

Getting started with model-driven programmability on Cisco Nexus 9000 Series Switches

Contents

YANG data model introduction	3
YANG models supported by Cisco Nexus 9000 Series NX-OS	5
Network Management Protocol for model-driven programmability	5
NETCONF Agent introduction.....	6
NETCONF transport and messages.....	6
NETCONF data store and operations	8
Remote-Procedure Call (RPC) contents	10
Benefits of NETCONF	10
RESTCONF Agent introduction	10
gRPC Agent introduction	11
Start model-driven programmability on Cisco Nexus 9000 Series Switches using OpenConfig YANG models	11
Step 1: Load the Cisco Nexus 9000 Series Switch with YANG model RPM packages	11
Step 2: Prepare YANG tools.....	14
Step 2a. Install and start Yang Explorer.	14
Step 2b. Login to Yang Explorer.....	14
Step 2c. Create a device profile, and load the device profile	15
Step 2d. Browse YANG Data models.	16
Step 2e. Generate/execute RPC payloads	17
Step 3: Start model-driven programmability using OpenConfig YANG model	19
Start model-driven programmability on a Cisco Nexus 9000 Series Switch using native YANG models	21
Sandbox usage example 1: Sandbox and NETCONF	21
Sandbox usage example 2: Sandbox and RESTCONF	24
Customer use-case example with a YANG model	26
Advanced tools for model-driven programmability on Cisco Nexus 9000 Series Switches using YANG models	26
Summary	27
References	27

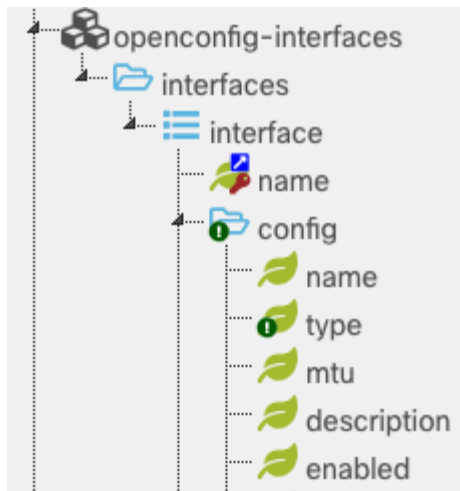
Model-driven programmability of Cisco® NX-OS Software devices allows you to automate the configuration and control of those devices. Data models are written in a standard, industry-defined language. Data modeling provides a programmatic and standards-based method of writing configurations to network devices, replacing the process of manual configuration. Although configuration using a Command-Line Interface (CLI) may be more human-friendly, automating the configuration using data models results in better scalability. This white paper first explains the basic knowledge of YANG models and Network Management Protocols such as NETCONF, RESTCONF, gRPC, then uses a Cisco Nexus® 9000 Series Switch as example to show you how to start using model-driven programmability with NETCONF, RESTCONF, native YANG, and OpenConfig YANG data models.

YANG data model introduction

The Cisco Nexus 9000 Series NX-OS Software devices support the YANG data model. YANG is a data-modeling language used to describe network device configuration and operational data. YANG was developed by the Internet Engineering Task Force (IETF). YANG models the hierarchical organization of data as a tree in which each node has a name and either a value or a set of child nodes. YANG provides clear and concise descriptions of the nodes and of the interaction between them. Details about YANG can be found at <https://tools.ietf.org/html/rfc6020>.

YANG defines four types of nodes for data modeling: Figure 1 shows a sample container and leaf node snippet.

Figure 1. Container and leaf node snippet



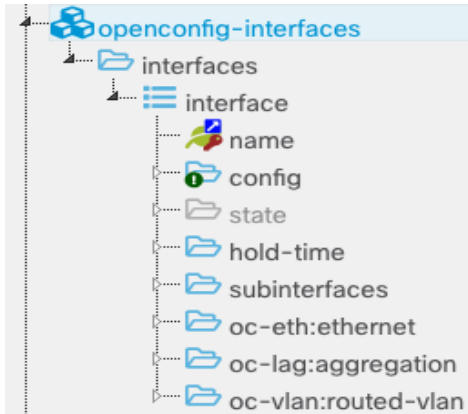
A container node is used to group related nodes in a subtree. A container has only child nodes and no value. A container may contain any number of child nodes of any type (including leaves, lists, containers, and leaf-lists). Using Figure 1a as an example, in the “openconfig-interfaces” model there is a top-level container “interfaces,” a list “interface” (explained in Figure 1b), and a child container “config.”

A leaf node contains simple data, such as an integer or a character string; it has exactly one value of a particular type and no child nodes. In Figure 1a, interface “name” inside the “config” container is a leaf node.

A list defines a sequence of list entries. Each entry is like a structure or a record instance, and is uniquely identified by the values of its key leafs. A list can define multiple key leafs and may contain any number of child nodes of any type (including leafs, lists, containers, etc.).

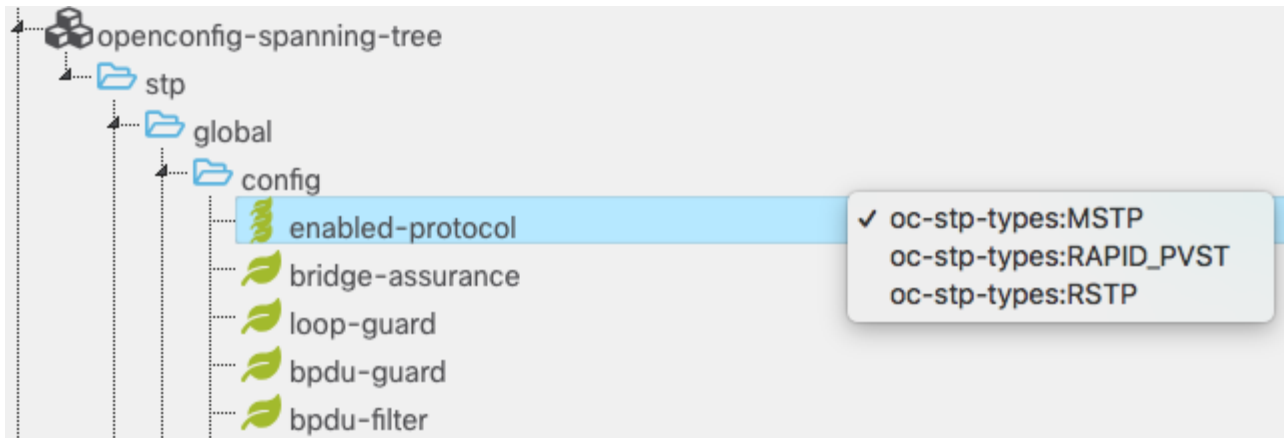
In Figure 2 below, the “openconfig-interfaces” model snippet shows a top-level container called “interfaces,” which contains a list called “interface”; this in turn contains a key “name” that identifies a specific interface, and a number of child container nodes, such as “config,” “hold-time,” etc.

Figure 2. “interface” list



A leaf list is a sequence of leaf nodes with exactly one value of a particular type per leaf. Figure 3 shows a leaf-list example with a Spanning Tree Protocol configuration. The “enabled-protocol” can be only one value from the “MSTP,” “RAPID_PVST,” or “RSTP.”

