NETCONF, YANG and OpenConfig

Khelil Sator
ksator@juniper.net
Juniper Networks


OVERVIEW

One Protocol – many languages to master …

```
router bgp as-number 1234
```

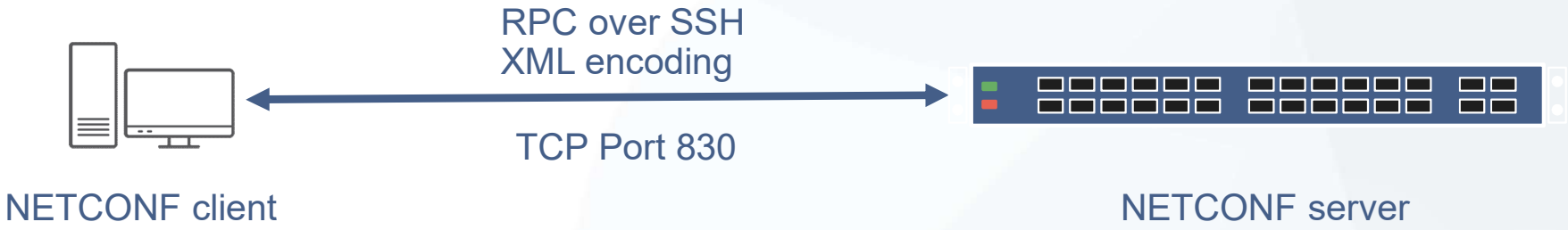
```
router bgp 1234
```

```
configure router autonomous-system 1234
```

```
set routing-options autonomous-system 1234
```

NETCONF overview

- Protocol to manipulate configuration of network devices.
- Defined by the IETF in RFC 6241



YANG overview

- Data model language for NETCONF
  - YANG definitions directly map to NETCONF XML content layer
- Human readable
- Defined by the IETF in RFC 6020

YANG Example

- https://github.com/openconfig/public/blob/master/release/models/bgp/openconfig-bgp-types.yang

```
...
  typedef peer-type {
    type enumeration {
      enum INTERNAL {
        description
          "Internal (iBGP) peer";
      }
      enum EXTERNAL {
        description
          "External (eBGP) peer";
      }
}
...
```

```
[edit openconfig-bgp:bgp peer-groups peer-group OC]
lab@dc-vmx-2# set config peer-type ?
Possible completions:
  EXTERNAL             external (eBGP) peer
  INTERNAL             internal (iBGP) peer
[edit openconfig-bgp:bgp peer-groups peer-group OC]
```

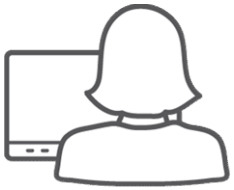
OpenConfig overview

- Vendor-neutral data models written in YANG
- Designed by a working group
- Published on Github

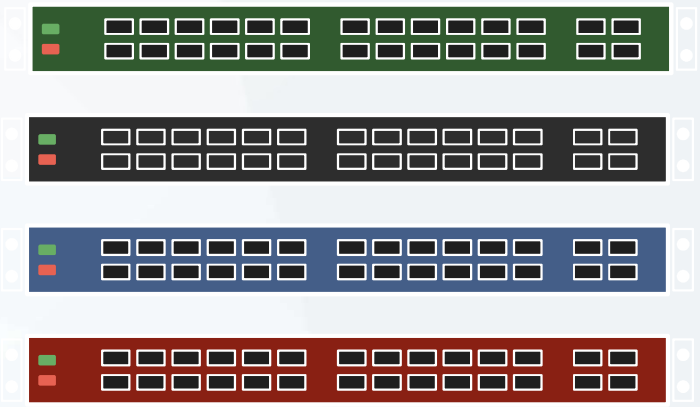
NETCONF + standardized data models written in YANG

- Common protocol manipulate configuration of network devices
  - NETCONF
- Standardized data models written in YANG
  - IETF defined YANG modules
  - OpenConfig defined YANG modules
- Result: All devices can be configured/monitored the same way

