Network Working Group
Internet-Draft
Intended status: Informational
Expires: 25 September 2026

H. Moussa
A. Akhavain
Huawei Canada
R. Pioli
Independent

24 March 2026

Task discovery in agentic networks
draft-mapmw-task-discovery-01

Abstract

This document defines an architectural framework for an open, interoperable ecosystem in which task owners publish tasks寰排epresented as structured task cards寰杯o a task-posting platform, enabling autonomous agents to discover tasks, negotiate execution terms, and coordinate multi-agent collaboration. The architecture introduces a set of functional layers寰芭ncluding the Task Owner Layer, Task Owner Access Layer, Task-Posting Platform, Agentic Layer, Agent Access Layer, and an optional Communication Link寰杯hat collectively support secure task publication, agent discovery, capability evaluation, and bilateral negotiation. The framework is designed to accommodate heterogeneous agents with diverse skill sets, trust requirements, and operational models, while ensuring consistent interaction patterns across platforms and vendors.

The document also surveys existing agent-discovery approaches, such as A2A, agntcy/OASF, ARDP, and DNS-AID, and identifies gaps that motivate a unified, interoperable model for task-centric and agent-initiated discovery and interaction. It also explores possible ways by which the current approached can enhance the proposed framework. The goal of this architecture is not to replace existing mechanisms but to provide a complementary framework that enables agent寰鍍ask interactions in scenarios that are difficult to support using traditional agent-to-agent or platform-centric interaction models. The document is concluded with some potential standardization venues for the IETF.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Copyright Notice

Table of Contents

1. Introduction 3
2. Task Discovery: an Agent initiated engagement 4
3. Traditional agent-task discovery approaches 8
  3.1. Approach 1 Agent-to-Agent protocol (A2A) 8
    3.1.1. General description of the approach 8
    3.1.2. Components, interfaces, and protocols 8
    3.1.3. Known implementations / ecosystem 9
    3.1.4. Shortcomings and limitations (expected or reported) 9
  3.2. Approach 2 Agntcy by Cisco 9
    3.2.1. General description of the approach 9
    3.2.2. Components, interfaces, and protocols 9
    3.2.3. Known implementations / ecosystem 10
    3.2.4. Shortcomings and limitations (expected or reported) 10
  3.3. Approach 3 DNS for AI Discovery (DNS-AID) 10
  3.4. Approach 4: Agent Registration and Discovery Protocol (ARDP) 12
    3.4.1. General Description of the Approach 12
    3.4.2. Core Characteristics 13
4. Our approach: task cards and task discovery 13
  4.1. Task Discovery Ecosystem 14
  4.2. Task cards 16
  4.3. Task discovery 17
  4.4. Interaction between task owners and agents 17

5. Complementary operations format 18
5.1. A2A and Agntcy role in task discovery 18
5.2. DNS-AID role in task discovery 18
5.3. ARDP role in task discovery 18
6. Conclusion and Discussions 19
6.1. Summary of advantages of the proposed Task Discovery Ecosystem 19
6.2. IETF work required to realize the vision 20
7. Security Considerations 21
8. IANA Considerations 21
9. Informative References 22
Contributors 22
Authors' Addresses 22

1. Introduction

To date, both industry and academic communities have shown strong interest in the problem of agent discovery, in which a task owner seeks to identify the most suitable agent or group of agents to perform a given task. Several frameworks have been proposed such as A2A, Agntcy, DNS-AID, ARDP, and DNA which we will examine in detail in later sections.

It is important to note that, although these methods offer valuable insights and each has distinct strengths and limitations, they share a common design principle that creates several challenges. They all assume agents are represented by identification records stored in a registry and made discoverable to task owners, who must then sift through piles of agent IDs to find suitable providers and assign tasks. This reliance on ID-card窶濶ased discovery introduces several limitations within the broader framework, including:

* Agent capabilities must be made discoverable to task initiators, and that information often needs to remain visible for the agent窶冱 lifetime; as the number of agents grows, this requirement increases the network窶冱 storage footprint. Ensuring global visibility typically requires geographic replication or federation of registries, which multiplies storage and synchronization costs and also raises additional operational burdens: increased bandwidth for replication and queries, higher indexing and lookup overhead, greater CPU and memory use for maintaining searchable indexes, and stronger consistency or reconciliation mechanisms to keep records correct and up to date. These costs together drive latency, operational complexity, and monetary expense as the agent population and geographic scope scale.

* In traditional agent discovery, task owners must sift through many agent identification records to find a suitable provider. This process assumes owners know the precise skills or requirements needed for a task寰蚤 strong and often unrealistic assumption. As a result, non-expert owners face high search friction, increased mismatch risk, and longer resolution times; these effects worsen for complex, multi-domain tasks

* Traditional discovery often funnels work to a few visible providers, creating task magnets: top-ranked agents get most jobs, newcomers struggle to get any, and reputation feedback loops reinforce the imbalance. This reduces competition, slows innovation, and may lead to missed opportunity to match with the best agent for the given task.

