YANG data models

A YANG data model is a standardized network modeling language that

- defines the structure and constraints for configuration and operational data on network devices,
- enables automated setup and management of heterogeneous networks, and
- supports communication using protocols such as NETCONF and gRPC for scalable, consistent network operations.

Model-driven programmability

Model-driven programmability uses YANG data models to provide a flexible, rich framework for device automation. This framework offers multiple ways to interface with Cisco IOS XR devices via different transports, protocols, and encodings, which are decoupled from the data models for greater flexibility.

Data model layers

YANG data models organize device functions and configurations into logical layers. For example, one layer may define device interfaces, while another handles routing protocols. This separation simplifies automation and standardizes management across device types.

Protocol operations

YANG data models work in conjunction with protocols such as Network Configuration Protocol (NETCONF) and gRPC. These protocols use YANG models to access device capabilities, automate configuration, and retrieve operational data across the network.

Benefits of YANG data models

Configuring routers with YANG data models overcomes traditional limitations because these models:

- Provide a common structure for both configuration and operational data, and support NETCONF actions.
- Enable protocols to get, manipulate, and delete network device configuration.
- Automate management and operation of multiple routers throughout a heterogeneous network.

Additional information

Traditional CLI-based configuration is typically proprietary and highly text-based, making bulk configuration challenging in complex networks. By contrast, YANG data models use industry-standard languages to facilitate automation, interoperability, and operational consistency.

YANG data models

A YANG data model is a data modeling language specification that

- defines a standard, hierarchical structure for the configuration and operational data of network devices,
- enables robust network communication by specifying data relationships, constraints, actions, and notifications, and
- supports integration with network management protocols such as NETCONF and gRPC for automated configuration and monitoring.

- **YANG module**: A file or group of files that together define a single data model. Each module is uniquely identified by a namespace URL.
- **NETCONF/gRPC**: Protocols that use YANG data models for configuration and operational data exchange.

YANG data models must be obtained from the router. These models define a valid structure for the data exchanged between the router and the client, and are consumed by NETCONF and gRPC-enabled applications (gRPC supported only on 64-bit platforms). YANG models are categorized as follows:

- **Cisco-specific models**: Proprietary YANG models unique to Cisco devices. For details and representation, see Native models.
- **Common models (Open Config/OC)**: Industry-standard YANG models, typically from organizations such as IETF and IEEE. OC models have separate YANG modules for configuration data, operational data, and actions. See OC models for examples.

All YANG data models are stamped with semantic version 1.0.0 as the baseline from release 7.0.1 and later. For more details on YANG, refer to RFC 6020 and RFC 6087.

Data model requirements

YANG data models handle these types of requirements on routers (RFC 6244):

- **Configuration data**: A set of writable data required to transform a system from its initial state. For example, configuring entries in the IP routing table, or setting interface MTU or speed.
- **Operational state data**: Data obtained by the system at runtime, reflecting its operational status, which is typically transient and influenced by internal or external events (for example, OSPF routing entries).
- **Actions**: Supported via NETCONF actions to enable robust network-wide configuration transactions, ensuring atomic changes across devices.

For more information, see *RFC 6244*.

YANG model structure

YANG data models use a tree-based, hierarchical structure with nodes, making the models easy to understand and navigate. Each feature typically has a synthesized YANG model built from schemas. A model in tree format includes:

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

YANG node types

YANG defines these four node types for data modeling:

- **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 uniquely identified by one or more key leaves.
- **Container node**: Contains a grouping of related nodes (which may be any of the four types).

Each node is named and either defines a value or contains child nodes according to its type.

Components of a YANG module

A YANG module defines a single data model but can reference other modules or sub-modules. These standard statements are used:

- **Import**: Imports external modules.
- **Include**: Includes one or more sub-modules.
- **Augment**: Adds new definitions to another module at specified locations.
- **When**: Sets conditions under which new nodes are valid.
- **Prefix**: References definitions in an imported module.

Note The gRPC YANG path or JSON data is based on the YANG module name, not the namespace.

