Network Working Group
Internet-Draft
Intended status: Informational
Expires: 6 May 2026

G. Zeng
J. Mao
B. Liu
N. Geng
X. Shang
Q. Gao
Z. Li
Huawei

2 November 2025

Using the Model Context Protocol (MCP) for Intent-Based Network Troubleshooting Automation
draft-zm-rtgwg-mcp-troubleshooting-01

Abstract

The Model Context Protocol (MCP) is an open standard that enables Large Language Model (LLM) applications to seamlessly integrate with external data sources and tools by exposing Resources, Prompts and Tools in a JSON-RPC 2.0 transport. This document describes a mapping of MCP roles, primitives and security model to the network management domain so that network devices act as MCP servers and network controllers act as MCP clients. This document also extends the model to Device-to-Device (D2D) collaboration, allowing network elements to perform distributed fault correlation when the controller is unreachable or when real-time cross-device data is required. The goal is to provide an intent-based, conversational and secure approach for automated network troubleshooting, configuration validation, and closed-loop remediation without inventing new protocols or device agents.

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 6 May 2026.

Table of Contents

1. Introduction

Network operators today face two converging demands: (1) reduce Mean Time to Repair (MTTR) while managing ever larger infrastructures, and (2) adopt intent-based interfaces that allow engineers to express high-level goals such as "verify reachability between Site-A and Site-B" instead of typing low-level CLI commands.

Simultaneously, Large Language Models (LLMs) have demonstrated utility in reasoning about semi-structured data such as device logs, configurations, and command outputs. However, safely exposing device-level actions and data to an LLM in real time remains an open problem. The Model Context Protocol (MCP), developed by Anthropic and published at https://modelcontextprotocol.io, provides a lightweight, capability-oriented RPC layer that already addresses this problem for general LLM applications.

This document specifies a deterministic mapping of MCP roles, primitives, and security workflows onto the network management plane so that:

* A network element (router, switch, firewall, etc.) becomes an "MCP server" that exposes management data and actions as MCP Resources, Prompts and Tools.

* A controller, orchestrator or chat-based assistant becomes an "MCP client" that consumes these primitives.

* Human operators interact with the controller using natural language; the controller translates the intent into a sequence of MCP calls, optionally consulting an LLM for reasoning.

* All interactions are subject to explicit user consent, audit, and capability-based access control, as mandated by MCP.

While the star-shaped Controller-to-Device model covers most brown-field deployments, some faults are visible only when two or more devices compare live data in real time. To address this we extend MCP to support Device-to-Device (D2D) troubleshooting collaboration. In this mode network elements autonomously form a transient "collaboration domain", exchange YANG/JSON-RPC calls, correlate results with an on-box LLM, and produce a signed report that can be retrieved by the controller once connectivity is restored. Security is maintained through mutual TLS, short-lived device certificates, and a white-list of neighbour-callable capabilities.

The result is an intent-based, conversational and secure automation framework that re-uses existing agents (NETCONF/RESTCONF/YANG, SNMP, CLI, gNMI, etc.) already present on devices instead of requiring new firmware.

2. Terminology

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.

MCP terms such as "Tool", "Resource", "Prompt", "Client", "Server", "Host", and "Capability" are used as defined in the MCP specification dated 2025-06-18.

Intent: A high-level, declarative statement of desired network behaviour, expressed in natural language or structured YAML/JSON, that the controller must translate into concrete device actions.

3. Mapping of MCP Primitives to Network Management

3.1. Resources

Resources are exposed under the URI scheme mcp://<device>/<yang-module>:<path>. Reading a resource returns YANG JSON-encoded data per [RFC7951]. Examples:

* mcp://router1/ietf-interfaces:interfaces/interface=eth0

* mcp://router1/openconfig-bgp:bgp/neighbors/neighbor=1.1.1.1

Servers SHOULD support the "content-id" header to enable E-tags for caching.

3.2. Tools

Tools are mapped to well-known RPC operations already exposed by devices via NETCONF/RESTCONF/YANG, gNMI, or CLI. A Tool is described by an OpenAPI 3.0 Operation Object and MUST be idempotent when possible.

Example Tool schema (ping):

```
{
  "name": "ping",
  "description": "Execute ICMP echo probe",
  "inputSchema": {
    "type": "object",
    "properties": {
      "destination": { "type": "string" },
      "count":       { "type": "integer", "default": 5 },
      "source":      { "type": "string" }
    },
    "required": ["destination"]
  }
}
```

Figure 1: Tool Schema Example

3.3. Prompts

Prompts are reusable prompt templates stored on the device. They allow vendors or operators to encode golden troubleshooting workflows in natural language. A prompt MAY contain variable placeholders such as "{{interface}}" that the client fills in before sending to an LLM.

