Network Working Group
Internet-Draft
Intended status: Informational
Expires: 6 December 2026

A. Akhavain
H. Moussa
Huawei Technologies Canada
D. King
Old Dog Consulting

4 June 2026

Problem Statement for the Discovery of Agents, Workloads, and Named Entities (DAWN)
draft-akhavain-moussa-dawn-problem-statement-03

Abstract

Interacting entities such as agents, tasks, users, workloads, data, compute, etc., in AI ecosystem/network are proliferating, yet there is no standardised way to discover what entities exist, what attributes such as skills, capabilities, physical characteristics, etc., they posses, what services they offer, or how to reach them across organisational boundaries.

Discovery today relies on proprietary directories or manual configuration, creating fragmented ecosystems that prevent cross-domain collaboration.

This document describes the problem space that motivates Discovery of Agents, Workloads, and Named Entities (DAWN). It clarifies the scope of work within entity ecosystems, identifies why current approaches are insufficient, and outlines the challenges a standardised discovery mechanism must address. It does not propose a specific solution or protocol.

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/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 6 December 2026.

Table of Contents

1. Introduction

Entities in entity ecosystem collaborate to render services and follow the lifecycle shown in Figure 1.

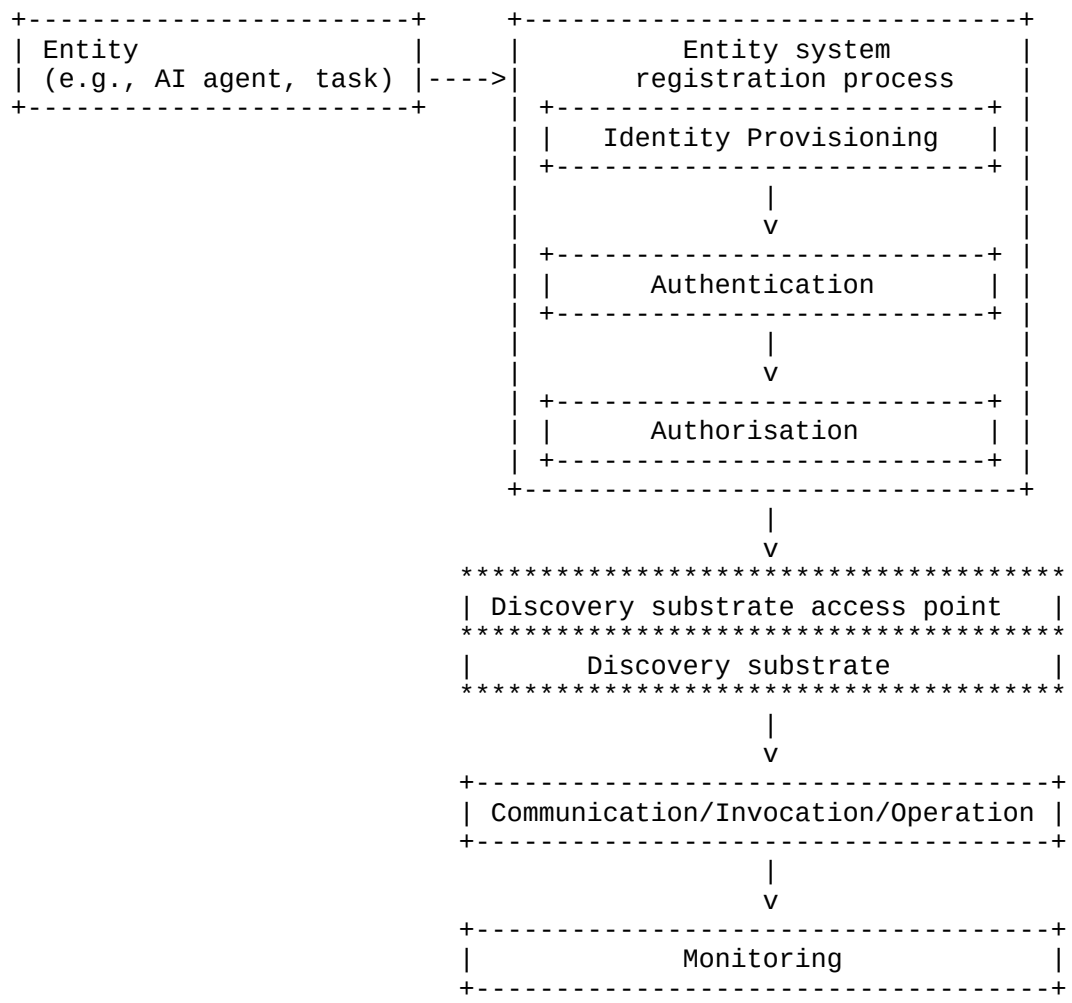

Figure 1: An example of Entity Lifecycle

As shown in Figure 1, an entity will pass through a set of important functional blocks before it becomes active and start interacting with other entities in the ecosystem. This document focuses on the discovery problem space in the above diagram namely: "Discovery substrate access point" and "Discovery substrate".

