DMSC Working Group
Internet-Draft
Intended status: Standards Track
Expires: 18 August 2026

X. Li
China Telecom
J. Liu
Beijing University of Posts and Telecommunications
C. Du
Zhongguancun Laboratory
L. Zhang
AsiaInfo Technologies (China) Inc

14 February 2026

Multi-agent Collaboration Protocol Suite
draft-li-dmsc-macp-00

Abstract

This document specifies a Multi-agent Collaboration Protocol Suite based on Agent Gateway, which enables scalable, secure, and semantically driven collaboration among distributed agents across heterogeneous networks. The protocol suite introduces Agent Gateways as control-plane entities responsible for agent registration, authentication, capability management, semantic routing and other functions, while preserving direct peer-to-peer semantic interactions among agents.

Table of Contents

1. Introduction

As multi-agent systems become increasingly distributed across heterogeneous networks and administrative domains, efficient, secure, and semantically meaningful collaboration among agents becomes a critical challenge. Traditional service-oriented or message-based interaction models are insufficient to capture agent-level capabilities, dynamic task decomposition, and semantic intent-driven communication.

This document specifies a Multi-agent Collaboration Protocol Suite based on Agent Gateway (AGW). The suite defines a set of coordinated protocols that enable agent registration, authentication, capability synchronization, semantic routing, task-based invocation, and peer-to-peer semantic interaction. The architecture leverages Agent Gateways as first-class network entities that mediate control, policy enforcement, and orchestration, while allowing agents to directly exchange semantic information once authorized.

The protocol suite is aligned with the architectural principles of control/forwarding plane separation, least-privilege authorization, and session-scoped semantic communication.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] .

3. Terminology

The following terms are defined in this draft:

* Agent: An Agent is a software entity with autonomous decision-making and execution capabilities, capable of perceiving the environment, acquiring contextual information, reasoning, and performing tasks independently or collaboratively with other agents.

* Agent Gateway. The Agent Gateway is the infrastructure service that provides interconnection functions for agent collaboration. .

* Agent Management Center (AMC): It is the trusted infrastructure service responsible for agent identity lifecycle management and credential issuance.

* Agent Identity Code (AIC): An Agent Identity Code (AIC) is a verifiable, globally unique identifier that represents the identity of an Agent.

* Agent Capability Specification (ACS): An Agent Capability Specification (ACS) is a structured description of an agent’s capabilities and service information that can be stored, retrieved, and matched.

* Agent Credential: An Agent Credential is a tamper-resistant data object issued by an Agent Management Center(or its credential authority component), used by an Agent to prove identity attributes and/or authorization to a relying party. Examples include X.509 certificates and security tokens.

* Agent Autonomous Domain: An Agent Autonomous Domain is an administrative and governance domain organized and managed by a specific IoA service provider. An Agent Autonomous Domain typically includes one Agent Management Center, one or more Agent Gateways, and the agents registered within its scope.

4. Multi-Agent Collaboration Function Entity Architecture

The DMSC architecture defines four functional entities and one external actor (User). Each functional entity represents a logical role in the IoA architecture; implementations MAY combine multiple entities into a single product or distribute a single entity across multiple services.

