Network Working Group
Internet-Draft
Intended status: Informational
Expires: 8 November 2026

S. Prabhu
Nokia

7 May 2026

Framework for Normalizing Multi-Vendor Network Inputs for LLM-Assisted Network Management
draft-prabhu-nmrg-prompt-schema-llm-00

Abstract

Large Language Models (LLMs) are increasingly used to assist network management tasks such as troubleshooting, intent translation, and automation. Network operations, however, rely on data from many vendors and sources: CLI output, configuration snippets, telemetry, alarms, and vendor-specific APIs. These inputs differ in format, structure, and semantics, which makes it difficult to present a consistent interface to an LLM. This document describes a framework for standardizing such multi-vendor inputs for LLM-assisted network management. Incoming messages from multi-vendor network elements are first handled by an Input Classifier, which determines the nature of each input and assigns it to one of three categories: performance, configuration, or response, using a hybrid approach (rule-based classification first, with escalation to a Small Language Model (SLM) when rules are insufficient). The classified input is then passed to the corresponding Structurer among the Performance Structurer, Configuration Structurer, and Response Structurer, each of which produces a normalized, structured representation (typically with SLM assistance and optional confidence scoring). That structured output is fed to a Prompt Schema Generator, which creates a structured, vendor-agnostic schema and supplies schema-aligned prompts to the central LLM. This document specifies the architecture and component roles for use in design and implementation. It does not define a wire protocol; it is published for informational purposes.

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/.

Copyright Notice

Table of Contents

1. Introduction

LLM-assisted network management uses Large Language Models to interpret network data, suggest actions, translate operator intent into device commands, and assist with root cause analysis and reporting. The effectiveness of such systems depends on the quality and consistency of the inputs presented to the LLM. In practice, those inputs are highly heterogeneous: show-command output and configuration dumps vary by vendor and device type; telemetry and alarms use different formats and naming; and management APIs differ across vendors. Without normalization, the same logical concept (e.g., "interface status" or "BGP neighbor state") appears in many forms, which degrades LLM performance and complicates prompt design and maintenance.

This document describes an architectural framework for standardizing multi-vendor inputs before they are used as prompts (or prompt context) for a central LLM in network management. The flow is as follows. (1) Multi-vendor network data is received by the Input Classifier, which classifies each message as performance-related, configuration-related, or response-related so that the appropriate Structurer receives the right type of input. The Input Classifier uses a hybrid approach: a fast, deterministic rule-based stage first; when pattern matching fails, multiple category conflicts arise, or structural certainty is insufficient, an SLM-based classification stage within the Input Classifier is invoked. (2) Based on that classification, the message is passed to exactly one of the Performance Structurer, Configuration Structurer, or Response Structurer, each of which interprets vendor-variable text and emits a structured record. (3) The structured output is fed to the Prompt Schema Generator, which creates or selects a vendor-agnostic schema and produces schema-aligned prompts. (4) Those prompts are supplied to the central LLM that performs network management (e.g., troubleshooting, intent translation, or automation).

The key components are the Input Classifier, the three Structurers, the Prompt Schema Generator, and the central LLM. Together, they enable multi-vendor inputs to be categorized, structured, normalized via a schema, and presented to the central LLM in a consistent form.

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.

* LLM (Large Language Model): A machine learning model used for natural language understanding and generation. In this document, the LLM is the consumer of standardized network management inputs for tasks such as troubleshooting, intent translation, and automation.

* Prompt: Input (or input context) supplied to an LLM for processing. In LLM-assisted network management, a prompt typically includes network-derived data such as CLI output, configuration snippets, telemetry, or alarms, often after normalization via a schema.

* Schema: A formal, vendor-agnostic description of the structure, types, and semantics of a class of network management inputs. A schema allows multi-vendor data to be normalized to a common shape before being presented to the LLM.

* Input Classifier: A component that determines the nature of incoming messages from multi-vendor network elements and assigns each input to one of three categories: performance, configuration, or response, so that the corresponding Structurer can process it. It uses a hybrid rule-based stage followed by SLM-based classification when the rule-based stage is inconclusive.

* Structurer: A component that receives input already labeled by the Input Classifier with one category and produces a structured, normalized representation for the Prompt Schema Generator. This document defines three structurers: Performance Structurer, Configuration Structurer, and Response Structurer.

* Small Language Model (SLM): A compact language model used inside the Input Classifier (for semantic classification when rules fail) and inside each Structurer (for extraction, normalization, and confidence). SLMs MAY be fine-tuned on curated multi-vendor network messages, telemetry, configuration text, and operational responses.

* Confidence level: An optional indicator (e.g., high, medium, low) associated with structuring or classification. Low-confidence cases MAY attach raw input for human-in-the-loop review and later feedback into training of the Input Classifier and Structurers.