Entities increasingly need to discover, connect with, and collaborate with one another to deliver services. This discovery process is driven by the need to identify the most suitable set of entities that satisfy the requirements of a particular service. To achieve this, an entity must be able to find others based on attributes such as

skills, capabilities, physical characteristics, names, and other relevant qualities they possess. Obviously, as static configuration is impractical at scale, an automated discovery of entities, their skills, and their capabilities becomes essential.

Discovery within an AI ecosystem can be multi-dimensional and complex. A discovery request may trigger a cascade of subsequent discovery requests by other AI entities, occurring either sequentially or in parallel and the process might become unbounded. In addition, the discovery step can be interactive. For example, an entity might be looking for another entity that might not be available at the time of request (e.g., the desired entity might be busy). Furthermore, entities might be looking for a variety of other entities with different cards/descriptors. Discovery might also be subjected to either a system wide or local policy and might span cross organisation. There also challenges w.r.t the nature of discovery request itself as will be explained later in this document.

Assuming that trust has already been established between entities and within the ecosystem in the steps prior to the discovery stage, the discovering entity must learn what the remote entity does, what attributes it posses, how to communicate with it, etc.

This document describes the problem space and informs the development of requirements set out in [I-D.king-dawn-requirements] and future solution proposals for Discovery of Agents, Workloads, and Named Entities (DAWN).

2. Terminology

This document uses the following terms defined in [I-D.farrel-dawn-terminology]:

* Attributes
* Discoverable Object
* Discovery
* Discovery Mechanism
* Entity
* Minimum Discoverable Information

3. Motivation

The main motivation behind DAWN is to tackle the discovery problem space within the entity ecosystem. It is driven by a few factors:

* Discovery use cases in real-world
  - Many practical scenarios require discovery, not only for agent-to-agent, but also agent-to-tools, agent-to-task, task-to-agent, and other forms of entity interaction.
* Limitations of traditional discovery methods
  - Existing discovery mechanisms are not designed to natively handle scenarios where entities are dynamic, mobile, cross-domain, or when they have complex attributes.
* Current approaches are ad-hoc, entity specific, and and do not scale:
  - Even in today’s implementations (e.g., MCP-based systems or A2A-based systems), discovery tends to be contained and handled through simple mechanisms such as name lookup, search engines, or static agent cards/tool cards. These approaches work only in small, closed environments. They do not address challenges such as inter-domain discovery, dynamic endpoint association, chained discovery queries, blind or exploratory search sessions, or large-scale environments. In addition, they do not address the need of other discoverable entities such as task, workloads, etc.
* Emergence of discoverable entities, discoverable objects, and discovery mechanism:
  - Entities may have associated MDIs (e.g., task , capabilities, endpoints, policies), and that a discovery substrate/mechanism/vehicle is needed. The discovery substrate may implement unified mechanism or may support multiple discovery strategies depending on the scenario.

3.1. Example of Discovery Lifecycle in AI Ecosystem

Consider a task owner (e.g., an entity such as an end user, AI agent, model, data owner, resource/compute owner) which intends to submit a task to the ecosystem and, as shown in Figure 1, has already been processed and accepted by the entity registration block. The following describes the steps after which the entity becomes available for discovery.

1. Discovery substrate access point validates the task owner’s credentials and verifies that its associated discoverable object meets compliance requirements. The discoverable object is what the discovery substrate makes available/visible to system participants. It contains entity’s different attributes and information that others need to initiate an interaction session with it once they discover the entity.

2. Task owner submits its tasks to the system. Submitted tasks are entities themselves. They have their own discoverable object (task card in this case) which the discovery substrates makes available/visible to other entities in the ecosystem once submitted tasks pass through the entity registration block.

