Ansible network Automation using YANG and NETCONF

Ganesh B. Nalawade
Principal Software Engineer
Github/IRC: ganeshrn
Twitter: @ganesh634

Rohit Thakur
Senior Software Engineer
Github: rohitthakur2590
Twitter: @iamrothakur

AnsibleFest

About me: Ganesh

- Works as Principal Software Engineer with Ansible Content Engineering team.
- Software developer with 13+ years of experience
- Worked extensively on network management plane to enhance automation and programmability

About me: Rohit

- Works as Senior Software Engineer with Ansible Network Engineering team.
- Interested in developing software for network management and automation use-case.
- Co-organiser for Ansible Bangalore meetup.

Agenda

What we will be talking about?

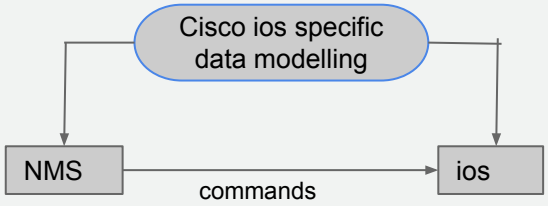
- What is YANG?
- Netconf introduction
- YANG variants
- YANG collection deep dive
- Demo
- Questions

What is YANG?

Vendor specific modelling language

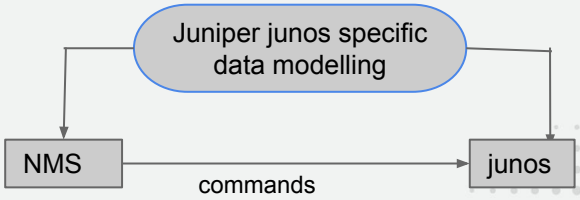
ios interface configuration

```
interface GigabitEthernet0/3
 description test-interface
 ip address 192.168.56.13 255.255.255.0
 shutdown
```



junos interface configuration

```
ge-0/0/2 {
    description test-interface;
    disable;
    unit 0 {
        family inet {
            address 192.168.56.14/24;
        }
    }
}
```

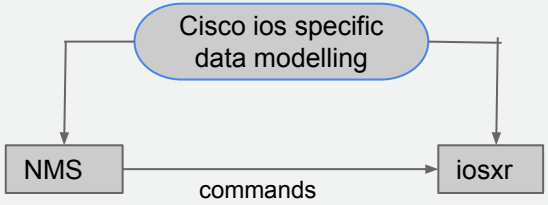


What is YANG?

Vendor specific modelling language

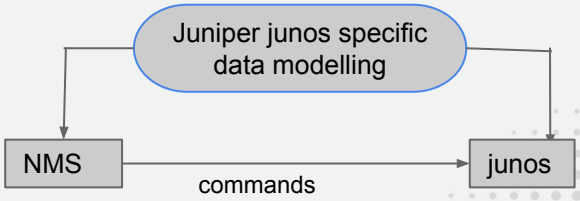
ios interface configuration

```
interface GigabitEthernet0/3
 description test-interface
 ip address 192.168.56.13 255.255.255.0
 shutdown
```



junos interface configuration

```
ge-0/0/2 {
    description test-interface;
    disable;
    unit 0 {
        family inet {
            address 192.168.56.14/24;
        }
    }
}
```



NMS: Network management system

What is YANG?

Yang IETF standard RFC

- YANG is a data modeling language used to model configuration and state data
- It is a IETF standard defined by RFC 6020
- Human readable representation of data-mode
- Hierarchy data representation
- Build in data types and constraints
- Extensible

What is YANG? (YANG Example)

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

    container system {
        leaf host-name {
            type string;
            description "Hostname for this system";
        }

        leaf-list domain-search {
            type string;
            description "List of domain names to search";
        }

        container login {
            leaf message {
                type string;
                description
                    "Message given at start of login session";
            }

            list user {
                key "name";
                leaf name {
                    type string;
                }
                leaf full-name {
                    type string;
                }
                leaf class {
                    type string;
                }
            }
        }
    }
}
```

NETCONF

Netconf IETF standard

- The NETCONF protocol defines a simple mechanism through which a network device can be managed, configuration data information can be retrieved, and new configuration data can be uploaded and manipulated
- It is a IETF standard defined by RFC 6241
- It is xml based encoding mainly build on top of ssh transport as a subsystem
- Configuration rpc's
  - edit-config, get-config, copy-config, delete-config, lock, unlock
- Operational state rpc's
  - get (maps to show commands)

NETCONF

Ansible Netconf modules

- **ansible.netcommon.netconf_config:**

  Configuration management (create, update, delete)
- **ansible.netcommon.netconf_get:**

  netconf_get: Retrieve state and configuration data (read)
- **ansible.netcommon.netconf_rpc:**

  Execute generic Netconf rpc (mainly vendor specific rpc's)

NETCONF

YANG to NETCONF mapping

