Drive Network Automation Using Programmable YANG Data Models

Table 1: Feature History

Feature Name	Release Information	Feature Description
NCS 4000 YANG Data models	Cisco IOS XR Release 6.5.31	YANG data models are supported on NCS 4000 instead of CLI commands. The data models are written in an industry-defined language and is used to automate configuration tasks and retrieve operational data across heterogeneous devices in a network. Using model-driven programmability results in scalability.

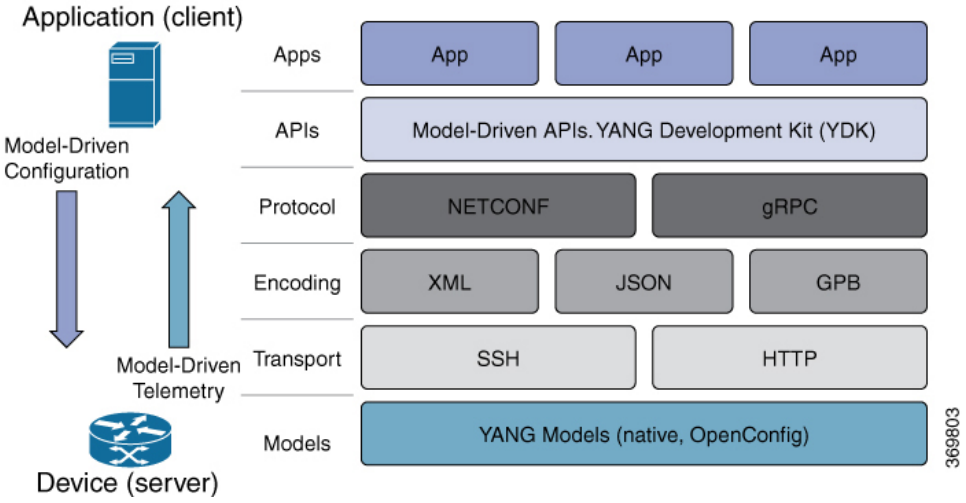
Typically, a network operation center is a heterogeneous mix of various devices at multiple layers of the network. Such network centers require bulk automated configurations to be accomplished seamlessly. CLIs are widely used for configuring and extracting the operational details of a router. But the general mechanism of CLI scraping is not flexible and optimal. Small changes in the configuration require rewriting scripts multiple times. Bulk configuration changes through CLIs are cumbersome and error-prone. These limitations restrict automation and scale. To overcome these limitations, you need an automated mechanism to manage your network.

Cisco IOS XR supports a programmatic way of configuring and collecting operational data of a network device using data models. They replace the process of manual configuration, which is proprietary, and highly text-based. The data models are written in an industry-defined language and is used to automate configuration task and retreive operational data across heterogeneous devices in a network. Although configurations using CLIs are easier and human-readable, automating the configuration using model-driven programmability results in scalability.

Model-driven programmability provides a simple, flexible and rich framework for device programmability. This programmability framework provides multiple choices to interface with an IOS XR device in terms of transport, protocol and encoding. These choices are decoupled from the models for greater flexibility.

The following image shows the layers in model-driven programmability:

Figure 1: Model-driven Programmability Layers



Data models provides access to the capabilities of the devices in a network using Network Configuration Protocol. The operations on the router are carried out by the protocols using YANG models to automate and programme operations in a network.

Benefits of Data Models

Configuring routers using data models overcomes drawbacks posed by traditional router management because the data models:

- Provide a common model for configuration and operational state data, and perform NETCONF actions.
- Use protocols to communicate with the routers to get, manipulate and delete configurations in a network.
- Automate configuration and operation of multiple routers across the network.

This article describes how you benefit from using data models to programmatically manage your network operations.

- YANG Data Model, on page 2
- Access the Data Models, on page 4
- Communication Protocols, on page 6

YANG Data Model

A YANG module defines a data model through the data of the router, and the hierarchical organization and constraints on that data. Each module is uniquely identified by a namespace URL. The YANG models describe the configuration and operational data, perform actions, remote procedure calls, and notifications for network devices.

The YANG models must be obtained from the router. The models define a valid structure for the data that is exchanged between the router and the client. The models are used by NETCONF enabled applications.

YANG models can be:

- **Cisco-specific models:** For a list of supported models and their representation.

For more details about YANG, refer RFC 6020 and 6087.

Components of a YANG Module

A YANG module defines a single data model. However, a module can reference definitions in other modules and sub-modules by using one of these statements:

- **import** imports external modules
- **include** includes one or more sub-modules
- **augment** provides augmentations to another module, and defines the placement of new nodes in the data model hierarchy
- **when** defines conditions under which new nodes are valid
- **prefix** references definitions in an imported module

The YANG models configure a feature, retrieve the operational state of the router, and perform actions.

Note The JSON data is based on YANG module name and not YANG namespace.

Structure of YANG Data Model

Data models handle the following types of requirements on routers (RFC 6244):

- **Configuration data:** A set of writable data that is required to transform a system from an initial default state into its current state. For example, configuring entries of the IP routing tables, configuring the interface MTU to use a specific value, configuring an ethernet interface to run at a given speed, and so on.
- **Operational state data:** A set of data that is obtained by the system at runtime and influences the behavior of the system in a manner similar to configuration data. However, in contrast to configuration data, operational state data is transient. The data is modified by interactions with internal components or other systems using specialized protocols. For example, entries obtained from routing protocols such as OSPF, attributes of the network interfaces, and so on.
- **Actions:** A set of NETCONF actions that support robust network-wide configuration transactions. When a change is attempted that affects multiple devices, the NETCONF actions simplify the management of failure scenarios, resulting in the ability to have transactions that will dependably succeed or fail atomically.

