

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 8 January 2026

Y. Cui
Tsinghua University
C. Du
Zhongguancun Laboratory
7 July 2025

Task-oriented Coordination Requirements for AI Agent Protocols
draft-cui-ai-agent-task-00

Abstract

AI agent communication requires intelligent task level coordination to manage dynamic workloads across large-scale, heterogeneous networking environments. This draft proposes general requirements for an agent protocol to enable autonomous task coordination at scale, including dynamic task discovery, negotiation, and context-aware scheduling with real-time adaptability.

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 8 January 2026.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	2
1.1. Purpose	2
1.2. Terminology	3
1.3. Task Coordination Framework	3
2. Use Cases	4
3. Necessity	4
3.1. Task Complexity	4
3.2. Resource Optimization	5
3.3. Quality of Service	5
3.4. Dynamic Adjustment	5
4. Protocol Requirements	5
4.1. Task Description	5
4.2. Task State	6
4.3. Communication Mechanism	6
4.4. Context Sharing	8
4.5. Exception Handling	8
5. Existing Protocol Analysis	8
6. Conclusions	9
7. IANA Considerations	9
8. Security Considerations	9
9. References	9
9.1. Normative References	9
9.2. Informative References	9
Authors' Addresses	10

1. Introduction

1.1. Purpose

With the rapid advancements of AI technologies and their applications, AI agents utilizing Large Language Models (LLMs) have emerged as a pivotal direction in global technological evolution and market development. The single-agent systems exhibit inherent limitations when addressing complex tasks in dynamic environments, the efficient multi-agent collaboration for complex task completion has garnered increasing attention, wherein task-oriented coordination constitutes a critical component of standardized multi-agent systems.

This document examines the requirements for standardizing AI Agent protocols to support task coordination in multi-agent systems.

1.2. Terminology

Task:

ISO/IEC 22989, task is actions required to achieve a specific goal. These actions can be physical or cognitive. For instance, computing or creation of predictions, translations, synthetic data or artefacts or navigating through a physical space.

Shared Message Pool:

A pool where agents publish structured messages and subscribe to relevant messages based on their profiles.

Coordinator Agent:

An agent that receives tasks and decomposes or distributes tasks to other agents.

Execution Agent:

An agent responsible for executing tasks distributed by the Coordinator Agent.

Normative Language:

The key words "MUST", "REQUIRED", and "SHOULD" in this document are to be interpreted as described in [RFC2119].

1.3. Task Coordination Framework

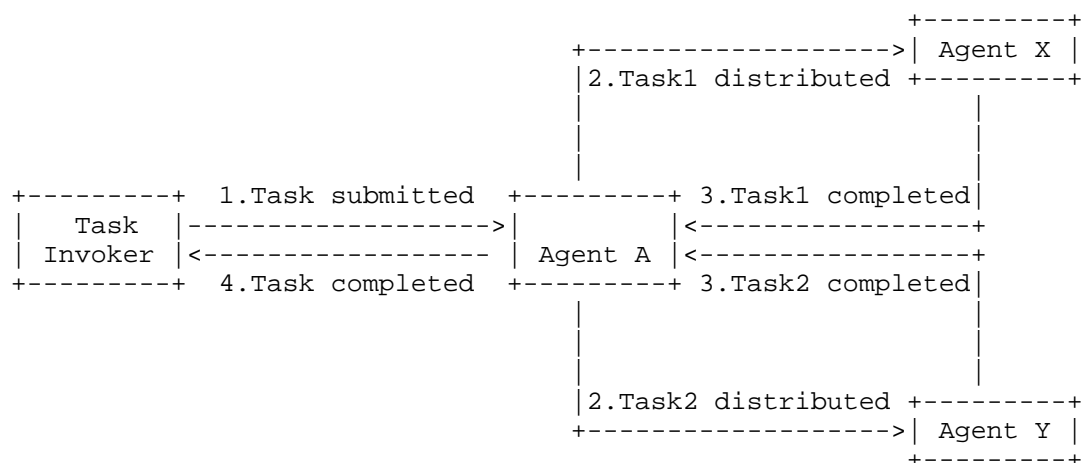


Figure 1: Task Coordination Framework

The system operates as follows: when a task invoker submits a task to Agent A (Coordinator), the agent performs task decomposition and distributes task 1 and task 2 to Agents X (Execution) and Y (Execution) respectively. Upon receiving completion notifications from both agents, Agent A aggregates the results and delivers the final output back to the originating user.

