Executive Summary

LLM Agents and protocols such as the Model Context Protocol (MCP) and Agent2Agent (A2A) are quickly becoming a hot-button topics of conversation especially with newer communication protocols (for both tools individually and agents themselves) coming out regularly for more questions to be asked and answered about how they strategically play a role within SnapLogic's ecosystem(s). These protocols aim to make the issue of interoperability of different ecosystems of Tools, LLMs, and Agents to align with a cross-functional cohesive experience for further agentic processing.

This document will summarize:

1. The LLM Protocols and their evolution
2. SnapLogic perspective on these LLM Protocols
3. FAQ for MCP and A2A
4. Enterprise Inter-Agent Collaboration Vision
5. Architecture for Status Reporting use-case and how to build to it

LLM Protocols

Model Context Protocol (MCP)

Anthropic initially developed the Model Communication Protocol (MCP) in November 2024 to enable third-party tool integration with Claude Desktop through a standardized protocol. While initially intended for Claude Desktop prior to its public release, its scope was broadened to become an open-source protocol and specification. This allows for providing tools to any Large Language Model (LLM) in a predefined manner, independent of individual LLM API specifications. Early implementations primarily support STDIO-based MCP Servers running locally with Claude Desktop, with future plans to include HTTP-based MCP Servers utilizing HTTP Server-Sent Events (SSE). The second version of the protocol, released in mid-April 2025, introduced a new Streamable-HTTP transport protocol for HTTP-based MCP Servers, while retaining SSE as a fallback mechanism. This version also considered authorization for HTTP-based MCP Servers.

The MCP specification defines a server providing tools, resources, and prompts to connected clients. It details the data format and possible protocols for communicating tool listings and invoking tools, but the transport mechanism is left to the implementers of both the MCP server and client. The proposed standards for transport are standard I/O and HTTP (using SSE and Streamable HTTP, both asynchronous methods). A key focus of MCP and its specified transport layers is the streaming of data from server to client for near real-time updates during processes like tool calls or resource retrieval.

MCP development predominantly utilizes TypeScript, JavaScript, and Python, primarily through stdio communication due to most MCP clients operating locally. Consequently, the HTTP-based transport layers and the handling of remote MCP use cases have experienced significant changes and ongoing adjustments.

Agent2Agent (A2A)

In April 2025, Google introduced the Agent2Agent Protocol (A2A) which defines a communication specification to an Agent Framework to standardize how agents interact with a number of partners. While A2A builds upon tool-calling communication, it also addresses unique agent capabilities and needs, such as providing a complete log of agent messages and providing other artifacts. This detailed context is valuable for an orchestrating agent to understand different parts of a process. The A2A specification offers tool-like guidelines for agent operation and provides structured response details for newly created artifacts as the agent processes, going beyond the limitations of plaintext communication common in most LLM APIs.

The Agent2Agent (A2A) protocol emphasizes an "Agent Framework" where coordinating systems can find and communicate with other agents. Registered agents within this framework expose their capabilities, allowing other agents to access and utilize them. Essentially, A2A facilitates inter-agent communication and tool discovery through a registration mechanism and then has a transport layer to expose the agents in a more standard way.

The A2A protocol is currently in functional draft status. SDKs and a complete specification are defined, with initial support outlined for Python and JavaScript.

SnapLogic perspective

MCP and A2A are early specifications designed to enhance the capabilities of LLM-based agents by enabling them to interact with a wider range of tools and accomplish more complex tasks. Unlike single-tool specification layers, these protocols facilitate multi-agent systems where individual agents specialize in specific tasks, leading to improved overall agent performance. Google's comparison highlights the intended use cases for each system and suggests that MCP could potentially support A2A though supporting A2A natively provides the far greater flexibility needed by many agents. The document indicates that A2A is particularly valuable in scenarios requiring greater flexibility and collaboration among sub-agents on a unified task.

SnapLogic prioritizes remote interactions for A2A and MCP due to its remote data plane. Currently, interacting with SnapLogic via these protocols requires exposing tools and agents over HTTP. SnapLogic is developing the ability to call remote MCP servers and plans to enable registering MCP servers within SnapLogic as pipelines or other connected systems. While actively monitoring the A2A landscape for support requirements, SnapLogic anticipates that its ongoing MCP support will address many of the technological needs for A2A, potentially making A2A support an adaptation of the MCP solution implementation-wise. The broader utility of the Agent Framework is also under consideration.

We have MCP support in beta as part of our SnapLabs R&D, enabling any collection of SnapLogic pipelines or SnapLogic APIM-managed APIs to be used as tools natively. We plan to also make it possible to launch and manage any MCP Server securely within SnapLogic hosted in a customer's own domain on the dataplane directly. Finally, SnapLogic Agents can both use MCP Servers and become a tool within a deployed MCP Server on SnapLogic.

We have supported high performance tool calling within agents natively since October '24. We are actively working to support MCP, but the protocol is missing required enterprise features and is still evolving too rapidly to rely on it in production (described as "immature" by Gartner in late April 2025). If customers or prospects have specific questions, we can set up a call with the AI engineering team to investigate further.

FAQ for MCP and A2A

What is the role of a MCP Server?

Model Context Protocol (MCP) Servers are lightweight programs that provide specific functionalities via the MCP. These servers expose tools accessible to LLM agents, including those developed with SnapLogic AgentCreator. MCP Servers can interface with Local Data Sources or Services (local files, databases, and services) or Remote Services (external systems accessible online via APIs).

What is the role of a MCP Client?

MCP Hosts (Agents) utilize MCP Clients to interact with MCP Servers, requesting data or actions. These clients act as intermediaries, forwarding LLM requests to the servers. In response, MCP Servers grant access to the requested data and capabilities. Essentially, MCP Clients enable MCP Hosts to leverage the functionalities offered by MCP Servers.