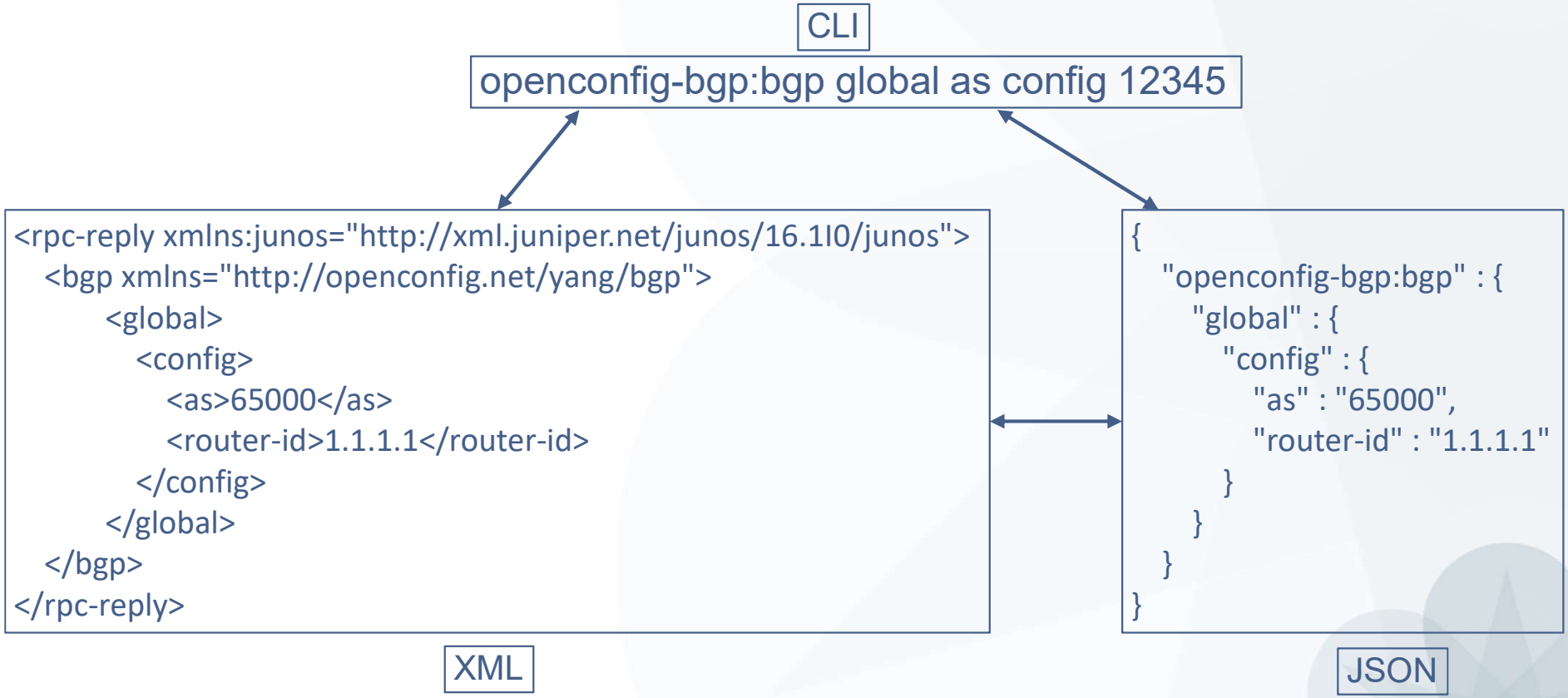
OpenConfig
One Protocol – one language to master all vendors



```
openconfig-bgp:bgp global as config 12345
```



OpenConfig
One Protocol – one language, many encoding




NETCONF

NETCONF overview

- Mechanisms to install, manipulate, and delete the configuration of network devices.
- Protocol defined in RFC 6241
- Client-server communication (the server is the network device)
- Server default port must be 830 and should be configurable (RFC 6242)
- Uses SSH
- Remote procedure calls (RPCs)
- Extensible Markup Language (XML) based data encoding for the configuration data as well as the protocol messages.

NETCONF layers

LAYER	EXAMPLE
Content	Configuration data
Operations	<edit-config>, <get-config>, …
Messages	<rpc>, <rpc-reply>
Transport protocol	SSH

NETCONF messages

- NETCONF peers use <rpc> and <rpc-reply> elements
- The <rpc> element is used to enclose a NETCONF request
  - It has a mandatory attribute "message-id", which is a string chosen by the sender of the RPC
- The <rpc-reply> message is sent in response to an <rpc> message.
  - It has a mandatory attribute "message-id", which is equal to the "message-id" attribute of the <rpc> for which this is a response.
  - The <rpc-error> element is sent in <rpc-reply> messages if an error occurs during the processing of an <rpc> request.
  - The <ok> element is sent in <rpc-reply> messages if no error occurred during the processing of an <rpc> request, and no data was returned from the operation.

NETCONF protocol operations

- low-level operations to manage device configurations and retrieve device state information.

