DMSC Working Group J. Liu
Internet-Draft K. Yu
Intended status: Informational K. Li
Expires: 25 July 2026 K. Chen
Beijing University of Posts and Telecommunications
21 January 2026

Agent Collaboration Protocols Architecture for Internet of Agents
draft-liu-dmsc-acps-arc-01

Abstract

Internet of Agent (IoA) aims to facilitate interconnection and collaboration among heterogeneous agents to address complex tasks and support various applications.

This IETF draft proposes the Agent Collaboration Protocols (ACPs) architecture, which outlines the key components and functionalities required for agent interoperability. ACPs cover all stages of agents in the network, from their access to collaboration, including: Agent Trusted Registration (ATR), Agent Identity Authentication (AIA), Agent Discovery (ADP), Agent Interaction (AIP), Tool Invocation (TIP), and Agent Monitoring (AMP). The long-term vision of ACPs is to support the future large-scale interconnected agents and construct the key infrastructure for IoA.

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 25 July 2026.

Copyright Notice

Table of Contents

1. Introduction

With the rapid development of artificial intelligence (AI), particularly large language model (LLM) technology, the number of AI agents has grown dramatically. With the capability of autonomous perception, decision-making, and execution, agents' applications are becoming increasingly widespread.

To overcome the limitations of single-agent systems, and to break free from the constraints of proprietary multi-agent frameworks developed by various vendors, the Internet of Agents (IoA) has emerged. IoA aims to enable seamless connectivity and efficient collaboration among agents, through standardized communication protocols and interfaces.

This draft proposes the Agent Collaboration Protocols (ACPs) [ACPs-Github], which is a standardized protocol suite for IoA to enable wide-area connectivity, cross-domain interoperability, and secure collaboration among heterogeneous agents. The main characteristics of ACPs are as follows:

* Multi-centralized architecture, which consists of multiple autonomous domains, each domain containing its own management nodes (such as registration, authentication, and discovery servers), to support efficient, reliable, and manageable large-scale agent interconnection scenarios.

* Standardized communication mechanisms with peer to peer and grouping mode, allowing agents to self-organize and negotiate autonomously, to facilitate rapid and accurate information exchange, as well as efficient task Collaboration.

* Robust registration and authentication mechanisms, to ensure the trusted access of agents and prevent unauthorized access and data breaches.

* Reliable registration and management of agent capability, along with intra-domain and cross-domain discovery based on capability matching, to support universal and efficient discovery of collaborative agents.

* Real-time monitoring of agent status and behavior, to support more complex application requirements such as agent auditing and transactions.

ACPs cover all stages of agents in the IoA, from their access to collaboration, to construct the key infrastructure for agent communication, task collaboration and resource allocation.

2. Terminology

Agent：An agent is a software or hardware entity with autonomous decision-making and execution capabilities, capable of perceiving the environment, acquiring contextual information, reasoning, and learning. Agent can independently or collaboratively with other agents to perform tasks. Each agent is created by a specific agent provider and needs to complete processes such as registration and authentication before providing services to obtain a legitimate identity and relevant credentials.

Agent Identity Code (AIC): AIC is a certifiable, globally unique identity that represents the identity of an agent. AIC MAY contain the following infomation: the registration service center, the agent provider, the serial number of agent entity and instance, and the check code.

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

Agent Registration Server (ARS): ARS is a service entity recognized by consensus, responsible for the registration of agents, the distribution of AICs, and the maintenance of the agent’s availability status.

Certificate Authority Server (CAS): CAS is a service entity that is responsible for verifying the detailed information of agents and issuing, managing agent certificates, establishing communication identity trust.

Agent Discovery Server (ADS): ADS is a service entity responsible for discovering available collaborative intelligent agents for specific task requirement.

Agent Autonomous Domain: Agent Autonomous Domain is a collaborative network domain organized and managed by specific IoA service provider. It may include multiple different entities, such as ARS, CAS, ADS, and agents.

3. ACPs Architecture

Agent Collaboration Protocols (ACPs) is a standardized protocol suite for IoA, which consists of the following protocols built on HTTPS:

(1) Agent Trusted Registration (ATR) protocol: Defines the requirements and process for agents to register with a registration service provider.

(2) Agent Identity Authentication (AIA) protocol: Defines the mechanisms for agents to obtain digital identity credentials and perform authentication during communication.

(3) Agent Discovery Protocol (ADP): Defines the mechanisms for agents to discover potential collaboration partners in an open environment, including query formats, matching processes, and result feedback.