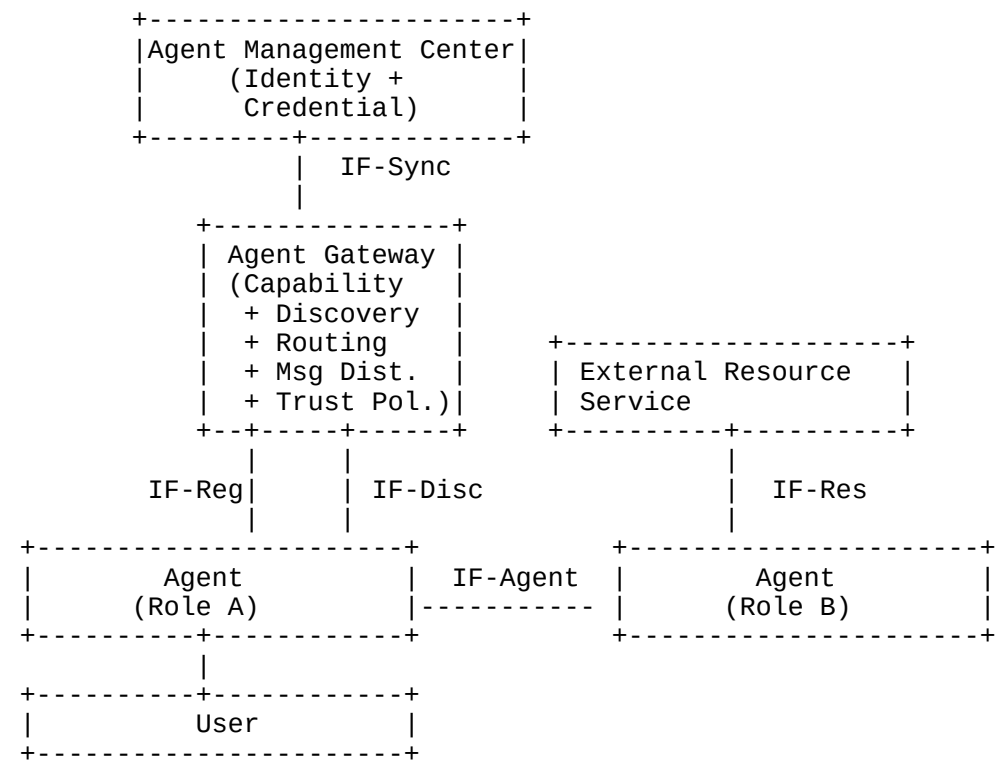


Figure 1 Function Entity Architecture Overview

4.1. Agent

An Agent is a software entity with autonomous decision-making and execution capabilities, capable of perceiving the environment, acquiring contextual information, reasoning, and performing tasks independently or collaboratively with other agents. Each Agent is created by a specific Agent Provider and MUST complete registration with an Agent Management Center before providing services in an IoA deployment.

An Agent is responsible for:

* Maintaining its own identity information (e.g., AIC) and credentials locally.

* Maintaining its capability description (e.g., ACS) and ensuring consistency with its current state.

* Performing authentication and authorization checks for interconnection, including mutual verification of peer agents and validation of presented credentials. - Conducting agent-to-agent interaction, including session establishment, message exchange, and task/context management.

* Accessing external resources when required to fulfill tasks.

* Producing monitoring and logging data for troubleshooting, auditing, and governance purposes.

These are internal agent capabilities described here for informational purposes. They are NOT standardized as separate architectural functional components. In an interaction, an Agent MAY assume different roles depending on the collaboration mode. The DMSC architecture does not constrain the set of possible roles; specific collaboration protocols MAY define role semantics appropriate to their interaction patterns.

For example, in a task-driven collaboration:

* Leader: The Agent that initiates tasks and organizes collaboration.

* Partner: The Agent that accepts tasks and provides services, executing assigned tasks and returning results to the Leader.

A single Agent implementation MAY act in different roles across different interactions. Role assignment is per-interaction, not per-deployment.

4.2. Agent Management Center

The Agent Management Center is the trusted infrastructure service responsible for agent identity lifecycle management and credential issuance. The Agent Authenticationserves as the trust anchor for an IoA deployment.

The Agent Management Center provides the following functions:

* Credential Management: Issuing, renewing, suspending, revoking, and publishing status of Agent Credentials (e.g., X.509 certificates, security tokens) bound to AICs.

* Credential Validation Support: Providing credential validation material (e.g., issuer public keys, certificate revocation lists, OCSP endpoints) to relying parties.

Multiple Agent Management Center MAY exist in an IoA deployment, each managing a subset of agents within its administrative scope. A deployment MAY realize the identity management function and the credential authority function as separate services, provided they maintain consistent identity-to-credential binding.

4.3. Agent Gateway

The Agent Gateway is the infrastructure service that provides interconnection functions for agent collaboration. In the control plane / data plane separation model, the Agent Gateway primarily operates in the control plane.

The Agent Gateway provides the following functions:

* Identity Management: Allocating, updating, and de-registering Agent Identity Codes (AICs); defining and enforcing policies for uniqueness and governance of issued AICs within its administrative scope.

* Capability Directory: Accepting, validating, publishing, versioning, and de-listing Agent Capability Specifications (ACSs); maintaining the capability directory for agents within its administrative scope.

* Discovery and Matching: Receiving discovery queries from agents, matching query constraints against available agent capability descriptions, and returning ranked candidate sets. In-domain discovery SHOULD be supported; cross-domain discovery is OPTIONAL within a configured trust scope.

* Routing and Forwarding: Routing agent interaction requests based on capability matching results or task requirements. The routing function MAY support semantic routing as an extension point for future capabilities such as intent-based routing.

* Message Distribution: Providing message distribution services for group interactions (e.g., publish/subscribe, queue-based delivery) when agents interact in grouping mode.

