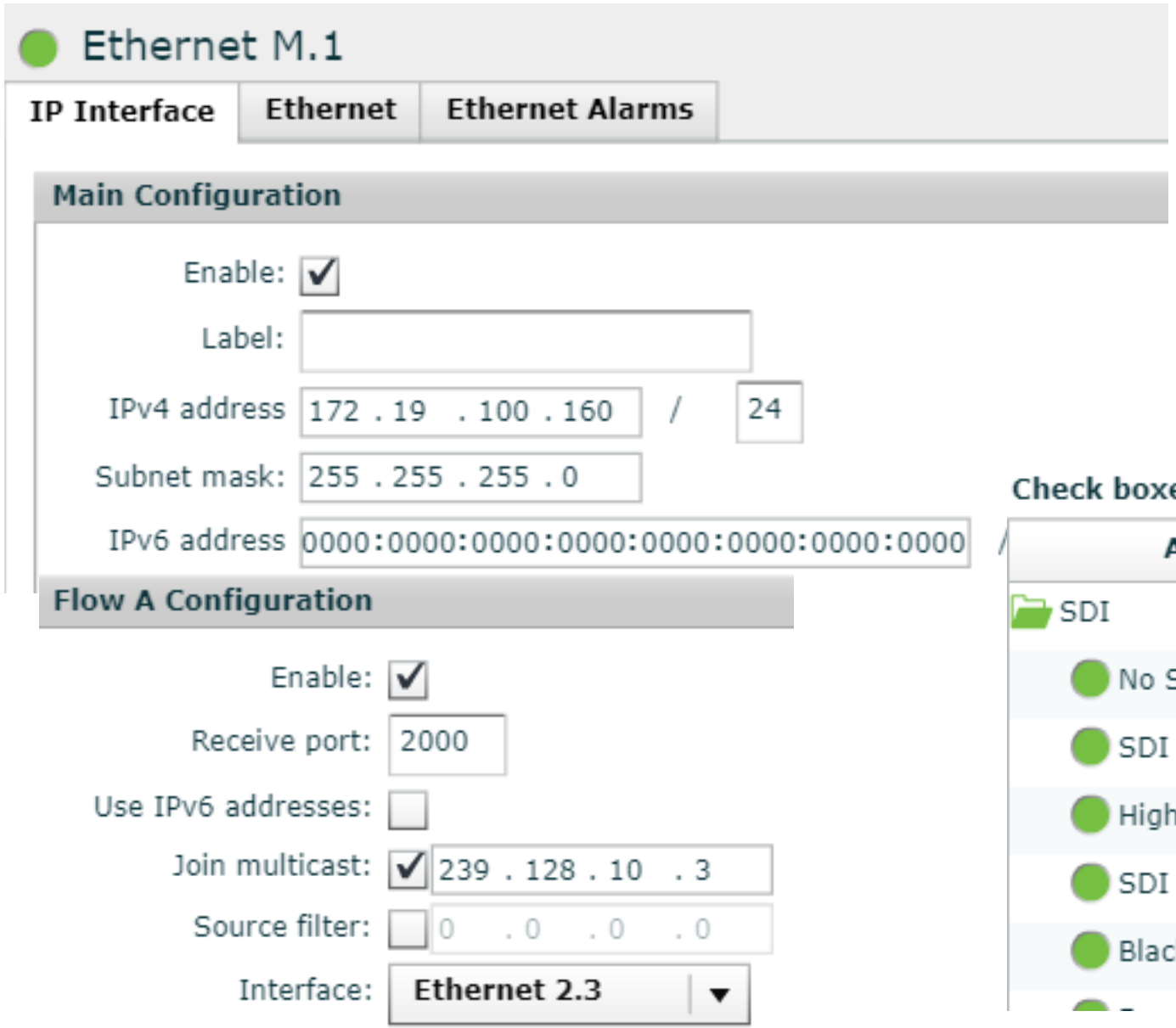
Proposal for Programmatic Configuration of Media Devices with NETCONF*/YANG