For more information about Data Models, see RFC 6244.

YANG data models can be represented in a hierarchical, tree-based structure with nodes. This representation makes the models easy to understand.

Each feature has a defined YANG model, which is synthesized from schemas. A model in a tree format includes:

- Top level nodes and their subtrees
- Subtrees that augment nodes in other YANG models

- Custom RPCs

YANG defines four node types. Each node has a name. Depending on the node type, the node either defines a value or contains a set of child nodes. The nodes types for data modeling are:

- leaf node - contains a single value of a specific type
- leaf-list node - contains a sequence of leaf nodes
- list node - contains a sequence of leaf-list entries, each of which is uniquely identified by one or more key leaves
- container node - contains a grouping of related nodes that have only child nodes, which can be any of the four node types

Access the Data Models

You can access the Cisco IOS XR native data models from GitHub, a software development platform that provides hosting services for version control.

You can also access the supported data models from the router. The router ships with the YANG files that define the data models. Use NETCONF protocol to view the data models available on the router using `ietf-netconf-monitoring` request.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <get>
    <filter>
      <netconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
        <schemas/>
      </netconf-state>
    </filter>
  </get>
</rpc>
```

All the supported YANG models are displayed as response to the RPC request.

```
<rpc-reply message-id="urn:uuid:74c81086-9706-4d59-b50e-25dc1a627c6b"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">

  <data>

    <netconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">

      <schemas>
        <schema>
          <identifier>Cisco-IOS-XR-ppp-ea-oper</identifier>
          <version>2015-11-09</version>
         <format>yang</format>
          <namespace>http://cisco.com/ns/yang/Cisco-IOS-XR-ppp-ea-oper</namespace>
          <location>NETCONF</location>
        </schema>

        <schema>
          <identifier>Cisco-IOS-XR-ppp-ea-oper-sub1</identifier>
          <version>2015-11-09</version>
          <format>yang</format>
          <namespace>http://cisco.com/ns/yang/Cisco-IOS-XR-ppp-ea-oper</namespace>
```

```
        <location>NETCONF</location>
      </schema>

      <schema>
        <identifier>Cisco-IOS-XR-ip-rsvp-oper</identifier>
        <version>2017-09-07</version>
        <format>yang</format>
        <namespace>http://cisco.com/ns/yang/Cisco-IOS-XR-ip-rsvp-oper</namespace>
        <location>NETCONF</location>
      </schema>

      <schema>
        <identifier>Cisco-IOS-XR-ip-rsvp-oper-sub1</identifier>
        <version>2017-09-07</version>
        <format>yang</format>
        <namespace>http://cisco.com/ns/yang/Cisco-IOS-XR-ip-rsvp-oper</namespace>
        <location>NETCONF</location>
      </schema>

      <schema>
        <identifier>Cisco-IOS-XR-ip-udp-oper</identifier>
        <version>2018-03-04</version>
        <format>yang</format>
        <namespace>http://cisco.com/ns/yang/Cisco-IOS-XR-ip-udp-oper</namespace>
        <location>NETCONF</location>
      </schema>

      <schema>
        <identifier>Cisco-IOS-XR-ip-udp-oper-sub4</identifier>
        <version>2018-03-04</version>
        <format>yang</format>
        <namespace>http://cisco.com/ns/yang/Cisco-IOS-XR-ip-udp-oper</namespace>
        <location>NETCONF</location>
      </schema>

      <schema>
        <identifier>Cisco-IOS-XR-ip-udp-oper-sub1</identifier>
        <version>2018-03-04</version>
        <format>yang</format>
        <namespace>http://cisco.com/ns/yang/Cisco-IOS-XR-ip-udp-oper</namespace>
        <location>NETCONF</location>
      </schema>

      <schema>
        <identifier>Cisco-IOS-XR-ip-udp-oper-sub3</identifier>
        <version>2018-03-04</version>
        <format>yang</format>
        <namespace>http://cisco.com/ns/yang/Cisco-IOS-XR-ip-udp-oper</namespace>
        <location>NETCONF</location>
      </schema>
            .
            .

      </schema>
        <identifier>openconfig-bgp-policy</identifier>
        <version>2017-02-02</version>
        <format>yang</format>
        <namespace>http://openconfig.net/yang/bgp-policy</namespace>
        <location>NETCONF</location>
      </schema>
    </schemas>
  </netconf-state>
 </data>
</rpc-reply>
```

The models are in the `.yang` format. A model with:

- `-oper` in the model name indicates an operational model. For example, `Cisco-IOS-XR-invmgr-oper.yang` is an operational model for inventory data.
- `-cfg` indicates a configuration model. For example, `Cisco-IOS-XR-controller-OTU-cfg.yang` is a configuration model for OTU.

Communication Protocols

Communication protocols establish connections between the router and the client. The protocols help the client to consume the YANG data models to, in turn, automate and programme network operations.

YANG uses the Network Configuration Protocol (NETCONF) .

The transport and encoding mechanisms for theis protocol are shown in the table:

Protocol	Transport	Encoding/ Decoding
NETCONF	ssh	xml
	tcp	json

NETCONF Protocol

NETCONF provides mechanisms to install, manipulate, or delete the configuration on network devices. It uses an Extensible Markup Language (XML)-based data encoding for the configuration data, as well as protocol messages. You use a simple NETCONF RPC-based (Remote Procedure Call) mechanism to facilitate communication between a client and a server. To get started with issuing NETCONF RPCs to configure network features using data models, see .