* Trust Policy: Maintaining trust relationships and federation policies between Agent Autonomous Domains (e.g., trusted peer domain lists, cross-domain discovery policies) to constrain cross-domain discovery and interactions.

The Agent Gateway SHOULD support synchronization with the Agent Management Center to ensure accuracy and timeliness of identity and capability information. Multiple Agent Gateways MAY exist in an IoA deployment, each serving a defined administrative scope.

A deployment MAY co-locate the Agent Gateway with the Agent Management Center. A deployment MAY also realize the message distribution function as a dedicated Message Broker service; in such cases, the Agent Gateway retains management responsibility (control plane) while the Message Broker handles message transport (data plane).

4.4. External Resource Service

The External Resource Service is a device, software component, or network-accessible service that provides specific functions to agents. Examples include APIs, databases, computation services, and other external resources.

The External Resource Service provides the following functions:

* Resource Exposure: Registering or exposing available resources and their invocation interfaces to authorized agents.

* Invocation Handling: Receiving and processing resource invocation requests from agents, executing the requested operations, and returning results.

* Access Control: Enforcing access control policies on resource invocations, including authentication of requesting agents and authorization checks.

An External Resource Service MAY represent a single resource, a resource gateway, or a resource execution environment. Deployments MAY use existing resource access protocols (e.g., MCP [Model Context Protocol], A2T [Agent-to-Tool Protocol]) for agent-to-resource interaction.

4.5. User

A User is a human or organizational entity that initiates tasks, provides authorization and policy input, and consumes results delivered by agents. A User interacts with agents but does not participate directly in agent-to-agent protocols defined by DMSC.

4.6. Data Objects

The following are protocol data objects referenced by the functional entities. They are not functional entities themselves.

Agent Identity Code (AIC): An Agent Identity Code (AIC) is a verifiable, globally unique identifier that represents the identity of an Agent. An AIC is allocated by an Agent Gateway during agent registration. An AIC MAY encode information such as the issuing registry, the Agent Provider, a serial number, and a check code.

Agent Capability Specification (ACS): An Agent Capability Specification (ACS) is a structured description of an agent’s capabilities and service information that can be stored, retrieved, and matched. An ACS MAY use the JSON [RFC8259] [RFC8259] format, typically including: the agent’s AIC, functional capabilities, technical characteristics, service interfaces, and security requirements.

Agent Credential: An Agent Credential is a tamper-resistant data object issued by an Agent Authentication (or its credential authority component), used by an Agent to prove identity attributes and/or authorization to a relying party. Examples include X.509 certificates and security tokens.

Agent Autonomous Domain: An Agent Autonomous Domain is an administrative and governance domain organized and managed by a specific IoA service provider. An Agent Autonomous Domain typically includes one Agent Authentication, one or more Agent Gateways, and the agents registered within its scope.

4.7. Entity Summary

The following table provides a summary of all functional entities and external actors in the simplified DMSC architecture.

#	Entity	Type	Role in Architecture
1	Agent	Core entity	Autonomous task execution and collaboration
2	Agent Management Center	Infrastructure	Identity lifecycle management and credential issuance
3	Agent Gateway	Infrastructure	Capability directory, discovery, routing, message distribution, trust policy
4	External Resource Service	Infrastructure	External resource exposure and invocation handling
5	User	External actor	Task initiation, authorization, and result consumption

Figure 2 A summary of all functional entities

5. Multi-Agent Collaboration Protocol Suite Overview

The Multi-Agent Collaboration Protocol Suite based on Agent Gateway defines a set of coordinated protocols as shown in figure 1 that collectively enable secure agent onboarding, ,distributed capability visibility, semantic request resolution, peer-to-peer semantic interaction, and task-oriented multi-agent orchestration. Rather than operating independently, these protocols are designed to be executed in a tightly coupled manner along the agent lifecycle and collaboration workflows. The Agent Gateway (AG) serves as the anchoring point for control-plane coordination, while semantic interactions are progressively delegated to agents once resolution and authorization are completed.

The protocol suite consists of the following categories:

* Agent Registration Protocol (ARP) and Agent Authentication and Authorization Protocol (AAAP), which jointly establish agent identity, trust, and operational scope.

* Capability Synchronization Protocol (CSP), which maintains distributed visibility of agent capability digest across gateways.

* Semantic Resolution and Routing Protocol (SRRP), which enables semantic request discovery and routing across gateway domains.

* Task-based Invocation Protocol (TIP), which extends semantic routing to multi-agent task decomposition and orchestration.

* Agent-to-Agent Semantic Session Protocol (A2ASP), which completes the collaboration lifecycle by enabling autonomous, peer-to-peer semantic interaction.