3. Architecture Overview

The framework is a logical system that interfaces with the central (existing) LLM that manages the network; it sits between multi-vendor network data sources and that central LLM. Raw inputs (e.g., CLI output, configuration, telemetry, alarms, or API responses) enter the Input Classifier. The classifier routes each message to the Performance Structurer, Configuration Structurer, or Response Structurer. The active Structurer’s output is fed to the Prompt Schema Generator, which produces schema-aligned prompts for the central LLM.

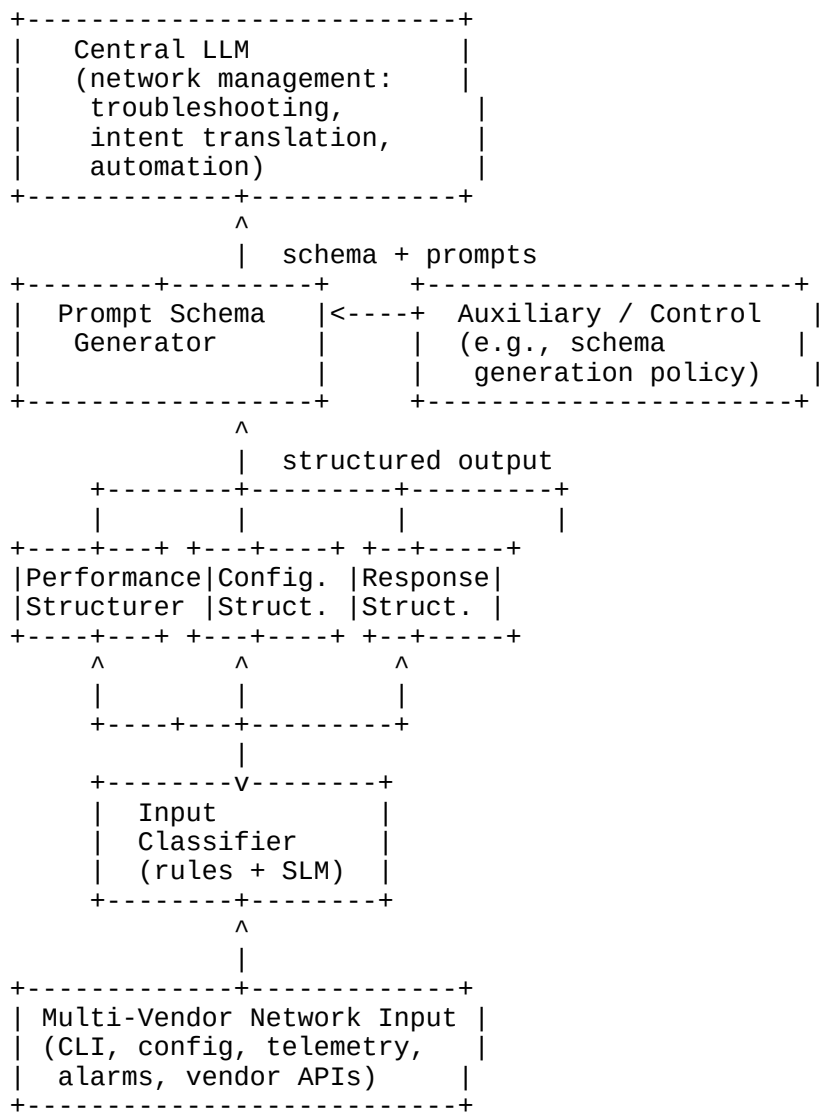

Figure 1: Reference architecture

4. Input Classifier

The Input Classifier is responsible for determining the nature of incoming messages from multi-vendor network elements. It assesses each input and classifies it into exactly one of three categories: performance, configuration, or response, so that the corresponding Structurer receives the appropriate type of input for further processing.

The Input Classifier uses a hybrid approach: rule-based classification is attempted first, followed by SLM-based classification when the rule-based stage cannot produce a reliable single category. The hybrid design balances speed, efficiency, and accuracy: rule-based logic is fast and lightweight for common patterns; when inputs are complex, ambiguous, or unmatched by rules, an SLM performs deeper contextual analysis.

4.1. Rule-based classification

In the first stage, the Input Classifier applies a deterministic, rule-based mechanism to categorize incoming network element messages. This layer is intended for high-frequency, well-structured, and common message formats without invoking computationally intensive semantic analysis.

The rule-based engine MAY use a combination of:

* Keyword dictionaries.

* Structured field detection.

* Regular expression matching.

* Protocol-aware parsing.

* Message header inspection.

Each message is evaluated against category-specific rule sets for performance, configuration, and response classification.

