Data Model-Driven Management: Latest Industry and Tool Developments

Benoit Claise
Distinguished Engineer, Cisco
Operations and Management Area Director, IETF

Agenda

- Data Model-driven Management
- Industry Developments
- Yangcatalog.org
- Conclusion
- References

YANG – A Data Modeling Language

- Human readable and easy to learn
- Hierarchical configuration data models
- Reusable types and groupings (structured types)
- Extensibility through augmentation
- Formal constraints for configuration validation
- Data modularity through modules and sub-modules
- Well defined versioning rules

Why you should care:

YANG is a full, formal contract language with rich syntax and semantics to build applications on

```
list interface {
        key "name";
        unique "type location";

        leaf name {
          type string;
          reference
            "RFC 2863: The Interfaces Group MIB - ifName";
        }

        leaf description {
          type string;

...

  container statistics {
          config false;
          leaf discontinuity-time {
            type yang:date-and-time;
          }

          leaf in-octets {
            type yang:counter64;
            reference
              "RFC 2863: The Interfaces Group MIB - ifHCInOctets";
          }
```

Why Data Model-Driven Management?

- APIs derived from the data models:
  - Data models = definitions and constraints
  - The protocol: NETCONF, RESTCONF, GRPC
  - The encoding: JSON, XML, protobuf
  - The programming language: Python, Ruby, Java, C, Erlang, ...
- Industry focusing on YANG as *the* data modeling language for services and devices

Non Standard
Possible links

Data Modeling Language
(schema language)

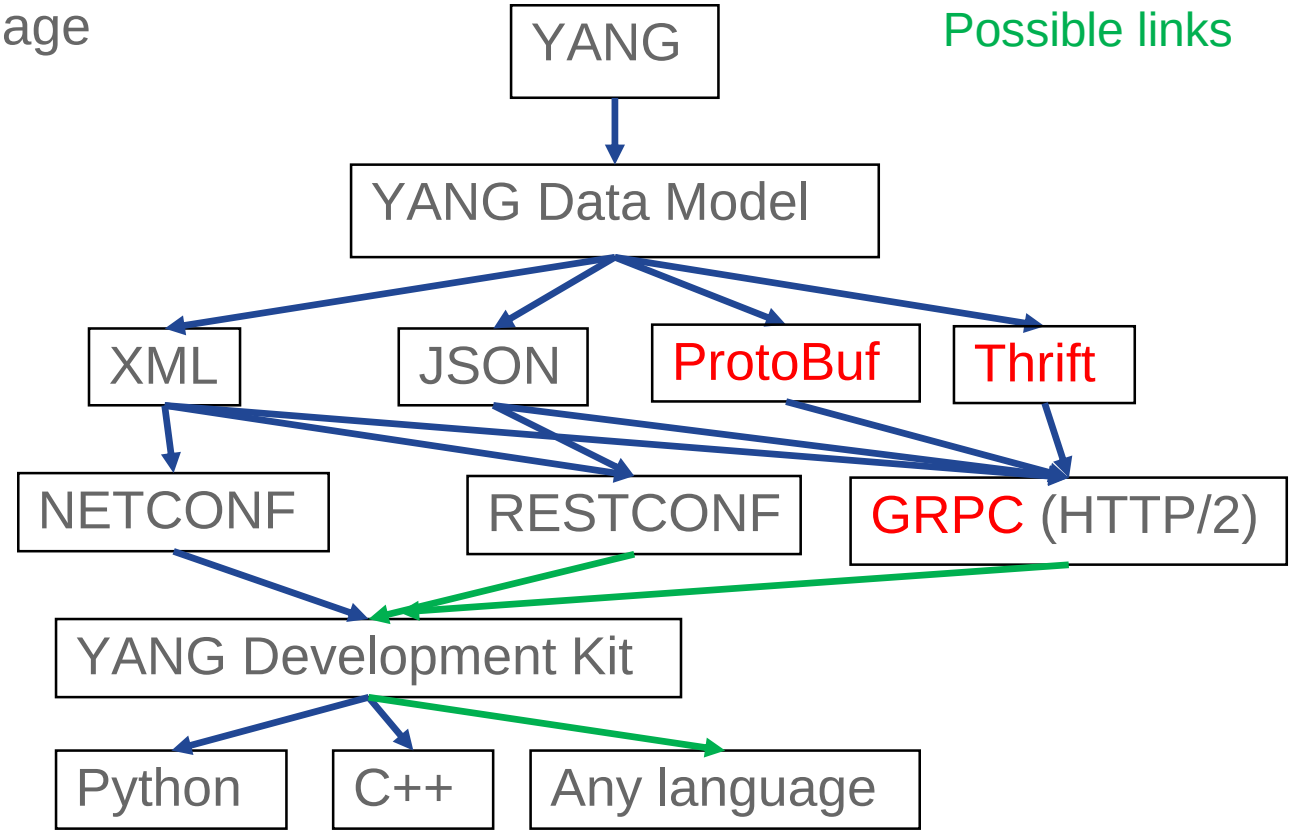
Data Modeling
(schema)

Encoding
(serialization)

Protocol

Application

Prog. Language



RESTCONF versus NETCONF: Summary

- RESTCONF: no notion of transaction
- RESTCONF: no notion of lock
- RESTCONF: no notion of candidate config and commit
- RESTCONF: so no notion of two phase commit
- RESTCONF: no <copy-config>
- RESTCONF: some more granularity for query => "config", "nonconfig“, "all".
- RESTCONF: XML or JSON (while NETCONF is XML only)

NETCONF might be better for router and switches
RESTCONF might be better for controller north-bound interface

Data Model-Driven Management: Example

Acting on resources

```
Module my-interfaces {
{
 namespace "com.my-interfaces";

   container interfaces {
     list interface {
       key name;
       leaf name { type string; }
       leaf admin-status { type enum;}

   rpc flap-interface {
     input {
       leaf name { type string; }
     }
     output {
       leaf result { type boolean; }
     }
}
```

GET : Gets a resource

GET /restconf/data/my-interfaces:interfaces

GET /restconf/data/my-interfaces:interfaces/interface/<*some name*>

POST : Creates a resource or invoke operation

POST /restconf/operations/my-interfaces:flap-interface
+ JSON/XML Form Data (including name)

Response will have JSON/XML result

PUT : Replaces a resource

PUT /restconf/data/my-interfaces:interfaces/interface/<some name> + JSON/XML Form Data (name, admin-status)

DELETE : Removes a resource

DELETE /restconf/data/my-interfaces:interfaces/interface/<some name>

Data Model-Driven Management

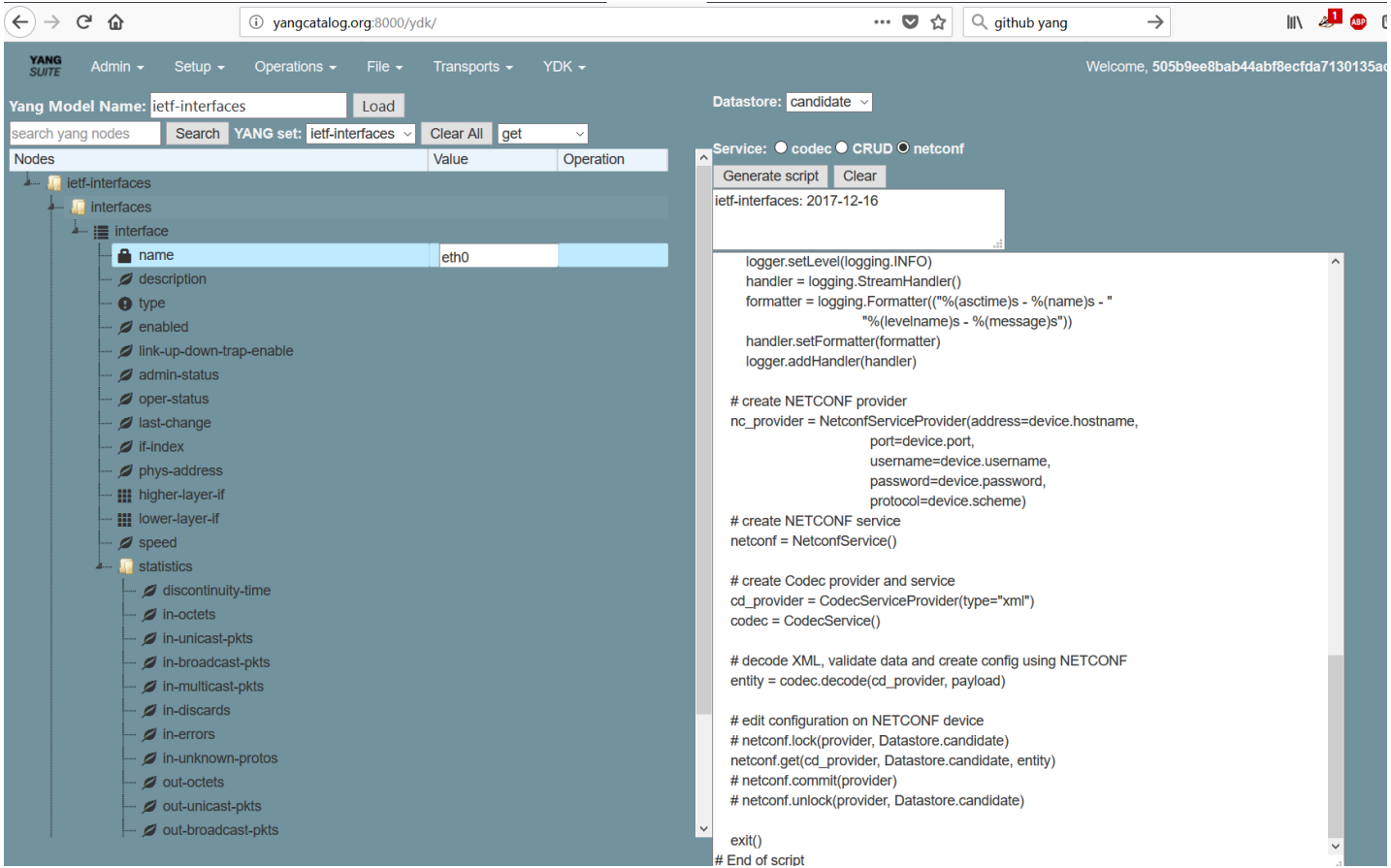
- Scripting: easy to create, hard to maintain/clean-up
  => Data model-driven set of APIs

Data Models = APIs

- However,

Automation is as good as your data models and your toolchains

Data Model-Driven Set of APIs



Why Should You Care?

