Model-driven programmability with the Nokia SR OS

Simplify and automate IP network operations

Application note
Abstract

Evolution toward open software systems is driving the need for model-based programmability and automation. The Nokia Service Router Operating System (SR OS) provides the foundation for our comprehensive portfolio of IP routers. This application note explains how model-driven programmability with the Nokia SR OS can help network operators with their IP network evolution initiatives by simplifying and automating IP network operations.
Contents

Abstract 2
Network automation challenges 4
Nokia SR OS: Model-driven programmability 4
  Modeling 5
  Transport and encoding 5
  Interfaces 5
  APIs 6
YANG 6
NETCONF 7
gRPC 8
  Model-driven telemetry with gRPC 9
MD-CLI 10
  Smooth migration to MD-CLI 10
  MD-CLI based on YANG models 10
  Support for transactional configuration as a default 10
  Configuration access control and safety 10
  Templating with configuration groups 11
Getting started with SR OS programmability 11
  Using SR OS model-driven applications 11
  Learning about SR OS network programmability 11
Summary 11
Learn more 12
Abbreviations 12
Network automation challenges

For many years network operators have been looking for ways to simplify repetitive tasks in multivendor networks. The massive growth of the cloud, 5G and IoT era is pushing the limits of network scaling. Network operators need to manage tens of thousands of network devices while ensuring they deliver innovative services in a timely manner and with excellent quality of experience for customers.

Network automation and the move to a DevOps-centric approach are critical elements to help deliver operational efficiency and cost savings. A key part of this paradigm shift includes the adoption of model-driven approaches, which are fundamentally different from the present mode of operation and deliver a dramatic improvement over the current process.

In the past, operators needed to create proprietary adapters that translate between their management systems and vendor-specific command line interfaces (CLIs). This approach is cumbersome and time consuming, and operators had to wait months for new equipment releases to be supported because equipment feature support and the necessary device and service object models needed to be changed in the management system code base (by vendor software designers in the case of vendor systems).

By contrast, model-driven approaches abstract service-specific configurations from their vendor-specific implementation. Translating between service-specific configurations and device configuration is best achieved by using a common modeling language. YANG (Yet Another Next Generation) based data modeling can help eliminate the need for proprietary adapters. Devices can then be configured using programmatic interfaces with protocols such as the Network Configuration protocol (NETCONF), gRPC Remote Procedure Call (gRPC) or a model-driven CLI (MD-CLI).

Nokia SR OS: Model-driven programmability

The Nokia Service Router Operating System (SR OS) is built to power the most demanding, dynamic and reliable Ethernet and IP/MPLS networks. This robust and scalable OS provides the foundation for our comprehensive portfolio of physical and virtualized routers. The SR OS has been proven in more than 1600 service provider, enterprise and webscale networks worldwide.

The primary driver for model-driven programmability is to move from manual, vendor-specific router configuration and control to centralized, automated orchestration based on industry-defined methods. This approach enables simplified, more reliable and efficient network management.

The Nokia SR OS model-driven framework focuses on three key areas: programmability, visibility and operations. Programmability gives humans and machines access to data models via programmatic interfaces. Visibility is about statistics and state, and includes the use of telemetry to send statistics to northbound management systems. Operations relates to the ability to manage network devices in automated network environments consistently using the CLI or through programmatic interfaces such as NETCONF and gRPC.
Figure 1 shows the elements and protocols that enable model-driven programmability with the Nokia SR OS.

Figure 1. Enabling model-driven programmability with the Nokia SR OS

APIs

Interfaces

<table>
<thead>
<tr>
<th>Interfaces</th>
<th>NETCONF</th>
<th>gRPC/gNMI</th>
<th>MD-CLI</th>
</tr>
</thead>
</table>

Encoding

<table>
<thead>
<tr>
<th>Encoding</th>
<th>XML</th>
<th>JSON</th>
<th>Structured plain text</th>
</tr>
</thead>
</table>

Transport

<table>
<thead>
<tr>
<th>Transport</th>
<th>SSH</th>
<th>HTTPS (TLS)</th>
<th>Telnet or SSH</th>
</tr>
</thead>
</table>

Modeling

<table>
<thead>
<tr>
<th>Modeling</th>
<th>YANG data models (vendor-neutral, vendor-specific)</th>
</tr>
</thead>
</table>

**Modeling**

YANG-based data modeling delivers the foundation for model-driven programmability within the SR OS. These data models provide a standards-based method of configuring and operating IP routers. YANG data models are human-readable, structured data models used for configuration and for providing operational state information. The SR OS supports a comprehensive set of vendor-neutral and vendor-specific data models.

