ETSI GR ZSM 020 V1.1.1 (2026-01)

GROUP REPORT

Zero-touch network and Service Management (ZSM); Study on the Utilization of Agents in Autonomous Networks

Disclaimer

The present document has been produced and approved by the Zero-touch network and Service Management (ZSM) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference
DGR/ZSM-020_GRonAgents

Keywords
agents, AI

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI IPR online database.

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**, **LTE™** and **5G™** logo are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Group Report (GR) has been produced by ETSI Industry Specification Group (ISG) Zero-touch network and Service Management (ZSM).

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document provides a comprehensive review on the definition of agents, how agents can be used for network and service management and how they can fit in the ZSM framework. To achieve this, the focus was on use cases involving agents for network and service management and on multi-agent communication use cases.

As autonomous technologies continue to advance, the use of autonomous agents for network and service management becomes relevant.

In addition, effective communication between agents is important for coordinating tasks, sharing knowledge, and optimizing performance across various applications. The present document reviews its applicability for zero-touch network and service management.

By analysing existing frameworks and methodologies, the present document provides recommendations of additional capabilities needed in the ZSM framework to support utilization of agents.

Collaboration with other SDOs (e.g. in IRTF NMRG, ITU-T SG13, TM Forum) is recommended when appropriate.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents may be useful in implementing an ETSI deliverable or add to the reader's understanding, but are not required for conformance to the present document.

[i.1] Stuart Russell and Peter Norvig: "Artificial Intelligence - A Modern Approach", Fourth Edition.

[i.2] ETSI GS ZSM 009-1: "Zero-touch network and Service Management (ZSM); Closed-Loop Automation; Part 1: Enablers".

[i.3] Google for Developers: "Announcing the Agnt2Agent Protocol (A2A)", R. Surapaneni, M. Jha, M. Vakoc, T. Segal.

[i.4] V. Pandey: "Building the Internet of Agents: Introducing AGNTCY.org".

[i.5] ETSI GS ZSM 016: "Zero-touch network and Service Management (ZSM); Intent-driven Closed Loops".

[i.6] ETSI GR ZSM 011: "Zero-touch network and Service Management (ZSM); Intent-driven autonomous networks; Generic aspects".

[i.7] ETSI GR ZSM 009-3: "Zero-touch network and Service Management (ZSM); Closed-Loop Automation; Part 3: Advanced topics".

[i.8] ETSI GS ZSM 002: "Zero-touch network and Service Management (ZSM); Reference Architecture".

[i.9] ETSI GS ZSM 014: "Zero-touch network and Service Management (ZSM); ZSM security aspects".

[i.10] Xiao, Y., Shi, G., Zhang, P.: "Towards Agentic AI Networking in 6G: A Generative Foundation Model-as-Agent Approach", March 2025.

[i.11] Dev, K., Khowaja, S. A., Singh, K., Zeydan, E., Debbah, M.: "Advanced Architectures Integrated with Agentic AI for Next-Generation Wireless Networks", February 2025.

[i.12] Arzo, S. T., Scotece, D., Bassoli, R., Granelli, F., Foschini, L., Fitzek, F.H.P.: "A New Agent-Based Intelligent Network Architecture", November 2022.

[i.13] Chen, W., You, Z., Li, R., Guan, Y., Qian, C., Zhao, C., Yang, C., Xie, R., Liu, Z., Sun, M.: "Internet of Agents: Weaving a Web of Heterogeneous Agents for Collaborative Intelligence", July 2024.

[i.14] CrewAI: "The Leading Multi-Agent Platform".

[i.15] LangGraph: "Balance agent control with agency".

[i.16] LangChain: "Engineer reliable agents".

[i.17] Sumers, T.R., Yao, S., Narasimhan, K., Griffiths, T.L.: "Cognitive Architectures for Language Agents", March 2024.

[i.18] ETSI GS ZSM 007: "Zero-touch network and Service Management (ZSM); Terminology for concepts in ZSM".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI GS ZSM 007 [i.18] and the following apply:

autonomous agent: intent-driven, autonomous AI entity designed to perceive its environment, analyse information using the agent's existing knowledge, make decisions also based on this knowledge and execute actions to achieve predefined goals described in the intent with minimal to no human intervention, as well as continuously learning and adapting its behaviours based on the changing conditions in the environment

task: element of operational management that the ZSM framework aims to automate entirely

NOTE 1: A task is typically realized programmatically as an action, operation, or complex workflow performed by underlying management functions or management services.

NOTE 2: In the context of the present document, task represents a structured unit of work that one autonomous agent delegates to another. It provides a persistent, trackable, and stateful context for asynchronous long-running collaborations between agents, ensuring clarity, disambiguation, and accountability.

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GS ZSM 007 [i.18] and the following apply:

CLA	Closed-Loop Automation
IME	Intent Management Entity
LLM	Large Language Model
MAS	Multi-Agent System
MnS	Management Service
OASF	Open Agent Schema Framework
URI	Unified Resource Identification

4 Multi-Agent Systems

4.1 Introduction to Agents

4.1.1 Concept of Agent

In its simplest form, an agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators in order to achieve a goal [i.1].

In the context of zero-touch network and service management, the definition of the autonomous agent can be found in clause 3.1. This definition emphasizes a number of capabilities already described for the Closed-Loop Automation (CLA) [i.2], with the implication that agents realize one or multiple close control loops with the combination and chaining of management services (monitoring, analytics, decision-making and execution). Consequently, agents are able to constantly monitor and assess the state of the network and services and take corrective actions when the goals of the agent are not fulfilled. Moreover, this definition highlights that agents are capable of interpreting declarative intents and autonomously selecting actions and decisions to fulfil the goals described in the intents. The definition also stresses the importance of self-learning and the adaptability of the agent based on the changing conditions in the environment.

The focus of the present document is to discuss the interoperability between these agents within the scope of ZSM framework, including agents running within the same domain and in different domains. The interactions between agents and external entities such as human operators are also considered, as this is important for the gradual increase of autonomy offered by these autonomous agents.

4.1.2 Agent Taxonomy

There are multiple ways to classify agents, based on their capabilities, functionality and other characteristics. Possible classifications include:

- By Decision-Maker:
  - Human as Decision-Maker. These agents (also known as Copilots) function primarily as augmentation tools for human operators, providing decision support, insights, and execution capabilities while maintaining the human as the primary control entity.
  - Agent as Decision-Maker. These agents possess decision-making capabilities and can independently analyse situations and formulate response strategies. There could be further classified into:
    - Supervised autonomous agents: These agents incorporate human or other agent oversight checkpoints where operator verification, adjustment, or explicit approval is required before critical actions are implemented.
    - Fully autonomous agents: These advanced agents operate with complete operational independence, continuously monitoring environments, analysing conditions, making decisions, and executing actions without human intervention. They leverage comprehensive knowledge bases and adaptive learning mechanisms to handle complex scenarios independently, enabling true zero-touch operations. Fully autonomous agents could also trigger "actions" to be executed by supervised autonomous agents.
- By functionality: According to the function they perform in the Management Domain (MD) such as managing faults and network changes or optimizing the network. Examples could include: Fault Management Agent, Network Change Agent, Quality Optimization Agent, Energy Efficiency Optimization Agent and Network Deployment Agent.
- Based on the type of actions they perform, e.g. recommender agent (provides recommendations), analyser agent (analyses data), configuration agents (use apis to change configs in the network), etc.

4.2 Overview of Multi-Agent Systems

4.2.1 Communication in Multi-Agent Systems

A Multi-Agent System (MAS) is a system composed of multiple interacting, autonomous agents, each capable of perceiving its environment, making decisions, and taking actions to achieve individual or collective goals. These agents can work independently or collaboratively, often engaging in coordination, negotiation, or competition to solve complex problems or accomplish tasks.

Communication is a central element in a MAS, as it enables interaction, collaboration, and coordination among agents that operate autonomously. Through communication, agents exchange information about their environment, share knowledge, advertise capabilities, negotiate fulfilment of tasks, and coordinate actions to achieve collective results. In this context, communication does not merely serve as a means to transmit data but functions as the foundation for distributed intelligence. It allows agents to establish shared situational awareness, align operational objectives, and maintain coherence in decision-making.

4.2.2 Industry References

The AI agent ecosystem has experienced significant momentum in recent months with multiple industry initiatives launching open standards for multi-agent communications. The AGNTCY coalition [i.4], established in early 2025 has introduced a comprehensive framework for agent interoperability. This open-source initiative develops standardized protocols enabling diverse AI agents to discover, communicate, and collaborate across platforms. Concurrently, Google released its Agent2Agent (A2A) protocol on April, 2025 [i.3], providing an alternative open standard for agent communication. Backed by over 50 major technology partners, A2A standardizes how autonomous agents discover and interact with one another using familiar web technologies like HTTP and JSON. The protocol enables agents to advertise their capabilities, exchange structured information, and coordinate actions regardless of their underlying frameworks or vendors.

