Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 20 September 2026

B. Liu
Huawei Technologies
M. Han
China Unicom

19 March 2026

Agent Metadata Synchronization Protocol
draft-liu-agent-metadata-sync-protocol-00

Abstract

The Internet of Agents (IoA) requires a robust infrastructure to manage the lifecycle, discovery, and interaction of autonomous agents across distributed network domains. While the Agent Gateway (AGW) provides a localized control point, large-scale deployments necessitate a mechanism for multiple gateways to synchronize agent metadata and routing information. This document specifies the Agent Metadata Synchronization Protocol (AMSP). AMSP facilitates the exchange of Agent Records between hierarchical gateways (Level-1 and Level-2), ensuring global reachability, efficient resource utilization, and loop-free metadata propagation.

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 20 September 2026.

Copyright Notice

Table of Contents

1. Introduction

The Internet of Agents (IoA) paradigm shifts communication from host-to-host towards dynamic, autonomous Agent-to-Agent interactions. As discussed in [I-D.liu-rtgwg-agent-gateway-requirements], the introduction of Agent Gateways (AGWs) is crucial for addressing the scalability, security, and efficiency challenges inherent in large-scale, unmanaged Agent networks. An Agent Gateway acts as a specialized intermediary that facilitates secure and efficient interaction between agents that either within a network domain or

across different network domains.

A natural progression in this architecture is the deployment of multiple, possibly distributed, Agent Gateways. For instance, an organization may deploy AGWs in different geographic locations, across various cloud environments, or to serve distinct business units to optimize for latency, locality, data sovereignty, or fault tolerance. In such scenarios, while Agents interact with their local gateway, the gateways themselves must collaborate to enable seamless communication and service continuity across the entire multi-gateway infrastructure.

This document specifies the Agent Metadata Synchronization Protocol (AMSP). AMSP defines the mechanisms for interconnection between Agent Gateways, the procedures for efficient metadata exchange, and the underlying design principles. By enabling a unified view of available agents across multiple gateway instances, AMSP ensures that the "Internet of Agents" (IoA) can scale effectively while maintaining robust governance and reachability.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Functional Requirements of Interaction between Agent Gateways

This sections discusses the high-level requirements of the interaction between agent gateways. Gap analysis of current protocols and designing of AMSP is described in below scetions accordingly.

2.1. Agent Gateway Discovery

For inter-gateway communication to be established, an Agent Gateway must first discover its peers. This discovery process can be static (via configuration) or dynamic. The protocol MUST support mechanisms for an AGW to discover and establish control sessions with other gateways, particularly in a hierarchical architecture where Level-1 gateways need to locate their designated Level-2 gateway(s), and Level-2 gateways need to discover each other. The discovery mechanism must be resilient and allow for network topology changes.

2.2. Metadata Synchronization

The core function of inter-gateway interaction is the synchronization of Agent metadata. The primary metadata unit is the Agent Record. A standardized format for this record, such as the AgentCard format [AgentCard], SHOULD be used to ensure interoperability. The metadata to be synchronized includes, but is not limited to:

* Agent Identifier (Agent ID)
* Agent Capabilities/Skills
* Reachability Information (e.g., the URI of the gateway serving the Agent)
* Associated Security Policies and Trust Attributes

The synchronization process MUST ensure that metadata is propagated efficiently and consistently across the inter-gateway network, respecting the defined hierarchy and any configured policies.

2.3. Forwarding Table Forming for Agent Messages

Based on the synchronized Agent Records, particularly the Agent IDs, skills, and capabilities, each Agent Gateway MUST be able to construct and maintain a logical forwarding table. This table maps a target Agent ID/skill to the next-hop gateway (or directly to the Agent, if local) for the purpose of routing application-layer Agent messages. The formation of this table is a direct outcome of the metadata synchronization process and is fundamental to enabling inter-domain Agent communication.

3. Gap Analysis of Current Approaches

An examination of existing protocols reveals that they are not fully suited to meet the specific requirements of inter-gateway Agent metadata exchange.

3.1. Routing Protocols

Traditional routing protocols (e.g., BGP, OSPF, IS-IS) are designed to exchange network-layer reachability information (IP prefixes). They operate on network and transport layer identifiers and are not equipped to handle the semantics of application-layer entities like Agents. They lack the ability to carry and process rich, variable-length metadata such as capability descriptions, skill sets, or application-specific security policies. Extending these protocols to support such data would be complex and would deviate from their core

design principles.