2. Use Cases

Some typical use cases in which multiple agents work together to complete tasks:

- * High throughput tasks: There are many tasks that have high bandwidth requirements. For example, in the collaborative framework for coordinating heterogeneous embodied agents—specifically, the robot dogs and drones—in a wide-area public network, the drone is assigned for wide-area surveillance and task delegation, while functionally specialized robot dogs perform ground-level operations such as video surveillance, material transport and obstacle clearance.
- * Low latency tasks: In collaborative multi-agent systems, control signal transmission tasks impose significantly more stringent latency requirements than routine model training data transfer. For example, the home robot remotely sends an alarm message to the end user.
- * High reliability tasks: Smart factory scenarios require critical reliability of agent task execution and fault-tolerant operation stability.

These categories of use cases (may be further extended) demonstrate the collaboration among agents spanning multiple distinct domains to achieve end-to-end task completion. The embodied agents (such as the robots and unmanned aerial vehicles) interacting with physical environments through embodied interfaces, while virtual agents (such as the various software applications and personal assistant) providing complementary capabilities, has demonstrated the advantages of collaboratively completing complex tasks in various scenarios.

3. Necessity

3.1. Task Complexity

As task complexity increases, heterogeneous agents require multiple interaction rounds, precise planning, ordered execution, and efficient context sharing mechanisms to enhance resolution quality and robustness.

3.2. Resource Optimization

Through task coordination and resource consumption monitoring, the multi-agent systems are able to support dynamic allocation of for example, the computing, storage and bandwidth resources to optimize the resource utilization efficiency.

3.3. Quality of Service

Task coordination may dynamically prioritize resources allocations based on for example, task priorities, agent expertise and Quality of Service requirements. This ensures timeliness and accuracy of critical tasks, reduces service response latency, and maintains output stability and reliability.

3.4. Dynamic Adjustment

The agents may update or adjust the task during task execution phase based on end user's inputs or contextual updates to better respond to the final task requirements.

4. Protocol Requirements

4.1. Task Description

Precise task descriptions or task templates are REQUIRED to ensure all agents maintain a consistent understanding of the objectives, operational constraints and criteria.

A well-defined task description:

- * Reduces ambiguity: Minimizes misinterpretations and conflicting actions among agents.
- * Enables verifiability: Translates abstract goals into executable and measurable plans.
- * Improves robustness: Ensures collaboration remains coherent and efficient under dynamic conditions.

Task descriptions assigned to different agents MUST follow the minimization principle, i.e., agents SHOULD receive only the minimal, contextually necessary information required to fulfill their tasks to prevent unauthorized access of sensitive information.

4.2. Task State

Upon receiving a task, the Coordinator Agent may decompose it into multiple tasks and delegate them to different Execution Agents via dynamic capability discovery mechanism. The AI Agent protocol design should support comprehensive state descriptions throughout the task execution lifecycle. For example, the definition of potential task states, (such as task submitted, running, suspended (awaiting external input or output from other agents), completed, canceled, rejected and failed) and the coordination operations (such as state queries, retrieval and push of intermediate results).

Based on the length of time to complete the tasks, the task can be categorized into Short-term tasks that require a single request-response interactions and Long-term tasks that may require multi-round interactions or extended waiting periods. The Coordinator Agent may dynamically adjust the target of the task according to the intermediate results of the Execution Agents and the context information. The AI Agent protocol design should support long-term and short-term tasks coordination.

4.3. Communication Mechanism

When multiple agents participate in coordinated tasks, they may need to maintain common context sharing, or subscriptions to identical message content. Different communication mechanisms, such as request/response and broadcast may need to be supported in the AI Agent protocol. In the moving large artifact task in intelligent factory case, the multiple robots responsible for the transportation needs to obtain consistent route and destination data to ensure operational coherence.

The applicable communication mechanisms may be vary for different agent communication structures, which directly affects the message delivery efficiency, implementation feasibility and system complexity. The typical agent communication structures [Multi-Agents] including:

- * Layered communication: Agents at different layers in the hierarchical structure have different roles or capabilities. Communication occurs either horizontally (intra-layer) or vertically (adjacent layers only).
- * Decentralized communication: Multiple agents form a peer-to-peer network permitting direct communications between any two agents.

- * Centralized communication: Central agents coordinate the communications, with the other agents primarily interact through the central node.
- * Shared message pool [MetaGPT]: Agents publish structured messages to the pool and subscribe to the message types matching their profiles.

