Network Working Group
Internet-Draft
Intended status: Informational
Expires: 23 November 2026

T. Reddy
Z. Sarker
Nokia
K. Yao
China Mobile

22 May 2026

Agentic AI Use Cases and Requirements
draft-agentic-ai-usecases-requirements-00

Abstract

This document describes use cases for agentic AI communication systems and derives protocol requirements from those use cases. The requirements are intended to guide IETF standardization work on protocols in the context of agent-to-agent communication, agent-to-tool communication, with focus on multimodal communication, session management, discovery, communication security, agent identity and authentication.

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 23 November 2026.

Copyright Notice

Table of Contents

1. Introduction

An AI agent is an autonomous, adaptive intelligent software system that uses AI models to complete a specific objective on behalf of a user or another AI agent. It makes decisions, executes actions, and interacts with other agents through tasks and tools. Unlike traditional software workloads that follow fixed execution paths, an AI agent dynamically determines at run time which actions to take, which tools to invoke, and which agents to collaborate with, based on reasoning over its goals and context.

This document presents use cases that illustrate the key interaction patterns of agentic AI communication systems, and derives protocol requirements from those use cases. The requirements are intended to drive development of protocols and a protocol framework for agentic AI systems.

The use cases in this document cover interaction patterns for agentic AI communication systems. This document takes into account related use case and problem statement documents including [SCRM], [YAO], [SONG], and [ROSENBERG], and existing protocol work including [A2A] and [MCP].

2. Terminology

AI Agent: An autonomous software entity that perceives its environment, maintains internal state, and executes actions to achieve specified goals, potentially including communication with other agents or invocation of external tools.

Agent Identity: Identity information associated with an AI agent, used for authentication and accountability within agentic communication systems.

Agentic AI Communication System: A system comprising one or more AI agents that communicate with each other, with users, and with external tools or services to complete tasks. The communication interfaces between these entities are the subject of protocol standardization in this document.

Agent-to-Agent Communication: Direct or brokered communication between two or more AI agents, where brokered communication involves an intermediary agent or coordination service, as distinguished from communication between an agent and a user or between an agent and a tool.

Capability: A description of what an agent can perform, including inputs, outputs, constraints, and required conditions.

Context: The set of data, state, and history shared between agents to enable task execution and coordination.

Coordinator Agent: An agent that distributes a shared problem or task to a group of peer agents, aggregates their outputs, and iteratively drives them toward a collective result or consensus.

Delegation: The act of an agent requesting another agent to execute a task on its behalf.

Initiating Agent: An agent that receives an initial request and delegates subtasks to peer agents. Any peer agent may itself delegate further to other agents without routing through the initiating agent.

Mediator: An entity that serves as a proxy or mediator for external tools, APIs, databases, or other resources that other agents require but cannot directly access.

Message: A discrete unit of communication exchanged between agents containing structured data such as a task request, response, progress update, event notification, or control signal.

Modality: A category of data format used for input or output in agent communication, such as text, audio, image, or video. A session may support one or more modalities simultaneously.

Orchestrator Agent: An agent that acts as a controller, coordinating the activity of other agents by decomposing goals into sub-tasks and delegating those sub-tasks to appropriate agents.

Peer Agent: An agent that receives delegated subtasks from another agent and may itself delegate further to other agents.

Session: A logical communication exchange shared between two or more agents over a period of time, which may persist across multiple individual message exchanges and network connections. A session can carry and maintain one or more contexts shared between agents.

Task: A unit of work submitted by a user to an agent, or delegated by one agent to another.

Task State: The current execution status of a task (e.g., pending, in-progress, completed, failed).

Tool: An external service invoked by an agent to retrieve data or perform operations. A tool is not necessarily an agent and may not participate in agent-to-agent communication.

User: A human that initiates interaction with an AI agent by submitting a request or task.

3. Common Protocol Requirements

The following baseline requirements apply to both agent-to-agent and agent-to-tool protocol interactions across all use cases and are not repeated per use case.

Each per-use-case requirement is tagged with one or more of the following protocol area tags to allow cross-use-case navigation:

* *Discovery*: Requirements related to locating, advertising, or selecting agents, tools, or capabilities.