(4) Agent Interaction Protocol (AIP): Defines the communication protocol for agents to establish sessions and exchange messages, serving as the core communication layer of ACPs.

(5) Tool Invocation Protocol (TIP): Provides a standard way for agents to invoke external tools or services.

(6) Agent Monitoring Protocol (AMP): Provides a framework for monitoring and measuring agent behavior and performance.

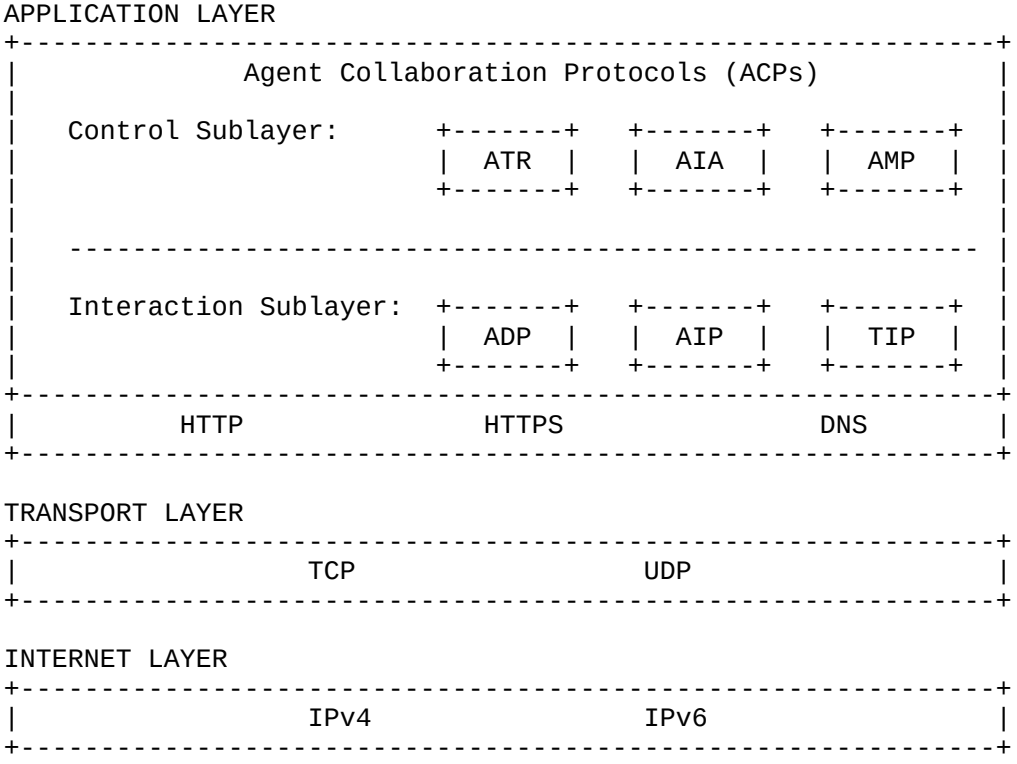
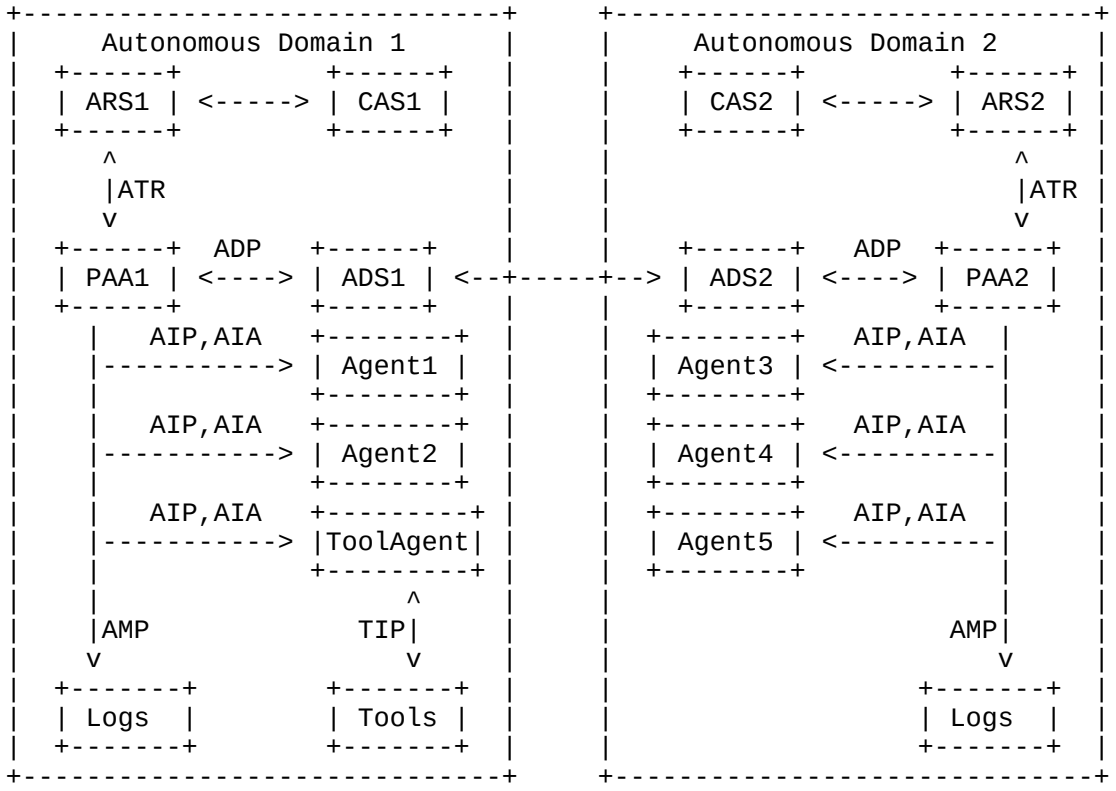


