Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 23 April 2026

G. Zeng
Q. Wu
Huawei

20 October 2025

Model Context Protocol (MCP) Extensions for Network Equipment Management
draft-zw-nmrg-mcp-network-mgmt-00

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 23 April 2026.

Copyright Notice

Table of Contents

1. Introduction 3
2. Terminology 3
3. Capability Advertisement 3
3.1. Rationale for Each Field 4
4. Tools 4
5. Resources 5
5.1. Standard URI Templates and MIME Types 5
5.2. Subscription and Pagination 7
5.3. Read-Write Resources (Optional) 7
5.4. Per-Resource Metadata 8
6. Prompts 8
6.1. Standard Prompt Templates 8
6.2. Example: Ping-Failure Diagnosis Flow 9
6.3. Prompt Metadata and Completion 9
6.4. Streaming and Human-in-the-Loop 10
7. Error Codes 10
7.1. Error Code Table 11
7.2. Example Error Response 12
7.3. IANA Registration 13
8. Complete Interaction Example (Controller → Router) 13
8.1. Transport and Initialize 13
8.2. Read Resource (Interface Status) 14
8.3. Prompt Diagnosis (Ping Failure) 14
8.4. YANG Edit + Confirmed Commit 15
8.5. Event Stream (Syslog) 16
8.6. Graceful Shutdown 16
9. Security Considerations 16
10. IANA Considerations 16
10.1. MCP Capability Tokens Registry 17
10.2. JSON-RPC Error Codes Registry 17
11. Normative References 17
12. Informative References 17
Appendix A. JSON Schema Examples 17
Authors' Addresses 18

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": "huawei-vrp",
  "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 equipment exposes configuration, operational and file data as MCP resources under the URI scheme network:///. All resources described below are read-only unless explicitly marked "read-write".

Servers MUST support resources/read and SHOULD support resources/subscribe for streaming resources. Large responses MAY be paginated using the standard MCP nextCursor mechanism.

5.1. Standard URI Templates and MIME Types

URI Template	MIME Type	Access	Description
network:///interface/{name}	application/yang-data+json	read-only	Single interface YANG node (OpenConfig or IETF)

network:///interfaces	application/yang-data+json	read-only	Full interface list; supports cursor pagination
network:///routing/ipv4/route-table	application/yang-data+json	read-only	IPv4 RIB (operational datastore)
network:///routing/ipv6/route-table	application/yang-data+json	read-only	IPv6 RIB (operational datastore)
network:///system/cpu-utilization	application/json	read-only	CPU percentage; updated every 30 s
network:///system/memory-summary	application/json	read-only	Memory usage summary
network:///log/syslog/last{count}	text/plain; charset=utf-8	read-only	Most recent syslog lines (max 10000)
network:///file/running-config	text/plain	read-only	Running configuration text
network:///file/startup-config	text/plain	read-only	Startup configuration text
network:///file/{slot}/crashinfo	application/json	read-only	Crash information file (JSON array)

Table 2

5.2. Subscription and Pagination

Resources that change over time (CPU, memory, syslog, interface counters) SHOULD implement resources/subscribe. The server sends unsolicited resources/updated notifications when the underlying data changes.

For large lists (e.g., full IPv4 RIB) the server MAY insert nextCursor in the response:

```
{
  "jsonrpc": "2.0",
  "id": 4,
  "result": {
    "contents": [
      {
        "uri": "network:///routing/ipv4/route-table",
        "mimeType": "application/yang-data+json",
        "data": { "ietf-routing:routes": [ … 1000 items … ] }
      }
    ],
    "nextCursor": "eyJzb3J0LWtleSI6IjEwLjAuMC4xLzI0In0="
  }
}
```

The client continues with:

```
{
  "jsonrpc": "2.0",
  "id": 5,
  "method": "resources/read",
  "params": {
    "uri": "network:///routing/ipv4/route-table",
    "cursor": "eyJzb3J0LWtleSI6IjEwLjAuMC4xLzI0In0="
  }
}
```

until nextCursor is absent.

5.3. Read-Write Resources (Optional)

If the device supports whole-config replace, the following URIs MAY be advertised with readOnly: false:

* network:///file/candidate-config (text/plain)

* network:///file/startup-config (text/plain)

Writing is done via the standard MCP resources/write method; the server MUST validate syntax before committing to candidate/running.

