
# YANG MODEL

# Revision History

| Revision No. | Description | Editor | Date |
|---|---|---|---|
| 1.0 | YANG Modules | Rida Hanif | May 31, 2023 |

# Table of Contents

# Introduction

This document provides the key concept of YANG (Yet Another Next Generation) and its significance in network configuration and management. It explains the role of YANG modules, the relationship between YANG & YIN representations, and the integration of YANG with the SONIC network operating system. The intended audience is network administrators, device manufacturers, software developers, and researchers who are interested in understanding the fundamentals of YANG and its application in network management.

# Network Automation

Managing a heterogeneous network environment with various devices and layers can be challenging and time-consuming if relying solely on CLI scraping and manual configuration methods. The limitations of CLI-based approaches, as you mentioned, can hinder automation and scalability efforts. However, by adopting a more automated mechanism, such as network automation tools and technologies, you can overcome these limitations and efficiently manage your network operations.

Network automation provides a systematic and programmable approach to configuring, managing, and monitoring network devices. It allows you to automate repetitive tasks, streamline configuration changes, and extract operational details more efficiently and flexibly. Instead of relying on manual CLI interactions, network automation tools leverage standardized protocols and data models to interact with network devices, enabling seamless and bulk configuration changes.

# Benefits of Network Automation

**Templated Configuration** Network automation tools allow you to create configuration templates that can be easily customized and applied across multiple devices. This eliminates the need for rewriting scripts for every small configuration change, making the process more efficient and less error-prone.

**Programmability and APIs** Automation tools often provide programmable interfaces and APIs (such as NETCONF or REST APIs) that enable direct communication with network devices. This allows for easier integration, automation, and control of network operations, making bulk configuration changes more streamlined.

**Centralized Management** Network automation platforms provide centralized management capabilities, allowing you to have a holistic view of your network and efficiently manage configurations, device inventory, and operational tasks from a single interface. This improves visibility, control, and troubleshooting capabilities.

**Standardized Data Models** YANG, a data modeling language used in network automation, provides a standardized and structured representation of network configurations and operational state. This allows for easier interoperability and consistency across devices from different vendors.

**Policy-driven Automation** With network automation, you can define policies and rules that govern the network behavior and automate compliance checks. This ensures that configurations adhere to predefined standards and security requirements.

**Scalability and Flexibility** Network automation enables you to scale your operations as your network grows by automating repetitive tasks and leveraging programmable interfaces. This allows for faster provisioning, configuration changes, and troubleshooting across a large number of devices.

# Network Automation Data Models

Data modeling languages are used to define the structure, attributes, and relationships of data elements within a network. These languages provide a standardized way to represent network configurations, operational state, and management data. The data modeling languages commonly used in network automation include SNMP, YANG, YAML, and OpenConfig.

**SNMP (Simple Network Management Protocol)** SNMP is a widely used network management protocol that allows monitoring and management of network devices. While SNMP itself is not a data modeling language, it uses a specific data model called Management Information Base (MIB). MIB is a hierarchical structure that defines the objects and attributes that can be managed via SNMP. It specifies the structure and meaning of data accessible through SNMP and allows network administrators to retrieve and configure information on network devices.

**YANG (Yet Another Next Generation)** YANG is a data modeling language specifically designed for network management. It provides a standardized way to define the structure, attributes, and relationships of data elements in network devices. YANG models are used to represent network configurations, operational state, and management data. YANG models can be used in conjunction with protocols like NETCONF and RESTCONF for device configuration, monitoring, and management.

**YAML (YAML Ain't Markup Language)** YAML is a human-readable data serialization format that is often used for configuration files in network automation. While not a dedicated data modeling language, YAML provides a structured and readable format for representing data. It is commonly used in tools like Ansible for describing network configurations and automation tasks. YAML is known for its simplicity and ease of use, making it a popular choice for network automation.

**OpenConfig** OpenConfig is an industry initiative that aims to develop vendor-neutral data models for network automation. OpenConfig models are typically defined using YANG and provide a standardized way to represent network configurations and operational state. These models are designed to be vendor-agnostic, allowing network automation tools to work consistently across different vendors' devices. OpenConfig promotes interoperability and simplifies network automation by providing a common data modeling framework.