* Task initiators need intelligent search controls寰敗toppage criteria and efficient search methods寰巴ecause agent populations grow continuously and exhaustive discovery is impractical; deciding when to stop expanding the search space and how to explore promising candidates efficiently is therefore a core design challenge.

2. Task Discovery: an Agent initiated engagement

Given the above foreseeable potential challenges to traditional approaches, we propose an alternative and complimentary framework where agents take active role in task to agent matching. In this framework, we consider what we refer to as task discovery, as follows:

* Consider a system in which task owners can post their tasks on some type of a platform (e.g., social media, internet, websites, ebillboards寰ヲetc).

* Also consider that agents are equipped with means and intelligence that enable them to access this platform.

* Agents can then discover tasks posted to this platform.

* Agents can select the tasks that suit their skillset and capabilities.

* Agents can then send necessary information (e.g. agent card, history, make, model, ratings...etc) to task owners and present themselves as potential candidates capable of fulfilling the posted task.

* Task owners can employ a contention resolution mechanism to assign the task to a candidate and provide permission to the selected agent to complete the task. Essentially, compared to traditional methods, in this propose approach, task discovery helps decreases the number of comparison processes needed to select the best matching agent.

* Agents can form coalitions (pre-organized or dynamically formed upon discovering posted tasks) to appear more capable than they would in their individual forms. These are teams of agents where each team has an overall team capability and skillset that is an aggregate of the capabilities and skillsets of the team members.

* In scenarios where multi-agent collaboration is needed, team formations enables the task owner to have the full knowledge of all different agents involved in fulfillment of the task from the onset. This is in contrast to the traditional task fulfillment where dynamically recruited downstream agents need to be continuously announced and approved by the task owner. Essentially, the propose approach can help ease up the need for chain of permission requests.

* Similarly, upon discovering posted tasks, agents can acquire additional agentic skills to augment their capabilities before submitting applications to task owners and becoming candidates.

The above proposed complimentary architecture can help facilitate various services, including:

* Increase agent visibility: Idle agents who hold identification cards but are not being selected by task owners can use the proposed architecture to increase their visibility. In this design, agents are allowed to actively approach task owners and offer their services, rather than remaining passive and waiting to be chosen.

* Reputation Building and fair selection: Reputation is a key factor that differentiates agents and influences their likelihood of being selected, typically based on task-owner ratings, accuracy, completion time, and overall performance. Because reputation reflects an agent寰洰 history of completed tasks, newly added agents face a cold-start problem in traditional discovery models where agents remain passive and must wait to be chosen, making it difficult to compete with established agents. The proposed platform mitigates this challenge by allowing agents to actively seek out tasks and offer their services, giving them opportunities to build or strengthen their reputation rather than relying solely on being selected.

* Guided Agent improvement: Agents are expected to be self-evolving entities that continuously learn new skills. In traditional passive discovery methods, organizations develop agents and are responsible for improving the quality of their published agents; to do this efficiently they conduct extensive market analyses to identify the skills their agents lack and evolve them accordingly. However, because discovery is passive, such analyses assume access to records showing which agent performed which task and why it was chosen; information that is often proprietary or confidential in competitive markets. In the proposed approach, agents can inspect task postings directly: they can infer required skills from task descriptions and metadata, compare those requirements to their own capabilities, and determine which skills to develop so they are better positioned to handle the tasks most commonly submitted by task owners.

* Improve agent-task matching: Selecting the right agent for a task is not straightforward. Traditional methods assume task owners know enough about a domain to specify the exact skills required, but that is often false: a car owner who reports 窶忤eird noise when braking窶may not know whether the problem is tires, brakes, or suspension, so they will search for a generic “mechanic,” yielding either too many matches or none if they must be more specific. Under the proposed approach, task owners can post a general request (e.g., “fix car making noise when braking”) and agents can proactively query for clarifications—asking about recent brake work, symptom duration, or offering to run a diagnostic—thereby shifting the diagnostic intelligence from the owner to service providers and improving match quality.

* Improved complex task handling: Tasks vary in complexity: some require a single agent (e.g., “translate this text from French to English”), while others need multiple agents (e.g., “help build a bicycle” — one agent to purchase parts, another to deliver them, a third to guide assembly). In traditional discovery models, complex tasks must be split into subtasks, but that process assumes the task owner can (1) decompose the task to the right level of granularity and (2) know what agents exist to handle each subtask; if the available agent landscape is unknown, splitting can be counterproductive (for example, designating a “purchase parts” subtask is useless if no purchasing agent exists). The proposed approach lets task owners post raw, unsplit tasks to the billboard, relieving them of the burden of decomposition; agents discover those postings, use their internal knowledge and awareness of other agents to form coalitions, and split and assign subtasks among themselves, thereby shifting the complexity to the service-providing side and enabling tasks to be partitioned in ways that match actual agent capabilities. Also, this provides a means by which agents can form coalitions with trusted parties to handle tasks together.

