WG Working Group
Internet-Draft
Intended status: Informational
Expires: 3 September 2026

R. Schott
Deutsche Telekom
J. Maisonneuve
Nokia
L. M. Contreras
Telefonica
J. Ros-Giralt
Qualcomm Europe, Inc.

2 March 2026

Agentic AI Use Cases
draft-scrm-aiproto-usecases-02

Abstract

Agentic AI systems rely on large language models to plan and execute multi-step tasks by interacting with tools and collaborating with other agents, creating new demands on Internet protocols for interoperability, scalability, and safe operation across administrative domains. This document inventories representative Agentic AI use cases and captures the protocol-relevant requirements they imply, with the goal of helping the IETF determine appropriate standardization scope and perform gap analysis against emerging proposals. The use cases are written to expose concrete needs such as long-lived and multi-modal interactions, delegation and coordination patterns, and security/privacy hooks that have protocol implications. Through use case analysis, the document also aims to help readers understand how agent-to-agent and agent-to-tool protocols (e.g., [A2A] and [MCP]), and potential IETF-standardized evolutions thereof, could be layered over existing IETF protocol substrates and how the resulting work could be mapped to appropriate IETF working groups.

About This Document

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

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-scrm-aiproto-usecases/.

Source for this draft and an issue tracker can be found at https://github.com/https://github.com/giralt/draft-scrm-aiproto-usecases.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Agentic AI systems—software agents that use large language models to reason, plan, and take actions by interacting with tools and with other agents—are seeing rapid adoption across multiple domains. The ecosystem is also evolving quickly through open-source implementations and emerging protocol proposals; however, open source alone does not guarantee interoperability, since rapid iteration and fragmentation can make stable interoperation difficult when long-term compatibility is required. Several protocols have been proposed to support agentic systems (e.g., [A2A], [MCP], ANP, Agntcy), each with different design choices and strengths, targeting different functions, properties, and operating assumptions.

This document inventories a set of representative Agentic AI use cases to help the IETF derive protocol requirements and perform gap analysis across existing proposals, with a focus on Internet-scale interoperability. The use cases are intended to highlight protocol properties that matter in practice—such as long-lived interactions, multi-modal context exchange, progress reporting and cancellation, and safety-relevant security and privacy hooks—and to help the IETF determine appropriate scope as well as how related work should be organized across existing working groups or, if needed, a new effort.

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.

3. Use Cases Requirements

The use cases in this document are intended to inform IETF standardization work on Agentic AI protocols by clarifying scope, enabling gap analysis, and guiding working group ownership. The requirements below define the minimum level of detail and structure expected from each use case so that the IETF can derive actionable protocol requirements and identify where coordination with other SDOs is necessary. Use cases that do not meet these requirements risk being insufficiently precise for protocol design and evaluation.

* *IETF scope guidance*: Use cases MUST clearly indicate which protocol behaviors are expected to fall under the IETF's domain (e.g., Internet-facing interoperability, transport/session semantics, media/session behavior, congestion and reliability considerations, security and privacy hooks) versus what is out of scope for the IETF (e.g., model internals, proprietary orchestration logic). Use cases SHOULD also identify where coordination with other SDOs or industry initiatives is required to achieve interoperable and scalable outcomes.

* *Ecosystem boundary mapping*: Use cases SHOULD describe the relevant protocol ecosystem and interfaces between components (e.g., agent-to-agent vs. agent-to-tool) so the IETF can understand what can be standardized as Internet protocols and what is better treated as application/framework conventions. Where applicable, use cases SHOULD illustrate complementary roles of protocols such as agent-to-agent interaction (e.g., [A2A]) and agent-to-tool interaction (e.g., [MCP]).

* *Gap analysis readiness*: Use cases MUST be structured so that an engineer can map them to existing proposals and then identify missing, underspecified, or insufficiently mature protocol capabilities that block deployment. Use cases SHOULD include enough detail to reveal gaps, and MUST distinguish between gaps that plausibly belong in IETF standardization versus gaps that are purely implementation choices.

* *Adoption and layering*: Use cases SHOULD explain how non-IETF protocols that may be brought into the IETF (e.g., an A2A-like protocol) could be layered on top of, and interoperate cleanly with, existing IETF protocols (e.g., HTTP, QUIC, WebRTC, TLS). Use cases MUST identify assumed transport/bindings and the key interoperation points (e.g., discovery, session establishment, streaming, error handling) needed to assess architectural fit and integration impact.

* *Communication mode detail*: Use cases MUST describe the communication modes required between agents and between agents and tools reachable over the Internet, such as interactive request/response, asynchronous workflows, bulk transfer, incremental streaming, and notification patterns. Use cases SHOULD also indicate modality needs (text, audio/video, files, structured artifacts) when relevant.