* *Transport*: Requirements related to message delivery, streaming, cancellation, session management, and data transfer.

* *Security*: Requirements related to confidentiality, integrity, and authorization.

* *Authentication*: Requirements related to identity verification and credential delegation.

REQ-ID	Description	Tag
CMN-1	The protocol is required to support any client application to communicate with any agent service.	Discovery, Authentication
CMN-2	Mutual authentication is required between all communicating parties.	Authentication
CMN-3	All protocol traffic is required to be encrypted and integrity-protected in transit.	Security
CMN-4	Structured error responses are required to include an authorization scope violation type, reported by the orchestrator or mediator when an agent attempts an action that exceeds or contradicts the scope delegated to it.	Security
CMN-5	Structured error responses are required, distinguishing at minimum: authentication failure, authorization	Transport

	failure, timeout, and internal error.	
CMN-6	The protocol is required to provide a means to signal task priority so that critical-path tasks can be scheduled ahead of lower-priority ones.	Transport
CMN-7	The protocol is required to support cryptographic algorithm agility, ensuring that cryptographic algorithms used for encryption, authentication, credential verification, and integrity protection can be negotiated and updated over time, in accordance with [RFC7696].	Security, Authentication
CMN-8	The protocol is required to provide a means to verify the agent authentication credentials validity used by agents at the time of use.	Authentication
CMN-9	The protocol is required to support signaling credential revocation and invalid credential outcomes.	Security

Table 1

4. Use Cases

4.1. Simple Single-Agent Task

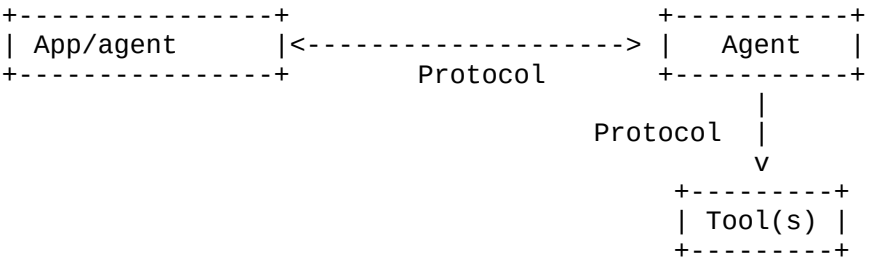
4.1.1. Description

A user submits a task to an AI agent via a client application. The agent executes the task by invoking one or more tools and returns results to the user. The tools invoked by the agent may reside in the same or a different administrative domain. The agent protocol is required to support multiple input and output modalities, and the client application and agent are required to be able to negotiate which modalities are active for the session.

This use case covers the protocol interface between the client application and the agent. The interaction between the user and the client application is out of scope. This use case assumes that the user communicates with the agent via a client application; direct communication between a user and an agent without an intermediary client application is not covered in this use case.

This interaction pattern is described in [ROSENBERG] and [SCRM].

4.1.2. Interaction Flow



4.1.3. Protocol Requirements

REQ-ID	Description	Tag
A1-1	The protocol is required to support incremental streaming of agent output, allowing partial results to be delivered to the client before the agent has completed processing.	Transport
A1-2	The protocol is required to define a task cancellation message that the client can issue at any point during task execution.	Transport
A1-3	The protocol is required to define structured error message types that distinguish at minimum: transport failure, tool invocation failure, and agent processing failure.	Transport
A1-4	The protocol is required to support multiple modalities for both input and output.	Transport
A1-5	The protocol is required to support modality negotiation at session setup, allowing the client and agent to agree on which modalities are active for the session.	Discovery, Transport
A1-6	The protocol is required to support agent-initiated notifications to	Transport

	the client during task execution.	
A1-7	The protocol is required to support concurrent invocation of multiple tools within a single agent task, where tools may be operated by distinct providers across different administrative domains, each with independent authentication and authorization requirements.	Discovery, Transport, Security
A1-8	The protocol is required to support bulk transfer of large data between communicating parties, applicable to both agent-to-tool and agent-to-agent interactions.	Transport
A1-9	A delegation mechanism is required to be defined by which an agent presents to a tool provider a credential attesting the authorization for the requested tool access, without exposing the client’s primary credentials.	Authentication