**Transport and encoding**

Transport and encoding are based on protocols standardized by the industry. Telnet or Secure Shell (SSH) for the MD-CLI is used with structured plain text encoding. NETCONF runs over SSH with eXtensible Markup Language (XML) encoding. gRPC/gNMI runs over HTTPS (TLS) with JavaScript Object Notation (JSON) encoding. The broad choice of protocols and the separation of encoding from the data model provides significant flexibility for implementations.

**Interfaces**

The SR OS includes support for model-driven interfaces, including NETCONF, gRPC and a MD-CLI. These interfaces are all based on common YANG models, ensuring a common look and feel across all interfaces. As an example, if an operator sends a configuration with NETCONF, and then views this configuration information in MD-CLI, the operator sees the exact same configuration. This makes it easy to integrate with network automation systems because both machines (for automation) and humans have access to and interpret the same thing, ensuring consistency between what automation sends and what operators type.
APIs

The SR OS allows applications to manage router features using the NETCONF, gRPC and MD-CLI interfaces. The following sections discuss several key elements of the SR OS model-driven framework in more detail.

YANG

At the root of the desire to automate IP networks is a need to represent network constructs — whether elements, policies or services — in a consistent language. The representation (model) must be a language that both humans and machines can understand without significant processing overhead.

Simple Network Management Protocol (SNMP) has been the dominant protocol to manage and monitor networks for the past few decades. However, SNMP and the Structure of Management Information Version 2 (SMIv2) modeling language lack the ability to provide common human-readable and machine-readable models, which are essential for the massive-scale network automation required for networks of the future.

YANG is a data modeling language for configuration and state data and is defined in several IETF RFCs that explain how data is organized hierarchically. YANG is designed to be readable by both humans and machines and is quickly becoming the standard way to model network devices and network device information. YANG models can also communicate constraints and dependencies in addition to the types and ranges that SMIv2 provides.

YANG models are at the heart of the Nokia SR OS. The network management functions are based on YANG, with configuration and state information plus the interfaces to the software all modeled in YANG or derived from YANG models.

The SR OS provides a fundamentally different approach to managing network devices. Model-driven management means that configuration and state information is stored in a manner prescribed by the YANG models provided with the software release.

The SR OS delivers standardized communication interfaces into this modeled information, so that configuration has a common look and feel and is consistent across the MD-CLI, NETCONF and gRPC interfaces (see Figure 2). Integration with many open source tools is straightforward and does not require proprietary vendor integration.
Figure 2. YANG providing consistency across model-driven interfaces

The SR OS approach to model-driven management makes programming when interfacing with the SR OS over NETCONF or gRPC intuitive to MD-CLI operators. This added ease for programming complements the network engineering teams’ efforts developing workflows and pipelines as part of their broader model-driven management and automation initiatives.

The Nokia SR OS is a unified routing OS that is used for the entire Nokia IP routing portfolio. YANG capabilities are being implemented for all functionality supported by the SR OS; this ensures consistency and simplifies operations for IP routers that are deployed across different domains in the IP network.

For additional information on SR OS YANG models, visit the Nokia SR OS Network Developer Portal or the Nokia YANG models repository on GitHub.

**NETCONF**

Prior to NETCONF, the only way to make automated configuration changes was to use CLI scripting or SNMP. CLI scripting lacked error management and often had changing syntax and a lack of structure, making CLI scripting costly and complex to manage. SNMP has historically been used for fault and performance monitoring. While SNMP has the capability to write changes, the data is not modeled in a flexible way, and there is no correlation between SNMP objects and the CLI.

NETCONF, as defined in IETF RFC 6241, provides mechanisms to install, manipulate and delete the configuration of network devices. NETCONF uses XML data encoding for the modeled configuration data as well as the protocol messages. The NETCONF operations are performed on top of a simple RPC layer.

Figure 3 shows a NETCONF communications workflow.
**Figure 3. NETCONF communications workflow**

NETCONF accesses one or more configuration datastores. A configuration datastore is the complete set of configuration data that is required to transform a device from its initial default state into a desired operational state.

The SR OS supports a comprehensive set of NETCONF features, including:

- Extensive RPC mechanisms
- Configuration datastores:
  - To store and access startup configuration at device boot
  - Running (currently active) and candidate (working) configurations
- ietf-netconf monitoring capability, which allows the device to be queried for supported YANG models
- The ability to test and validate configurations
- Execution of operations modeled in YANG:
  - The generic md-cli-raw-command RPC that executes any operational command and returns unstructured text output
  - Individual operations with modeled input and output.

**gRPC**

gRPC is an open source, high-performance RPC framework that can run in any environment. gRPC enables client and server applications to communicate transparently and makes it easier to build connected systems.