Each protocol operates at a specific phase of the collaboration lifecycle and may be invoked independently or in combination with others. The following sections describe how these protocols are integrated into coherent operational flows.

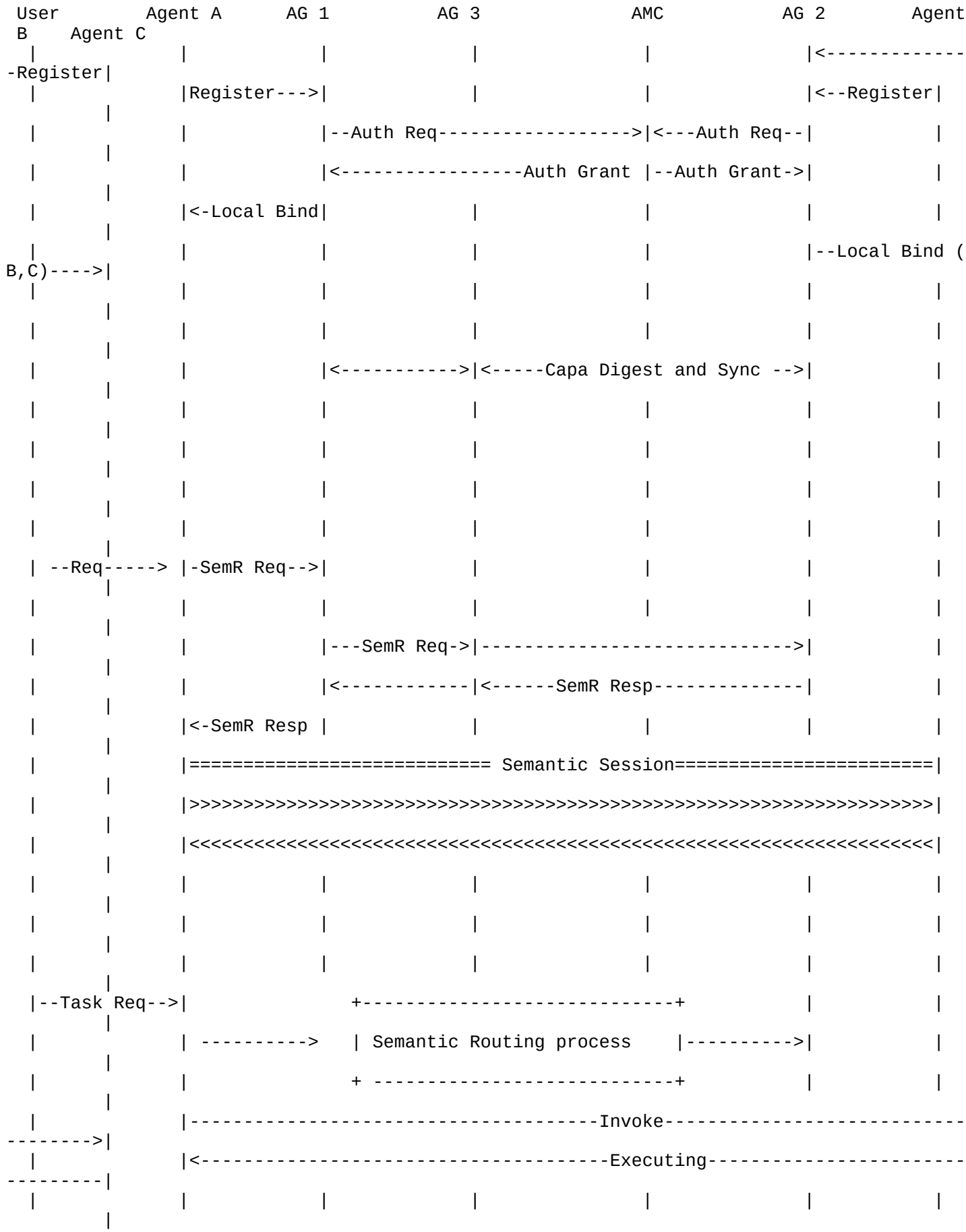

User
Agent A
AG 1
AG 3
AMC
AG 2
Agent B
Agent C
Register
Register
Auth Req
Auth Req
Auth Grant
Auth Grant
Local Bind
Local Bind (B,C)
Capa Digest and Sync
Req
SemR Req
SemR Req
SemR Resp
SemR Resp
Semantic Session
Task Req
Semantic Routing process
Invoke
Executing

Figure 3 The overall sequence diagram of MACP

5.1. Agent Registration and Authorization Process

An agent MUST register with its locally attached Agent Gateway before participating in any collaboration. This process is governed jointly by ARP and AAAP and establishes the agent's identity, trust status, and capability binding. Upon receiving a registration request, the Agent Gateway performs preliminary validation of the agent's identity attributes and initial capability description. The gateway then initiates an authentication and authorization request to the Central Authentication Service, conveying the agent identity, gateway identity, and requested operational scope.