* *Performance and safety needs*: Use cases SHOULD include explicit performance requirements when meaningful (e.g., latency sensitivity, bandwidth intensity, jitter tolerance, session duration, scalability expectations). Use cases MUST also call out safety-relevant requirements that have protocol implications (e.g., authorization and consent gates, provenance/citation needs, integrity and replay protection, isolation boundaries for tool invocation).

* *WG ownership signals*: Use cases SHOULD be decomposable into protocol functions that can be mapped to existing IETF working groups (e.g., transport, security, applications, operations/management, identity). Use cases MUST highlight cross-area dependencies (e.g., session + media + security) so the IETF can assess whether coordination across existing WGs is sufficient or whether forming a new WG is justified.

* *Operational realism*: Use cases SHOULD reflect real deployment constraints on the Internet. This requirement helps ensure the resulting protocol requirements are implementable and deployable at scale, rather than being tied to a single controlled environment.

* *Trust boundaries explicit*: Use cases MUST identify administrative domains and trust boundaries (e.g., user device, enterprise perimeter, third-party tool providers, external agent providers) and SHOULD summarize the expected security posture at those boundaries (authentication, authorization, confidentiality, and auditability expectations). This helps ensure the IETF does not miss protocol hooks needed to safely operate agentic systems across domains.

4. Use Cases

This section inventories representative Agentic AI use cases to make their protocol-relevant requirements explicit and comparable. The use cases are written to expose concrete needs such as multi-step delegation, agent-to-agent coordination, agent-to-tool interactions, and long-lived and multi-modal exchanges that must operate safely and reliably across administrative domains. By grounding the discussion

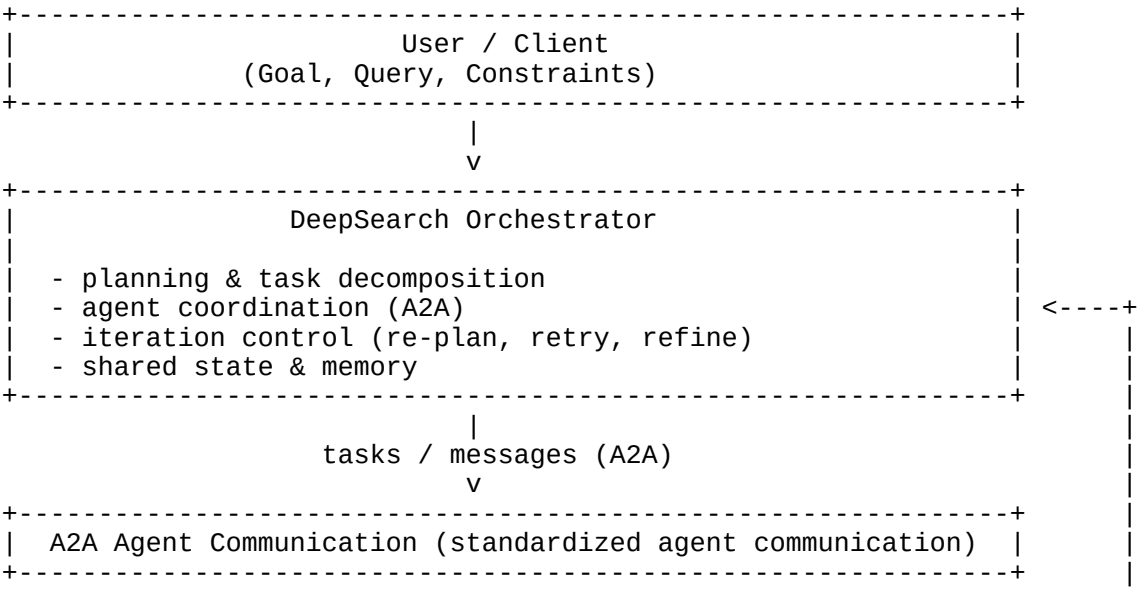
in specific scenarios, the document supports gap analysis against emerging agent protocols (e.g., agent-to-agent and agent-to-tool approaches such as A2A and MCP) and clarifies how candidate solutions could be layered over existing IETF protocol substrates and mapped to appropriate IETF working groups, including the necessary security and privacy hooks.