The gNMI (gRPC Network Management Interface) protocol is a unified management service that is built on top of gRPC. It defines how configuration operational data interacts with a network element.

Like NETCONF, gRPC has a client-server architecture. While gRPC supports configuration management, its biggest use case is for monitoring through the use of streamed telemetry. Streaming telemetry does
not rely on collectors continuously pulling data from the network elements. Instead, network elements push statistics and operational state data to the subscribing collector(s) based on defined paths/frequencies or on change of values.

The Nokia SR OS gRPC implementation includes support for several use cases, including:

- Configuration management: Configuring and retrieving configuration information
- Telemetry monitoring: Pushing operational information from network elements to management systems
- RIB programming API: Allows the operator to go directly into the FIB, the RIB or the MPLS label table to program forwarding or filter entries
- Automation of operations tasks such as certificate management, software upgrades and file management.

gRPC uses protocol buffers (protobufs), which are a flexible, efficient, automated mechanism for serializing structured data. The SR OS supports multiple protobuf-defined services, including gNMI and elements of gRPC Network Operations Interface (gNOI). A protobuf-defined service is a set of predefined protobufs to perform networking operations. For example, the gNMI protobuf-defined service provides the Capabilities, Get, Set and Subscribe RPCs.

Model-driven telemetry with gRPC

Unlike legacy monitoring platforms, streaming telemetry does not rely on collectors continuously pulling data from the network elements. Instead, network elements continuously push statistics and operational data to the subscribed collector(s) based on a defined path/frequency.

The Nokia SR OS gRPC implementation supports dial-in and dial-out telemetry (see Figure 4). Model-driven telemetry uses the Subscribe RPC. The client requests the server to send its values of particular paths (e.g., statistics) within the data tree.

Figure 4. Model-driven telemetry with gRPC (dial-in connection)
The path information can be sent using various streaming modes:
- A SAMPLE stream subscription tells the server to provide updates every "n" seconds; an example is a minimum of 1 second.
- An ON_CHANGE stream subscription tells the server to provide updates only when there has been a change.
- A TARGET_DEFINED stream subscription allows the client to delegate the selection of SAMPLE or ON_CHANGE streaming to the server.
- JSON, JSON_IETF, BYTES or PROTO encoding in telemetry subscribe responses.

These subscription modes provide flexibility regarding the amount of and frequency at which statistics data can be streamed depending on the architecture implemented for telemetry.

**MD-CLI**

Nokia has evolved the SR OS to support a MD-CLI. The SR OS MD-CLI is designed to align with model-driven concepts. The SR OS MD-CLI supports several key design attributes and features that are critical for model-driven programmability.

**Smooth migration to MD-CLI**

It is imperative that the transition to the enhanced SR OS MD-CLI is a smooth one. The SR OS MD-CLI was developed with this in mind and the user interface provides a familiar experience to operators coupled with the features of a model-driven interface. The SR OS MD-CLI provides enhanced functionality and familiarity for both operators and developers. To aid the transition, the SR OS automatically migrates the router configuration when the operator decides to make the move — and not before. There is no forced migration to the MD-CLI built into the SR OS.

**MD-CLI based on YANG models**

The MD CLI is designed to provide consistency, which is provided across all model-driven interfaces based on the implementation of YANG models. This ensures that all model-driven interface configuration over NETCONF, gRPC or MD-CLI provides the same results and is very consistent, making it easy to operate, automate and troubleshoot.

**Support for transactional configuration as a default**

Usability and ease of operation were key design considerations for the SR OS MD-CLI. It is essential for the operator to be able to control when changes are applied. The MD-CLI supports transactional configuration as a standard capability in the implementation. When an operator makes changes in the MD-CLI, the changes go into a candidate configuration, and this candidate configuration is applied when the operator is ready to apply (commit) the changes. In addition, configuration changes can be entered in any order and are reordered by the SR OS.

**Configuration access control and safety**

The MD-CLI supports configuration access control and safety for operating in a multiuser environment. Operators can take exclusive access to the configuration, so no one else can make changes at the same time, providing configuration safety. Private candidates provide the operator with a private copy of the configuration that can be edited without another operator interfering with the configuration changes. Read-only candidates allow operators to monitor candidate changes made by other humans or machines while the changes are in progress.
**Templating with configuration groups**

Templating helps simplify configuration. For example, if the operator has 1,000 ports and has a setting that’s applied on all of these ports without a template, the operator would need to make that configuration change 1,000 times for each port. With templating, an operator can define a template with required commands and then apply that template to each port instead of applying each command individually. Any subsequent changes are also applied to all the ports at the same time, providing a lot of configuration simplification and flexibility.