- The base protocol includes the following protocol operations:
  - Get
  - get-config
  - edit-config
  - copy-config
  - delete-config
  - Lock
  - Unlock
  - close-session
  - kill-session

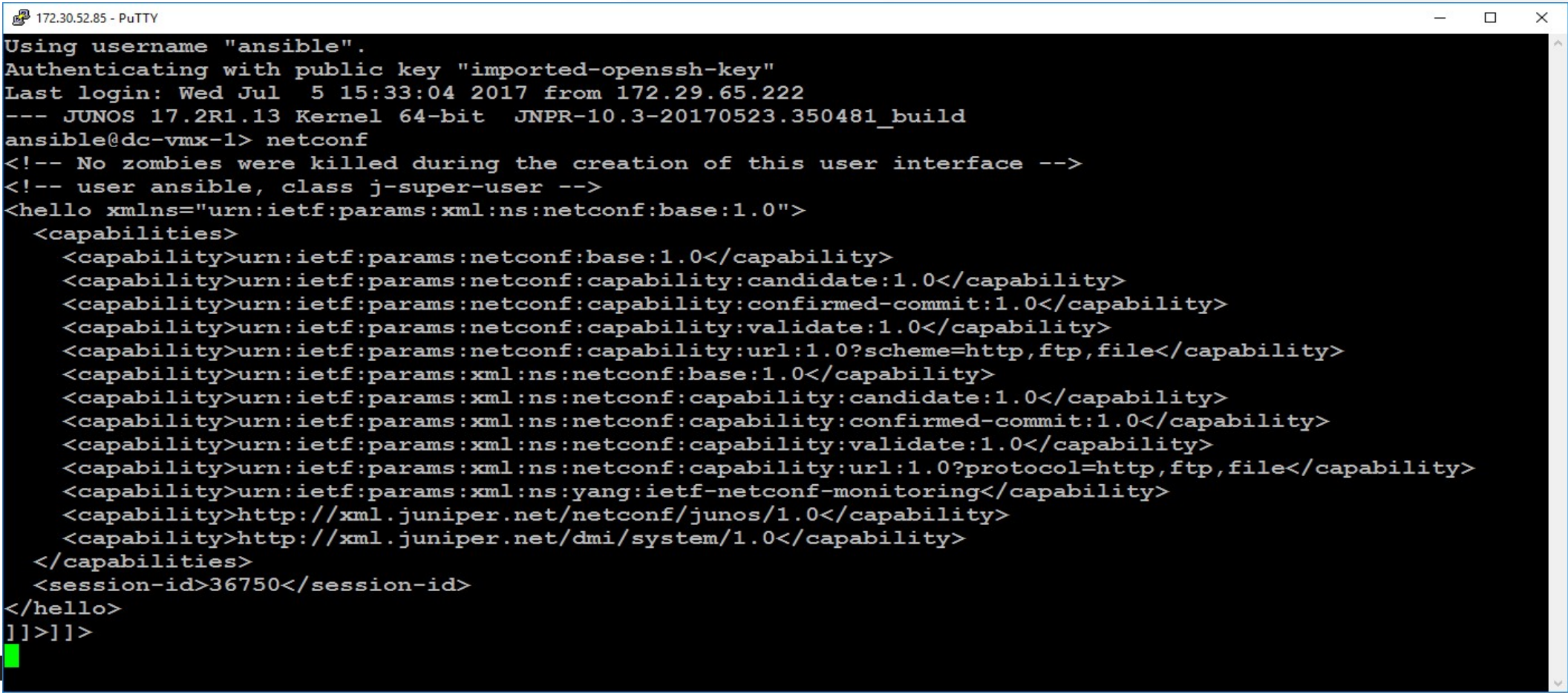
NETCONF capabilities

- Supplements the base NETCONF specification.
- Each peer advertises its capabilities by sending them during an initial capabilities exchange.
- The client discovers the server's capabilities and can use any additional operations defined by those capabilities. As example, a <commit> or <discard-changes> NETCONF operation can be used by the client if the server advertised the NETCONF capability "Candidate Configuration"
- NETCONF capabilities examples:
  - Writable-Running
  - Candidate Configuration
  - Confirmed Commit
  - Validate

NETCONF over SSH

- NETCONF over SSH is discussed in a separate RFC (6242)
- In order to open a NETCONF session inside an SSH connection, there are two options:
  - establish an SSH connection to a NETCONF server, and then run the command:
    ```
    netconf
    ```
  - invoke the NETCONF subsystem using the following command (the -s option causes the command netconf to be invoked):
    ```
    ssh device -s netconf -p 830
    ```

NETCONF server capabilities exchange using RFC 6242



NETCONF clients

- There are many NetConf clients implementations available.
  - ncclient is a Netconf Client python library.

