DMSC Working Group
Internet-Draft
Intended status: Standards Track
Expires: 1 November 2026

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

30 April 2026

Multi-agent Collaboration Protocol Suites Architecture
draft-li-dmsc-macp-04

Abstract

This document defines a protocol suite and architectural framework for secure and scalable multi-agent collaboration. The proposed Multi-Agent Collaboration Protocol (MACP) enables trusted agent onboarding, capability-based discovery, distributed capability synchronization, and secure interaction among agents and external resources. The architecture introduces key entities such as the Agent Management Center (AMC), Agent Gateway (AGW), Agents, and External Resource Services (ERS), along with a set of protocols that collectively support dynamic, capability-driven collaboration across administrative domains.

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 1 November 2026.

Table of Contents

1. Introduction

The rapid evolution of large-scale multi-agent systems introduces new requirements for coordination, security, and service discovery across distributed environments. Agents are no longer confined to isolated execution contexts, but increasingly operate across administrative domains, network boundaries, and heterogeneous infrastructures. However, existing mechanisms for service interaction and discovery exhibit several limitations when applied to multi-agent

collaboration:

* Lack of unified trust establishment: Current agent interaction models often assume pre-established trust or rely on application-layer authentication, without a network-level mechanism to ensure that participating agents are authenticated, authorized, and accountable across domains.

* Insufficient capability abstraction and discoverability: Traditional service discovery mechanisms (e.g., DNS-based or registry-based approaches) focus on endpoint resolution rather than capability-oriented matching, making them unsuitable for dynamic agent collaboration where tasks are fulfilled based on functional capabilities rather than fixed service locations.

* Limited visibility across distributed environments: Existing systems lack a mechanism to construct a distributed, up-to-date view of available agent capabilities, especially when agents are registered under different control points or administrative domains.

* Inefficient or ad hoc discovery mechanisms: Without coordinated discovery strategies, agent systems rely on broadcast-like or centralized queries, leading to scalability challenges and increased latency in locating suitable collaborators.

* Fragmented protocol landscape: While protocols such as A2A, MCP, or other interaction mechanisms exist, they operate in isolation and do not provide an integrated framework for authentication, registration, discovery, and coordination.

These limitations become more critical as multi-agent systems scale, where dynamic task composition, cross-domain collaboration, and secure interaction are fundamental requirements. To address these challenges, this document proposes the Multi-Agent Collaboration Protocol (MACP), a protocol suite and architectural framework that:

* Establishes a trusted onboarding mechanism via a centralized authentication and authorization entity

* Introduces capability-based abstraction and identification for agents

* Enables distributed capability synchronization across control points

* Supports both proactive and reactive discovery mechanisms

* Integrates existing interaction protocols into a cohesive collaboration framework

By shifting from endpoint-centric interaction to capability-driven collaboration, MACP enables scalable, secure, and flexible multi-agent systems that can operate effectively across heterogeneous and distributed environments.

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 automated intelligent entity capable of e.g interacting with its environment, acquiring contextual information, reasoning, self-learning, decision-making, executing tasks (autonomously or in collaboration with other Al Agents) to achieve a specific goal.

* Agent Gateway: The Agent Gateway is a functional entity that serves as the infrastructure for enabling interconnection and collaboration among agents. While its core role remains consistent, it is inherently flexible in deployment and can be realized in various forms— ranging from a network service to a dedicated gateway— depending on the architectural and operational requirements of different network environments.

* 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 Registration Protocol (ARP): The ARP governs how an agent formally registers with a locally attached agent gateway.

* Agent Authentication and Authorization Protocol (AAAP): The AAAP defines how authentication and authorization decisions are requested and enforced.

* Capability Directory Synchronization Protocol (CDSP): The CDSP synchronizes abstracted agent capability digests across agent gateways.

* Agent Discovery Protocol (ADP) : The ADP enables AGWs to locate agents that provide required capabilities, supporting both local and distributed discovery.

4. Multi-agent Collaboration Protocol Architecture

The MACP architecture consists of the following key entities:, as shown in figure 1. Each functional entity represents a logical role in the IoA architecture, implementations MAY combine multiple entities into a single product.