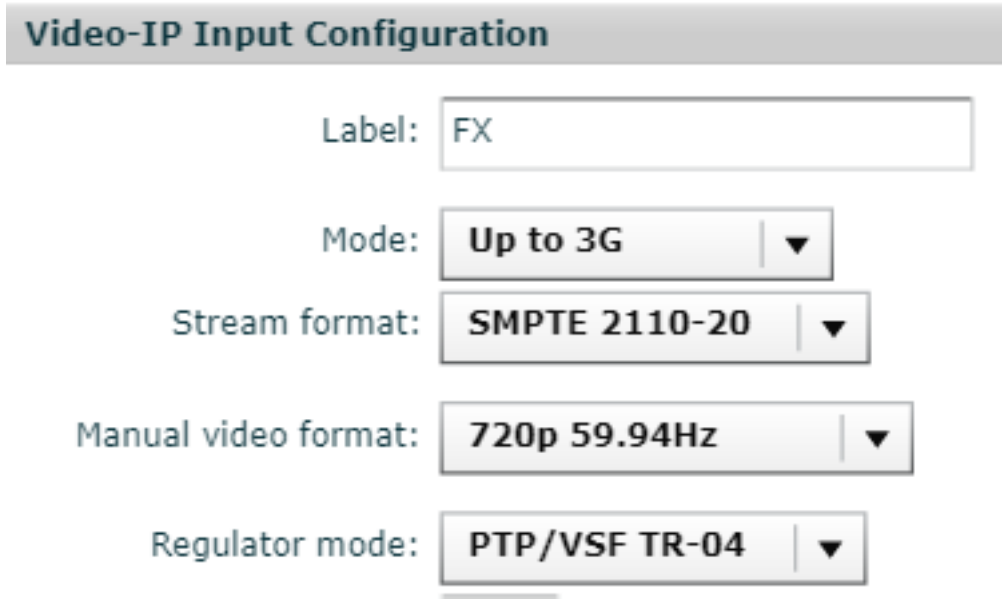
Thomas Edwards

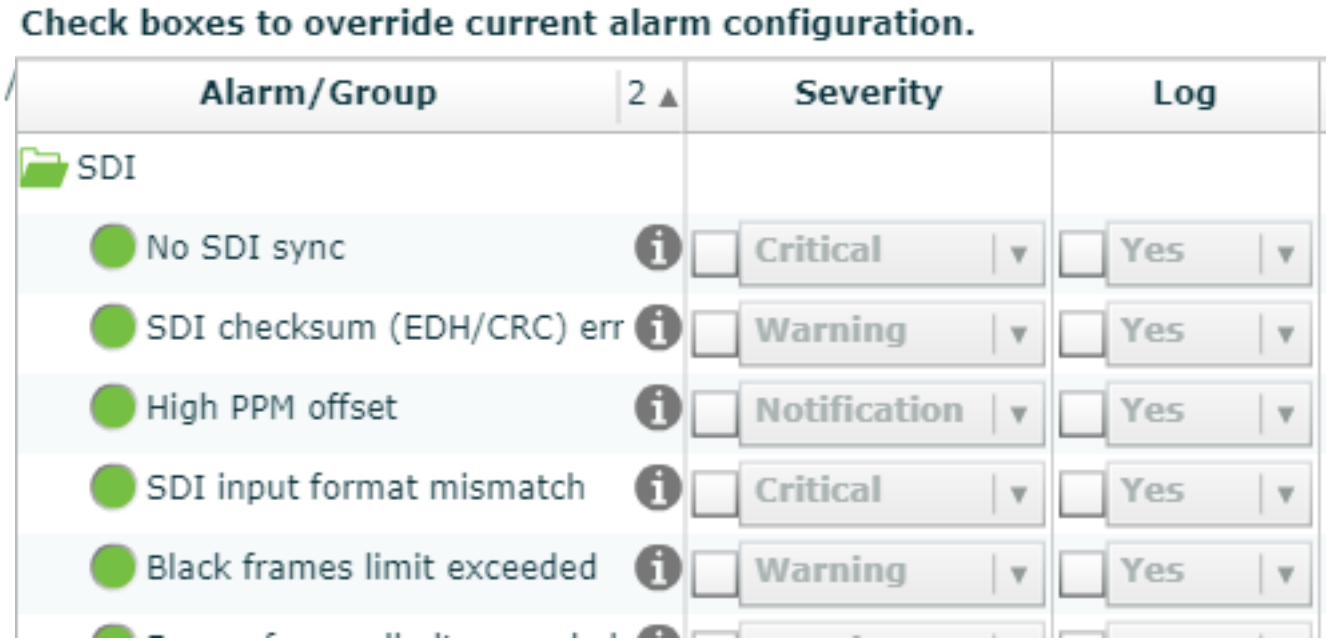
Walt Disney Television

Note: This is a technical proposal that does not necessarily represent business viewpoints or strategies of The Walt Disney Company.

The problem…


Ethernet M.1
IP Interface
Ethernet
Ethernet Alarms
Main Configuration
Enable:
Label:
IPv4 address
172 . 19 . 100 . 160
/
24
Subnet mask:
255 . 255 . 255 . 0
IPv6 address
0000:0000:0000:0000:0000:0000:0000:0000
Flow A Configuration
Enable:
Receive port:
2000
Use IPv6 addresses:
Join multicast:
239 . 128 . 10 . 3
Source filter:
0 . 0 . 0 . 0
Interface:
Ethernet 2.3


Video-IP Input Configuration
Label:
FX
Mode:
Up to 3G
Stream format:
SMPTE 2110-20
Manual video format:
720p 59.94Hz
Regulator mode:
PTP/VSF TR-04


Check boxes to override current alarm configuration.
Alarm/Group
2
Severity
Log
SDI
No SDI sync
Critical
Yes
SDI checksum (EDH/CRC) err
Warning
Yes
High PPM offset
Notification
Yes
SDI input format mismatch
Critical
Yes
Black frames limit exceeded
Warning
Yes