5.4. Per-Resource Metadata

Each resource descriptor MAY include a _meta object (MCP reserved) to carry unit, precision, or hardware-specific attributes, e.g.:

```
"_meta": {
  "unit": "percent",
  "precision": 0.1,
  "hardware-slot": "1/0"
}
```

6. Prompts

Interactive assistants (e.g., CLI copilot, chat-bot) MAY expose prompt templates whose names start with network.. Templates are listed by prompts/list and executed via prompts/get; both reuse the standard MCP prompt meta-data and JSON Schema for arguments.

Each entry below includes:

* template name (prompts/list)
* one-line description
* input JSON Schema
* expected tool calls / steps
* return MIME type (text/markdown by default)

6.1. Standard Prompt Templates

Template Name	Description	Input Schema
network.troubleshoot.ping-fail	Step-by-step ping failure diagnosis	{src: string, dst: string, vrf?: string}
network.config.add-vlan	Interactive VLAN creation wizard	{vlan_id: uint16, name: string, ports?: string[]}
network.security.audit	Generate and run security	{profile: "basic"|"detailed"}

```
|                                |compliance   |                   |
|                                |checks       |                   |
+--------------------------------+-------------+-------------------+
```

Table 3

6.2. Example: Ping-Failure Diagnosis Flow

The client calls prompts/get:

```
{
  "jsonrpc": "2.0",
  "id": 10,
  "method": "prompts/get",
  "params": {
    "name": "network.troubleshoot.ping-fail",
    "arguments": {
      "src": "192.168.1.1",
      "dst": "10.0.0.5",
      "vrf": "mgmt"
    }
  }
}
```

Server returns a markdown prompt that embeds tool calls:

```
## Ping failure diagnosis: 192.168.1.1 → 10.0.0.5 (VRF mgmt)

1. Check local ARP entry:
   ‘network.cli.exec {"cmd": "show ip arp vrf mgmt 10.0.0.5"}‘

2. Verify outbound interface status:
   ‘network.yang.get {"path": "/openconfig-interfaces:interfaces/interface[name=’Vlan1
00’]", "datastore": "operational"}‘

3. Run extended ping:
   ‘network.cli.exec {"cmd": "ping vrf mgmt 10.0.0.5 source 192.168.1.1 repeat 5"}‘

... (interactive steps continue) ...
```

The client MAY stream the markdown to the user and **inline-execute** each tool call, then feed results back into the same prompt context (follow-up prompts/get with previousContextId) until the template outputs "Diagnosis complete".

6.3. Prompt Metadata and Completion

Prompt descriptors returned by prompts/list MAY include:

```
{
  "name": "network.config.add-vlan",
  "description": "Interactive VLAN creation wizard",
  "arguments": [
    {
      "name": "vlan_id",
      "description": "IEEE 802.1Q VLAN ID",
      "type": "integer",
      "minimum": 1,
      "maximum": 4094
    },
    {
      "name": "name",
      "description": "VLAN name (no spaces)",
      "type": "string",
      "pattern": "^[A-Za-z0-9-_]+$"
    },
    {
      "name": "ports",
      "description": "List of interface names to add to VLAN",
      "type": "array",
      "items": { "type": "string" }
    }
  ],
  "required": ["vlan_id", "name"]
}
```

Controllers can therefore auto-generate UI forms or CLI wizards without hard-coding any vendor logic.

6.4. Streaming and Human-in-the-Loop

Templates MAY set "stream": true in their meta-data. In this mode the server returns a prompts/stream handle and pushes incremental markdown + tool calls, allowing step-by-step confirmation by the user.

When the template issues a network.commit step it MUST include a confirmation prompt; the client SHALL wait for explicit user approval before invoking the commit tool.

7. Error Codes

This document registers six new JSON-RPC application-specific error codes in the range -32081 to -32086. Servers that implement optional features MAY return additional codes -32087 to -32090 defined below. All codes point to this document as reference.