Figure 3. Example of a leaf list



YANG models supported by Cisco Nexus 9000 Series NX-OS

Cisco Nexus 9000 Series NX-OS supports both open YANG models and native YANG models. A Cisco native YANG model (or “native YANG model,” as the phrase will be used in this document) is defined in YANG data-modeling language but specific to NX-OS. An open YANG model is defined by various standards bodies and industry consortia: for example, the IEEE YANG model, IETF YANG model, and OpenConfig YANG model, etc. OpenConfig is an informal working group of network operators sharing the goal of moving networks toward a more dynamic, programmable infrastructure by adopting software-defined networking principles such as declarative configuration and model-driven management and operations^[1]. OpenConfig is compiling a consistent set of vendor-neutral OpenConfig YANG data models based on actual operational needs from use cases and requirements from multiple network operators. Compared with IEEE and IETF YANG models, the OpenConfig YANG model has more defined models; it also imports some of the existing IEEE and IETF YANG models into its models. For example, “openconfig-interfaces” model imports the “ietf-interfaces” model definition defined by IETF. OpenConfig YANG data models are published at: <https://github.com/openconfig/public/tree/master/release>.

Please note that the OpenConfig YANG data models are updated frequently, with the latest revision number at this GitHub directory. Each vendor may not support the latest revision of the OpenConfig model. Because the OpenConfig model is vendor-neutral, the models are the least-common denominators among vendors; hence, the number of OpenConfig models is much fewer than the number of native models. For more a complete set of feature support, and for support of specific Cisco NX-OS Software features, using a native YANG model with NX-OS is recommended.

For Cisco Nexus 9000 Series Switches, beginning with Cisco Nexus NX-OS 7.0(3)I6(2), a Cisco native YANG model is included in the NX-OS image and installed automatically when the image is loaded. However, OpenConfig YANG models are not included in the NX-OS image by default. Corresponding OpenConfig YANG RPM packages need to be downloaded from the Cisco Artifactory and installed on the switch. RPM Package Manager is a program for installing, uninstalling, and managing software packages in Linux. The OpenConfig YANG model RPM packages are organized by NX-OS release-specific directories and support specific revisions of the OpenConfig model defined by the OpenConfig consortium. Ensure that you are downloading the correct RPM packages from the corresponding NX-OS release directory: <https://devhub.cisco.com/artifactory/open-nxos-agents>

Please refer to the “Cisco Nexus 9000 Series NX-OS Programmability Guide” (the chapter on “Model-Driven Programmability”) for procedures on installing RPM packages.

<https://www.cisco.com/c/en/us/support/switches/nexus-9000-series-switches/products-programming-reference-guides-list.html>

Network Management Protocol for model-driven programmability

To manipulate and automate on the data models supported on the Cisco Nexus 9000 Series Switches, a Network Management Protocol needs to be used between the Application Client and the Nexus 9000 switches. Cisco NX-OS Nexus 9000 Series Switches support NETCONF, RESTCONF, and gRPC Network Management Protocols via corresponding Programmable Interface Agents: NETCONF, RESTCONF or gRPC Agent.

When a request from client is received via NETCONF, RESTConf, or gRPC protocol, the corresponding Programmable Interface Agent converts the request into an abstract message object that is distributed to the underlying model infrastructure based on the namespace in the request. Using the namespace, the appropriate model is selected and the request is passed to it for processing. The model infrastructure executes the request (read or write) on the device data store returning the results to the Agent of origin for response transmission back to the requesting client.

For Cisco Nexus 9000 Series Switches, beginning with NX-OS 7.0(3)I6(2), the NX-OS Programmable Interface Agent Component RPM packages are included in the NX-OS image and installed automatically when the image is loaded. However, by default the Agents are not enabled. To start the Agent, you need to enable the corresponding Agent by issuing following commands:

```
Enable NETCONF Agent: feature netconf
Enable RESTCONF Agent: feature restconf
Enable gRPC Agent: feature grpc
```

Table 1 shows that each Agent supports a different transport and protocol and the encoding of communication with the Nexus 9000 device, providing different interfaces for configuration management of the device via YANG models.

Table 1. NX-OS Programmable Interface Agents^[2]

Agent	Transport	Protocol	Encoding
NETCONF	SSH	RFC 6241	XML
RESTCONF	HTTP	draft-ietf-netconf-restconf-10	XML or JSON
gRPC	HTTP	gRPC Protocol Spec	Google Protobuf

NETCONF Agent introduction

The Cisco NX-OS NETCONF Agent is a client-facing interface that provides secure transport for client requests and server responses in the form of YANG models, encoded in XML. It is based on the NETCONF protocol RFC 6241^[3] and implements secure transport, based on RFC 6242. The NETCONF Agent facilitates retrieval of operational states and manipulation of configuration data in the data store via YANG models (RFC 6020^[4]).

NETCONF transport and messages

NETCONF uses a client-server model with Remote Procedure Call (RPC) paradigm. First, the NETCONF client establishes a Secure Shell (SSH) protocol connection with the NETCONF server.

The NETCONF server sends a <hello> message encoded with XML and declares the NETCONF capabilities that it is capable of (for example, some NETCONF base capabilities), the YANG data models the device knows, and some other proprietary models, etc.

The NETCONF client replies with its capabilities. Once the capabilities match, the client can start to send a series of RPC requests encoded in XML format and sends it to the server using the secure, connection-oriented session (SSH).

The server responds with an RPC reply encoded in XML. The contents of both the request and the response are fully described in XML schemas, allowing both parties to recognize the syntax constraints imposed on the exchange.

The example given below shows the NETCONF session establishment and capability exchange with Cisco Nexus 9000 NX-OS 9.2.1 loaded with all OpenConfig RPM packages. Please note that in the example below, the capabilities returned from the NETCONF server contain "deviations." The deviation files document the parameters defined in the OpenConfig YANG model that NX-OS is not supporting or partially supporting. The deviation files are published in the GitHub Cisco NX-OS OpenConfig model page at:

<https://github.com/YangModels/yang/tree/master/vendor/cisco/nx/9.2-1>.

```
ssh -s admin@10.2.25.61 -p 830 netconf
User Access Verification
Password:
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
    <capability>urn:ietf:params:netconf:base:1.1</capability>
    <capability>urn:ietf:params:netconf:capability:writable-
running:1.0</capability>
    <capability>urn:ietf:params:netconf:capability:rollback-on-
error:1.0</capability>
    <capability>urn:ietf:params:netconf:capability:candidate:1.0</capability>
    <capability>urn:ietf:params:netconf:capability:validate:1.1</capability>
    <capability>urn:ietf:params:netconf:capability:confirmed-
commit:1.1</capability>
    <capability>http://cisco.com/ns/yang/cisco-nx-os-device?revision=2018-07-
17&module=Cisco-NX-OS-device&deviations=Cisco-NX-OS-device-
deviations</capability>
    <capability>http://openconfig.net/yang/acl?revision=2017-05-
26&module=openconfig-acl&deviations=openconfig-acl-
deviations</capability>
    <capability>http://openconfig.net/yang/bgp-policy?revision=2017-07-
30&module=openconfig-bgp-policy&deviations=openconfig-bgp-policy-
deviations</capability>
    .....
    <snip>
    .....
    <capability>http://openconfig.net/yang/vlan?revision=2017-07-
14&module=openconfig-vlan&deviations=openconfig-vlan-
deviations</capability>
  </capabilities>
  <session-id>511548866</session-id>
</hello>
]]>>><hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.1</capability>
  </capabilities>
</hello>
]]>>>
```

NETCONF data store and operations

Cisco Nexus 9000 NX-OS supports <running>, <startup>, and <candidate> data stores.

The <running> configuration data store holds the complete configuration currently active on the network device.

The <startup> configuration data store is loaded by the device when it boots. Operations that affect the running configuration will not be automatically copied to the startup configuration. An explicit <copy-config> operation from <running> to <startup> is used to update the startup configuration to the current contents of the running configuration.

The <candidate> configuration data store is used to hold configuration data that can be manipulated without affecting the device's running configuration. The changes made to the <candidate> configuration can be committed to the <running> configuration.