4.1.1. Performance message detection

Performance-related messages typically describe operational metrics, telemetry readings, utilization values, or statistical measurements. Example keyword families include: utilization, throughput, latency, jitter, packet loss, bandwidth, CPU usage, memory usage, temperature, receive/transmit rates, error rate, and queue depth (exact vocabularies are implementation defined).

Pattern-based heuristics MAY include: numeric values with percentage signs; metric names adjacent to numeric measurements; telemetry-style field-value formats. As an illustrative example, a regular expression such as:

```
(utilization|load|usage)\s*[:=]?\s*\d+(\.\d+)?\s*%
```

can match lines such as "CPU utilization: 78%", "Traffic load = 45%", and "Network usage 62%". Similar patterns MAY be defined for other performance metrics (e.g., latency, throughput).

4.1.2. Configuration message detection

Configuration-related messages indicate intent to modify, retrieve, delete, or update network state. Example keywords include: set, configure, apply, update, modify, delete, add, remove, commit, rollback, enable, and disable.

As an illustrative example, a regular expression such as:

```
(set|update|configure|modify)\s+(interface|vlan|route|policy|qos|ip)
```

can match phrases such as "Set interface Gi0/1 bandwidth 100Mbps", "Update VLAN 10 configuration", and "Modify route table entry".

4.1.3. Response message detection

Response messages typically represent outcomes of prior operations. Example keywords include: success, completed, applied, acknowledged, error, failed, failure, invalid, timeout, rejected, and warning.

As an illustrative example, a regular expression such as:

```
(success|completed|applied|acknowledged)\b
```

can match phrases such as "Configuration applied successfully", "Operation completed", and "Request acknowledged".

When pattern matching fails, multiple categories conflict, or structural certainty is insufficient, implementations SHOULD escalate to SLM-based classification within the Input Classifier.

4.2. SLM-based classification

When the rule-based stage cannot categorize a message (for example, due to absence of rule matches, conflicting rule matches, or ambiguous structure), the message is processed by an SLM-based semantic classification stage inside the Input Classifier.

The SLM interprets context, semantics, and structure to assign the message to performance, configuration, or response. The model MAY be trained or fine-tuned on a curated corpus that includes historical multi-vendor network messages, telemetry logs, configuration commands, and operational responses, so that nuanced and vendor-specific formats can be handled using contextual cues (e.g., attention over token relationships and embeddings).

5. Structurers

After classification, the message is delivered to exactly one of three Structurers. Each Structurer interprets inputs that the Input Classifier has already labeled with the matching category. Because vendors represent performance, configuration, and response text in widely varying forms, each Structurer typically employs an SLM for adaptability, extraction, normalization, and confidence assignment. Structured records are then passed to the Prompt Schema Generator.

5.1. Performance Structurer

The Performance Structurer processes inputs categorized as performance by the Input Classifier.

Typical processing steps include:

1. Input reception: The structurer receives performance-labeled input (e.g., a line such as "Traffic load = 75%").

2. Value extraction: The SLM extracts numeric and unit information (e.g., isolating "75" and "%"). Heterogeneous forms (e.g., ratios such as "8/10") MAY be converted to a comparable representation (e.g., 80%).

3. Normalization: Values are normalized to a standard scale for downstream use (commonly a percentage), so that fractions, raw throughput figures, and percentages align to a uniform representation.

4. Confidence scoring: The SLM MAY assign high (well-recognized pattern), medium (ambiguous; flag for review), or low (unrecognized pattern). For low confidence, implementations MAY attach the raw network element input when sending output to the Prompt Schema Generator for human-in-the-loop review; feedback from that review MAY later be used to retrain or refine the Input Classifier and Structurers.

5. Performance structuring: The structurer emits a structured representation to the Prompt Schema Generator in an implementation-defined format consistent with the chosen schema.

5.2. Configuration Structurer

The Configuration Structurer processes inputs categorized as configuration by the Input Classifier.

Typical processing steps include:

1. Input reception: Configuration-labeled input is received (e.g., "Set interface bandwidth to 100 Mbps").

2. Operation identification: The SLM detects the configuration operation (e.g., set, update, delete, enable, disable).

3. Target identification: The structurer identifies the configuration target (e.g., interface bandwidth, firewall rule, routing policy).

4. Value extraction: When present, parameter values are extracted (e.g., "100 Mbps"); some operations (e.g., "enable logging") MAY have no separate value beyond the operation and target.

5. Normalization of targets: Vendor-specific phrases for the same concept (e.g., "interface bandwidth" vs. "link speed") are mapped to a standard target representation.

6. Configuration structuring: A structured record is emitted to the Prompt Schema Generator in an implementation-defined format consistent with the chosen schema.