3. A registered entity (e.g., an AI agent) then issues a discovery query to identify and/or locate suitable tasks it can perform, or to find other agents, resources, etc., it must interact with to complete a given task.

4. The discovery substrate processes the above query and returns the relevant discoverable objects such as tasks, agents, resources, etc., to the entity that issued the query. It must be noted that the nature and structure of the query, the format of the discoverable objects (e.g., standardised object cards), and the discovery mechanism employed (e.g., simple name lookup or semantic matching) are key factors influencing the accuracy, volume, timeliness, etc., of the results.

   For example, the querying entity may need to provide details about its skills, capabilities, pricing, or other relevant attributes so the discovery substrate can match its request with an appropriate subset of registered entities in the system.

5. Upon receiving the discovery results (e.g., a list of suitable entities), the querying entity (e.g., an AI agent) might need additional information before initiating its interaction with the discovered entities. For example, it might need to know more about the parent entity of the discovered entity whose name/ID can be potentially found in the discovered entity’s discoverable object.

The example above illustrates the broader concept of discovery within an ecosystem. Other factors such as entity's mobility can further complicate the problem space. The example, underscore the significance and complexity of the problem space that DAWN aims to address. It highlights why a structured problem definition, clear requirements, and well-designed solutions are essential for enabling robust, scalable, and interoperable discovery across diverse entities and use cases.

4. Applicability

The challenges outlined in the Motivation section underscore the need for mechanisms that allow entities and other discoverable entities to dynamically locate and interact within a decentralised ecosystem. DAWN is applicable in scenarios where discovery serves as a key enabler for autonomous operation, collaboration, and adaptive decision-making. In such systems, entities may need to find other entities or entities, while task owners including agents, users, or services may advertise tasks that suitable entities can discover and execute. Data sources can make datasets discoverable to support reasoning or training by AI agents and models. Compute resources may advertise their capabilities, serving as rendezvous points for entities, models, and datasets to facilitate training workflows. Similarly, models may advertise their functionality to allow users or entities to discover them for inference tasks. The following subsections provides more details, illustrating the contexts in which DAWN provides value and a consistent foundation for the functional requirements and design considerations.

4.1. Entities discovering Entities

Entities frequently need to locate other entities to coordinate actions, share information, or engage in collaborative workflows. In some situations, an entity may already be aware of a counterpart possessing the required skills or capabilities. In other cases, entities must actively query the system to discover suitable peers by specifying the skills or attributes they are looking for. DAWN provides a framework to support both modes of discovery, enabling dynamic, capability-driven interactions in decentralised and heterogeneous environments.

DAWN enables entities to advertise their presence, capabilities, and status, facilitating peer-to-peer interactions in dynamic, multi-entity ecosystems. This is particularly relevant in large-scale deployments, multi-domain environments, or systems where entities may join or leave unpredictably.

4.2. Entities discovering tasks

In addition to discovering other entities, entities may need to locate tasks that require attention or contribution within a system. Tasks can be advertised by users, other entities, or services, along with information such as required skills, priority, or dependencies. Since entities are aware of their own capabilities, they can match their skill sets against advertised tasks and proactively apply for or claim tasks for which they are suitable. DAWN provides mechanisms to make tasks discoverable, enabling entities to query, filter, and select tasks efficiently, supporting autonomous orchestration, dynamic workflow formation, and load distribution across heterogeneous environments.

4.3. Entities/models discovering data

Entities and models often require access to data distributed across multiple systems or administrative domains to perform training, inference, or reasoning tasks. This includes datasets, knowledge bases, or document repositories that may be advertised as discoverable entities with information such as format, availability, and access requirements. In Retrieval-Augmented Generation (RAG) scenarios, entities or models need to dynamically locate relevant external knowledge sources or documents to supplement generative reasoning, enabling more accurate and context-aware responses. Additionally, data in these environments may be dynamic, changing over time as new information is added or existing data is updated. DAWN provides mechanisms for discovering, tracking, and querying such evolving data sources, allowing entities and models to identify relevant information in real time while respecting access controls and provenance information.

4.4. Entities/models and data discovering compute

Entities and models often require access to compute resources to perform tasks such as training, fine-tuning, or indexing. These resources may be distributed across multiple systems or administrative domains, and their availability, capacity, or configuration can change over time. In this context, compute resources can serve as rendezvous points, allowing entities, models, and datasets to converge and interact efficiently. DAWN provides mechanisms for entities, models, and data sources to discover compute resources that meet their requirements, including hardware capabilities, scheduling constraints, and current availability. By enabling dynamic identification of suitable compute nodes, DAWN supports elastic scaling of training workloads, efficient utilisation of heterogeneous infrastructure, and adaptive collaboration in decentralised and changing environments.