« If a feature can’t be automated, it doesn’t exist »

« CLI is so 1990’s »

- Must be thinking of data models first
  - even before CLI, then deduce the CLI, the APIs, the documentation, the feature support
  - Testing through models, not only CLI

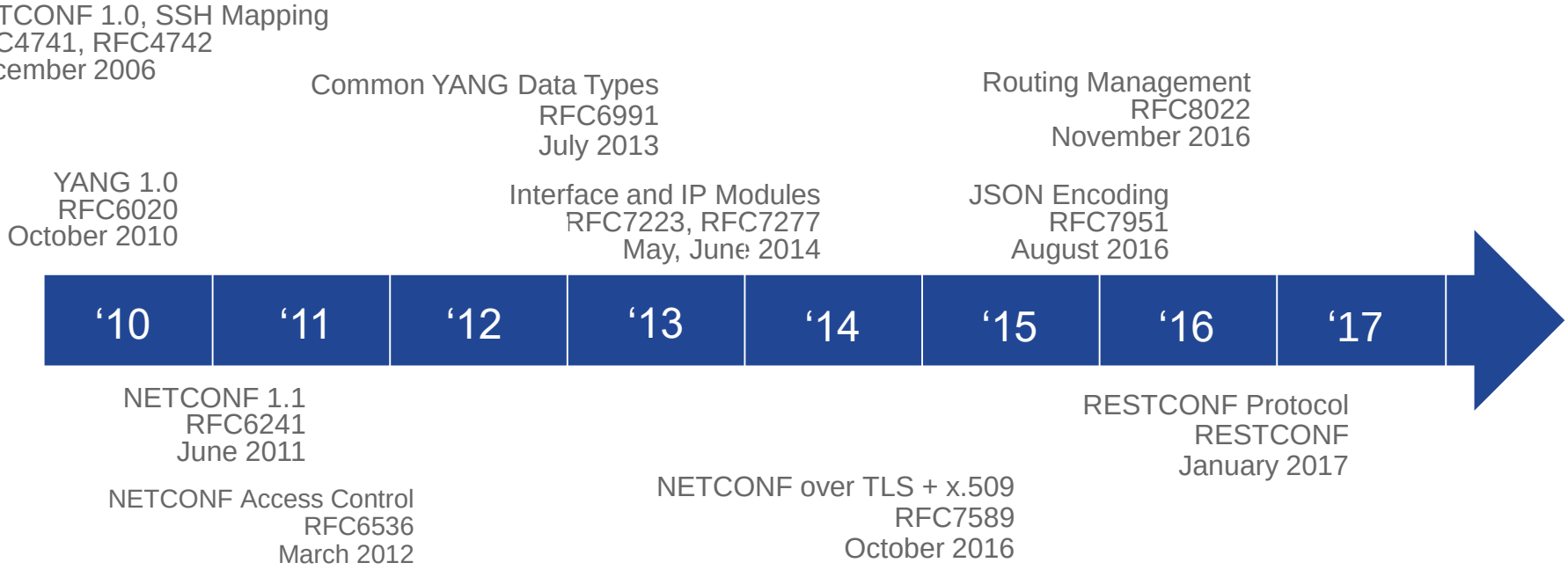
Agenda

- Data Model-driven Management
- Industry Developments
- Yangcatalog.org
- Conclusion
- References

Internet Engineering Task Force

- Open process for Internet Standards
  - Produce the RFCs
- The Standard Development Organization that specifies
  - NETCONF, RESTCONF, and YANG,
  - but also SNMP and MIBs
- Foresaw a « tsunami of YANG models »
- The IESG redistributed workload in order to allow for resources to be focused on YANG model coordination (Dec 2014)
  - “Primary oversight responsibility and coordination of this work across areas (AD document ownership) becomes the responsibility of Benoit Claise”

IETF: Timeline of Important Specifications



MIB Modules versus YANG Data Models

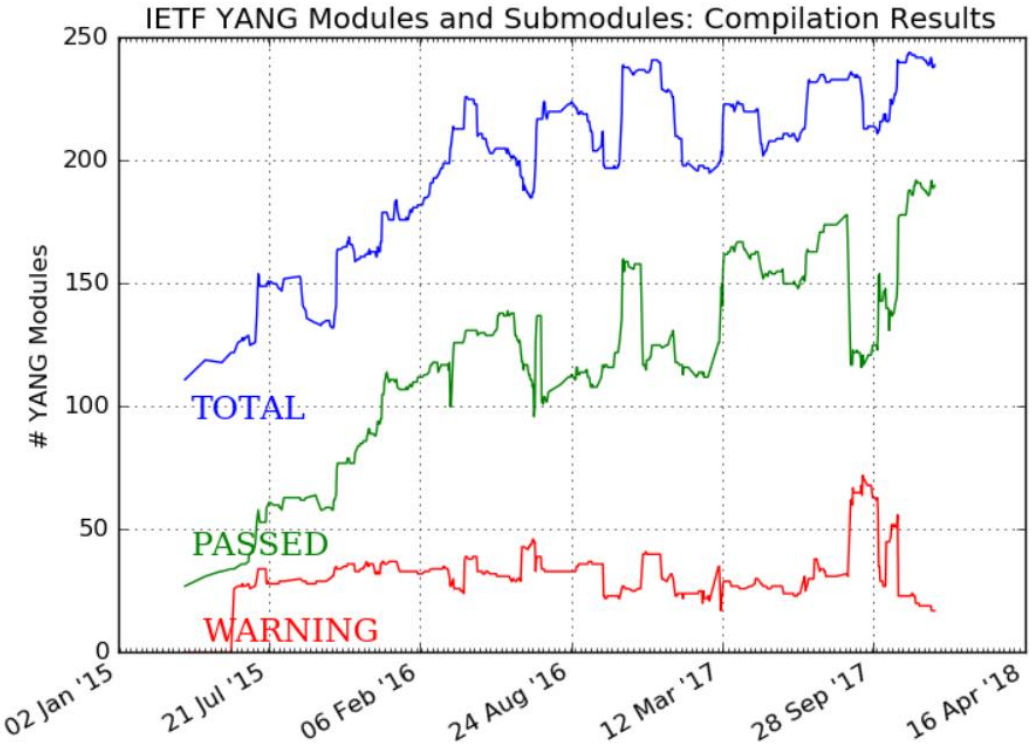
- Writable MIB Module IESG Statement (March 2014):

  "IETF working groups are therefore encouraged to use the NETCONF/YANG standards for configuration, especially in new charters"

  https://www.ietf.org/iesg/statement/writable-mib-module.html

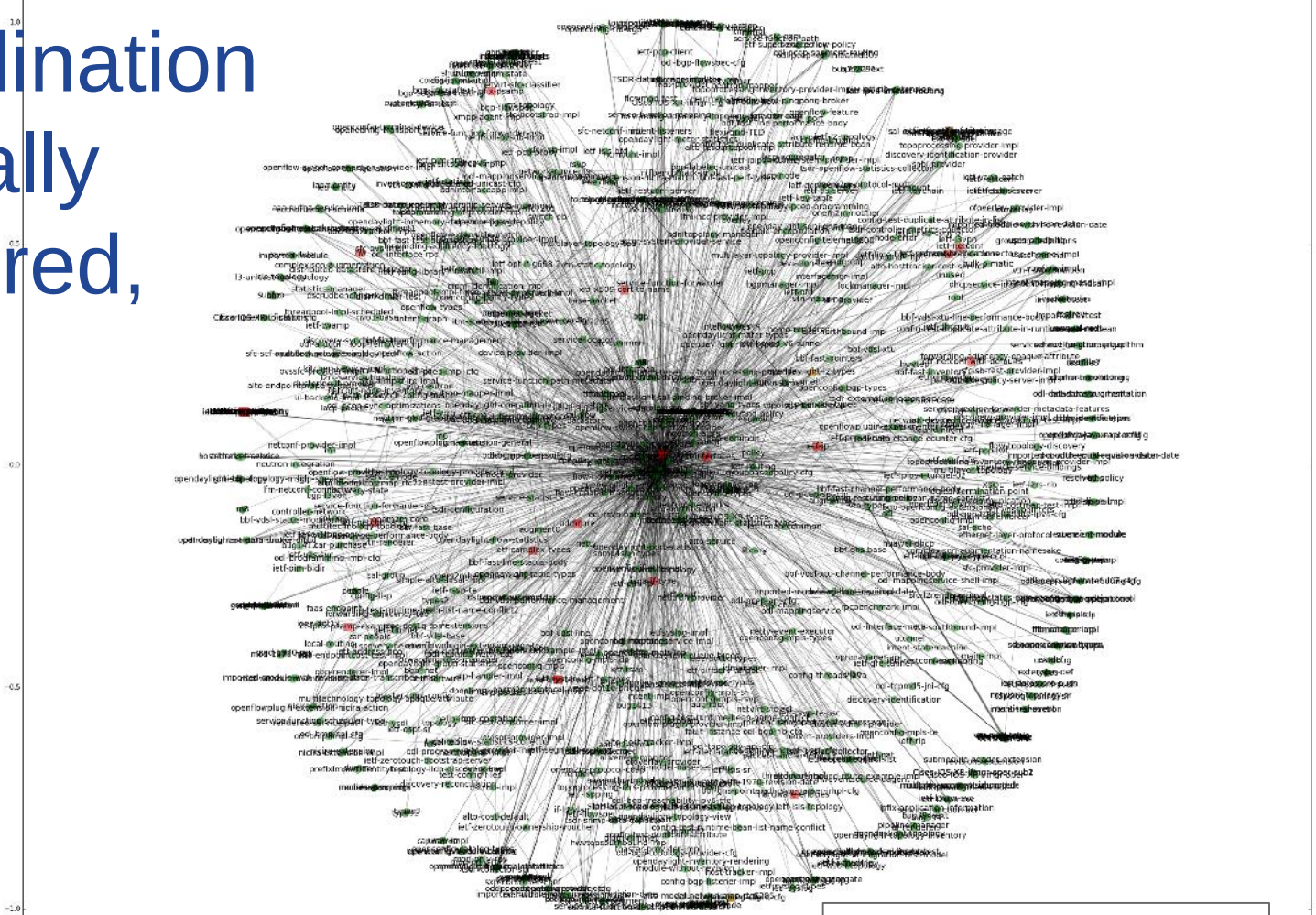
- RFC 6643: Translation of Structure of Management Information Version 2 (SMIv2) MIB Modules to YANG Modules.
- YANG data models for configuration and monitoring of new features
- Will SNMP disappear?
  - No: SNMP and MIB models do a good job for monitoring
  - SNMP MIBs are configuration and state information, but represented in a way that is unsuitable for configuration

IETF: YANG Models Growth



http://claise.be/IETFYANGPageCompilation.png

Coordination is Really Required, now!



YANG dependency graph

Coordination is Really Required, Now!

- Previous picture is about the IETF YANG models
  - New dimensions: different SDOs/Opensource projects
  - New dimension: versioning

- These YANG models must work together to create services
- Good problem to have: All YANG models arrive at the same time
  - As opposed to MIB modules in the past

- Standard Development Organizations (SDOs) can’t work in isolation: industry wide coordination is required

- Openconfig:
  - Pro: a few editors, for all YANG modules
  - Con: YANG modules change on regular basis

OPENCONFIG

- Operators-led YANG models
  - Google, AT&T, British Telecom, Microsoft, Facebook, Comcast, Verizon, Level3, Cox Communications, Yahoo!, Apple, Jive Communications, Deutsche Telekom / TeraStream, Bell Canada
- Focus: 123 network elements YANG models
  - Routing (BGP, ISIS, RIB, network-instance), routing policy, interfaces
  - Layer2 (vlan, spanning tree), ACL, optical transport, MPLS, etc.
- YANG models not aligned with the IETF
- Location: https://github.com/openconfig/public