## What we use before YANG

Before YANG, network management relied heavily on SNMP (Simple Network Management Protocol) as the primary means of monitoring and managing network devices. SNMP is a protocol that allows network administrators to retrieve and configure information on network devices.

In 2002, the Internet Architecture Board (IAB) called attention to SNMP's disadvantages in configuration management, triggering the emergence of NETCONF. Although the NETCONF protocol is standardized, the data content is not. As a result, a better modeling language — YANG — was developed, making the data model simpler and easier to understand.

## Disadvantages of SNMP

**Lack of Structure** SNMP uses a hierarchical structure called Management Information Base (MIB) to define the objects and attributes that can be managed. However, MIBs are often vendor-specific and lack a standardized and consistent structure. This makes it challenging to develop interoperable network management solutions.

**Limited Data Modeling** SNMP's MIBs have limited capabilities for data modeling. They primarily focus on scalar variables and do not provide a robust mechanism for representing complex data structures and relationships. This limitation hinders the ability to model and manage modern network devices with rich configurations and operational states.

**Lack of Flexibility** SNMP's configuration and management operations are based on a set of predefined commands and responses. This rigid approach makes it difficult to extend SNMP for new requirements or custom configurations. Modifying or extending MIBs to support new functionality often requires complex and time-consuming processes.

**Complexity and Scalability** SNMP's underlying mechanisms, such as traps and polling, can become complex and resource-intensive in large-scale networks. The management of a heterogeneous mix of devices with different SNMP implementations can be challenging. SNMP lacks built-in mechanisms for data validation, constraints, and transactional operations.

**Interoperability Challenges** Due to the lack of standardized data models and vendor-specific implementations, achieving interoperability between different network devices and management systems can be difficult. This can hinder the development of comprehensive network automation solutions that work seamlessly across multiple vendors.

# YANG

YANG (Yet Another Next Generation) is a data modeling language used to define the structure, hierarchy, and semantics of data that is exchanged between network devices. YANG is used to model the configuration and state data of network devices, as well as operational parameters and event notifications. It provides a standardized way to describe the data elements, their types, relationships, and constraints. YANG is typically used for operations based on network configuration management protocols (such as NETCONF/RESTCONF)

The operations include configuration, status data, remote procedure calls (RPCs), and notifications. Compared with the SNMP model MIB, YANG is more hierarchical, can distinguish between configurations and status, and provides high extensibility.

YANG is defined in the following RFC standards: RFC 6020, RFC 6021, RFC 6991, RFC 7950.

Through ongoing standardization, YANG is gradually becoming a mainstream data description specification in the industry. Standards organizations, vendors, carriers, all define their own YANG models.The YANG model is integrated on the devices, which function as the servers. Network administrators can use NETCONF or RESTCONF to centrally manage, configure, and monitor various YANG-capable network devices, simplifying network O&M and reducing O&M costs.

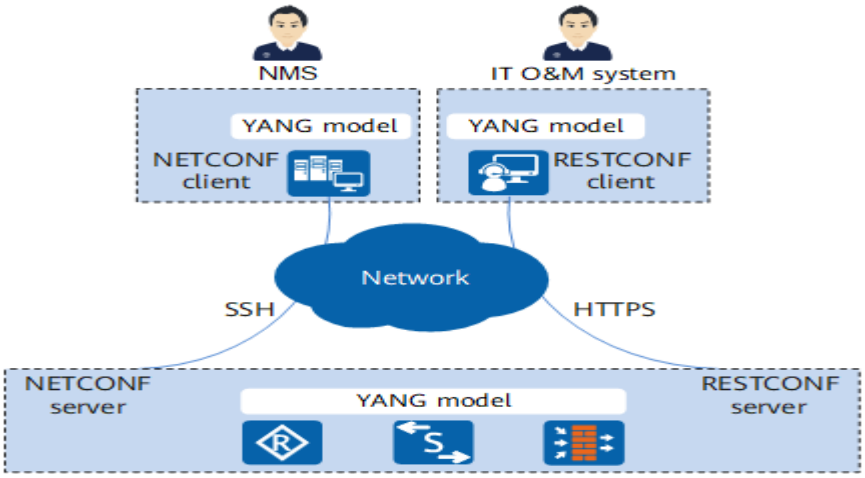



*Figure 01: Network management architecture based on NETCONF/RESTCONF and YANG*

# YANG Modules

YANG modules define the structure and semantics of data that can be exchanged between network devices or managed by network management systems. YANG modules typically have the file extension .yang.

