Network Working Group
Internet-Draft
Intended status: Informational
Expires: 30 August 2026

D. Lopez
Telefonica
N. R. Moreno
Deutsche Telekom
L. Tailhardat
Orange
Q. Ma
Q. Wu
Y. Yang
Huawei
S. Prabhu
Nokia

26 February 2026

Applicability of A2A to the Network Management
draft-yang-nmrg-a2a-nm-02

Abstract

This document discusses the applicability of A2A protocol to the network management in the multi-domain heterogeneous network environment that utilizes IETF technologies. It explores operational aspect, key components, generic workflow and deployment scenarios. The impact of integrating A2A into the network management system is also discussed.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://Yuanyuan4666.github.io/A2A/draft-yang-a2a-nm.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-yang-nmrg-a2a-nm/.

Source for this draft and an issue tracker can be found at https://github.com/Yuanyuan4666/A2A.

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 30 August 2026.

Table of Contents

1. Introduction

With the advancement of large language models (LLMs), the concept of AI agents has gradually attracted significant attention. An AI agent refers to a category of software applications that utilizes LLMs to interact with users or other agents and accomplish specific tasks. Take a multimodal AI agent as an example, it can collaborate with other domain-specific agents to complete diverse tasks such as translation, configuration generation, and API development.

A2A protocol [A2A] provides a standardized way for AI agents to communicate and collaborate across different platforms and frameworks through a structured process, regardless of their underlying technologies. Agents can advertise their capabilities using an 'Agent Card' in JSON format, or send messages to communicate context, replies, artifacts, or user instructions, which make it easier to build AI applications that can interact with heterogeneous AI ecosystems in specific domains.

With significant adoption of AI Agents across the Internet, Agent to Agent Communication protocol may become the foundation for the next wave of Internet communication technologies across domains [I-D.rosenberg-ai-protocols]. The application of A2A in the network management field is meant to develop various rich AI driven network applications, realize intent based networks management automation in the multi-vendor heterogeneous network environment. By establishing standard interfaces for dynamic Capability Discovery, intelligent message routing, heterogeneous AI ecosystems interaction, cross-platform collaboration, A2A enables AI Agents to:

- Understand contextual nuances

- Negotiate and adapt in real-time

o Make collaborative decisions

o Maintain persistent, intelligent interactions

This document discusses the applicability of A2A to the network management in the multi-domain heterogeneous network environment that utilizes IETF technologies. It explores operational aspect, key components, generic workflow and deployment scenarios. The impact of integrating A2A into the network management system is also discussed.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

* AI Agent: A software system or program that is capable of autonomously performing goals and tasks on behalf of a user or another system.

* Agent Card: A common metadata file that describes an agent’s capabilities, skills, interface URLs, and authentication requirements. Clients discover and identify the agent through this file.

* A2A Server: An AI agent that receives requests and performs tasks

* A2A Client: An AI agent that sends requests to servers

* message: An A2A message represents a single turn of communication between a client and a server Agent. A message contains one or more Part objects.

* part: A part object is a granular container for the actual content, which can hold different types of content using exactly one of the following content fields:

  - text: A string containing plain textual content.
  - raw: A byte array containing binary file data (inline).
  - url: A string URI referencing external file content.
  - data: A structured JSON value (e.g., object, array) for machine-readable data.

3. Overview of key challenges for the network management

As described in [I-D.wmz-nmrg-agent-ndt-arch], 3 key challenges to apply A2A protocol to network management have been listed:

o High Risk Operations of Agent2Agent interaction

* High risk operation can can to large-scale network outages.

o The timeliness requirement of Agent2Agent collaboration

* "Token-based" generation and reasoning approach, limited by computing power and algorithms, result in slow reasoning speeds.