3.4. Sampling (Optional)

If the client advertises the "sampling" capability, the server MAY request LLM inference on behalf of the device. This is useful for recursive troubleshooting where the device needs to ask clarifying questions. All sampling requests MUST be approved by the human operator via explicit consent UI.

4. Controller-to-Device Troubleshooting

4.1. Architecture

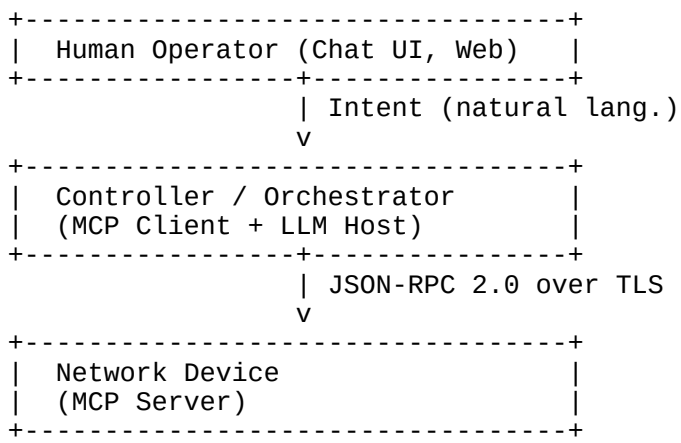

Figure 2: Controller-to-Device model

4.2. Role Allocation

MCP Server: Runs on or proxied in front of the network element. Exposes:

* Resources: Read-only YANG datastores, syslog, tech-support

* Tools: Idempotent actions, e.g., "ping", "traceroute", "clear counters", "rollback config"

* Prompts: Re-usable prompt templates, e.g., "Diagnose BGP session down"

MCP Client: Runs inside the controller. Maintains a persistent JSON-RPC 2.0 connection to each server. Optionally hosts an LLM that reasons about the data returned by servers.

4.3. Capability Negotiation

On connection establishment, the client and server exchange capability objects as defined in MCP. Servers list their supported YANG modules [RFC8525], CLI command sets, and Tool schemas encoded in OpenAPI 3.0. Clients list optional features such as "sampling" (LLM recursion) or "roots" (URI scoping).

4.4. Example Use Cases

4.4.1. Intent: "Verify reachability between Site-A and Site-B"

The following steps illustrate the flow:

1. Operator enters intent in chat UI.

2. Controller’s LLM deduces required Tools: – ping (Tool) from router1 to 10.2.2.2
   – show interfaces (Resource) on router2

3. Controller issues MCP calls.

4. Devices return results.

5. LLM summarizes: "Packet loss 0%; MTU mismatch detected on router2 ge-0/0/0. Recommend ’set interfaces ge-0/0/0 mtu 1500’."

4.4.2. Intent: "Diagnose why BGP neighbor 1.1.1.1 is down"

1. Controller retrieves: -/openconfig-bgp:bgp/neighbors/neighbor=1.1.1.1/state -/ietf-interfaces:interfaces/interface=loopback0

2. Controller calls Tool "tcpdump" filtered on port 179.

3. LLM correlates: "No TCP SYN received; ACL foo on interface loopback0 denies port 179."

4. Controller offers one-click remediation: remove ACL entry.

5. Device-to-Device Troubleshooting Collaboration

Some root causes are scattered across several nodes (e.g., unidirectional fiber, one-way ACL, single-sided BFD Down, localized SRv6 SID failure). Although the controller can collect data centrally, the north-bound link may be impaired, time-synchronisation is costly, and uploading bulk data is expensive. Allowing nearby devices to form a transient "trusted collaboration domain" and exchange data, correlate and infer root causes locally can significantly shorten MTTR.

5.1. Architecture

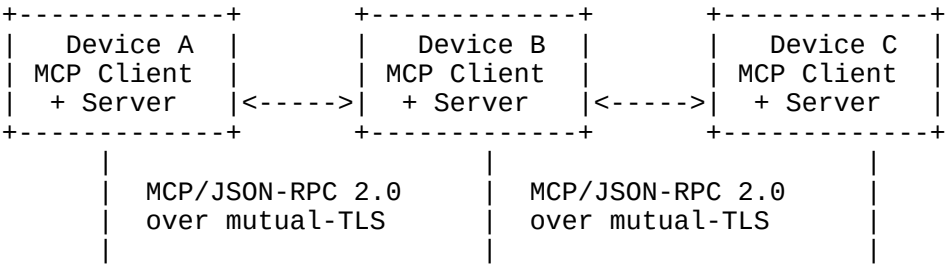

Figure 3: Device-to-Device Collaboration Model

5.2. Role Allocation

* MCP-Server, MCP-Client: moved into a "Collaboration Agent" (CA) running on the device. Every CA is both client and server; the MCP message layer MUST support mutual TLS and capability negotiation.