Enterprise Inter-Agent Collaboration Vision

Vision: An Open, Interoperable Agentic Ecosystem for the Enterprise

Goal: Enable enterprise systems and teams to collaborate through AI agents that interoperate across domains, platforms, and governance boundaries—without platform lock-in.

Key Principles

- **Domain-first, not platform-first.**
  Agents should serve business outcomes, not be constrained by specific tools.

- **Composable architecture.**
  Each agent acts as a reusable, modular service. Teams can orchestrate these across platforms (SnapLogic, LangChain, MCP, etc.).

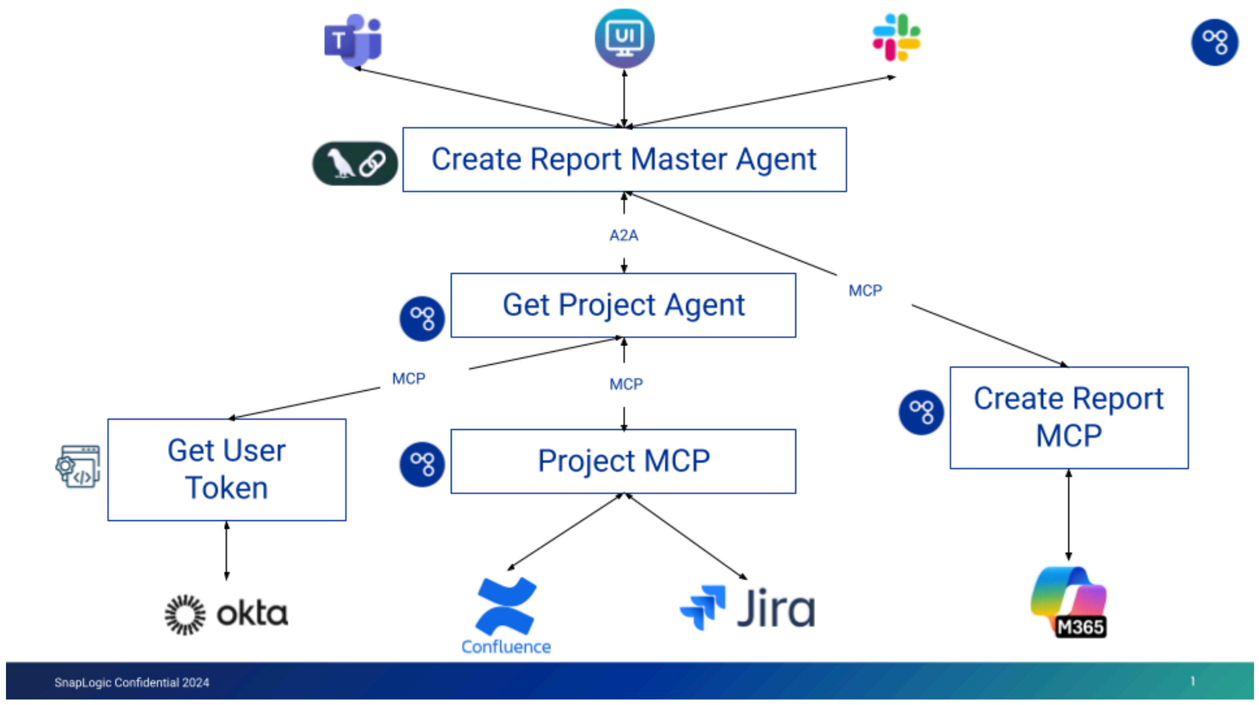
- **Inter-agent communication via open protocols.**
  Use emerging standards like MCP and A2A to enable agents to invoke one another across environments.
  Example: A SnapLogic integration agent calling a LangGraph data science agent.

- **Human-in-the-loop and out-of-the-loop.**
  Design agents with escalation and feedback pathways. Support manual review and approval early on, with a roadmap to autonomy. Some of this is included as part of the A2A spec already, but may be missing key pieces for orchestration within SnapLogic

- **Single source of truth for connectors.**
  Avoid connector sprawl. Route all data access through centralized integration layers (API gateway + SnapLogic), ensuring security and reusability.

- **Security-first, not service-account-first.**
  Agents must respect user-level access control. No backdoor escalations via shared service tokens. Example: Confluence reads scoped to Okta-authenticated user.

Architecture for Status Reporting Agent



This diagram outlines a future architecture for the Status Reporting Agent. Initially, due to security and privacy considerations, interactions with external tools and agents will be through direct tool calls from SnapLogic pipelines, aligning with current protocol maturity. As protocols evolve with enterprise-grade support for security and authorization, they can be integrated into the broader architecture.

For instance, while an Agent2Agent (A2A) interaction between the Create Report Master Agent and the Get Project Agent is envisioned given a stronger Agent Framework in SIE, this could temporarily be a direct tool call to the Get Project Agent or utilize a local MCP Server that invokes a SnapLogic Triggered or Ultra Task. Once a secure and private remote HTTP-based A2A communication is established, it can replace the interim solution.

Similarly, the Project MCP functionality could start as individual tool calls within SnapLogic, leveraging the Multi-Pipeline Function Generator. This can transition to SnapLogic acting as an MCP Server when that capability becomes available.

Long-term, for agents developed within SnapLogic (within the SnapLogic ecosystem, utilizing SnapLogic's native capabilities), supported by the May 2025 release of APIM 3.0 (Beta), Pipelines and OpenAPI Spec is the preferred approach for managing tools and Agents within SnapLogic or when interacting with external systems, rather than relying on MCP. This approach addresses the tools and agents entirely within the SnapLogic ecosystem or when calling out to external systems.