Figure 1: ACPs Protocol Stack

The ACPs-based Internet of Agents consists of multiple autonomous domains, each domain containing registration, authentication, and discovery servers, as well as agents. The overall architecture is as follows.



Legend:
- ARS = Agent Registration Server
- CAS = Certificate Authority Server
- ADS = Agent Discovery Server
- PAA = Personal Assistant Agent
- ATR = Agent Trusted Registration
- AIA = Agent Identity Authentication
- ADP = Agent Discovery Protocol
- AIP = Agent Interaction Protocol
- TIP = Tool Invocation Protocol
- AMP = Agent Monitoring Protocol

Figure 2: ACPs Architecture

The ACPs-based IoA operates based on the following fundamental rules:

(1) All agents are required to follow the ATR to register with the Trusted Agent Registry Server.

(2) An agent must comply with AIA verification to ensure the reliability of the collaborator’s identity before collaboration.

(3) When a user proposes a task request, the personal assistant agent decomposes the task and queries the ADS, in accordance with the ADP, to identify collaborative agents whose capabilities match the task requirements.

(4) Agents follow AIP to form agent collaboration networks to complete complex tasks. When an external tool is needed, Agent contacts the tool agengt and invokes the appropriate tool.

4. Agent Trusted Registration (ATR)

The IoA should be a safe and reliable agent ecosystem, and the agents running in it with the ability to perform tasks independently should be reliable entities. To achieve this, each agent should have the following two necessary conditions:

* Obtain a globally unique identity from the ARS, which is an AIC.

* Obtain a digital certificate from the CAS designated by the ARS that can be used for authentication, called a Certificate of Agent Identity (CAI).

Each agent should have a unique AIC from the ARS on which the agent first registered. In the IoA, there can be multiple ARS, each of which should be a service entity that has been recognized by consensus (e.g., certified by a governing body). Each agent can choose a different ARS to register with and receive an assigned AIC based on their needs.

After the agent obtains the AIC, it also needs to obtain the CAI from the CAS. In the IoA, there can be multiple CAS, and each CAS should be a service entity recognized by consensus (such as certification by a management agency).

4.1. Basic Workflow of Agent Trusted Registration

The basic workflow of agent trusted registration includes the following steps:

(1) The agent provider sends a registration request to the ARS to register the agent, which must include all or part of the ACS information. Apply for AIC;

(2) The ARS manually reviews the content of the registration request;

(3) After the ARS approves the approval, the agent is assigned an AIC;

(4) The agent provider applies for an CAI from CAS, and the request must be accompanied by an AIC;

(5) CAS requests the ARS to obtain the agent's details based on the AIC;

(6) The ARS returns the agent details to the CAS based on the AIC;

(7) CAS reconciles the agent details;

(8) The CAS issues an ACME HTTP-01 challenge to the agent provider for domain validation and issues a CAI upon successful verification.