4.1. Deep Search

Deep Search refers to an _agentic_ information-seeking workflow in which an AI agent plans, executes, and iteratively refines multi-step research across heterogeneous sources such as open web, enterprise knowledge bases, APIs, files, and computational tools, among others. Unlike one-shot retrieval or a single RAG call, Deep Search is _long-horizon_ and _goal-directed_: the agent decomposes a task into sub-goals, issues searches and crawls, reads and filters evidence, runs auxiliary computations (e.g., code or math), verifies claims, tracks provenance/citations, and synthesizes a final answer---often over minutes or hours rather than milliseconds. This loop is typically implemented as _think -> act (tool) -> observe -> reflect -> refine plan_ until success criteria (e.g., coverage, confidence, cost/time budgets) are met.

4.1.1. Building Blocks

A Deep Search workflow may generally comprise the components shown in the next Figure:



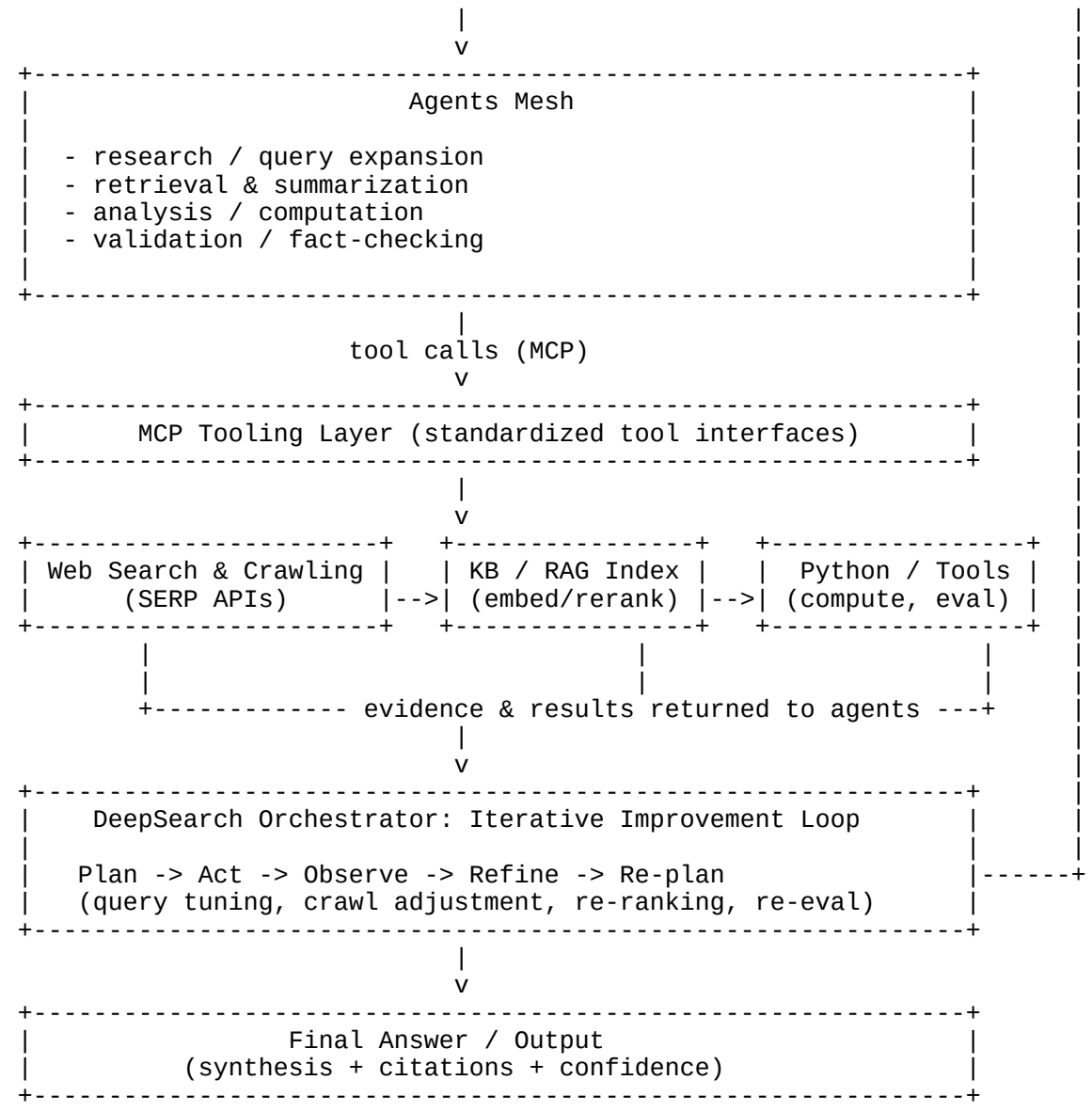


Figure 1: Deep Search agentic workflow

Each building block in the DeepSearch architecture represents a logical function rather than a specific implementation, and multiple components may be co-located or distributed in practice.