* The user (human or controller) only needs to inject an Intent on any one device in the domain; the CA will then perform the chained calls autonomously and roll-up the final report.

5.3. Example Use Cases

5.3.1. Intent: Verify packet loss on the SRv6 path from PE-1 to PE-3 via P-2

The following steps illustrate SRv6 packet-loss localization between three routers: PE-1 (head-end), P-2 (mid-point), and PE-3 (tail-end). The same pattern can be applied to any multi-box fault domain.

Step-0: Intent Injection

An operator types the following natural-language goal in the chat window that is served by PE-1:

Intent: "Verify packet loss on the SRv6 path from PE-1 to PE-3 via P-2"

PE-1 Collaboration Agent (CA) parses the Intent, extracts the SID list {PE-1, P-2, PE-3}, and starts the D2D workflow.

Step-1: Collaboration Domain Discovery

PE-1 CA sends an LLDP/IS-IS Extended TLV that contains:

```
    + mcp-d2d-port = 9514
    + supported-capabilities = [ietf-srv6-ping, ietf-interface-counters]
    + cert-thumbprint = <SHA-256 of device certificate>
```

P-2 and PE-3 reply with their own TLV. All three nodes are now aware of each other's MCP endpoint and capabilities.

Step-2: Mutual TLS & Capability Negotiation

PE-1 opens a TLS 1.3 connection to P-2 and PE-3 on port 9514. Both sides send their X.509 device certificates and the MCP initialize message:

```
{ "jsonrpc": "2.0",
  "method": "mcp/initialize",
  "params": {
      "protocolVersion": "2025-12",
      "capabilities": [ "ietf-srv6-ping", "ietf-interface-counters" ]
  },
  "id": 1 }
```

The responder echoes its own capability list. If the intersection is non-empty the session is marked authorised for those capabilities.

Step-3: Parallel Resource & Tool Calls

PE-1 CA schedules three operations in parallel:

Local call (no network RPC)

```
  Tool: "ietf-srv6-ping"
  Params: { Head=PE-1, Tail=PE-3, SID-List=[P-2, PE-3], Count=100 }
```

RPC toward P-2

```
  POST https://[P-2]:9514/mcp/tool/call
  Body: { tool: "ietf-srv6-ping",
          arguments: { Local=P-2, Tail=PE-3, Count=100 } }
```

RPC toward PE-3

```
POST https://[PE-3]:9514/mcp/resource/read
Body: { uri: "urn:ietf:params:xml:ns:yang:ietf-interfaces/interfaces/interface=SID-Endp
oint/statistics" }
```

Each callee returns a JSON-RPC result plus an ed25519 signature covering the result body and a monotonic nonce. The nonce prevents replay if the controller later fetches the audit-log.

Step-4: Local Correlation & Inference

PE-1 CA feeds the three data sets into its on-box LLM with the prompt:

"Compare loss % from PE-1 vs P-2; if PE-1>0 and P-2=0, root cause is in {PE-1→P-2 link, upstream ACL}; output a single sentence."

Model output:

"Loss 3.2 % observed only on PE-1→P-2 direction; P-2→PE-1 0 %; suggest checking PE-1 egress ACL 2001."

The deterministic part of the reply (ACL 2001) is extracted as a structured fault hypothesis and stored in the candidate list.

Step-5: Roll-up Report & User Consent

PE-1 CA assembles the signed results, inference, and raw packets into an MCP resource:

```
URI:  urn:ietf:params:xml:ns:yang:ietf-mcp/report/74e8
Body: { "creator": "PE-1",
        "created": "2025-10-30T14:23:42Z",
        "hypothesis": "PE-1 egress ACL 2001",
        "evidence": [ <base64 encoded PCAP>, ... ],
        "signatures": { "PE-1": <sig1>, "P-2": <sig2>, "PE-3": <sig3> } }
```

If the operator is still on-line the CA presents the hypothesis and asks for explicit approval before any mitigating action (e.g., edit ACL) is executed. If the controller is reachable the report is pushed through the conventional star channel; otherwise it remains on-box until the next controller sync.

Failure Handling

If any D2D call times out or returns an authentication error, PE-1 CA marks that node "untrusted" and falls back to controller-based polling if available. All intermediate temporary states (e.g., cleared counters) are rolled back immediately to preserve atomicity.

6. Transport & Encoding Considerations

JSON-RPC 2.0 is carried over TLS 1.3 [RFC8446] with TCP/443 or QUIC [RFC9000] as the underlying transport. Servers present X.509 device certificates; clients use mutual TLS with short-lived SPAKE2 [RFC9383] tokens to achieve zero-touch onboarding.