Furthermore, the high churn rate of Agents would lead to excessive route flapping if mapped directly to traditional routing tables.

3.2. Pub/Sub Protocols

General-purpose publish/subscribe protocols (e.g., MQTT, XMPP) are efficient for distributing messages based on topics. However, they are typically designed for data distribution rather than state synchronization with strong consistency and routing table computation requirements. They often lack built-in mechanisms for loop prevention, policy-based filtering on the publisher side, and the construction of a forwarding state that maps identifiers to next hops. And they typically require a centralized broker or a flat cluster, which does not align with the hierarchical and decentralized nature of cross-domain IoA deployments.

While a Pub/Sub model could be a component of the AMSP, it alone does not fulfill the complete set of requirements for a routing and synchronization protocol between gateways.

4. Design Principles of AMSP

To address the identified gaps and requirements, the Agent Metadata Synchronization Protocol (AMSP) is designed according to the following principles.

4.1. Protocol Layer

AMSP is designed as an application-layer (Layer 7) protocol. It is intended to operate on top of reliable transport protocols. The initial specification will define bindings to two common application frameworks:

* JSON + HTTP

* gRPC + HTTP

This approach leverages widely adopted web technologies, facilitating implementation, debugging, and integration with existing network services and orchestration platforms.

4.2. Agent Gateway Identification

An Agent Gateway participating in AMSP MUST be uniquely identifiable. Two primary methods are supported:

* IP Address + Port: A gateway can be identified by its IP address and a designated AMSP service port. This is a straightforward method suitable for many deployments.

* Uniform Resource Identifier (URI): A gateway can be identified by a URL (e.g., https://gw1.example.io/amsp). This method offers greater flexibility, enabling the use of DNS for addressing, load balancing, and the colocation of AMSP services.

4.3. Gateway Hierarchy (Level-1, Level-2)

To ensure scalability and operational efficiency, AMSP defines a two-level hierarchical model for gateway interconnection:

* Level-1 Gateway (L1 AGW): These gateways are deployed closer to the edge, serving specific localities such as a campus, a data center, or a broadband access network (BRAS). A Level-1 gateway maintains Agent Records only for the Agents directly connected to or registered with it. It discovers and connects to its designated Level-2 gateway(s) to report local records and to query or subscribe to records for Agents outside its local domain.

* Level-2 Gateway (L2 AGW): These are core nodes that maintain a global or regional view of Agent metadata. A Level-2gateway could be operated by a network provider, a large enterprise, or a major Agent service provider. Level-2 gateways maintain the full set of Agent Records for their domain and are responsible for synchronizing this full view with other Level-2 gateways. The addresses (IP/port or URL) of Level-2 gateways can be publicly advertised (similar to public DNS servers) or privately disclosed to authorized Level-1 gateways.

This hierarchical structure provides two key advantages:

* Improved Convergence: Full-mesh synchronization of all records is confined to a smaller set of Level-2 gateways, enhancing overall synchronization speed and stability.

* Reduced State at the Edge: Level-1 gateways only need to store local records and a subset of remote records that are actively being used (via subscription), significantly reducing their memory and processing requirements.

4.4. Gateway Discovery

The discovery process follows the established hierarchy.

* Level-1 Discovers Level-2 Gateway: A Level-1 gateway is configured, either manually or via automated mechanisms (e.g., DHCP options, anycast), with the address of its upstream Level-2 gateway(s). It then initiates a connection to this Level-2 gateway to establish an AMSP peering session.

* Level-2 Gateway Inter-Discovery: Level-2 gateways are typically configured with the addresses of their peers (neighboring Level-2 gateways). This configuration can be done manually by an operator or through a controller or orchestration system. Once configured, Level-2 gateways establish AMSP sessions with their peers to facilitate full-mesh or partial-mesh synchronization of the global metadata set.

4.5. Synchronization mechanisms

The synchronization of Agent Records is the core function of AMSP. The mechanisms are defined separately for the two levels of the hierarchy. Per current common practise, an Agent Record is estimated to be between 5-12 KB in size. For example, storing records for 1 million Agents would therefore require approximately 12GB of storage, which is feasible for a Level-2 gateway.