4.5. Discovering models for inference

Users, agents, and services (i.e., entities) may need to leverage pre-trained models for inference in tasks such as prediction, recommendation, or decision-making. Models may be distributed across various systems or administrative domains, and their availability, capabilities, or performance characteristics can evolve over time. DAWN supports mechanisms to discover models that are most suitable for different contexts. This enables users, agents, services, etc. to dynamically adapt to newly available models, take advantage of improvements, and ensure interoperability in heterogeneous and evolving environments.

4.6. Taxonomy of Entities

Classifying entities along multiple dimensions helps derive common as well as specific requirements in discovery processes and protocols for different types of entities.

The following aspects are listed. This is not an exclusive list and it is expected that the list will evolve iteratively as new use cases are developed. Primary dimensions include:

* Identity Binding:
  Describes the relationship between an entity’s identifier and its runtime deployment artifact (e.g., a single instance, cluster, physical device, IP/port, etc.). This determines the persistence of the identifier as well as the lifecycle and reachability characteristics of the resolved target object. Discovery mechanisms need to consider these characteristics when designing caching policies, refresh frequencies, and re-discovery conditions.

  For example: An AI agent may be bound to a dynamic service instance in a cloud computing environment, or it may be bound to a specific physical mobile device. The former may have an ephemeral identifier that changes across sessions, while the latter has a persistent identifier.

* Control Ownership: The administrative entity that has authority over the entity’s lifecycle, configuration, and policy. This determines who publishes discovery information and who can update or withdraw it.

  For example: A computing workload may be owned by the user who submits it (control ownership lies with the user), while a network function deployed by a service provider is owned by that provider’s organization.

* Responsible Party: The party that is operationally accountable for the entity's behaviour, including security, correctness, and policy compliance. This may differ from the control owner. This dimension relates to trust and attestation requirements in discovery.

  For example: In a Retrieval-Augmented Generation (RAG) scenario, the data source that is discovered and retrieved is the responsibility of its data owner. Meanwhile, the AI agent that consumes that data and generates new results is the responsibility of its agent provider.

* Dynamic Characteristic: The rate and predictability of change in the entity's discoverable properties (e.g., location, availability, load). This influences how discovery information should be published and cached.

  For example: The current load of a computing workload changes rapidly (high dynamism). In contrast, a deployed network function change rarely (low dynamism), allowing long-lived caching.

* Discovery Payload Richness The level of detail and structure of discovery information that an entity can publish or that a query can expect to retrieve. This determines whether discovery information is embedded directly in the response or provided via references, and it also affects query language capabilities and caching granularity.

  For example: A simple network function may only advertise its IP address and port (minimal). An AI agent may publish a full capability card listing its skills, input/output schemas, and authentication method (rich).

* Cross-domain Visibility Which administrative domains are permitted to discover an entity, and what level of discovery information is exposed to each domain. This impacts the discovery mechanism to support access control, inter-domain authentication, and selective publication of information.

  For example: An AI agent's capabilities may be discoverable across domains to enable collaboration, but the backend computing services that execute the agent's tasks are only discoverable within the same administrative domain for security or policy reasons.

The following presents a table of entity types. This is not an exclusive list and it is expected that more entity types will continue to be added as new use cases are developed. The table shows:

Entity Type	Identity Binding	Control Ownership	Responsible Party	Dynamic Characteristic
AI Agent	End device /Instance	Organization /User	Developer& Deployer&User	High
Software Service	Instance /Cluster	Provider Organization	Developer & Deployer	Medium
Compute Workload	Variable ID	Submitter & Orchestrator	Submitter & Deployer	High
Network Function	Node /Instance	Provider Organization	Operator	Low
Application Endpoint	IP/Port /Instance	Owning Organization	Developer & Deployer	Medium

Figure 2: Taxonomy of Entities

5. Functional Requirements

5.1. Discovering Entities and Query Granularity

Discovery in ecosystem should support different levels of granularity. Queries may range from broad capability-based searches (such as identifying all models with mathematical abilities) to more specific lookups. The discovery system should also enable entities to be found through the attributes reflected in their discoverable objects that capture aspects like their skill sets, functionality, name/ID, ratings, regional associations, and more.