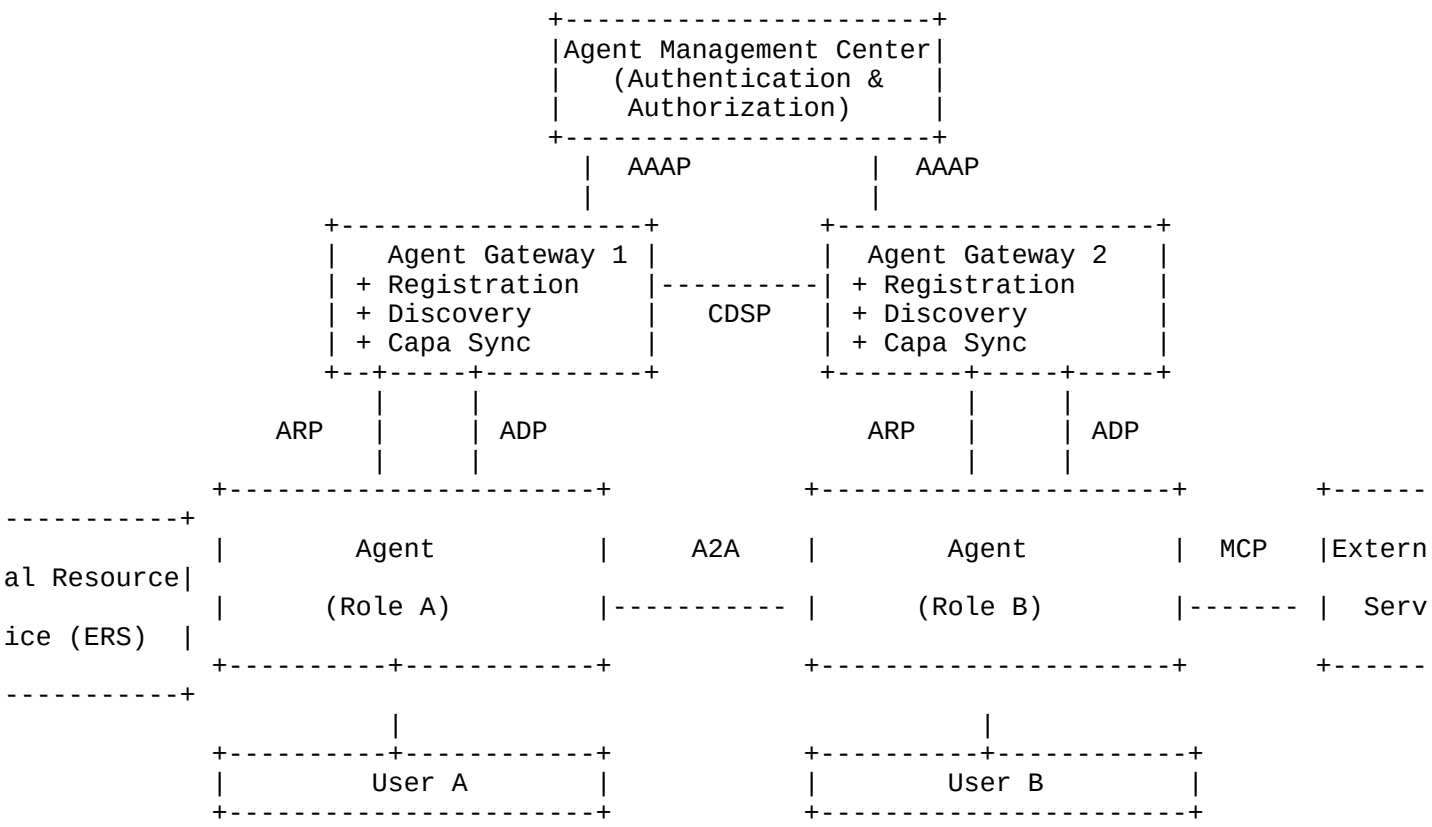


Figure 1 MACP Architecture Overview

4.1. Agent Management Center

The Agent Management Center is the trusted infrastructure service responsible for agent identity lifecycle management and credential issuance. The AMC provides centralized authentication and authorization services. It ensures that only legitimate and trusted agents are allowed to join the system. Specifically, the AMC:

* Authenticates agent identity

* Determines authorization scope and execution permissions

* Issues authorization credentials (e.g., certificates or tokens)

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.

4.2. Agent Gateway

The Agent Gateway is a functional entity that serves as the infrastructure for enabling interconnection and collaboration among agents. While its core role remains consistent, it is inherently flexible in deployment and can be realized in various forms—ranging from a network service to a dedicated gateway—depending on the architectural and operational requirements of different network environments.