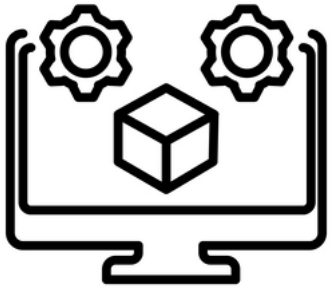
Media Device Configuration Today…

- Web GUI

- Java App

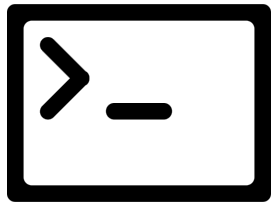
- Front Panel

- Virtual machine running a web GUI



- Proprietary REST API

- Command line

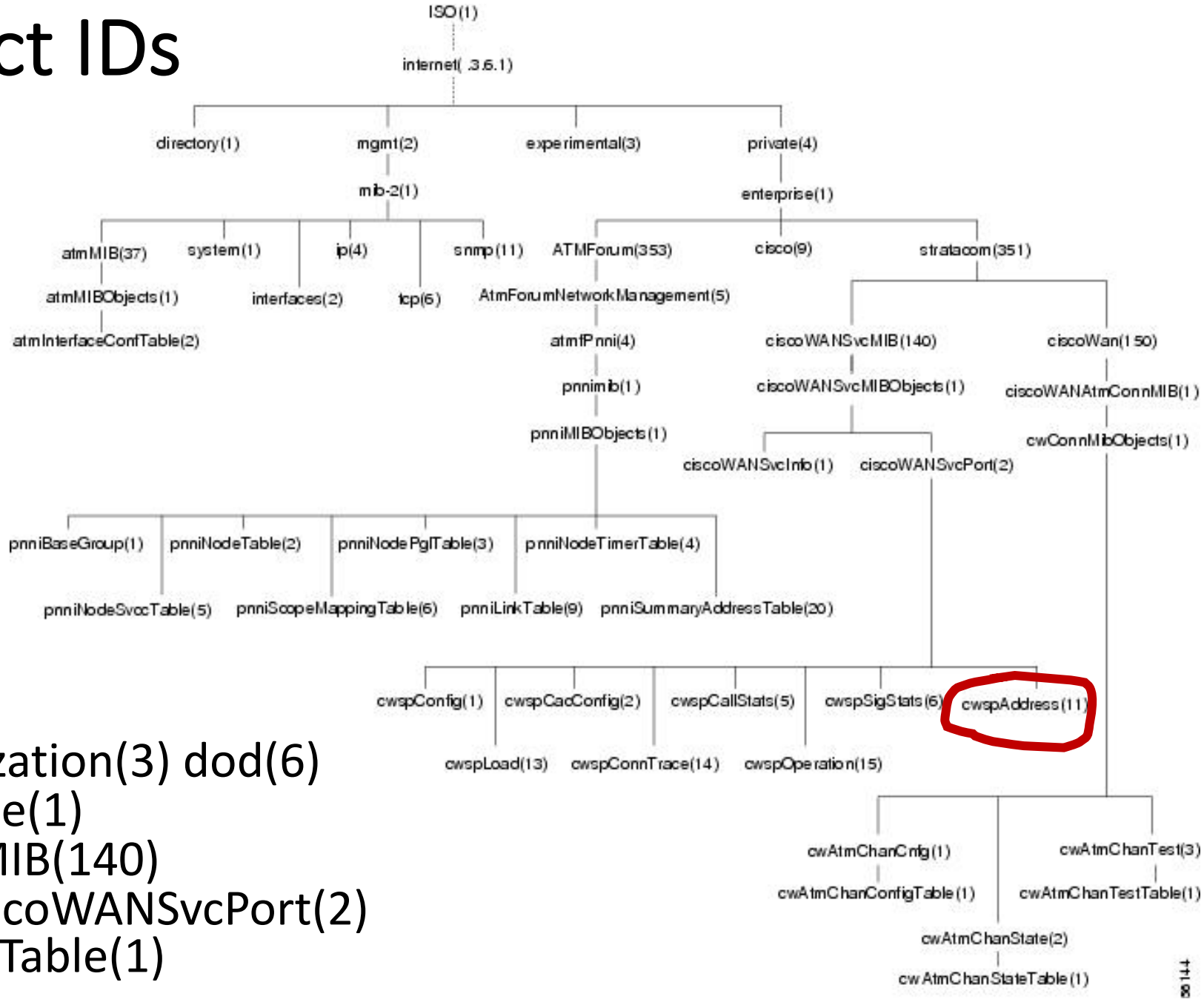


How to solve Programmatic Configuration… SNMP?

- IETF Simple Network Management Protocol (SNMP, RFC 1067) in 1988
  - Designed as a network management protocol
  - But ended up used mainly as a network monitoring protocol

- Internet Architecture Board (IAB), held a workshop in June, 2002 to identify the shortcomings of SNMP and look for alternatives
  - SNMP protocol constraints made it hard to implement complex management information bases (MIBs)
  - SNMP does not support easy retrieval, playback, or rollback of configurations
  - Scaling problems of SNMP
  - CLI commands easier to deal with than ASN.1-encoded SNMP

SNMP ASN.1 Object IDs



cwspAddress

1.3.6.1.4.1.351.140.1.2.11

A.K.A.: iso(1) identified-organization(3) dod(6) internet(1) private(4) enterprise(1) stratacom(351) ciscoWANSvcMIB(140) ciscoWANSvcMIBObjects(1) ciscoWANSvcPort(2) cwspAddress(11) cwspAddressTable(1)

Command Line Interface (CLI) to the Rescue?

- CLI much easier to use than SNMP OIDs
- Different vendors had different CLIs
- Not (usually) transactional
- CLIs can change with SW version
- CLI outputs not always easily parsable

```
Router# clock set 13:32:00 February 01
                                    ^
% Invalid input detected at '^' marker.
```

RFC 3535 - Operator Requirements (2003)

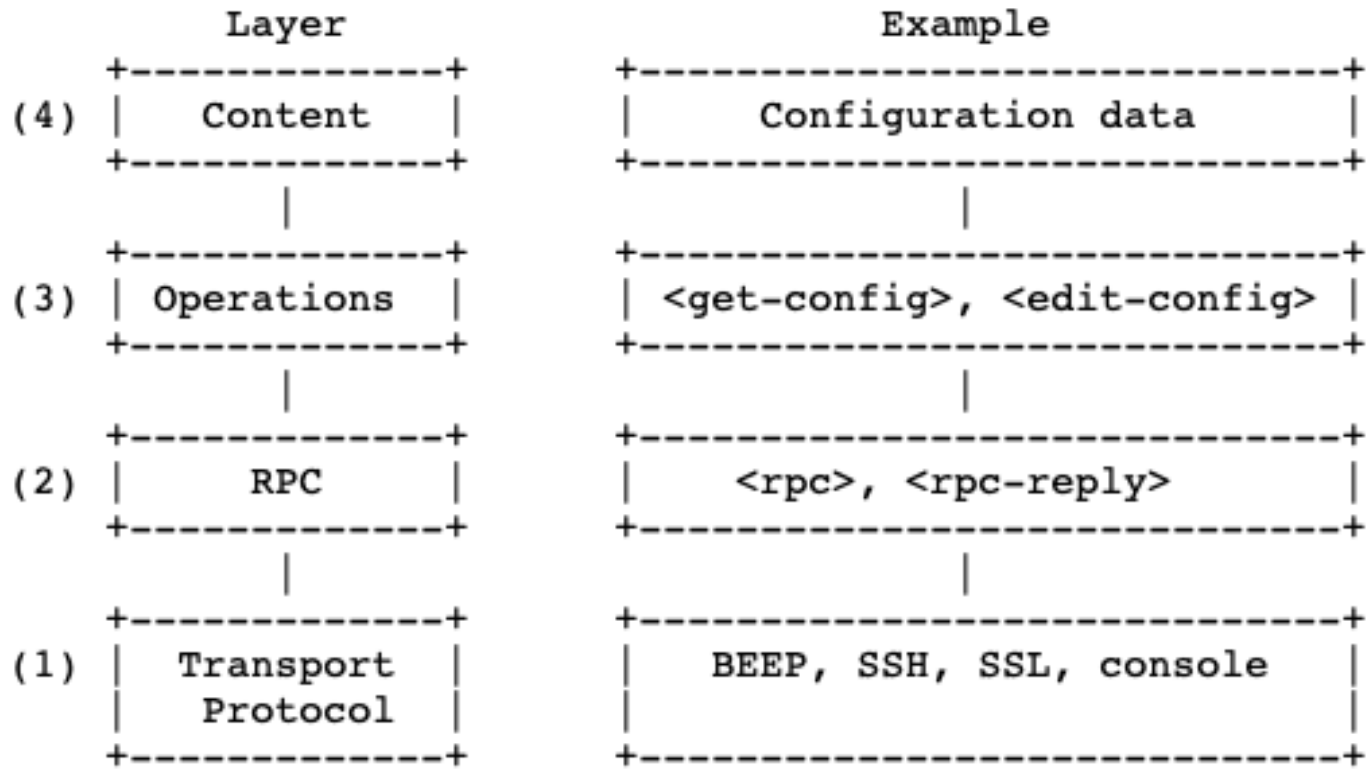
- Ease of use
- Clear distinction between configuration data & statistics
- Separately fetch configs, operational state, and statistics from devices
- Concentration on configuration of the network as a whole rather than individual devices
- Support for configuration transactions, dump & restore configs, config activations, consistency checks of configs
- Text processing tools & version management tools to process configs

Transactions: ACID

- Atomic
  - All changes to data are performed as if they are a single operation
  - Transactions are all-or-nothing
  - No internal ordering of transaction required
- Consistent
  - Data is in a consistent state when a transaction starts and when it ends
  - Configurations must meet validation rules
  - Interrupted configurations rolled back to previous state
- Isolated
  - Simultaneous transactions do not interfere with each other
- Durable
  - Committed data sticks – remains in the system even in the case of a fail-over, power failure, restart, etc.