These complementary initiatives reflect the industry's recognition that standardized agent communication protocols are essential for realizing the full potential of multi-agent systems in enterprise environments, where specialized AI agents often operate in isolation due to interoperability challenges.

4.3 Agents in the ZSM Framework

4.3.1 Agent Entity

As explained in clause 4.1, the agent entity realizes one or multiple closed loops and it is capable of interpreting declarative intents. The concepts of close loop and Intent Management Entity (IME) are fully described in ETSI GS ZSM 009-1 [i.2] and ETSI GR ZSM 011 [i.6] respectively. This is shown in Figure 1.

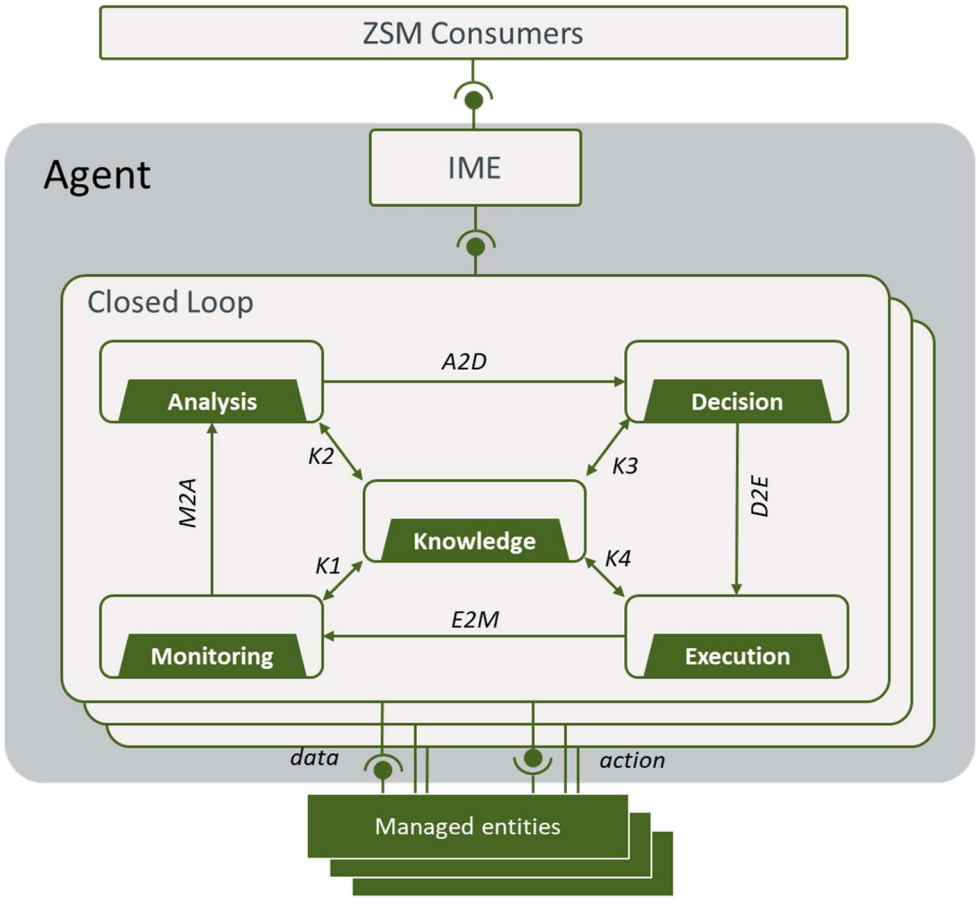

Figure 1: Agents in ZSM Framework

Figure 1 illustrates how the agent implements the CLA capability, which is a core component of the ZSM framework. Thus, the agent is capable of constantly monitor the network, analyse, make decisions and execute those decisions when the goals that were provided to the agent via the intent are not fulfilled.

As defined previously, the autonomous agent is continuously learning and adapting its behaviours to changing conditions in the environment. The concept of cognitive loops was introduced in ETSI GR ZSM 009-3 [i.7], where it was described that closed-loops may be logically partitioned into two interacting internal loops: a Cognitive Loop and an Operational Loop. The cognitive loop enables self-learning by analysing the outcomes of actions executed by the operational loop, and thus provides the agent with its learning and adaptive capabilities.

4.3.2 Multi-Agent Systems in ZSM

The agents in a MAS need to communicate and share information. The focus of the present document is to report on the standardization requirements for these agents to discover their capabilities, communicate with, and work alongside each other. These standardization requirements are among agents operating within the same or in different MDs within the ZSM Framework.

Figure 2 illustrates the architecture of the ZSM framework, focusing on the integration of agents across multiple Management Domains (MDs) and the role of cross-domain communication. It highlights how agents operate within individual domains and collaborate across domains to achieve end-to-end service automation. Figure 2 also highlights the use of an Agent Registry, where agents can register their capabilities. This registry facilitates capability discovery by enabling agents to identify and interact with neighbouring agents, allowing for efficient collaboration across domains.

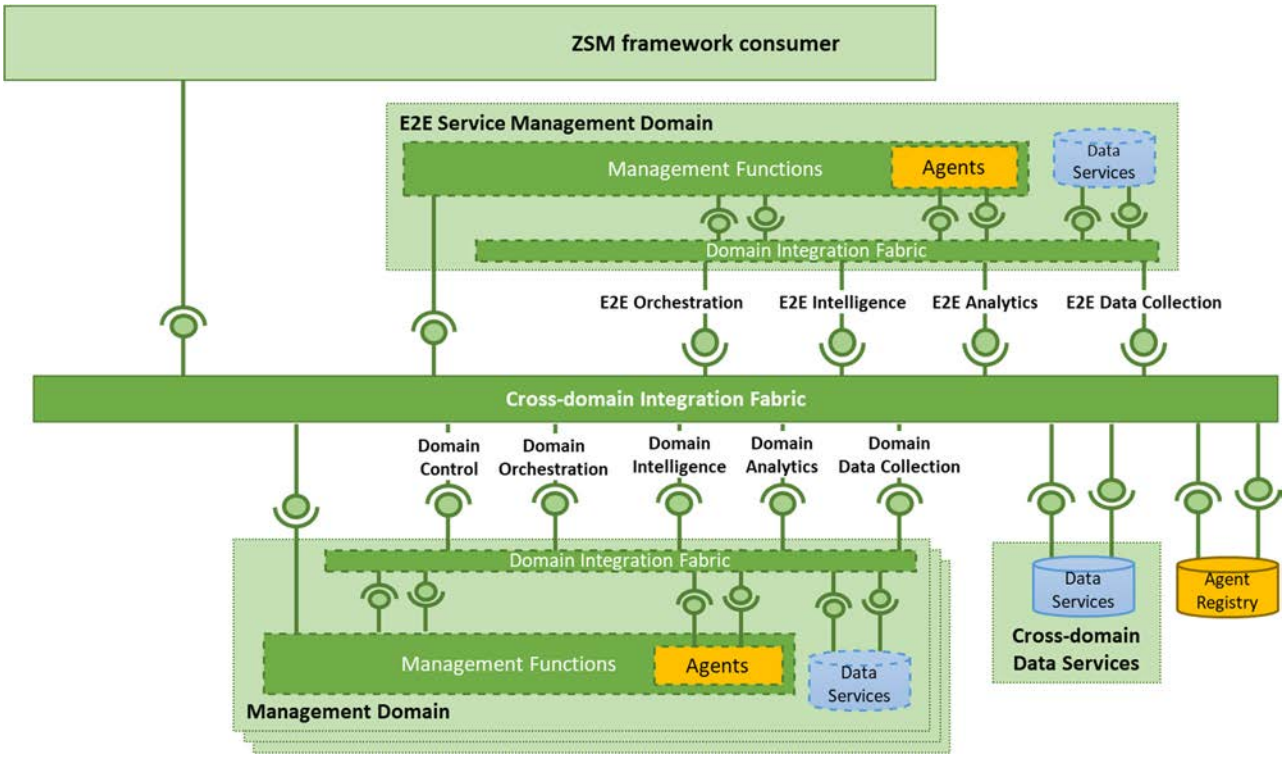

Figure 2: Agents in the ZSM Framework

NOTE: The agent registry is a management function that could be provided as part of the integration fabric, similar to how management service registration is handled, or as part of the cross-domain data services.

As show in Figure 2, the agent entity is the element within the ZSM framework providing the management services for agent communications. It produces management services to receive communication from other agents or authorized ZSM consumers and it consumes management services to send communication to other agents or ZSM entities.