Additional reference information

- All data models from release 7.0.1 and later are stamped with semantic version 1.0.0.
- For further reading on YANG, see RFC 6020 and RFC 6087.
- For industry-standard open YANG models, see the OpenConfig public repository.

Access data models

Cisco IOS XR routers ship with YANG files that define supported data models. You can use the NETCONF protocol and the ietf-netconf-monitoring request to view these models directly from the router.

Before you begin

- Ensure NETCONF is enabled on your router.
- Verify you have credentials for NETCONF client access.

Procedure

Step 1 Connect to your Cisco IOS XR router using a NETCONF client.

Step 2 Send an **ietf-netconf-monitoring** RPC request to retrieve the list of supported models.

Example:

Example RPC request:

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

Step 3 Review the RPC response to see the available YANG models and details such as identifier, version, format, namespace, and location.

Example:

Example RPC response snippet:

```
<rpc-reply message-id="101" xmlns="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-crypto-sam-oper</identifier>
<version>1.0.0</version>
<format>yang</format>
<namespace>http://cisco.com/ns/yang/Cisco-IOS-XR-crypto-sam-oper</namespace>
<location>NETCONF</location>
</schema>
<schema>
<identifier>openconfig-mpls-ldp</identifier>
<version>1.0.0</version>
<format>yang</format>
<namespace>http://openconfig.net/yang/ldp</namespace>
<location>NETCONF</location>
</schema>
<!-- Additional schema entries... -->
</schemas>
</netconf-state>
</data>
</rpc-reply>
```

YANG action

A YANG action is a network management operation that

- is defined in a YANG model using an RPC statement
- enables execution of specific commands or operations (such as ping or reload) on a network device via standardized management protocols like NETCONF or gRPC, and
- provides structured responses indicating the outcome of the requested operation.

YANG actions allow remote management systems to execute device-specific operations programmatically. Each action is modeled as an RPC (Remote Procedure Call) statement within a YANG module, making it accessible through automation tools and network controllers that communicate over NETCONF (using XML) or gRPC (using JSON). When an action request is received, the device executes the operation and returns a protocol-specific response. For example, common actions include "ping," "traceroute," "copy," and process restart commands.

Supported YANG actions and models

The following table lists common YANG actions and the associated YANG models. For a complete list of supported actions, refer to the YANG Data Models Navigator.

Table 1: YANG actions and models

Actions	YANG Models
logmsg	Cisco-IOS-XR-syslog-act
snmp	Cisco-IOS-XR-snmp-test-trap-act
rollback	Cisco-IOS-XR-cfgmgr-rollback-act
clear isis	Cisco-IOS-XR-isis-act
clear bgp	Cisco-IOS-XR-ipv4-bgp-act

PING NETCONF action

This example shows the IOS XR NETCONF action request to run the ping command on the router.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <ping xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ping-act">
  <destination>
   <destination>1.2.3.4</destination>
  </destination>
 </ping>
</rpc>
```

This section shows the NETCONF action response from the router.

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <ping-response xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ping-act">
  <ipv4>
```

```
   <destination>1.2.3.4</destination>
   <repeat-count>5</repeat-count>
   <data-size>100</data-size>
   <timeout>2</timeout>
   <pattern>0xabcd</pattern>
   <rotate-pattern>0</rotate-pattern>
   <reply-list>
    <result>!</result>
    <result>!</result>
    <result>!</result>
    <result>!</result>
    <result>!</result>
   </reply-list>
   <hits>5</hits>
   <total>5</total>
   <success-rate>100</success-rate>
   <rtt-min>1</rtt-min>
   <rtt-avg>1</rtt-avg>
   <rtt-max>1</rtt-max>
  </ipv4>
 </ping-response>
</rpc-reply>
```

XR process restart action

This example shows the process restart action sent to NETCONF agent.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <sysmgr-process-restart xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-sysmgr-act">
     <process-name>processmgr</process-name>
     <location>0/RP0/CPU0</location>
  </sysmgr-process-restart>
</rpc>
```

This example shows the action response received from the NETCONF agent.