www.openconfig.net

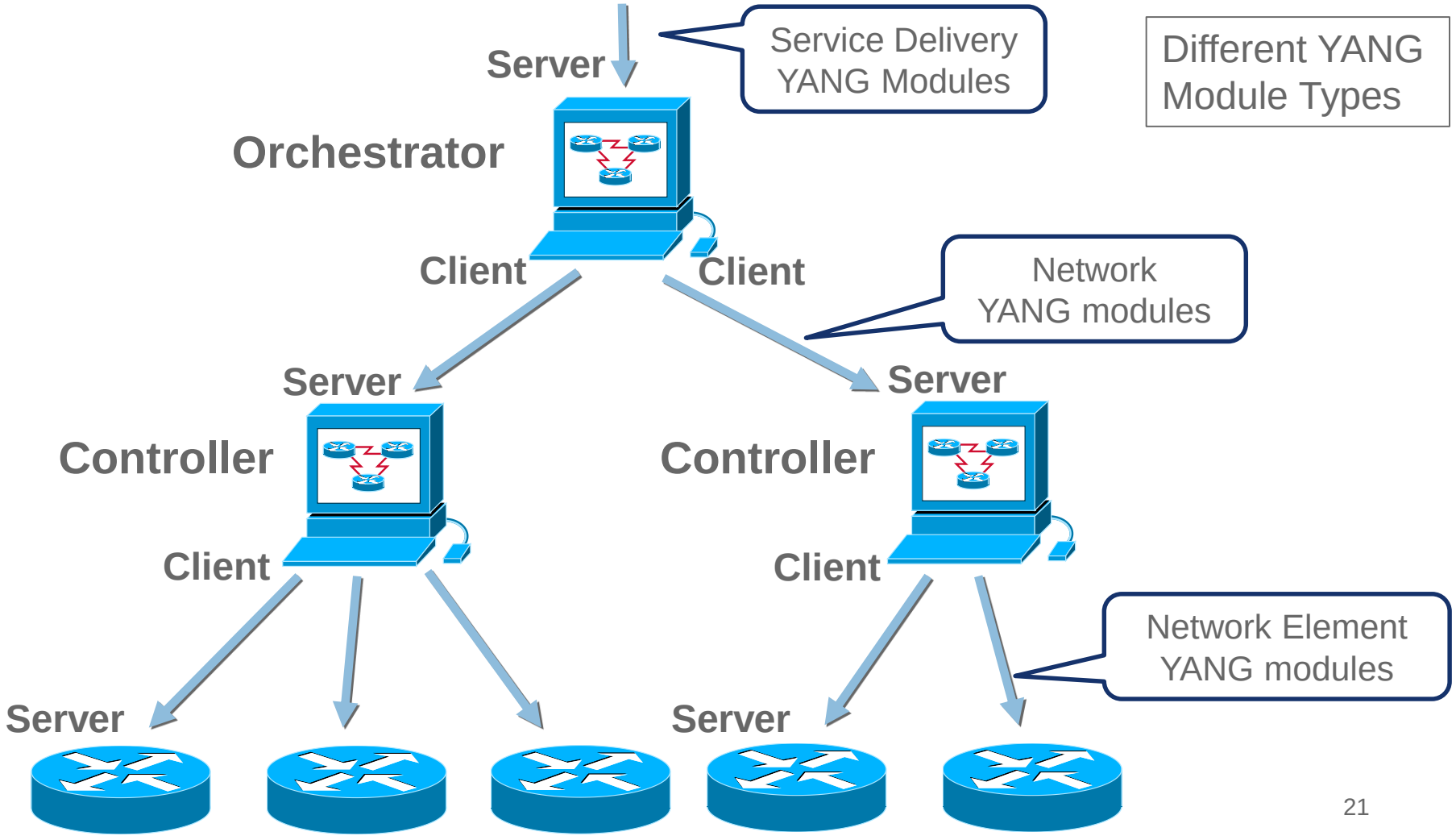
OPENCONFIG

- Streaming Telemetry specifications and configuration
- gRPC Network Management Interface (gNMI)
  - Protocol: gRPC
  - Encoding: protobuf
- Network management paradigm:
  - config without transaction,
  - then telemetry to check when applied

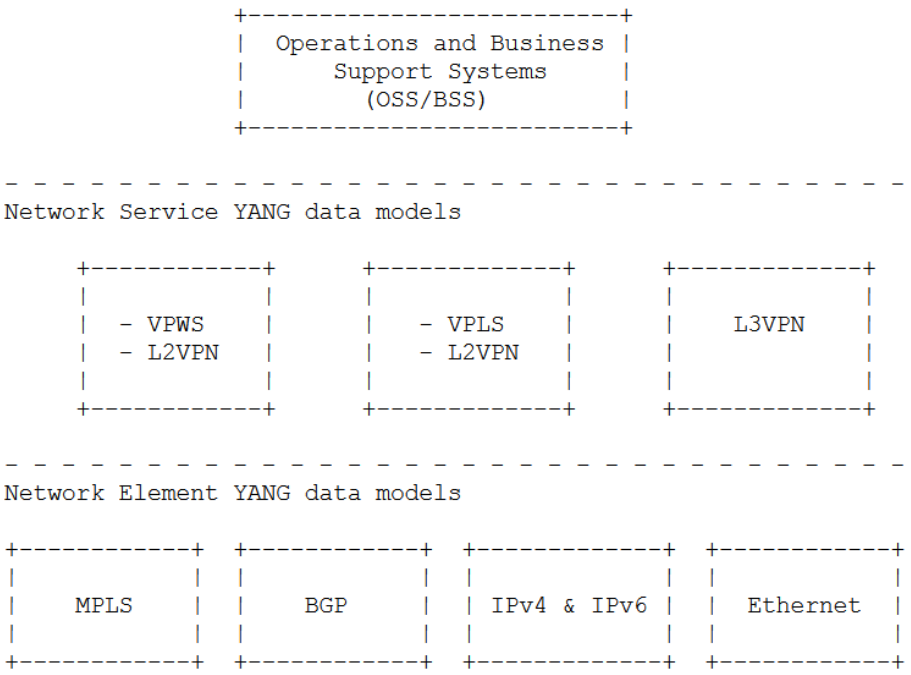
www.openconfig.net

TREND

YANG Tsunami in the Industry

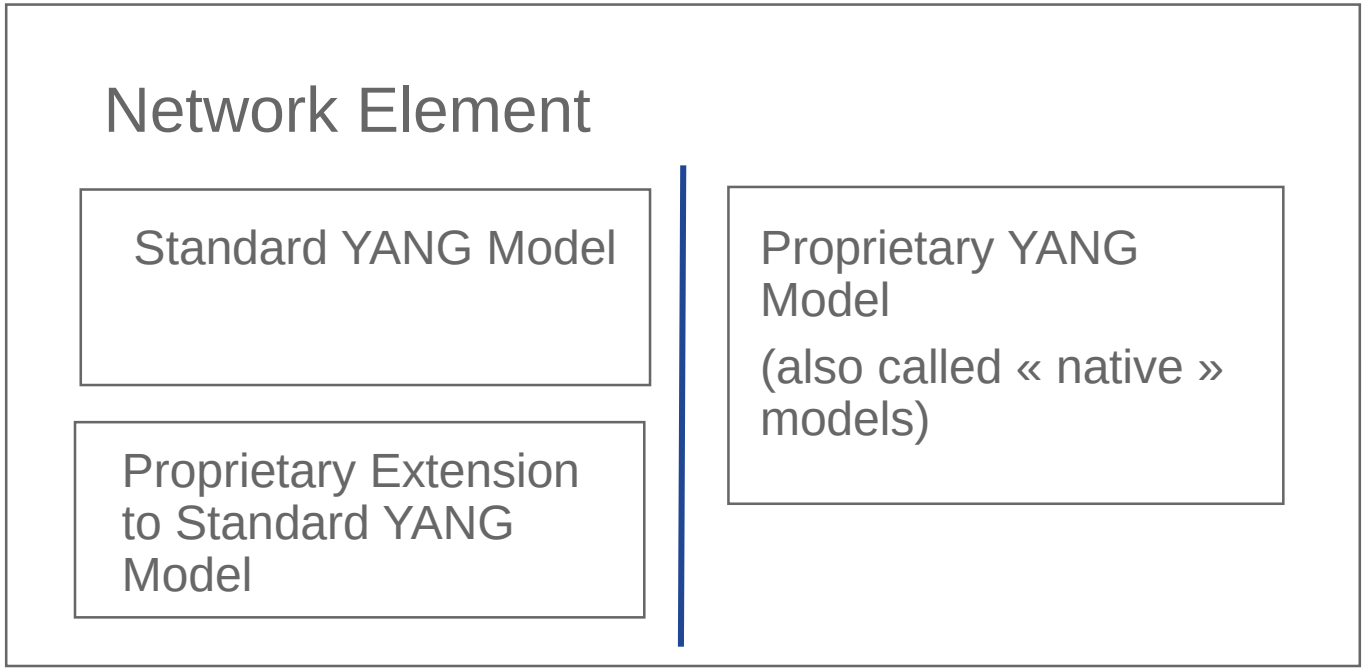

Server
Service Delivery
YANG Modules
Different YANG
Module Types
Orchestrator
Client
Client
Network
YANG modules
Server
Server
Controller
Controller
Client
Client
Network Element
YANG modules
Server
Server

SDOs Alignment and Trajectory



RFC 8199

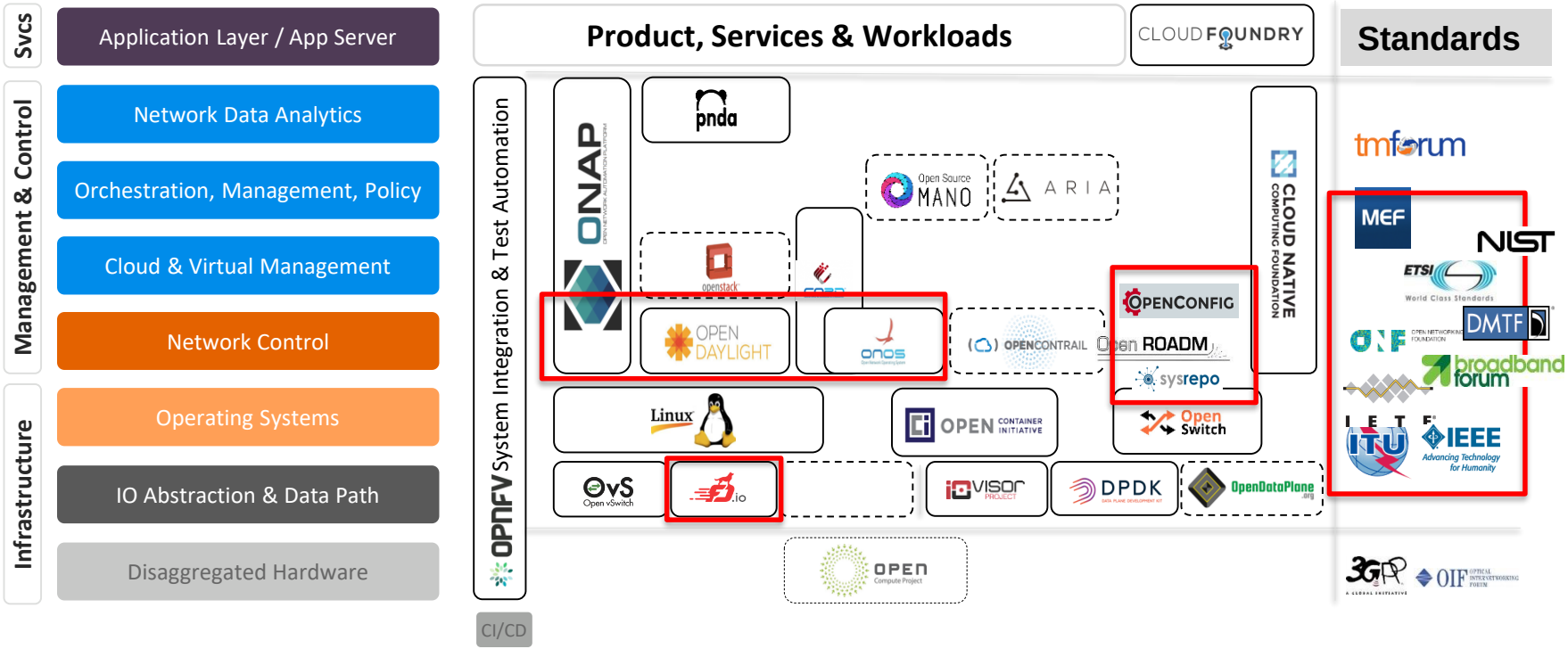
Data Model Location and Type (Network Element)



Open Source and SDOs Landscape

Linux Foundation Hosted

Outside Linux Foundation



Automation of Network + Infrastructure + Cloud + Apps + IOT

Numbers

- IETF YANG modules
  - Total from RFC: 50
  - Total in drafts: 237
- Openconfig
  - Total: 123
- Number of YANG data models in my VM
  - Total : 11510
  - Duplicates removed: 2591
  - Operational removed: 2423
  - Vendors removed: 1140

This becomes an industry problem!