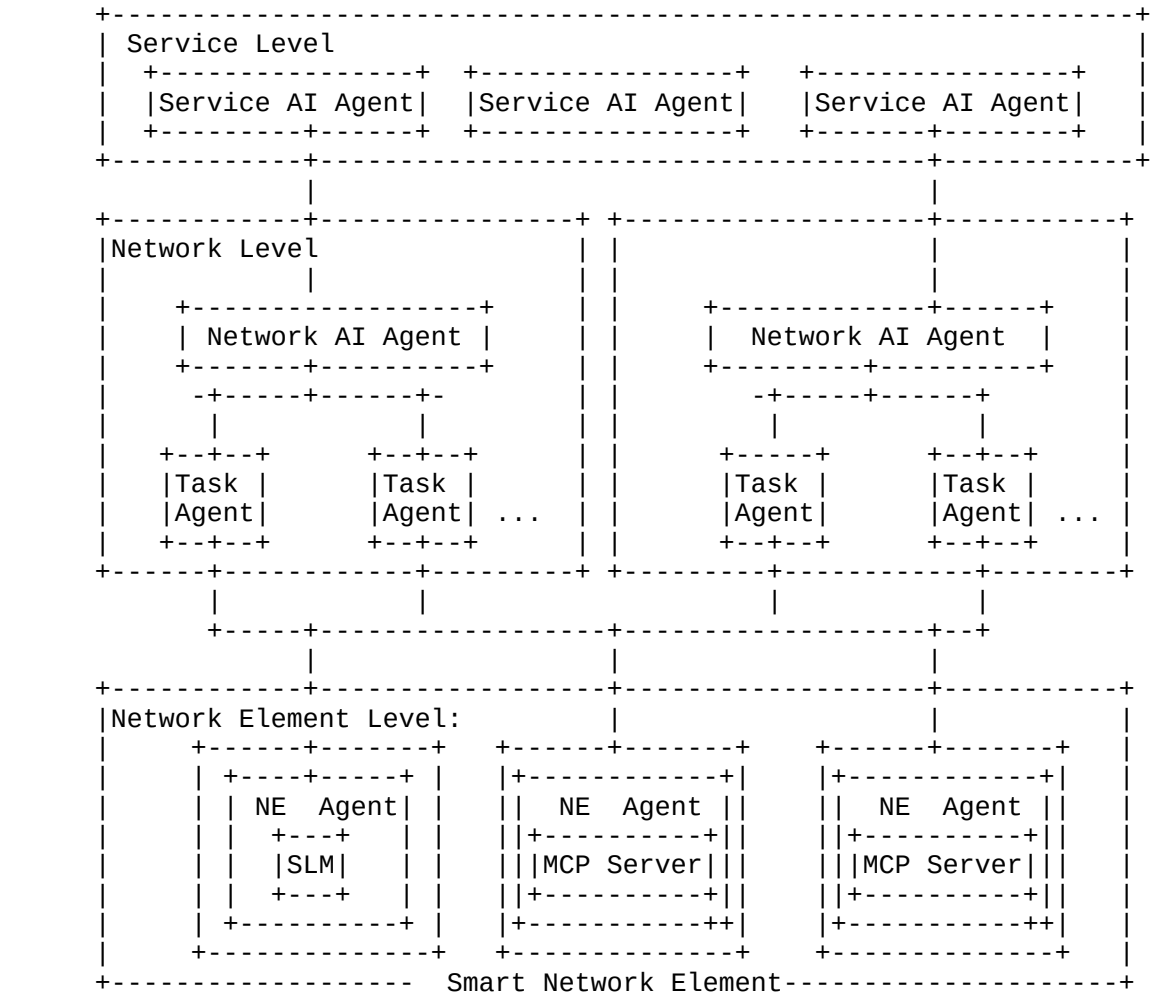
* Task-oriented "request-response" model without event subscription is unlike to meet time constraints for tasks such as fault diagnosis, complaint handling, and user experience improvement.

o The Agent2agent Collaboration Reliability

* Incorrect or outdated network configuration data results in incorrect repair advice when diagnosing network incidents or faults.

* Task collaboration is incomplete or not sufficient to handle strategies such as task rejection, missing information during task collaboration, and failure to achieve task objectives.

4. Agent2Agent Architecture Design for Network Management



As described in [I-D.wmz-nmrg-agent-ndt-arch], in the multi-agent communication deployment scenario, AI Agents can be deployed at both service layer,network layer and Network Element Level, e.g., both service orchestrator and network controller can introduce AI Agent and allow Agent to Agent communication. Service AI Agent within the service orchestrator can provide registration center for other network AI agents and task agents within the network controller to register its location. In the meanwhile Network AI Agent within the network Controller can provide registration center for task agents at the network level and Network element level AI agent within the smart network element.

The interaction in the multi-agent communication deployment scenario can be broken down into:

* _AI Agent to Agent interaction_

* _AI Agent to Tools/APIs/Models interaction_

For AI Agent to Tools/APIs/Models interaction, to enable comprehensive functionality (e.g., large AI model and small AI model

collaboration, network AI agent and Network element AI agent collaboration for routing protocol troubleshooting), additional protocol extensions are required to address two critical aspects: (1) standardized tool invocation mechanisms for agent-tool/api/model interoperability, and (2) monitoring frameworks for tool usage tracking and auditing.