7.1. Error Code Table

Code	Message	Typical Scenario	Recommended Recovery
-32081	Network.Timeout	CLI or NETCONF RPC exceeded device timeout	Retry with shorter command or increase timeout
-32082	Network.Unreachable	Transport session (SSH/TLS) down	Check IP reachability, credentials, routing
-32083	Network.AccessDenied	User role lacks permission for command/ path	Escalate role or use lower-privilege account
-32084	Network.ConfigIncompatible	Command/ YANG node not supported on this platform	Fall back to alternate model or CLI dialect
-32085	Network.RollbackFailed	Confirmed-commit rollback could not be applied	Manual intervention required; check logs
-32086	Network.ConfirmedCommitTimeout	Confirmed commit window expired; config auto-rolled back	Re-apply change with longer timer or fix root cause
-32087	Network.RollbackNotSupported	Device	Skip rollback

		does not support rollback operation	step or use file-based restore	
-32088	Network.YangSyntaxError	YANG payload failed server-side validation	Fix offending node and re-submit	
-32089	Network.HardwareFailure	Hardware resource exhausted (TCAM, memory)	Reduce scale or upgrade hardware	
-32090	Network.SnmpFailure	SNMP SET returned error-status ≠ noError	Map error-status to text and retry with correct value	

Table 4

7.2. Example Error Response

A device that rejects a VLAN because the ID is out of range would return:

```
{
  "jsonrpc": "2.0",
  "id": 12,
  "error": {
    "code": -32084,
    "message": "Network.ConfigIncompatible",
    "data": {
      "detail": "VLAN 4095 > maximum 4094",
      "path": "/openconfig-vlan:vlans/vlan/config/vlan-id",
      "retryPossible": false
    }
  }
}
```

The data object is optional but SHOULD contain detail (human-readable), path (affected YANG path), and retryPossible boolean to help controllers decide whether to retry automatically.

7.3. IANA Registration

This document requests IANA to register the following error codes in the "JSON-RPC Application-Specific Error Codes" registry:

* -32081 to -32086 (mandatory set)

* -32087 to -32090 (optional set)

All codes reference this document and are reserved for network equipment implementing MCP.

8. Complete Interaction Example (Controller → Router)

This example walks through a full MCP session between a controller (client) and a router (server). Transport is TLS-over-TCP; frames are sent as length-prefixed JSON-RPC 2.0 objects per MCP spec.

8.1. Transport and Initialize

Controller opens TCP 443 and performs TLS handshake with ALPN protocol mcp. Then sends:

```
-> {"jsonrpc":"2.0","id":1,"method":"initialize","params":{
     "protocolVersion":"2025-06-18",
     "capabilities":{
       "prompts":{"listChanged":true},
       "resources":{"subscribe":true},
       "tools":{"listChanged":true}
     },
     "clientInfo":{"name":"ACME-NetCtrl","version":"4.7.0"}
   }}
```

Router replies:

```
<- {"jsonrpc":"2.0","id":1,"result":{
     "protocolVersion":"2025-06-18",
     "capabilities":{
       "prompts":{"listChanged":false},
       "resources":{"subscribe":true},
       "tools":{"listChanged":false},
       "network":{
         "yangModules":["ietf-interfaces","openconfig-interfaces"],
         "cliDialect":"huawei-vrp",
         "configDatastore":["running","candidate","operational"],
         "notificationStream":["syslog","netconf-stream"],
         "maxBulkEdit":1000,
         "supportsRollback":true,
         "rollbackTimeout":300
       }
     },
     "serverInfo":{"name":"IOS-XR-MCP","version":"7.5.3"}
   }}
```

8.2. Read Resource (Interface Status)

Controller fetches operational data:

```
-> {"jsonrpc":"2.0","id":2,"method":"resources/read","params":{
     "uri":"network:///interface/TenGigE0/0/0/0"}}
<- {"jsonrpc":"2.0","id":2,"result":{
     "contents":[{
       "uri":"network:///interface/TenGigE0/0/0/0",
       "mimeType":"application/yang-data+json",
       "data":{
         "openconfig-interfaces:interface":{
           "name":"TenGigE0/0/0/0",
           "state":{"admin-status":"UP","oper-status":"DOWN"}
         }
       }
     }]
   }}
```