4.5.1. Synchronization between Level-1 and Level-2 Gateways

The interaction between a Level-1 and its Level-2 gateway follows a hybrid push-and-pull model.

1) Local Registration and Upstream Push: When an Agent registers with its local Level-1 gateway, the gateway stores the Agent’s Record locally. It then immediately pushes this record to its connected Level-2 gateway(s). This ensures the upper layer has an up-to-date view of all edge Agents.

2) On-Demand Subscription from Level-1: When a Level-1 gateway receives a resolution request from a local Agent (e.g., a natural language query to find Agents with a specific capability), it first checks its local cache. If the required information is not present, it sends a subscription request to its Level-2 gateway for the relevant topic or capability. This is a pull-based, on-demand mechanism.

3) Targeted Push from Level-2: The Level-2 gateway, upon receiving a subscription, pushes the relevant Agent Records to the requesting Level-1 gateway. This ensures that the edge gateway only receives the metadata it actually needs.

4) Update Propagation: If an Agent Record changes (e.g., an Agent goes offline or its capabilities are updated), the change is first pushed from the Level-1 to the Level-2 gateway. The Level-2 gateway then propagates this update to all Level-1 gateways that have subscribed to that specific record or its topic. It also propagates the update to other Level-2 gateways.

5) Subscription Cleanup: To manage resources, a Level-1 gateway MAY unsubscribe from a topic or a set of records if they have not been used for a defined period.

4.5.2. Synchronization between Level-2 Gateways

Synchronization among Level-2 gateways aims to maintain a consistent global view. This process requires robust loop-prevention mechanisms, drawing inspiration from path-vector protocols like BGP.

* Full State Synchronization: Level-2 gateways exchange complete views of their Agent Record database upon session establishment. Subsequently, only incremental updates (changes) are propagated.

* Loop Prevention Mechanisms:

-- Source Pruning (Split Horizon): When a Level-2 gateway sends records to its peers, it MUST implement a form of source pruning. Records received from a specific neighbor are not transmitted back to that same neighbor. This is analogous to maintaining a separate Routing Information Base (RIB) per peer.

-- PATH Attribute: Each update message for an Agent Record MUST carry a PATH attribute. This attribute is an ordered list of the URIs of the Level-2 gateways through which the record update has passed.

a) When a gateway sends an update, it prepends its own URI to the PATH attribute.

b) Upon receiving an update, a gateway checks the PATH attribute. If its own URI is already present in the PATH, the update is silently discarded to prevent a loop.

c) If a gateway receives multiple updates for the same Agent Record from different paths, it SHOULD select the update with the shortest PATH length (i.e., the one that has traversed the fewest Level-2 gateways) as the best path. This helps in selecting the most direct path to the originating gateway.

4.5.3. Synchronization Policies

The protocol should be flexible enough to support different policy models for controlling the flow of metadata. Several approaches are possible:

* Send-Filter Policy: In this model, the sending gateway applies filters to determine which records to transmit to which neighbors.

-- Advantage: Allows for strong policy control at the source, which is useful in scenarios with strict data governance.

-- Disadvantage: May result in the transmission of many records that are not of interest to the receiver, leading to bandwidth waste as the receiver drops them.

-- Use Case: Suitable for more controlled environments where outbound policy is paramount.

* Pub/Sub Model: The receiving gateways explicitly subscribe to the topics or specific records they are interested in. The sending gateway only publishes updates related to those active subscriptions.

-- Advantage: Highly efficient, as data is only sent where it is needed. This model is receiver-driven.

-- Disadvantage: It is more complex to implement, requiring the management of subscription state. It may offer less fine-grained control to the publisher over what is being sent.

-- Use Case: Well-suited for open, large-scale discovery scenarios where edge gateways only want relevant information.

* Hybrid Model: A combination of both approaches. A sending gateway can have its own export policies, while also respecting subscriptions from its neighbors. This offers maximum flexibility.

-- Advantage: Combines the policy control of the send-filter model with the efficiency of the pub/sub model.

-- Disadvantage: Results in the most complex implementation, requiring both policy engines and subscription state management.

-- Use Case: Complex operational environments where both publisher control and receiver efficiency are required.