* Mechanisms for authentication and trust establishment: task owners generally need to interact with authenticated, registered agents, and traditional discovery models place the burden of vetting on owners—reviewing credentials, certifications, provenance, and other trust signals—which assumes owners have the expertise and resources to perform reliable vetting. Prior proposals (for example, DNS-AID and ARDP) shift parts of that responsibility to third parties such as DNS-based certification systems, but they still depend on standardized, often rigid trust criteria and centralized records that may be proprietary or vulnerable to compromise. To increase flexibility and usability, the proposed approach lets task owners define their own authentication and trust requirements (for example, accepting agents by origin and publisher, or requiring background checks, tests, or cryptographic attestations); agents apply for tasks by connecting to the owner, and the owner admits only those agents that meet its chosen criteria. The billboard used for posting tasks can also enforce admission policies, allowing only authenticated and trusted agents to submit proposals. It would be appreciated to mention that under this approach, various grounding mechanisms would be needed so that proposed flexibility non-standardized design does not lead to fragmentation, spoofing, or unscalable manual vetting.

It should be clear from the above that the primary advantage of the proposed architecture is to shift part of the discovery burden from task owners to service-providing agents rather than to replace existing registry-based discovery. Task owners retain the option to

search agent registries, but the complementary billboard model lets owners post raw, unsplit tasks and lets agents perform decomposition, diagnostics, and coalition formation. Agents still rely on discovery to assemble teams, but their team formation is driven by objectives inferred from the task posting and by agents' own knowledge of capabilities and trust relationships. To make this shift practical and secure, the platform should combine owner-configurable trust policies with policy profiles, machine-readable credentials, standardized task posting mechanisms, standardized task metadata formatting, billboard admission controls, and budgeted search/stop rules so owner flexibility does not produce fragmentation, spoofing, or unscalable manual vetting. All of these aspects can benefit from discussions and developments within the IETF as will be discussed later.

3. Traditional agent-task discovery approaches

This section aims at providing a brief high level collective description of traditional known agent discovery mechanisms (i.e., some form of a summary for the different approaches that outlines the general theme)

3.1. Approach 1 Agent-to-Agent protocol (A2A)

3.1.1. General description of the approach

A2A uses Agent Cards (structured JSON metadata) plus registry servers and discovery clients so that agents and clients can retrieve machine-readable descriptions of remote agents' capabilities and endpoints. Discovery is tightly coupled with the A2A messaging and lifecycle model.

3.1.2. Components, interfaces, and protocols

* Agent Card: the primary discovery artifact (JSON schema) describing name, capabilities, endpoints, and interaction hints.

* AgentRegistry / DiscoveryClient: servers that host Agent Cards and clients that query registries or use local discovery strategies.

* A2A transport and security stack: discovery is integrated with the protocol' s authentication and session establishment flows so that discovery leads directly into authenticated communication.

3.1.3. Known implementations / ecosystem

Google published A2A and maintains a public GitHub repository and documentation; multiple SDKs and language bindings (including a Python discovery module) implement the Agent Card and registry patterns.

3.1.4. Shortcomings and limitations (expected or reported)

* Registry and card freshness: Agent Cards must be kept up to date; stale cards can misrepresent runtime capabilities.

* Owner specification burden: discovery via structured cards assumes owners or publishers can accurately enumerate capabilities; vague tasks still require additional negotiation or diagnostics.

* Interoperability gaps without profiles: different deployments may extend Agent Cards or discovery strategies; without standardized profiles, cross-domain behavior can vary.

3.2. Approach 2 Agntcy by Cisco

3.2.1. General description of the approach

AGNTCY provides a registry-and-messaging stack for agent interoperability: agents publish metadata and capability records to discoverable services, use secure messaging for interaction, and rely on observability/attestation components to verify behavior. Discovery within Agntcy centers on a registry-style directory that indexes agent metadata and capability schemas; agents publish machine-readable schema-backed records (built on the Open Agentic Schema Framework (OASF)) and discovery queries retrieve matching entries.

3.2.2. Components, interfaces, and protocols

* OASF schema server and schema repository for capability and metadata definitions.

* Agent Directory / Agent Directory Service (ADS) that indexes signed agent records and maps capability descriptors to content locations.

* Secure low-latency messaging (SLIM) and transport bindings for agent-to-agent communication that integrate discovery with secure agent-to-agent messaging and observability pipelines.

* Identity, observability, and evaluation subsystems for provenance, runtime telemetry, and capability assessment.

3.2.3. Known implementations / ecosystem

AGNTCY code and reference components are available on GitHub from Outshift/Cisco and collaborators; the project has been adopted into a Linux Foundation initiative with multiple industry participants.