The Central Authentication Service evaluates the request and returns an authorization grant or denial. Upon successful authorization, the Agent Management Center issues an Agent Credential. And the Agent Gateway finalizes the registration by assigning the agent a globally unique Agent Identity Code (AIC) and Capability Identifier(s). The AIC represents the agent's persistent identifier within the architecture. The AIC is allocated by the Agent Gateway only after successful authentication and authorization.

5.2. Capability Digest and Synchronization Process

Each Agent Gateway maintains detailed capability information only for its locally registered agents, and generates a structured Agent Capability Specification (ACS). The ACS is a gateway-generated representation that normalizes and organizes authorized capabilities associated with the assigned AIC. It reflects only those capabilities permitted within the approved operational scope. Gateways do not synchronize full agent capability states with each other. Instead, to support inter-gateway semantic resolution, gateways exchange capability digests using the Capability Synchronization Protocol (CSP). A capability digest is a locally generated, abstract summary of available capabilities, designed solely to indicate what kinds of capabilities exist behind a gateway, rather than how those capabilities are internally implemented or executed by agents.The structure and semantics of capability digests are intentionally decoupled from agent-internal capability representations, allowing gateways to evolve local capability models without impacting inter-gateway interoperability.

CSP distributes these capability digests incrementally. An initial exchange establishes basic inter-gateway visibility, while subsequent updates convey only digest changes, such as newly advertised capabilities, capability updates, or withdrawals. Digest updates are versioned and acknowledged to support consistency and conflict resolution. Through this digest-based mechanism, gateways maintain a scalable and privacy-preserving view of distributed agent capabilities without requiring centralized directories or full capability replication.

5.3. Semantic Resolution and Routing Process

When a user issues a request to an agent (e.g., Agent A), the agent abstracts the request into a semantic request and submits it to its locally attached Agent Gateway (AG1). This interaction is governed by the Semantic Resolution and Routing Protocol (SRRP). Upon receiving the semantic request, AG1 performs semantic parsing and normalization and consults its local capability directory. If no matching capability identifier is found, AG1 forwards the semantic

request to a peer or upstream gateway (e.g., AG3), which repeats the same resolution procedure. If the request remains unresolved, it is further forwarded to another gateway (e.g., AG2).

When a gateway identifies a matching capability in its local directory, it generates a semantic resolution response containing the resolved capability identifier and the corresponding target agent information. This response is propagated hop-by-hop back to the originating gateway and ultimately delivered to Agent A.

Following successful resolution, Agent A and the target agent (e.g., Agent B) directly establish a semantic session. During the lifetime of this session, semantic data is exchanged directly between agents in a peer-to-peer manner, while gateways remain responsible for resolution, authorization scope enforcement, and security policy application during session establishment.

5.4. Task-based Multi-Agent Invocation Process

Task-based collaboration extends semantic resolution to scenarios requiring multiple agents and coordinated execution, as defined by the Task-based Invocation Protocol (TIP). When a user initiates a task request, the request is delivered to Agent A, which performs semantic understanding of the task and decomposes it into one or more sub-tasks along with the required capabilities. If Agent A does not possess task decomposition capabilities, its attached Agent Gateway MAY act as a proxy to analyze and decompose the task on behalf of the agent.

For each sub-task, Agent A submits a semantic request to its local gateway, triggering the same multi-hop semantic resolution process defined by SRRP. Unlike pure point-to-point semantic communication, gateways additionally apply task-level constraints, policy considerations, and capability selection logic to identify suitable target agents.

The resolved results are returned to Agent A, which then directly invokes the selected agents and establishes the necessary semantic sessions for execution. Through this mechanism, multiple agents can be dynamically selected and coordinated to collaboratively execute complex tasks, while maintaining consistent authorization and security enforcement through gateway-mediated control-plane functions.

5.5. Agent-to-Agent Semantic Session Process

The Agent-to-Agent Semantic Session Protocol (A2ASP) defines how two authorized agents establish, maintain, and terminate a semantic session for peer-to-peer semantic communication. A2ASP operates entirely between agents in the data plane. Distributed nodes participate only during session authorization and parameter provisioning phases and are not required in the semantic data path once the session is established.