- NETCONF provides low level operations
  - not designed for humans.
  - Rather used under the hood by larger applications

- The below tools use NETCONF under the hood and brings a level of abstraction so they are easy to use.
  - junos-eznc (PyEZ) is a Python library for JUNOS. Uses ncclient under the hood
  - NAPALM Python library uses PyEZ for JUNOS.
  - JSNAPy Python library uses PyEZ
  - Ansible modules for JUNOS uses PyEZ
  - Saltstack for JUNOS uses PyEZ

NETCONF connections on JUNOS

To allow NETCONF connections on junos devices, either configure:

```
set system services netconf ssh
```

Or alternatively, just configure:

```
set system services ssh
```

The former allows NETCONF connections on port 830,
while the latter allows NETCONF connections on port 22.

To allow access to the netconf SSH subsystem over another ports:

```
set system services netconf ssh port port-number
```

NETCONF demo

https://github.com/ksator/openconfig-demo-with-juniper-devices/wiki/06.-Openconfig-demo-using-NetConf-over-SSH


YANG

YANG overview

- RFC 6020

- A data modeling language used to model both configuration data and state data manipulated by NETCONF
  - YANG definitions directly map to NETCONF XML content
  - It determines the structure, syntax of the data used in NETCONF.
  - Allows a complete description of all data sent between a NETCONF client and server.

YANG modules

- Data model written in YANG are defined in YANG modules.

- YANG modules can be categorized as
  - IETF defined YANG modules
  - OpenConfig defined YANG modules
  - Customer defined YANG modules

YIN

- YANG modules can be translated into an equivalent XML syntax called YANG Independent Notation (YIN)

- You can do it with Pyang
  - https://github.com/ksator/openconfig-demo-with-juniper-devices/wiki/07.-Pyang-(Python-program-for-Yang)#convert-yang-into-yin

Configuration data and state data

- YANG can model configuration data (writable data), as well as state data (read-only status), based on the "config" statement
- The " config " statement is either "true" or "false".
- When a node is tagged with config true, it is flagged as configuration data.
  - it can be returned using NETCONF <get-config> operation.
  - it can be manipulated using the NETCONF <edit-config> operation.
- When a node is tagged with config false, it is flagged as state data.
  - it can be returned using NETCONF <get> operation,
  - It can not be returned using the NETCONF <get-config> operation
  - it cannot be manipulated using the NETCONF <edit-config> operation.

Data definition statements

- Data definition statement: A statement that defines new data nodes
  - Container
  - Leaf
  - leaf-list
  - list
  - ...

Leaf statement

- A leaf has exactly one value, of a particular type. And no children.

```
leaf host-name {
  type string;
  mandatory true;
  config true;
  description "Hostname for this system";
}
```

host-name is a leaf
- It has no children
- It is a string

It is a configuration data
- it can be manipulated using <edit-config> NETCONF operation
- it is returned in NETCONF <get-config> NETCONF operation

Example of NETCONF content:

```
<host-name>my.example.com</host-name>
```

Leaf statement

```
leaf cpu-temp {
  type int32
  units degrees-celsius;
  config false;
  description "Current temperature in CPU";
}
```

cpu-temp is a leaf

- It has no children
- It has a value of type int32 (32-bit signed integer)

It is a state data

- it is returned in NETCONF <get> operation
- it is not returned in NETCONF <get-config> operation
- it can not be manipulated using <edit-config> netconf operation

leaf-list statement

- A list of nodes
  - Each node is a leaf

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

Example of NETCONF content:

```
<domain-search>high.example.com</domain-search>
<domain-search>low.example.com</domain-search>
```

container statement

- has no value
- has a set of child nodes.