How to Organize the Industry?

www.yangcatalog.org

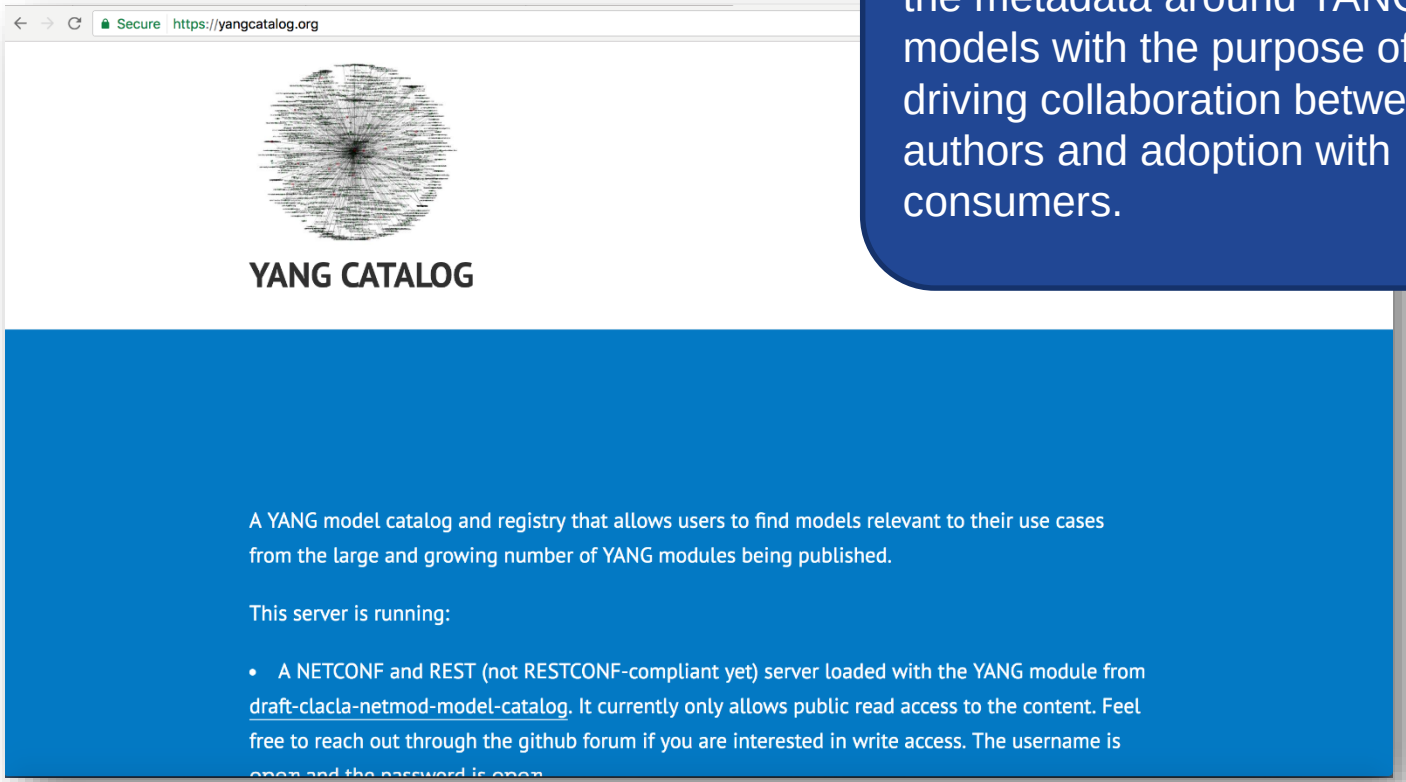
- With a YANG catalog, which contains all the modules
- The related metadata regarding maturity level, model type, implementation (which ones are important?), etc
- Based on the draft-clacla-netmod-model-catalog-02
- The inventory of all YANG modules, cross SDOs, cross vendors
  - SDOs on board: IETF, BBF, IEEE, ONF… some under discussion
  - Some vendors on board: Cisco, Huawei… some under discussion (Juniper)
  - Openconfig
- Started as IETF hackathons

Agenda

- Data Model-driven Management
- Industry Developments
- Yangcatalog.org
- Conclusion
- References

https://yangcatalog.org

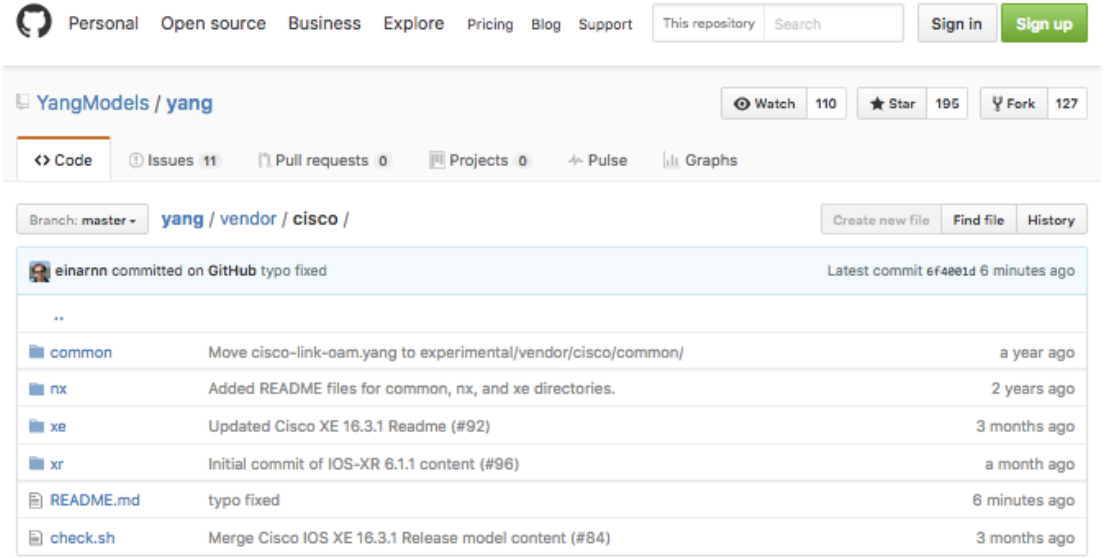
A repository of YANG tools and the metadata around YANG models with the purpose of driving collaboration between authors and adoption with consumers.



Programming the Networks

1. Understand the data model-driven management concept
2. Learn the NETCONF/RESTCONF/YANG basics
3. Where are the YANG data models?
4. Finding the right YANG data model
5. Experiment from a GUI and Code Generation
6. Testing
7. What about upgrading a device?
8. What’s missing? PubSub / Telemetry

3. Where are the Supported YANG Data Models?



YANG models for all platforms;

- **common** - across NX-OS, IOS-XE, and IOS-XR
- **nx** – NX-OS specific models
- **xe** - IOS-XE specific models
- **xr** - IOS-XR specific models

Each subdirectory has OS/platform-specific info in a README file

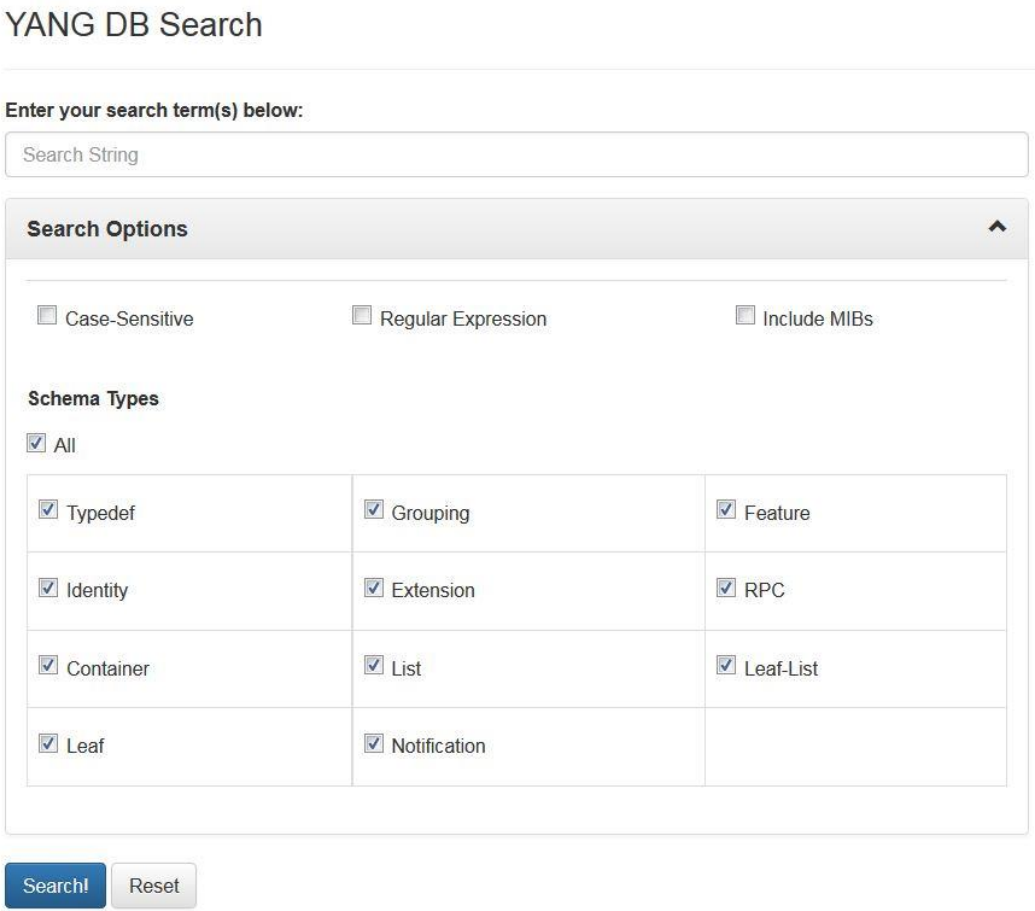
https://github.com/YangModels/yang/tree/master/vendor/cisco

3. Where are the Other YANG Data Models?

- IETF RFC:
  - https://github.com/YangModels/yang/tree/master/standard/ietf
  - http://www.claise.be/2018/01/ietf-yang-modules-statistiques/ (tar file)
- IETF drafts:
  - http://www.claise.be/2018/01/ietf-yang-modules-statistiques/ (tar file)
- Much statistic information on www.claise.be (daily cron job)
- And don’t forget the discovery capability
- Now all loaded in www.yangcatalog.org
  - See next slide

4. Finding the Right YANG Data Model

- http://www.yangcatalog.org
  - Busy including all the YANG modules from the industry
  - Contains a YANG DB search
- Allow one to find out what YANG modules and features are supported by a given platform, OS, license, etc.
- Demo: YANG search + YANG metadata + YANG tree



www.yangcatalog.org

YANG DB Search

Enter your search term(s) below:

vrf

Search Options

- [ ] Case-Sensitive
- [ ] Regular Expression
- [ ] Include MIBs

Schema Types

- [ ] All

[x] Typedef	[ ] Grouping	[ ] Feature
[x] Identity	[ ] Extension	[ ] RPC
[ ] Container	[ ] List	[ ] Leaf-List
[ ] Leaf	[ ] Notification	

Search! Reset

YANG DB Search Results for 'vrf'