A YANG module consists of various statements that define data elements, their relationships, attributes, and operations. Here are some key components of YANG modules:

**Module Statement** Every YANG module begins with a module statement that defines the module's name, namespace, and revision information. It serves as the top-level container for all other statements in the module.

**Data Definition Statements** YANG modules define data elements, also known as data nodes, using data definition statements. These statements include container, list, leaf, leaf-list, and choice, among others, to define the structure and data types of the elements.

**Data Modeling Statements** YANG modules provide various statements to specify the data modeling aspects, such as grouping, typedef, import, include, and extension statements. These statements enhance modularity, reusability, and extensibility of YANG modules.

**Operations Statements** YANG modules can include RPC (Remote Procedure Call) and notification statements to define operations that can be performed on network devices or to represent asynchronous events.

**Constraints and Validations** YANG modules allow the specification of constraints and validations on data nodes using statements like range, pattern, and must. These ensure data integrity and enforce data consistency.

**Metadata Statements** YANG modules support metadata statements such as description, reference, and organization. These statements provide additional information about the module and its elements.

**Module Dependencies** YANG modules can import or include other YANG modules to reuse definitions or extend existing data models. These dependencies are defined using import and include statements.

YANG modules play a crucial role in standardizing network management protocols, defining data models for network devices, and facilitating automation and interoperability in network operations. They are used in protocols like NETCONF, RESTCONF, and others to enable configuration, monitoring, and management of network devices in a consistent and vendor-agnostic manner.

# Module Statement

The Module Statement is a fundamental statement in a YANG module. It serves as the starting point and top-level container for all other statements in the module. The Module Statement defines the module's metadata, namespace, and revision information.

Break down of Module Statement are:

**<module-name>** Specifies the name of the YANG module.

**<namespace-URI>** Provides a unique identifier for the module in the form of a Uniform Resource Identifier (URI). It distinguishes the module from others to prevent naming conflicts.

**<module-prefix>** Defines a short prefix used to refer to data elements defined in the module. It helps avoid naming collisions when multiple modules are used together.

**<module-identifier>** Identifies another YANG module that is being imported for reuse.

**<file-name>** Specifies the name of a YANG module file to be included within the current module.

**<organization-name>** Indicates the organization or entity responsible for creating and maintaining the YANG module.

**<contact-information>** Provides contact details (such as an email address) for inquiries or feedback related to the YANG module.

**<module-description>** Describes the purpose and functionality of the YANG module.

**<reference-information>** Specifies any references, such as RFCs or other documentation, relevant to the YANG module.

**<revision-statement>** Represents a specific revision of the module, usually denoted by a revision date.

Sample Example:

```
module acme-system {
  namespace "http://acme.example.com/system";
  prefix "acme";

  import ietf-yang-types {
    prefix "yang";
  }

  include acme-types;

  organization "ACME Inc.";
  contact
    "Joe L. User

     ACME, Inc.
     42 Anywhere Drive
     Nowhere, CA 95134
     USA

     Phone: +1 800 555 0100
     EMail: joe@acme.example.com";

  description
    "The module for entities implementing the ACME protocol.";

  revision "2007-06-09" {
    description "Initial revision.";
  }

  // definitions follow...
}
```

# Data Definition Statements

Data Definition Statements in a YANG module are used to define the structure, data types, and semantics of data elements (also known as data nodes) within the module. These statements allow you to specify containers, lists, leaf nodes, leaf-list nodes, choices, and more. Here are some commonly used data definition statements in YANG:

**Container Statement** Defines a grouping of related data nodes within the module.

It serves as a hierarchical structure to organize other data nodes.

Syntax: container <container-name> { ... }

**List Statement** Defines an ordered collection of entries, each represented by a set of data nodes.

Lists allow multiple instances of the same data structure.

Syntax: list <list-name> { ... }

**Leaf Statement** Represents a single atomic value within the data hierarchy.

Specifies the data type and other properties (e.g., default value, units, constraints).

Syntax: leaf <leaf-name> { type <data-type>; ... }

**Leaf-list Statement** Represents an ordered list of atomic values.

Allows multiple values for the same data node.

