# OpenConfig & NAPALM

Matej Vadnjal | Arnes

SINOG 4.0 | 24. 5. 2017

## Network automation is hard

Different vendors, different CLIs

Various data formats

Inconsistencies

Different APIs or even no API

## YANG

Data modeling language - how to structure and represent data

RFC6020

## OpenConfig

Working group of network operators

A bunch of YANG models

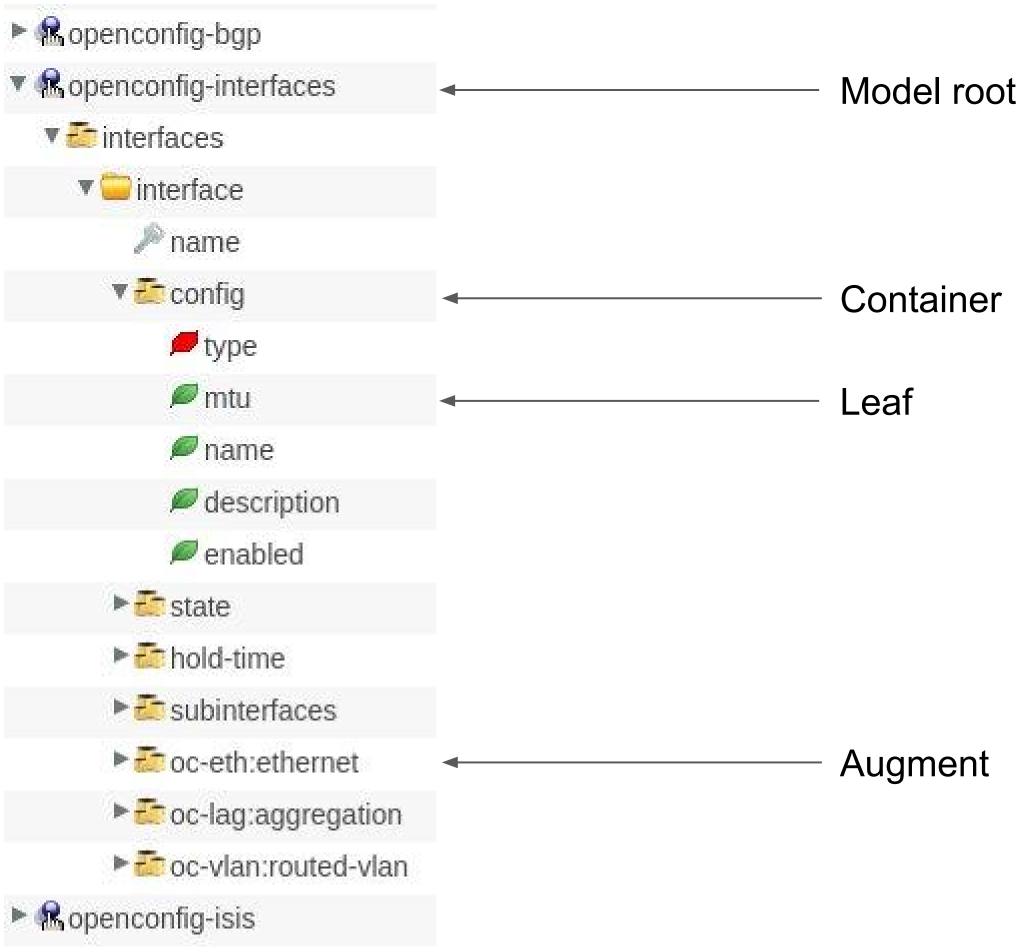
## OpenConfig

Models for:

- Interfaces
- Vlans
- Routing protocols
- Routing policy
- ACLs

and many more

# OpenConfig interfaces model



```
▶ openconfig-bgp
▼ openconfig-interfaces
    ▼ interfaces
        ▼ interface
              name
            ▼ config
                  type
                  mtu
                  name
                  description
                  enabled
            ▶ state
            ▶ hold-time
            ▶ subinterfaces
            ▶ oc-eth:ethernet
            ▶ oc-lag:aggregation
            ▶ oc-vlan:routed-vlan
▶ openconfig-isis
```

## OpenConfig and network devices operating systems

Some vendor support for YANG and OpenConfig models incoming

Still a way off for full support

Older equipment unlikely to become supported

## NAPALM

Network Automation and Programmability Abstraction Layer with Multivendor support

Unified API no matter the vendor or protocol

Retrieve state data from the device (getters)

Loading configuration (in vendor format) to the device

CLI tool and Ansible modules available

# Using NAPALM from Ansible

```
- name: Get facts from device
  napalm_get_facts:
    provider: "{{ provider }}"
    filter:
      - "facts"
      - "interfaces"
      - "interfaces_ip"
      - "lldp_neighbors"
  register: result

- name: Print gathered facts
  debug: var=result
```