4.1.1.1. User / Client

The _User / Client_ is the entry point to the system. It provides the initial goal or query, along with optional constraints (e.g., scope, freshness, format). The user does not interact directly with tools or agents; all interactions are mediated by the DeepSearch Orchestrator.

4.1.1.2. DeepSearch Orchestrator

The _DeepSearch Orchestrator_ acts as the control plane of the system. Its responsibilities include:

* Planning and task decomposition of the user's request.
* Coordinating agents via Agent-to-Agent (A2A) communication.
* Managing shared state and memory across iterations.
* Controlling iterative execution, including retries and refinements.

The orchestrator does not perform retrieval or computation directly; instead, it delegates work to agents and manages the overall execution flow.

4.1.1.3. A2A Agent Communication Bus

The _A2A Agent Communication Bus_ provides a standardized messaging layer that enables agent-to-agent coordination. It supports:

* Task dispatch and response exchange.
* Collaboration among specialized agents.
* Decoupling of agent implementations from orchestration logic.

This bus allows agents to operate independently while still contributing to a coherent end-to-end workflow.

4.1.1.4. Agents Mesh

The _Agents Mesh_ block represents a set of specialized, cooperative agents operating over the A2A bus. Typical agent roles include:

* Research and query expansion.
* Retrieval and summarization.

* Analysis and computation.

* Validation and fact-checking.

Agents are responsible for invoking tools, interpreting results, and returning structured observations to the orchestrator.

4.1.1.5. MCP Tooling Layer

The _MCP Tooling Layer_ provides a standardized interface between agents and external tools. It enables:

* Discovery and invocation of tools using a common protocol.

* Consistent input/output schemas across heterogeneous tools.

* Isolation of agent logic from tool-specific details.

MCP acts as an abstraction boundary that simplifies integration and evolution of external capabilities.

4.1.1.6. Web Search & Crawling

The _Web Search & Crawling_ component combines content discovery and acquisition. It typically includes:

* Search engine or SERP APIs for identifying relevant sources.

* Focused crawling or fetching to retrieve selected content.

This component supplies raw external data that can be further processed and indexed.

4.1.1.7. Knowledge Base (KB) / Retrieval Augmented Generation (RAG) Index

The _KB / RAG Index_ component manages knowledge representation and retrieval. Its responsibilities include:

* Embedding and indexing retrieved content.

* Ranking or re-ranking results based on relevance.

* Supplying context to agents for retrieval-augmented generation (RAG).

This block provides structured, queryable knowledge derived from external sources.

4.1.1.8. Python / Tools

The _Python / Tools_ component represents general-purpose computation and evaluation capabilities. Examples include:

* Data processing and transformation.

* Numerical analysis or simulations.

* Quality evaluation, scoring, or consistency checks.

These tools are typically invoked by analysis-oriented agents via the MCP layer.

4.1.1.9. Iterative Improvement Loop

The _Iterative Improvement Loop_ captures the system's ability to refine results over multiple passes and is also implemeted by the DeepSearch Orchestrator. Conceptually, it follows a cycle of:

Plan -> Act -> Observe -> Refine -> Re-plan

Observations and intermediate results are fed back into the orchestrator, which may adjust plans, agent assignments, or tool usage before producing the final output.

4.1.1.10. Final Answer / Output

The _Final Answer / Output_ is the synthesized result returned to the user. It may include:

* A consolidated response or report.

* References or citations to supporting evidence.

* Confidence indicators or stated limitations.

This output reflects the outcome of one or more iterative refinement cycles.

4.1.2. Why This Use Case Matters

Deep Search is inherently _compositional_: it coordinates _multiple_ agents and _many_ tools over extended time. Without standard protocols, systems devolve into brittle, one-off integrations that are hard to test, secure, or reuse. Two complementary interoperability layers in the DeepSearch are especially relevant:

* *Agent-to-Tool standardization.* The _Model Context Protocol (MCP)_ defines a standardized mechanism by which agents and hosts can discover, describe, and invoke tools, resources, and prompts using JSON-RPC over multiple transports (e.g., stdio, HTTP with Server-Sent Events, and WebSocket). MCP enables portable and reusable tool catalogs (including search, crawling, retrieval-augmented generation (RAG), and general-purpose computation) with consistent schemas, capability negotiation, progress reporting, cancellation semantics, and explicit security prompts and user consent. Further details are specified in the MCP specification and related project documentation [MCP][MCP-GITHUB].

* *A2A Agent Communication Bus.* The _Agent2Agent (A2A)_ protocol focuses on standardized inter-agent collaboration. It defines mechanisms for agent capability discovery (e.g., Agent Cards), task lifecycle management (creation, cancellation, and status reporting), and streaming updates for long-running operations. A2A is designed to support opaque collaboration among agents while avoiding the need to disclose proprietary internal implementations. An overview of the protocol, along with its specification and design rationale, is available from the A2A project documentation [A2A][A2A-GITHUB].