The Agent Gateway provides the following functions:

* Agent Registration: Maintains agent identity and capability information.

* Agent Capability Directory Management: Stores and organizes registered agent capabilities.

* Agent Capability Discovery: Supports both proactive and reactive agent discovery.

* Agent Capability Directory Synchronization: Exchanges agent capability digests with peer gateways.

* Agent Group Communication Support: Enables multi-agent coordination.

Each AGW maintains a local capability view and participates in forming a distributed capability knowledge plane. More specific requirements are specified in [draft-liu-dmsc-gw-requirements][GW-REQ].

4.3. Agent

The Agent is an automated intelligent entity capable of e.g interacting with its environment, acquiring contextual information, reasoning, self-learning, decision-making, executing tasks (autonomously or in collaboration with other Al Agents) to achieve a specific goal.

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 [draft-yang-dmsc-ioa-task-protocol] [draft-yang-dmsc-ioa-task-protocol]:

* 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.

4.4. External Resource Service (ERS)

The External Resource Service (ERS) represents external systems such as APIs, databases, or compute services that agents may invoke. ERS is conceptually external to the agent collaboration system and is accessed via existing protocols.

ERS may include both domain-specific services and shared infrastructure services. In particular, certain ERS instances MAY correspond to widely deployed Internet-scale infrastructure (e.g., naming, data access, or knowledge retrieval systems), which provide common capabilities that are not specific to agent collaboration but are essential for its operation. In this sense, ERS can be viewed as leveraging existing or future shared Internet service infrastructure, rather than replicating such functionality within the agent system itself.

A key design principle is the separation of responsibilities between the agent collaboration system and ERS. The agent system is responsible for:

* Agent identity, trust establishment, and authorization

* Agent capabilities registration, abstraction, and discovery

* Coordination and interaction among agents

In contrast, ERS is responsible for:

* Providing external data, computation, or domain-specific functionality

* Supporting tasks that require capabilities beyond the agent system itself

Agents interact with ERS when executing tasks that require external resources, while core collaboration functions— such as discovery, routing, and coordination— remain within the agent system. The MACP architecture intentionally avoids redefining general-purpose Internet services (e.g., naming or data retrieval), and instead focuses on enabling agents to discover, select, and utilize such services in a coordinated manner.

4.5. 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.

* 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 AMC (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.

4.6. 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,agent discovery, synchronization
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

MACP defines a set of protocols to enable interaction among entities.

5.1. Agent Registration Protocol (ARP)

The Agent Registration Protocol (ARP) is used between an Agent and an AGW to onboard the agent into the system [draft-sz-dmsc-iaip]. ARP is responsible for:

* Agent identity declaration

* Capability registration (capability-based onboarding)

* Lifecycle management (registration, update, deregistration)

Operational flow:

1. The agent sends a registration request to the AGW, including: agent identity information and capability vector.

2. The AGW triggers AAAP to validate the agent (as described in Section 5.2).

3. Upon successful authorization, the AGW assigns a globally unique Agent ID (AIC).

4. The master AGW store the agent’s capabities information in the local capability directory.

5. The agent is considered available once ARP and AAAP are completed successfully.

5.2. Agent Authentication and Authorization Protocol (AAAP)

The Agent Authentication and Authorization Protocol (AAAP) is used between the Agent Gateway (AGW) and the Agent Management Center (AMC) to establish trust for agents attempting to join the system.

AAAP is responsible for:

* Verifying the identity of an agent (via AGW as a proxy)

* Determining the authorization scope and permitted operations

* Issuing authorization credentials (e.g., tokens or certificates)

Operational flow:

1. The AGW receives a registration request from an agent.

2. The AGW initiates an AAAP request to the AMC, carrying agent identity information.

3. The AMC performs authentication and authorization checks.

4. Upon success, the AMC returns an authorization credential and policy constraints.

5. The AGW enforces the received authorization decision.

6. The AMC acts as the trust anchor of the system, while the AGW acts as the policy enforcement point (PEP).

Note that although AAAP establishes the initial trust relationship between an agent and the system (i.e., onboarding trust), subsequent interactions between agents may require additional, context-specific authentication and authorization. Such interaction-level mechanisms are out of scope for AAAP and MAY leverage existing frameworks (e.g., OAuth-based token exchange or similar delegation mechanisms) to support secure, fine-grained access control between agents.