Table 2

4.2. Orchestrator and Agent Collaboration

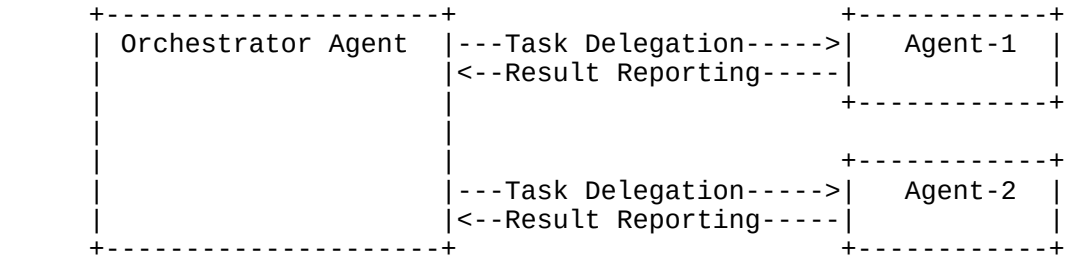
4.2.1. Description

An orchestrator agent acts as a controller, decomposing a task into subtasks and delegating them asynchronously to one or more other agents. The orchestrator decides which other agents to invoke, sequences the delegation, and aggregates results to continue task execution. Each agent executes the respective subtask independently and reports results back to the orchestrator.

It should be noted that AI models are stateless by nature — each inference call processes only what is explicitly provided with a particular context, with no persistent memory between calls. The application layer is responsible for maintaining the context across the calls by carrying conversation history, intermediate results, and task state. Session continuity is therefore required to preserve this accumulated context across network interruptions, ensuring that a reconnecting agent can restore the prior task context without having to reconstruct it from scratch.

This pattern is described in [ROSENBERG] and reflected in [A2A], and is implemented in deployed multi-agent frameworks including [AUTOGEN], [LANGCHAIN], and [OPENAI-AGENTS].

4.2.2. Interaction Flow



4.2.3. Protocol Requirements

REQ-ID	Description	Tag
B1-1	The protocol is required to facilitate task delegation for an orchestrator to agents that includes task delegation and acknowledgement message types.	Transport
B1-2	The protocol is required to support asynchronous delegation, allowing the orchestrator to delegate to multiple agents without waiting for each to complete before proceeding.	Transport
B1-3	The protocol is required to define a result reporting message by which an agent returns its completed output to the orchestrator.	Transport
B1-4	The protocol is required to support streaming of intermediate results from the agent to the orchestrator during task execution.	Transport
B1-5	The protocol is required to define a task cancellation message that the orchestrator can send to an agent to abort a delegated subtask.	Transport

B1-6	The protocol is required to support persistent session identifiers that survive network interruption, and is required to define a session resumption message by which an agent re-attaches to an interrupted session restoring the prior task context.	Transport

Table 3

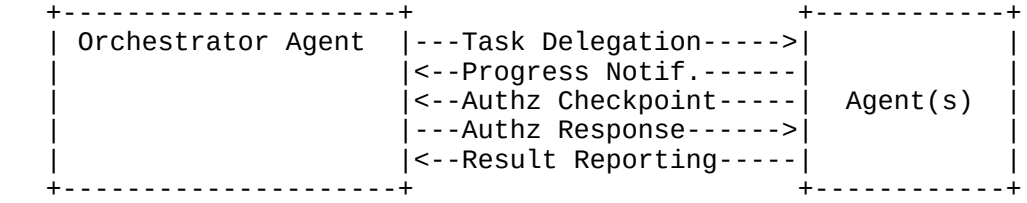
4.3. Long-Running Delegated Task with Authorization Checkpoint

4.3.1. Description

An orchestrator agent delegates a long-running task to other agents. The delegated agent executes the task autonomously and sends progress notifications to the orchestrator. At any certain point one or more delegated agents pause and request explicit authorization from the orchestrator before proceeding further. The orchestrator may relay this authorization request to the invoker (user or agent) or resolve it autonomously based on policy.

