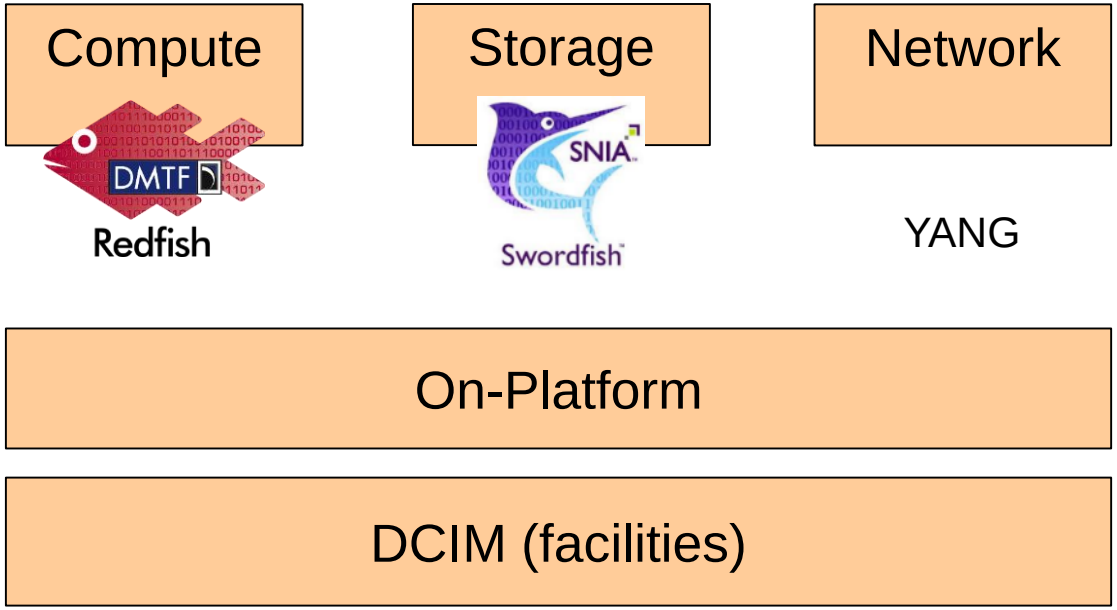
Redfish Model for an Ethernet Switch Phase 3

DMTF Redfish Forum

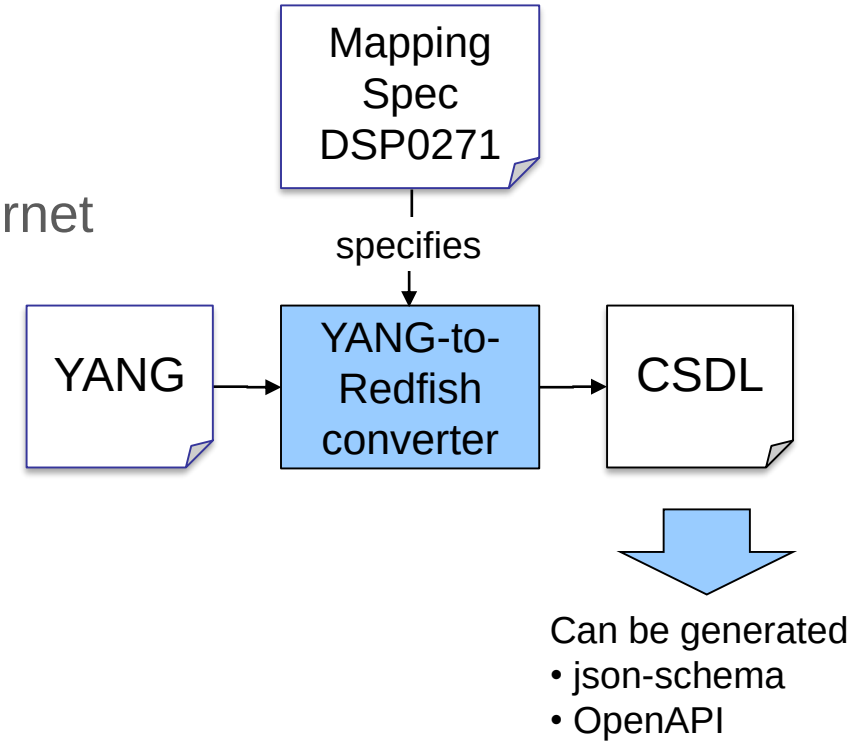
Using Redfish to manage the Network devices