An agent MUST successfully complete AAAP before participating in discovery, invocation, or collaboration. However, successful onboarding via AAAP does not eliminate the need for authentication and authorization during runtime interactions between agents.

5.3. Capability Directory Synchronization Protocol (CDSP)

The Capability Directory Synchronization Protocol (CDSP) is used between Agent Gateways (AGWs) to synchronize capability directory information and construct a distributed view of agent capabilities across the network. More detailed treatment of the gateway capability directory in [draft-zhang-dmsc-gateway-directory-sync] [draft-zhang-dmsc-gateway-directory-sync].

CDSP is designed to:

* Enable distributed visibility of agent capabilities across multiple AGWs

* Avoid centralized bottlenecks in capability management

* Preserve privacy and scalability through abstraction

* Support incremental and policy-controlled synchronization.

Each AGW maintains a local capability directory, which is populated through agent registration and updated over time. CDSP enables AGWs to exchange selected portions of these directories so that capability information is not confined to a single gateway but becomes visible, in an abstracted form, across multiple administrative or network domains. To ensure scalability and protect sensitive information, CDSP does not transfer complete capability descriptions. Instead, AGWs exchange capability directory entries in a summarized form. Each entry represents a capability exposed by an agent and is associated its corresponding capability vector. The exchanged information MAY include semantic descriptions or structured representations of the capability, along with limited metadata such as version or category. Detailed implementation-specific information and sensitive attributes MUST NOT be propagated through CDSP.

CDSP supports different synchronization scopes depending on deployment requirements. During initial establishment between peer AGWs or recovery scenarios, a gateway MAY perform a full synchronization of its capability directory. In steady-state operation, synchronization is typically incremental or selective, where only updated or policy-permitted entries are exchanged. The selection of entries MAY be governed by administrative policies, trust relationships, or capability classification. Synchronization

can be triggered in multiple ways. An AGW MAY initiate periodic synchronization to maintain freshness of the distributed view. It MAY also perform event-driven updates when local changes occur, such as agent registration, deregistration, or capability updates. Additionally, synchronization MAY be requested on demand by peer gateways when needed for discovery or coordination purposes.

Given the distributed nature of CDSP, strict consistency across all AGWs is not required. Instead, the system operates under an eventual consistency model, where capability directory views converge over time. To support this, capability entries SHOULD include versioning or timestamp information, allowing AGWs to reconcile updates and prefer the most recent information. Conflict resolution policies MAY be applied when inconsistencies arise. All CDSP exchanges MUST be authenticated and integrity-protected.

5.4. Agent Discovery Protocol (ADP)

The Agent Discovery Protocol (ADP) enables AGWs to locate agents that provide required capabilities, supporting both local and distributed discovery.

ADP is designed to:

* Enable capability-based matching instead of endpoint-based lookup

* Support both low-latency local discovery and network-wide search

* Adapt to dynamic and partially known environments

When an agent or user request is received, the AGW evaluates the requested capability against its local capability directory. This directory includes both locally registered agents and capability summaries learned from other AGWs via CDSP. If a matching capability is found locally, the AGW can directly identify candidate agents and proceed with interaction setup. If the required capability cannot be satisfied using local information, the AGW initiates a distributed discovery process. In this case, the request is propagated to other AGWs in the network.

ADP supports both proactive and reactive discovery behaviors within a unified framework. In proactive scenarios, AGWs maintain an updated distributed capability view through CDSP, thereby enabling most discovery requests to be resolved locally with minimal latency. In reactive scenarios, when local knowledge is insufficient, AGWs dynamically query the network to identify suitable agents. Upon receiving a request, an AGW evaluates it against its capability directory and returns matching results if available. Responses

SHOULD include the relevant Capability vectors, the corresponding Agent ID, and abstracted routing or reachability information sufficient for subsequent interaction.

Multiple candidate agents MAY be returned for a single request. The requesting AGW is responsible for selecting an appropriate agent based on local policies, optimization criteria, or contextual requirements.

5.5. Agent to Agent Communication overall Protocol Suites

From the perspective of an Agent, Agent-to-Agent interaction is realized through a layered protocol suite, where each layer is responsible for a distinct aspect of task execution, semantic interpretation, runtime interaction, and protected connectivity, as shown in the figure below. These layers collectively ensure that agent collaboration is not only functionally correct but also semantically consistent, policy-aware, and secure.