8.3. Prompt Diagnosis (Ping Failure)

Controller invokes the standardized prompt:

```
   -> {"jsonrpc":"2.0","id":3,"method":"prompts/get","params":{
        "name":"network.troubleshoot.ping-fail",
        "arguments":{"src":"192.168.1.1","dst":"10.0.0.5","vrf":"mgmt"}}}
   <- {"jsonrpc":"2.0","id":3,"result":{
        "description":"Step-by-step ping failure diagnosis",
        "messages":[{
          "role":"assistant",
          "content":"## Ping failure: 192.168.1.1 → 10.0.0.5 (VRF mgmt)\n\n1. Check loca
l ARP:\n   'network.cli.exec {\"cmd\":\"show ip arp vrf mgmt 10.0.0.5\"}'"
        }],
        "_meta":{"toolCalls":[{"tool":"network.cli.exec","arguments":{"cmd":"show ip arp
vrf mgmt 10.0.0.5"}}]}
      }}
```

Client executes the embedded tool and sends follow-up:

```
   -> {"jsonrpc":"2.0","id":4,"method":"tools/call","params":{
        "name":"network.cli.exec","arguments":{"cmd":"show ip arp vrf mgmt 10.0.0.5"}}}
   <- {"jsonrpc":"2.0","id":4,"result":{"stdout":"10.0.0.5  00:50:56:ab:cd:ef  Vlan100"}}
```

Prompt continues until root cause is found and printed to user.

8.4. YANG Edit + Confirmed Commit

Controller adds a loopback interface:

```
   -> {"jsonrpc":"2.0","id":5,"method":"tools/call","params":{
        "name":"network.yang.edit","arguments":{
          "target":"candidate",
          "edit":[{
            "path":"/openconfig-interfaces:interfaces/interface[name='Loopback100']",
            "value":{
              "config":{"name":"Loopback100","type":"iana-if-type:softwareLoopback"},
              "subinterfaces":{"subinterface":[{"index":0,"config":{"index":0,"ipv4":{"ad
dresses":{"address":[{"ip":"100.100.100.100","config":{"ip":"100.100.100.100","prefix-len
gth":32}}]}}}}]}
            }
          }]
        }
      }}
   <- {"jsonrpc":"2.0","id":5,"result":{"commit-id":"2025-10-19-03-27-00"}}
```

Confirmed commit with 120 s rollback window:

```
   -> {"jsonrpc":"2.0","id":6,"method":"tools/call","params":{
        "name":"network.commit","arguments":{"confirmed":120}}}
   <- {"jsonrpc":"2.0","id":6,"result":{"status":"committed","rollbackTimeout":120}}
```

Controller verifies connectivity; if OK it sends final confirm:

```
   -> {"jsonrpc":"2.0","id":7,"method":"tools/call","params":{
        "name":"network.commit","arguments":{"confirm":true}}}
   <- {"jsonrpc":"2.0","id":7,"result":{"status":"confirmed"}}
```

8.5. Event Stream (Syslog)

Controller subscribes to syslog stream:

```
   -> {"jsonrpc":"2.0","id":8,"method":"resources/subscribe","params":{
        "uri":"network:///log/syslog/last200"}}
   <- {"jsonrpc":"2.0","id":8,"result":{"subscriptionId":"syslog-42"}}
```

Server later pushes:

```
   <- {"jsonrpc":"2.0","method":"resources/updated","params":{
        "subscriptionId":"syslog-42",
        "uri":"network:///log/syslog/last200",
        "contents":[{
          "mimeType":"text/plain",
          "data":"Oct 19 03:29:01.123: %LINEPROTO-5-UPDOWN: Line protocol on Interface Lo
opback100, changed state to up\n"
        }]
      }}
```

8.6. Graceful Shutdown

Controller unsubscribes and closes the session:

```
   -> {"jsonrpc":"2.0","id":9,"method":"resources/unsubscribe","params":{
        "subscriptionId":"syslog-42"}}
   <- {"jsonrpc":"2.0","id":9,"result":{}}

   -> {"jsonrpc":"2.0","id":10,"method":"close"}
   <- {"jsonrpc":"2.0","id":10,"result":{}}
```

TCP connection is closed by controller.

9. 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.

10. IANA Considerations

10.1. MCP Capability Tokens Registry

IANA is requested to register the following value:

* Token: network
* Description: Network equipment extensions for MCP
* Reference: this document

10.2. 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.

11. 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>.

12. Informative References

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

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"]
  }
}
```

Authors' Addresses

Zeng Guanming
Huawei
Email: zengguanming@huawei.com

Wu Qin
Huawei
Email: bill.wu@huawei.com