Show 10 entries

Search: Entire Table

Name	Revision	Schema Type	Path	Module	Origin	Organization	Maturity	Description
active	2015-11-09	leaf	/ip-rip-oper:rip/ip-rip-oper:default-vrf/ip-rip-oper:global/ip-rip-oper:vrf-summary/ip-rip-oper:active	Cisco-IOS-XR-ip-rip-oper (Impact Analysis)	Vendor-Specific	cisco		VRF Active indicator
address-families	2017-03-12	container	/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/ldp:peers/ldp:peer/ldp:state/ldp:address-families	ietf-mpls-ldp (Impact Analysis)	Industry Standard	ietf	WG DRAFT	Per-vrf per-af params.
address-families	2017-03-12	container	/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/ldp:peers/ldp:peer/ldp:config/ldp-ext:address-families	ietf-mpls-ldp-extended (Impact Analysis)	Industry Standard	ietf	WG DRAFT	Per-vrf per-af params.
address-family-vrf-grouping	2017-02-07	grouping	/ios-rip:address-family-vrf-grouping	Cisco-IOS-XE-rip	Vendor-Specific	cisco		
af-ipv4-uc-and-vrf-cmds	2010-11-29	grouping	/bgp:af-ipv4-uc-and-vrf-cmds	brocade-bgp (Impact Analysis)	Vendor-Specific	brocade.com		
all-inclusive	2016-09-14	container	/ospf-act:act-ospf-instance-vrf/ospf-act:input/ospf-act:instance/ospf-act:all-inclusive	Cisco-IOS-XR-ipv4-ospf-act (Impact Analysis)	Vendor-Specific	cisco		Clear all non-default and default OSPF VRFs

4. Finding the Right YANG Data Model

Metadata are Important => Notion of Health Metric

- Organization: contact, maturity level
- Module: name, prefix, version, type, category, dependencies, document uri, submodules
- Implementation: status, platform, software release, opensource, contact
- And new one all the time. Ex: tree structure, generated from MIB, expired

Automation is as good as your data models, their metadata, and your toolchain

POST https://yangcatalog.org:8443/search-filter Parar

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
{
    "input": {
    "organization": "openconfig",
    "implementations": {
        "implementation" :[
        {
            "vendor": "cisco",
            "software-version": "6.1.3",
            "platform": "ASR9K"
        }]
      }
    }
}
```

Body Cookies Headers (4) Test Results Status: 20

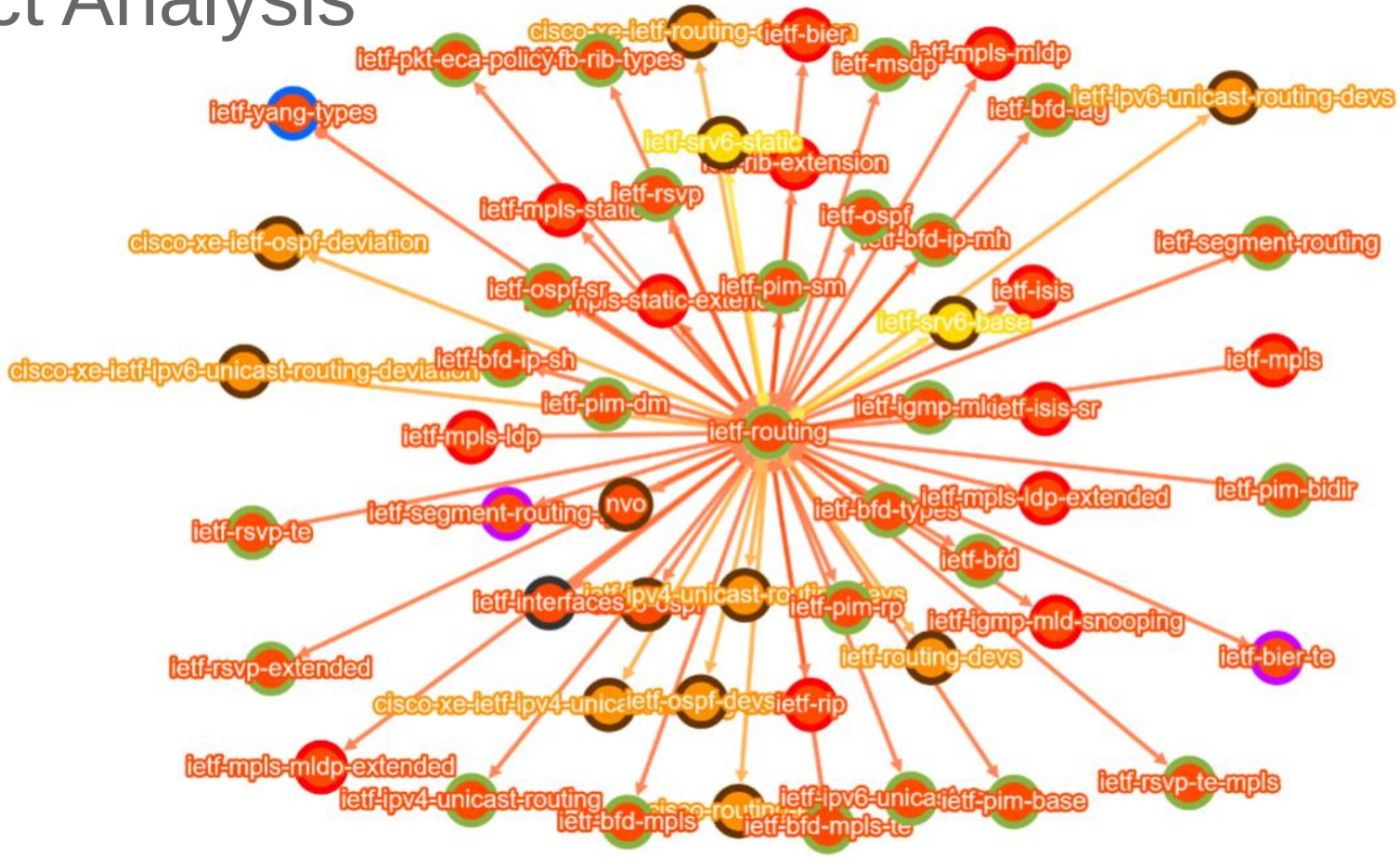
Pretty Raw Preview JSON


```
{
    "yang-catalog:modules": {
        "module": [
            {
                "name": "openconfig-acl",
                "revision": "2016-08-08",
                "organization": "openconfig",
                "namespace": "http://openconfig.net/yang/acl",
                "schema": "https://raw.githubusercontent.com/YangModels/yang/master/vendor/cisco/xe/1661/openconfig-acl.yang",
                "generated-from": "not-applicable",
                "module-classification": "unknown",
                "compilation-status": "failed",
                "compilation-result": "https://yangcatalog.org/results/openconfig-acl@2016-08-08_openconfig.html"
```

Impact Analysis

- Demo
- https://www.yangcatalog.org/yang-search/impact_analysis.php?modules[]=ietf-tls-client&modules[]=ietf-tls-server&modules[]=ietf-ssh-client&modules[]=ietf-ssh-server&modules[]=ietf-restconf-client&modules[]=ietf-restconf-server&modules[]=ietf-keystore&modules[]=ietf-netconf-client&modules[]=ietf-netconf-server&orgs[]=ietf&recurse=&rfcs=1
- https://www.yangcatalog.org/yang-search/impact_analysis.php?modules[]=ietf-interfaces&recurse=0&rfcs=1&show_subm=1&show_dir=both

Impact Analysis



Impact Analysis

YANG Impact Graph for Module(s): ietf-routing

Graph Options

Click on legend elements below to toggle highlighting on the graph.

Highlight All

Element Colors

- IETF
- CISCO

Rim Colors

- Maturity: ADOPTED
- Maturity: INITIAL
- FAILED
- Bottleneck to Ratification
- Maturity: N/A
- Maturity: COMPILATION
- Maturity: RATIFIED

Modules: ietf-routing x

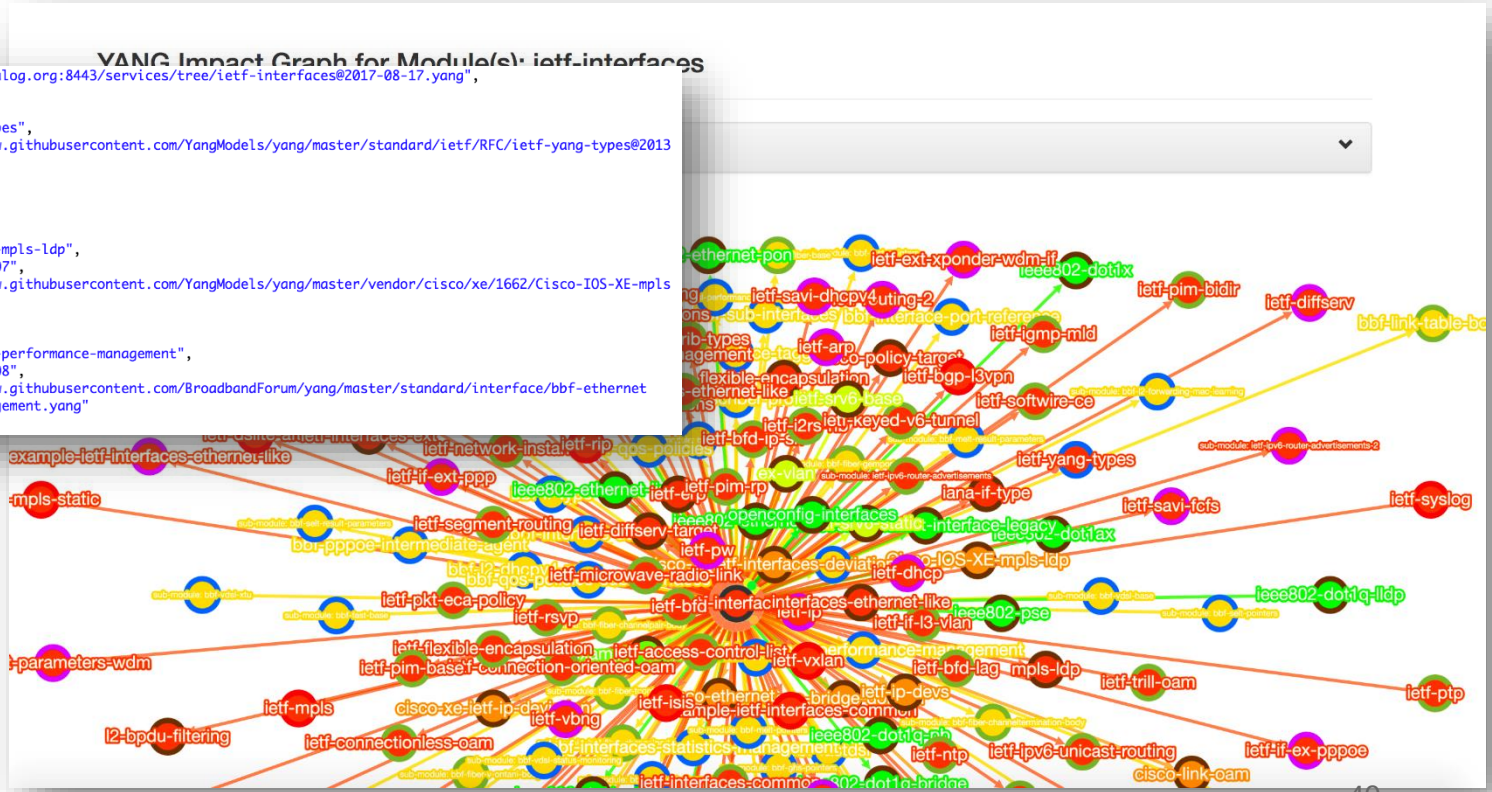
Orgs:

Recursion Levels: 0 **Include Ratified Standards?** ☑ **Include Sub-modules?** ☑ **Show Graph Direction:** Both

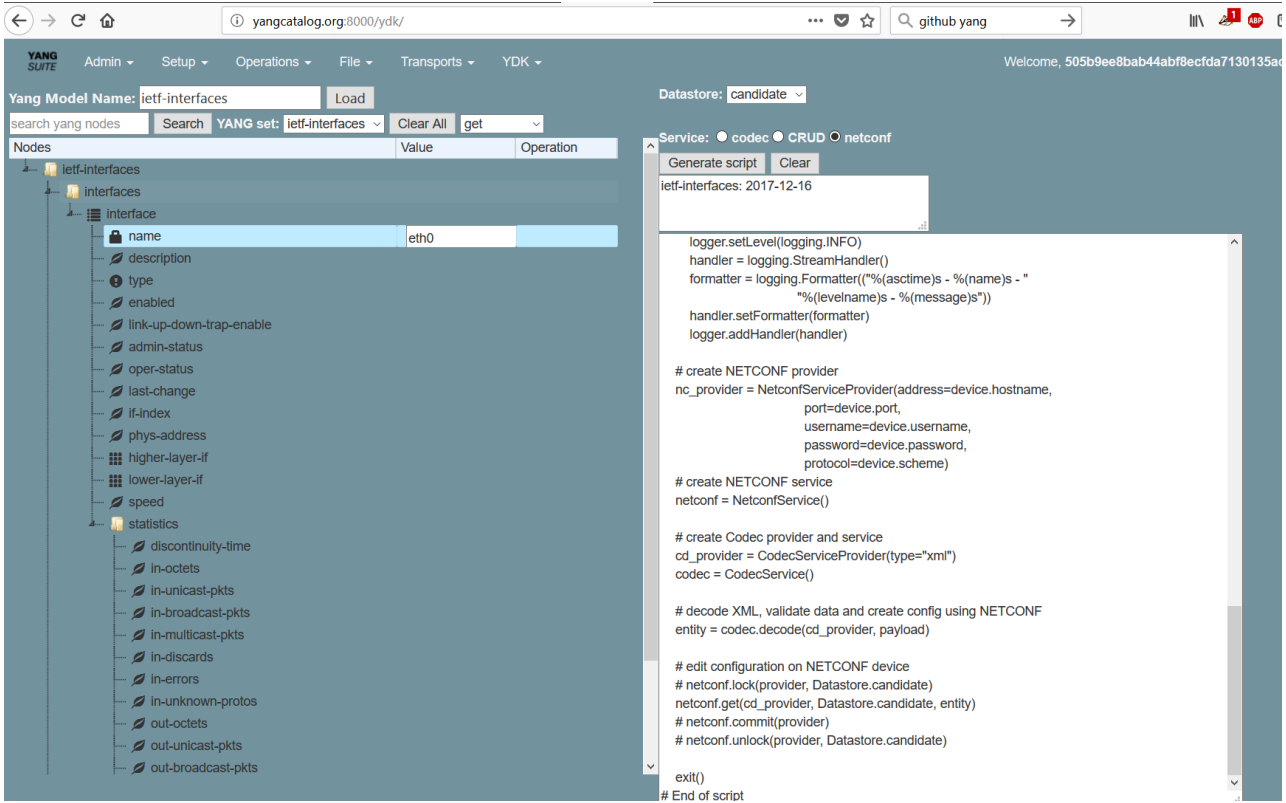
Generate Export

Tracking Dependencies and Dependents