At the Application Layer, the agent is responsible for structuring and governing the execution of tasks prior to interaction with other agents. This includes task orchestration capabilities, where complex objectives are decomposed into smaller executable units and managed through an internal state model that tracks execution progress. In parallel, the agent applies policy and governance logic to determine how tasks should be executed or delegated. Such policies MAY include rule-based constraints and dynamic capability-based access control (CBAC), which influence decision-making based on context and authorization scope. The Application Layer also incorporates mechanisms for robustness and operational control, including human-in-the-loop (HITL) intervention and failure handling strategies such as escalation, retry, or fallback. The output of this layer is a structured and policy-compliant task intent that is ready for semantic processing.

The Semantic Layer provides the mechanisms required to transform application-level intent into a representation that can be consistently interpreted across agents. At this layer, agents rely on shared or compatible ontology and profile models to express capabilities in terms of classes, properties, and constraints[draft-zhang-dmsc-ioa-semantic-interaction] . Before interaction, the agent validates the generated semantic representation to ensure consistency, applicability, and determinism, thereby avoiding ambiguity during execution. For inputs expressed in natural language [draft-verma-dmsc-nlip-notes], the agent performs intent normalization [draft-sz-dmsc-iaip], translating unstructured descriptions into structured semantic forms while accounting for confidence levels and potential ambiguity. To ensure

interoperability across different domains or versions, the Semantic Layer also supports alignment and version governance, enabling mapping between heterogeneous schemas and maintaining compatibility. In addition, agents MAY utilize supporting knowledge resources, such as standardized glossaries or remediation knowledge, to improve interpretation and resolve inconsistencies. The output of this layer is a normalized and validated semantic intent that can be directly mapped to interaction protocols.

The Interaction Layer defines how agents execute collaborative interaction at runtime once task intent has been semantically prepared. This layer includes task invocation, session handling, message exchange, artifact exchange, and interaction-state progression among peer agents. Protocols such as the Task Invocation Protocol (TIP) [draft-yang-dmsc-ioa-task-protocol] [draft-yang-dmsc-ioa-task-protocol] and the Agent-to-Agent Protocol (A2A) can operate at this layer. These protocols carry task inputs, outputs, updates, and related metadata, while relying on the Semantic Layer for consistent interpretation of capabilities, intents, tasks, and context. In this sense, runtime interaction protocols and the Semantic Layer are complementary: the former provides interaction mechanics, while the latter provides interoperable meaning.