5. Agent Identity Authentication (AIA)

In the basic workflow of agent trusted registration, agent get the necessary AIC and CAI.

Agent Identity Authentication supports multiple security protocols. TLS 1.3 mutual TLS (mTLS) [RFC8446] is recommended. In this process, the connection-initiating agent acts as the TLS client, while the receiving agent serves as the TLS server. The mutual TLS handshake comprises three phases: Key Exchange, Server Parameters, and Authentication. All communications beyond the key exchange phase are encrypted.

5.1. Phase 1: Key Exchange

(1) Client sends ClientHello message with supported cipher suites and ephemeral public key;

(2) Server responds with ServerHello message, selecting cipher suites and returning ephemeral public key;

(3) Both parties derive a pre-master secret using ECDHE with the peer's ephemeral public key and their own ephemeral private key, then generate session keys and handshake keys via HKDF. All subsequent messages are encrypted;

5.2. Phase 2: Server Parameters Transmission

(4) Server transmits CertificateRequest (requesting client certificate) and Encrypted Extensions messages;

5.3. Phase 3: Authentication

(5) Server sends CAI, CertificateVerify (signing all previous handshake messages hash using signature algorithm from ClientHello's signature_algorithms extension and server's private key), and Finished message;

(6) Client sends CAI, CertificateVerify, and Finished message;

(7) Both parties validate peer's CAI and CertificateVerify.

(8) Upon successful mutual verification, encrypted data transfer begins.

6. Agent Discovery (ADP)

Agent discovery refers to the process of dynamically identifying and matching the functions and services provided by distributed agents according to the requirements of a user task. Agent capability discovery enables dispersed agents to be efficiently searched and invoked. It is the prerequisite and foundation for large-scale collaboration among agents.

6.1. Core Roles in Agent Discovery

* Leader: Agent who issues tasks and organizes interactions. Leader is the initiator of agent discovery requests.

* Partner: Agent who accepts tasks and provides services.

* Agent Discovery Server: Receives agent discovery requests and returns discovery results. Each agent discovery server must establish and maintain mutual trust relationships with external discovery servers, maintain a trusted discovery server list, and perform agent discovery within the trusted scope.

6.2. Data Sources for Agent Discovery Servers

Agent discovery in the IoA relies on a multi-level agent discovery architecture. Through synchronization mechanisms, it obtains authoritative registration data from Agent Registration Servers, to support capability-based search, matching, and decision-making.

An Agent Registration Server is responsible for the submission, verification, approval, revocation, and version management of agent identities, to ensure data integrity, legitimacy, and consistency. It also maintains the agent's AIC and version history as the reliable baseline for discovery. After obtaining registration data, the discovery server stores it in a local database and builds customized indexes to enable efficient handling of discovery requests. A discovery server may synchronize with multiple registration servers to support multi-source registration scenarios.

6.2.1. Data Synchronization

The primary data synchronization methods between discovery servers and registration/monitoring servers include:

* Full Snapshot: Used to obtain a complete data view for initialization or reconstruction when starting up for the first time, when data is lost, or when consistency is compromised;

* Incremental Changes: Used after synchronization becomes stable to continuously synchronize newly added or updated data at low cost, to maintain nearly real-time consistency;

* Active Push: For stricter timeliness requirements, the data provider may proactively push changes instead.

Note that the discovery server may obtain operational status data from agent monitoring related servers, which will be elaborated in Section 9. They may use the same data synchronization mechanism.

6.3. Basic Workflow of Agent Discovery

The basic workflow of agent discovery includes the following steps:

(1) The Leader receives a user task and analyzes it to determine the required capabilities (skills) of Partners;

(2) The Leader sends an agent discovery request to the Agent Discovery Server, where the request contains explicit query information or constrained query conditions;

(3) The Agent Discovery Server matches the query conditions with the available agent descriptions;

(4) The Agent Discovery Server returns a list of candidate agents that satisfy the requirements to the Leader;

(5) The Leader selects Partners from the candidate list according to its own strategy, performs identity verification, and collaborates with them to complete the task.

6.4. Different Agent Discovery Methods

* In-Domain Discovery: Upon receiving a discovery request, the discovery server performs matching, ranking, and filtering of available Partners within the local domain; if the Leader' s requirements are met, it directly returns the candidate list of serving agents.

* Cross-Domain Discovery: Discovery servers across multiple autonomous domains can establish connections based on trust relationships between agent providers. If the discovery request for an agent cannot be fulfilled within one autonomous domain, the discovery server in one domain may forward the discovery request to the discovery servers in the neighboring domain. If the neighboring discovery server locates a suitable agent, it returns a discovery response.