RFC 4741 NETCONF (2006)

- Mechanisms to install, manipulate, and delete the configuration of network devices
- XML for encoding configs
- Realized on Remote Procedure Call (RPC) layer
- Transport mainly SSH



Configuration Datastores (NETCONF RFC 6241)

Set of configuration information required to get device from initial default state to desired operational state.

<running>

- Currently active configuration of device

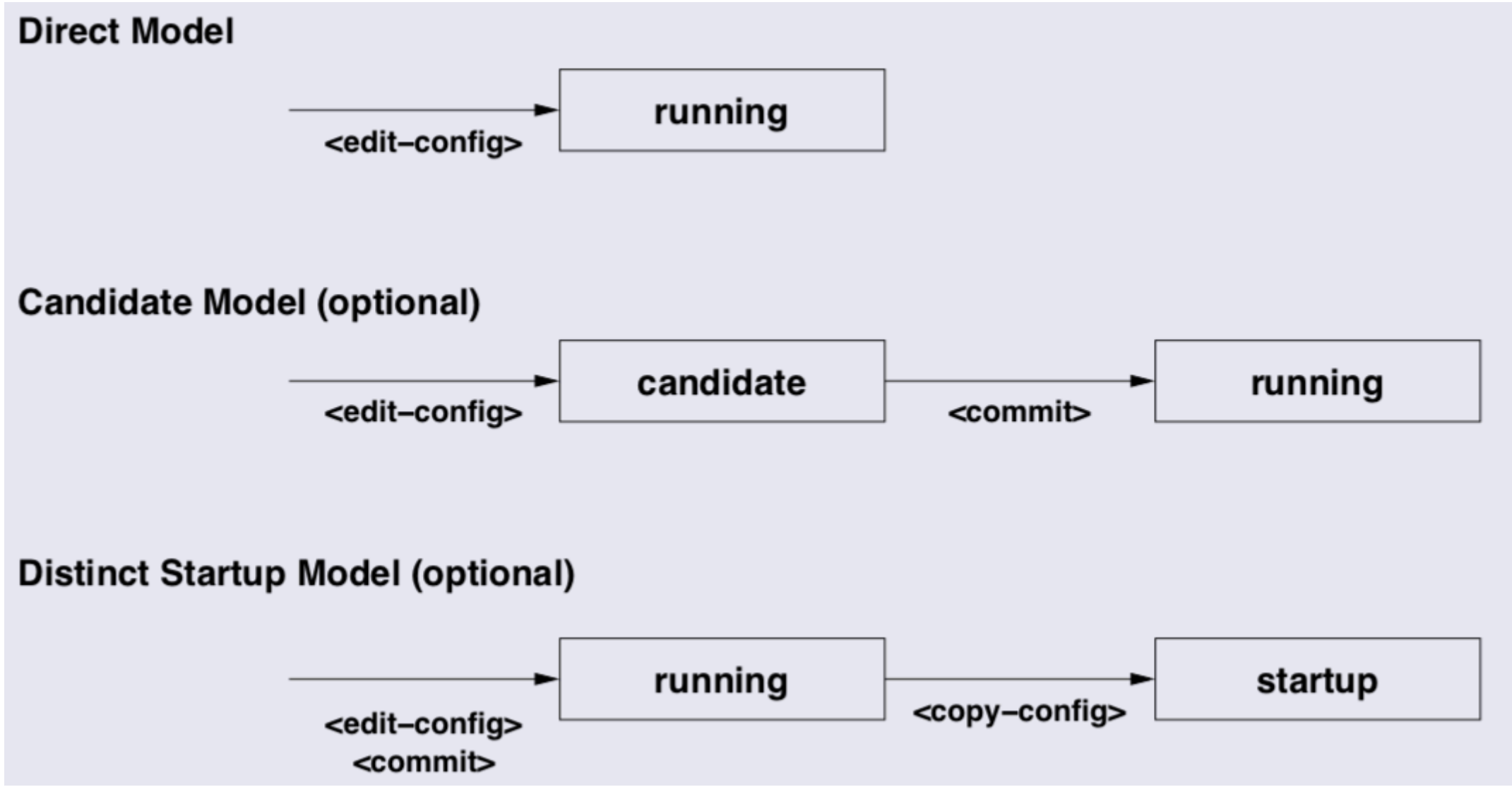
<startup>

- Configuration to be used during next startup

<candidate>

- A configuration that may become <running> through commit

Transaction Models



YANG RFC 6020 (2010)

- NETMOD IETF Working Group formed in April, 2008 to create a data modeling language for NETCONF
- "**Y**et **A**nother **N**ext **G**eneration" [modeling language]
- YANG models configuration and state data manipulated by the NETCONF remote procedure calls & notifications
- Designed to serialize data model instances into XML
- YANG 1.1 RFC 7950 (2016)

Who Makes YANG Models?

- Standards Orgs/Industry Forums:

- Vendors:

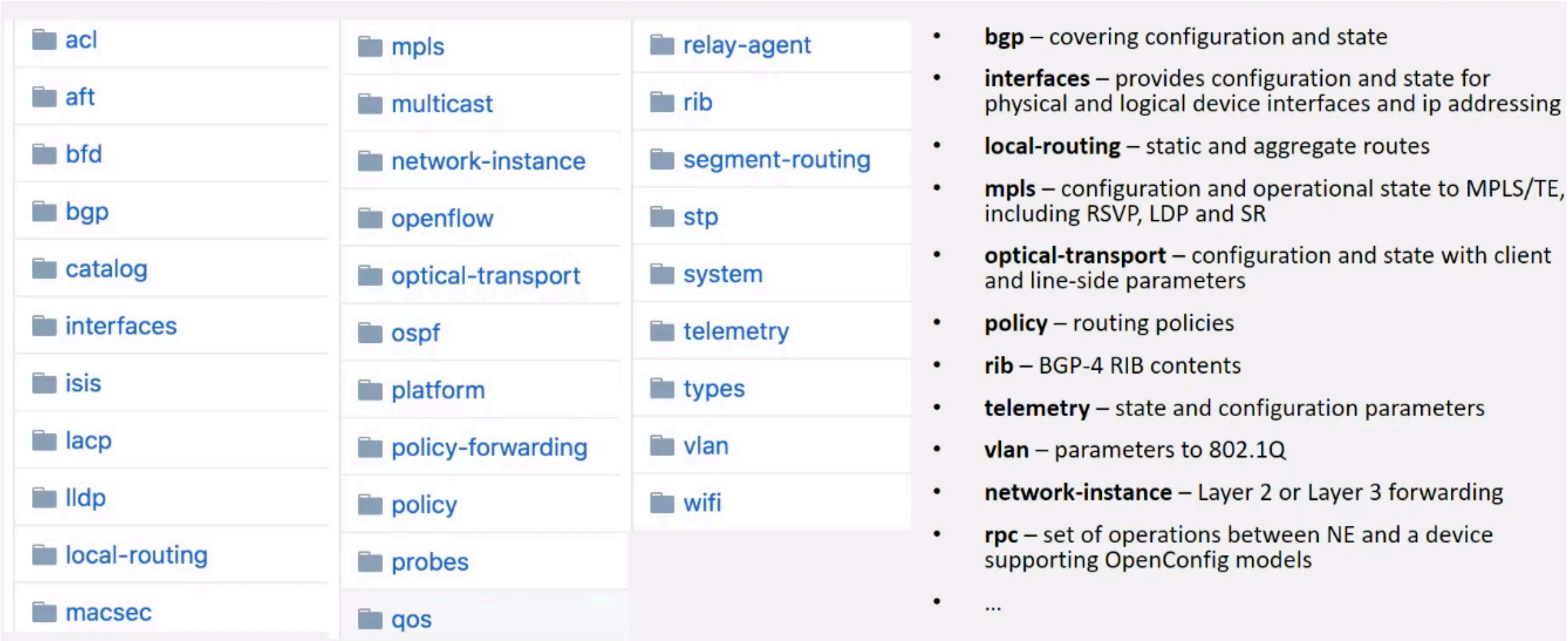
- OpenConfig (a “users group”)
  - Vendor-neutral data models based on network user requirements