A2ASP consists of four phases:

* Session Initiation: A semantic session is initiated when an agent, having obtained a valid resolution result, sends a session initiation message to the target agent. The message includes the Agent Identifiers, a proposed Session Identifier, the resolved Capability Identifier, and a verifiable authorization artifact derived from prior control-plane procedures. Upon transmission, the initiating agent awaits confirmation from the target agent.

* Mutual Verification and Session Establishment: Upon receiving the initiation request, the target agent verifies the initiator's identity, validates the authorization context, and confirms that the referenced capability is locally bound and permitted by policy. If validation succeeds, the target agent returns a session acceptance message, and both agents transition to an established state. If validation fails, the request is rejected and no session is created.

* Semantic Interaction: Once established, semantic data is exchanged directly between the agents and is explicitly bound to the Session Identifier and Capability Identifier. Each agent enforces the authorized capability scope locally. The protocol does not mandate a specific transport, but confidentiality and integrity protection are expected to be ensured by the underlying secure channel.

* Session Update: If session parameters require adjustment, either agent may request an update. Any modification must remain within the originally authorized scope unless additional authorization is obtained. Updated parameters take effect upon mutual agreement.

* Session Termination: A session is terminated when either agent sends a termination message or when authorization expires. Upon termination, both agents release associated resources and reject further messages referencing the Session Identifier.

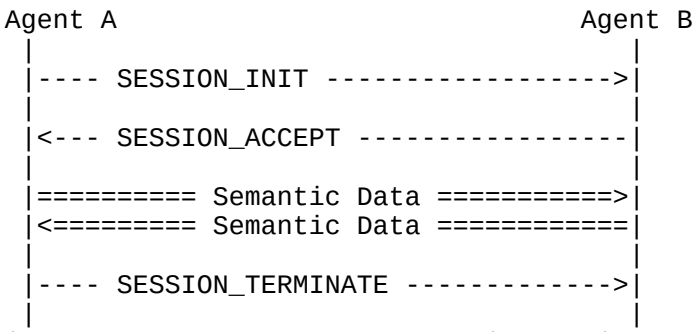
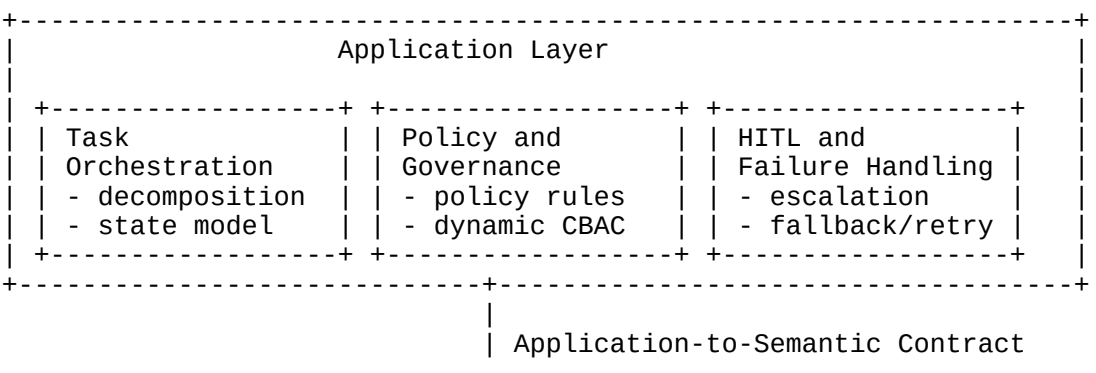


Figure 4 Agent-to-Agent Semantic Session Process

6. Layered Responsibility and Protocol Positioning

From the perspective of an end agent, the architecture can be understood as a layered responsibility model in which different functional concerns are clearly separated, as shown in figure 4. This document does not attempt to standardize all layers; rather, it focuses on a specific interaction and coordination layer within the overall system.

The protocols specified in this document primarily belong to the Interaction Layer. This layer governs how agents register, discover each other, resolve capabilities, establish sessions, and execute collaborative tasks. Specifically, the Agent Registration Protocol (ARP) and Agent Authentication and Authorization Protocol (AAAP) address identity and trust establishment. The Capability Digest and Synchronization Protocol (CDSP) and Semantic Resolution and Routing Protocol (SRRP) support capability visibility and intent resolution. The Task-based Invocation Protocol (TIP) manages coordinated multi-agent invocation. The Agent-to-Agent Semantic Session Protocol (A2ASP) enables direct semantic interaction once authorization is established. These protocols collectively form the standardized coordination and interaction control framework between agents.