# Using NAPALM from Ansible

```
"result": {
      "ansible_facts": {
          "facts": {
              "fqdn": "lbravos-tsp.cpe.arnes.si",
              "hostname": "lbravos-tsp",
              "model": "WS-C4500X-16",
              "serial_number": "JAE123456789",
              "uptime": 3442980,
              "vendor": "Cisco"
          },
          "interfaces": {
              "TenGigabitEthernet1/1": {
                  "description": "-- sbravos --",
                  "is_enabled": true,
                  "is_up": true,
                  "last_flapped": -1.0,
                  "mac_address": "C0:8C:60:6D:21:E8",
                  "speed": 1000
              },
          },
          "interfaces_ip": {
              "Loopback0": {
                  "ipv6": {
                      "2001:1470:100A::48": {
                          "prefix_length": 128
                      }
                  }
              }
```

```
"lldp_neighbors": {
    "TenGigabitEthernet1/1": [
        {
            "hostname": "sbravos.arnes.si",
            "port": "Gi0/10"
        }
    ],
    "TenGigabitEthernet1/16": [
        {
            "hostname": "lanso.arnes.si",
            "port": "Te1/1"
        }
    ]
```

# Using NAPALM from Ansible

```
"result": {
      "ansible_facts": {
          "facts": {
              "fqdn": "sbravos2.arnes.si",
              "hostname": "sbravos2",
              "model": "EX2200-24T-4G",
              "serial_number": "CW0123456789",
              "uptime": 32554440,
              "vendor": "Juniper"
          },
          "interfaces": {
              "ge-0/0/0": {
                  "description": "",
                  "is_enabled": true,
                  "is_up": false,
                  "last_flapped": 55277398.0,
                  "mac_address": "40:B4:F0:CB:4E:43",
                  "speed": -1
              },
          },
          "interfaces_ip": {
              "vlan.501": {
                  "ipv4": {
                      "178.172.79.204": {
                          "prefix_length": 29
                      }
                  }
              }
```

```
"lldp_neighbors": {
    "ge-0/0/23.0": [
        {
            "hostname": "sbravos.arnes.si",
            "port": "-- sbravos2 --"
        }
    ]
```

## NAPALM is great, but...

Many more getters available

- But not all on every platform

Arbitrary data structure

- Takes time to integrate with other software

## napalm-yang

Uses YANG models to transform native configs into Python objects and vice versa

“Profiles” define mappings of data between native config and YANG model values

# IOS Profile - config parser

```
interfaces:
    _process: unnecessary
    interface:
        _process:
            - mode: block
              regexp: "(?P<block>interface (?P<key>(\\w|-)*\\d+)\n(?:.|\n)*?^!$)"
              from: "{{ bookmarks.interfaces.0 }}"
        config:
            enabled:
                _process:
                    - mode: is_present
                      regexp: "(?P<value>no shutdown)"
                      from: "{{ bookmarks.interface[interface_key] }}"
            description:
                _process:
                    - mode: search
                      regexp: "description (?P<value>.*)"
                      from: "{{ bookmarks.interface[interface_key] }}"
            type:
                _process:
                    - mode: map
                      regexp: "(?P<value>(\\w|-)*)\\d+"
                      from: "{{ interface_key }}"
                      map:
                          GigabitEthernet: ethernetCsmacd
                          Loopback: softwareLoopback
                          Vlan: l3ipvlan
            mtu:
                _process:
                    - mode: search
```

## Parsing native configs

IOS

```
[...]
interface TenGigabitEthernet1/16
 description classroom
 ip address 10.100.100.10 255.255.255.0
[...]
```

Junos

```
<configuration>
    <interfaces>
        [...]
        <interface>
            <name>ge-0/0/16</name>
            <unit>
                <name>0</name>
                <description>classrooms</description>
                <family>
                    <inet>
                        <address>
                            <name>10.100.100.10/24</name>
                        </address>
                    </inet>
                </family>
            </unit>
        </interface>
        [...]
    </interfaces>
</configuration>
```

## Parsing native configs

IOS

```
[...]
interface TenGigabitEthernet1/16
 description classroom
 ip address 10.100.100.10 255.255.255.0
[...]
```

Junos

```
<configuration>
    <interfaces>
        [...]
        <interface>
            <name>ge-0/0/16</name>
            <unit>
                <name>0</name>
                <description>classrooms</description>
                <family>
                    <inet>
                        <address>
                            <name>10.100.100.10/24</name>
                        </address>
                    </inet>
                </family>
            </unit>
        </interface>
        [...]
    </interfaces>
</configuration>
```

```
{
  "interfaces": {
    "interface": {
      "ge-0/0/16": {
        "config": {
          "enabled": true,
          "name": "ge-0/0/16",
          "type": "ethernetCsmacd"
        },
        "name": "ge-0/0/16",
        "subinterfaces": {
          "subinterface": {
            "0": {
              "config": {
                "description": "classrooms",
                "enabled": true,
                "name": "0"
              },
              "index": "0",
              "ipv4": {
                "addresses": {
                  "address": {
                    "10.100.100.10": {
                      "config": {
                        "ip": "10.100.100.10",
                        "prefix-length": 24
                      },
                      "ip": "10.100.100.10"
                    }
                  }
                },
                "config": {
                  "enabled": true
                }
              }
```