Syntax: leaf-list <leaf-list-name> { type <data-type>; ... }

**Choice Statement** Defines a set of mutually exclusive data nodes.

Only one of the choice's branches can be present at a time.

Syntax: choice { ... }

**Case Statement** Used within a choice statement to define a branch representing a specific case.

Syntax: case <case-name> { ... }

**Grouping Statement** Defines a reusable grouping of data nodes.

Groupings can be included in multiple places within the module.

Syntax: grouping <grouping-name> { ... }

Sample Example:

```
module example-module {
  namespace "urn:example-module";
  prefix "exm";

  container example-container {
    leaf leaf-node {
      type string;
      description "A single atomic value within the container.";
    }
    leaf-list leaf-list-node {
      type int32;
      description "An ordered list of atomic values.";
    }
    list example-list {
      key "id";
      description "An ordered collection of entries.";
      leaf id {
        type string;
        description "Identifier for each entry.";
      }
      leaf name {
        type string;
        description "Name of each entry.";
      }
    }
    choice example-choice {
      description "A set of mutually exclusive data nodes.";
      case case1 {
        leaf case1-node {
          type string;
          description "Data node for case1.";
        }
      }
      case case2 {
        leaf case2-node {
          type int32;
          description "Data node for case2.";
        }
      }
    }
    grouping example-grouping {
      leaf grouping-node {
        type boolean;
        description "A reusable grouping node.";
      }
    }
  }
}
```

# Data Modeling Statements

Data Modeling Statements in a YANG module are used to enhance the modularity, reusability, and extensibility of the data model. These statements provide additional capabilities for defining data types, grouping related data nodes, importing external modules, and extending existing data models. Here are some commonly used data modeling statements in YANG:

**Typedef Statement** Defines a new data type based on an existing or built-in data type.Enables the creation of custom data types with specific constraints or semantics.

Syntax: typedef <typedef-name> { type <data-type>; ... }

**Grouping Statement** Defines a reusable grouping of data nodes.

Allows grouping related data nodes together for reuse in multiple places within the module.

Syntax: grouping <grouping-name> { ... }

**Import Statement** Imports definitions from another YANG module for reuse in the current module.Enables the composition of modules by incorporating definitions from other modules.

Syntax: import <module-identifier> { ... }

**Include Statement** Includes another YANG module file within the current module.

Useful for modularizing large data models or reusing portions of existing data models.

Syntax: include "<file-name>";

**Extension Statement** Defines an extension point that allows the extension of existing data models.Enables the addition of custom properties or behaviors to data nodes defined in other modules

Syntax: extension <extension-name> { ... }

**Feature Statement** Defines a named feature that can be used to enable or disable parts of the data model.Allows conditional inclusion or exclusion of certain data nodes based on feature support.

Syntax: feature <feature-name>

Sample Example:

```
module example-module {
  namespace "urn:example-module";
  prefix "exm";

  container example-container {
    leaf leaf-node {
      type string;
      description "A single atomic value within the container.";
    }
    leaf-list leaf-list-node {
      type int32;
      description "An ordered list of atomic values.";
    }
    list example-list {
      key "id";
      description "An ordered collection of entries.";
      leaf id {
        type string;
        description "Identifier for each entry.";
      }
      leaf name {
        type string;
        description "Name of each entry.";
      }
    }
    choice example-choice {
      description "A set of mutually exclusive data nodes.";
      case case1 {
        leaf case1-node {
          type string;
          description "Data node for case1.";
        }
      }
      case case2 {
        leaf case2-node {
          type int32;
          description "Data node for case2.";
        }
      }
    }
    grouping example-grouping {
      leaf grouping-node {
        type boolean;
        description "A reusable grouping node.";
      }
    }
  }
}
```

# Operations Statements

Operations Statements in a YANG module are used to define operations that can be performed on network devices or to represent asynchronous events. These statements enable the specification of remote procedure calls (RPCs) and notifications within the YANG data model. Here are some commonly used operations statements in YANG:

**RPC Statement** Defines a remote procedure call (RPC) that can be invoked on the network device.Specifies the input and output parameters of the RPC, including their data types and any associated constraints.

Syntax: rpc <rpc-name> { ... }

**Input Statement** Defines the input parameters of an RPC. Specifies the data nodes that represent the input values required for invoking the RPC.