5.3. Response Structurer

The Response Structurer processes inputs categorized as response by the Input Classifier, using an SLM to cope with variability in how vendors phrase operational outcomes.

Typical processing steps include:

1. Input reception: Response-labeled input is received (e.g., "Configuration applied successfully on interface Gi0/1").

2. Response intent interpretation: The SLM determines the outcome category, such as success, failure, partial success, warning, timeout, or rejection; whether the message relates to a prior configuration action; and whether an explicit error condition is present.

3. Error and context extraction: For failure or warning outcomes, the SLM MAY extract error reason, error code (if present), parameters involved, and affected object (e.g., interface, route, policy).

4. Confidence assignment: High confidence MAY correspond to clear success/failure wording; medium to ambiguity; low to non-standard or unclear outcomes; raw input MAY be attached for the Prompt Schema Generator when confidence is low.

5. Response structuring: A standardized structured representation is produced for the Prompt Schema Generator in an implementation-defined format consistent with the chosen schema.

The three structurers are distinct logical components; for a given message, only the structurer matching the Input Classifier’s category is used. Implementations MAY pipeline batches such that multiple structured records are produced over time and assembled by the Prompt Schema Generator according to policy.

6. Prompt Schema Generator

The Prompt Schema Generator receives structured output from the active Structurer (performance, configuration, or response), optionally including raw input attachments and confidence metadata from low-confidence paths. It creates or selects a structured, vendor-agnostic schema appropriate to that category and message shape, and produces schema-aligned prompts for the central LLM.

Functions of the Prompt Schema Generator in this context include:

* Creating or selecting schemas that define structure, fields, types, and semantics of network management inputs in a form suitable for the central LLM, with schema choice driven by the classification and structurer output.

* Enabling mapping from vendor-specific representations to a single schema so the central LLM receives consistent input regardless of source.

* Feeding the schema and resulting prompts (or prompt context) to the central LLM for tasks such as troubleshooting, intent translation, and automation.

Implementations may use static schema definitions (e.g., based on common network management information models), vendor-specific mapping tables, or dynamically updated schemas. The output of the Prompt Schema Generator is consumed by the central LLM; implementations may also use it for normalization logic that maps structured records into schema-conformant prompts before LLM invocation.

7. Output and Downstream Use

The output of the framework is schema-aligned prompts (or prompt context) from the Prompt Schema Generator, derived from classified and structured multi-vendor input. Each prompt is associated with the performance, configuration, or response category assigned by the Input Classifier, with the schema used to create it. These prompts are supplied to the central LLM that performs network management (e.g., troubleshooting, intent translation, compliance analysis, or automation). Downstream components (e.g., prompt assembly, LLM invocation, or result parsing) use the category, structurer output, and schema alignment to choose models, templates, and response handling. The exact format and semantics of data passed from the Prompt Schema Generator to the central LLM are implementation defined.

8. Security Considerations

The structure of this section is informed by the guidelines in [RFC3552]. Implementations of this framework in LLM-assisted network management should consider the following:

* Network management inputs often contain sensitive data: credentials, topology, IP addressing, and configuration details. Data flows through the Input Classifier, the active Structurer, the Prompt Schema Generator, and the central LLM; each stage may expose this information. Implementations SHOULD apply data minimization (e.g., redacting or tokenizing secrets during classification, structuring, and schema generation), secure transport and storage for intermediate data, and policies that limit what is sent to external or cloud-hosted LLMs. Compliance with organizational and regulatory requirements for network and personal data SHOULD be ensured.

* Schema definitions and mapping rules determine what structure and fields are sent to the LLM. Schema generation or selection that relies on untrusted input can introduce injection or policy bypass risks. Implementations SHOULD validate and constrain schema content and SHOULD restrict who can create or modify schema definitions and vendor mappings.

* Incorrect classification or structuring can change which prompts and schemas are used and can affect prioritization and access control. Crafted inputs that are misclassified or misstructured could lead to incorrect LLM behavior, privilege escalation, or denial of service. Implementations SHOULD treat classifier and structurer output as advisory where security is critical and SHOULD apply additional checks before acting on LLM outputs that affect network configuration or access.

* Consolidating multi-vendor inputs increases attack surface: parsers, regex engines, SLM inference, schema mappings, and paths to the central LLM. Implementations SHOULD harden the Input Classifier, each Structurer, the Prompt Schema Generator, and any normalizers against malformed or malicious input and SHOULD monitor for abuse or unexpected classification patterns.

9. IANA Considerations

This document has no IANA actions.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

10.2. Informative References

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <https://www.rfc-editor.org/info/rfc3552>.

Author’s Address

Shailesh Prabhu
Nokia
India
Email: shailesh.prabhu@nokia.com