The example below shows how to use <candidate> data store to add BGP configurations:

Step1: Initialize the <candidate> data store with the <running> data store.

```
<rpc message-id="461" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <copy-config>
    <target>
      <candidate/>
    </target>
    <source>
      <running/>
    </source>
  </copy-config>
</rpc>
```

Step2: Edit the <candidate> data store without affecting the <running> configuration.

```
<rpc message-id="664" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <error-option>rollback-on-error</error-option>
    <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
        <bgp-items>
          <inst-items xc:operation="replace">
            <asn>100</asn>
            <dom-items>
              <Dom-list>
                <name>blue</name>
              </Dom-list>
              <Dom-list>
                <name>green</name>
                <rtrId>10.20.30.40</rtrId>
              </Dom-list>
            </inst-items>
          </bgp-items>
        </System>
      </config>
    </edit-config>
  </rpc>
```



```
        </Dom-list>
        <Dom-list>
            <name>red</name>
            <rtrId>1.2.3.4</rtrId>
        </Dom-list>
    </dom-items>
</inst-items>
</bgp-items>
</System>
</config>
</edit-config>
</rpc>
```

Step3: Validate the contents of <candidate> configuration. Note the main purpose of using the <candidate> data store is to hold configuration data that can be manipulated without affecting the device's <running> configuration. Then the configuration is verified before being committed to the <running> configuration.

```
<rpc message-id="469" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <validate>
    <source>
      <candidate/>
    </source>
  </validate>
</rpc>
```

Step 4: Commit the <candidate> configuration to the <running> configuration.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="102">
  <commit/>
</rpc>
```

After step 4, the BGP configurations below are enabled:

```
router bgp 100
  vrf blue
  vrf green
    router-id 10.20.30.40
  vrf red
    router-id 1.2.3.4
```

The following NETCONF protocol operations are supported by the Cisco Nexus 9000 NX-OS NETCONF Agent. The operations are performed on the data stores as either source or target data stores.

```
get
get-config
edit-config
close-session
kill-session
lock
unlock
validate
```

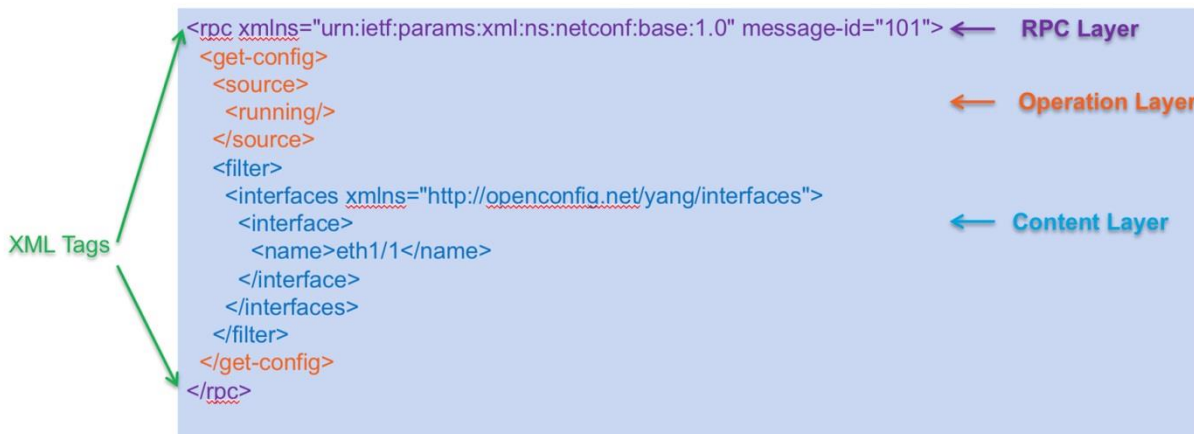
```
commit
cancel-commit
discard-changes
```

Some of the operations are only supported on <candidate> data store, for example, commit, cancel-commit, and discard-changes. The delete-config operation on the data stores is not allowed. For more details on these operations and data stores, please refer to the NETCONF protocol RFC 6241^[3].

Remote-Procedure Call (RPC) contents

After the NETCONF client has established SSH connectivity with a NETCONF server and exchanged capabilities, the client starts to send a series of RPC requests encoded in XML. The server responds with an RPC reply encoded in XML. XML is a markup language similar to HTML; it is designed to encode structured data with self-defined XML tags. Depending on the device's capabilities, the content data elements may be represented by a native YANG model, an OpenConfig YANG model, or some other model.

Below is an example of an RPC payload: an RPC layer, an operation layer, and a content layer represented by an OpenConfig YANG model. This RPC payload is trying to do a "get-config" operation from a <running> configuration data store for interface eth1/1.



Benefits of NETCONF

The main benefit of NETCONF is that it is a standard-based network management protocol. NETCONF is transaction-based management; either all configurations are applied or none are; hence inconsistent states are avoided. And management can be performed at both single- device and network-wide levels. NETCONF also provides rich capabilities, such as locking, confirmed-commit, and manipulate/validate without affecting a device's running configuration by <candidate> data store feature, as explained in previous sections.

RESTCONF Agent introduction

The Cisco Nexus 9000 NX-OS RESTCONF Agent is a REST-like interface running on top of HTTP/HTTPS to access data defined in YANG, using the data store defined in NETCONF. Quoting from RESTCONF standard RFC8040:

"RESTCONF uses HTTP methods to implement the equivalent of NETCONF operations, enabling basic CRUD operations on a hierarchy of conceptual resources.

The HTTP POST, PUT, PATCH, and DELETE methods are used to edit data resources represented by YANG data models. These basic edit operations allow the running configuration to be altered by a RESTCONF client.

RESTCONF is not intended to replace NETCONF, but rather to provide an HTTP interface that follows Representational State Transfer (REST) principles and is compatible with the NETCONF datastore model.”

RESTCONF supports both XML and JSON encoding formats. RESTCONF provides a simplified interface that follows REST-like principles running on top of HTTP/HTTPS transport; making RESTCONF an attractive choice for application developers.

RESTCONF does not support <candidate> data store; it uses the <running> configuration data store only. Hence RESTCONF does not support locking, candidate configuration, and commit features, as described in the previous NETCONF section. Multiphase transactions are not supported with RESTCONF.

For more information on RESTCONF, please refer to the RESTCONF configuration guide at:

https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus9000/sw/9-x/programmability/guide/b_Cisco_Nexus_9000_Series_NX-OS_Programmability_Guide_9x/b_Cisco_Nexus_9000_Series_NX-OS_Programmability_Guide_9x_chapter_011000.html.

gRPC Agent introduction

The Cisco Nexus 9000 NX-OS gRPC Agent is a client-facing interface that uses a Remote Procedure Call (RPC) where an external client manipulates device configurations using Google Protocol Buffer (also Google Protobuf) (GPB)– defined API calls. The RPC reply is defined in the same GPB API context^[5]. For more information on gRPC, please refer to the gRPC configuration guide at:

https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus9000/sw/9-x/programmability/guide/b_Cisco_Nexus_9000_Series_NX-OS_Programmability_Guide_9x/b_Cisco_Nexus_9000_Series_NX-OS_Programmability_Guide_9x_chapter_011001.html.

Start model-driven programmability on Cisco Nexus 9000 Series Switches using OpenConfig YANG models

Step 1: Load the Cisco Nexus 9000 Series Switch with YANG model RPM packages.

In “YANG models supported by Cisco Nexus 9000 NX-OS,” it was stated that Cisco Nexus 9000 NX-OS supports both native YANG models and open YANG models.

For Cisco Nexus 9000 Series Switches, using NX-OS 9.2.1 release as an example, a Cisco native YANG model is included in the NX-OS image and installed automatically when the image is loaded.

The Open YANG models are not included in NX-OS 9.2.1.

Step1a: Download the 9.2.1-release open YANG model RPMs from the Cisco Artifactory directory:

https://devhub.cisco.com/artifactory/open-nxos-agents/9.2-1/x86_64/

As shown in Figure 4, for the 9.2.1 release, there is an all-in-one RPM package that includes all of the supported open YANG models, called “mtx-openconfig-all-1.0.0.0-9.2.1.lib32_n9000.rpm”. There are also individual RPM packages for each supported pen YANG model: for example: “mtx-openconfig-bgp-policy-1.0.0.0-9.2.1.lib32_n9000.rpm” for the BGP-Policy OC YANG model, etc. If you need to use only certain specific open YANG models, you can load only the models you need.

For the sake of simplicity, downloading the all-in-one RPM package onto the Cisco Nexus 9000 Series Switch is recommended.

Figure 4. Index of open-nxos-agents/9.2-1/x86_64

Name	Last modified	Size
../		
repodata/	18-Jul-2018 19:57	-
mtx-openconfig-acl-1.0.0.0-9.2.1.lib32_n9000.rpm	18-Jul-2018 19:55	1.06 MB
mtx-openconfig-acl-1.0.0.0-9.2.1.lib32_n9000.rpm.md5	18-Jul-2018 19:55	32 bytes
mtx-openconfig-acl-1.0.0.0-9.2.1.lib32_n9000.rpm.sha1	18-Jul-2018 19:55	40 bytes
mtx-openconfig-all-1.0.0.0-9.2.1.lib32_n9000.rpm	18-Jul-2018 19:55	25.23 MB
mtx-openconfig-all-1.0.0.0-9.2.1.lib32_n9000.rpm.md5	18-Jul-2018 19:55	32 bytes
mtx-openconfig-all-1.0.0.0-9.2.1.lib32_n9000.rpm.sha1	18-Jul-2018 19:55	40 bytes
mtx-openconfig-bgp-policy-1.0.0.0-9.2.1.lib32_n9000.rpm	18-Jul-2018 19:55	1.12 MB
mtx-openconfig-bgp-policy-1.0.0.0-9.2.1.lib32_n9000.rpm.md5	18-Jul-2018 19:55	32 bytes
mtx-openconfig-bgp-policy-1.0.0.0-9.2.1.lib32_n9000.rpm.sha1	18-Jul-2018 19:55	40 bytes
mtx-openconfig-if-aggregate-1.0.0.0-9.2.1.lib32_n9000.rpm	18-Jul-2018 19:55	843.16 KB
mtx-openconfig-if-aggregate-1.0.0.0-9.2.1.lib32_n9000.rpm.md5	18-Jul-2018 19:55	32 bytes
mtx-openconfig-if-aggregate-1.0.0.0-9.2.1.lib32_n9000.rpm.sha1	18-Jul-2018 19:55	40 bytes
mtx-openconfig-if-ethernet-1.0.0.0-9.2.1.lib32_n9000.rpm	18-Jul-2018 19:55	932.05 KB
mtx-openconfig-if-ethernet-1.0.0.0-9.2.1.lib32_n9000.rpm.md5	18-Jul-2018 19:55	32 bytes
mtx-openconfig-if-ethernet-1.0.0.0-9.2.1.lib32_n9000.rpm.sha1	18-Jul-2018 19:55	40 bytes
mtx-openconfig-if-ip-1.0.0.0-9.2.1.lib32_n9000.rpm	18-Jul-2018 19:55	1.09 MB
mtx-openconfig-if-ip-1.0.0.0-9.2.1.lib32_n9000.rpm.md5	18-Jul-2018 19:55	32 bytes

Step1b: Upload the NX-OS 9.2.1 software and the corresponding all-in-one OpenConfig RPM package to the Cisco Nexus 9000 Series Switch bootflash, then verify that the files are on bootflash.

