Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 19 April 2026

Z. Guanming
Huawei Technology

16 October 2025

Model Context Protocol (MCP) Extensions for Network Equipment Management
draft-zeng-mcp-network-mgmt-01

Abstract

The Model Context Protocol (MCP) provides a JSON-RPC 2.0 framework for interaction between AI applications and external context sources. This document specifies minimal extensions that allow network equipment (routers, switches, etc.) to act as MCP servers while controllers act as MCP clients. New capability tokens, tools, resources, prompts, and error codes are defined without breaking existing MCP implementations.

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 19 April 2026.

Copyright Notice

Table of Contents

1. Introduction

Network controllers today need to speak CLI, YANG/NETCONF, SNMP, gNMI, and vendor-private APIs. Implementing a separate adapter for each protocol is expensive and error-prone.

The Model Context Protocol (MCP) already defines a JSON-RPC 2.0 based framing, capability negotiation, and extensible tool/resource model. By adding a small set of network-specific capability flags and tool names, a device can expose its CLI, YANG datastores, and event streams through the same MCP channel that AI applications use for retrieving context.

This document specifies those extensions. All new elements live in their own capability namespace and can be ignored by generic MCP clients, preserving backward compatibility.

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.

2. Terminology

MCP Model Context Protocol

Server MCP server running on the network device

Client MCP client running on the controller

Datastore Conventional YANG datastore (running, candidate, operational)

3. Capability Advertisement

Servers that implement this specification MUST include the following object inside serverCapabilities in the initialize response:

```
"network": {
  "yangModules": ["ietf-interfaces", "openconfig-interfaces"],
  "cliDialect": "cisco-iosxr",
  "configDatastore": ["running", "candidate", "operational"],
  "notificationStream": ["syslog", "netconf-stream", "snmp-trap"],
  "maxBulkEdit": 1000,
  "supportsRollback": true,
  "rollbackTimeout": 300
}
```

Figure 1

3.1. Rationale for Each Field

yangModules: Lists YANG modules the server can serve. Clients use this to decide whether to invoke network.yang.* tools or fall back to CLI.

cliDialect: Identifies CLI syntax (cisco-iosxr, huawei-vrp, etc.). Controllers can adjust prompt regex and command sequences accordingly.

configDatastore: Bit-mask hint— running = editable live config; candidate = two-phase commit; operational = read-only state DB. Avoids unnecessary NETCONF hello round-trips.

notificationStream: Tells the client which async event streams the server can translate into MCP notifications. Client can subscribe only to available types.

maxBulkEdit: Device-level limit to avoid oversized edit-config requests. Controllers can chunk large changes.

supportsRollback / rollbackTimeout: Boolean plus numeric seconds. Lets client know a confirmed-commit can be rolled back automatically if not confirmed within the window.

4. Tools

Seven new tool names are defined. All reuse the standard MCP tools/call request and MUST be listed by tools/list.

Name	Description
network.cli.exec	Execute operational CLI show commands
network.cli.configure	Enter config mode and send commands
network.yang.get	Retrieve YANG data node
network.yang.edit	Edit candidate datastore
network.commit	Commit candidate to running
network.rollback	Rollback to previous commit
network.file.pull	Backup config file
network.file.push	Restore config file

Table 1

Arguments are described by JSON Schema inside tool metadata, so SDKs can auto-generate bindings. If a device only supports CLI, it advertises network.yang.* tools with available=false; controllers automatically downgrade.

5. Resources

Network state is exposed as read-only resources with URI scheme network:///. Examples:

network:///interface/TenGigE0/0/0/0
network:///routing/ipv4/route-table
network:///file/startup-config

MIME type MUST match the encoding (application/yang-data+json, text/plain, etc.). Large data can be paginated using the native MCP resource/subscribe cursor mechanism.

6. Prompts

Interactive assistants MAY expose prompt templates such as

network.troubleshoot.ping-fail, network.config.add-vlan, network.security.audit.

Arguments are implementation specific but SHOULD use the same JSON Schema style as tools for consistency.

7. Error Codes

New JSON-RPC error codes (registered in IANA section):

Code	Message
-32081	Network.Timeout
-32082	Network.Unreachable
-32083	Network.AccessDenied
-32084	Network.ConfigIncompatible
-32085	Network.RollbackFailed
-32086	Network.ConfirmedCommitTimeout

Table 2

These codes allow controllers to distinguish transport failures from authorization or device-level errors without parsing free-text messages.

8. Security Considerations

All operations run with the privileges of the authenticated MCP session. Servers MUST enforce role-based access control for configuration commands. Commit confirmed SHOULD be used for potentially disruptive changes. Transport security is provided by the underlying MCP transport (TLS for HTTP, SSH port-forward for stdio). Sensitive data (passwords, SNMP communities) MUST be redacted in logs and MCP traces.

9. IANA Considerations

9.1. MCP Capability Tokens Registry

IANA is requested to register the following value:

* Token: network

* Description: Network equipment extensions for MCP

* Reference: this document

9.2. JSON-RPC Error Codes Registry

IANA is requested to register the error codes -32081 to -32086 in the "JSON-RPC Application-Specific Error Codes" registry, all pointing to this document.

10. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, 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, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

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

[RFC8259] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

11. Informative References

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

Appendix A. Appendix A. JSON Schema Examples

Example tool metadata snippet (pretty printed):

```
{
  "name": "network.yang.get",
  "description": "Retrieve a YANG data node",
  "inputSchema": {
    "type": "object",
    "properties": {
      "path": { "type": "string" },
      "datastore": { "enum": ["running", "operational"] }
    },
    "required": ["path"]
  }
}
```

Author’s Address

Zeng Guanming
Huawei Technology
Email: zengguanming@huawei.com