5 Zero-touch automation scenarios using multi-agent systems

5.1 Agent Capabilities Registration and Discovery

5.1.1 Description

In order to enhance interaction and collaboration among agents, it should be possible for agents to register or advertise their capabilities so that they can be discovered by other agents.

Currently, there is no industry-wide consensus on a standardized method for describing agent capabilities:

- A2A [i.3]: Agents can advertise their capabilities using an "Agent Card" in JSON format, allowing the client agent to identify the best agent that can perform a task and leverage A2A to communicate with the remote agent.
- Agntcy [i.4]: Open Agent Schema Framework (OASF) provides a standard form in which the identity and capabilities of an agent can be described in a vendor and platform agnostic manner. It provides a common, interoperable agentic definition which can be used for describing and discovering agents and multi-agent software.

Similarly to ETSI GS ZSM 016 [i.5], where the IME profile was introduced, it is proposed the usage of an Agent Capability Profile that describes at least:

- The agent means of communication.
- The management domain of the agent.
- Agent capabilities.

5.1.2 Scenario details for centralized registration

A precondition of this use case is that there is an agent registry that can be accessed by authorized consumers (note that an authorized consumer may only have access to some of the agent details in the registry).

The present clause describes the detailed steps that an agent should follow to use the registration and discovery capabilities of the ZSM Framework:

1) An agent will use a registration/deregistration MnS for the registration/de-registration of its agent profile.
2) Once registered, the agent registry may optionally advertise the registered agents' capabilities.
3) An agent will use a discovery MnS for agents for the discovery of registered agents and the means to access them, supporting both synchronous and asynchronous communication mechanisms.

5.1.3 Scenario details for de-centralized agent advertisement

A precondition of this use case is that there is a mechanism for agents to advertise their existence to other authorized consumers.

The present clause describes the detailed steps that an agent may use to advertise their existence to other agents in the ZSM Framework:

1) The agent will use an advertisement capability (such as using a publish/subscribe communication mechanism) to inform other agents about its profile.
2) Another agent will use an agent discovery MnS to query additional information from the agent that advertised its existence if necessary.

5.1.4 Group-based Agents Registration and Discovery

In multi-agent environment, agents can register themselves with different functionality or capability. As described in clause 4.1.2, agents can be classified in several dimensions:

- By Decision-Maker, depending on the role of the agent.
- By functionality, assigning the agent to different functional scenario.
- By the type of actions, which correspond to specific capabilities the agent has.

When agents need to collaborate, discovering the correct target agents becomes challenging due to mismatching in agent descriptions. For example: Agent A from one MD domain registers itself based on functionality (e.g. as a Fault Management Agent), while Agent B from a different MD domain registers itself based on capability (e.g. as an Analyzer Agent). The mismatch of terminology and granularity makes it difficult for Agent A to discover Agent B in cross-domain fault handling collaboration task.

Therefore, the agent registration can be further expanded by including an Agent Grouping Mechanism to enhance the efficiency of registration and discovery among agents. For example, a registry owner may pre-define a set of "Agent Groups" and at registration time an agent may choose which groups it should belong to. An example would be a Functional Scenario Agent Group, with the possible values of Fault Management, Quality Optimization and Energy Efficiency to be selected at registration time.

Agent Groups can be created by different categories including by capability (e.g. "analytical capability"), by scenario (e.g. "Fault Diagnosis scenario") and by role ("Human Decision") in registry. Agents from different domains or vendors can choose any of these Agent Groups at registration time.

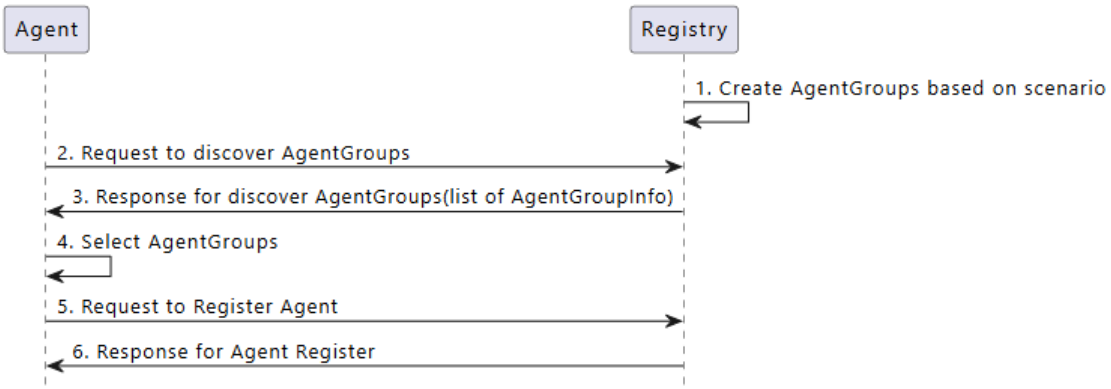

Figure 3: Basic Interaction Flow for Group based Registration

1) Registry pre-creates AgentGroups based on business scenarios or capabilities.

2) Agent sends discovery request to the Registry to retrieve AgentGroups information.

3) Registry responses to the Agent with the list of AgentGroups information.

4) Agent determines which AgentGroups to join based on the business scenarios or capabilities that are supported by both itself and the target groups, as reflected in the AgentGroups information. If there are no suitable AgentGroup the Agent can create a new one dynamically, and register it to Registry with the next Agent registration step.

5) Agent sends a registration request to the Registry, applying to join the AgentGroups selected. If the AgentGroup is dynamically created by the Agent, the Registry will first complete the registration of it.

6) Registry responses for agent register.

5.2 Cross-domain fault resolution using E2ES MD cross-domain agent

5.2.1 Description

This use case demonstrates how multiple agents collaborate within the ZSM framework, which supports agent capability registration and discovery as explained in the previous use case. The scenario focuses on the escalation of a fault detected in one MD, leading to the formation of a cross-domain investigation team of agents led by the E2ES MD agent to resolve the issue.

The actors for this use case include multiple MD fault agents which can detect faults in their own domain and perform additional analysis on the status of the network when requested and E2ES MD agent which coordinates activities across multiple MDs and forms teams for complex, cross-domain issue resolutions.

5.2.2 Cross-domain fault resolution with E2ES MD cross-domain agent

Preconditions

- All agents have registered their capabilities in a centralized agent registry using standardized profiles and the agent registry supports discovery of agent profiles and capabilities to ZSM authorized consumers (centralized scenario).
- Agents can advertise their existence and capabilities to other existing agents, and discovery mechanisms are in place for agents to enquire additional information from advertising agent (decentralized scenario).

The present clause describes the detailed steps that agents will execute:

1) Fault Detection by one MD agent: The fault agent of a MD, detects a persistent fault affecting its domain. After analysing the fault, the agent determines that the root cause may extend beyond its own domain.

NOTE 1: An example would be a wireless MD detecting a fault that affects the wireless service quality but the source of the fault is in a transmission MD which provides backhaul connectivity for the wireless MD.

2) Capability Discovery and Escalation: The agent from the MD that has the fault queries the agent registry (centralized use case) or other existing agents (decentralized use case) to discover agents capable of handling cross-domain issues. It then identifies a E2ES MD cross-domain agent as having the necessary coordination and investigation capabilities. The MD agent escalates the fault to the cross-domain E2ES MD agent, providing detailed context and diagnostic data.

3) Cross-Domain Agent Forms Investigation Team: The E2ES MD cross-domain agent receives the escalation and analyses the fault context. It queries the registry (centralized use case) or other existing agents (decentralized use case) to identify and select additional agents with relevant expertise based on their capabilities and it forms an investigation team.

NOTE 2: For the example of the wireless MD being affected by the transmission MD, the cross-domain agent will form a team comprising at least one agent from the wireless MD, one agent from the transmission MD and the E2ES MD cross-domain agent.

4) Collaborative Investigation and Resolution: Using the communication means specified in their capability profile, the E2ES MD cross-domain agent coordinates the investigation and assigns tasks to the rest of the agents in the investigation team. These tasks may include remediation actions across different domains.

NOTE 3: The task assignation process may include a negotiation stage where agents will agree about the task details and completion details.

5) Post-Resolution Actions: Once the fault is fully identified or resolved, the E2ES MD cross-domain agent provides feedback to all agents that were part of the investigation team.

5.3 Autonomous Agents Negotiation

5.3.1 Description

This use case describes how autonomous agents negotiate the assignment of tasks among themselves. Unlike static allocation models where tasks are pre-assigned based on predefined rules, negotiation enables agents to dynamically determine the most suitable executor for a given task.