5.2. Discovering Response and Minimum Discoverable Information

Information an entity discovers about another entity must be meaningful and useful for delivering the required service. Accordingly, a response to a discovery query should include attributes that describe the discovered entity: such as what it can do, the skills it possesses, the protocols it supports, the security guarantees it claims to offer, the policies it can potentially enforce, its pricing for services, its current operational status

(e.g., available, busy, or offline), communication means, etc.

Such information can be either embedded within the entity's discoverable object or retrieved through a subsequent interaction outside the discovery substrate (for example, after discovery, an interview-style exchange may be conducted using the communication method indicated by the entity).

In either case, there is a need for a standardized structure for discoverable objects that provides the minimum set of information needed for the discovery substrate to return results that meaningfully support service delivery within the AI ecosystem.

5.3. Cross-Domain Collaboration

Entities operating across organisational boundaries need to discover counterparts without depending on a shared infrastructure. For example, a customer-service agent in one organisation may need to find a logistics-tracking agent in another. Models in one administrative domain may need to find compute resources in another administrative domain for training. Similarly, a model or agent in one domain might need to use data in another domain for retrieval-augmented generation (RAG) based inference. Current platform-specific mechanisms do not interoperate, so entities remain invisible outside their own ecosystem.

Administrative domains are typically unwilling to disclose their internal structures or detailed operational information to one another. In traditional networking, for instance, they use abstraction and aggregation techniques to share only high-level insights about their operations. A standards-based mechanism to support controlled information sharing while ensuring administrative domain interoperability without exposing sensitive internal details is potentially desirable.

5.4. Discovery and Dynamic Attributes in Discoverable Objects

Entities whose discoverable objects contain dynamic attributes introduce distinct challenges for discovery. Dynamic attributes such as location information, dataset samples, compute capacity, etc., can change at different rates. These dynamics introduce variability that static discovery systems are not designed to handle. Such dynamic attributes complicate the assumptions in traditional discovery approaches and demand careful consideration when defining the problem space.

5.5. Broker and Aggregator Discovery

In large-scale networks, entities may need to discover intermediary broker nodes that operate across multiple administrative domains and provide dynamic operational information such as availability, capabilities, or decision guidance via the use of mechanisms that support interoperable and standards-compliant discovery procedures.

In large-scale networks, entities may need to discover intermediary broker nodes. These brokers often operate across multiple administrative domains with different jurisdictions. They also provide dynamic operational information, such as availability, capabilities, or decision guidance. In these scenarios, the intermediary brokers might need to discover other brokers. This makes the broker nodes another type of entity with its own discoverable object in an ecosystem. Discovery substrate needs to provide support for this capability via standards-compliant procedures.

5.6. Human-Initiated Discovery

Operators need to discover and inspect entities for operational purposes: auditing deployed agents, verifying capability claims, or troubleshooting failures. Discovery must be usable by humans through standard tooling, not only by automated systems.

5.7. Discovery and OAM

Discovery systems require operational visibility, management, and diagnostic capabilities to ensure reliable operation across potentially distributed and multi-domain environments. Operators should be able to determine the behaviour of the discovery system and understand the reasons for discovery outcomes, failures, and policy decisions. The following outlines some of the operational aspects that need to be considered.

* Discovery Infrastructure Observability and Diagnostics: Operational managers require visibility into the availability, health, performance, and reachability of discovery system components, as well as the ability to determine whether discovery transactions can be successfully completed. Operational managers should also be able to observe discovery activity and determine the reasons for discovery outcomes, failures, performance degradation, and other system behaviours.

* Policy and Security Visibility: Discovery behaviour may be influenced by policies, trust relationships, authorisation decisions, and security mechanisms. Operational managers need visibility into how these factors affect discovery outcomes and the ability to identify abnormal or unauthorised activity.

* Discovery Information Tracking Management: Operational managers need to be able to monitor the status of discoverable objects and information in the system including its creation, modification, propagation, expiration, and removal. Clients might also need to access different statistics about their discoverable objects.

* Discovery Chains and Multi-Domain Operations: Discovery may involve a sequence of dependent discovery operations, potentially spanning multiple systems and administrative domains. Operational managers require visibility into discovery chains, dependencies, and interactions in order to understand discovery outcomes and to monitor and troubleshoot failures or performance issues that may occur along the discovery path.