AMSP SHOULD be designed to accommodate all three models, allowing implementations and deployments to choose the most appropriate mechanism.

4.6. Agent-specific Routing Table Forming

The synchronization of metadata directly leads to the formation of application-layer routing tables on each gateway. This process is illustrated with a simple example.

Consider a network with three Agent Gateways: GW1, GW2, and GW3. GW2 is a Level-2 gateway, while GW1 and GW3 are Level-1 gateways connected to GW2. Agent2 is directly connected to GW2, and Agent1 is connected to GW1.

1) Record Synchronization and Forwarding Table Generation:

- Agent2 registers with its local gateway, GW2. GW2 stores Agent2’s Record locally. It then generates a local forwarding table entry indicating a direct connection to Agent2: ‘<Agent ID2, Interface/Connection to Agent2>‘.

- GW2, acting as a Level-2 gateway, synchronizes Agent2’s Record to its neighbor, GW3. The synchronization message includes the Agent Record and a PATH attribute containing GW2’s URI.

- GW3 receives the Record for Agent2, stores it, and installs a forwarding table entry pointing to GW2 as the next hop: ‘<Agent ID2, Connection to GW2>‘.

- GW3 further synchronizes the Record to its neighbor, GW1, appending its own URI to the PATH attribute (PATH: GW3 -> GW2).

- GW1 receives the Record for Agent2. It stores the Record and installs a forwarding table entry pointing to GW3 as the next hop for Agent2: ‘<Agent ID2, Connection to GW3>‘.

2) Data Plane Forwarding:

- Agent1 sends a request to its local gateway, GW1, to resolve a target, Agent2. GW1’s semantic resolution process matches the request to Agent2’s Record and returns a response to Agent1. In this response, the reachability URI for Agent2 can be replaced with GW1’s own URI to simplify routing for the Agent.

- Agent1 establishes a data connection to the provided URI (which points to GW1) and sends its message for Agent2.

- GW1 receives the message. It examines the destination Agent ID (Agent2) and looks up its application-layer forwarding table. The table entry ‘<Agent ID2, Connection to GW3>‘ instructs GW1 to forward the message to GW3.

- GW3 receives the message, performs its own forwarding table lookup, and forwards the message to GW2 based on its entry‘<Agent ID2, Connection to GW2>‘.

- GW2 receives the message, looks up its table, and delivers the message directly to Agent2. The complete forwarding path is: ‘Agent1 --> GW1 --> GW3 --> GW2 --> Agent2‘.

This example demonstrates how AMSP’s control plane (metadata sync) automatically builds the necessary state for the data plane (application-layer message forwarding) across multiple gateways.

5. Protocol Specification

TBD.

6. Security Considerations

Authentication: All AGW peering MUST be authenticated using mechanisms such as TLS certificates or pre-shared keys.

Integrity: Sync messages SHOULD be signed to prevent tampering with Agent URIs or PATH attributes.

Privacy: Agent Records may contain sensitive metadata. AGWs SHOULD support attribute-level encryption or policy-based redaction during synchronization between domains.

DDoS Mitigation: L2 AGWs SHOULD implement rate-limiting for subscriptions from L1 AGWs to prevent resource exhaustion attacks.

7. IANA Considerations

TBD.

8. Acknowledgements

TBD

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

9.2. Informative References

[I-D.liu-rtgwg-agent-gateway-requirements] Liu, B., Geng, N., Shang, X., Gao, Q., Li, Z., and J. Gao, "Requirements for Agent Gateway", Work in Progress, Internet-Draft, draft-liu-rtgwg-agent-gateway-requirements-01, 27 November 2025, <https://datatracker.ietf.org/doc/html/draft-liu-rtgwg-agent-gateway-requirements-01>.

[AgentCard] "Agent2Agent (A2A) Protocol Specification (Release Candidate v1.0)", 5 March 2026, <https://github.com/a2aproject/A2A/blob/main/docs/specification.md#5-agent-discovery-the-agent-card>.

Authors' Addresses

Bing Liu
Huawei Technologies
No. 156 Beiqing Road
Beijing
China
Email: leo.liubing@huawei.com

Mengyao Han
China Unicom
No. 9 Shouti South Road
Beijing
China
Email: hanmy12@chinaunicom.cn