This pattern is reflected in the In-Task Authorization mechanism defined in [A2A].

4.3.2. Interaction Flow



4.3.3. Additional Protocol Requirements

This use case builds on the requirements defined for Section 4.2 and introduces additional requirements specific to authorization checkpoints in delegated tasks.

REQ-ID	Description	Tag
B2-1	The protocol is required to support agent-initiated progress notifications to the delegating agent during task execution.	Transport
B2-2	The protocol is required to define an authorization checkpoint message by which an agent pauses task execution and requests explicit authorization from the orchestrator before proceeding. The message is required to include sufficient context for the authorizing party to make an informed decision, including the action to be taken and its potential consequences.	Transport, Security
B2-3	The protocol is required to define the valid responses to an authorization checkpoint, including at minimum: approve, deny, and approve with modified parameters. A denial is required to be conveyed as an explicit error response.	Transport, Security
B2-4	The protocol is required to support a response timeout, after which the agent treats the request as unresolved and halts the affected subtask.	Transport

Table 4

4.4. Peer Collaborative Multi-Agent Problem Solving

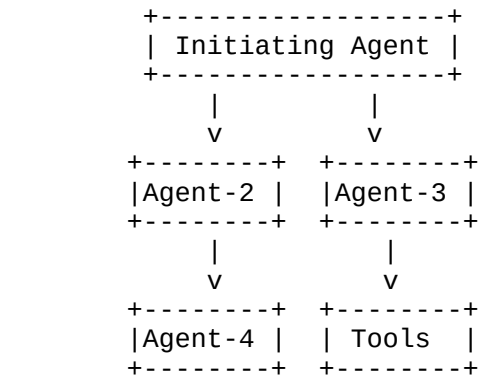
4.4.1. Description

A task requires coordinated problem solving across multiple agents, where no single agent has full authority or capability to complete the task alone. The agent that receives the initial request dynamically delegates subtasks to peer agents based on their advertised capabilities. Any agent may itself delegate further to other agents and use other tools, forming a dynamic collaboration graph. Each agent remains opaque to others, collaborating only through the protocol interface.

This use case introduces multi-hop delegation chains that are not present in Section 4.2. Each agent in the chain may delegate further to other agents, and authorization scope is required to be progressively constrained at each hop.

This use case is described in [A2A] and [ROSENBERG].

4.4.2. Interaction Flow



4.4.3. Additional Protocol Requirements

This use case builds on the requirements defined for Section 4.2 and introduces additional requirements specific to multi-hop delegation chains.

REQ-ID	Description	Tag
B3-1	The protocol is required to support multi-hop delegation chains, where an agent that receives a delegated subtask may itself delegate further to other agents. At each hop, the delegating agent is required to present a credential that does not exceed the authorization scope of the credential it received.	Authentication, Security
B3-2	The protocol is required to preserve the identity of the originating entity across all hops in the delegation chain, such that any agent in the chain can determine the identity of the	Authentication, Security

	entity that originally authorized the task.	
B3-3	The protocol is required to support transferable credentials that carry the original authorization constraints across all hops in the delegation chain. Each receiving agent is required to be able to cryptographically verify that the credential presented to it was issued by the delegating agent and that the chain of delegation traces back to the original authorization.	Authentication, Security
B3-4	The protocol is required to ensure that authorization granted to an agent in a delegation chain, including for tool invocations, is derived from the authorization issued by the initiating agent, and not from the identity or authorization scope of any intermediate agent in the chain.	Authentication, Security
B3-5	The protocol is required to define a capability registration mechanism by which agents can publish metadata describing their capabilities, supported protocols, rate limits, authentication methods, authorization mechanisms, and authorization scopes to a discovery service or registry.	Registration, Discovery
B3-6	The protocol is required to define a capability discovery mechanism by which agents can query a discovery service or registry to discover and select appropriate peer agents at runtime without requiring prior peer-specific configuration.	Discovery
B3-7	The protocol is required to define an agent identifier format that uniquely represents an agent identity and is resolvable to the agent’s communication endpoint	Discovery

		using an appropriate discovery mechanism.	
B3-8	The protocol is required to ensure that advertised capabilities are integrity-protected, such that a discovering agent can verify they have not been tampered with.	Discovery, Security	