AI Agent to Agent interaction, human operators require a real-time monitoring interface for long-running workflows or tasks requiring continuous supervision with dual capabilities: (1) live network state observation and (2) validation of agent-proposed remediation actions during anomaly resolution scenarios.

A general workflow is as follows:

* User Input Submission: An operator submits a natural language request to a Service AI agent.

* Agent Intent Processing: The service AI agent processes natural language inputs by parsing instructions into structured tasks.

* Autonomous Close Loop Workflow Management: The service AI agent decomposing tasks into workflow map with subtasks, and distributes subtasks via an Agent Card Registry to specialized task agents based on their capabilities.

* Task Execution: Iteration continues until all tasks reach executable task agents in the hierarchy.

* Task Report: Task agents report outcomes to the central agent, which dynamically adjusts the workflow based on result analysis and policy rules.

5. YANG-based Structured Data for A2A Communication

While the A2A framework natively supports unstructured textual content for agent-to-agent data exchange, network management and operation scenarios usually demand rigorous clarity, unambiguous intent transmission and machine-interpretable data interaction - attributes that natural language cannot reliably provide due to its inherent ambiguity, contextual variability and lack of standardized syntax. Natural language expressions of network operational intent may have incomplete information, leading to incorrect task execution, inconsistent configuration deployment and potential large-scale network outages, which are unacceptable in the high-reliability requirements of network management.

YANG [RFC7950], as a standardized data modeling language defined by the IETF for network management, provides a extensible way to structure network management data and operational service and network intent; using YANG-modeled structured data to populate A2A communication payloads could help eliminate the ambiguity of natural language, and align A2A communication with the existing IETF-based network management ecosystem, enabling seamless integration with traditional network management protocols such as NETCONF [RFC6241] and RESTCONF [RFC8040]. The well-defined hierarchies and structures of YANG data models also enable Agents to quickly parse, validate and process communication data and support the definition of service or network intent for specific multi-domain and multi-vendor heterogeneous network scenarios. In addition, YANG-structured data can serve as a precise supplement to natural language input, where implicit parameters, missing constraints, or detailed operational conditions that are not fully expressed in natural language can be explicitly defined and carried in the YANG data part.

The following example illustrates a A2A [A2A] message with both a

YANG-based structured data and natural language parts to balance human readability and machine parse-ability. The message could be sent from a network AI Agent, after receving the intent from the operator to diagnose a specific network incident, to a incident diagnosis task Agent. The data part complies with the Network Incident YANG data model defined in [I-D.ietf-nmop-network-incident-yang].

```
POST /agents/network-ai-agent HTTP/1.1
Host: example.com
Content-Type: application/json
{
  "jsonrpc": "2.0",
  "method": "message/send",
  "params": {
    "message": {
      "message_id": "123e4567-e89b-12d3-a456-426614174000",
      "context_id": "conversation-12345",
      "role": "ROLE_USER",
      "parts": [
        {
          "text": "Please diagnose the service degeneration incident
                   for 'optical-svc-A' in 'FAN' domain. Provide root
                   cause, severity level, and resolution
                   recommendations.",
          "media_type": "text/plain"
        },
        {
          "data": {
            "incident": {
              "name": "Service Degradation",
              "type": "network_problem",
              "incident_id": "56433218",
              "service_instances": ["optical-svc-A"],
              "domain": "FAN",
              "priority": "critical",
              "status": "raised",
              "occurrence_time": "2026-02-10T04:01:12Z",
              "last_updated": "2026-02-10T04:01:12Z",
              "probable_events": [
                {
                  "event_id": "8921834",
                  "type": "alarm"
                }
              ],
              "related_events": [
                {
                  "event_id": "8921832",
                  "type": "alarm"
                },
                {
                  "event_id": "8921833",
                  "type": "alarm"
                },
                {
                  "event_id": "8921834",
                  "type": "alarm"
                }
              ]
            }
          },
          "media_type": "application/json",
          "metadata": {
            "yang_module": "ietf-incident",
            "revision": "2025-09-16"
          }
```

```
          }
        ]
      },
      "configuration": {
        "accepted_output_modes": ["application/json"],
        "blocking": false,
        "history_length": 3
      },
      "metadata": {
        "request_type": "incident_diagnosis",
        "priority": "critical",
        "response_deadline": "2026-02-10T06:00:00Z"
      }
    }
  }
```

6. Operational Considerations

The introduction of A2A-based agent interactions into network management has several operational implications that must be considered when deploying the architecture in large-scale or multi-domain networks. This section highlights key aspects related to performance, scalability, reliability, latency, and agent lifecycle operations.