```
module openconfig-interfaces {

  container interfaces {

    list interface {
        key "name"
        description "The list of configurre interfaces";

        container config {
            leaf name {
                type string;
                description "Name of interface";
            }
            leaf description {
                type string;
                description "Description of interface"
            }
            leaf enabled {
                type boolean;
                default "true";
            }
        }
    }
  }
}
```

```
<rpc>
  <edit-config>
    <target>
        <running/>
    </target>
    <interfaces xmlns="http://openconfig.net/yang/interfaces">
      <interface>
        <name>ge-0/0/2</name>
        <config>
            <name>ge-0/0/2</name>
            <description>test-interface</description>
            <enabled>false</enabled>
        </config>
      </interface>
    </interfaces>
  </edit-config>
</rpc>
```

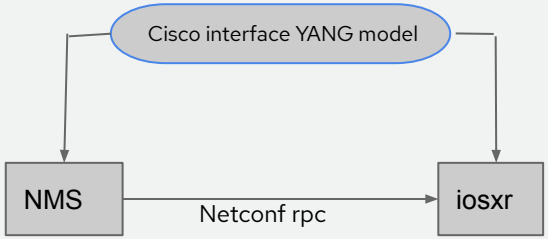
Yang variants

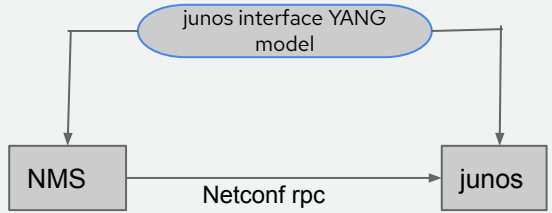
Vendor specific

- Vendor defined
  - The data represented in yang model varies based on vendor implementation and the released yang models are published and maintained by vendors themselves.
  - https://github.com/YangModels/yang/tree/master/vendor/

Yang variants

Vendor specific






```
<Configuration>
  <InterfaceConfigurationTable>
    <InterfaceConfiguration>
      <Naming>
        <Description>test-interface</Description>
        <InterfaceName>GigabitEthernet0/3</InterfaceName>
      </Naming>
      <Shutdown/>
    </InterfaceConfiguration>
  </InterfaceConfigurationTable>
</Configuration>
```

```
<configuration>
    <interfaces>
        <interface>
            <name>ge-0/0/2</name>
            <description>test-interface</description>
            <disable/>
            <unit>
                <name>0</name>
                <family>
                    <inet>
                        <address>
                            <name>192.168.56.14/24</name>
                        </address>
                    </inet>
                </family>
            </unit>
        </interface>
    </interfaces>
</configuration>
```

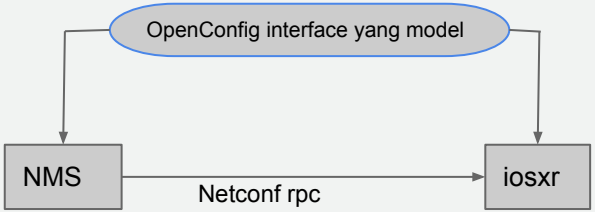
Yang variants

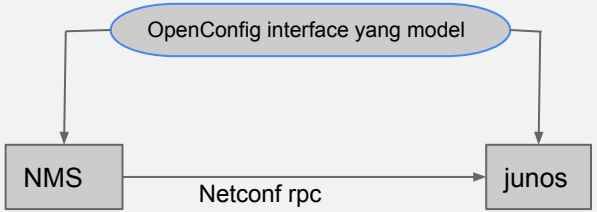
Vendor neutral

- Vendor neutral
  - Standard yang model defined by ietf, ieee etc https://github.com/YangModels/yang/tree/master/standard/ietf/RFC
  - OpenConfig which is an informal working group of network operators that promotes vendor-neutral model
  - https://github.com/openconfig/public/tree/master/release/models.

Yang variants

Vendor neutral






```
<interfaces xmlns="http://openconfig.net/yang/interfaces">
  <interface>
    <name>GigabitEthernet0/3</name>
    <config>
        <name>GigabitEthernet0/3</name>
        <description>test-interface</description>
    </config>
  </interface>
</interfaces>
```