3.2.4. Shortcomings and limitations (expected or reported)

* Registry-centric bias: relies on owners or agents to keep registry entries current; static entries can become stale relative to runtime capability.

* Operational complexity: full stack (registry, messaging, observability) increases deployment and integration effort for platforms.

* Cold-start for newcomers: while registries provide authoritative identity, newcomers still need paths to earn visibility and reputation unless additional onboarding flows are defined.

3.3. Approach 3 DNS for AI Discovery (DNS-AID)

DNS-AID, as described in [I-D.draft-mozleywilliams-dnsop-dnsaid] is a means of utilizing the Domain Name System (DNS) to facilitate scalable and interoperable discovery between AI agents that can also be applied to tasks. It provides a well-known entry point to facilitate discovery and automation, in much the same way as the defacto label www does for web sites. This could be used on internal or external networks.

An agent might discover tasks it could process directly, or do this via another agent that brokers communication, for instance ensuring authentication and authorization to perform a task and a token to allow direct interaction with the task itself.

* DNS AID creates a well-known entry point to an organization’s agents. Currently, many agents and capabilities are listed in host files accessed through gateways (MCP, for example). By moving the capability descriptors closer to the clients (in this case we assume there are many types of clients dispersed throughout an organization looking for agents and vice versa), you improve scalability, performance, and federate discovery to control plane that is natively integrated to the current existing network stack.

* Facilitated through a leaf-attribute zone (e.g. _agents.example.com) containing SVCB records which embed metadata within the record: IPv4/6 hint, alpn (h2, h3, a2a, mcp1, etc), port, and any other key value pair (e.g. model card URL, token bundle input, modalities, etc) without recommending any changes to existing DNS protocols.

* It’s likely that for particularly volatile workloads, ephemeral and others that the propagation time for AXFR/IXFR may exceed the lifetime of the task. It’s better to consider two approaches: use SVCB-param-key to signify tasks (e.g. tasks = true or false) which then tasks are exchanged through the application protocol itself between workers, or embed task metadata within the specific agents as a subzone (e.g. tasks.chat._agents.example.com). Either provide a mechanism to be listed in a centralized location for federated task discovery and completion.

The key components and principles of DNS-AID that could be applied to task discovery include:

* use of the “feature rich” DNS SVCB record type to indicate task protocols and IP connectivity for efficiency with a defined structure (as opposed to an unstructured TXT record)

* DNS-Based Service Discovery (DNS-SD) to standardize the well-known entry point and to group tasks by protocol type or other categories

* providing additional properties to facilitate efficient communication

* organizations could optionally provide specific SVCB records for long-running tasks to improve efficiency

An example of discovering tasks could be an agent connecting to an end-point that has details of image processing tasks for an organization.