The task collaboration mode may also affect the communication mechanism between agents:

- * In the primary/secondary mode, a Coordinator Agent decomposes a task into multiple tasks and distributes them to Execution Agents for processing. The Execution Agents return their execution results to the Coordinator Agent, which aggregates the results and delivers them to the end user.
- * In the peer negotiation mode, a Coordinator Agent distributes the task of the end user to the Execution Agents, which independently execute their assigned tasks. The Execution Agents return their results directly to the end user via the Coordinator Agent, without requiring further processing by the Coordinator Agent.
- * In subscription mode, a Coordinator Agent may delegate subscription-based tasks to Execution Agents. For example, a task including "book a ticket to Beijing one week before the New Year holiday" will be assigned by the Coordinator Agent which will perform the task at the specified time to the Execution Agents.

Different tasks may require long and short connections, the AI Agent protocol should be able to provide mechanisms beyond simple request/response, including the complex interaction modes for example message multicast, publication/subscription (PUB/SUB), asynchronous notifications.

The AI Agent protocol design MUST consider support for relay nodes to facilitate task message forwarding. Relay nodes SHOULD prioritize message scheduling and forwarding based on task requirements to ensure efficient agent collaboration and meet transmission QoS objectives.

For example, relay nodes MAY implement the following priority hierarchy (from highest to lowest):

- * Control signaling transmission tasks
- * Media stream transmission tasks

- * Training data stream transmission tasks

This prioritization scheme ensures that critical messages receive preferential treatment during congestion or resource contention scenarios.

4.4. Context Sharing

When delegating tasks to Execution Agents, the Coordinator Agent may include task-relevant contextual about the contact information of the end user, the task itself, the historical preference information known by the Coordinator Agent, and other necessary conversation data, to facilitate the task execution. For example, in trip planning case, this may encompass historically booked flight/hotel preferences or dynamically perceived context like recent user dialog. The AI Agent protocol design should consequently support context sharing mechanisms through standardized definitions of context types, length constraints, and encoding formats to enhance the effectiveness of task execution.

The context sharing MAY have an impact on privacy of the user, it is necessary to consider the limitations of the scope of context sharing, especially for the sensitive information e.g. name, age, address of the user.

4.5. Exception Handling

Exception handling constitutes a critical mechanism for multi-agent collaborative task execution. If an execution agent cannot complete an assigned task due to lack of skills or overloaded, the failure in task execution may lead to such as releasing the connections.

5. Existing Protocol Analysis

Task-oriented coordination is compatible with multiple types of existing protocols, such as TCP, HTTP and etc.

Transmission Control Protocol (TCP) is to provide reliable, orderly, connection-oriented data transmission services for end-to-end communication. In the task-oriented coordination scenarios, some TCP capabilities can be directly reused (e.g. retransmission mechanism, congest control and etc.).

The Hypertext Transfer Protocol (HTTP) is the application layer protocol for distributed, collaborative, hypermedia information systems. Some HTTP features are also applicable to task-oriented coordination.

However, the task-oriented coordination needs to support finer-grained access control and context information anonymization mechanisms, which may need the enhancements on the protocol, defining the task-oriented coordination mechanism at the session layer has some advantages.

6. Conclusions

Task-oriented coordination constitutes a critical function for multi-agent collaboration. This document discusses the necessity of introducing task-oriented coordination to address complex tasks, optimize resource utilization, and guarantee service quality. Consequently, it analyzes the requirements imposed by task-oriented coordination on AI Agent protocol design, specifically concerning task descriptions, task states, communication mechanisms, context sharing, and exception handling.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

When designing the task-oriented coordination for AI agents communication, privacy should always be considered.

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/rfc/rfc2119>>.

9.2. Informative References

[Multi-Agents]

Guo, T., Chen, X., Wang, Y., Chang, R.i., Pei, S., Chawla, N.V., Wiest, O., and X. Zhang, "Large Language Model based Multi-Agents: A Survey of Progress and Challenges", 2024.

[MetaGPT]

Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Zhang, C., Wang, J., Wang, Z., Yau, S., Lin, Z., Zhou, L., Ran, C., Xiao, L., Wu, C., and J. Schmidhuber, "MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework", 2023.

Authors' Addresses

Yong Cui
Tsinghua University
Beijing, 100084
China
Email: cuiyong@tsinghua.edu.cn
URI: <http://www.cuiyong.net/>

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