Implications for Deep Search. Using A2A and MCP together lets implementers compose portable Deep Search stacks:

* Tools like crawlers, scholarly search, RAG, and Python are exposed via *MCP* with typed inputs/outputs and consent flows.

* Long-running research tasks, delegation to specialized researcher/verifier agents, background execution, progress streaming, and result handoff occur via *A2A*.

* Provenance (URIs, hashes, timestamps) and citation schemas can also be standardized at the protocol boundary to enable verifiable research traces across vendors.

* Enterprise requirements (authn/z), quotas, observability/tracing, policy enforcement (robots/copyright), and safety reviews—become portable rather than per-integration glue.

4.1.3. Example: Open Deep Search (ODS)

Open implementations illustrate agentic architectures for Deep Search.

Open Deep Search (ODS) is a modular, open-source framework that augments a base large language model with a dedicated Reasoning Agent and an Open Search tool. The framework is designed to support extensible, agentic search workflows in which an agent iteratively plans, invokes search tools, and synthesizes results to answer complex queries. Further details are available in the ODS publication and accompanying reference implementation [ODS][ODS-GITHUB].

ODS exemplifies the building blocks described earlier in this document and is consistent with the proposed interoperability layering, using standardized tool invocation for search and retrieval and agent-centric coordination to manage planning, execution, and refinement.

4.2. Hybrid AI

Hybrid AI generally refers to an _edgecloud cooperative_ inference workflow in which two or more models collaborate to solve a task: (1) a *smaller on-device model* (typically a few billion parameters) that prioritizes low latency, lower cost, and privacy; and (2) a *larger cloud model* (hundreds of billions to trillion-scale parameters) that offers higher capability and broader knowledge. The two models coordinate over an agent-to-agent channel and may invoke tools locally or remotely as needed. Unlike single-endpoint inference, Hybrid AI is _adaptive and budget-aware_: the on-device model handles as much work as possible locally (classification, summarization, intent detection, light reasoning), and escalates to the cloud model when additional capability is required (multi-hop reasoning, long-context synthesis, domain expertise). The models can exchange plans, partial results, and constraints over [A2A], and both sides can discover and invoke tools via [MCP].

4.2.1. Building Blocks

A Hybrid AI workflow may generally comprise the components shown in the next Figure:

* *On-device Model (Edge).* A compact LLM or task-specific model (a few billion parameters) running on user hardware (e.g., phone, laptop). Advantages include: low latency for interactive turns, reduced cost, offline operation, and improved privacy by default (data locality). Typical functions: intent parsing, entity extraction, local retrieval, preliminary analysis, redaction/ summarization prior to escalation.