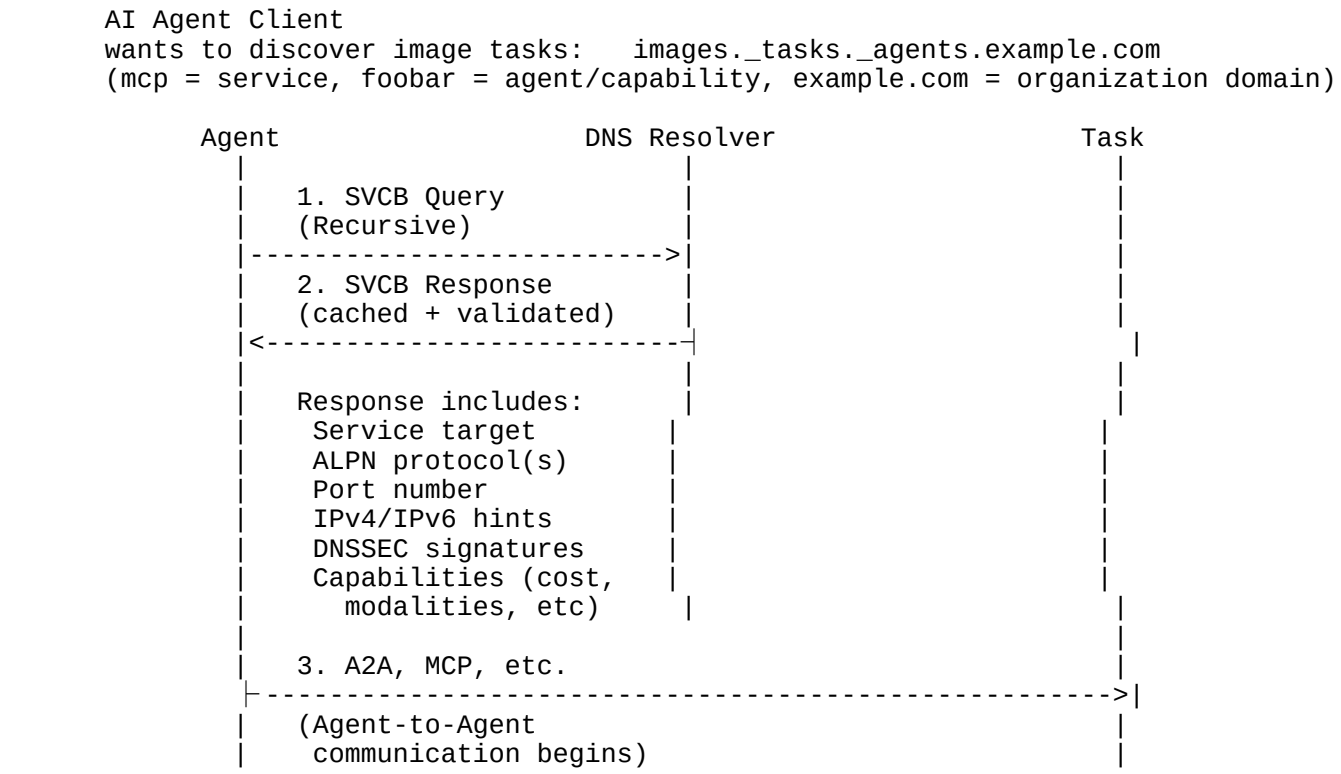

Figure 1: DNS-AID task discovery example 1

Note that DNS is not used for agents to communicate with tasks, rather it indicates the protocol that should be used, the IP address and potentially other defined properties. DNS would not index all tasks and is not suited to the discovery of lists of tasks, rather the use case is facilitating efficient communication to a known end-point.

Implementation examples of DNS-AID can be found at [DNSAIDsite].

3.4. Approach 4: Agent Registration and Discovery Protocol (ARDP)

3.4.1. General Description of the Approach

The Agent Registration and Discovery Protocol (ARDP) is a control-plane protocol designed to provide stable agent identifiers, authenticated registration, and authorized endpoint resolution in distributed and federated environments.

ARDP is not limited to static agent “card” storage. It defines a dynamic, soft-state registration model in which bindings expire and must be refreshed, and in which resolution is subject to authorization policy and privacy controls. This distinguishes ARDP from passive directory models and enables its use in mobile, federated, and multi-tenant environments.

Unlike static registries that merely store agent identification records, ARDP defines:

* Identity-bound registration with cryptographic proof-of-control (JWS-based PoC for REGISTER/refresh operations).

* Soft-state bindings with TTL and refresh semantics, allowing dynamic endpoint mobility.

* Fine-grained authorization for RESOLVE and QUERY operations.

* Capability advertisement independent of interaction protocol (e.g., MCP, A2A, HTTP, gRPC).

* Explicit federation model with provenance metadata and TTL honoring.

* Privacy-aware redaction defaults for discovery responses.

3.4.2. Core Characteristics

ARDP operates as a minimal control-plane discovery layer. It does not define session management, task execution semantics, runtime authorization tokens, billing mechanisms, or governance frameworks. Instead, it focuses on:

* Binding a stable Agent Identifier (AID) to active endpoints and capability metadata.

* Ensuring that only entities that can prove control of an AID may register or refresh it.

* Providing authorized resolution of endpoints and capabilities.

4. Our approach: task cards and task discovery

This section aims to provided general high-level description and architecture of the proposed Task Discovery Ecosystem. It also poses some questions that need answers and opens up venues for discussions.