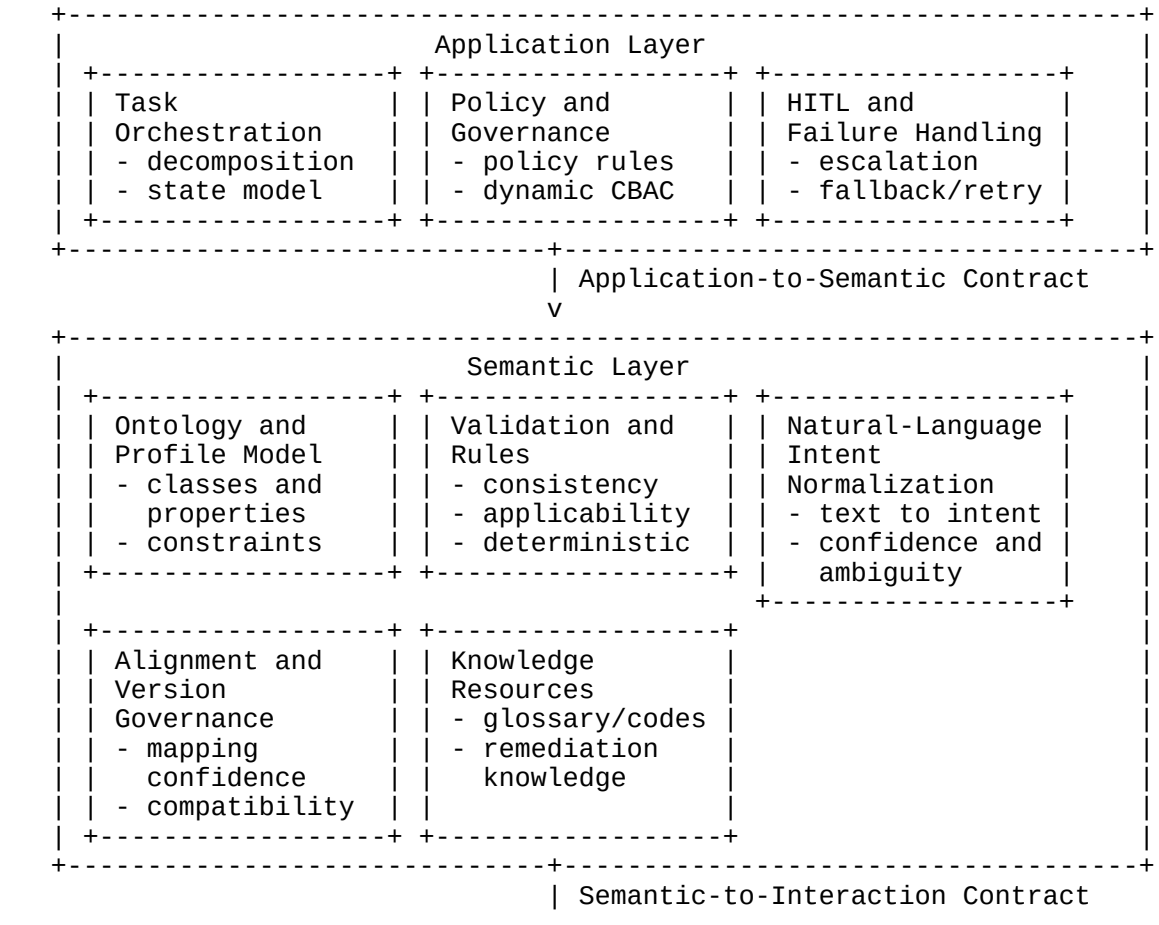
A2A does not by itself define a full semantic interoperability layer. In particular, it does not standardize ontology/profile models, alignment procedures, semantic validation rules, or semantic renegotiation behavior. However, A2A does provide useful mechanisms that can carry or advertise semantic information, including Agent Cards, task and message metadata, artifacts, and extension declarations. The Semantic Layer complements A2A by defining the interoperable semantic structures and rules needed for consistent interpretation of capabilities, intents, tasks, and context across agents and domains. Accordingly, A2A provides runtime interaction mechanics, while the Semantic Layer provides interoperable meaning for the content carried within those mechanics.

Beneath these layers, Transport Connectivity + Security Enforcement provides the protected communication substrate required for agent interaction. This includes transport reachability, channel protection, and related security enforcement functions needed to support reliable and secure protocol execution.

In addition to the vertical interaction stack, gateway-facing control and support functions are required to make distributed collaboration operational. These functions are not themselves part of the application/semantic/interaction execution chain, but provide the surrounding control-plane environment in which that chain operates. Such functions include registration and identity maintenance through

the Agent Registration Protocol (ARP), authorization context through the Authentication and Authorization Protocol (AAAP), capability visibility synchronization through the Capability Directory Synchronization Protocol (CDSP), and candidate lookup through the Agent Discovery Protocol (ADP). Together, these gateway control-plane and support functions enable secure, dynamic, and capability-driven interaction among agents.

These layers and support functions operate in a coordinated manner across well-defined boundaries. The Application Layer defines what needs to be done, the Semantic Layer defines how it is described and understood, the Interaction Layer defines how it is executed across agents, and the transport/security substrate ensures protected connectivity. Gateway control-plane and support functions maintain the registration, authorization, discovery, and capability-visibility conditions needed for those runtime interactions to occur.