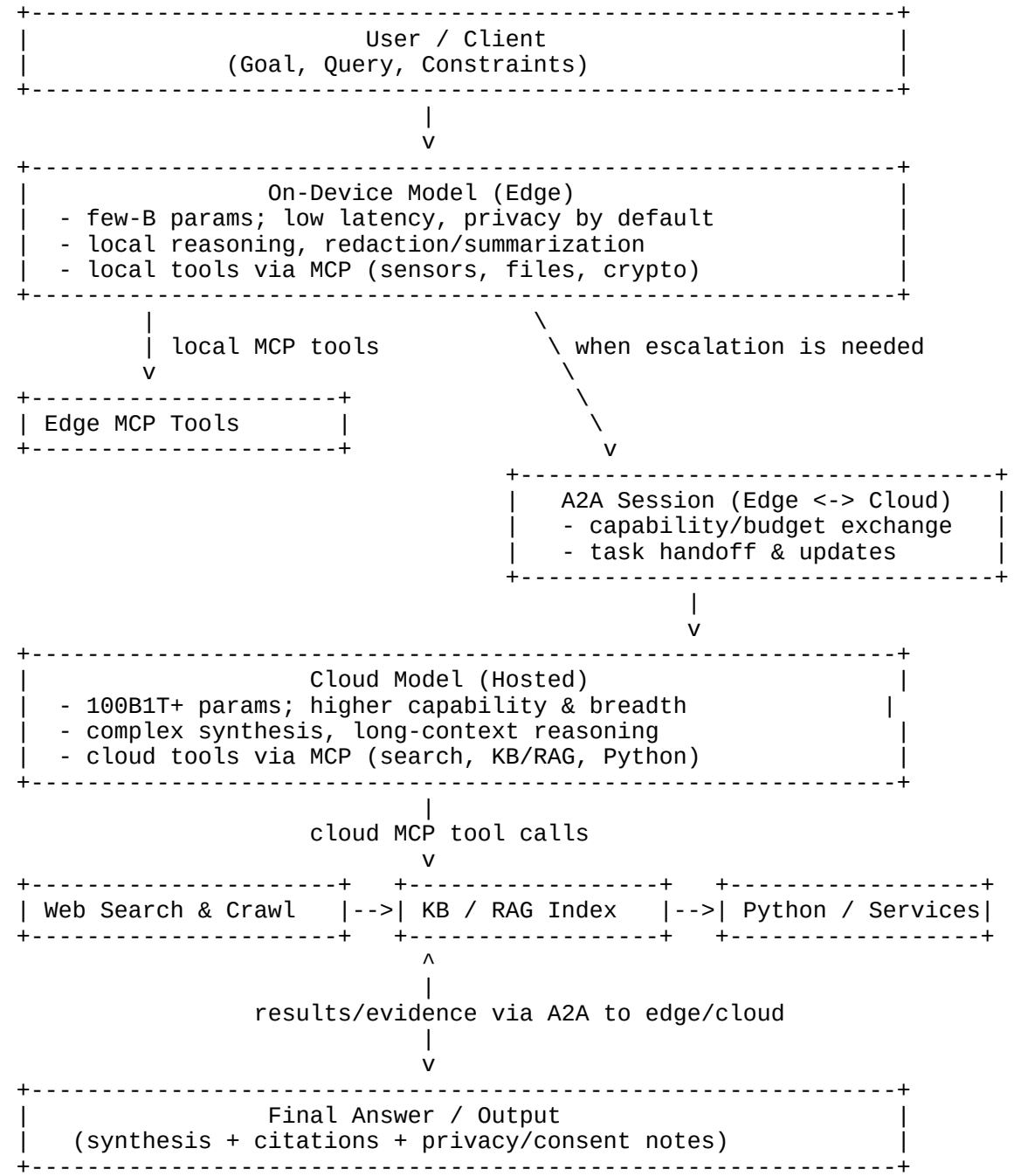
* *Cloud Model (Hosted).* A large, higher-capability LLM (hundreds of billions to ~trillion parameters) with stronger reasoning, knowledge coverage, tool-use proficiency, and longer context windows. Typical functions: complex synthesis, multi-step reasoning, broad web/KG retrieval, code execution, and advanced evaluation.

* *A2A Inter-Model Coordination.* The edge and cloud models communicate via an *Agent-to-Agent* channel to exchange *capabilities*, *cost/latency budgets*, *privacy constraints*, *task state*, and *partial artifacts*. Common patterns include _negotiate-and-delegate_, _ask-for-help with evidence_, _propose/accept plan updates_, and _critique-revise_ cycles [A2A].

* *MCP Tooling (Edge and Cloud).* Both models use the *Model Context Protocol* to discover and invoke tools with consistent schemas (e.g., local sensors/files, calculators, vector indexes on edge; search/crawling, KB/RAG, Python/services in cloud). MCP enables capability discovery, streaming/progress, cancellation, and explicit consent prompts across transports [MCP].

* *Policy, Budget, and Privacy Controls.* Guardrails and policies that encode user/enterprise constraints (e.g., do not send raw PII to cloud; enforce token/time budgets; require consent for specific tools). The edge model may redact or summarize data before escalation; both sides log provenance and decisions for auditability.

* *Shared Task State and Provenance.* A compact state (goals, sub-tasks, citations, hashes, timestamps) that both models can read/update to enable reproducibility, debugging, and verifiable traces.


User / Client
(Goal, Query, Constraints)
On-Device Model (Edge)
- few-B params; low latency, privacy by default
- local reasoning, redaction/summarization
- local tools via MCP (sensors, files, crypto)
local MCP tools
when escalation is needed
Edge MCP Tools
A2A Session (Edge <-> Cloud)
- capability/budget exchange
- task handoff & updates
Cloud Model (Hosted)
- 100B1T+ params; higher capability & breadth
- complex synthesis, long-context reasoning
- cloud tools via MCP (search, KB/RAG, Python)
cloud MCP tool calls
Web Search & Crawl
KB / RAG Index
Python / Services
results/evidence via A2A to edge/cloud
Final Answer / Output
(synthesis + citations + privacy/consent notes)

Each building block in the Hybrid AI architecture represents a logical function rather than a specific implementation, and components may be co-located or distributed in practice.

4.2.2. Interaction Model

A typical Hybrid AI session proceeds as follows:

1. *Local First.* The on-device model interprets the user goal, applies local tools (e.g., retrieve snippets, parse files), and attempts a low-cost solution within configured budgets.

