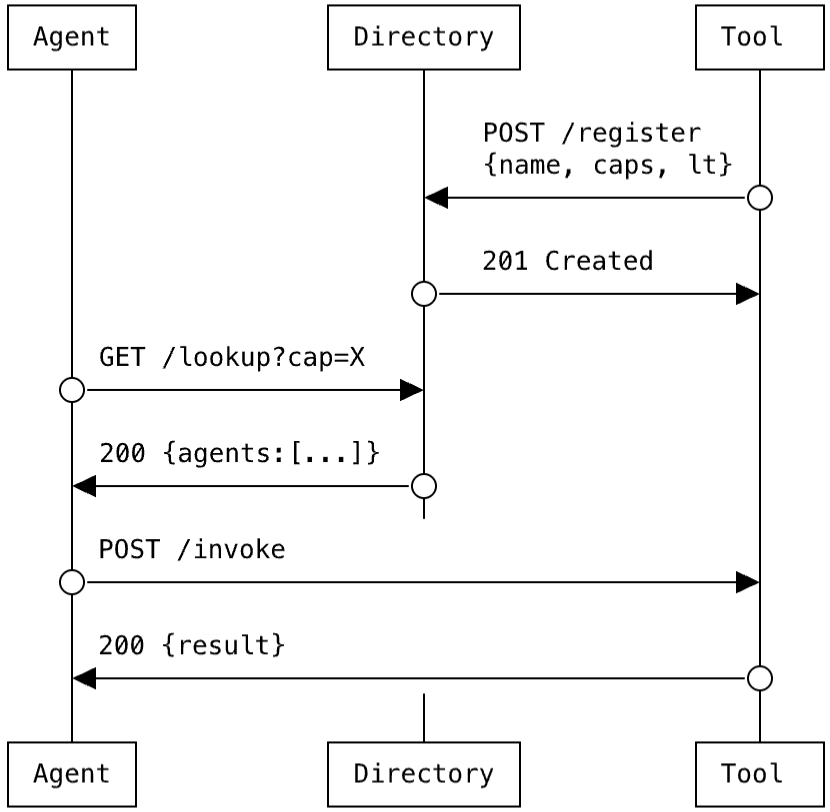


Registries and Directories

Approach	Model
CAPIF	Operator registry (3GPP)
IoA	Semantic (Tsinghua)
ADS	DHT + OASF (Cisco)
ARDP	Federated, JWS
AD	Soft-state (Ericsson)

Trade per-host autonomy for cross-domain search.

Open question: how do they federate?



No Mechanism Covers All Use Cases

Approach	Curated set	Local context	Known partner	Federated consortium	Free-form search
api-catalog	✓	-	✓	~	~
A2A Agent Cards	-	-	✓	—	—
MCP Capabilities	✓	~	—	—	—
Agent Directory	✓	✓	✓	~	~
DNS-AID	—	✓	✓	—	—

You need to compose mechanisms.

What's Missing

Semantic discovery
No single standard way to find agents by capability. ADS, IoA, AIDIP each sligthly different.

Interoperable federation
ARDP, ADS, AD each federate differently. No single common protocol.

3GPP CAPIF Exposure
3GPP CAPIF has API capabilities but would need to be updated to support A2A/MCP.

Metadata format fragmentation
A2A Cards, MCP schemas, ANP JSON-LD, OASF records. No explicit mappings exists.

Trust bootstrapping
DNS lookup is not enough to establish trust. TLS gives a domain certificate but you need more to establish trust in the agent (root of trust via registry, WIMSE, ...)

Lifecycle for pull-based systems
Well-known URIs serve static metadata. No change notification.