4.1. Task Discovery Ecosystem

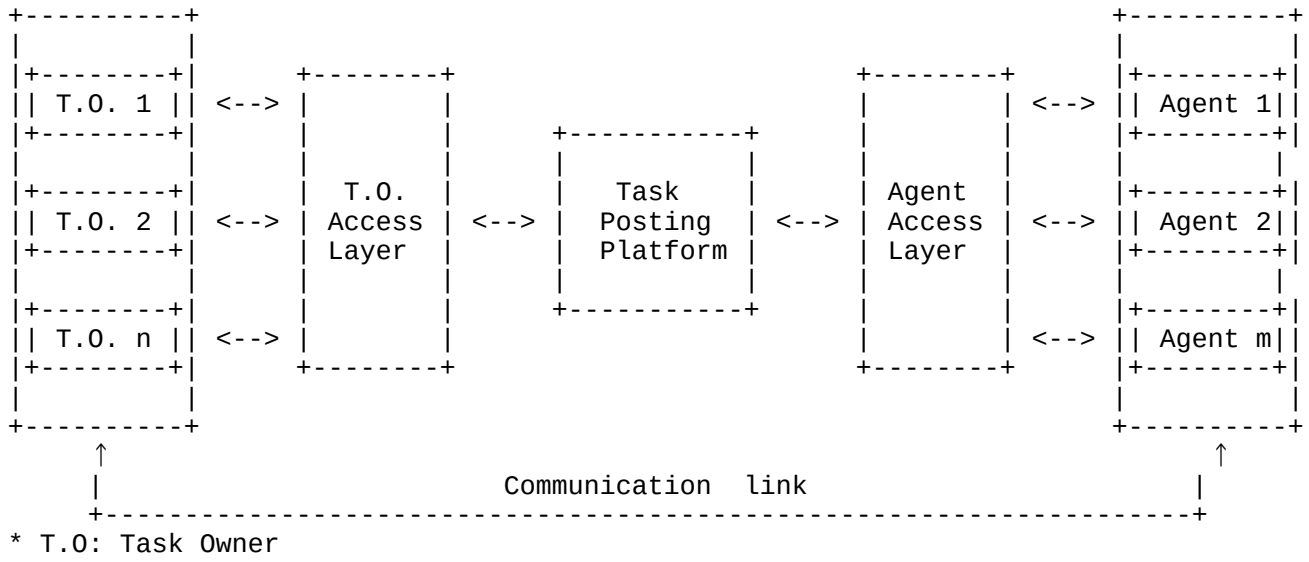

Figure 2: Task Discovery Ecosystem

As shown in the figure, the ecosystem consists of different layers.

* Task Owner Layer: This layer hosts the entities that submit tasks requiring assistance from AI agents. It provides mechanisms for verifying, authenticating, and registering trusted task owners, ensuring that only authorized parties can participate in the system. This layer also manages the lifecycle of owner activity—including task submission, accounting and charging, and policy enforcement—and offers the privacy and security controls needed to protect owner data and identity.

* Agentic Layer: The Agentic Layer hosts the autonomous agents that evaluate, select, and execute tasks. Agents in this layer expose diverse skill sets and operate as independent decision-making entities. They are expected to discover posted tasks through standardized search interfaces, analyze task requirements, compare them against their own capabilities, and determine whether they are suitable candidates. Once engaged, agents can communicate with task owners to request assignment, seek clarification, provide progress updates, or deliver results. This layer may also support additional functions such as capability attestation, agent registration and authentication, reputation tracking, coalition formation for multi-agent tasks, and mechanisms for agents to manage their own operational state (availability, load, or cost).

Within this layer, agent-to-agent discovery, communication, and session-management protocols—such as A2A, agntcy, ARDP, and DNS-AID—may be used to support coalition formation, peer coordination, and agent-level discovery needed to assemble appropriate teams for tasks that require multi-agent collaboration. In some implementations, specialized "scouting agents" may continuously search for suitable tasks and, upon discovery, rely on agent-to-agent protocols to subcontract or delegate the task to the most appropriate agents within their network.

* Task Owner (T.O.) Access Layer: This Layer provides the interfaces through which task owners interact with the task-posting platform. It exposes the protocols required for safe and reliable operation, including mechanisms for submitting new tasks, tracking task status, and updating, retracting, or modifying previously posted tasks. This layer also supports the communication requirements needed for rich interaction—such as multi-modal exchanges, session management, and secure transport between task owners and the platform. Standardizing this layer enables task owners of different types and from different vendors' ecosystems to interact with task-posting platforms in a consistent, interoperable manner. It should be noted here that also the task-posting platforms can come from different vendors as well, so standardized access layer is essential.

* Agent Access Layer: The Agent Access Layer provides the standardized interfaces through which agents interact with the task-posting platform and with other system components. It exposes the protocols required for safe task discovery, capability advertisement, proposal submission, and session establishment. This layer also supports secure communication, multi-modal exchanges, status reporting, and lifecycle management for agent-initiated interactions. By defining common access and communication primitives, the Agent Access Layer enables heterogeneous agents—potentially from different vendors or ecosystems—to interoperate reliably with task-posting platforms and with one another.

* Task-Posting Platform: The Task-Posting Platform is the environment in which task owners publish tasks (task cards for instance) and make them visible to eligible agents. It maintains the authoritative catalog of active tasks and enforces the policies under which tasks may be posted, viewed, or acted upon. Platforms may apply subject-matter restrictions, require specific agent capabilities, or limit participation to task owners who meet defined trust or compliance requirements. The platform is responsible for ensuring fair and consistent visibility across all

posted tasks and for exposing standardized APIs that allow agents to search, filter, and retrieve tasks in a predictable manner. In addition to basic posting and retrieval, the platform may support task lifecycle management, admission control, rate limiting, prioritization rules, and mechanisms for handling updates, cancellations, and task-owner clarifications. This allows it to provide necessary OAM functionality for task owners. Although platforms may differ in internal design, they are expected to expose interoperable interfaces that conform to the specifications defined by the Task Owner Access Layer and the Agent Access Layer, enabling cross-vendor compatibility and consistent behavior across deployments.