The negotiation process allows agents to self-organize by factoring in agent capabilities, resource availability, performance history, costs, and priorities. Through structured communication, agents propose, bid, or decline tasks until an agreement is reached. This results in more efficient and adaptive task distribution, especially in environments subject to change, uncertainty, or resource constraints.

5.3.2 Task Negotiation process

Preconditions

- Agents are discoverable, and their capabilities can be dynamically queried.
- Agents can exchange information securely and reliably.
- Agents have defined decision models (e.g. utility functions, constraints, preferences) to guide negotiation.
- External inputs such as an operator intent are available to influence negotiation outcomes (e.g. cost thresholds, fairness criteria, service-level parameters).
- Agents should provide an interface for negotiation.

The detailed steps for this use case are the following:

1) Task Initiation: An initiating agent identifies a task or set of tasks that require execution.

2) Candidate Identification: Using agent capability discovery, the initiating agent identifies potential agents capable of executing the task.

3) Negotiation: Initiating agent provide task details to potential agents, who in return communicate back their willingness, constraints, or costs for executing the task. The initiating agent may compare proposals based on capability fit, efficiency, cost, reliability, or other defined metrics. There could be several rounds of communication during the negotiation phase.

4) Agreement Formation: The selected agent formally accepts execution responsibilities, including agreed parameters such as time, quality, or resource use.

5) Task Execution: The task is executed by the assigned agent, with progress updates provided as agreed.

6) Completion and Confirmation: Upon success (or failure), results are confirmed to the initiating agent.

7) Post-process Analysis: The negotiation and execution records may be analysed for continuous improvement (e.g. learning agent preferences, negotiation strategies or improving fairness).

5.3.3 Task Execution Information Phase

In multi-agent systems, agents resolve complex business scenario problems through task mechanisms. For example, an OSS Agent can assign a cross-domain task (e.g. Base Station fault diagnosis) to domain-specific agents (e.g. IP Fault Agent), which will accomplish the task based on its autonomous capability.

However, during the task execution, the domain-specific agent may encounter execution abnormal states due to insufficient contextual information. Taking base station fault diagnosis task as an example, with the internal reasoning using domain-specific Large Language Models (LLMs), the IP Fault Agent detects that for the task progression, there is a lack of device configuration and traffic log information from the wireless domain, leading to the task entering a blocking state or failure state.

To resolve the abnormal states problem encountered during task execution, there should be a task execution information phase for the multi-agent negotiation mechanism.

Preconditions: Assuming the same preconditions as in clause 5.3.2 and adding:

- Agent Communication Message Extension:

  Augment attributes of the agent message model to include information negotiation request:

  - abnormalStateInfo:
    - abnormalState: Indicates the abnormal state of the task (e.g. blocking state).
    - stateAnalysis: Indicates the reason causing the task to be in an abnormal state.
  - requirementContent: Indicates the required information needed to handle the abnormal state of the task, including the identification of such information.

  Augment attributes of the agent message model for information negotiation response:

  - InfoContent: Contains the information needed to handle the abnormal state of the task.

The detailed steps for this use case are the following:

1) OSS Agent receives a fault report from a specific domain (e.g. wireless domain), it determines it is a cross-domain fault that the source domain is unable to handle, and assigns a cross-domain fault diagnosis task to the IP Fault Agent.

2) During the task execution process, IP Fault Agent detects an abnormal state (e.g. blocking state) of the fault diagnosis task, and using local reasoning algorithms, the agent outputs the "abnormalStateInfo" of the task, which contains the abnormalState info and the stateAnalysis info, and based on the "abnormalStateInfo" of the task, the agent outputs the "requirementContent" with the further reasoning algorithms.

3) IP Fault Agent sends negotiation request to OSS Agent, which includes attributes: "taskId", "abnormalStateInfo" of task, and "requirementContent".

4) If the OSS Agent has all the required information indicated by the "requirementContent", it prepares it, and additionally, through local reasoning algorithms, a further auxiliary information may be outputted by OSS Agent based on the "abnormalStateInfo" of task. If the OSS Agent does not have all the required information, it will request the missing information from the Wireless agent before preparing it.

5) OSS Agent sends the response of negotiation to IP Fault Agent, which includes attributes: "taskId", "infoContent". The attribute "infoContent" contains not only the required information indicated by the "requirementContent", but also the additional auxiliary information from OSSAgent.

6) IP Fault Agent handles the abnormal state of the current task with the "infoContent".

7) IP domain fault Agent completes the cross-domain fault diagnosis task.

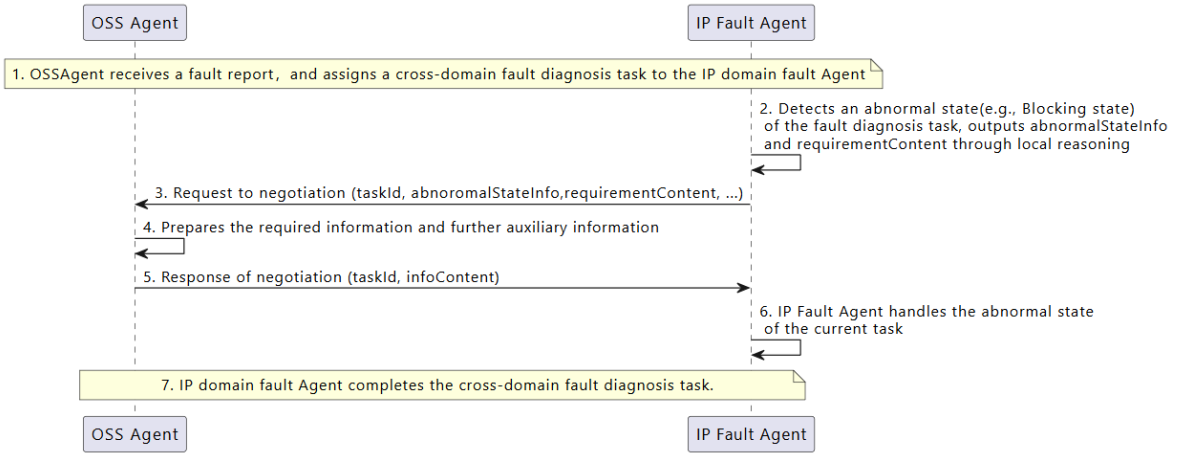

Figure 4: Basic Interaction Flow for Task Execution Information phase

5.4 Agent-based Dynamic Workflow

5.4.1 Description

Agent-based dynamic workflows are processes where autonomous agents make decisions, coordinate, and execute tasks independently with minimal human intervention. Unlike basic automation, which follows static predefined rules, agents in dynamic workflows perceive their environment, analyse data in real time, and dynamically adapt their actions to achieve specific network goals.

The actors for this use case include an autonomous agent in the role of workflow coordinator from an MD or E2ES MD and additional autonomous agents (selected based on dynamic capability discovery) that can perform tasks delegated from the workflow coordinator agent. The role of the workflow coordinator is not mandatory, but used in this use case to ensure agent tasks are properly sequenced without overlapping.

5.4.2 Collaboration and coordination

A key distinction in multi-agent systems is between coordination and collaboration:

- In collaboration, agents work together closely and interact as peers, sharing information and jointly tackling tasks to achieve a common objective. Collaboration often involves flexible real-time negotiation, dynamic task allocation, and synthesis of results as a collective effort.

- In contrast, coordination describes agents working on separate tasks in pursuit of a shared outcome, with their efforts structured normally by a coordinator agent. The coordinator assigns responsibilities, manages dependencies, and sequences actions to avoid conflicts or overlaps, enabling agents to operate independently yet remain aligned toward the overall workflow goal.

5.4.3 Dynamic workflow using agents and a workflow coordinator

Preconditions

- Agents and their capabilities can be discovered (either using the centralized scenario or decentralized scenario).
- Agents can communicate among themselves.

The present clause describes the detailed steps that agents will execute:

1) Workflow Trigger: A workflow coordinator agent in an MD or E2ES MD identifies a condition, based on internal states or external inputs such as an operator intent, that requires a workflow to be executed.
2) Task Identification and Assignment: The workflow coordinator agent breaks down the workflow into discrete tasks (single or multiple, potentially with dependencies) and for each task:
    a) Using the agent capability discovery, the coordinator determines which agents are capable of executing the task(s).
    b) Assigns the task that needs to be completed to the identified agent, negotiating with the agent the completion details. If the negotiation is successful the task is delegated, otherwise the coordinator attempts to assign the task to a different agent. If no additional agents are available the workflow completion fails (escalating to human operator for example).
3) Task Execution: As tasks are completed, and task completion reports are provided by the executing agent, the coordinator may uncover further dependent or subsequent tasks.
    a) The process in Step 2 repeats for new tasks, allowing for conditional branching - where some tasks depend on outputs from prior steps.
    b) The coordinator ensures tasks are executed in the correct sequence or in parallel as workflow logic demands, handling synchronization and preventing conflicts.
4) Workflow Completion: Once all tasks (including dynamically added or branched ones) have been completed successfully, the coordinator finalizes the workflow.
5) Post-processing: Post-process activities may include logging, performance analysis, and learning updates to improve future workflow agility.