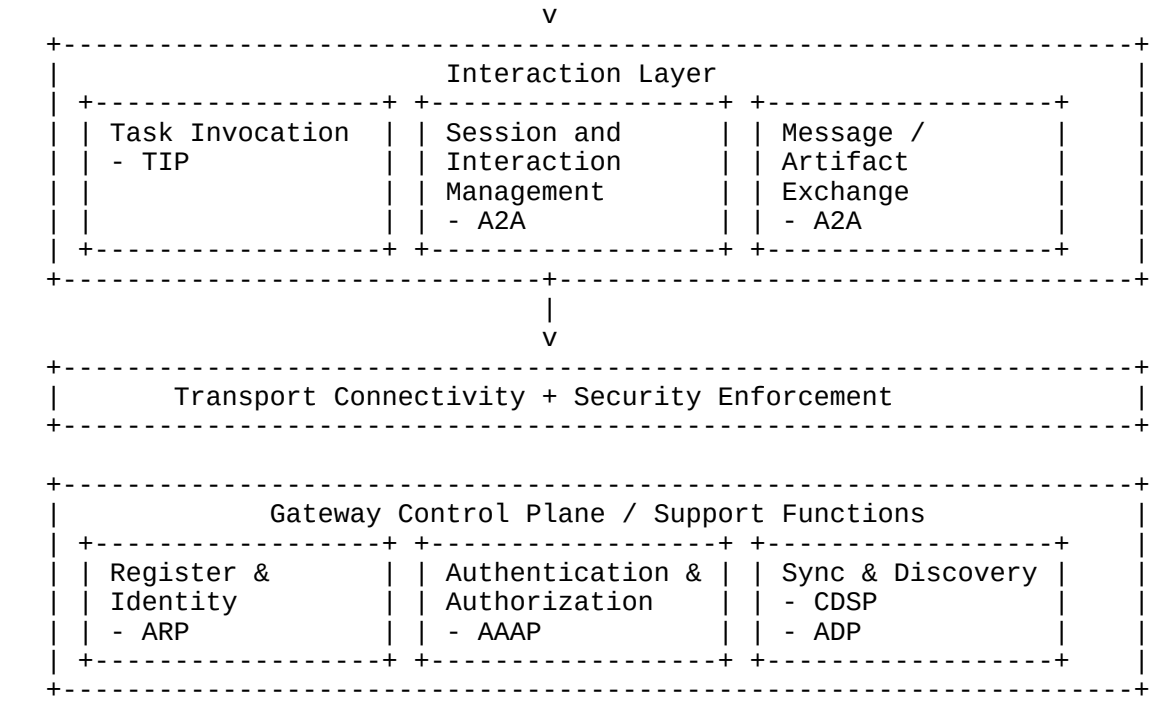


Figure 3 Agent to Agent Communication overall Protocol Suites

5.6. Agent to External Resource Service Protocol

Interaction between agents and external resources (ERS) is supported via existing protocols such as:

* Model Context Protocol (MCP)

* Agent-to-Tool (A2T)

MACP does not redefine these protocols but enables their integration within the architecture.

6. Capability Model

Capabilities are the core abstraction in MACP.

* Each capability is associated with a Capability vector.

* Capabilities are registered, indexed, and discovered via AGWs.

* Capability information is abstracted during synchronization to protect sensitive details.

* Capability descriptions MAY be semantic or structured depending on use case.

This abstraction enables flexible and scalable service composition.

7. IANA Considerations

TBD

8. Acknowledgement

TBD

9. Normative References

[draft-sz-dmsc-iaip]
S, S., "Intent-based Agent Interconnection Protocol at Agent Gateway. draft-sz-dmsc-iaip. <https://datatracker.ietf.org/doc/draft-sz-dmsc-iaip/>", 9 February 2026.

[draft-verma-dmsc-nlip-notes]
V, D., "Use of Natural Language for Agent Communication. draft-verma-dmsc-nlip-notes. <https://datatracker.ietf.org/doc/draft-verma-dmsc-nlip-notes/>", 11 February 2026.

[draft-yang-dmsc-ioa-task-protocol]
Y, C., "Internet of Agents Task Protocol (IoA Task Protocol) for Heterogeneous Agent Collaboration. draft-yang-dmsc-ioa-task-protocol. <https://datatracker.ietf.org/doc/draft-yang-dmsc-ioa-task-protocol/>", 14 January 2026.

[draft-zhang-dmsc-gateway-directory-sync]
Z, L., "Gateway Capability Directory and Synchronization for Internet of Agents. draft-zhang-dmsc-gateway-directory-sync. <https://datatracker.ietf.org/doc/draft-zhang-dmsc-gateway-directory-sync/>", 28 April 2026.

[draft-zhang-dmsc-ioa-semantic-interaction]
Z, L., "Ontology-based Semantic Interaction for Internet of Agents. draft-zhang-dmsc-ioa-semantic-interaction. <https://datatracker.ietf.org/doc/draft-zhang-dmsc-ioa-semantic-interaction/>", 4 February 2026.

[GW-REQ] L, B., "Gateway Requirements for Dynamic Multi-agents Secured Collaboration. draft-liu-dmsc-gw-requirements. <https://datatracker.ietf.org/doc/draft-liu-dmsc-gw-requirements/>", 16 January 2026.

[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

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