Many IETF YANG Models for Networking

- RFC 7224: IANA Interface Type YANG Module
- RFC 8177: YANG Data Model for Key Chains
- RFC 8294: Common YANG Data Types for the Routing Area
- RFC 8299: YANG Data Model for L3VPN Service Delivery
- RFC 8346: A YANG Data Model for Layer 3 Topologies
- RFC 8347: A YANG Data Model for the Virtual Router Redundancy Protocol (VRRP)
- RFC 8348: A YANG Data Model for Hardware Management
- RFC 8431: A YANG Data Model for the Routing Information Base (RIB)
- RFC 8466: A YANG Data Model for Layer 2 Virtual Private Network (L2VPN) Service Delivery
- RFC 8512: A YANG Module for Network Address Translation (NAT) and Network Prefix Translation (NPT)
- RFC 8519: YANG Data Model for Network Access Control Lists (ACLs)
- RFC 8542: A YANG Data Model for Fabric Topology in Data-Center Networks
- RFC 8561: A YANG Data Model for Microwave Radio Link
- RFC 8575: YANG Data Model for the Precision Time Protocol (PTP)
- RFC 8632: A YANG Data Model for Alarm Management
- RFC 8652: A YANG Data Model for the Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD)
- RFC 8675: A YANG Data Model for Tunnel Interface Types
- RFC 8676: YANG Modules for IPv4-in-IPv6 Address plus Port (A +P) Softwires

OpenConfig Data Models

https://github.com/openconfig/public



- **bgp** – covering configuration and state
- **interfaces** – provides configuration and state for physical and logical device interfaces and ip addressing
- **local-routing** – static and aggregate routes
- **mpls** – configuration and operational state to MPLS/TE, including RSVP, LDP and SR
- **optical-transport** – configuration and state with client and line-side parameters
- **policy** – routing policies
- **rib** – BGP-4 RIB contents
- **telemetry** – state and configuration parameters
- **vlan** – parameters to 802.1Q
- **network-instance** – Layer 2 or Layer 3 forwarding
- **rpc** – set of operations between NE and a device supporting OpenConfig models
- …

From: BGP Analytics with OpenConfig Telemetry and gRPC: https://media.ccc.de/v/denog11-29-bgp-analytics-with-openconfig-telemetry-and-grpc

“hello” message Capability Exchange

Server advertisement of **capabilities**

```
S: <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
S:   <capabilities>
S:     <capability>
S:       urn:ietf:params:xml:ns:netconf:base:1.1
S:     </capability>
S:     <capability>
S:       urn:ietf:params:xml:ns:netconf:capability:startup:1.0
S:     </capability>
S:     <capability>
S:       urn:ietf:params:xml:ns:yang:ietf-interfaces?
S:         module=ietf-interfaces&amp;revision=2012-04-29
S:     </capability>
S:   </capabilities>
S:   <session-id>4<session-id>
S: </hello>
```

From Schönwälder, “Network Configuration Management with NETCONF and YANG" https://www.ietf.org/slides/slides-edu-netconf-yang-00.pdf

Remote Procedure Calls

```
C: <rpc message-id="101"
C:      xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
C:   <get-config>
C:     <source>
C:       <running/>
C:     </source>
C:   </get-config>
C: </rpc>

S: <rpc-reply message-id="101"
S:      xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
S:   <data><!-- ...contents here... --></data>
S: </rpc-reply>
```

RPC Errors with <rpc-error>

Validation of protocol

```
C: <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
C:   <get-config><source><running/></source></get-config>
C: </rpc>

S: <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
S:   <rpc-error>
S:     <error-type>rpc</error-type>
S:     <error-tag>missing-attribute</error-tag>
S:     <error-severity>error</error-severity>
S:     <error-info>
S:       <bad-attribute>message-id</bad-attribute>
S:       <bad-element>rpc</bad-element>
S:     </error-info>
S:   </rpc-error>
S: </rpc-reply>
```

NETCONF Operations

- get-config
- edit-config
  - Merge/replace/create/delete
- copy-config
- delete-config
- lock
- unlock
- validate
- commit
- cancel-commit

Editing (Patching) with <edit-config>

```
C:  <rpc message-id="101"
C:       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
C:    <edit-config>
C:      <target>
C:        <running/>
C:      </target>
C:      <config xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
C:        <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
C:          <interface nc:operation="replace">
C:            <name>Ethernet0/0</name>
C:            <mtu>1500</mtu>
C:          </interface>
C:        </interfaces>
C:      </config>
C:    </edit-config>
C:  </rpc>
```

Filtering Data Example

```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base: 1.0">
  <get>
    <filter xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      <interfaces>
        <interface>
          <name>eth0</name>
        </interface>
      </interfaces>
    </filter>
  </get>
</rpc>
```

```
<rpc-reply message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      <interface>
        <name>eth0</name>
        <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
          iana-ift:ethernetCsmacd
        </type>
        <enabled>true</enabled>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <address>
            <ip>2001:db8:c18:1::3</ip>
            <prefix-length>128</prefix-length>
          </address>
        </ipv6>
</interface> </interfaces> </data> </rpc-reply>
```

YANG Structure

Module: Self-contained tree of nodes. Modules are the smallest unit that can be “compiled” by YANG tools.

Container: A node with a set of children.

