MANATECH

RESEARCH REPORT

Understanding the Model Context Protocol (MCP): The Universal Interface for AI Agency

Executive Summary

The Model Context Protocol (MCP) represents a fundamental shift in the artificial intelligence landscape, moving from isolated Large Language Models (LLMs) to "agentic" systems capable of interacting with the real world. Developed by Anthropic and released in late 2024, MCP is an open standard that functions as a "universal adapter" (frequently compared to USB-C) for AI. It standardizes the connection between AI models and the tools, data sources, and services they require to perform meaningful work.

Prior to MCP, integrating an LLM with external systems like Jira, Slack, or a local database required fragmented, custom "glue code" for every unique connection. MCP replaces this complexity with a standardized client-server architecture. By 2026, MCP is projected to become the backbone of enterprise AI, evolving to support multimodal data (audio, video, images), hierarchical agent orchestration, and open governance. While competing protocols exist, MCP has gained significant traction and adoption by industry leaders including Microsoft, Google, and Red Hat.

Detailed Analysis of Key Themes

1. The Core Architecture: Host, Client, and Server

MCP operates on a tripartite structure that decouples the AI orchestration from the specific implementation of tools and data access:

- **The Host:** The primary AI application (e.g., Claude Desktop, VS Code, a custom microservice) that houses the AI model and manages user interactions.
- **The Client:** A component living inside the host that initiates connections to servers and maintains a 1-to-1 connection through the MCP protocol.
- **The Server:** A lightweight program that exposes specific capabilities (tools, resources, or prompts) to the client. Servers can run locally on a machine or remotely in the cloud.

Communication and Transports

MCP uses **JSON-RPC 2.0** for messaging. The communication method, known as "transport," varies based on the environment:

- **Local Transport (stdio):** Uses standard input/output pipes for communication between processes on the same machine. This offers low latency and no network overhead.

- **Remote Transport (HTTP + SSE or Streamable HTTP):** Uses Server-Sent Events (SSE) or streamable HTTP for cloud-based connections. Streamable HTTP is increasingly preferred as it supports both stateful and stateless interactions, making it easier to deploy in complex networking environments.

2. Primary Functional Components

MCP servers provide three main types of capabilities to an AI agent:

Component	Definition	Example Use Case
Tools	Executable functions that the model can invoke to perform an action.	Creating a Jira ticket, executing a Python script, or sending an email.
Resources	Read-only data that provides context to the model.	Reading a local Markdown file, checking logs, or querying a database.
Prompt Templates	Pre-defined, structured blueprints for interacting with a server.	A "Analyze Spreadsheet" template that includes best-practice instructions for the LLM.

Trend Note: Data suggests that many developers are currently favoring the **Tools API** over the **Resources API**. Instead of querying a resource, they use a tool call (like a method call with parameters) to fetch data, finding it a more flexible way to interact with dynamic information.

3. MCP vs. AI "Skills"

A point of tension in the developer community is the emergence of "Skills" alongside MCP. While they overlap, they serve different primary orientations:

- **Skills:** Essentially Markdown files (containing instructions) accompanied by local directories of scripts (Bash, Python) or static reference data. Skills are highly localized, living in the file system where the agent can see them. They are ideal for local coding environments like Claude Code.
- **MCP:** Focused on a standardized interface for connecting to remote or cloud-based resources. MCP is necessary for "agentic microservices" that need to scale beyond a single laptop and require a robust, stateful communication protocol.

4. Enterprise Governance and Implementation

Large-scale organizations are adopting MCP to bridge the gap between "sandbox" experimentation and production-ready AI.

Red Hat's Framework for Enterprise MCP

Enterprise implementation requires a structured life cycle:

1. **Registry:** A secure staging zone where MCP servers are scanned, quarantined, and certified.
2. **Catalog:** A curated list of approved, enterprise-ready MCP servers for AI engineers to consume.
3. **Gateway:** A runtime enforcement layer that applies Policy and Role-Based Access Control (RBAC) to tool calls, ensuring agents do not exceed their authority.