* The Communication Link: The Communication Link is an optional component that enables direct, bilateral interaction between task owners and agents after initial discovery. Its purpose is to off-load certain interaction responsibilities from the access layers by providing a secure channel through which both parties can negotiate task-specific details. Once an agent identifies a suitable task, it may use this link to contact the task owner directly and establish the terms under which the task will be fulfilled. These terms may include trust requirements, handling expectations, privacy constraints, compensation models, escalation paths, or any other operational parameters relevant to the task. This link supports private sessions that operate outside the default policies of the task-posting platform while still relying on standardized, regulated, and secured communication protocols defined by the ecosystem. It allows richer negotiation patterns—such as multi-modal exchanges, iterative clarification, or structured contract formation—without requiring the platform to mediate every interaction.

4.2. Task cards

* what are task cards?

* how are they generated? (there should be a mechanism to generate these) -- potential for standardization.

* What are some proposed task card structures? -- potential for many IETF drafts and solution proposal.

* How are task cards used by agents to satisfy the different scenarios considered? -- (these task cards need to be designed such that they can be utilized to fulfill the above scenarios. Mechanism by which this is done might not be in IETF scope, but perhaps would invite creative designing proposals)

4.3. Task discovery

* How are task cards stored? (authorization, authentication, standardization) --- potential for standardization

* How are task cards published and handled after being published? (expiration, prioritization, fairness) -- potential for standardization

* What are the possible different approaches to discovering them in light of the different scenarios being considered? -- potential for standardization

* How to ensure secure and private access to task cards? -- potential for standardization

* Centralized and decentralized discovery?

* Efficient discovery? -- This is likely out of scope for the IETF, making it more suitable as a research topic for the IRTF. Alternatively, this could be handled through engineered, standardized methods, such as hierarchical or DNS-based discovery.

4.4. Interaction between task owners and agents

* How do agents connect with task owners? (authentication, validation, and authorization) -- potential for standardization

* How can task owners monitor progress on their tasks? -- potential for standardization

* How do agents charge and bill for their service? -- This is likely out of scope for the IETF

* What happens if agents did not fulfill the task up to standards required? -- IETF can provide a way to express satisfaction, similar to ai-pref

* how to insure privacy and security? -- potential for standardization. Perhaps proposal such as ARDP might find utility here.

* authorization and authentication of agent? -- potential for standardization

* authorization and authentication of task owner? -- potential for standardization

* authorization and authentication of billboards? -- potential for standardization

5. Complementary operations format

This section aims to describe how the two models can operate side-by-side: registry-based discovery continues to offer a stable, authoritative directory of agents, while the billboard-driven approach adds dynamic task posting, proactive agent proposals, richer reputation building, and improved handling of vague or complex tasks. Together, they form a hybrid discovery ecosystem in which registries provide long-term identity and credentials, and the billboard layer provides real-time matching, diagnostics, coalition formation, and opportunities for newcomers—each compensating for the other's weaknesses.

5.1. A2A and Agntcy role in task discovery

To be completed..

5.2. DNS-AID role in task discovery

To be completed..

5.3. ARDP role in task discovery

Being a control-plane protocol that is designed to provide stable agent identifiers, authenticated registration, and authorized endpoint resolution in distributed and federated environment, the ARDP can facilitate many features necessary for the operation of the Task Discovery Ecosystem proposed here. To be exact, ARDP can:

* Serve as the authoritative identity and endpoint binding layer for agents that respond to task postings.

* Allow Agents discovering tasks via a billboard mechanism to:

  - Validate peer agent identities when forming coalitions.

  - Resolve authoritative endpoints for inter-agent coordination.

  - Verify provenance and federation trust relationships.

* Task billboards may reference ARDP-resolvable AIDs instead of duplicating identity records.

However, what the ARDP is not meant to do is:

* Task decomposition logic.

* Coalition formation strategies.

* Runtime authorization enforcement within an agent.

* Reputation scoring or bidding mechanisms.

These functions can operate above ARDP or alongside it in complementary architectures such as the proposed task discovery model.

Thus, in light of the above, it is clear that ARDP complements both registry-based discovery and billboard-based task discovery by providing a secure and interoperable control-plane foundation.

6. Conclusion and Discussions

6.1. Summary of advantages of the proposed Task Discovery Ecosystem

* A task-card storage entity is dynamic and demand-driven: posted tasks expire or are archived once completed, so the active dataset shrinks and grows with workload. This contrasts with registry-based approaches that must persist and replicate every agent profile indefinitely. By keeping only active task records readily searchable, the proposed model reduces persistent storage, indexing, replication bandwidth, and update churn, aligning infrastructure cost with actual demand while still supporting optional archival and a small authoritative registry for long-lived credentials, making it a better scalable solution.

* The proposed model reduces the task-owner’ s burden of finding the right agent by shifting discovery work to agents: agents that believe they are a good fit proactively submit proposals, which narrows the owner’ s search space and removes the need for owners to precisely specify required skills.

* For non urgent tasks that owners do not want to spend a lot for them to be done, they are posted for agents to bid on and owners can choose the best offer. This also allows idle agents to have a chance to make money and perhaps help with load balancing

* By allowing agents to proactively propose for posted tasks, the platform enables newcomers and under-exposed agents to build verifiable reputation through completed work, reducing cold-start barriers and improving match quality—provided the system enforces admission controls, verification steps, and incentive mechanisms to limit noise and gaming.