5.5 Terminology Alignment between Agents

5.5.1 Description

In cross-domain multi-agent scenario (e.g. fault diagnosis), the agents using diverse LLMs will encounter semantic misinterpretations during task collaboration. When agents exchange messages (e.g. task assignment request, information request), the receiving agent may fail to accurately understand some domain-specific terms used by the sender agent, for examples:

- Term "CIR": means "Carrying Information Rate" for sender agent (using LLM A), means "committed information rate" for receiver agent (using LLM B).
- Term "RAN": means "Radio Access Network" for sender agent (from RAN domain), means "Remote Area Network" for receiver agent (from IP domain).

This ambiguity will reduce agent message accuracy and task success rates, especially for telecommunication-specific terms (e.g. "CIR", "RAN") that carry different meanings across domains.

5.5.2 Cross-Domain Terminology Alignment for Agent Communication

Preconditions: A potential solution for this challenge is to augment agent messages with attributes containing:

- **Term Definitions**: key-value pairs clarifying terms (e.g. termExplain: [{"CIR": "Committed Information Rate"}]), the "key" represents the term, and the "value" represents its explanation, the first agent can send key-value pairs to a second agent via the attribute.
- **Term File Info**: includes a file (e.g. json file, model file) or a URI pointing to the file where the file stores key-value pairs and the corresponding information for all terms.

The basic Interaction Flow is as follows:

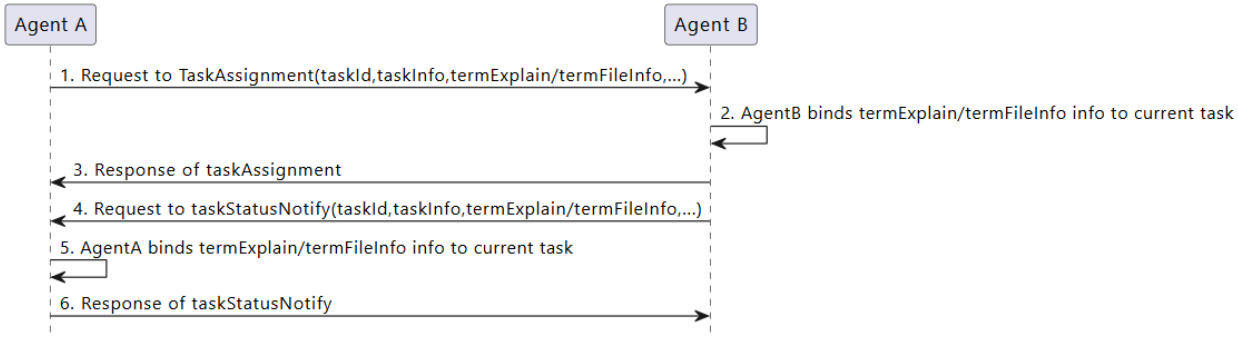

Figure 5: Terminology Alignment Interaction Flow

1) Agent A obtains key-value pairs based on its domain knowledge, and sends a TaskAssign request to Agent B, which includes "taskId", "taskInfo", "termExplain"/"termFileInfo" attributes. The attributes can be presented in natural language or structured format and the "taskInfo" contains specific information (e.g. taskobject, taskgoal, taskorders) of the task.
2) Agent B binds "termExplain"/"termFileInfo" to the task identified by "taskId", and interprets the "keys" contained in the "taskInfo" with the mapped values from the "termExplain"/"termFileInfo".
3) Agent B responds to Agent A accepting the task.
4) During the task processing, Agent B obtains key-value pairs based on its domain knowledge, and sends taskStatusNotify request to Agent A, which include "taskId", "taskInfo", "termExplain"/"termFileInfo" attributes.
5) Agent A binds "termExplain"/"termFileInfo" info to the task identified by "taskId", and interprets the "keys" in the same way as described in step 2.
6) Agent A responds to Agent B acknowledging the notification.

6 Importance of Communication in Multi-Agent Systems

6.1 Agent Communication Models

6.1.1 Introduction to Communication Models

In Multi-Agent Systems (MAS), agents need to share information to achieve common goals, resolve conflicts, or divide tasks. Communication models define how messages are transmitted, how agents discover each other, and how coordination emerges. Broadly, these models fall into three categories:

- Centralized: a central coordinator or message broker manages all communications.

- Decentralized: Agents interact directly without centralized control.
- Hybrid: Combines both centralized and decentralized components.

Two models are foundational and widely applicable to agentic systems designed for autonomy and resilience: peer-to peer and publish/subscribe. These communication models are based on those described in ETSI GS ZSM 002 [i.8], Annex B.

6.1.2 Peer-to-Peer Communication

In the Peer-to-Peer (P2P) communication model, each agent acts as both a sender and receiver of messages, communicating directly with specific peers. This model requires these capabilities:

- a discovery capability: before the sender can be able to send a message, it needs to discover the communication endpoint of the receiver, the platform or the agents themselves can maintain a dynamic registry of known peers through service discovery protocols or gossip mechanisms.
- a messaging capability: to exchange messages directly between agents using standardized transport protocols.
- a security verification capability: when an agent starts a new communication, an authorization or a security policy could be checked before the receiver receives/accepts the message.

This is a selective communication because Agents target specific peers based on context, roles, or prior relationships. It can either be asynchronous (delayed) or synchronous (real-time).

Advantages:

- It is scalable with minimal central infrastructure.
- It enables fine-grained (point to point) coordination and negotiation.
- It supports trust-building between agents.

Challenges:

- It introduces the complexity of the peer discovery and topology maintenance.
- It requires robust conflict resolution strategies.

6.1.3 Publish/Subscribe Communication

In the publish/subscribe communication model, a message is sent by one agent and received by one or more agents in the network or group that are subscribed to a specific topic/channel. This model requires these elements:

- a publisher: the agent initiating the communication emits a message tagged with metadata (e.g. type, timestamp, priority);
- a network: it utilizes shared channels (e.g. pub/sub bus, multicast UDP, message queues like Kafka or NATS);
- a receiver logic: receivers are all the agents subscribed to the topic/channel which receive the message and decide how to act upon it.

This is a loose coupled model: the sender is not aware of (and does not need to know) who are the receivers of its message. It is a one-to-many messaging, e.g. a single action of the sender generates many copies of the message one for each subscriber. This is a stateless communication mainly used for notifications, updates or environment changes.

Advantages:

- It is simple and efficient for group awareness.
- It ensures uniform dissemination of critical data.
- It has a low coordination overhead.

Challenges:

- It has a potential information overload and redundant processing.
- The scalability is limited in high-frequency systems.
- The lack of message targeting may reduce efficiency.

6.1.4 Multicast-Group Communication

It is a particular use case for a Publish/Subscribe communication model that is implemented when within a group there is the need that one sender creates a message that is received by all the receivers that are in that group, but only few of the receivers are expected to reply to the sender. In any case, all the members of the group are aware of all the replies too. In addition to the already mentioned elements, this model requires also:

- an addressee list: the agent initiating the communication, includes a list of addressees within the message metadata that are the receivers who are expected to reply within the same topic/channel;
- a reply logic: receivers that are in the addressee list are in charge to consider the message as a request to be replied and will act accordingly. In their replies, the original sender will be the only receiver in the addressee list.

Advantages:

- It allows to share information (requests/responses) within a group overtaking the limits of the peer-to-peer communication.

Challenges:

- It has an increased complexity due to the integration of the request/response management as part of the message exchange.

6.2 Agent Interaction and Coordination

6.2.1 Identity Management for Agents

Agent identity management is a foundational element establishing trust, security, and efficiency within interoperable multi-agent systems. It ensures that agents can reliably register, advertise, and discover one another's identities and capabilities, thereby fostering trusted interactions in both centralized and decentralized environments.

These agents frequently act on behalf of human users or other systems and agents and require their identities to be short-lived, issued dynamically at runtime with narrowly scoped, purpose-bound permissions rather than pre-provisioned, static accounts as traditional identity management systems may do.

Some of the basic security and identity management concepts were reviewed in ETSI GS ZSM 014 [i.9]. However, identity management in the ZSM framework for autonomous agents should evolve to provide task-specific, time-bound, and adaptive controls.

Additionally, auditing poses unique challenges since agent actions occur rapidly and can be across management domains. Thus, the ZSM framework should additionally support security log collection and audit report generation within the same MD or across multiple MDs.