## Translating into native configs

```
{
  "interfaces": {
    "interface": {
      "ge-0/0/10": {
        "config": {
          "enabled": false,
          "mtu": 1260,
          "name": "ge-0/0/10",
          "type": "ethernetCsmacd"
        },
        "name": "ge-0/0/10",
        "subinterfaces": {
          "subinterface": {
            "0": {
              "config": {
                "description": "laboratory"
              },
              "index": "0",
              "ipv4": {
                "addresses": {
                  "address": {
                    "10.200.200.10": {
                      "config": {
                        "ip": "10.200.200.10",
                        "prefix-length": 24
                      },
                      "ip": "10.200.200.10"
                    }
                  }
                }
              }
```

# Translating into native configs

```
{
  "interfaces": {
    "interface": {
      "ge-0/0/10": {
        "config": {
          "enabled": false,
          "mtu": 1260,
          "name": "ge-0/0/10",
          "type": "ethernetCsmacd"
        },
        "name": "ge-0/0/10",
        "subinterfaces": {
          "subinterface": {
            "0": {
              "config": {
                "description": "laboratory"
              },
              "index": "0",
              "ipv4": {
                "addresses": {
                  "address": {
                    "10.200.200.10": {
                      "config": {
                        "ip": "10.200.200.10",
                        "prefix-length": 24
                      },
                      "ip": "10.200.200.10"
                    }
                  }
                }
              }
```

IOS

```
interface ge-0/0/10
 shutdown
 mtu 1260
 exit
interface ge-0/0/10.0
 ip address 10.200.200.10 255.255.255.0
 description laboratory
 exit
```

Junos

```
<configuration>
  <interfaces>
    <interface>
      <name>ge-0/0/10</name>
      <unit>
        <name>0</name>
        <family>
          <inet>
            <address>
              <name>10.200.200.10/24</name>
            </address>
          </inet>
        </family>
        <description>laboratory</description>
      </unit>
      <disable/>
      <mtu>1260</mtu>
    </interface>
  </interfaces>
</configuration>
```

## Populating values programmatically

```
config = napalm_yang.utils.base.Root()
config.add_model(napalm_yang.models.openconfig_interfaces)

iface = config.interfaces.interface.add('ge-0/0/10')
iface.config.name = 'ge-0/0/10'
iface.config.enabled = False
iface.config.mtu = 1260
iface.config.type = 'ethernetCsmacd'

sub_iface = iface.subinterfaces.subinterface.add('0')
sub_iface.config.description = 'laboratory'

sub_iface.ipv4.addresses.address.add('10.200.200.10')
sub_iface.ipv4.addresses.address['10.200.200.10'].config.ip = '10.200.200.10'
sub_iface.ipv4.addresses.address['10.200.200.10'].config.prefix_length = 24
```

## Diff YANG objects

```
{
    "interfaces": {
        "interface": {
            "both": {
                "Port-Channel1": {
                    "config": {
                        "mtu": {
                            "first": "0",
                            "second": "9000"
                        }
                    }
                }
            },
            "first_only": [
                "Loopback0"
            ],
            "second_only": [
                "Loopback1"
            ]
        }
    }
}
```

## Validation

```
---
- to_dict:
    _kwargs:
        filter: true
    interfaces:
        interface:
            Et1:
                config:
                    mtu: 9000
            Et2:
                config:
                    mtu: 9000
            _mode: strict
```

```
>>> report = config.compliance_report("validate.yaml")
```

# Validation

```
{
    "complies": false,
    "skipped": [],
    "to_dict": {
        "complies": false,
        "extra": [],
        "missing": [],
        "present": {
            "interfaces": {
                "complies": false,
                "diff": {
                    "complies": false,
                    "extra": [],
                    "missing": [],
                    "present": {
                        "interface": {
                            "complies": false,
                            "diff": {
                                "complies": false,
                                "extra": [],
                                "missing": [],
                                "present": {
                                    "Et1": {
                                        "complies": true,
                                        "nested": true
                                    },
```

```
"Et2": {
    "complies": false,
    "diff": {
        "complies": false,
        "extra": [],
        "missing": [],
        "present": {
            "config": {
                "complies": false,
                "diff": {
                    "complies": false,
                    "extra": [],
                    "missing": [],
                    "present": {
                        "mtu": {
                            "actual_value": 1500,
                            "complies": false,
                            "nested": false
                        }
                    }
                }
            [...]
```

## Ansible modules

napalm_parse_yang

Parses native configuration from a device or file

napalm_diff_yang

diff two YANG objects

napalm_translate_yang

Translate a YANG object into native configuration

## Resources & questions

http://www.openconfig.net/

https://napalm.readthedocs.io/

https://napalm-yang.readthedocs.io/