* The billboard model augments registry-based discovery rather than replacing it: owners retain registry search capabilities while also posting raw tasks to receive proactive proposals. This hybrid approach preserves existing workflows, lowers the barrier for non-expert owners, and enables richer matching strategies; to work well it requires interoperable metadata, admission controls, and clear ranking/deduplication rules so the two channels remain complementary and secure.

6.2. IETF work required to realize the vision

The proposed billboard-like agent initiated task discovery model offers clear benefits, but substantial IETF work is required to make it interoperable, secure, and scalable. Standards are needed for task-posting formats, machine-readable credentials, admission and discovery APIs, policy profiles, privacy-preserving attestations, and budgeted search semantics; without these, owner-defined trust policies will fragment the ecosystem, invite spoofing, and force unscalable manual vetting. The IETF should therefore define minimal, composable building blocks that preserve owner flexibility while enabling automated verification, cross-domain discovery, and fair matching across implementations.

* Task posting and metadata formats: a compact, extensible schema for raw task descriptions, required attributes, and diagnostic prompts.

* Discovery and billboard APIs: standardized endpoints and query semantics for posting tasks, submitting proposals, and retrieving shortlists.

* Standardized machine-readable credentials and attestations: interoperable formats and protocols for signed capability claims, provenance, selective-disclosure identity attributes, and revocation signals, enabling automated trust decisions between task owners, billboards, and agents without relying on proprietary or ad-hoc vetting.

* Policy profiles and admission rules: interoperable assurance levels (e.g., basic/verified/high-assurance) and a way to express owner preferences as machine-evaluable policies.

* Search semantics and stop rules: standardized notions of budgeted search, progressive widening, and marginal-gain stoppage so clients and servers can interoperate on when to halt exploration. This is also needed to limit the possibility of overload the read function of the agent/task registries.

* Coalition formation primitives: lightweight protocols for capability advertisement, role negotiation, payment split templates, and failure recovery.

* Fairness and anti-gaming controls: mechanisms that give newcomers a fair chance and prevent manipulation of the matching and reputation systems. This includes onboarding micro-tasks for new agents, freshness boosts so proposals from less-visible agents are not buried, reputation-normalization rules to prevent score inflation, and signals for detecting sybil or duplicate agents.

* Privacy and selective disclosure: mechanisms (attribute attestations, minimal disclosure) that let agents prove claims without exposing sensitive data.

* Audit, logging, and revocation: tamper-evident logging mechanisms, standardized revocation lists, and dispute-resolution hooks that allow task owners, agents, and billboards to verify what happened during a task lifecycle across different administrative domains. This includes consistent formats for recording proposals, decisions, completions, and failures; mechanisms for revoking compromised or misbehaving agents; and interoperable audit trails that support cross-domain verification and accountability. Without these shared mechanisms, it becomes difficult to detect misconduct, resolve disputes, or maintain trust in a multi-vendor, multi-platform ecosystem.

* Operational guidance and standardized profiles — a set of shared deployment guidelines, recommended defaults, and well-defined configuration profiles that help different implementations behave consistently. This includes safe presets for non-expert task owners (e.g., “fast match” , “high assurance”), normative guidance on how to apply admission policies and search budgets, and test vectors that allow implementers to verify correct behavior. Without this layer of operational guidance, platforms may interpret the same mechanisms differently, leading to fragmentation, inconsistent trust decisions, and unpredictable user experience.

7. Security Considerations

Security considerations are as outlined within the document under the privacy and security requirements

8. IANA Considerations

This document has no IANA actions.

9. Informative References

[DNSAIDsite]
"DNS-AID web site for implementation examples", n.d., <https://dns-aid.org/>.

[I-D.draft-mozleywilliams-dnsop-dnsaid]
Mozley, J., Williams, N., Sarikaya, B., and R. Schott, "DNS for AI Discovery", Work in Progress, Internet-Draft, draft-mozleywilliams-dnsop-dnsaid-01, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-mozleywilliams-dnsop-dnsaid-01>.

Contributors

Hesham Moussa
Huawei Canada
Email: hesham.moussa@huawei.com

Arashmid Akhavain
Huawei Canada
Email: arashmid.akhavain@huawei.com

Roberto Pioli
Independent
Email: roberto.pioli@example.com

Jim Mozley
Infoblox, Inc.
Email: jmozley@infoblox.com

Nic Williams
Infoblox, Inc.
Email: nwilliams@infoblox.com

Authors' Addresses

Hesham Moussa
Huawei Canada
Email: hesham.moussa@huawei.com

Arashmid Akhavain
Huawei Canada
Email: arashmid.akhavain@huawei.com

Roberto Pioli
Independent
Email: roberto.pioli@example.com