```
N9KFX1# dir
      26455883   Jul 20 20:56:00 2018  mtx-openconfig-all-1.0.0.0-
      9.2.1.lib32_n9000.rpm
      1308795904   Jul 20 20:50:51 2018  nxos.9.2.1.bin
```

Step1c: Upgrade Nexus 9000 with NX-OS 9.2.1. After bootup of the switch, the Agents (gRPC, NETCONF, RESTCONF) and mtx-device, mtx-infra, and mtx-telemetry RPM packages are automatically installed.

You can verify those automatically installed RPM packages from the bash shell. Make sure **feature bash** is configured on the device in order to use bash shell.

```
N9KFX1# run bash sudo su
bash-4.3#
bash-4.3# yum list installed | grep mtx
mtx-device.lib32_n9000                2.0.0.0-9.2.1                installed
mtx-grpc-agent.lib32_n9000            2.0.0.0-9.2.1                installed
mtx-infra.lib32_n9000                 2.0.0.0-9.2.1                installed
mtx-netconf-agent.lib32_n9000         2.0.0.0-9.2.1                installed
mtx-restconf-agent.lib32_n9000        2.0.0.0-9.2.1                installed
```

```
mtx-telemetry.lib32_n9000          2.0.0.0-9.2.1          installed
bash-4.3#
```

Step1d: Install the all-in-one OpenConfig RPM package on the switch from the bash shell:

```
N9KFX1# run bash sudo su
bash-4.3#
bash-4.3# cd /bootflash
bash-4.3# yum install mtx-openconfig-all-1.0.0.0-9.2.1.lib32_n9000.rpm
Loaded plugins: downloadonly, importpubkey, localrpmDB, patchaction, patching,
                : protect-packages
groups-repo                       | 1.1 kB    00:00 ...
localdb                            | 951 B    00:00 ...
patching                           | 951 B    00:00 ...
thirdparty                        | 951 B    00:00 ...
Setting up Install Process
Examining mtx-openconfig-all-1.0.0.0-9.2.1.lib32_n9000.rpm: mtx-openconfig-all-
1.0.0.0-9.2.1.lib32_n9000
Marking mtx-openconfig-all-1.0.0.0-9.2.1.lib32_n9000.rpm to be installed
Resolving Dependencies
--> Running transaction check
---> Package mtx-openconfig-all.lib32_n9000 0:1.0.0.0-9.2.1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                               Arch      Version Repository                               Size
=====
Installing:
  mtx-openconfig-all
    lib32_n9000 1.0.0.0-9.2.1
                                     /mtx-openconfig-all-1.0.0.0-9.2.1.lib32_n9000 88 M

Transaction Summary
=====
Install      1 Package

Total size: 88 M
Installed size: 88 M
Is this ok [y/N]:
```

Step1e: Verify that the OpenConfig all-in-one RPM is installed successfully.

```
bash-4.3# yum list installed | grep mtx
mtx-device.lib32_n9000                2.0.0.0-9.2.1      installed
mtx-grpc-agent.lib32_n9000            2.0.0.0-9.2.1      installed
mtx-infra.lib32_n9000                 2.0.0.0-9.2.1      installed
mtx-netconf-agent.lib32_n9000         2.0.0.0-9.2.1      installed
mtx-openconfig-all.lib32_n9000      1.0.0.0-9.2.1      @/mtx-
openconfig-all-1.0.0.0-9.2.1.lib32_n9000
mtx-restconf-agent.lib32_n9000        2.0.0.0-9.2.1      installed
mtx-telemetry.lib32_n9000            2.0.0.0-9.2.1      installed
bash-4.3#
```

Step1f: Exit from the bash shell, then enable NETCONF Agent in CLI mode and save the configuration.