There are two cross-domain discover mechanisms: the chained forwarding and the aggregated forwarding. For the chained forwarding, the discovery request is forwarded hop-by-hop along a trusted chain, and the continued forwarding process is constrained by a maximum hop count limit. For the aggregated forwarding, the upstream discovery server can select multiple downstream discovery servers from the trusted list to initiate parallel discovery requests, and the continued forwarding process is constrained by a maximum participant limit. During agent discovery, each discovery server must ensure that the forwarding depth or parallel fanout do not exceed the configured limits. The AICs of all discovery servers along the route must be recorded, to ensure that the discovery results are trustworthy and that the discovery process is controllable and traceable.

7. Agent Interaction (AIP)

In the Internet of Agents, agents may form different collaborative networks under various task scenarios. To ensure that agents can communicate safely and effectively across different scenarios, AIP defines standardized mechanisms and processes to promote deep collaboration between agents.

7.1. Core Roles in Agent Interaction

In AIP, the interaction between agents are task-driven, based on which agents can be divided into two roles:

* Leader: Agent who issues tasks and organizes interactions. There should only be one Leader in a complete task execution process.

* Partner: Agent who accepts tasks and provides services. After Partner receives a task from the Leader, it executes and returns the execution result.

7.2. Interaction mode in AIP

In AIP, there are three interaction modes between the Leader and the Partner: peer to peer mode, grouping mode and hybrid mode.

* Peer to peer mode: In this mode, the Leader maintains separate communication connections with each Partner, ensuring context isolation. Each message in this mode has only one sender and receiver.

* Grouping mode: In this mode, interaction messages between agents are distributed through a message queue. Message in this mode has only one sender but may has multiple receivers.

* Hybrid mode: In this mode, the Leader and Partner may interact directly or through the message queue according to the Leader’s task planning.

Agents in the existing agent collaboration network can interact with other agents as Leaders through the interaction mode defined in AIP to introduce them into the agent domain, forming a more complex agent collaboration network.

7.3. Messaging in AIP

Messaging in AIP includes three types: RPC, streaming, and notification.

* RPC: For instant scenarios like synchronous messages, managing task states, or performing actions requiring immediate responses.

* Streaming: For scenarios such as chunking large files, real-time log pushes, or long-running data transfer operations.

* Notification: Event-driven asynchronous callback methods for agents to send notifications to other agents when specific conditions are met.

8. Tool Invocation (TIP)

In the Internet of Agents, to accomplish users' complex tasks, agent collaboration often involves calling external tools and accessing external data. TIP provides a standardized interface for agents to call tools, enabling different agents to use various external tools or services through a unified method.

8.1. Core Roles in Tool Inbocation

TIP follows a client-host-server architecture:

* Tool Agent (TA): Tool Agent is the agent that uses the TIP to call tool, which usually acts as a Partner to interact with the Leader through AIP, and autonomously planning to call tools to complete the tasks published by the Leader.

* TIP Host: The TIP Host process is typically created and managed by the Tool Agent, serving as a container and coordinator for TIP Clients, and managing cross-client context aggregation.

* TIP Client: Each TIP Client maintains an isolated TIP Server connection, handle protocol negotiation and feature exchange between both parties, and send requests to the TIP Server to use its corresponding features.

* TIP Server: The TIP Server provides specialized context and functionality, exposing tools through TIP primitives, which call tools based on received requests and return execution results.

8.2. Basic Workflow of Tool Invocation

The basic workflow of tool invocation includes the following steps:

(1) Tool Agent creates a connection between TIP Client and TIP Server;

(2) Tool Agent autonomously discovers available tools and invokes them as needed;

(3) Tool Agent proactively disconnects from the TIP Server once the desired result is obtained and it is confirmed that no further tool calls are needed.

9. Agent Monitoring (AMP)

To ensure that agents' status information can be aggregated in real time and distributed in an orderly manner under a unified specification, AMP adopts log files as the data carrier for agents' states and behaviors. Through standardized mechanisms for log collection and storage, it enables centralized retention and controlled distribution of agent information, thereby providing agent registration servers, discovery servers, and other agents with capabilities for status querying and traceability.