```
"yang-tree": "https://yangcatalog.org:8443/services/tree/ietf-interfaces@2017-08-17.yang",
"dependencies": [
    {
        "name": "ietf-yang-types",
        "schema": "https://raw.githubusercontent.com/YangModels/yang/master/standard/ietf/RFC/ietf-yang-types@2013
            -07-15.yang"
    }
],
"dependents": [
    {
        "name": "Cisco-IOS-XE-mpls-ldp",
        "revision": "2017-02-07",
        "schema": "https://raw.githubusercontent.com/YangModels/yang/master/vendor/cisco/xe/1662/Cisco-IOS-XE-mpls
            -ldp.yang"
    },
    {
        "name": "bbf-ethernet-performance-management",
        "revision": "2017-05-08",
        "schema": "https://raw.githubusercontent.com/BroadbandForum/yang/master/standard/interface/bbf-ethernet
            -performance-management.yang"
    },
```

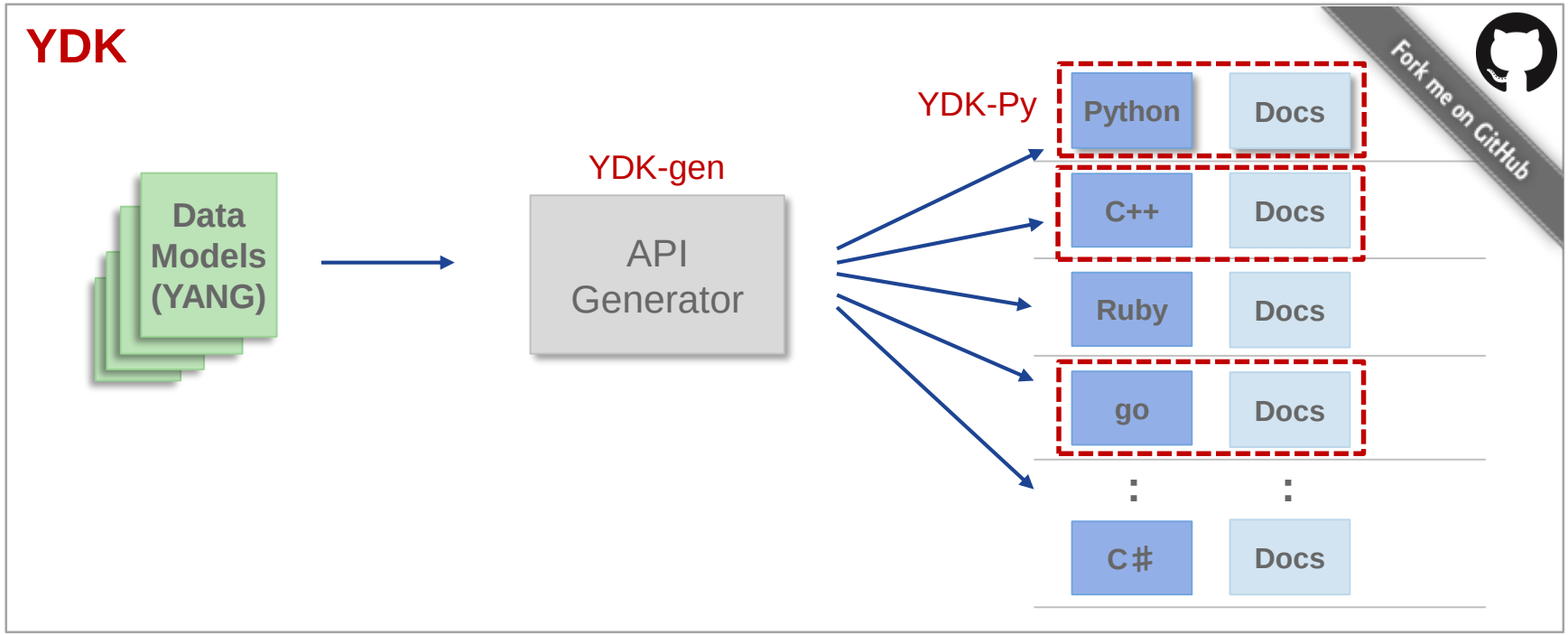
5. Experiment from a GUI and Code Generation



Demo:

- YANG Suite
- YDK

5. Generation of Model-Driven APIs Using YANG Development Kit (YDK)



7. What About Upgrading?

Semantic Versioning: Tracking Module Changes

- YANG modules backward compatibility
  - What if I upgrade my router? Will my automation/automated service break?
- Note: the native models might not be backward compatible…
- Demo: check-semantic-version API

7. What About Upgrading?

Semantic Versioning: Tracking Module Changes

ietf-interfaces@2013-12-23

```
        }
      ],
      "derived-semantic-version": "1.0.0",
      "implementations": {
        "implementation": [
```

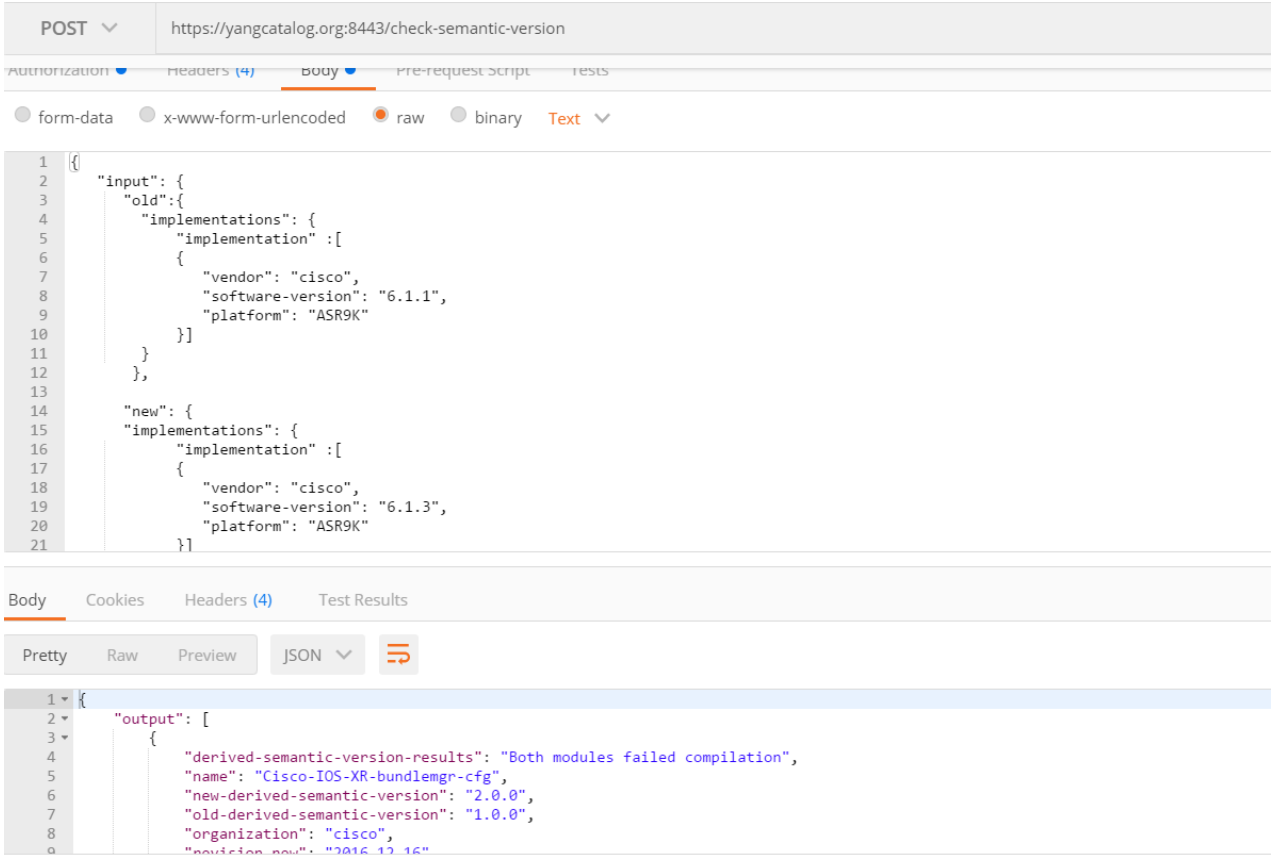
ietf-interfaces@2017-08-17