Table 5

4.5. Cooperative Reasoning and Consensus Formation

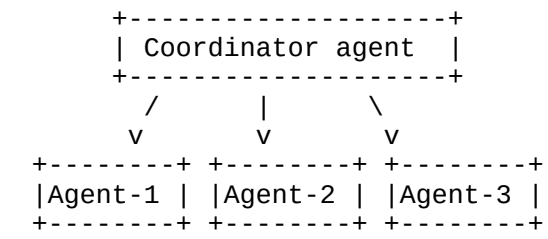
4.5.1. Description

A set of peer agents is tasked with analyzing a shared problem independently and exchanging intermediate reasoning outputs to converge on a collective conclusion. A coordinator agent distributes the problem to all participating agents, collects their reasoning outputs, and drives the convergence process across multiple rounds until a consensus conclusion is reached. Unlike Section 4.4, all agents work on the same problem rather than different subtasks.

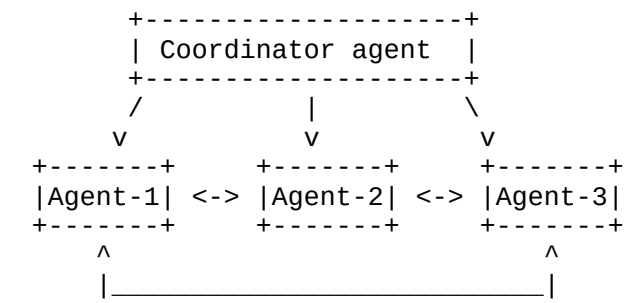
Two communication topologies are possible. In the first, agents communicate only through the coordinator, which acts as the central hub for all message exchange. In the second, agents may also communicate directly with each other to exchange intermediate reasoning outputs without routing through the coordinator agent. The second topology introduces the same multi-hop authorization requirements defined in Section 4.4.

4.5.2. Interaction Flow

The coordinator-mediated topology:



The direct agent-to-agent topology:



4.5.3. Protocol Requirements

This use case builds on the requirements defined for Section 4.2 and introduces additional requirements specific to group message delivery.

4.5.4. Additional Protocol Requirements

REQ-ID	Description	Tag
B4-1	The protocol is required to support one-to-one, one-to-many, and many-to-many message delivery among a defined group of agents. Group membership is required to be dynamic, allowing agents to join or leave the group during the course of an exchange.	Transport, Security

Table 6

4.6. Tool, Data, and API Mediation Between Agents

4.6.1. Description

In many multi-agent deployments, access to external resources — APIs, databases, enterprise systems, or hardware interfaces — is intentionally mediated through a designated mediator. Other agents request the mediator to perform actions or retrieve data on their behalf, rather than directly invoking external systems. This architecture allows access control, auditing, rate limiting, and schema normalization to be applied uniformly at the mediation layer.

The mediator may also serve as an adapter between the agent protocol and non-agent systems or other services that do not natively support agent communication protocols, or between different agent

communication protocols such as translating between the agentic protocol suite being developed at the IETF and existing protocols such as [MCP] and [A2A]. In this role, the mediator is responsible for protocol translation and for presenting the appropriate credentials to the target system on behalf of the requesting agent.

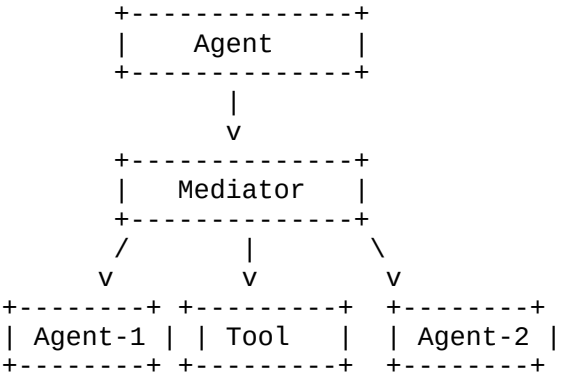
The mediator may additionally act as a request router, dispatching requests to appropriate agents or tools based on the content and context of the request, without requiring the requesting agent to have prior knowledge of which agent or tool is most appropriate.