```
<interfaces xmlns="http://openconfig.net/yang/interfaces">
  <interface>
    <name>ge-0/0/2</name>
    <config>
        <name>ge-0/0/2</name>
        <description>test-interface</description>
    </config>
  </interface>
</interfaces>
```

Yang collection deep-dive

Goals

- Should work with all variant of YANG models
- Abstract YANG related complextixies using Ansible modules
- Easy to use with structured data in JSON format as input/output
- Ability to fetch YANG models from network appliance (if supported)
- Render skeleton structured data (JSON, XML, Yang tree) for given YANG model
- Community supported collection
  https://github.com/ansible-collections/community.yang

Yang collection deep-dive

Fetch Yang models function

- **community.yang.fetch:**

  Fetch yang model and its dependent yang models from device using netconf get schemas capability if supported and optionally store the yang files locally on disk.

Yang collection deep-dive

Ansible fetch task example

```
- name: Fetch list of supported yang model names
  community.yang.fetch:
```

```
- name: Fetch given yang model and it's dependent models from remote host
  community.yang.fetch:
    name: openconfig-interface
```

```
- name: Fetch all the yang models supported by remote host and store it in dir location
  community.yang.fetch:
    name: all
    dir: "{{playbook_dir}}/{{inventory_hostname}}/{{ansible_net_version}}/yang_files"
```

Yang collection deep-dive

Get configuration function

- **community.yang.get:**

  The module uses netconf get RPC to fetch configuration data from remote host and render it in JSON format (as per RFC 7951 which defines JSON encoding of data modeled with YANG)

Yang collection deep-dive

Ansible get task example

```
- name: fetch interface configuration and return it in JSON format
  community.yang.get:
    filter: |
        <interface-configurations
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ifmgr-cfg"><interface-configuration>
        </interface-configuration></interface-configurations>
    file: "{{ playbook_dir }}/YangModels/yang/tree/master/vendor/cisco/xr/613/*.yang"
    search_path: "{{ playbook_dir }}/YangModels/yang:{{ playbook_dir }}/pyang/modules"
  register: result
```

Yang collection deep-dive

Ansible get task example

```
"json_data": {
    "Cisco-IOS-XR-ifmgr-cfg:interface-configurations": {
        "interface-configuration": [
            {
                "active": "act",
                "description": "configured by yang collection",
                "interface-name": "GigabitEthernet0/0/0/0"
            },
        ]
    }
}
```

Yang collection deep-dive

configure function

- **community.yang.configure:**
  - The module takes the JSON configuration as input (as per RFC 7951 which defines JSON encoding of data modeled with YANG).
  - Pre-validates the config with the corresponding YANG model.
  - Converts input JSON configuration to XML payload to be pushed on the remote host using netconf connection.

Yang collection deep-dive

Ansible configure task example

```
- name: configure by reading JSON config data from file
  community.yang.configure:
    config: "{{ lookup('file', 'interfaces-config.json') | to_json }}"
    file: "{{ playbook_dir }}/public/release/models/interfaces/openconfig-interfaces.yang"
    search_path: "{{ playbook_dir }}/public/release/models"
  register: result
```

Yang collection deep-dive

generate_spec function

- **community.yang.generate_spec:**

The module will be read the given Yang model and generate the corresponding JSON, XML schema and the YANG tree representation (as per RFC 8340) of the model.

Yang collection deep-dive

Ansible generate_spec task example

```
- name: set spec dir path location
  set_fact:
    spec_dir_path: "{{ playbook_dir }}/{{ inventory_hostname }}/{{ ansible_net_version }}"

- name: generate spec from openconfig interface config data and store it in file
   community.yang.generate_spec:
     file: "{{ playbook_dir
}}/openconfig/public/release/models/interfaces/openconfig-interfaces.yang"
     search_path: "{{ playbook_dir }}/openconfig/public/release/models:pyang/modules"
     json_schema:
       path: "{{ spec_dir_path }}/spec/openconfig-interfaces-config.json"
     xml_schema:
       path: "{{ spec_dir_path }}/openconfig-interfaces-config.xml"
       defaults: True
     tree_schema:
       path: "{{ spec_dir_path }}/openconfig-interfaces-config.tree"
```

Demo

Questions?

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

youtube.com/user/RedHatVideos

linkedin.com/company/Red-Hat

facebook.com/ansibleautomation

twitter.com/ansible