2. *Escalate with Minimization.* If the local model estimates insufficient capability (e.g., confidence below threshold, missing evidence), it *redacts/summarizes* sensitive data and *escalates* the task—along with compact evidence and constraints—over *[A2A]*.

3. *Cloud Reasoning + Tools.* The cloud model performs deeper reasoning and may invoke *[MCP]* tools (web search/crawl, KB/RAG, Python) to gather evidence and compute results.

4. *Refine & Return.* Intermediate artifacts and rationales flow back over *[A2A]*. The edge model may integrate results, perform final checks, and produce the user-facing output.

5. *Iterate as Needed.* The models repeat plan-act-observe-refine until success criteria (quality, coverage, cost/time budget) are met.

4.2.3. Why This Use Case Matters

Hybrid AI is inherently _trade-off aware_: it balances *privacy*, *latency*, and *cost* at the edge with *capability* and *breadth* in the cloud. Without standard protocols, inter-model negotiations and tool interactions become bespoke and hard to audit. Two complementary interoperability layers are especially relevant:

* *Inter-Model Coordination (A2A).* A2A provides a structured channel for *capability advertisement*, *budget negotiation*, *task handoffs*, *progress updates*, and *critique/revision* between edge and cloud models. This enables portable escalation policies (e.g., “do not send raw PII”, “cap tokens/time per turn”, “require human consent for external web calls”) and consistent recovery behaviors across vendors [A2A].

* *Tool Invocation (MCP).* MCP standardizes tool discovery and invocation across both environments (edge and cloud), supporting consistent schemas, streaming/progress, cancellation, and explicit consent prompts. This allows implementers to swap local or remote tools—search, crawling, KB/RAG, Python/services—without rewriting agent logic or weakening privacy controls [MCP].

Implications for Hybrid AI. Using standardized protocols lets implementers compose portable edgecloud stacks:

* Edge-first operation with *escalation* only when needed, guided by budgets and confidence.

* *Data minimization* (local redaction/summarization) and *consent* workflows at protocol boundaries.

* Consistent *provenance* (URIs, hashes, timestamps) and *observability* across edge and cloud for verifiable traces.

* Seamless *tool portability* (local/remote) and *policy enforcement* that travel with the task rather than the deployment.

4.3. AI-based Troubleshooting and Automation

Telecom networks have significantly increased in scale, complexity, and heterogeneity. The interplay of technologies such as Software-Defined Networking (SDN), virtualization, cloud-native architectures, network slicing, and 5G/6G systems has made infrastructures highly dynamic. While these innovations provide flexibility and service agility, they also introduce substantial operational challenges, particularly in fault detection, diagnosis, and resolution.

Traditional troubleshooting methods, based on rule engines, static thresholds, correlation mechanisms, and manual expertise, struggle to process high-dimensional telemetry, multi-layer dependencies, and rapidly evolving conditions. Consequently, mean time to detect (MTTD) and mean time to repair (MTTR) may increase, impacting service reliability and user experience.

Artificial Intelligence (AI) and Machine Learning (ML) offer new capabilities to enhance troubleshooting. AI-driven approaches apply data-driven models and automated reasoning to detect anomalies, determine root causes, predict failures, and recommend or execute corrective actions, leveraging telemetry, logs, configuration, topology, and historical data.

Beyond troubleshooting, it is essential to further exploit network and service automation to enable coordinated, policy-based actions across multi-technology (e.g., RAN, IP, optical, virtualized), multi-layer, and dynamic environments. As degradations and faults often span multiple devices, domains, and layers, effective handling requires intelligent and increasingly autonomous mechanisms, ranging from proactive service assurance to automated fault-triggered workflows.

This use case envisions a multi-agent AI framework that enhances network and service automation. Agents perform diagnosis and root cause analysis (RCA), while also supporting anomaly prediction, intent-based protection, and policy-driven remediation. The proposed multi-agent interworking autonomously maintains the network in an optimal operational state by correlating heterogeneous data sources, performing collaborative reasoning, and interacting with network elements and operators through standardized protocols, APIs, and natural language interfaces.

AI agents form a distributed and scalable ecosystem leveraging advanced AI/ML, including generative AI (Gen-AI), combined with domain expertise to accelerate RCA, assess impact, and propose corrective actions. Each agent encapsulates capabilities such as data retrieval, hypothesis generation, validation, causal analysis, and action recommendation. Designed as composable and interoperable building blocks, agents operate across diverse domains (e.g., RAN, Core, IP, Optical, and virtualized infrastructures), while supporting lifecycle management, knowledge bases, and standardized interfaces.