```
        }
      ],
      "derived-semantic-version": "2.0.0",
      "implementations": {
        "implementation": [
          {
```

- Derived-semantic-version is determined using:
  1. Order all modules of the same name by revision from oldest to newest.
  2. If module A, revision N+1 has failed compilation, bump its derived semantic MAJOR version.
  3. Else, run "pyang --check-update-from" on module A, revision N and revision N+1 to see if backward-incompatible changes exist.
  4. If backward-incompatible changes exist, bump module A, revision N+1's derived MAJOR semantic version.
  5. If no backward-incompatible changes exist, compare the pyang trees of module A, revision N and revision N+1.
  6. If there are structural differences (e.g., new nodes), bump module A, revision N+1's derived MINOR semantic version.
  7. If no structural differences exist, bump module A, revision N+1's derived PATCH semantic version.

7. What About Upgrading the IOS?

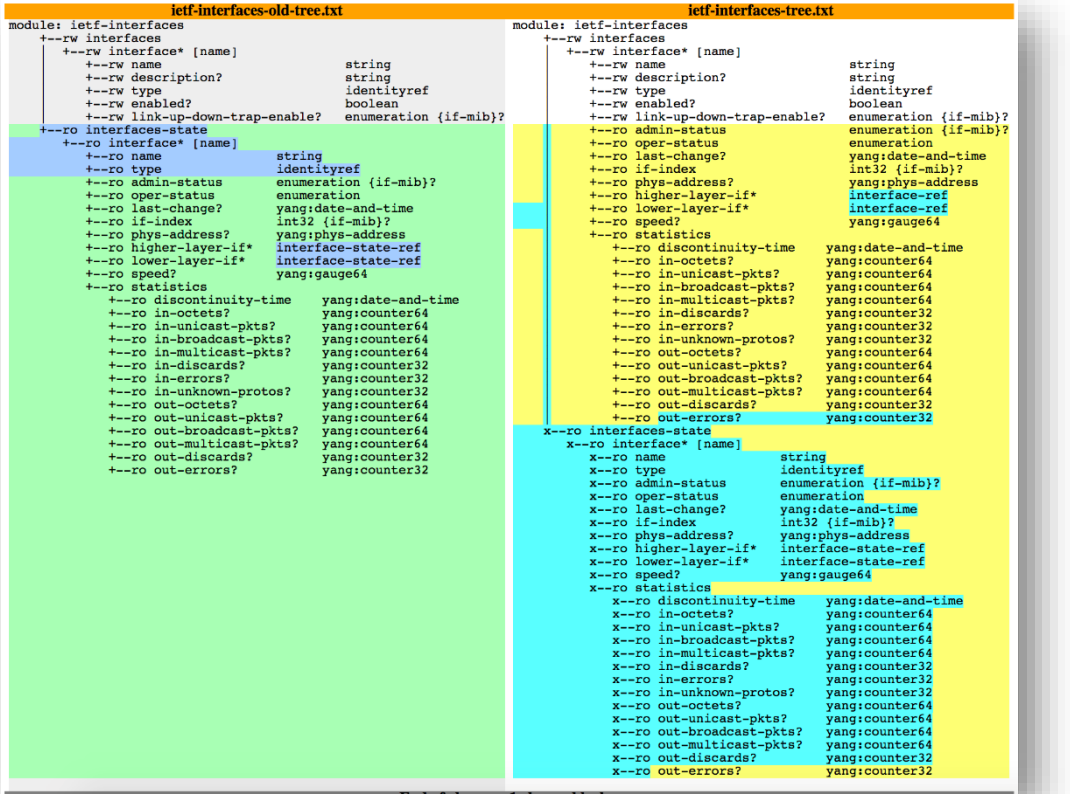
Semantic Versioning



7. What About Upgrading the IOS?

Semantic Version Diffs

1.0.0 2.0.0



ietf-interfaces-old-tree.txt

```
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 {if-mib}?
   +--ro interfaces-state
      +--ro interface* [name]
         +--ro name                   string
         +--ro type                   identityref
         +--ro admin-status           enumeration {if-mib}?
         +--ro oper-status            enumeration
         +--ro last-change?           yang:date-and-time
         +--ro if-index               int32 {if-mib}?
         +--ro phys-address?          yang:phys-address
         +--ro higher-layer-if*       interface-state-ref
         +--ro lower-layer-if*        interface-state-ref
         +--ro speed?                 yang:gauge64
         +--ro statistics
            +--ro discontinuity-time    yang:date-and-time
            +--ro in-octets?            yang:counter64
            +--ro in-unicast-pkts?      yang:counter64
            +--ro in-broadcast-pkts?    yang:counter64
            +--ro in-multicast-pkts?    yang:counter64
            +--ro in-discards?          yang:counter32
            +--ro in-errors?            yang:counter32
            +--ro in-unknown-protos?    yang:counter32
            +--ro out-octets?           yang:counter64
            +--ro out-unicast-pkts?     yang:counter64
            +--ro out-broadcast-pkts?   yang:counter64
            +--ro out-multicast-pkts?   yang:counter64
            +--ro out-discards?         yang:counter32
            +--ro out-errors?           yang:counter32
```

ietf-interfaces-tree.txt

```
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 {if-mib}?
         +--ro admin-status           enumeration {if-mib}?
         +--ro oper-status            enumeration
         +--ro last-change?           yang:date-and-time
         +--ro if-index               int32 {if-mib}?
         +--ro phys-address?          yang:phys-address
         +--ro higher-layer-if*       interface-ref
         +--ro lower-layer-if*        interface-ref
         +--ro speed?                 yang:gauge64
         +--ro statistics
            +--ro discontinuity-time    yang:date-and-time
            +--ro in-octets?            yang:counter64
            +--ro in-unicast-pkts?      yang:counter64
            +--ro in-broadcast-pkts?    yang:counter64
            +--ro in-multicast-pkts?    yang:counter64
            +--ro in-discards?          yang:counter32
            +--ro in-errors?            yang:counter32
            +--ro in-unknown-protos?    yang:counter32
            +--ro out-octets?           yang:counter64
            +--ro out-unicast-pkts?     yang:counter64
            +--ro out-broadcast-pkts?   yang:counter64
            +--ro out-multicast-pkts?   yang:counter64
            +--ro out-discards?         yang:counter32
            +--ro out-errors?           yang:counter32
   x--ro interfaces-state
      x--ro interface* [name]
         x--ro name                   string
         x--ro type                   identityref
         x--ro admin-status           enumeration {if-mib}?
         x--ro oper-status            enumeration
         x--ro last-change?           yang:date-and-time
         x--ro if-index               int32 {if-mib}?
         x--ro phys-address?          yang:phys-address
         x--ro higher-layer-if*       interface-state-ref
         x--ro lower-layer-if*        interface-state-ref
         x--ro speed?                 yang:gauge64
         x--ro statistics
            x--ro discontinuity-time    yang:date-and-time
            x--ro in-octets?            yang:counter64
            x--ro in-unicast-pkts?      yang:counter64
            x--ro in-broadcast-pkts?    yang:counter64
            x--ro in-multicast-pkts?    yang:counter64
            x--ro in-discards?          yang:counter32
            x--ro in-errors?            yang:counter32
            x--ro in-unknown-protos?    yang:counter32
            x--ro out-octets?           yang:counter64
            x--ro out-unicast-pkts?     yang:counter64
            x--ro out-broadcast-pkts?   yang:counter64
            x--ro out-multicast-pkts?   yang:counter64
            x--ro out-discards?         yang:counter32
            x--ro out-errors?           yang:counter32
```

The Catalog can provide links to diff the module's tree and full structure

Eating our own Dog Food: yangcatalorg.org is API driven

- Yangcatalog is Confd based, with the YANG module in draft-clacla-netmod-model-catalog-02
  - All APIs are automatically generated
- Demo: POSTMAN collection
  - Ex: https://yangcatalog.org:8443/search/name/Cisco-IOS-XR-ipv4-bgp-cfg
  - Ex: all openconfig YANG modules on OS/platform
- We want operators to integrate the APIs in their toolchain
  - Download yangcatalog postman collection here

Eating our own Dog Food: yangcatalorg.org is API driven

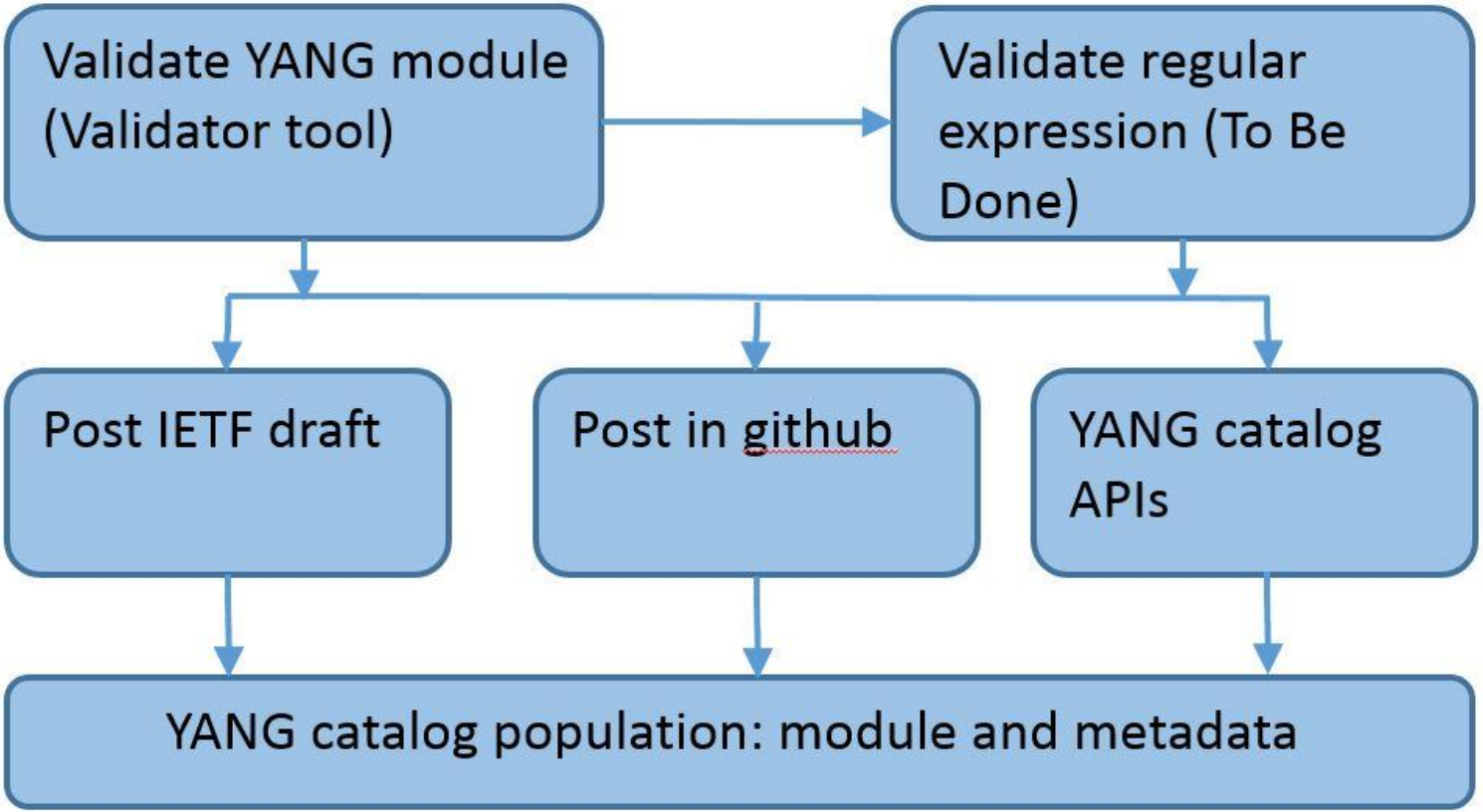
Module Sub-tree