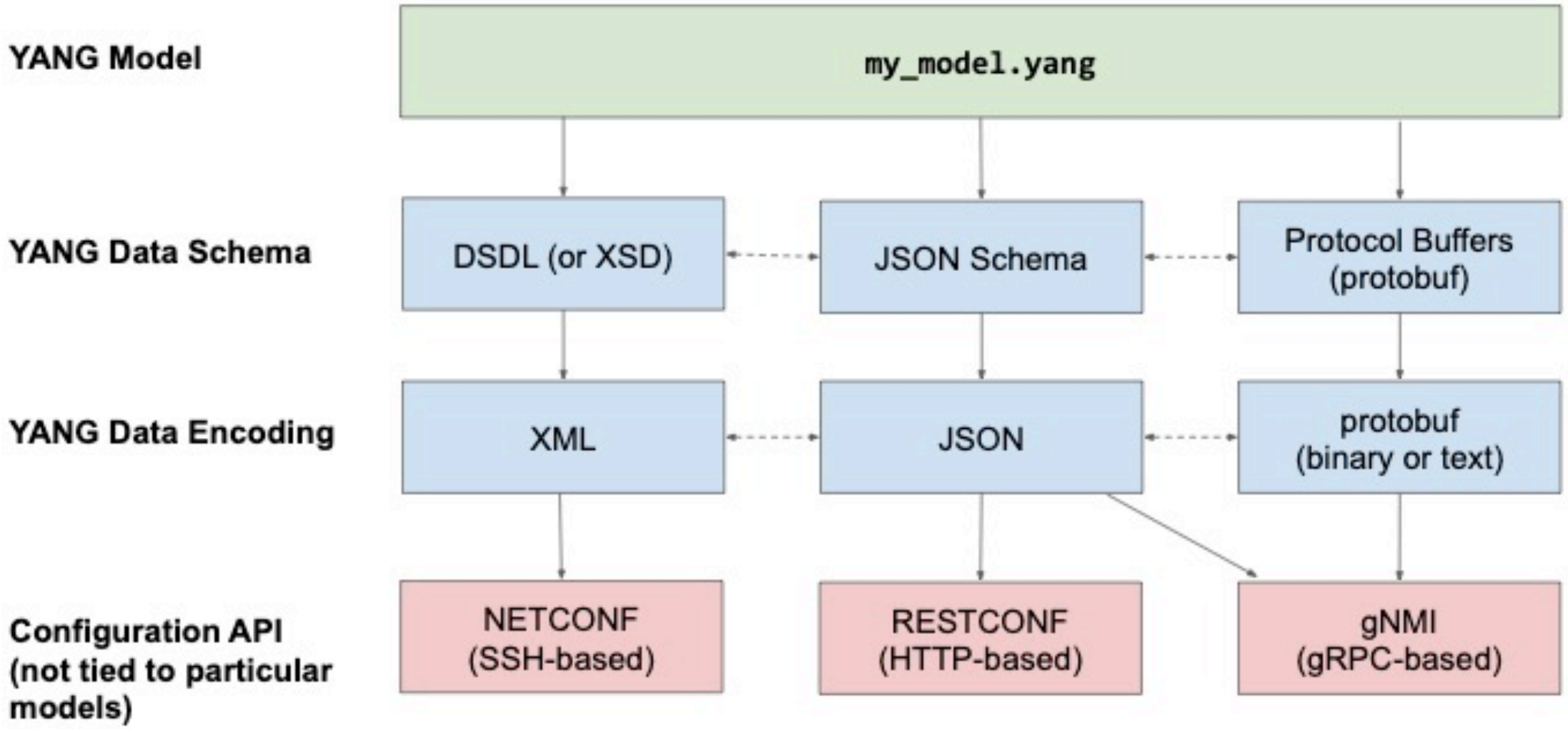
List: Node that contains a set of multiple children of the same type. Lists elements are identified by a key.

Leaf: Node contains one value of a particular data type and no child nodes.

```
module: ietf-interfaces
+--rw interfaces
|  +--rw interface* [name]
|     +--rw name                          string
|     +--rw description?                  string
|     +--rw type                          identityref
|     +--rw enabled?                      boolean
|     +--rw link-up-down-trap-enable?     enumeration
+--ro interfaces-state
   +--ro interface* [name]
      +--ro name                  string
      +--ro type                  identityref
      +--ro admin-status          enumeration
      +--ro oper-status           enumeration
      +--ro last-change?          yang:date-and-time
      +--ro if-index              int32
      +--ro phys-address?         yang:phys-address
```

YANG Tree Diagrams – RFC 8340

NETCONF / RESTCONF / gNMI all use YANG



From: https://www.opennetworking.org/wp-content/uploads/2019/10/NG-SDN-Tutorial-Session-2.pdf

Example YANG Module – “sdi-port” (1/3)

```
module sdi-port {
  yang-version 1;
  namespace "http://xxxx.tv/sdi-port";
  prefix sdi;

  import ietf-yang-types {
    prefix yang;
  }

  organization
    "xxxxx Working Group";
  contact
    "Editor: Thomas Edwards";
  description
    "This module contains a collection of YANG definitions
     for managing SDI ports";

  revision 2020-02-19 {
    description
      "Initial revision.";
    reference
      "XXXXX A YANG Model for SDI Ports";
  }

// typedefs
  typedef embedded_pairs {
    type uint8 {
      range "0..16";
    }
  }
  description
    "There can be up to 16 embedded audio pairs in SDI.";
  reference
    "SMPTE ST 299-1: 24-Bit Digital Audio Format for SMPTE 292
      Bit-Serial Interface
      SMPTE ST 299-2: Extension of the 24-Bit Digital Audio Format
      to 32 Channels for 3 Gb/s Bit-Serial Interfaces";
}
```

Example YANG Module – “sdi-port” (2/3)

```
// VIDEO_FORMAT identities

identity VIDEO_FORMAT {
  description
    "base type to specify SDI video format";
  reference
    "Rec. ITU-R BT.1120-8 and Rec. ITU-R BT.2020";
}

identity FORMAT_720p50 {
  base VIDEO_FORMAT;
  description
    "720 lines progressive @ 50 fps on SMPTE ST 292";
}

identity FORMAT_720p59.94 {
  base VIDEO_FORMAT;
  description
    "720 lines progressive @ 60/1.001 fps on SMPTE ST 292";
}
```

Identity: unique, abstract type

```
identity FORMAT_1080i50 {
  base VIDEO_FORMAT;
  description
    "1080 lines interlaced @ 50 fps on SMPTE ST 292";
}

identity FORMAT_1080i59.94 {
  base VIDEO_FORMAT;
  description
    "1080 lines interlaced @ 60/1.001 fps on SMPTE ST 292";
}

identity FORMAT_1080p23.98 {
  base VIDEO_FORMAT;
  description
    "1080 lines progressive @ 24/1.001 fps on SMPTE ST 292";
}

                                                    . . .
```