6.1. Agent Skills as Expertise Expansion of AI Agents

While AI agents have intelligence and capabilities, they may not always have expertise that we expect when performing specific network management and operation tasks. Agent Skills [Agent-skills], introduced by Anthropic, is an open standard that allows developers to package specialized knowledge, workflow, and scripts and empowers a general AI Agent to become an expert in a specified field. Each skill is organized as a seperate folder that consists of a "skill.md" to define the basic information of the skill, and other files such as scripts, reference documents, etc. Agent Skills use progressive disclosure as a design pattern to load these resources to reduce token consumption and use less of the context window.

Network operators could encode their domain expertise (e.g., troubleshooting workflow logic for fault scenarios) into structured skills, e.g., by defining the fault_diagnose_link skill to organize the logic of "checking link connectivity -> analyzing error syslogs -> verifying hardware status -> outputting a resolution solution"), it quickly equips AI Agents with domain expertise.

6.2. Knowledge Base as Ground Truth Data

Another critical operational consideration for Agent-to-Agent (A2A) communication in network management is addressing the hallucination of Large Language Models (LLMs), which primarily stems from knowledge gaps within the models themselves. To mitigate this, a dedicated and machine-interpretable knowledge base can be constructed using unstructured product documents, maintenance manuals, historical fault tickets, network topology diagrams, configuration specifications, and expert experience, etc. The knowledge base enables LLMs to retrieve accurate, network operation and maintenance-specific information for reliable responses while ensuring data privacy and security, supporting domain knowledge plug-in to supplement knowledge Q&A. A shared knowledge base helps eliminate information asymmetry between heterogeneous Agents, ensuring that all Agents across the entire A2A system base their judgments and actions on consistent knowledge standards to avoid miscommunication or inconsistent operations. Throughout the entire A2A operational lifecycle, the knowledge base is not a static resource but a dynamic core that permeates Agent initialization, communication and interaction, task execution, and

result feedback, making knowledge base management a prerequisite for effective A2A system operational design.

Notably, The emerging Model Context Protocol (MCP) can facilitate the efficient updates and queries of the knowledge base by providing standardized interfaces, and build a standardized external knowledge base for multi-Agent system.

6.3. Event-driven Agent to Agent Communication

The event-driven Agent to Agent communication enhances the task-based A2A procotol to support proactive and real-time communication based on network events. By leveraging the Message Broker such as Apache Kafka to facilitate the exchange of event messages among different AI agents, it allows the real-time response to maintain network reliability and service assurance in network management and operations. Figure 1 gives an overview of the architecure.

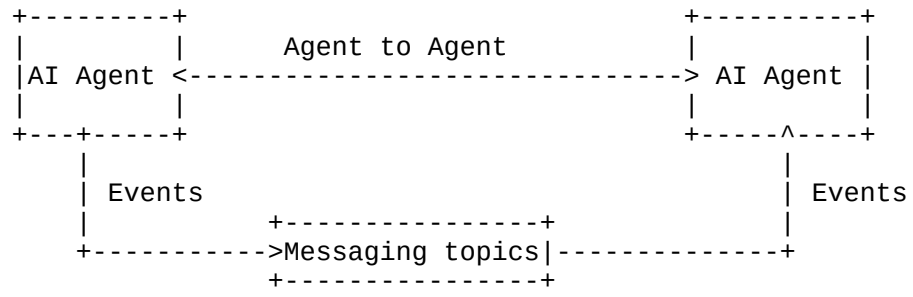


Figure 1: An Architecture for Event-driven A2A

The event-driven extension introduces a publish/subscribe paradigm alongside the primary task-driven interaction. For example, a "Fault Agent" identifies a link failure by data analyzing and publishes a message to the Message Broker, the "recovery agent" subscribes to the topic and it could initiate a task to generate recovery strategies such as link switching and traffic rerouting and submit to the operator for approval upon consuming the message.