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

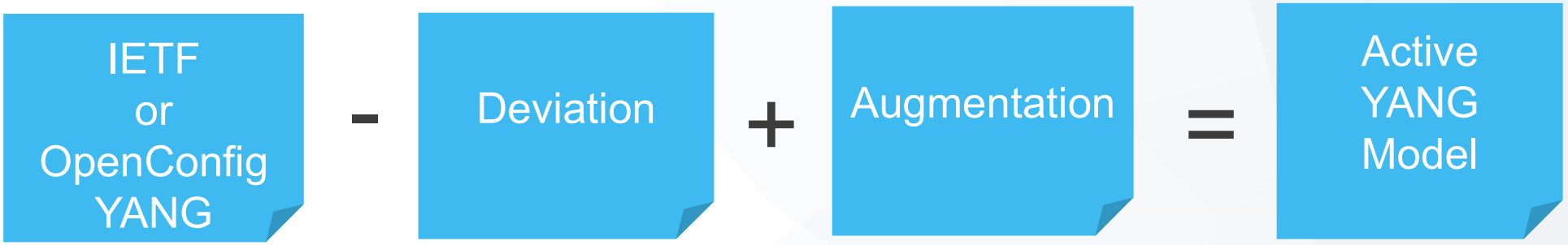
Example of NETCONF content:

```
<system>
   <login>
         <message>Good morning</message>
   </login>
</system>
```

container statement

```
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";
    }
  }
}
```

YANG Augmentation and Deviation



- Disable not supported containers and leafs
- Add containers and leafs (even into existing containers)

YANG tools - Pyang

- Validate YANG modules against YANG RFC
- Convert YANG modules into equivalent YIN module (YANG to XML).
- Generate a tree representation of YANG models for quick visualization.
- Demo: https://github.com/ksator/openconfig-demo-with-juniper-devices/wiki/07.-Pyang-(Python-program-for-Yang)

YANG tools - Pyangbind

- Generates a set of Python classes from a YANG module:
  - It converts a YANG module into a Python module, such that Python can be used to manipulate data which conforms with a data model written in YANG.

- Demo: https://github.com/ksator/openconfig-demo-with-juniper-devices/wiki/08.-Pyangbind-(Pyang-plugin)


OPENCONFIG

OPENCONFIG

- A working group of technical contributors from a variety of network operators
- They published a set of YANG modules
  - vendor-neutral data models written in YANG
  - Configuration and operational states.
- Models available on GitHub
  https://github.com/openconfig/public/tree/master/release/models

OPENCONFIG PARTICIPANTS

Google

AT&T

Microsoft

British Telecom

Facebook

Comcast

Verizon

Level 3

Cox Communications

Yahoo!