- ✓ **Compute** - the DMTF released models to manage the compute platform and service
- ✓ **Storage** - SNIA released models to manage networked storage and storage services
- **Network**
  - YANG modules represent the consensus of networking industry experts
  - Convert YANG modules to Redfish schema to leverage this expertise
  - Focus on YANG modules to manage an Ethernet switch



Tactic - Convert YANG modules to Redfish CSDL

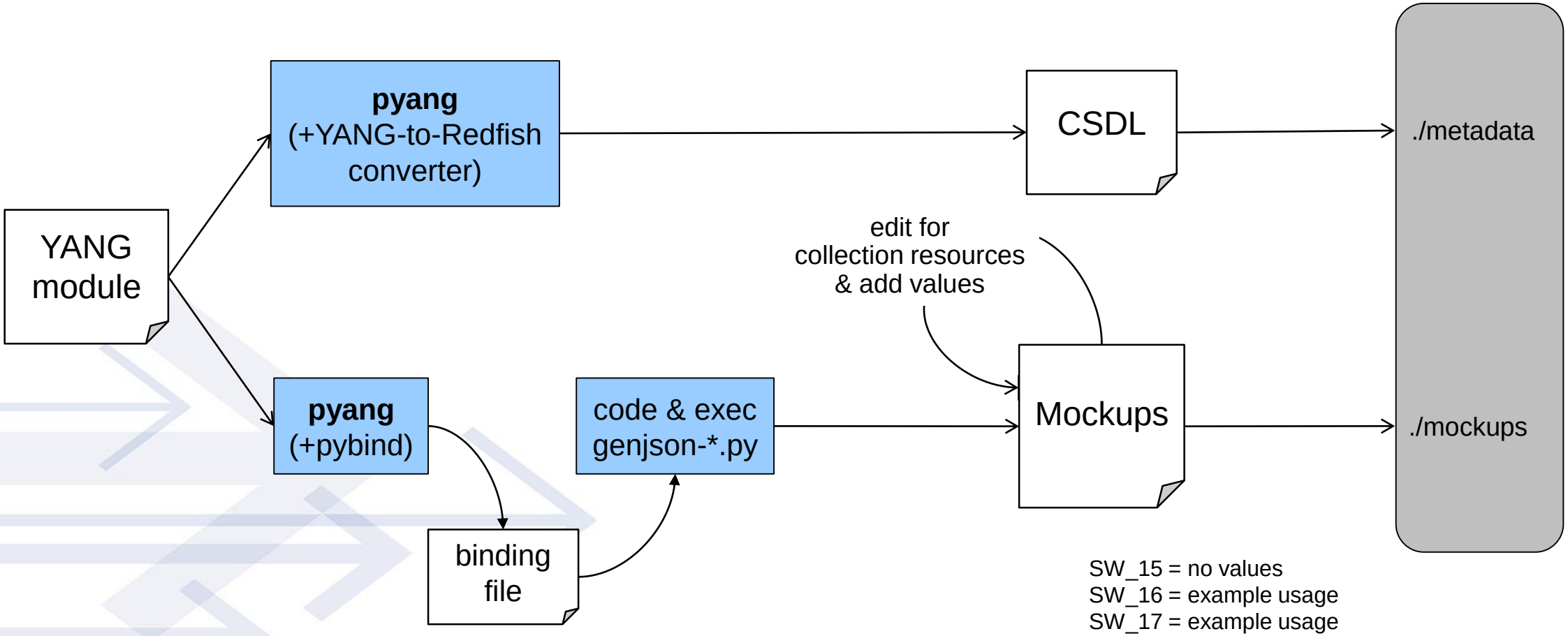
- Phase 1 - DSP-IS00004 v0.9 WIP
  - Converted IETF YANG modules - manage a network device
  - Created mockups by hand
- Phase 2 - DSP-IS00004 v0.9b WIP
  - Converted OpenConfig YANG modules - manage an Ethernet Switch
  - Minor edits to CSDL for converter issues
  - Auto-generated mockup structures
- Phase 3
  - Convert OpenConfig YANG modules
  - Auto-generate mockup structures
  - Propose - WIP release as DSP-IS00004 v0.9c