```
module: yang-catalog
    +--rw catalog
       +--rw modules
          |  +--rw module* [name revision organization]
          |     +--rw name                          yang:yang-identifier
          |     +--rw revision                      union
          |     +--rw organization                  string
          |     +--rw ietf
          |     |  +--rw ietf-wg?   string
          |     +--rw namespace                     inet:uri
          |     +--rw schema?                       inet:uri
          |     +--rw generated-from?               enumeration
          |     +--rw maturity-level?               enumeration
          |     +--rw document-name?                string
          |     +--rw author-email?                 yc:email-address
          |     +--rw reference?                    inet:uri
          |     +--rw module-classification         enumeration
          |     +--rw compilation-status?           enumeration
          |     +--rw compilation-result?           inet:uri
          |     +--rw prefix?                       string
          |     +--rw yang-version?                 enumeration
          |     +--rw description?                  string
          |     +--rw contact?                      string
          |     +--rw module-type?                  enumeration
          |     +--rw belongs-to?                   yang:yang-identifier
          |     +--rw tree-type?                    enumeration
          |     +--rw submodule* [name revision]
          |     |  +--rw name         yang:yang-identifier
          |     |  +--rw revision     union
          |     |  +--rw schema?      inet:uri
          |     +--rw dependencies* [name]
          |     |  +--rw name         yang:yang-identifier
          |     |  +--rw revision?    union
          |     |  +--rw schema?      inet:uri
          |     +--rw dependents* [name]
          |     |  +--rw name         yang:yang-identifier
          |     |  +--rw revision?    union
          |     |  +--rw schema?      inet:uri
          |     +--rw semantic-version?             yc:semver
          |     +--rw derived-semantic-version?     yc:semver
          |     +--rw implementations
          |        +--rw implementation* [vendor platform software-version software-flavor]
          |           +--rw vendor              string
          |           +--rw platform            string
          |           +--rw software-version    string
          |           +--rw software-flavor     string
          |           +--rw os-version?         string
          |           +--rw feature-set?        string
          |           +--rw os-type?            string
          |           +--rw feature*            yang:yang-identifier
          |           +--rw deviation* [name revision]
          |           |  +--rw name         yang:yang-identifier
          |           |  +--rw revision     union
          |           +--rw conformance-type?   enumeration
```

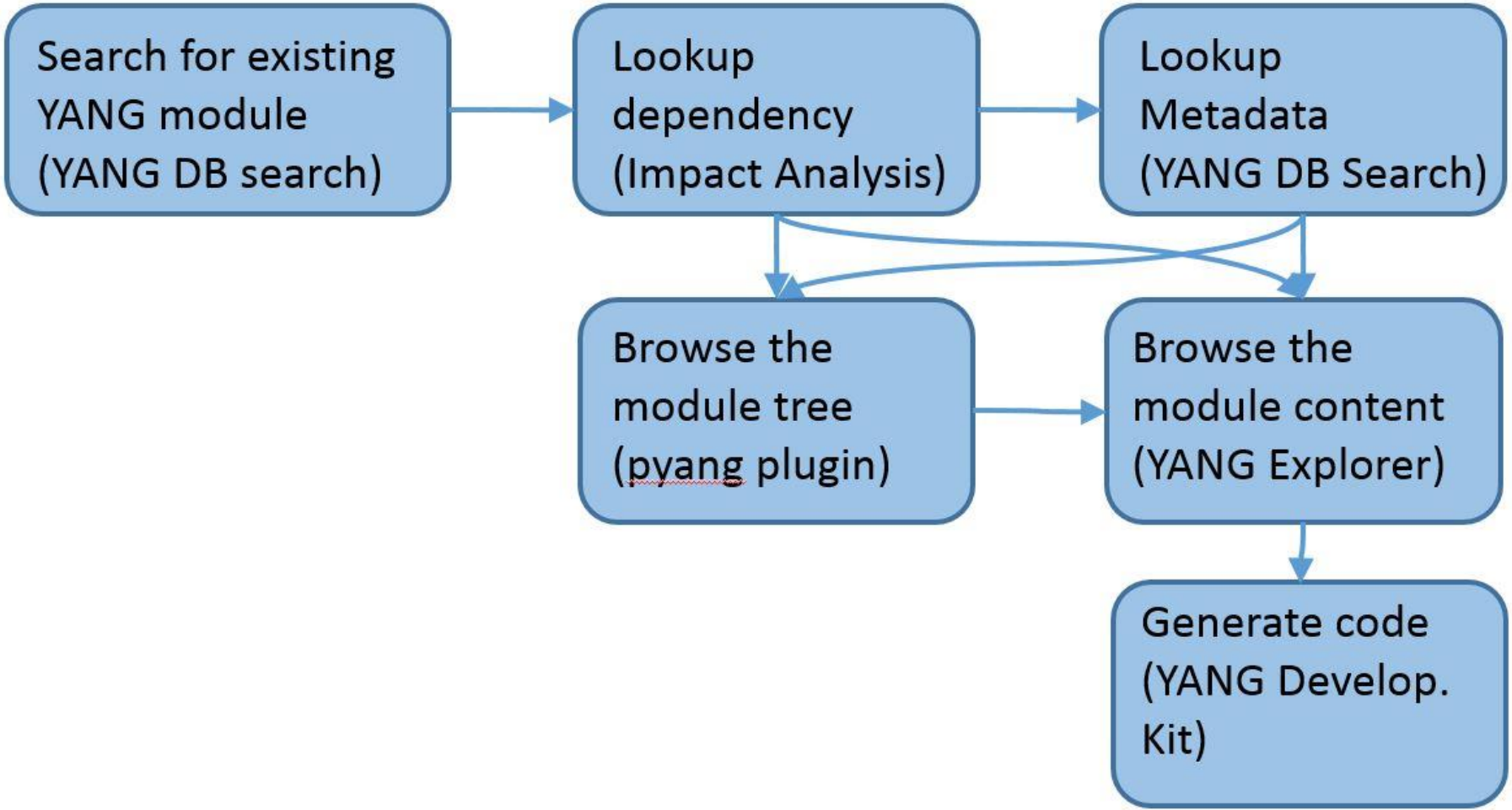
Vendor Sub-tree

```
      +--rw vendors
         +--rw vendor* [name]
            +--rw name          string
            +--rw platforms
               +--rw platform* [name]
                  +--rw name                  string
                  +--rw software-versions
                     +--rw software-version* [name]
                        +--rw name                  string
                        +--rw software-flavors
                           +--rw software-flavor* [name]
                              +--rw name          string
                              +--rw protocols
                              |  +--rw protocol* [name]
                              |     +--rw name                  identityref
                              |     +--rw protocol-version*     string
                              |     +--rw capabilities*         string
                              +--rw modules
                                 +--rw module* [name revision organization]
                                    +--rw name                  -> /catalog/modules/module/name
                                    +--rw revision              -> /catalog/modules/module/revision
                                    +--rw organization          -> /catalog/modules/module/organization
                                    +--rw os-version?           string
                                    +--rw feature-set?          string
                                    +--rw os-type?              string
                                    +--rw feature*              yang:yang-identifier
                                    +--rw deviation* [name revision]
                                    |  +--rw name         yang:yang-identifier
                                    |  +--rw revision     union
                                    +--rw conformance-type?   enumeration
```

YANG Module Designer



YANG Module User



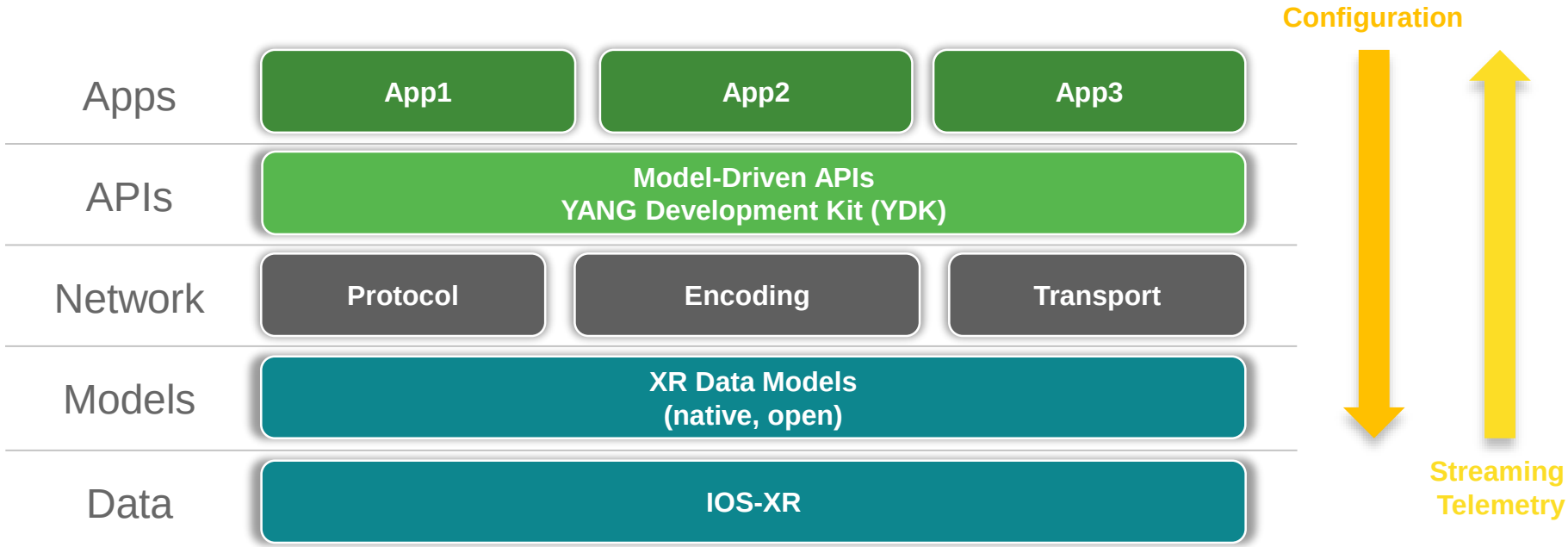
Who Should Know about YANG?

- Even if I push YANG everywhere, not everybody should (have to) know about YANG.

- Operator: Not really, as YANG modules = APIs.
- Operator service designer: should know YANG
- Architect: should design your (new) features from a YANG point of view
- Involved in opensource or SDOs: you should care about YANG toolchain

8. What's Missing?
Data Model-driven Telemetry

Solving a SNMP Polling, Data Modeling, and OPEX Issues



Agenda

- Data Model-driven Management
- Industry Developments
- Yangcatalog.org
- Conclusion
- References

Summary and Key Messages

- Automation and programmability are required these days
- Data Modeling-driven set of APIs is key for automation
  - The key is the data models
- YANG is the data modeling language for configuration and monitoring: a full, formal contract language with rich syntax and semantics to build applications on.
- Many YANG data model developments
  - In different standard development organizations (but primarly at the IETF),
  - In open source
- The YANG catalog set of data models, metadata, and tools is here to help

Data Model-Driven Management: Latest Industry and Tool Development

Benoit Claise

References

General Training

- Link to IETF 94 Recording: NETCONF, YANG, pyang
  - Link to slides at IETF 94: NETCONF Slides, YANG Slides,
- RFC 6244, An Architecture for Network Management Using NETCONF and YANG

Tooling – Exploring and using NETCONF/YANG

- Editor plug-ins
  - emacs (yang-mode.el)
  - vim (yang.vim)
  - sublime text (sublime-yang-syntax)
- pyang
  - an extensible YANG validator written in Python. (Video trainining: pyang)
  - can be used standalone as a validator of YANG modules, or to generate YIN, YANG, DSDL and XSD from YANG and YIN.
  - https://github.com/mbj4668/pyang
  - http://www.yangvalidator.com/
- libsmi
  - A library allowing the generation of YANG models from SMI/SMIv2 compliant MIBs

Tooling – Exploring and using NETCONF/YANG

- ncclient
  - a Python library that facilitates client-side scripting and application development around the NETCONF protocol (only supports NETCONF 1.0)
- Postman
  - a Chrome plugin for RESTCONF, allowing for customized sets of REST snippets to be easily built, maintain and shared. Useful for NETCONF via RESTCONF, for example Open Daylight
- OpenDaylight
  - enables auto-generation of RESTconf APIs from YANG models, with NETCONF client support
  - APIdocs feature provides a way to explore both local and mounted YANG models