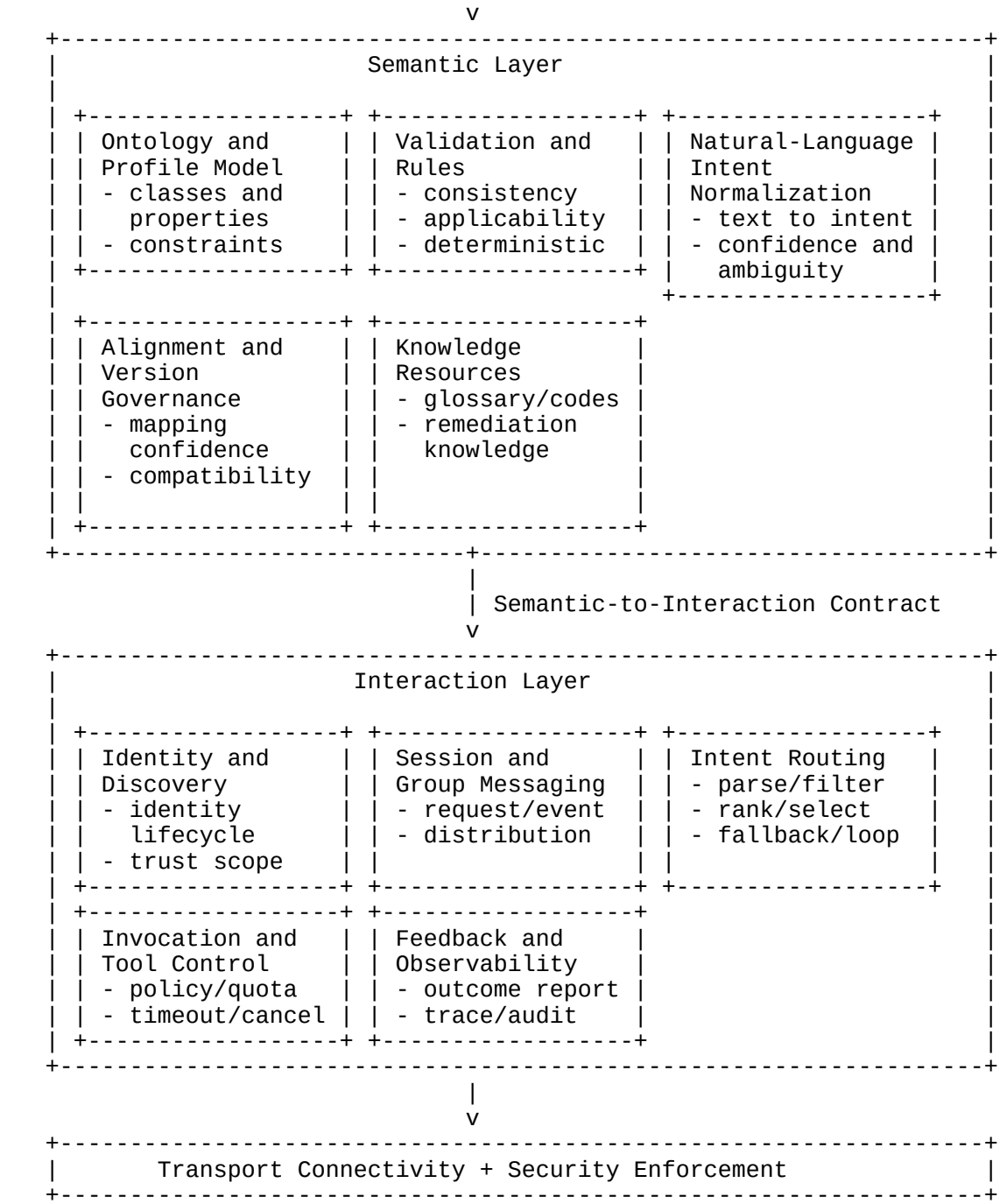


Figure 5 Unified Layered Architecture and Modules

7. Deployment Example: Fixed Network with Agent Gateway Realization

This section provides a deployment example illustrating how the proposed architecture can be realized in a fixed network environment. The purpose of this example is not to constrain implementation choices, but to demonstrate how distributed nodes defined in the architecture may be instantiated using existing or enhanced network elements.

In a fixed broadband network, distributed nodes can be realized as enhanced Agent Gateway functions deployed at aggregation layers, metro nodes, or service edge points. These nodes are positioned logically between access networks and service domains, enabling them to perform registration, authorization mediation, capability abstraction, and semantic routing functions without requiring changes to end-user access infrastructure.