```
conf t
feature netconf
end
```

Now the Cisco Nexus 9000 Series Switch is enabled with the NETCONF Agent and the OpenConfig YANG model, and is ready to receive the YANG model client request from the controller or applications.

Step 2: Prepare YANG tools

There are many open-source and commercial YANG tools in the market to help you visualize YANG models and generate NETCONF RPC payloads. A good tool is very useful for a beginner to start the model-driven programmability journey. One example of an open-source YANG tool is Yang Explorer^[6], which is available at GitHub: <https://github.com/CiscoDevNet/yang-explorer/wiki>

Yang Explorer is an open-source web application that provides a web-based user interface to:

- Browse YANG data models
- Create NETCONF RPC payloads
- Execute NETCONF RPCs
- Save RPCs to collections

The section below shows the basics of Yang Explorer. This will help you understand the structure of YANG models and get started with Model-Driven Programmability. Please be aware that Yang Explorer is an open-source tool and so may not work coherently with Cisco Nexus 9000 software. Please use it at your own discretion.

Cisco has developed internal YANG tools that have greater functionality than Yang Explorer. To try out Cisco's internal YANG tool, please contact the Cisco sales team to get more information.

Step 2a. Install and start Yang Explorer.

You can install and start Yang Explorer following the Wiki page instructions:

<https://github.com/CiscoDevNet/yang-explorer/wiki>

Step 2b. Login to Yang Explorer.

Yang Explorer installed on a MAC laptop is used as an example to illustrate how to use Yang Explorer. You can also refer to the YANG user guide at: <https://github.com/CiscoDevNet/yang-explorer/wiki/User-Guide>

Log in to the Yang Explorer GUI at <https://localhost:8088/> using username: guest, password: guest.

Step 2c. Create a device profile, and load the device profile.

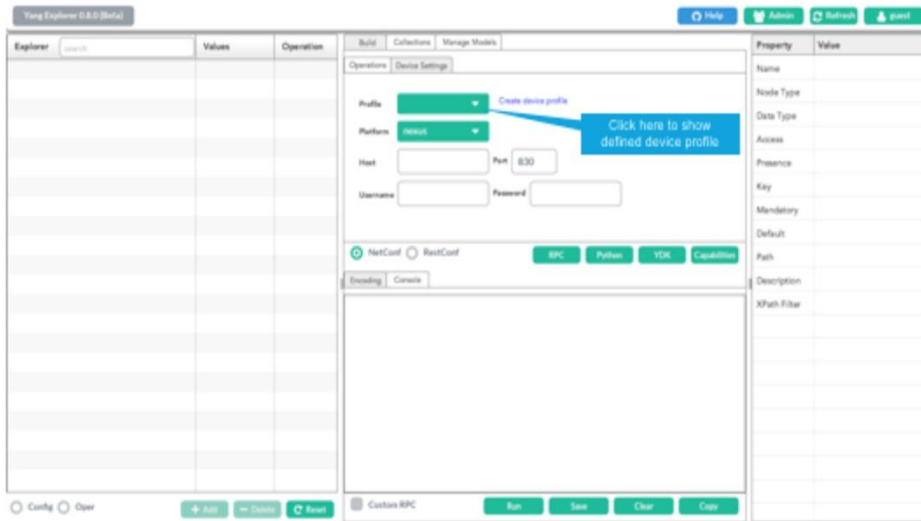
First create a device profile, and fill in the corresponding IP address, username, password, etc. This is the switch that Yang Explorer will access to execute NETCONF RPC calls.

Home > Explorer > Device profiles > N9KEX2

Change device profile

Profile:	N9KEX2
Device:	nexus
User:	guest
NetConf IP:	172.17.0.1 Optional NetConf IP
NetConf Port:	830 Optional NetConf Port
NetConf Username:	admin Optional NetConf Username
NetConf Password:	XXXXXXXXXX Optional NetConf Password
RestConf IP:	172.17.0.1 Optional RestConf IP
RestConf Port:	8008 Optional RestConf Port
RestConf Username:	admin Optional RestConf Username
RestConf Password:	XXXXXXXXXX Optional RestConf Password
Description:	N9K with Yang Explorer Test
<input type="checkbox"/> Shared Device ?	

Then click the Profile dropdown list; the device profile you defined will show up. Select that device profile; the Host, Port, Username, and Password will be automatically loaded with the device profile information.



Step 2d. Browse YANG Data models.

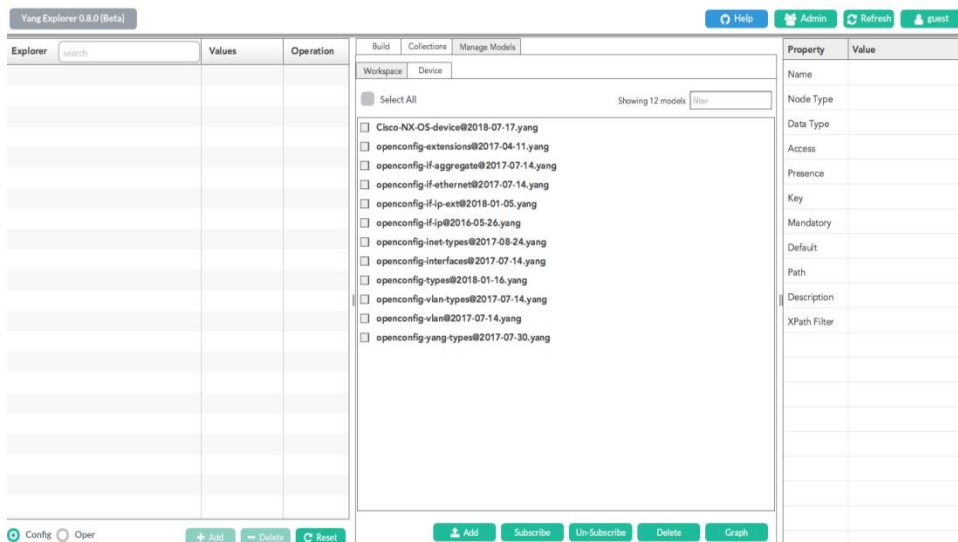
To browse YANG Data Models in Yang Explorer, you need to upload the supported YANG Data Models in Yang Explorer. You can find the supported native and open YANG models for NX-OS 9.2.1 release at:

<https://github.com/YangModels/yang/tree/master/vendor/cisco/nx/9.2-1>