Tooling to enable this function should be integral to the design of any solution and specific requirements for operational considerations can be found in [I-D.king-dawn-requirements].

Some guidance (specific to DNS, but more generally applicable to any discovery system) can be found in [RFC6168].

6. Current Approaches and Their Limitations

6.1. Proprietary Directories

Cloud providers, AI platforms, and other entity systems maintain their own registries, tightly coupled to their ecosystem. Entities registered in one platform are invisible to another, creating walled gardens.

6.2. Static Configuration

Manually configured endpoint lists cannot scale, cannot adapt to dynamic environments, and cannot convey the capability and trust metadata needed for cross-domain discovery.

6.3. DNS-SD and SRV Records

TBD

6.4. Well-Known URIs

TBD

6.5. Ad Hoc Agent Discovery Proposals

TBD

7. Core Challenges

7.1. Discovering Skills and Capabilities at Scale

The central challenge is enabling entities to discover other entities based on what they can do, such as:

* Agents
* Skills
* Capabilities
* TBD

A discovery mechanism that supports structured, scalable discovery of an entity’s capabilities across organisational boundaries is therefore required.

7.2. Fragmented Discovery Ecosystem

Each platform develops its own discovery approach. This fragmentation prevents entities from being discoverable across boundaries and limits the value of interoperable protocols such as A2A and MCP.

7.3. Trust in Discovery Information

When discovery crosses organisational boundaries, the discovering entity must verify that the information is authentic. Without authenticated discovery, entities are vulnerable to poisoning attacks directing them to malicious endpoints. DNSSEC provides a foundation, but discovery mechanisms must be designed to use it.

7.4. Scalability and Decentralisation

Discovery must operate at Internet scale without a single centralised registry. Each organisation must be able to publish its entities’ capabilities independently, mirroring the DNS delegation model.

7.5. Static Versus Dynamic Properties

Entity properties range from static (type, supported protocols, skills) to dynamic (availability, load, capacity). A discovery mechanism must handle both without causing stale results or excessive query load.

7.6. Extensibility

New agent types, skill taxonomies, and capability formats will emerge. Discovery must accommodate them without changes to the core mechanism.

8. Relationship to Existing Work

TBD

9. Security Considerations

This document describes a problem space, not a protocol.

Discovery information is a high-value target. Poisoned responses could direct entities to malicious endpoints. Any mechanism must provide integrity and authenticity guarantees. Specific security-related requirements for any solution are captured in [I-D.king-dawn-requirements].

Cross-domain discovery raises two distinct trust questions: whether the discovery source is authoritative, and whether the registered entity is what it claims to be.

Discovery may expose sensitive information about an organisation’s entities and capabilities. Selective visibility mechanisms are needed.

10. Privacy Considerations

Querying for entities may reveal the discovering entity’s intentions or interests. Discovery should minimise information leakage through the query process.

Published entity properties, such as skills, capabilities, and organisational affiliations, may be sensitive. Entities and their operators should control the granularity and audience of published information.

11. Operational Consideration

TBD

12. IANA Considerations

This document makes no requests of IANA.

13. Potential Topics for the Use Case Document

TBD

Acknowledgements

Thanks to Adrian Farrel and Linda Dunbar for review comments.

References

Normative References

[I-D.farrel-dawn-terminology]
Farrel, A., Yao, K., Schott, R., and N. Williams, "Terminology for the Discovery of Agents, Workloads, and Named Entities (DAWN)", Work in Progress, Internet-Draft, draft-farrel-dawn-terminology-02, 4 June 2026, <https://datatracker.ietf.org/doc/html/draft-farrel-dawn-terminology-02>.

[RFC6168] Hardaker, W., "Requirements for Management of Name Servers for the DNS", RFC 6168, DOI 10.17487/RFC6168, May 2011, <https://www.rfc-editor.org/rfc/rfc6168>.

Informative References

[I-D.king-dawn-requirements]
King, D. and A. Farrel, "Requirements for the Discovery of Agents, Workloads, and Named Entities (DAWN)", Work in Progress, Internet-Draft, draft-king-dawn-requirements-01, 28 April 2026, <https://datatracker.ietf.org/doc/html/draft-king-dawn-requirements-01>.

Contributors

Kehan Yao
China Mobile
China
Email: yaokehan@chinamobile.com

Authors' Addresses

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

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

Daniel King
Old Dog Consulting
United Kingdom
Email: daniel@olddog.co.uk