Example YANG Module – “sdi-port” (3/3)

```
// ports container

container ports {
  description
    "SDI port parameters.";
  list port {
    key "name";
    description
      "The list of SDI ports on this device.";
    leaf name {
      type string;
      description
        "The name of the SDI port.";
    }
    leaf description {
      type string;
      description
        "Textual description of the SDI port.";
    }
    leaf enabled {
      type boolean;
      mandatory true;
      description
        "Is SDI port is enabled?";
    }
    leaf signal-direction {
      type enumeration {
        enum input;
        enum output;
      }
      default "input";
    }
    leaf video-format {
      type identityref {
        base VIDEO_FORMAT;
      }
      description
        "Video and SDI format.";
    }
    leaf number-embedded-pairs {
      type embedded_pairs;
      default "0";
      description
        "# embedded audio pairs";
    }
    leaf CRC-errors {
      type yang:counter32;
      description
        "Count of SDI CRC errors.";
    }
  }
}
```

Pyang Tree View (RFC 8340)

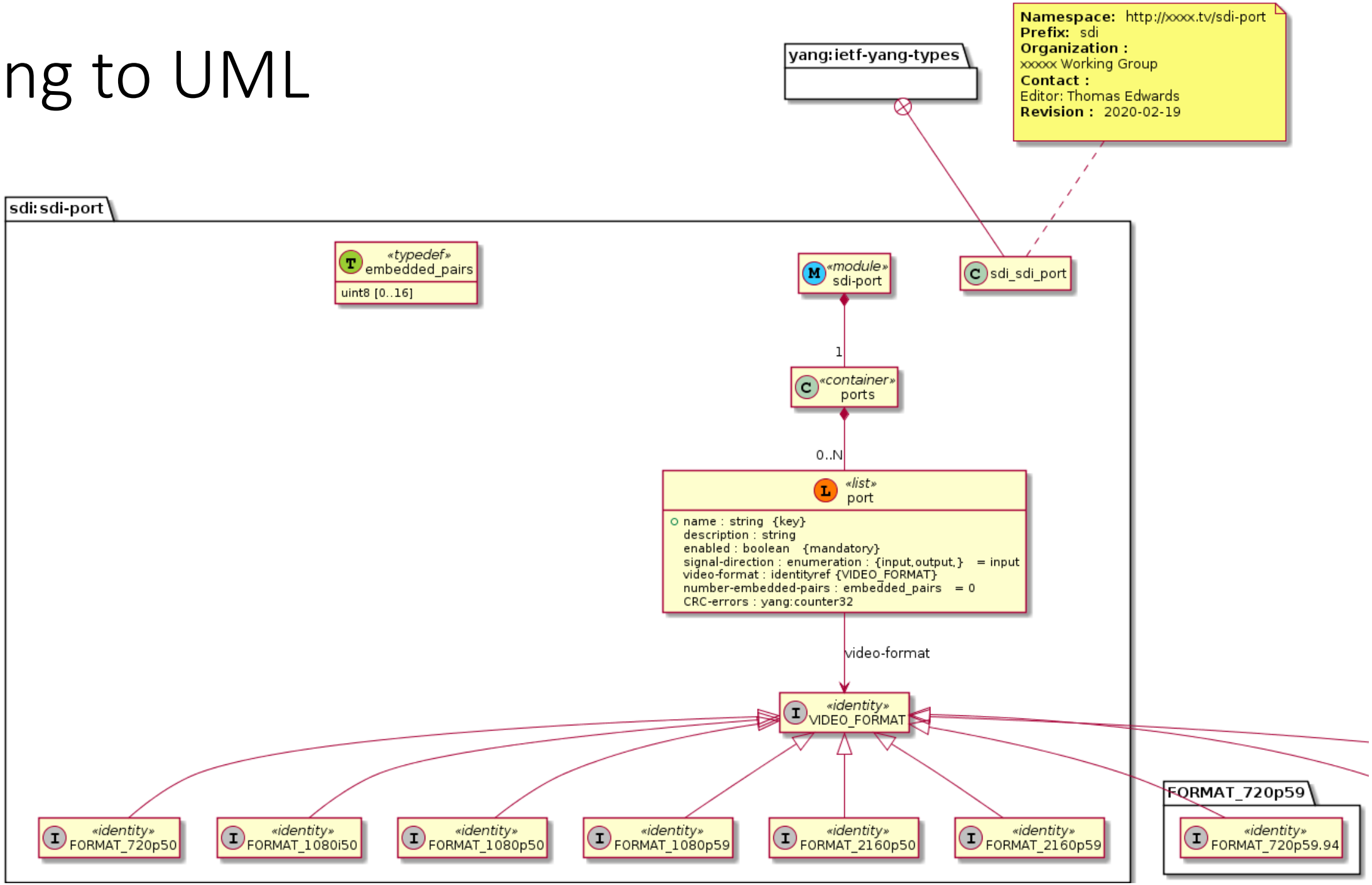
https://pypi.org/project/pyang/

YANG validation & format conversion

```
$ pyang -f tree sdi-port.yang

module: sdi-port
  +--rw ports
     +--rw port* [name]
        +--rw name                          string
        +--rw description?                  string
        +--rw enabled                       boolean
        +--rw signal-direction?             enumeration
        +--rw video-format?                 identityref
        +--rw number-embedded-pairs?        embedded_pairs
        +--rw CRC-errors?                   yang:counter32
```