From this vendor-specific YANG model directory, you can open each model file to find out the revision number. Please note that the revision may not be the latest revision defined by the OpenConfig Consortium. Also, from the YANG model directory, there are some deviation files, for example, "cisco-nx-openconfig-acl-deviations.yang." The deviation files document the nonsupported YANG model nodes with the corresponding NX-OS release. The deviation files document the parameters defined in the OpenConfig YANG model that NX-OS does not support or partially supports.

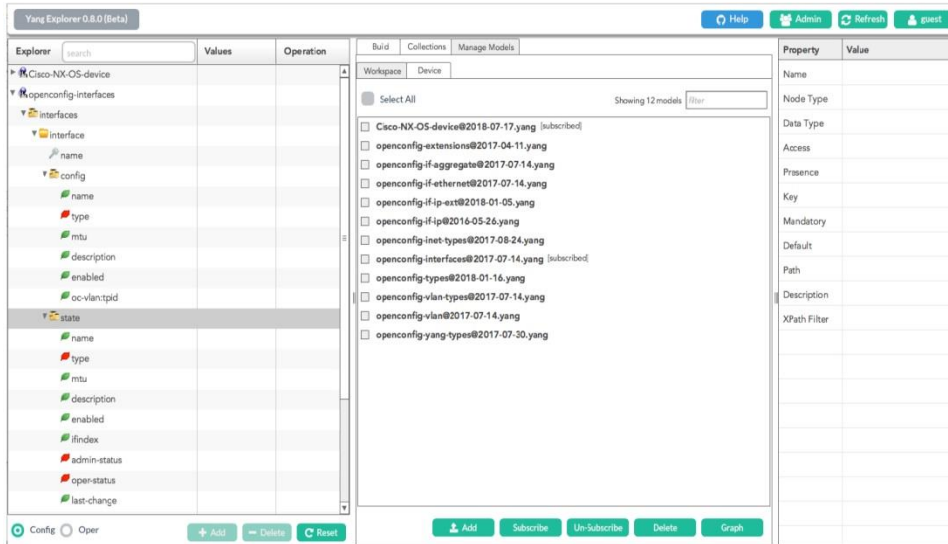
After downloading the models, you need to upload the models in Yang Explorer, as shown in Figure 5.

Figure 5. Uploading the models in Yang Explorer



Click the “Subscribe” button, so the model structures are displayed in the left panel, as shown in Figure 6.

Figure 6. Display of model structures in left panel

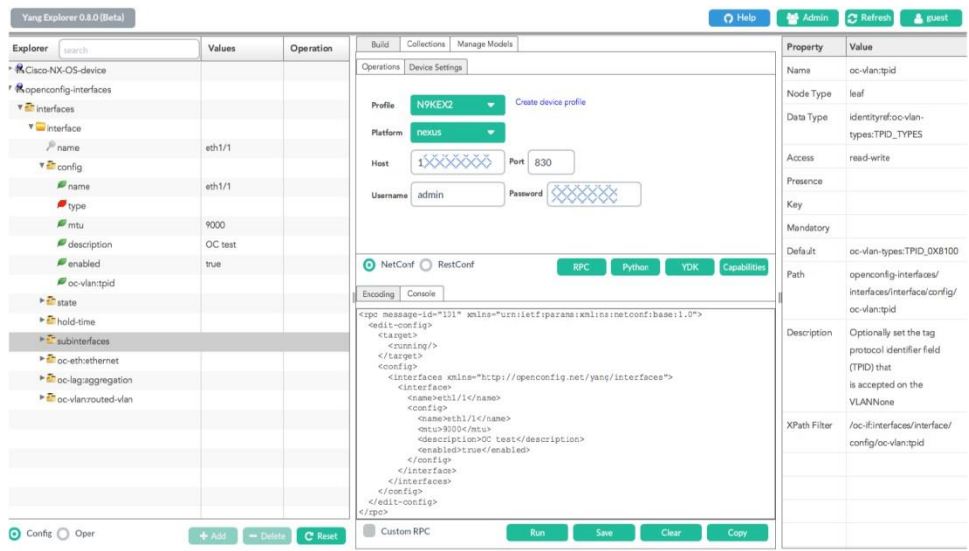


By going through the model structure, it is much easier to understand the YANG models than by going through the original YANG definition files.

Step 2e. Generate/execute RPC payloads

You can also use Yang Explorer to generate and execute NETCONF RPC payloads. Figure 7 shows an example of modifying the interface configuration with an open YANG model, generating the corresponding NETCONF RPC payload, and sending the structured payload to the Cisco Nexus 9000 Series Switch to modify the configuration. For example, in Figure 6, in the left-hand “Explorer/Values/Operation” panel, to generate an RPC payload for a configuration, you need to click the “Config” radio button at the bottom of the left-hand panel. The “Explorer” Column lists the model structure, then you fill in the parameters in the “Values” section. In this example, we’d like to configure interface eth1/1 with an MTU value and description, and enable the interface, in the “Values” Column; it either allows you to fill in parameters, such as “eth1/1” or gives you a drop-down menu for a selection; for example, an enabled state can only be “true” or “false.” Once the corresponding “Values” are specified, in the middle-bottom panel, select the “NetConf” radio button, and then click “RPC” to generate the RPC payload.

Figure 7. Modifying the interface configuration with an open YANG model



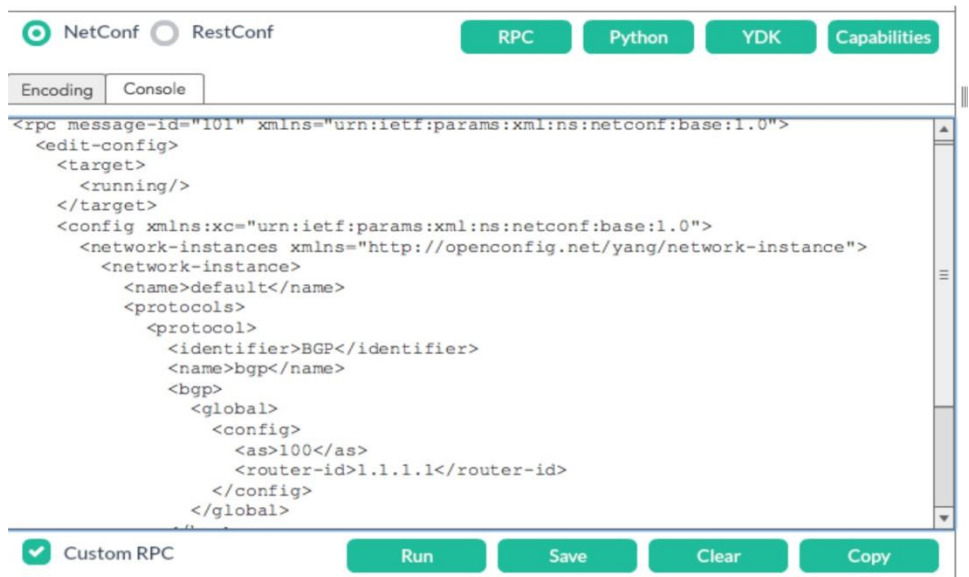
For the Cisco Nexus 9000 NX-OS release, there are many OpenConfig NETCONF payload examples documented at:

<https://developer.cisco.com/site/cisco-nexus-nx-api-references/>

Click the “Cisco Nexus OpenConfig YANG, Release 9.x” on that page to find Nexus OpenConfig YANG NETCONF payload examples.

You can take these payload examples, modify them, and send to Cisco Nexus 9000 Series Switch via the “Custom RPC” show in Figure 8. It is much easier to modify the payload examples than generate the payloads from scratch using tools.

Figure 8. Paste your payloads into the “Custom RPC” window



Step 3: Start model-driven programmability using OpenConfig YANG model

Once you get familiar with YANG models, you can start your automation journey by manipulating the structured input and output data with different programming languages. For example:

Below is an example NETCONF RPC call to get information for interface ethernet1/1: type, mtu, name, description, and enable state.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <get-config>
    <source>
      <running/>
    </source>
    <filter>
      <interfaces xmlns="http://openconfig.net/yang/interfaces">
        <interface>
          <name>eth1/1</name>
          <config>
            <type/>
            <mtu/>
            <name/>
            <description/>
            <enabled/>
          </config>
        </interface>
      </interfaces>
    </filter>
  </get-config>
</rpc>
```