YANG data is encoded in JSON [RFC7951] unless the client explicitly requests XML.

Large tech-support files MAY be streamed via HTTP/2 chunked transfer and are integrity-protected by SHA-256 hashes delivered in the JSON-RPC result.

Servers MAY support the MCP "progress" notification to report percentage completion for long-running Tools such as "tracepath".

7. Security Considerations

This section extends the security model defined in MCP with network-specific requirements.

7.1. User Consent and Authorization

All Tool invocations that change device state MUST be confirmed by an authenticated user via an out-of-band consent channel (e.g., click-through UI or signed JWT). The consent object includes:

* Tool name and parameter values
* Estimated impact window (rollback timer)
* User identity and role
* Signed timestamp

7.2. Least-Privilege Capability Tokens

Servers issue short-lived OAuth 2.0 [RFC6749] access tokens scoped to individual YANG subtrees or Tools. Tokens are bound to the mutual TLS channel to prevent replay.

7.3. LLM Isolation

When the controller hosts an LLM, the LLM is placed in a sandbox with no direct layer-3 reachability to devices. All interactions MUST traverse the MCP client to mitigate prompt-injection attacks.

7.4. Audit and Post-Mortem

Every JSON-RPC request and response is appended to an immutable audit trail (e.g., syslog [RFC5424] or IETF syslog over TLS). Servers include a "session-id" field to allow cross-device correlation.

7.5. Privacy

Resources containing customer data (e.g., DPI logs) are redacted by default. Servers expose a "privacy-level" capability; clients request explicit user consent before retrieving level-3 data.

8. IANA Considerations

This document requests IANA to register the following well-known URI:

* URI suffix: mcp

* Description: Model Context Protocol over TLS

* Reference: This document

Additionally, the YANG module "ietf-mcp" is requested to be added to the IETF YANG module registry with namespace "urn:ietf:params:xml:ns:yang:ietf-mcp".

9. Normative References

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

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, 2016, <https://www.rfc-editor.org/info/rfc7951>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC9000] Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, 2021, <https://www.rfc-editor.org/info/rfc9000>.

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

[RFC8525] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Library", RFC 8525, 2019, <https://www.rfc-editor.org/info/rfc8525>.

[RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, 2009, <https://www.rfc-editor.org/info/rfc5424>.

[RFC6749] Hardt, D., "The OAuth 2.0 Authorization Framework", RFC 6749, 2012, <https://www.rfc-editor.org/info/rfc6749>.

[RFC9383] Sethi, M. and R. Struik, "SPAKE2+, an Augmented Password-Authenticated Key Exchange (PAKE) Protocol", RFC 9383, 2023, <https://www.rfc-editor.org/info/rfc9383>.

10. Informative References

[MCP-SPEC] Anthropic, "Model Context Protocol Specification 2025-06-18", URL https://modelcontextprotocol.io/specification/2025-06-18/basic, 2025.

Appendix A. JSON-RPC Examples

A.1. Client Call: ping

```
--> {
      "jsonrpc": "2.0",
      "id": 1,
      "method": "tools/call",
      "params": {
        "name": "ping",
        "arguments": {
          "destination": "10.2.2.2",
          "count": 5,
          "source": "192.0.2.1"
        }
      }
    }

<-- {
      "jsonrpc": "2.0",
      "id": 1,
      "result": {
        "loss": 0,
        "rtt": { "min": 2.1, "max": 2.3, "avg": 2.2 }
      }
    }
```

Figure 4: ping Request and Response

A.2. Server Resource Subscription

```
--> {
      "jsonrpc": "2.0",
      "id": 2,
      "method": "resources/subscribe",
      "params": {
        "uri": "mcp://router1/ietf-interfaces:interfaces/interface=eth0",
        "content-id": "etag-1234"
      }
    }

<-- {
      "jsonrpc": "2.0",
      "id": 2,
      "result": {
        "subscription-id": "sub-42",
        "initial": { "admin-status": "up", "oper-status": "up" }
      }
    }
```

Figure 5: Resource Subscribe Example

Authors' Addresses

Guanming Zeng
Huawei
Email: zengguanming@huawei.com

Jianwei Mao
Huawei
Email: maojianwei@huawei.com

Bing Liu
Huawei
Email: leo.liubing@huawei.com

Nan Geng
Huawei
Email: gengnan@huawei.com

Xiaotong Shang
Huawei
Email: shangxiaotong@huawei.com

Qiangzhou Gao
Huawei
Email: gaoqiangzhou@huawei.com

Zhenbin Li
Huawei
Email: robinli314@163.com