Best practice for identity management:

- Utilize mechanisms for ephemeral identity management, issuing short-lived credentials that expire and are revoked automatically.
- Employ task-specific identity management to grant narrowly scoped, purpose-specific permissions required for an agent's assigned tasks.
- Support capabilities to generate security/audit reports based on security logs collected from MD/E2ES MD log services.

6.2.2 Task Delegation and Escalation

Autonomous agents are transforming the management of network and services by enabling intelligent, distributed decision-making and automation [i.10], [i.11]. As these agents proliferate, their ability to interact, coordinate, and resolve conflicts becomes critical for achieving robust, scalable, and reliable network operations.

Task delegation and escalation refers to the process by which an autonomous agent assigns specific tasks or subtasks to other agents, in the same or different MD, based on their capabilities, current workload, and network context. In telecom networks, this is essential for:

- Dynamic resource allocation: Agents can delegate tasks such as traffic routing, fault diagnosis, or network slicing to specialized agents. For example, an E2ES MD agent can request resources to be allocated by a MD agent.
- Hierarchical control: Multi-level agent architectures allow high-level agents to delegate operational tasks to lower-level agents, improving scalability and efficiency.
- Adaptive escalation: When an agent encounters a task beyond its scope or authority, it escalates the issue to a more capable or supervisory agent, ensuring timely and effective resolution.

Recent research highlights several enabling technologies and frameworks:

- LLM agents demonstrate advanced reasoning and autonomous decision-making, allowing for effective delegation and escalation in complex wireless network scenarios.
- Collaborative agent frameworks such as CrewAI [i.14] and LangGraph [i.15] provide structured approaches for task assignment, monitoring, and escalation, ensuring that tasks are handled by the most suitable agents and that unresolved issues are promptly escalated.
- Agent collaboration architectures emphasize coordinated agent-to-agent communication and efficient memory management, supporting seamless delegation and escalation across distributed network environments.

Best practices for task delegation and escalation:

- Use capability-based task assignment to match tasks to agent expertise.
- Implement escalation protocols for handling exceptions, failures, or tasks requiring higher-level authorization.
- Implement interfaces to monitor the progress of delegated/escalated tasks and a way to provide feedback loops to optimize future delegation strategies.

6.2.3 Negotiation and Collaboration

Negotiation and collaboration are fundamental to enabling autonomous agents to achieve shared goals, resolve resource contention, and optimize overall network performance in telecom environments [i.12], [i.13].

Negotiation involves agents engaging in structured exchanges to reach mutually beneficial agreements, particularly when resources are limited or objectives overlap. Modern negotiation strategies for autonomous agents integrate traditional negotiation theory with AI-specific methods, accounting for the unique characteristics of autonomous systems. These include:

- Automated negotiating and consensus-seeking protocols.
- Multi-round negotiation mechanisms that enable agents to adapt strategies based on network context and historical outcomes.
- Formal models and benchmarks for evaluating agent negotiation skills in asymmetric and incomplete information scenarios.

Collaboration refers to agents working together to achieve a goal, to accomplish complex tasks, share information, and pursue network-wide objectives. Without collaboration, autonomous agent would be limited to their internal workflow and could not be able to always complete all the assigned tasks. Advances in AI-native networks have expanded device intelligence, fostering federation and collaboration among distributed autonomous agents. Key aspects include:

- Goal-oriented and semantic communication frameworks, that enable agents to coordinate actions and share intent efficiently.
- Collaborative learning and knowledge transfer mechanisms, allowing agents to learn from each other and improve collective performance.
- Incentive structures and social contracts, inspired by decentralized technologies, to ensure transparency, accountability, and trust among agents and human stakeholders.

Best practices for negotiation and collaboration:

- Design agents with negotiation protocols that can handle dynamic, multi-agent environments.
- Foster collaboration through shared goals, transparent communication, and incentive alignment.
- Leverage collaborative frameworks to enable distributed problem-solving and learning.

6.2.4 Conflict Resolution

As autonomous agents interact within shared network environments, conflicts may arise due to overlapping responsibilities, resource contention, or divergent objectives. Effective conflict resolution mechanisms are essential for maintaining network stability and performance.

Types of conflicts:

- Resource conflicts: Multiple agents competing for limited bandwidth, compute resources, or access to network elements.
- Policy conflicts: Agents following different or outdated policies, leading to inconsistent actions.
- Goal conflicts: Agents pursuing intents with objectives that may be incompatible or mutually exclusive.

Recent advances in conflict resolution include:

- Negotiation protocols: Agents employ negotiation or consensus-seeking algorithms to resolve resource and policy conflicts, ensuring fair and efficient outcomes.
- Hierarchical arbitration: Supervisory agents or arbitration modules mediate disputes and enforce global policies, especially in large-scale or hierarchical agent architectures.
- Explainable AI (XAI): The integration of explainable reasoning allows agents to justify their decisions, improving transparency and facilitating conflict resolution by making the rationale behind actions understandable to both humans and other agents.
- Formal verification and security: Robust agent architectures incorporate formal verification methods and secure data handling to prevent and resolve conflicts arising from adversarial actions or misconfigurations.

Best practices for conflict resolution:

- Design agents with negotiation and arbitration capabilities.
- Ensure clear policy dissemination and synchronization among agents.
- Integrate explainability and transparency to support both automated and human-in-the-loop conflict resolution.

6.3 Agents Organization

6.3.1 Agent Capability Profile

The organization of autonomous agents within communication networks is foundational to achieving scalable, secure, and intelligent network management. Effective organization ensures that agents can be efficiently discovered, their capabilities accurately profiled, and their interactions governed by robust frameworks. The present clause explores the latest research and best practices for agent capability profiling and the establishment of agent registries in communication environments [i.3], [i.4].

An Agent Capability Profile defines the functional, operational, and contextual attributes of an autonomous agent. This profile is essential for enabling dynamic task assignment, interoperability, and secure operation within complex communication networks.

Key Components of an Agent Capability Profile:

- Functional Capabilities: specifies the tasks and services the agent can perform (e.g. network slicing, fault diagnosis, resource allocation).
- Operational Context: details the environments or network domains where the agent is effective (e.g. 5G RAN, optical core, edge computing).
- Performance Metrics: includes quantitative measures such as latency, throughput, and reliability under various scenarios.
- Security and Trust Attributes: describes authentication methods, compliance with security standards, and trust anchors.
- Learning and Adaptation: indicates whether the agent supports online learning, knowledge sharing, or collaborative adaptation with other agents.

Best Practices for capability profiles:

- Standardize capability profiles using machine-readable formats (e.g. JSON-LD, ontologies) for interoperability.
- Continuously update profiles with real-time performance data and learning outcomes.
- Integrate security attributes directly into capability profiles to support zero-trust architectures.

6.3.2 Agent Registry: Registration and Discovery of Agents

An Agent Registry is a centralized or distributed system that manages the registration, discovery, and lifecycle of autonomous agents in communication networks. It acts as the backbone for agent orchestration, enabling dynamic composition and collaboration.

Core Functions of an Agent Registry:

- Registration: Agents submit their capability profiles, identity credentials, and operational constraints to the registry upon onboarding.
- Discovery: Network functions or other agents can query the registry to find agents matching specific capabilities, locations.
- Authentication and Authorization: The registry verifies agent identities and enforces access control policies.
- Lifecycle Management: Tracks agent status, updates, and de-registration.

Best Practices for agent registry:

- Employ telecommunication-grade identity and credential management for secure registration and discovery.
- Support federated and decentralized registry architectures to enhance scalability and resilience.

- Enable semantic and context-aware discovery queries for efficient agent selection in dynamic environments.
- Ensure auditable and transparent registration processes to foster trust and compliance with regulatory standards.

6.4 Knowledge Sharing between Agents

6.4.1 Key Concepts

Knowledge sharing among autonomous agents is a cornerstone for achieving intelligent, adaptive, and resilient telecom networks. As networks evolve toward 6G and beyond, the ability of agents to efficiently exchange, transfer, and reuse knowledge becomes essential for dynamic resource management, collaborative problem-solving, and continuous improvement of network operations.

Key Concepts in Agent Knowledge Sharing:

- Knowledge Transfer: the process by which agents share learned models, experiences, or insights to accelerate learning and improve decision-making across the network.
- Collaborative Reasoning: agents work together, combining their knowledge bases to solve complex tasks that exceed the capabilities of any single agent.
- Semantic Communication: moving beyond raw data exchange, agents share information in a goal-oriented, context-aware manner, enabling more efficient and meaningful interactions.

Within the ZSM framework, agents share knowledge by producing management services which are consumed by other agents. Using these management services, agents can share knowledge directly, when these agents exchange trained models or policy parameters, or share historical observations or experiences enabling other agents to learn from past successes and failures. Additionally, agents can interact together to achieve collaborative learning so that they evolve their capabilities through shared learning experiences.