Syntax: input { ... }

**Output Statement** Defines the output parameters of an RPC. Specifies the data nodes that represent the output values returned by the RPC.

Syntax: output { ... }

**Notification Statement** Represents an asynchronous event that can be sent by the network device.Defines the data nodes and their associated data types that comprise the notification payload.

Syntax: notification <notification-name> { ... }

**Augment Statement** Allows the addition of new data nodes to an existing data model defined in another module.Useful for extending existing data models to support additional operations or events.

Syntax: augment  <target-node> { ... }

Sample Example:

```
module example-operations {
  namespace "http://example-operations";
  prefix "exop";

  import ietf-inet-types {
    prefix "inet";
  }

  // RPC statement
  rpc my-rpc {
    description "Example RPC";
    input {
      leaf input-parameter {
        type string;
      }
    }
    output {
      leaf output-parameter {
        type inet:ipv4-address;
      }
    }
  }
  notification my-notification {
    description "Example notification";
    leaf notification-parameter {
      type uint32;
    }
  }
  augment "/some-existing-node" {
    description "Example augmentation";
    container augmented-container {
      leaf augmented-leaf {
        type string;
      }
    }
  }
}
```

# Constraints and Validations Statements

Constraints and validations in a YANG module are used to enforce rules and conditions on the data that is being modeled. These statements help ensure that the data conforms to specific requirements and constraints. Here are some commonly used constraints and validations statements in YANG:

**Range Statement** Specifies a numerical range constraint on a leaf node.

Ensures that the value of the leaf node falls within the specified range.

Syntax: range "<min-value>.."<max-value>";

**Length Statement** Defines the minimum and maximum lengths for a string or binary leaf node.Validates that the length of the value falls within the specified range.

Syntax: length "<min-length>.."<max-length>";

Pattern Statement Specifies a regular expression pattern that the value of a leaf node must match.Validates the format or structure of the value against the defined pattern.

Syntax: pattern "<regular-expression>";

**Enum Statement** Defines a set of allowed enumeration values for a leaf node.

Ensures that the value of the leaf node matches one of the defined enumeration values

Syntax: enum <enum-name> { ... }

**Mandatory Statement** Specifies that a particular leaf or choice node must have a value.Ensures that the presence of the node is required in the data instance.

Syntax: mandatory true;

**When Statement** Defines a conditional constraint based on the values of other data nodes.Specifies that the constraints within the "when" statement are valid only when the condition evaluates to true.

Syntax: when <condition> { ... }

**Unique Statement** Specifies that the values of the key leaf node within a list must be unique.Ensures that there are no duplicate values for the specified key leaf.

Syntax: unique "<key-leaf>";

**Must statement** Specifies a constraint or a validation rule on a data node's value. It specifies a condition that must be satisfied for the data node to be considered valid. If the condition evaluates to false, the data node is considered invalid.

Syntax: must <condition> { ... }

error-message { ... }

```
module example {
  namespace "http://example.com";
  prefix ex;

  container data {
    leaf temperature {
      type int32;
      range "0".."100";
    }

    leaf description {
      type string;
      length "1".."255";
    }

    leaf name {
      type string;
      pattern "[A-Za-z0-9]+";
    }

    leaf status {
      type enumeration {
        enum active;
        enum inactive;
      }
      mandatory true;
    }

    list users {
      key "id";
      unique "id";

      leaf id {
        type int32;
      }

      leaf username {
        type string;
      }

      leaf password {
        type string;
        length "8".."20";
        must "string-length(../username) > 0";
      }
    }
  }
}
```