CSDL = Common Schema Definition Language

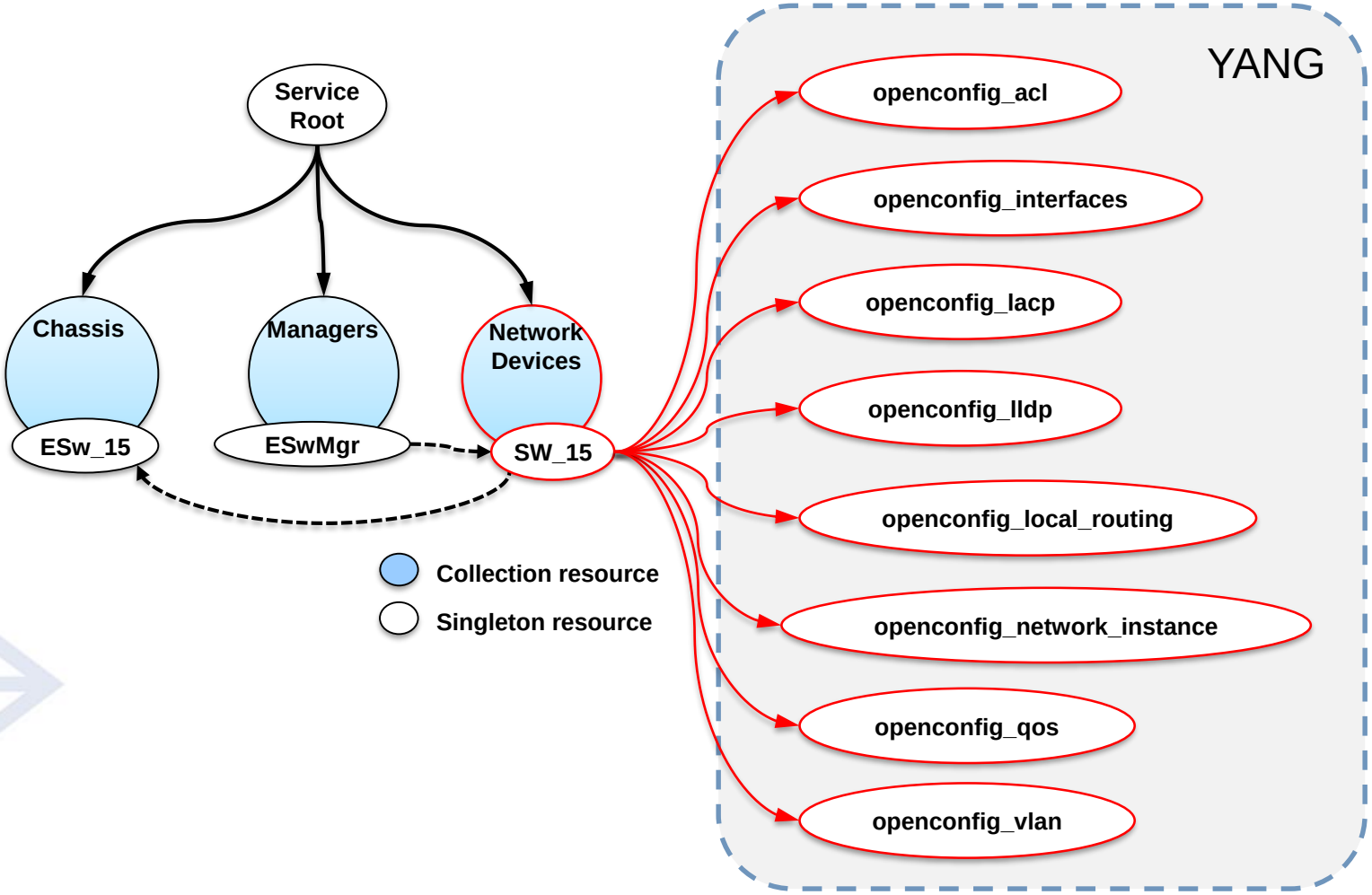
Mapping Spec - https://www.dmtf.org/dsp/DSP0271