```
<?xml version="1.0"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <ok/>
</rpc-reply>
```

Copy action

This example shows the RPC request and response for `copy` action:

RPC request:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <copy xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-shellutil-copy-act">
    <sourcename>//root:<location>/100MB.txt</sourcename>
    <destinationname>/</destinationname>
    <sourcefilesystem>ftp:</sourcefilesystem>
    <destinationfilesystem>harddisk:</destinationfilesystem>
    <destinationlocation>0/RSP1/CPU0</destinationlocation>
  </copy>
</rpc>
```

RPC response:

```
<?xml version="1.0"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
```

```
 <response xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-shellutil-copy-act">Successfully
completed copy operation</response>
</rpc-reply>

8.261830565s elapsed
```

Delete action

This example shows the RPC request and response for `delete` action:

RPC request:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
<delete xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-shellutil-delete-act">
    <name>harddisk:/netconf.txt</name>
  </delete>
</rpc>
```

RPC response:

```
<?xml version="1.0"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <response xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-shellutil-delete-act">Successfully
 completed delete operation</response>
</rpc-reply>

395.099948ms elapsed
```

Supported protocols and operational restrictions

- NETCONF actions use XML formatting; gRPC actions use JSON.
- Some actions—such as installing software—may have restrictions (for example, deprecated commands or maximum parameter limits).
- System admin models support only <get> and <get-config> operations; <edit-config> supports only merge. Delete, remove, and replace are not supported for system admin models.

YANG actions and data elements

A YANG action acts like a remote control button: when pressed (invoked through a protocol), the device performs the specified operation and returns the result, similar to how an ATM processes withdrawal commands remotely.

Configuration containers or leaves in YANG are not actions—they only represent data to store or retrieve, not executable operations.

YANG input validators and Get requests

A YANG input validator is a system component that:

- checks the validity of configuration data against OpenConfig YANG models
- verifies Get requests processed through protocols such as NETCONF or gNMI
- ensures that only explicitly configured OpenConfig leaves are returned in response to Get requests.

A Get request is an operation that:

- retrieves configuration or operational data from a network device,
- uses management protocols like NETCONF or gNMI
- returns either OpenConfig leaves or Cisco native model items based on system settings and operational modes.

YANG input validators and Get requests ensure configuration integrity and accurate state retrieval in Cisco IOS XR devices. Recent enhancements to these components result in stricter input validation, with OpenConfig models providing consistent handling of configuration queries. By default, Get requests now return only leaves explicitly configured via OpenConfig, improving data accuracy and compliance.

Legacy mode remains available for limited releases, preserving previous behaviors where Get requests could return all convertible Cisco native items.

Table 2: Feature History Table

Feature Name	Release Information	Description
Improved YANG Input Validator and Get Requests	Release 7.10.1	The OpenConfig data models provide a structure for managing networks via YANG protocols. With this release, enhancements to the configuration architecture improve input validations and ensure that the Get requests made through gNMI or NETCONF protocols return only explicitly configured OpenConfig leaves. Previously, Get requests returned all the items in the Cisco native data models that the system could convert into OpenConfig items, regardless of whether they were initially configured via OpenConfig. We have added a new legacy mode option for a limited number of releases which helps you preserve this behaviour.

Legacy Mode Usage

- NETCONF: Add a legacy mode attribute to the **<get-config>** request **tag:get-config xmlns:xr-md="http://cisco.com/ns/yang/cisco-xr-metadata" xr-md:mode="legacy"**
- gNMI: Set the origin to **openconfig-legacy** in the request.

By default, OpenConfig leaves now return default values consistently using Explicit Basic Mode (see RFC6243).

YANG input validators usage guidelines and limitations

Use these best practices and follow the warnings when working with YANG input validators and OpenConfig in Cisco IOS XR:

Usage guidelines

Consider the following usage guidelines:

- After upgrading to Cisco IOS XR Release 7.10.1 or later, always commit OpenConfig changes to ensure OpenConfig leaves appear in Get requests.
- Use either gNMI or NETCONF as the management agent for OpenConfig, but do not use both simultaneously. The first successful commit determines the active agent.