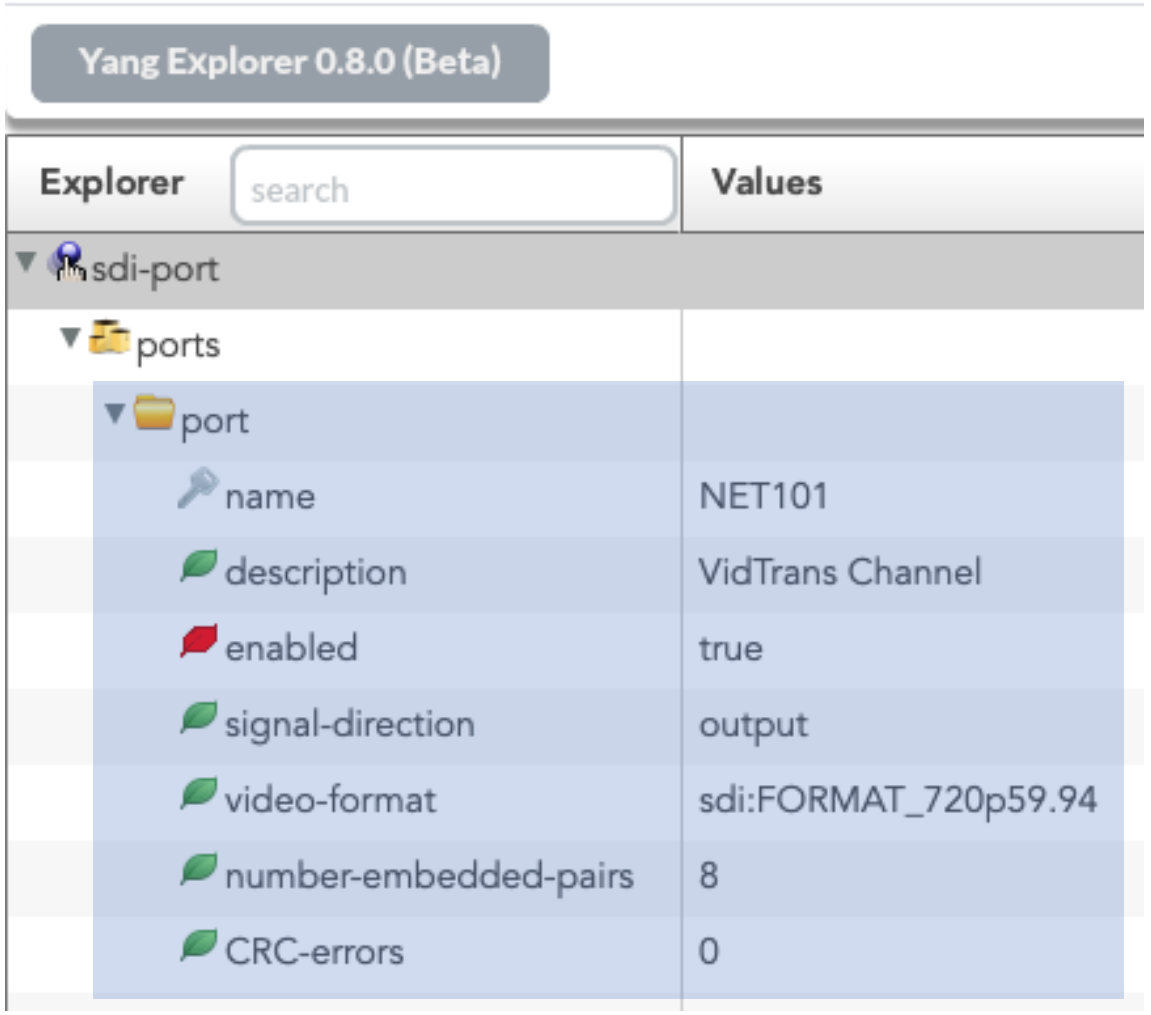
Pyang to UML


Namespace: http://xxxx.tv/sdi-port
Prefix: sdi
Organization :
xxxxx Working Group
Contact :
Editor: Thomas Edwards
Revision : 2020-02-19
yang:ietf-yang-types
sdi:sdi-port
«typedef»
embedded_pairs
uint8 [0..16]
«module»
sdi-port
sdi_sdi_port
1
«container»
ports
0..N
«list»
port
name : string {key}
description : string
enabled : boolean {mandatory}
signal-direction : enumeration : {input,output,} = input
video-format : identityref {VIDEO_FORMAT}
number-embedded-pairs : embedded_pairs = 0
CRC-errors : yang:counter32
video-format
«identity»
VIDEO_FORMAT
«identity»
FORMAT_720p50
«identity»
FORMAT_1080i50
«identity»
FORMAT_1080p50
«identity»
FORMAT_1080p59
«identity»
FORMAT_2160p50
«identity»
FORMAT_2160p59
FORMAT_720p59
«identity»
FORMAT_720p59.94

YANG Explorer

Yang Browser and RPC Builder Application

Property	Value
Name	video-format
Node Type	leaf
Data Type	identityref:VIDEO_FORMAT
Access	read-write
Presence	
Key	
Mandatory	
Default	
Path	sdi-port/ports/port/video-format
Description	Video format of the SDI portNone
XPath Filter	/sdi:ports/port/video-format



https://github.com/CiscoDevNet/yang-explorer

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <running/>
  </target>
  <config>
    <ports xmlns="http://xxxx.tv/sdi-port">
     <port>
      <name>NET101</name>
      <description>VidTrans Channel</description>
      <enabled>true</enabled>
      <signal-direction>output</signal-direction>
      <video-format>sdi:FORMAT_720p59.94</video-format>
      <number-embedded-pairs>8</number-embedded-pairs>
      <CRC-errors>0</CRC-errors>
     </port>
    </ports>
  </config>
 </edit-config>
</rpc>
```

NETCONF RPC XML Example from YANG Explorer

```
import lxml.etree as ET
from argparse import ArgumentParser
from ncclient import manager
from ncclient.operations import RPCError


payload = """
<config xmlns:xc=
"urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ports xmlns="http://xxxx.tv/sdi-port">
    <port>
      <name>NET101</name>
      <description>VidTrans Channel</description>
      <enabled>true</enabled>
      <signal-direction>output</signal-direction>
      <video-format xmlns=
"http://xxxx.tv/sdiport">sdi:FORMAT_720p59.94
</video-format>
      <number-embedded-pairs>8
</number-embedded-pairs>
      <CRC-errors>0</CRC-errors>
    </port>
  </ports>
</config>
"""
```

Python NETCONF from YANG Explorer

```
if __name__ == '__main__':

  …

    # connect to netconf agent
    with manager.connect(host=args.host,
                         port=args.port,
                         username=args.username,
                         password=args.password,
                         timeout=90,
                         hostkey_verify=False,
                         device_params={'name': 'csr'}) as m:


        # execute netconf operation
        try:
            response = m.edit_config(target='running',
config=payload).xml
            data = ET.fromstring(response)
        except RPCError as e:
            data = e._raw
```

Concluding thoughts…

- Programmatic media device configuration & monitoring is tough
- YANG is a widely used modeling language for expression of configuration and state
- YANG models are widely produced by networking SDOs, vendors, and user groups
- YANG tools are widely available
- NETCONF/RESTCONF/gNMI have strong traction to enable programmatic configuration & monitoring of network devices based on YANG models

Next Steps…

- Is this something the professional media industry should consider?
- How does NETCONF/YANG fit into AMWA NMOS?
  - ➢IS-05 Connection Management has some flow configuration
  - ➢IS-06 Network Control has items we could move into OpenConfig models
- Should professional media lean towards NETCONF, RESTCONF, or gNMI?
  - ➢gNMI seems to be getting traction for OpenConfig models

Let's Think About It! Thanks!