Converting YANG module to metadata and mockups

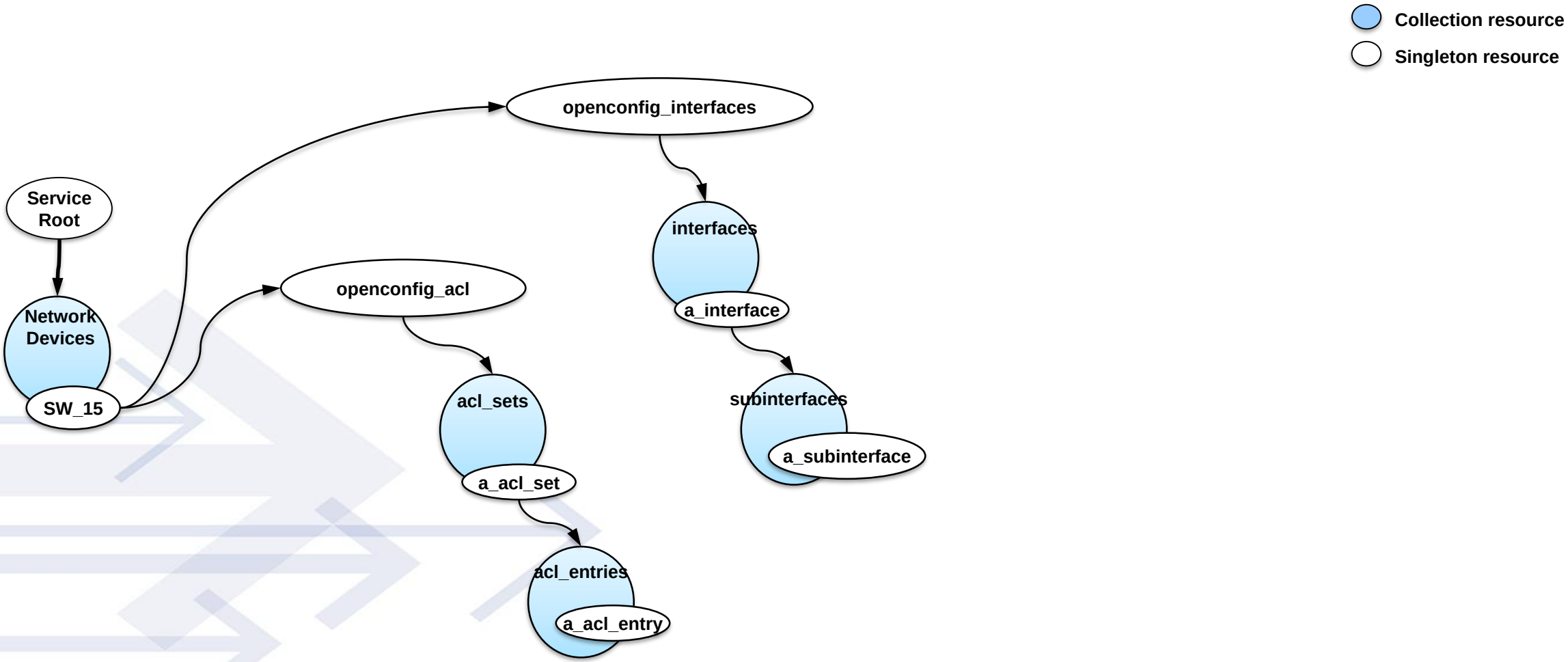


Managing an Ethernet Switch

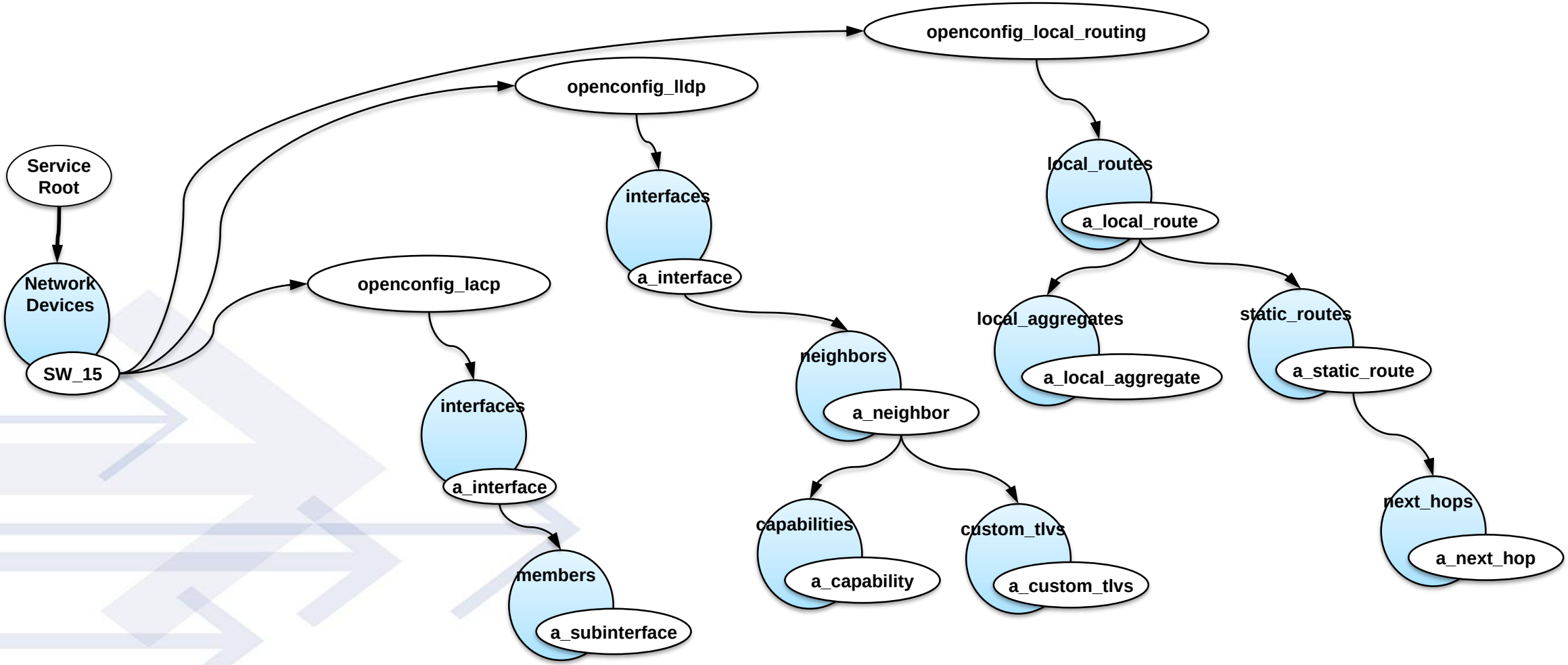
- The EthernetSwitch is a type NetworkDevice resource.
- The NetworkDevice resource is the attachment point for the resources mapped from the YANG modules