- Fully configure each feature via OpenConfig, native model, or CLI. Items overridden in native models do not appear in the OpenConfig view.

Operational limitations

- Observe the same commitment requirements when downgrading or upgrading the software.
- The non-active agent can only configure native model items or perform Get requests. It cannot modify OpenConfig items until all OpenConfig leaves are first removed by the active agent.
- Changes made using CLI or Config Scripts are not reflected in OpenConfig system views.
- During commits, you can issue Get requests only on the running datastore. Edits, commits, or Get requests on candidate datastores belonging to other clients are rejected.

Warnings

- The command `show running-config | (xml | json) openconfig` displays the running OpenConfig configuration but cannot be filtered using XR CLI configuration keywords. Starting from Cisco IOS XR Release 2.4.4.1, this command is not supported.
- Do not use `load rollback changes` or `load commit changes` for rollbacks or commits that include OpenConfig leaves; these operations are not supported for OpenConfig data.
- System events, such as install operations or startup configuration failures, may remove configurations from the system, leaving OpenConfig views temporarily unsynchronized. Watch for syslog messages and reapply OpenConfig configurations when needed.
- Performing a `Commit Replace` operation through CLI removes all OpenConfig entries from the system.

Communication protocols

A communication protocol is a set of rules that

- establishes standardized exchanges of information between network devices,
- enables reliable and secure communication between clients and routers, and
- facilitates automation and programmable operations across different platforms.

In YANG-based systems, communication protocols connect the router and the client. These protocols enable clients to consume YANG data models, automating and programming network operations.

Supported protocols

YANG uses one of these protocols:

- Network Configuration Protocol (NETCONF)
- RPC framework (gRPC) by Google

Note gRPC is supported only on 64-bit platforms.

The transport and encoding mechanisms for these two protocols are shown in the table:

Table 3: Protocols and their supported transport and encoding or decoding mechanisms

Protocol	Transport	Encoding or Decoding
NETCONF	SSH	XML
gRPC	HTTP/2	JSON

NETCONF protocols

A NETCONF protocol is a network management protocol family that

- enables installation, modification, and deletion of device configuration data,
- uses XML-based data encoding for both configuration and protocol messages, and
- supports client-server communication through a simple RPC (Remote Procedure Call) mechanism.

NETCONF protocols provide standard tools for automating the management of network devices. Clients can programmatically update or retrieve configuration and operational data by exchanging XML-formatted messages with servers.

A NETCONF client issues an RPC call to configure a new interface, and the NETCONF server responds with an XML message confirming the change.

gRPC protocols

A gRPC protocol is a network communication framework that

- is based on Protocol Buffers (Protobuf), an open-source binary serialization format,
- enables flexible, efficient, and automated serialization of structured data for remote procedure calls (RPCs), and
- requires you to define message types using `.proto` files, where each message contains a series of name-value pairs for clear data structure.

Additional reference information: gRPC is open source and offers a lightweight alternative to traditional protocols like XML for transmitting data. Its design aims for both simplicity and high performance in exchanging data between systems.

Note gRPC is supported only on 64-bit platforms.

Examples:

- A network management system can use gRPC protocols to issue NETCONF RPCs and configure device features using data models.
- Developers use `.proto` files to define the messages and service methods that gRPC clients and servers exchange.
- REST APIs typically use JSON payloads and HTTP methods and do not rely on Protocol Buffers or `.proto` definitions as gRPC does.

Unified data models

CLI-based YANG data models, also known as unified configuration models are introduced in Cisco IOS XR Software Release 7.0.1. The unified models provide a full coverage of the router functionality, and serves as a single abstraction for YANG and CLI commands. Unified models are generated from the CLI and replaces the native schema-based models.

The unified models are available in `pkg/yang` location. The presence of `um` in the model name indicates that the model is a unified model. For example, Cisco-IOS-XR-**um**-<feature>-cfg.yang.

You can access the models supported on the router using the following command:

```
Router#run
[node]$cd /pkg/yang
[node:pkg/yang]$ls
```

The unified models are also available in the Github repository.