In such deployments as shown in figure 3:

* Agent registration is performed via the local AG.

* Capability digest exchange occurs between AGs.

* Semantic resolution is handled hop-by-hop across AGs.

* Authorization policies are enforced at the gateway boundary.

* Agents establish peer-to-peer semantic sessions after gateway-mediated coordination.

This example demonstrates that the architecture does not mandate a centralized control entity. Instead, distributed nodes represent logical coordination functions that can be embedded into gateway platforms within fixed networks. Other deployment environments, such as mobile networks or enterprise networks, may realize the same logical functions using different network elements while preserving protocol behavior and architectural principles.

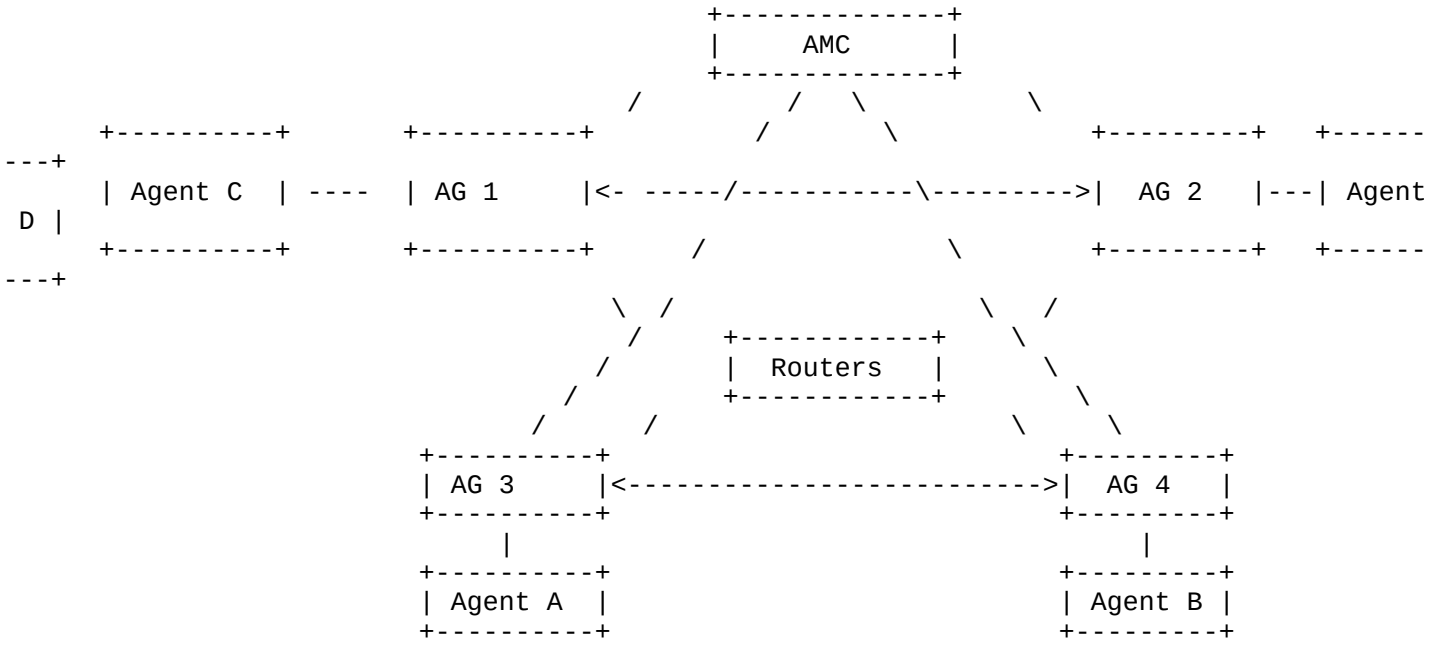


Figure 6 Deployment Example: Fixed Network with Agent Gateway Realization

8. Deployment Example: Mobile Network

TBD

9. IANA Considerations

TBD

10. Acknowledgement

TBD

11. 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>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

Authors' Addresses

Xueting Li
China Telecom
Beiqijia Town, Changping District
Beijing
Beijing, 102209
China
Email: lixt2@foxmail.com

Jun Liu
Beijing University of Posts and Telecommunications
10 Xitucheng Road, Haidian District
Beijing
100876
China
Email: liujun@bupt.edu.cn

Chenguang Du
Zhongguancun Laboratory
Beijing
100094
China
Email: ducg@zgclab.edu.cn

Lianhua Zhang
AsiaInfo Technologies (China) Inc
Beijing
100000
China
Email: zhanglh2@asiainfo.com