9.1. Log File Categories

The raw log data is generated by agents. Logs include the following types, and different types of logs are stored in different files:

* Heartbeat Logs: Periodically record an agent's operational status and health condition. The core purpose of heartbeat logs is to express the fact that "the agent is still alive." They are typically generated at a fixed interval (e.g., every 10 to 60 seconds) and usually contain only lightweight information such as status indicators and brief metric summaries.

* Metrics Logs: Record an agent's performance metrics and resource utilization, focusing on "how the agent is performing," presented in structured numerical form. Typical contents include task queue status, latency percentiles (P50/P90/P99), and CPU/memory/disk/ network utilization.

* Access Logs: Record every interaction between an agent and external systems, including requests, responses, errors, debugging information, and tracing data. Access logs are "generalized interaction logs," covering traditional request logs, error logs, debug logs, and distributed tracing logs.

* Message Logs: Record sending and receiving behaviors of an agent through messaging channels (queues, topics, streams), focusing on reliable delivery, ordering, and retry status in message-driven architectures. They help diagnose issues such as message backlogs, out-of-order delivery, and duplicate delivery.

* Auditing Logs: Record security-related operations and access activities of an agent. Auditing logs serve security compliance and post-incident forensics needs, typically including fields such as operator, target, operation type, and operation result. Auditing logs usually have a long retention period (depending on regulatory requirements) and impose high requirements on integrity and tamper resistance. Transaction logs of billing systems can also be categorized as a special form of auditing logs.

* System Logs: Record status information of the agent runtime environment and related services. System logs help diagnose issues in the agent runtime environment, and are typically used together with access logs and metrics logs to provide a comprehensive troubleshooting perspective.

9.2. Log Collection and Storage

Log Collection Layer: Various logs (heartbeat, access, metrics, audit, message, system, etc.) are written by agents into local log files by type. After necessary steps such as parsing, filtering, and labeling, this layer delivers logs by type to the log storage layer in a reliable and controllable manner.

Log Storage Layer: This layer provides unified storage for different log information from different agents. According to the structural characteristics, access patterns, and lifecycle requirements of different log types, it provides differentiated storage solutions, and also supports index building and fast retrieval of log data, providing data support for subsequent monitoring analysis, alert triggering, and root-cause tracing.

After logs are stored, they can be read and processed on demand by different analysis and application systems for querying and aggregation analysis, operational monitoring and alerting, fault localization and root-cause tracing, security auditing and compliance retention, and unified presentation of key conclusions via reports or visualization, thereby supporting a continuous monitoring and governance closed loop.

9.3. The Use of Logs

Logs are the most fundamental and most trustworthy “source of truth” for an agent monitoring system. The monitoring systems continuously collect and reliably store, by type, the status changes, request interactions, key metrics, exception information, and audit records generated during agent operation, and then distribute them in a controlled manner to different entities for processing and analysis.

Logs can provide long-term data foundations for agent auditing, agent transaction, as well as network performance optimization. The IoA can achieve operational observability, alerting and diagnosis, tracing and accountability, security and compliance assurance.

10. Conclusions

This draft introduces the Agent Collaboration Protocols (ACPs) architecture for Internet of Agents. By defining a comprehensive protocol suite, including trusted registration, identity authentication, agent discovery, agent interaction, tool invocation, and agent monitoring, the ACPs address the core requirements for interoperability and secure collaboration among heterogeneous agents. The long-term goal of the ACPs is to enable IoA to serve as the critical infrastructure for agent collaboration on complex tasks, as well as to support the future large-scale and diverse applications.

11. Conventions and Definitions

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] and [RFC8174].

12. Security Considerations

This document focuses on the Agent Collaboration Protocols architecture for IoA. Security of IoA is not detailed in this document. Security considerations relevant to deployment with multiple agent service providers are suggested to be deeply discussed through other proposals.

13. IANA Considerations

This document makes no request for IANA action.

14. References

14.1. Normative References

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[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>.

14.2. Informative References

[ACPs-Github] "ACPs GitHub repository", n.d., <https://github.com/AIP-PUB>.

Authors' Addresses

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

Ke Yu
Beijing University of Posts and Telecommunications
10 Xitucheng Road, Haidian District
Beijing
100876
China
Email: yuke@bupt.edu.cn

Ke Li
Beijing University of Posts and Telecommunications
10 Xitucheng Road, Haidian District
Beijing
100876
China
Email: like1990@bupt.edu.cn

Keliang Chen
Beijing University of Posts and Telecommunications
10 Xitucheng Road, Haidian District
Beijing
100876
China
Email: chenkl@bupt.edu.cn