6.4.2 Memory Structures in Autonomous Agents

Frameworks like Langchain [i.16] and research papers like Cognitive Architectures for Language Agents [i.17] have emphasized the importance of agentic memory structures for effective knowledge transfer and collaboration.

Agentic memory can be categorized into two main types: working (short-term) memory, and long-term memory and the latter can be further categorized into episodic memory, semantic memory, and procedural memory. Each type plays a distinctive role in facilitating knowledge sharing.

Short-term memory:

- **Working Memory**: serves as short-term, task-specific storage that enables agents to temporarily hold information relevant to ongoing interactions or collective goals. When agents share knowledge, working memory allows them to synchronize context, exchange intermediate results, and coordinate their current actions; a critical prerequisite for real-time collaboration.

Long-term memory:

- **Episodic Memory**: retains records of past interactions and events; specific episodes that have occurred over time. By sharing episodic memories, agents can inform others about previous experiences, lessons learned, and context-dependent outcomes. This enables a group of agents to avoid redundancies, replicate successful strategies, and recover from collective failures more effectively.
- **Semantic Memory**: comprises structured factual knowledge, concepts, and generalized understanding. Sharing semantic memory among agents means exchanging definitions, ontologies, rules, and domain knowledge. This creates a common knowledge base that agents can use to interpret messages, resolve ambiguity, and accelerate project onboarding in multi-agent systems.

- **Procedural Memory**: encapsulates learned skills, routines, and action policies; that is how to perform tasks and respond to situations. By sharing procedural memories, agents propagate best practices, optimized routines, and adaptive policies. This way, newly instantiated or retrained agents can quickly acquire operational proficiency without starting from scratch.

This structured approach helps agents maintain context awareness, prevent redundant work, and foster innovative solutions by building upon the experiences and knowledge of their peers. Integrating different types of agentic memory into knowledge sharing promotes consistency by maintaining shared context and reducing miscommunication and errors, and enhances efficiency by minimizing redundant computations and repeated learning through collective experience.

7 Potential new ZSM Capabilities to support Multi-Agent Systems

7.1 General Capabilities

7.1.1 Agent Management Capabilities

Capability-7.1.1-1: The ZSM framework should support the capability to create agent instances.

Capability-7.1.1-2: The ZSM framework should support the capability to modify agent instances.

Capability-7.1.1-3: The ZSM framework should support the capability to delete agent instances.

Capability-7.1.1-4: The ZSM framework should support the capability to activate an agent in a multi-agent system.

NOTE 1: Based on the ZSM framework's concepts applied to closed loops and IMEs, the lifecycle of agents could be separated into agent instantiation and agent activation, where instantiation is the process of creating the entity and preparing it for use, while activation is the process of enabling its operation to achieve its goals.

Capability-7.1.1-5: The ZSM framework should support the capability to deactivate an agent in a multi-agent system.

NOTE 2: The lifecycle of agents could be separated into agent deletion and agent deactivation, where deletion is the process of removing the entity from existence, while deactivation is the process of temporally rendering the entity inoperable.

Capability-7.1.1-6: The ZSM framework should support the capability to scale agents within a multi-agent system.

Capability-7.1.1-7: The ZSM framework should support communication between agents which can include hierarchical structured agents in a multi-agent system.

Capability-7.1.1-8: The ZSM framework should support the capability for agents work with a centralised global knowledge data base in a multi-agent system.

Capability-7.1.1-9: The ZSM framework should support the capability for agents work without a centralised global knowledge data base in a multi-agent system.

Capability-7.1.1-10: The ZSM framework should support the capability for agents to identify the deviance from their assigned intents in a multi-agent system.

Capability-7.1.1-11: The ZSM framework should support the capability for agents to mitigate or resolved the detected deviance from their assigned intents in a multi-agent system.

7.1.2 Agent Capability Profile Capabilities

From the use case described in clause 5.1 and from clause 6.3.1 of the present document:

Capability-7.1.2-1: The ZSM framework should provide a standardized mechanism for the definition of Agent Capability Profiles using machine-readable formats (e.g. JSON-LD, ontologies) for interoperability.

Capability-7.1.2-2: The ZSM framework should provide a mechanism to continuously update profiles with real-time performance data and learning outcomes.

Capability-7.1.2-3: The ZSM framework should integrate security attributes directly into capability profiles to support zero-trust architectures.

7.1.3 Registration and Discovery Capabilities

From the use case described in clause 5.1 and from clause 6.3.2 of the present document:

Capability-7.1.3-1: The ZSM framework should support an agent registry or the capability for agents to advertise their existence to other agents.

Capability-7.1.3-2: The ZSM framework should support the capability for the registration and de-registration of agent profiles, if an agent registry is provided.

Capability-7.1.3-3: The ZSM framework should support the capability for the discovery of agent profiles, if an agent registry is provided.

Capability-7.1.3-4: The ZSM framework should support the capability for the agent registry to advertise agent registration events, if an agent registry is provided.

Capability-7.1.3-5: The ZSM framework should support pre-defined /dynamic Agent Groups creation by business scenario or capability in the agent registry.

Capability-7.1.3-6: The ZSM framework should support Group-based Agents Registration and Discovery mechanism.

Capability-7.1.3-7: The ZSM framework should employ telecommunication-grade identity and credential management for secure registration and discovery

Capability-7.1.3-8: The ZSM framework should support federated and decentralized registry architectures to enhance scalability and resilience.

Capability-7.1.3-9: The ZSM framework should support semantic and context-aware discovery queries for efficient agent selection in dynamic environments.

Capability-7.1.3-10: The ZSM framework should support auditable and transparent registration processes to foster trust and compliance with regulatory standards.

7.1.4 Identity Management Capabilities

From clause 6.2.1 of the present document:

Capability-7.1.4-1: The ZSM framework should support the capability of dynamic identity management (e.g. create, read, update and delete identity) for autonomous agents.

Capability-7.1.4-2: The ZSM framework should support the capability of ephemeral identity management, issuing short-lived credentials that expire and are revoked automatically.

Capability-7.1.4-3: The ZSM framework should support the capability of task-specific identity management, granting narrowly scoped, purpose-specific permissions.

Capability-7.1.4-4: The ZSM framework should support the capability to generate security/audit reports based on security logs collected from MD/E2ES MD log services tracking all the actions across agent identity transactions.

7.1.5 Interaction Capabilities

From clauses 6.2.2, 6.2.3 and 6.2.4 of the present document:

Capability-7.1.5-1: The ZSM framework should support the collaboration between several agents to perform tasks.

Capability-7.1.5-2: The ZSM framework should support the ability of agents to negotiate among themselves in a multi-agent system.

Capability-7.1.5-3: The ZSM framework should support the communication between several agents to exchange data to complete tasks.

Capability-7.1.5-4: The ZSM framework should support the capability to exchange task-related data between agents in a multi-agent system.

Capability-7.1.5-5: The ZSM framework should support the coordination between several agents, typically under the direction of a coordinator agent, to perform tasks and to solve problems.

Capability-7.1.5-6: The ZSM framework should support resolution or mitigation of conflicts that may arise among agents that are working together on a task.

Capability-7.1.5-7: The ZSM framework should support the ability of agents to include a task execution information phase in the multi-agent negotiation mechanism.

Capability-7.1.5-8: The ZSM framework should support escalation protocols for handling exceptions, failures, or tasks requiring higher-level authorization.

Capability-7.1.5-9: The ZSM framework should support monitoring the progress of delegated/escalated tasks.

Capability-7.1.5-10: The ZSM framework should integrate explainability and transparency capabilities to support both automated and human-in-the-loop conflict resolution.

7.1.6 Knowledge Sharing Capabilities

From clause 6.4 of the present document:

Capability-7.1.6-1: The ZSM framework should support the capability for agents to agents exchange trained models or policy parameters allowing other agents to benefit from localized learning without redundant training.

Capability-7.1.6-2: The ZSM framework should support the capability for agents to share historical observations or experiences, enabling others to learn from past successes and failures.

Capability-7.1.6-3: The ZSM framework should support the capability for agents to interact in order to achieve collaborative learning, discover complementary skills from other agents and evolve their capabilities through shared learning experiences.

Capability-7.1.6-4: The ZSM framework should support the capability for agents to use intent-based and semantic communication allowing agents to interpret and act upon shared knowledge based on network objectives rather than just exchanging raw data.

Capability-7.1.6-5: The ZSM framework should support the capability for agents to share working memory allowing them to synchronize contextual information, exchange intermediate outcomes, and coordinate their ongoing actions during collaborative tasks.