The mediator may also validate agent requests before invocation, checking whether the action being requested matches the authorization granted to the agent and whether execution would cause unintended or irreversible side effects.

Note that mediating can be a function within an orchestrator.

This pattern is reflected in the MCP server architecture defined in [MCP] and the agent routing patterns discussed in [A2A].

4.6.2. Interaction Flow



4.6.3. Additional Protocol Requirements

REQ-ID	Description	Tag
B5-1	The protocol is required to define error response types for request validation failure (rejected due to potential unintended or irreversible side effects) and protocol translation failure (rejected on unsuccessful translation of a request	Transport, Security

	or response between supported protocols), distinct from authorization failure.	
B5-2	The protocol is required to support exchange of structured (audit) record for each action performed on behalf of a requesting agent, including the requesting agent's identity, the authorization credential presented, the action taken, and the outcome.	Security

Table 7

5. Security Considerations

Security considerations are addressed throughout this document via the Identity, Authentication, and Delegation requirements defined for each use case. Agent identity and authentication mechanisms are further discussed in [KLRC].

Agent identity information is considered to be sensitive, particularly in multi-domain deployments. Use of persistent identifiers across sessions and domains can enable tracking and correlation of agent activity. Protocol designers need to consider mechanisms such as pseudonymous or temporary identifiers to reduce linkability, while preserving the ability to audit and enforce accountability where required. The trade-offs between privacy, accountability, and traceability need to be considered in the design of agent identity mechanisms.

6. IANA Considerations

This document has no IANA actions.

Acknowledgements

Thanks to Borislava Gajic, Julien Maisonneuve, Parisa Foroughi, Laurent Ciavaglia, Peter Leis and Sina Khatibi for the discussions and comments.

Informative References

[A2A] "Agent2Agent Protocol Specification", n.d., <https://a2a-protocol.org/latest/specification/>.

[AUTOGEN] "AutoGen: A Framework for Multi-Agent Conversation", n.d., <https://microsoft.github.io/autogen/stable/>.

[KLRC] "AI Agent Authentication and Authorization", n.d., <https://datatracker.ietf.org/doc/draft-klrc-aiagent-auth>.

[LANGCHAIN] "LangChain Agent Framework", n.d., <https://python.langchain.com/docs/concepts/agents/>.

[MCP] "Model Context Protocol Specification", n.d., <https://modelcontextprotocol.io/specification/2025-11-25>.

[OPENAI-AGENTS] "OpenAI Agents SDK", n.d., <https://openai.github.io/openai-agents-python/>.

[RFC7696] "Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithms", n.d., <https://www.rfc-editor.org/rfc/rfc7696>.

[ROSENBERG] "Framework, Use Cases and Requirements for AI Agent Protocols", n.d., <https://datatracker.ietf.org/doc/draft-rosenberg-aiproto-framework>.

[SCRM] "Agentic AI Use Cases", n.d., <https://datatracker.ietf.org/doc/draft-scrm-aiproto-usecases>.

[SONG] "Problem Statement and Requirements for Dynamic Multi-agent Secured Collaboration", n.d., <https://datatracker.ietf.org/doc/draft-song-dmsc-problem-statement>.

[YAO] "Problem Space Analysis of AI Agent Protocols in IETF", n.d., <https://datatracker.ietf.org/doc/draft-yao-catalist-problem-space-analysis>.

Authors' Addresses

Tirumaleswar Reddy
Nokia
Bangalore
Karnataka
India
Email: kondtir@gmail.com

Zaheduzzaman Sarker
Nokia
Sweden
Email: zaheduzzaman.sarker@nokia.com

Kehan Yao
China Mobile
Email: yaokehan@chinamobile.com