After receiving the RPC call, the NETCONF Agent sends back an RPC response with all the information requested for interface ethernet1/1. As you can see, the returned data for interface ethernet1/1 is in structured XML format, which is easier to parse than the unstructured CLI output.

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:f316ead1-ea60-4b28-9259-3ce7830f69a9">
  <data>
    <interfaces xmlns="http://openconfig.net/yang/interfaces">
      <interface>
        <name>eth1/1</name>
        <config>
          <enabled>>false</enabled>
          <mtu>1500</mtu>
          <name>eth1/1</name>
          <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type>
        </config>
        <hold-time>
          <config>
```

```

        <down>100</down>
    </config>
</hold-time>
<subinterfaces>
    <subinterface>
        <config>
            <index>1</index>
            <enabled>>false</enabled>
        </config>
        <index>1</index>
        <ipv4 xmlns="http://openconfig.net/yang/interfaces/ip">
            <addresses>
                <address>
                    <config>
                        <ip>10.2.1.1</ip>
                        <prefix-length>24</prefix-length>
                    </config>
                    <ip>10.2.1.1</ip>
                </address>
            </addresses>
        </ipv4>
    </subinterface>
    <subinterface>
        <config>
            <index>0</index>
        </config>
        <index>0</index>
        <ipv4 xmlns="http://openconfig.net/yang/interfaces/ip">
            <addresses>
                <address>
                    <config>
                        <ip>10.1.1.1</ip>
                        <prefix-length>24</prefix-length>
                    </config>
                    <ip>10.1.1.1</ip>
                </address>
            </addresses>
        </ipv4>
    </subinterface>
</subinterfaces>
<ethernet xmlns="http://openconfig.net/yang/interfaces/ethernet">
    <config>
        <auto-negotiate>>true</auto-negotiate>
    </config>
</ethernet>

```

```
        </interface>
    </interfaces>
</data>
</rpc-reply>
```

Start model-driven programmability on a Cisco Nexus 9000 Series Switch using native YANG models

In “YANG Models Supported by Cisco Nexus 9000 NX-OS,” it was stated that Cisco Nexus 9000 NX-OS supports both the native and OpenConfig YANG models.

Using native YANG models is much easier than using OpenConfig YANG models. Using NX-OS 9.2.1 release as an example, first, a Cisco native YANG model is included in the Cisco Nexus 9000 NX-OS image and installed automatically when the image is loaded. OpenConfig YANG model RPM packages need to be installed separately, as detailed in a previous chapter. Second, to generate a native YANG model payload, NX-OS provides a tool called NX-API Developer Sandbox, which is a web form hosted on the switch. Sandbox can translate NX-OS CLI commands directly into the native YANG model payload, which is much easier to use.

Sandbox usage example 1: Sandbox and NETCONF

Example 1 shows you how to get a native YANG model payload using Sandbox and how to generate a NETCONF RPC payload.

For more details of how to use Sandbox, please refer to the configuration guide at:

https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus9000/sw/9-x/programmability/guide/b_Cisco_Nexus_9000_Series_NX-OS_Programmability_Guide_9x/b_Cisco_Nexus_9000_Series_NX-OS_Programmability_Guide_9x_chapter_010100.html

Step 1: Prepare a Nexus 9000 switch loaded with NX-OS 9.2.1 software.

Enable feature nxapi and feature netconf using the following commands:

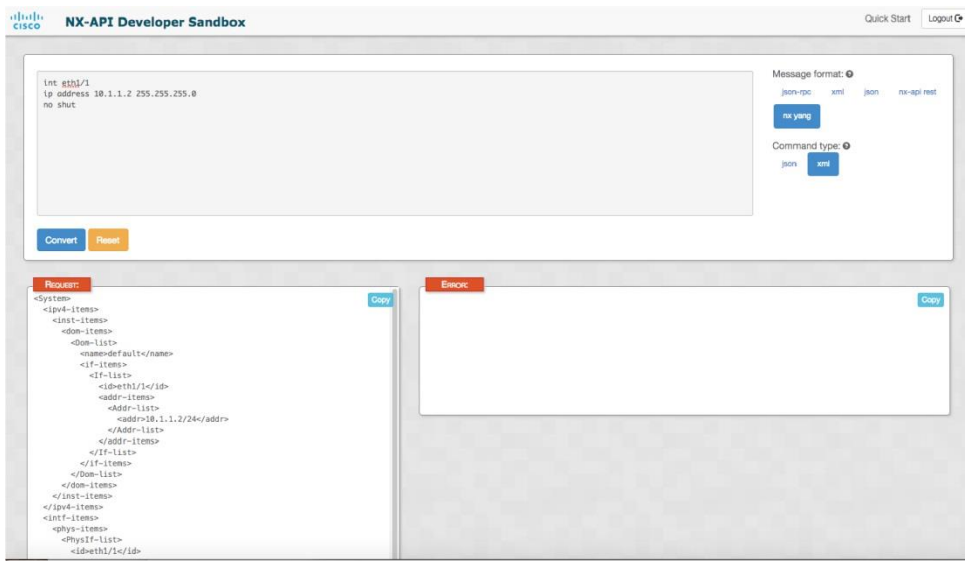
```
switch# configure terminal
switch(config)# feature nxapi
switch(config)# feature netconf
```

Step 2: Open a browser and enter the following URL: <https://management-ip-address>, to launch the NX-API Developer Sandbox.

Step 3: Generate a native YANG model payload using Sandbox.

As show in Figure 9, in the Sandbox command pane enter the corresponding CLI commands; choose the message format with “nx-yang”; then click the “Convert” button to generate the corresponding native YANG model payload.

Figure 9. Generating a native YANG model payload using Sandbox



In the example in Figure 8, the CLI configuration is:

```
int eth1/1
ip address 10.1.1.2 255.255.255.0
no shut
```

The generated native YANG model payload is:

```
<System>
  <ipv4-items>
    <inst-items>
      <dom-items>
        <Dom-list>
          <name>default</name>
          <if-items>
            <If-list>
              <id>eth1/1</id>
              <addr-items>
                <Addr-list>
                  <addr>10.1.1.2/24</addr>
                </Addr-list>
              </addr-items>
            </If-list>
          </if-items>
        </Dom-list>
      </dom-items>
    </inst-items>
  </ipv4-items>
  <intf-items>
    <phys-items>
      <phys-if-list>
        <id>eth1/1</id>
```

```

    <PhysIf-list>
      <id>eth1/1</id>
      <adminSt>up</adminSt>
      <userCfgdFlags>admin_state</userCfgdFlags>
    </PhysIf-list>
  </phys-items>
</intf-items>
</System>

```

Step 4: Add the corresponding NETCONF command to generate a NETCONF RPC payload with the native YANG model payload, and use NETCONF to push the configuration to the device.

```

<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
        <ipv4-items>
          <inst-items>
            <dom-items>
              <Dom-list>
                <name>default</name>
                <if-items>
                  <If-list>
                    <id>eth1/1</id>
                    <addr-items>
                      <Addr-list>
                        <addr>10.1.1.2/24</addr>
                      </Addr-list>
                    </addr-items>
                  </If-list>
                </if-items>
              </Dom-list>
            </dom-items>
          </inst-items>
        </ipv4-items>
        <intf-items>
          <phys-items>
            <PhysIf-list>
              <id>eth1/1</id>
              <adminSt>up</adminSt>
              <userCfgdFlags>admin_state</userCfgdFlags>
            </PhysIf-list>
          </phys-items>
        </System>
      </config>
    </edit-config>
  </rpc>

```

```
</intf-items>
</System>
</config>
</edit-config>
</rpc>
```

Sandbox usage example 2: Sandbox and RESTCONF

From NX-OS release 9.2.2, Sandbox will support RESTCONF with a native YANG model. A RESTCONF request is sent directly from the Sandbox GUI to the switch. It is very convenient to configure the Cisco Nexus 9000 Series Switch using RESTCONF directly from Sandbox, as shown in the steps below:

Step 1: Prepare a Cisco Nexus 9000 Series Switch loaded with NX-OS 9.2.2 software.

Enable feature nxapi and feature restconf using the following commands:

```
switch# configure terminal
switch(config)# feature nxapi
switch(config)# feature restconf
```

Step 2: Open a browser and enter the following URL: <https://management-ip-address>, to launch the NX-API Developer Sandbox.

Step 3: Generate a native YANG RESTCONF payload using Sandbox.

From Sandbox, on the right-hand side of the Sandbox GUI (shown in Figure 10), select “RESTCONF (Yang)” for “Method.” The message format can be in “xml” or “json.”

Type in the CLI command in the command window, that is:

```
int eth1/1
ip address 10.1.1.11/24
no shut
```

In the middle section of the Sandbox GUI, click the “Convert” button to convert the CLI to the native YANG model payload.

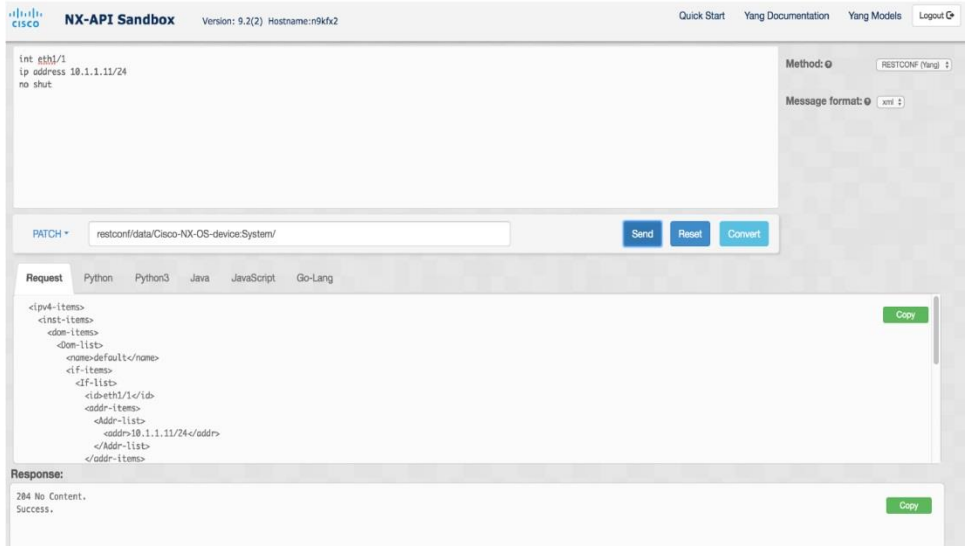
Then in the middle section of the Sandbox GUI, select RESTCONF “PATCH” operation, click “Send” button to send the native YANG payload via RESTCONF “PATCH” operation.

A RESTCONF “PATCH” operation is equivalent to a NETCONF “<edit-config> (operation=“merge”)” operation.

Please refer to the RESTCONF configuration guide for more RESTCONF operations:

https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus9000/sw/9-x/programmability/guide/b_Cisco_Nexus_9000_Series_NX-OS_Programmability_Guide_9x/b_Cisco_Nexus_9000_Series_NX-OS_Programmability_Guide_9x_chapter_011001.html

Figure 10. Generating a native YANG RESTCONF payload using Sandbox



The following is the corresponding native YANG payload:

```

<ipv4-items>
  <inst-items>
    <dom-items>
      <Dom-list>
        <name>default</name>
        <if-items>
          <If-list>
            <id>eth1/1</id>
            <addr-items>
              <Addr-list>
                <addr>10.1.1.11/24</addr>
              </Addr-list>
            </addr-items>
          </If-list>
        </if-items>
      </Dom-list>
    </dom-items>
  </inst-items>
</ipv4-items>
<intf-items>
  <phys-items>
    <PhysIf-list>
      <id>eth1/1</id>
      <adminSt>up</adminSt>
      <userCfgdFlags>admin_state</userCfgdFlags>
    </PhysIf-list>
  
```

```
</phys-items>
</intf-items>
```

Customer use-case example with a YANG model

In “YANG Models Supported by Cisco Nexus 9000 NX-OS,” it stated that Cisco Nexus 9000 NX-OS supports both open and native YANG models. There are pros and cons for each model. An open YANG model is defined by a standards body in a vendor-neutral way; it can be used to manage and monitor different vendors’ equipment, but is less flexible because different vendors have to agree on the models, and it doesn’t support vendor-specific features. A native YANG model is defined by Cisco, using standard YANG data-modeling language and is specific to NX-OS, but supports many more features than the OpenConfig YANG model. The Cisco native YANG model is also fully integrated with Cisco Network Services Orchestrator (NSO). More detailed information on NSO is available at Cisco.com. Depending on each customer’s requirements, they may choose to use either a Cisco native YANG model or an OpenConfig YANG model.

Below are two customer examples that utilize model-driven programmability with a YANG model.

The first customer has multivendor equipment in their network and are rapidly expanding their data center fabric environment. They use a Layer-3 IP fabric enabled mainly with Border Gateway Protocol (BGP). They prefer to use the OpenConfig YANG model to bring up the new fabric for different vendors using the same automation script; for example, a script based on the OpenConfig BGP YANG model. For Cisco Nexus 9000 Series Switches, there are many OpenConfig BGP NETCONF payload examples documented at: <https://developer.cisco.com/site/cisco-nexus-nx-api-references/>, under section of “Cisco Nexus OpenConfig YANG, Release 9.x”

These BGP payload examples can be used to construct a complete BGP payload based on the customer’s BGP configuration.

The second customer prefer to use the Cisco native YANG model so they can automate their network day-1 configuration and day-2 monitoring with a more comprehensive feature set. For this customer, NX-API Developer Sandbox is very helpful to generate a native YANG payload, then use NETCONF or RESTCONF to send the payload to the device.

In summary, Cisco Nexus 9000 Series NX-OS gives you flexible choices for a YANG model. Depending on your network environment and preference, you can choose to use either a native or an open YANG model to start your automation journey.

Advanced tools for model-driven programmability on Cisco Nexus 9000 Series Switches using YANG models

There are many advanced open-source tools to automate management on Cisco Nexus devices. Below are three open-source tools you can use to automate your network in a programmatic manner:

- **NCCLIENT** is a Python library that facilitates client-side scripting and application development around the NETCONF protocol.
- **PYANG** is a YANG validator and code generator, written in Python.
- **YDK** is an open-source tool, developed by Cisco, to facilitate network programmability using data models. YDK takes YANG model definitions as input and generates Python or C++ APIs. It can take any YANG models as long as they are valid. YDK has built-in YANG data validation. Benefits of YDK are as follows:
 - Automation script is built by calling APIs generated by YDK.

- There is no manual NETCONF transport manipulations; for example, RPC calls, edit-config, error handling, etc., are all automated in the API.
- There are no manual YANG XML payload manipulations and no need to generate a NETCONF XML payload; there is only a need to pass in the API parameters.

Summary

Cisco is a leader in network automation and programmability. The rich model-driven programmability of the Cisco Nexus 9000 Series Switches enables you to automate the configuration and control of the devices at scale in an agile way. With a basic knowledge of YANG models, Network Management Protocols such as NETCONF and RESTCONF, and examples of various tools to visualize Cisco native YANG models and OpenConfig YANG models supported by Cisco Nexus 9000 Series Switches, you'll be able to inaugurate model-driven programming and accelerate your automation journey.

References

[1] OpenConfig, <https://www.openconfig.net/>

[2] Cisco Nexus 9000 Series NX-OS Programmability Guide, Release 9.x

https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus9000/sw/9-x/programmability/guide/b_Cisco_Nexus_9000_Series_NX-OS_Programmability_Guide_9x/b_Cisco_Nexus_9000_Series_NX-OS_Programmability_Guide_9x_chapter_010111.html

[3] R. Enns, Ed., "Network Configuration Protocol (NETCONF)" RFC6241

[4] M. Bjorklund, Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)" RFC6020

[5] CiscoNXOS gRPC Protocol Specification

[6] Yang Explorer, <https://github.com/CiscoDevNet/yang-explorer/blob/master/README.md>

[7] RESTCONF Protocol draft-ietf-netconf-restconf-10, <https://tools.ietf.org/html/draft-ietf-netconf-restconf-10>

[8] M. Bjorklund, Ed., "YANG Central Wiki", <https://www.yang-central.org/twiki/bin/view/Main/WebHome>




Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at <https://www.cisco.com/go/offices>.

 Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: <https://www.cisco.com/go/trademarks>. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1110R)