For additional details about the Nokia SR OS MD-CLI see the Model-driven CLI (MD-CLI) configuration videos playlist.

**Getting started with SR OS programmability**

**Using SR OS model-driven applications**

A critical question that is relevant to SR OS model-driven programmability is which configuration tools and interfaces to use and for what function.

The SR OS provides a choice of interfaces for managing the router. The choice of interface depends on what the operator is trying to do and which technology or protocol best fits into the operational framework and plans.

- **NETCONF** is developed to manage configurations on network devices. It is a fully featured configuration management protocol with many additional benefits, such as locking, multiple datastore access and transactional configuration.
- **gRPC** is a newer technology than NETCONF. A key use case for gRPC is telemetry, but gRPC is also used for configuration and operational tasks.
- While machines may be more suitable for provisioning and monitoring functions, when it comes to debugging network issues, humans can do that very efficiently using the CLI.

**Learning about SR OS network programmability**

The Nokia SR OS Developer Portal is designed to help developers integrate SR OS network devices into their own infrastructure. The portal provides tips, examples and tutorials to help developers using Nokia-developed tools and third-party open source tools to integrate with the SR OS-based product families that run many of today’s global networks.

To learn more, visit the Nokia SR OS Developer Portal.

**Summary**

Evolution toward open software systems is driving the need for model-based programmability and automation. The Nokia SR OS supports a comprehensive set of features that can help network operators with their IP network evolution initiatives. The Nokia SR OS YANG models address this need with a comprehensive set of configuration and state data models designed for automation. The NETCONF gRPC and MD-CLI interfaces leverage the YANG models to provide a set of model-driven management interfaces with a common human- and machine-readable configuration syntax. The end result is simplified and automated IP network operations — which also drive cost savings.
Learn more

To learn more about how to simplify and automate IP network operations with the Nokia SR OS:

- Visit the Nokia SR OS Developer Portal
- See the Model-driven CLI (MD-CLI) configuration video playlist
- Visit the Nokia SR OS web page.

Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CLI</td>
<td>command line interface</td>
</tr>
<tr>
<td>FiB</td>
<td>forwarding information base</td>
</tr>
<tr>
<td>gNMI</td>
<td>gRPC Network Management Interface</td>
</tr>
<tr>
<td>gNOI</td>
<td>gRPC Network Operations Interface</td>
</tr>
<tr>
<td>gRPC</td>
<td>gRPC Remote Procedure Call</td>
</tr>
<tr>
<td>HTTP</td>
<td>HyperText Transfer Protocol</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>MD-CLI</td>
<td>model-driven command line interface</td>
</tr>
<tr>
<td>MPLS</td>
<td>Multiprotocol Label Switching</td>
</tr>
<tr>
<td>NETCONF</td>
<td>Network Configuration Protocol</td>
</tr>
<tr>
<td>NMS</td>
<td>network management system</td>
</tr>
<tr>
<td>OSS</td>
<td>operations support system</td>
</tr>
<tr>
<td>RFC</td>
<td>Request for Comments</td>
</tr>
<tr>
<td>RIB</td>
<td>routing information base</td>
</tr>
<tr>
<td>RPC</td>
<td>Remote Procedure Call</td>
</tr>
<tr>
<td>SDN</td>
<td>software-defined network</td>
</tr>
<tr>
<td>SNMP</td>
<td>Simple Network Management Protocol</td>
</tr>
<tr>
<td>SR OS</td>
<td>Nokia Service Router Operating System</td>
</tr>
<tr>
<td>SSH</td>
<td>Secure Shell</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
<tr>
<td>YANG</td>
<td>Yet Another Next Generation (data modeling language)</td>
</tr>
</tbody>
</table>

Disclaimer

Nokia operates a policy of ongoing development. This document provides preliminary information that is subject to change without notice. Nokia reserves the right to make changes and improvements to any of the products described in this document or withdraw this document at any time without prior notice. The contents of this document are proprietary and confidential property of Nokia. This document is provided subject to confidentiality obligations of the applicable agreement(s).

About Nokia

We create technology that helps the world act together.

As a trusted partner for critical networks, we are committed to innovation and technology leadership across mobile, fixed and cloud networks. We create value with intellectual property and long-term research, led by the award-winning Nokia Bell Labs.

Adhering to the highest standards of integrity and security, we help build the capabilities needed for a more productive, sustainable and inclusive world.

Nokia is a registered trademark of Nokia Corporation. Other product and company names mentioned herein may be trademarks or trade names of their respective owners.

© 2021 Nokia

Nokia OYJ
Karakaari 7
02610 Espoo
Finland
Tel. +358 (0) 10 44 88 000

Document code: (June) CID207070