Apple

Jive Communications


Deutsche Telekom / Terastream

Bell Canada

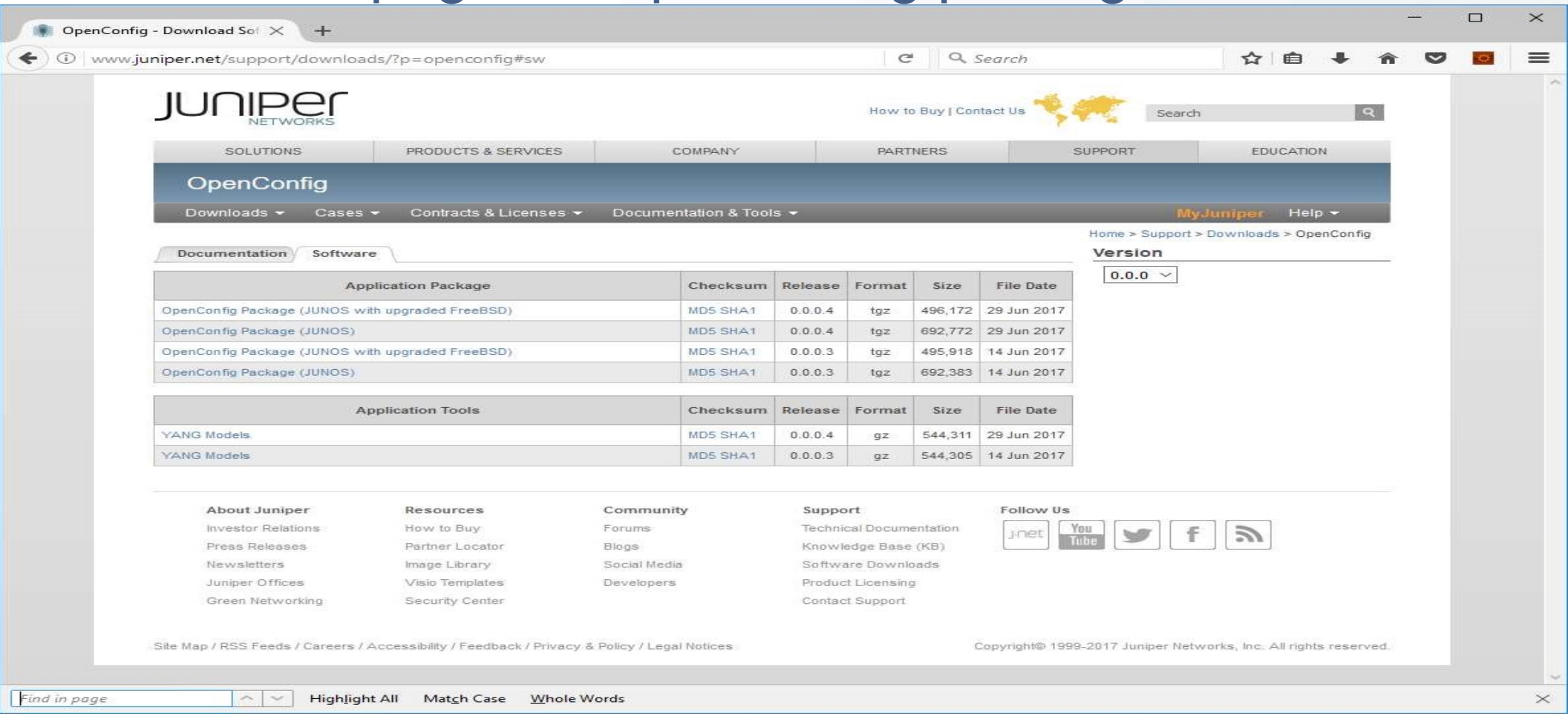
SK Telecom

Bloomberg

Netflix

Cloudflare

Download page for OpenConfig package



OpenConfig package installation on Juniper devices

- Download the OpenConfig package from Juniper download website
- Install the it on the devices:

```
lab@fabric-02> request system software add junos-openconfig-xxx
```

- Check the Openconfig details on Juniper devices:

```
lab@fabric-02> show version detail | match openconfig
JUNOS Openconfig [0.0.0.3]
```

- The YANG modules, the equivalent YIN modules, and the translation scripts are stored into these directories:

```
lab@fabric-02> file list /var/db/scripts/translation
lab@fabric-02> file list /opt/yang-pkg/junos-openconfig/yang
lab@fabric-02> file list /opt/yang-pkg/junos-openconfig/yin
lab@fabric-02> file list /opt/yang-pkg/junos-
openconfig/translation/
```

OpenConfig defined YANG modules and translation scripts

```
lab@fabric-01> show system yang package
Package ID              :junos-openconfig
YANG Module(s)          :iana-if-type.yang ietf-inet-types.yang ietf-interfaces.yang ietf-yang-types.yang jnx-aug-
openconfig-bgp.yang jnx-aug-openconfig-if-ip.yang jnx-aug-openconfig-interfaces.yang jnx-aug-openconfig-
lacp.yang jnx-aug-openconfig-lldp.yang jnx-aug-openconfig-local-routing.yang jnx-aug-openconfig-mpls.yang
jnx-aug-openconfig-routing-policy.yang jnx-openconfig-dev.yang junos-extension.yang openconfig-bgp-
multiprotocol.yang openconfig-bgp-operational.yang openconfig-bgp-policy.yang openconfig-bgp-types.yang
openconfig-bgp.yang openconfig-extensions.yang openconfig-if-aggregate.yang openconfig-if-ethernet.yang
openconfig-if-ip-ext.yang openconfig-if-ip.yang openconfig-interfaces.yang openconfig-lacp.yang openconfig-
lldp-types.yang openconfig-lldp.yang openconfig-local-routing.yang openconfig-mpls-igp.yang openconfig-mpls-
ldp.yang openconfig-mpls-rsvp.yang openconfig-mpls-sr.yang openconfig-mpls-static.yang openconfig-mpls-
te.yang openconfig-mpls-types.yang openconfig-mpls.yang openconfig-platform-types.yang openconfig-
platform.yang openconfig-policy-types.yang openconfig-rib-bgp-ext.yang openconfig-rib-bgp-types.yang
openconfig-rib-bgp.yang openconfig-routing-policy.yang openconfig-terminal-device.yang openconfig-transport-
types.yang openconfig-types.yang openconfig-vlan-types.yang openconfig-vlan.yang
Translation Script(s) :openconfig-bgp.slax openconfig-interface.slax openconfig-lldp.slax openconfig-local-
routing.slax openconfig-mpls.slax openconfig-policy.slax
Translation script status is enabled
```

OPENCONFIG DEMO

https://github.com/ksator/openconfig-demo-with-juniper-devices

https://github.com/ksator/openconfig-demo-with-juniper-devices/wiki


Thank you