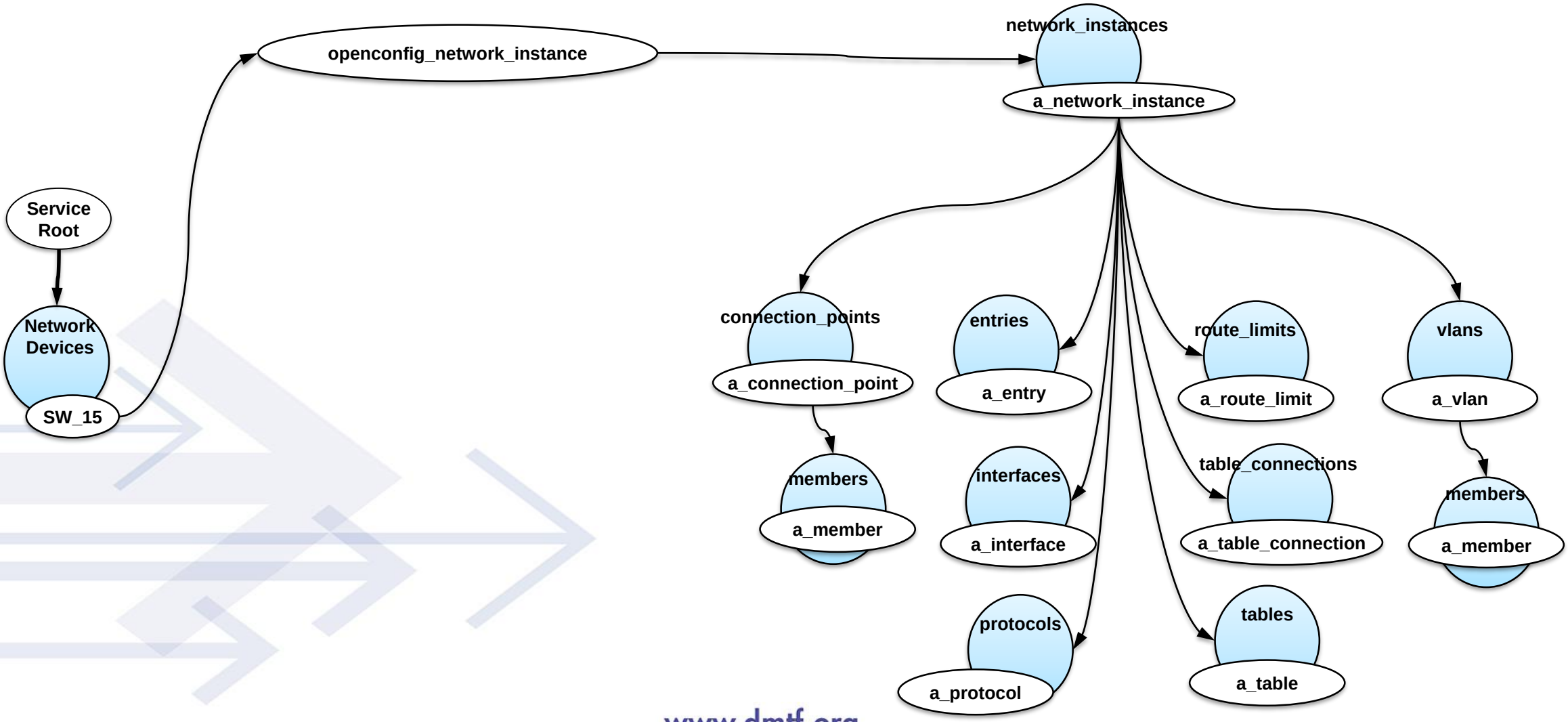
The NetworkDevice Resource - openconfig_acl & openconfig_interfaces