Permission Models in Business Central

Microsoft's implementation in Dynamics 365 Business Central highlights the transition from read-only to write-access. Default configurations are strictly read-only to prevent agents from inadvertently altering data. "Named Configurations" allow administrators to explicitly grant permissions for "Create," "Update," or "Delete" operations on specific API pages once trust in the model's reasoning is established.

Important Quotes and Context

On the Rapid Evolution of the Landscape

> *"This is without a doubt, the most rapidly evolving landscape I have seen in more than 30 years of software engineering." Context: An observation on how quickly the MCP and AI agent ecosystem is shifting, making continuous monitoring essential for developers.*

On the Necessity of Standardization

> *"Instead of every app building tool integration differently, MCP creates a common language between the model and the tool layer. It's like a USBC moment for AI tooling." Context: Explaining how MCP eliminates the need for "snowflake" integrations—where every connection is unique and fragile—by creating a plug-and-play infrastructure.*

On the Business Implications

> *"AI isn't coming for your job; it's coming for your customers... If you're not actively engaging and making sure that this distribution channel... [is] there grabbing the attention, someone else will." Context: Highlighting MCP as a new distribution channel. If a company does not provide an MCP server for its services, users may gravitate toward competitors whose tools are easier for AI agents to discover and use.*

On Security and Control

> *"Access is mediated by servers... so there's a human behind that server... enforcing authentication, authorization, logging, rate limits, and policy checks... we are not just letting it go and do whatever it wants." Context: Addressing concerns about agent autonomy by emphasizing that the MCP server acts as a controlled gateway between the AI and sensitive data.*

Security and Trust Framework

MCP incorporates several layers of protection to facilitate enterprise adoption:

- **User Consent:** No tool call or resource access occurs without explicit user approval within the host application.

- **Authentication Standards:** Support for OAuth 2.1 allows users to have familiar login experiences (e.g., a GitHub pop-up) when an agent requires access to their accounts.
- **Sandboxing:** Local MCP servers (often run via Docker) provide an isolated environment, ensuring that if an agent performs an action, it is contained within specified boundaries.
- **Auditability:** Standardized logging allows organizations to track every tool call and its output, which is critical for regulated industries like FinTech and Healthcare.

Actionable Insights

For Developers and AI Engineers

- **Start Small with Workflows:** Do not attempt to build a complex multi-tool agent immediately. Begin by creating an MCP server for a simple "Research + Ticket" pipeline—pulling info from a doc and creating a structured Jira ticket.
- **Leverage Existing Ecosystems:** Before building from scratch, explore the thousands of pre-built MCP servers available for PostgreSQL, Google Drive, Slack, and GitHub.
- **Utilize Docker for Local Servers:** Use the Docker MCP Toolkit to run servers. This abstracts the environmental setup and allows for easier management of secrets (API keys/tokens) outside of the raw code.

For Organizations and Product Managers

- **Identify the "Write" Scenarios:** Start with low-risk, read-only scenarios (e.g., status checks). As the system proves reliable, identify high-value "Write" operations (e.g., updating a CRM or initiating a deployment) to expose via MCP.
- **Develop an AI Service Catalog:** Centralize approved MCP servers. This prevents "shadow AI" and ensures that all agents in the company are using secure, certified versions of tools.
- **Consider MCP as a Distribution Channel:** For B2B or SaaS companies, building an official MCP server for your product ensures your service is "visible" to the growing number of users navigating the web via AI agents and IDs.

For Security Professionals

- **Implement "Least Privilege" Access:** Only grant an MCP server the specific permissions it needs for its intended tasks.
- **Monitor the 2026 Roadmap:** Prepare for multimodal support. Security policies will need to expand to cover how agents process and transmit images, audio, and video via MCP.

Want to explore this topic further?

Book a free discovery call to discuss how ManaTech can help your business implement these ideas.

Book a Discovery Call