4.3.1. Building Blocks

The use case relies on decentralized multi-agent coordination, where agents exchange structured, context-enriched information to enable dynamic activation and collaborative troubleshooting workflows. A resource-aware orchestration layer manages agent deployment, scaling, and optimization across the networkcloud continuum. Policy frameworks ensure security, compliance, trustworthiness, and explainability, supporting resilient AI-driven network operations.

4.3.2. Why This Use Case Matters

This use case highlights the need for interoperable, protocol-based integration of AI-driven troubleshooting and automation components within heterogeneous, multi-vendor environments. Telecom networks are inherently composed of equipment and control systems from different providers, spanning multiple administrative and technological domains. A multi-agent AI framework operating across such environments requires standardized mechanisms for data modeling,

telemetry export, capability advertisement, and control interfaces. In particular, consistent information models (e.g., YANG-based models), secure transport protocols, and well-defined APIs are needed to ensure that AI agents can reliably discover, interpret, and act upon network state information across vendor boundaries.

Service discovery and capability negotiation are also critical. AI agents must be able to dynamically identify available data sources, peer agents, orchestration functions, and control points, as well as understand their supported features and policy constraints. This implies the need for standardized discovery procedures, metadata descriptions, and context exchange formats that enable composability and coordinated workflows in decentralized environments. Without such interoperability mechanisms, multi-agent troubleshooting systems risk becoming vertically integrated and operationally siloed.

Furthermore, governance, security, and trust frameworks are fundamental considerations. AI-driven agents capable of recommending or executing remediation actions introduce new requirements for authentication, authorization, accountability, and auditability. Secure communication channels, role-based access control, policy enforcement, and explainability mechanisms are necessary to prevent misuse, contain faults, and ensure compliance with operational and regulatory constraints.

4.4. AI-Based Operation Models

4.4.1. Agentic AI for Improved User Experience

AI agents have the potential to enhance future user experience by being integrated—individually or as collaborating groups—into telecom networks to deliver user-facing services. Such services may include autonomous multi-level Internet/Intranet search, coordination of calendar and email tasks, and execution of multi-step workflows involving multiple agents, as well as pre-built domain agents (e.g., HR, procurement, finance). This shift can fundamentally change enterprise operating models: agents can support decision-making and, where authorized, act on behalf of employees or the organization. In multi-agent scenarios, agents from different vendors communicate over networks and must interoperate. These interactions require coordinated communication flows and motivate a standardized agent communication protocol and framework. Given the need to comply with regulatory requirements (beyond network regulation), an open, standardized approach is preferable to proprietary implementations. Interoperability across operators and vendors implies an open ecosystem; therefore, a standardized AI agent protocol is required.

4.4.1.1. Building Blocks

TODO

4.4.1.2. Why this Use Case Matters

TODO

4.4.2. Voice-Based Human-to-Agent Communication

With the integration of AI and AI agents into networks, voice services may see renewed importance as a natural, low-friction interface for interacting with agents. Voice-based human-to-agent communication can complement text-based chat interfaces and enable rapid task initiation and conversational control. This use case introduces additional considerations, including security, authorization/permissions, and charging/accounting. Because voice services are regulated in many jurisdictions, this further motivates a standardized framework and standardized AI agent protocol. Network-integrated AI agents can assist users through voice interaction and improve overall user experience.

4.4.2.1. Building Blocks

TODO

4.4.2.2. Why this Use Case Matters

TODO

5. Security Considerations

TODO Security

6. IANA Considerations

This document has no IANA actions.

7. References

7.1. 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>.

7.2. Informative References

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

[A2A-GITHUB] "Agent2Agent Protocol GitHub Repository", n.d., <https://github.com/a2aproject/A2A>.

[MCP] "Model Context Protocol (MCP) Specification", March 2025, <https://modelcontextprotocol.io/specification/2025-03-26>.

[MCP-GITHUB] "Model Context Protocol GitHub Organization", n.d., <https://github.com/modelcontextprotocol>.

[ODS] "Open Deep Search", 2025, <https://arxiv.org/abs/2503.20201>.

[ODS-GITHUB] "OpenDeepSearch", n.d., <https://github.com/sentient-agi/OpenDeepSearch>.

Acknowledgments

TODO acknowledge.

Authors' Addresses

Roland Schott
Deutsche Telekom
Email: Roland.Schott@telekom.de

Julien Maisonneuve
Nokia
Email: julien.maisonneuve@nokia.com

L. M. Contreras
Telefonica
Email: luismiguel.contrerasmurillo@telefonica.com

Jordi Ros-Giralt
Qualcomm Europe, Inc.
Email: jros@qti.qualcomm.com