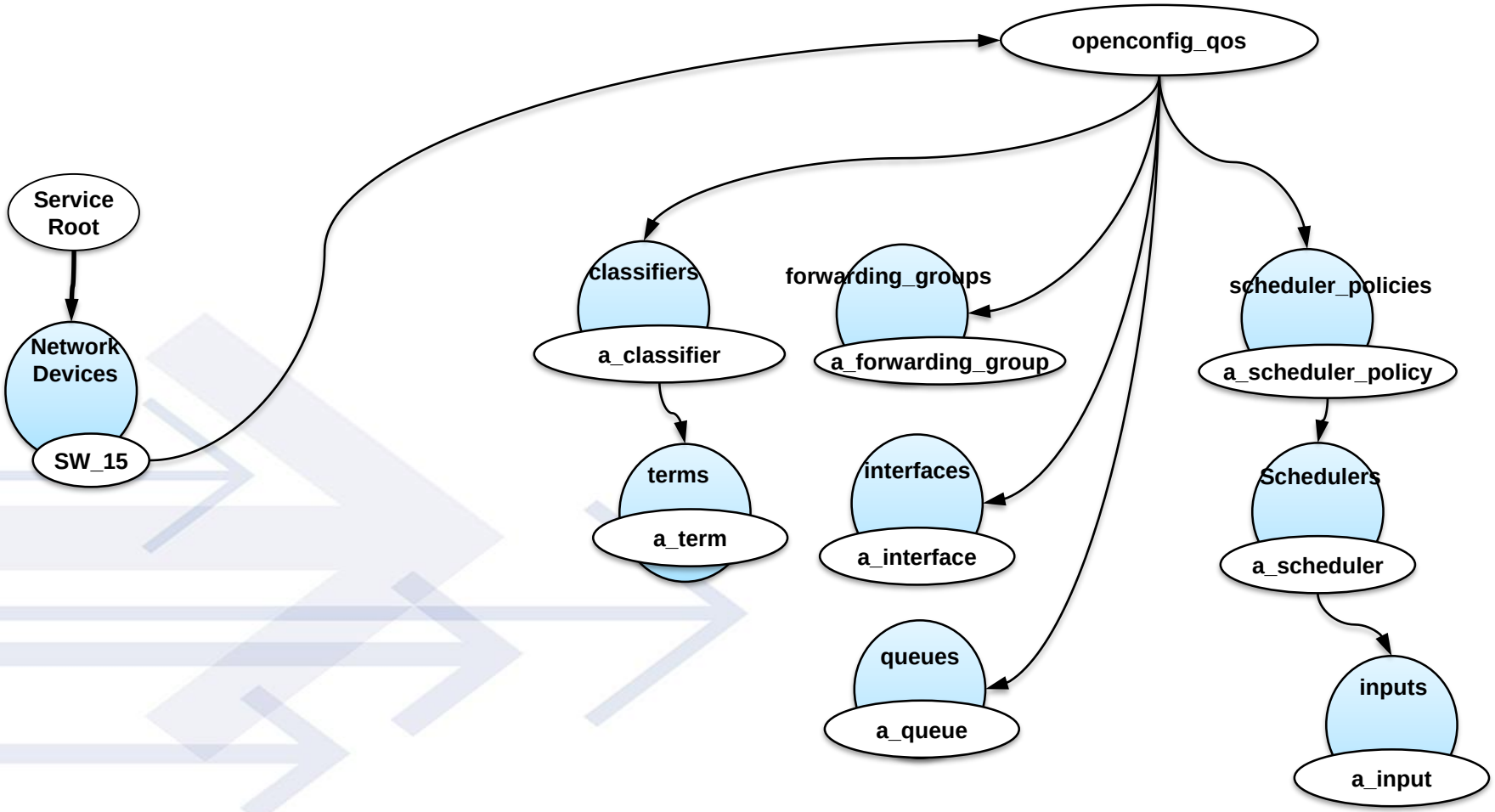
The NetworkDevice Resource - openconfig_lacp, _lldp & _local_routing