6.4. Scalability

Large operational networks may contain tens of thousands of devices, multiple administrative domains, and a distributed set of controllers and orchestrators. The use of conversational, context-rich A2A messaging increases message volume compared to static RPC-based interfaces such as NETCONF or RESTCONF.

Operators should evaluate:

* The number of agents required per domain or per function.

* The expected message growth as workflows involve multiple agents performing negotiation, capability discovery, and state exchange.

* The impact of concurrent multi-agent workflows on control-plane stability.

* Whether hierarchical or federated agent structures are needed to avoid message storms and to localize decisions.

Mechanisms for rate-limiting, backoff, and prioritization may be needed to prevent overload.

6.5. Latency and Performance Constraints

Task decomposition and negotiation across multiple agents can

introduce non-trivial latency, especially when agents rely on external AI inference engines or large language models (LLMs).

Operational environments may impose strict timing requirements, for example during:

* Service activation with customer-facing SLAs.

* Fault detection and automated remediation loops.

* Real-time telemetry-driven control such as congestion mitigation.

Implementations should define performance envelopes, including:

* Maximum agent-to-agent message processing latency.

* Timeout and retry behavior for workflow steps.

* Acceptable degradation under load or partial failures.

Fallback mechanisms (e.g., reverting to direct controller APIs or static policies) should be provided when A2A interactions cannot meet timing constraints.

6.6. Reliability and Failure Handling

A2A workflows may involve long-lived tasks that span multiple agents and systems. Operational networks require predictable and safe behavior under partial failures.

Operators should consider:

* How workflow state is checkpointed or restored if an agent becomes unreachable.

* How to detect and mitigate inconsistent or stale agent state.

* Whether workflows can be retried idempotently.

* Requirements for transactionality or rollback comparable to NETCONF confirmed-commit semantics.

Implementations should include mechanisms for workflow monitoring, circuit-breakers, and automatic escalation to human operators in case of sustained failure.

6.7. Agent Lifecycle and Resource Management

Production deployment of A2A-based systems requires active management of agent lifecycles, including:

* Agent onboarding and registration.

* Updates and model re-training (for AI-driven agents).

* Decommissioning and revocation of compromised agents.

* Resource consumption limits for CPU, memory, and inference workloads.

Operators should maintain visibility into the operational state of all agents and their dependencies, including telemetry on message rates, errors, and workflow completion metrics.

6.8. Inter-Domain Operational Challenges

In multi-domain scenarios (e.g., between business units, operators, or federated networks), operational concerns are amplified due to:

* Differences in local policies or SLAs.

* Variations in controller capabilities and data models.

* Latency and reliability across administrative boundaries.

* Need for shared or interoperable agent capability descriptions.

Standardized operational practices may be required for agent discovery, trust establishment, conflict resolution, and accountability.

7. Security Considerations

The communication between Agents for the exchange of context information, capability information and user instruction is security sensitive and requires authentication, authorization, and integrity protection. Legacy communication protocols such as HTTPS/TLS, designed for human-centric interactions, simply cannot withstand the high-speed exchanges between intelligent agents. Key security challenges in AI agent communication include:

* Identity Verification: Ensuring that agents are who they claim to be

* Data Integrity: Preventing unauthorized modifications during transmission

* Confidentiality: Protecting sensitive information from potential breaches

* Scalable Security: Maintaining robust protection across diverse and complex networks

8. IANA Considerations

This document has no IANA actions.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

9.2. Informative References

[A2A] "Agent2Agent (A2A) protocol", April 2025, <https://google-a2a.github.io/A2A/#/documentation?id=agent2agent-protocol-a2a>.

[Agent-skills] "Agent Skills", 2025, <https://agentskills.io/home>.

[I-D.ietf-nmop-network-incident-yang] Hu, T., Contreras, L. M., Wu, Q., Davis, N., and C. Feng, "A YANG Data Model for Network Incident Management", Work

in Progress, Internet-Draft, draft-ietf-nmop-network-incident-yang-08, 13 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-nmop-network-incident-yang-08>.

[I-D.rosenberg-ai-protocols] Rosenberg, J. and C. F. Jennings, "Framework, Use Cases and Requirements for AI Agent Protocols", Work in Progress, Internet-Draft, draft-rosenberg-ai-protocols-00, 5 May 2025, <https://datatracker.ietf.org/doc/html/draft-rosenberg-ai-protocols-00>.