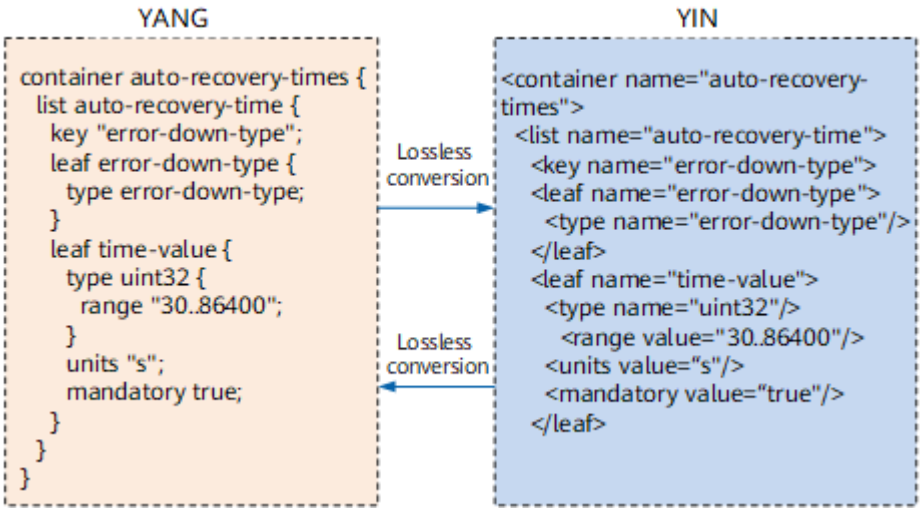
# YANG & YIN

YIN (YANG Independent Notation) is an XML-based format that represents YANG modules in a more compact and machine-readable form. YIN is derived from YANG and provides an alternative representation of YANG modules that can be easily parsed and processed by software applications. YIN files have the file extension ".yin" and are often used for exchanging YANG modules between different tools and systems.

YANG and YIN are closely related, and YIN is considered a serialization of YANG. YIN provides a more concise and optimized representation of YANG modules, making it easier to parse and process the module's structure and semantics. YIN files can be converted back to YANG format if needed.

YIN files follow the XML syntax and structure, which makes them suitable for parsing and manipulation using XML parsers. Each YANG statement and construct is represented using XML elements and attributes. The hierarchical structure of YANG modules is preserved in the YIN representation, allowing tools and applications to navigate and extract information from the module.

Together, YANG and YIN provide a powerful set of tools for modeling, managing, and exchanging configuration and operational data in network devices and systems.



# YANG & SONiC

In SONIC (Software for Open Networking in the Cloud), YANG models play a crucial role in defining the configuration, operational state, and management interfaces of network devices. YANG Model in SONiC provides following support:

**Standardization** YANG provides a standardized way to describe the structure, semantics, and behavior of network devices. By using YANG models, SONIC ensures consistency and interoperability across different network devices from various vendors. It allows network operators to manage and configure devices in a consistent manner, regardless of the underlying hardware or software implementation.

**Modular and Hierarchical Representation** SONIC leverages the hierarchical and modular nature of YANG to represent different aspects of network devices, such as interfaces, routing protocols, VLANs, QoS policies, and more. This modular approach makes it easier to understand, organize, and manage complex network configurations and functionalities.

**Configuration Management** YANG models in SONIC define the configuration parameters and settings for network devices. These models specify the valid values, data types, and constraints for each configuration item, ensuring that only valid configurations are applied. YANG models also provide mechanisms to validate and verify the correctness of configuration changes before applying them to the network devices, reducing the risk of misconfigurations and network disruptions.

**Operational State Monitoring** SONIC utilizes YANG models to represent the operational state of network devices. These models define the data elements and attributes that provide real-time information about the health, performance, and status of the network devices. By monitoring the operational state data, SONIC can proactively detect issues, collect performance metrics, and provide visibility into the network's behavior.

**Management Interfaces** YANG models serve as the basis for management interfaces in SONIC. They define the RPC (Remote Procedure Call) operations that allow network operators to interact with the devices, such as configuring parameters, retrieving operational information, and triggering actions. The management interfaces built using YANG models provide a standardized and programmable way to manage and control the network devices in SONIC.

# Conclusion

YANG modules serve as the foundation for defining network data models. These modules specify the structure, attributes, and constraints of network elements, allowing for consistent representation and management across diverse devices and vendors. The hierarchical nature of YANG modules enables a clear and organized depiction of network configurations and operational states, facilitating efficient communication and collaboration among stakeholders.

# Reference


https://www.juniper.net/documentation/us/en/software/junos/netconf/topics/concept/netconf-yang-module-junos-extension.html

https://info.support.huawei.com/info-finder/encyclopedia/en/YANG.html

https://developer.cisco.com/docs/nso/guides/#!the-yang-data-modeling-language/

https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/datamodels/configuration/xe-16/data-models-xe-16-book/yang-netconf.pdf