The NetworkDevice Resource - openconfig_network_instance



The NetworkDevice Resource - openconfig_qos



Contents of Work-in-Progress

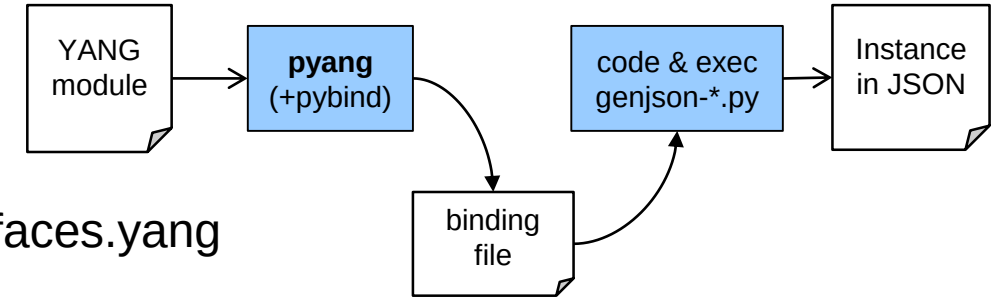
./mockups/public-ethernet-switch-openconfig

- ./Chassis/EthernetSwitch_15
- ./Managers/EthernetSwitchManager
- ./NetworkDevices/SW_15
- ./NetworkDevices/SW_16
- ./NetworkDevices/SW_17

./metadata

- RedfishYangExtension_v1.xml
- **openconfig_acl_v1.xml**
- **openconfig_interfaces_v1.xml**
- **openconfig_lacp_v1.xml**
- **openconfig_lldp_v1.xml**
- **openconfig_local_routing_v1.xml**
- **openconfig_network_instance_v1.xml**
- **openconfig_qos_v1.xml**
- **openconfig_vlan_v1.xml**
- ietf_inet_types_v1.xml
- ietf_yang_types_v1.xml
- minimal_v1.xml
- minimal_base_v1.xml
- openconfig_extensions_v1.xml
- openconfig_inet_types_v1.xml
- openconfig_network_instance_types_v1.xml
- openconfig_policy_types_v1.xml
- openconfig_types_v1.xml
- openconfig_vlan_types_v1.xml
- openconfig_yang_types_v1.xml

Generating Mockup Files



- Generate binding file

  $ pyang -f pybind -o binding_interfaces.py openconfig-interfaces.yang

- Create python file - genjson-interface.py

```
import pyangbind.lib.pybindJSON as pybindJSON
from binding_interfaces import openconfig_interfaces
oclr = openconfig_interfaces()
print(pybindJSON.dumps(oclr, filter=False))
```

'filter=False' means display empty properties

- Execute python file

  $ python genjson-interfaces.py

```
{
    "name": "",
    "interfaces": {
        "interface": {}
    }
    . . .
}
```

Example ../openconfig_interfaces/interfaces/TenGig_2

- Edit genjson-interfaces.py

```
import pyangbind.lib.pybindJSON as pybindJSON
from binding_interfaces import openconfig_interfaces
oclr = openconfig_interfaces()
rt = oclr.interfaces.interface.add("TenGig_2")
print(pybindJSON.dumps(ocl, filter=False))
```

- Execute

$ python genjson-interfaces.py ——— output →

```
{
   "name": "TenGig_2",
   "state": {
      "name": "",
      "type": "",
      "loopback-mode": false,
      "ifindex": 0,
      "mtu": 0,
      "oper-status": "",
      "enabled": 1,
      "counters": {
         "in-discards": 0,
         "out-unicast-pkts": 0,
         "in-unknown-protos": 0,
         "carrier-transitions": 0,
      . . .
      }
   },
   "subinterfaces": {
      "subinterface": {}
   },
   "config": {
      "type": "",
      "loopback-mode": false,
      "name": "",
      "mtu": 0,
      "enabled": 1
   },
   "hold-time": {
      "state": { "down": 0, "up": 0 },
      "config": { "down": 0, "up": 0 }
   }
}
```

- Edit output
  - Add Redfish properties
    - @odata properties
    - "Id" property
  - Add property values
  - Create collection resources

Additional Documentation

- YANG-to-Redfish Mapping Specification (DMTF work-in-progress)
  - Specifies how to convert a YANG model to a Redfish model
  - http://www.dmtf.org/sites/default/files/standards/documents/DSP0271_0.5.6.pdf
- YANG-to-Redfish-Converter
  - Open source repository for converter
  - https://github.com/DMTF/YANG-to-Redfish-Converter