[I-D.wmz-nmrg-agent-ndt-arch] Wu, Q., Zhou, C., Contreras, L. M., Han, S., and Y. Hong, "Network Digital Twin and Agentic AI based Architecture for AI driven Network Operations", Work in Progress, Internet-Draft, draft-wmz-nmrg-agent-ndt-arch-03, 23 February 2026, <https://datatracker.ietf.org/doc/html/draft-wmz-nmrg-agent-ndt-arch-03>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/rfc/rfc6241>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/rfc/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/rfc/rfc8040>.

Appendix A. Usage Example

This section describes the deployment of a network configuration within a secure video meeting context. The scheduling agent is deployed to the Service Orchestrator, while the worker agent is deployed to the network controller. Registered on the Service Orchestrator, the agent card formally defines a worker agent's capabilities, interfaces, and operational characteristics within network management systems.

See the following Agent card examples for two worker agents (QoS Agent and Security Agent):

```
 # Worker Agents Capabilities
 {
     "name": "QoSAgent",
     "description": "Automatically configure QoS policies",
     "url": "https://qos-agent.example.com/tasks/send",
     "capabilities": ["QoS_Policy"],
     "skills": [
         {
             "id": "set_qos",
             "name": "QoS configuration",
             "description": "QoS configuration",
             "inputModes": ["text/structured"],
             "outputModes": ["text/status"]
         }
     ]
 }
 {
     "name": "SecurityAgent",
     "description": "Automatically configure network security policies",
     "url": "https://security-agent.example.com/tasks/send",
```

```
    "capabilities": ["IPSEC", "DTLS"],
    "skills": [
        {
            "id": "enable_encryption",
            "name": "Encryption method configuration",
            "description": "Encryption method configuration",
            "inputModes": ["text/structured"],
            "outputModes": ["text/status"]
        }
    ]
}
```

Suppose a user submits a natural language request such as "The meeting will have 100 participants. The security level is Top Secret" to the platform integrated with the Service Orchestrator. The platform parses the request and converts it into JSON format as follows:

```
# Requested Service Configuration
{
    "taskId": "task-multi-001",
    "action": "deploy_network_configuration",
    "parameters": {
        "context": "secure_video_meeting",
        "scope": "100",
        "secure_level": "Top Secret",
    }
}
```

The Service Orchestrator sends subtasks in a structured format to the Network Controller. For example, the subtasks for set_qos and enable_encryption are structured as follows:

```
# Set QoS and Enable Encryption Subtasks
{
    "taskId": "task-multi-001",
    "subTasks": [
        {
            "agent": "QoSAgent",
            "action": "set_qos",
            "parameters": {
                    "configuration": {
                            "acceptedOutputModes": [
                                    "text/status"
                    ]
                    },
                    "minimum_bandwidth": "100Mbps",
                    "priority": "0"
            }
        },
        {
            "agent": "SecurityAgent",
            "action": "enable_encryption",
            "parameters": {
                    "configuration": {
                            "acceptedOutputModes": [
                                    "text/status"
                    ]
                    },
                    "encryption_method": "ipsec",
                    "key_management": "dtls",
            }
        }
    ]
}
```

The network controller executes network management operations on network devices and returns the results to the Service Orchestrator in JSON format. Example responses for the subtasks are shown below:

```
# Network Configuration Feedback Results
{
    "taskId": "task-multi-001",
    "action": "deploy_network_configuration",
    "parameters": {
        "context": "secure_video_meeting",
        "scope": "100",
        "secure_level": "Top Secret",
    }
}
{
  "taskId": "subtask-qos-001",
  "status": "completed",
  "artifacts": [{"type": "text", "content": "QoS setup completed"}]
}
{
  "taskId": "subtask-sec-001",
  "status": "completed",
  "artifacts": [{"type": "text", "content": "IPSEC encryption enabled"}]
}
```

Contributors

Houda Chihi
InnovCOM Sup'COM
Email: houda.chihi@supcom.tn

Authors' Addresses

Diego Lopez
Telefonica
Email: diego.r.lopez@telefonica.com

Nathalie Romo Moreno
Deutsche Telekom
Email: nathalie.romo-moreno@telekom.de

Lionel Tailhardat
Orange
Email: lionel.tailhardat@orange.com

Qiufang Ma
Huawei
Email: maqiufang1@huawei.com

Qin Wu
Huawei
Email: bill.wu@huawei.com

Yuanyuan Yang
Huawei
Email: yangyuanyuan55@huawei.com

Shailesh Prabhu
Nokia

Email: shailesh.prabhu@nokia.com