Capability-7.1.6-6: The ZSM framework should support the capability for agents to share episodic memories enabling them to communicate previous experiences, share lessons learned, and communicate context-dependent outcomes to other agents.

Capability-7.1.6-7: The ZSM framework should support the capability for agents to share semantic memory, including definitions, ontologies, rules, and domain knowledge, to establish a common knowledge base that supports consistent message interpretation, ambiguity resolution, and faster collaboration.

Capability-7.1.6-8: The ZSM framework should support the capability for agents to share procedural memories, allowing best practices, optimized routines, and adaptive policies to be propagated to other agents.

7.1.7 Agent Teams Capabilities

From the use case described in clause 5.2 of the present document:

Capability-7.1.7-1: The ZSM framework should enable the dynamic formation of agent teams, allowing multiple agents to be grouped together to enhance overall performance in a multi-agent system.

Capability-7.1.7-2: The ZSM framework should support the capability to dissolve agent teams in a multi-agent system.

7.1.8 Terminology Alignment Capabilities

From the use case described in clause 5.5 of the present document:

Capability-7.1.8-1: The ZSM framework should support the capability for agents to embed term definitions within agent interaction messages.

Capability-7.1.8-2: The ZSM framework should support the capability for agents to share term definition files with reference URI.

Capability-7.1.8-3: The ZSM framework should support the capability for agents to bind term definition to specific task.

7.2 Recommendations for Normative work

Based on the use cases and the expected capabilities described in the present document regarding the utilization of Agents in Autonomous Networks within the Zero-touch network and Service Management (ZSM) framework, the following recommendations are provided for future normative work:

- **Recommendation 1**: The ZSM framework should formally define the Agent Entity and its lifecycle, clarifying its role as a functional component capable of consuming and producing ZSM management services. Normative specifications should be developed to define the architectural roles and responsibilities for integrating multi-agent systems across MDs and E2ES MD.

- **Recommendation 2**: The ZSM framework should standardize management services that enable Agent-to-Agent communication to facilitate collaboration, coordination, and task negotiation or assignment. Existing ZSM framework management services should be analysed to identify any missing capabilities required to fully support agent utilization within the framework.

- **Recommendation 3**: The ZSM framework should provide a stage 2 normative specification of the Agent Capability Profile, which formalizes the functional, operational, and contextual attributes of an autonomous agent. By standardizing these attributes, the profile enables seamless agent interoperability and facilitates the identification of agents for efficient task delegation within the ZSM framework.

- **Recommendation 4**: The topic of Agent Security and Trust, which is only briefly addressed in the present document, should be expanded. Normative specifications should define the necessary security capabilities and services to ensure the trustworthiness and secure operation of agents. This includes mechanisms for identity management and dynamic access control for agents operating in multi-agent systems, particularly when functioning across multiple MDs. Additionally, comprehensive audit capabilities should be integrated to enable continuous monitoring, logging, and analysis of agent actions, access, and communications to detect anomalies, enforce accountability, and support investigations. Existing ZSM security and trust management services should be reviewed to identify possible extensions and enhancements to support agents.

- **Recommendation 5**: The ZSM framework should specify normative means to define the mechanisms, services, and formats necessary for agents to efficiently exchange learned knowledge between autonomous agents across MDs and E2ES MD. Normative specifications should define mechanisms for agents to exchange trained models, historical observations, episodic experiences, semantic knowledge, and procedural routines, supporting collaborative reasoning and collective learning. Existing ZSM framework management services should be reviewed to identify the necessary extensions and new capabilities required for robust, context-aware, and secure knowledge exchange among agents.

- **Recommendation 6**: ISG ZSM should continue the coordination and cooperation efforts with relevant Standards Developing Organizations (SDOs) and open-source groups (e.g. IRTF NMRG, ITU-T SG13, TM Forum) to align the agent definitions, communication models, and integration points, thus ensuring wider industry applicability of the resulting specifications.

Annex A:
Change history

Date	Version	Information about changes
April 2025	V0.0.1	Initial skeleton
May 2025	V0.0.2	Incorporated contributions: • ZSM(25)000044r3_ZSM020_Concept_of_Agent • ZSM(25)000045r2_ZSM020_Agent_Taxonomy • ZSM(25)000046r1_ZSM020_Agents_in_the_ZSM_Framework • ZSM(25)000047r2_ZSM020_Overview_of_Multi-Agent_Systems • ZSM(25)000048r1_ZSM020_Agent_Capabilities_Registration_and_Discovery_Scenario • ZSM(25)000057_ZSM020__Additional_potential_ZSM_Capabilities_to_support_MAS • ZSM(25)000058_ZSM020__Additional_potential_ZSM_Capabilities_to_support_MAS • ZSM(25)000062_ZSM020__Additional_potential_ZSM_Capabilities_to_support_MAS • ZSM(25)000065_ZSM020__Additional_potential_ZSM_Capabilities_to_support_MAS • ZSM(25)000092r1_ZSM020_Agent_term
July 2025	V0.0.3	Incorporated contributions: • ZSM(25)000056r1_ZSM020__Add_subsections_and_a_description_related_to_clause_ • ZSM(25)000059r1_ZSM020__Additional_potential_ZSM_Capabilities_to_support_Mul • ZSM(25)000060r1_ZSM020__Additional_potential_ZSM_Capabilities_to_support_Mul • ZSM(25)000061r1_ZSM020__Additional_potential_ZSM_Capabilities_to_support_Mul • ZSM(25)000067r1_ZSM020__Additional_potential_ZSM_Capabilities_to_support_Mul • ZSM(25)000068r1_ZSM020__Additional_potential_ZSM_Capabilities_to_support_Mul • ZSM(25)000071r1_ZSM020__Additional_potential_ZSM_Capabilities_to_support_Mul • ZSM(25)000074_ZSM020__Additional_potential_ZSM_Capabilities_to_support_Mul • ZSM(25)000075_ZSM020__Additional_potential_ZSM_Capabilities_to_support_Mul • ZSM(25)000105_Change_ZSM_Agent_term • ZSM(25)000116_ZSM020_Cross-domain_fault_resolution_using_E2E_SDM_cross-dom • ZSM(25)000118r1_ZSM020_Agent_Communication_Models • ZSM(25)000127r1_ZSM020_Cross-Domain_Terminology_Alignment_for_Agent_Communic • ZSM(25)000128_ZSM020_Group-based_Registration_and_Discovery
August 2025	V0.0.4	Incorporated contributions: • ZSM(25)000069r1_ZSM020__Additional_potential_ZSM_Capabilities_to_support_Mul • ZSM(25)000070r1_ZSM020__Additional_potential_ZSM_Capabilities_to_support_Mul • ZSM(25)000072r1_ZSM020__Additional_potential_ZSM_Capabilities_to_support_Mul • ZSM(25)000073r1_ZSM020__Additional_potential_ZSM_Capabilities_to_support_Mul • ZSM(25)000154_ZSM020_Agents_Identity_Management • ZSM(25)000155_ZSM020_Agentic_Dynamic_Workflow • ZSM(25)000156r1_ZSM020_CR_for_Cross-Domain_Terminology_Alignment_for_Agent_C

Date	Version	Information about changes
September 2025	V0.0.5	Incorporated contributions: • ZSM(25)000149r1_ZSM020_Agent_Interaction_and_Coordination_ • ZSM(25)000150r1_ZSM020_Agents_Organization • ZSM(25)000157r1_ZSM020_CR_for_Group-based_Registration_and_Discovery • ZSM(25)000183_ZSM020_Coordination_and_collaboration • ZSM(25)000184r1_ZSM020_Task_Negotiation_use_case
November 2025	V0.0.6	Incorporated contributions: • ZSM(25)000151_ZSM020_Knowledge_Sharing_between_Agents • ZSM(25)000204r1_Move Section 6.2.5 as Use case • ZSM(25)000210r1_ZSM020_Definition of task • ZSM(25)000212r1_ZSM020_Agent producer and consumer of MnS • ZSM(25)000216_ZSM020_Hanging_paragraph_in_4.3 • ZSM(25)000217_ZSM020_Overview of Multi-Agent Systems • ZSM(25)000221_ZSM020_Identity_Management_update • ZSM(25)000222_ZSM020_Additional_registration_requirements • ZSM(25)000223_ZSM020_Additional_profile_requirements • ZSM(25)000224_ZSM020_Additional_agent_interaction_requirements • ZSM(25)000225_ZSM020_Clause7_reorg • ZSM(25)000226r1_ZSM020_Clause72_Normative_Recommendations • ZSM(25)000227_ZSM020_Clause61_Hanging_Paragraph • ZSM(25)000241r1_ZSM020_Editorial notes removal
December 2025	V0.0.7	Editorial changes

History

Version	Date	Status
V1.1.1	January 2026	Publication