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.
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
Data Modeling Language (schema language)

Data Modeling (schema)

Encoding (serialization)

Protocol

Application

Prog. Language

Non Standard Possible links

YANG

YANG Data Model

XML

JSON

ProtoBuf

Thrift

NETCONF

RESTCONF

GRPC (HTTP/2)

YANG Development Kit

Python

C++

Any 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**

```plaintext
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

NETCONF 1.0, SSH Mapping
RFC4741, RFC4742
December 2006

YANG 1.0
RFC6020
October 2010

Common YANG Data Types
RFC6991
July 2013

Interface and IP Modules
RFC7223, RFC7277
May, June 2014

Routing Management
RFC8022
November 2016

JSON Encoding
RFC7951
August 2016

NETCONF 1.1
RFC6241
June 2011

NETCONF Access Control
RFC6536
March 2012

NETCONF over TLS + x.509
RFC7589
October 2016

RESTCONF Protocol
RESTCONF
January 2017
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"


- 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
• 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
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
YANG Tsunami in the Industry
SDOs Alignment and Trajectory

Network Service YANG data models
- VPWS
- L2VPN
- VPLS
- L2VPN
- L3VPN

Network Element YANG data models
- MPLS
- BGP
- IPv4 & IPv6
- Ethernet
Data Model Location and Type (Network Element)

<table>
<thead>
<tr>
<th>Network Element</th>
<th>Standard YANG Model</th>
<th>Proprietary YANG Model (also called « native » models)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Proprietary Extension to Standard YANG Model</td>
</tr>
</tbody>
</table>
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?

• 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

- IETF drafts:

- 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](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
<table>
<thead>
<tr>
<th>Name</th>
<th>Revision</th>
<th>Schema Type</th>
<th>Path</th>
<th>Module</th>
<th>Origin</th>
<th>Organization</th>
<th>Maturity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>address-family-vrf-grouping</td>
<td>2017-02-07</td>
<td>grouping</td>
<td>/ios.rip:address-family-vrf-grouping</td>
<td>Cisco-IOS-XE-rip (Impact Analysis)</td>
<td>Vendor-Specific</td>
<td>cisco</td>
<td></td>
<td></td>
</tr>
<tr>
<td>af-ipv4-uc-and-vrf-cmds</td>
<td>2010-11-29</td>
<td>grouping</td>
<td>/bgp:af-ipv4-uc-and-vrf-cmds</td>
<td>brocade-bgp (Impact Analysis)</td>
<td>Vendor-Specific</td>
<td>brocade.com</td>
<td></td>
<td>Clear all non-default and default OSPF VRFs</td>
</tr>
</tbody>
</table>
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
Yangcatalog.org is API driven

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

```json
{
  "yang-catalog:modules": {
    "module": {
      "name": "openconfig-acl",
      "revision": "2016-08-08",
      "organization": "openconfig",
      "namespace": "http://openconfig.net.yang/acl",
      "generated-from": "not-applicable",
      "module-classification": "unknown",
      "compilation-status": "failed",
      "compilation-result": "https://yangcatalog.org/results/openconfig-acl102016.08.08/openconfig_html"
    }
  }
}
```
Impact Analysis

- Demo


Impact Analysis
Impact Analysis

YANG Impact Graph for Module(s): ietf-routing
Tracking Dependencies and Dependents

```
"yang-tree": "https://yangcatalog.org:8443/services/tree/ietf-ifaces?08-17.yang",
"dependencies": [],
"dependents": []
```

![Diagram of YANG impact graph for module ietf-interfaces](image)
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

- 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.

```
[ietf-interfaces@2013-12-23]
  
  
  [ ]
  "derived-semantic-version": "1.0.0",
  "implementations": { }
  "implementation": [ 
  
  
  [ ]
  "derived-semantic-version": "2.0.0",
  "implementations": { }
  "implementation": [ ]
```
7. What About Upgrading the IOS?

Semantic Versioning

```
POST https://yangcatalog.org:8443/check-sematic-version

Authorization  Headers (4)  Body  Pre-request Script  Tests

form-data  x-www-form-urlencoded  raw  binary

1
2  "input": {
3    "old": {
4      "implementations": {
6        "implementation": [
8          "vendor": "cisco",
10         "software-version": "6.1.1",
11         "platform": "ASR9K"
13      ]
15    }]
17  },
18  "raw": {
20    "implementations": [
22      "implementation": [
24        "vendor": "cisco",
26         "software-version": "6.1.3",
28         "platform": "ASR9K"
30      ]
32  ]
34
```

```

Body  Cookies  Headers (4)  Test Results

Pretty  Raw  Preview  JSON

1
2  "output": [
4    "derived-sematic-version-results": "Both modules failed compilation",
5    "name": "Cisco-105-XR-bundlemgr-cfg",
6    "new-derived-sematic-version": "2.0.0",
7    "old-derived-sematic-version": "1.0.0",
8    "organization": "cisco",
9    "AuditTimeUTC": "2018-12-18"
10  ]
```
7. What About Upgrading the IOS?

Semantic Version Diffs

**1.0.0**

**2.0.0**

The Catalog can provide links to diff the module’s tree and full structure.
Eating our own Dog Food: yangcatalog.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

Vendor Sub-tree
YANG Module Designer

Validate YANG module (Validator tool) → Validate regular expression (To Be Done)

Post IETF draft → Post in github → YANG catalog APIs

YANG catalog population: module and metadata
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

<table>
<thead>
<tr>
<th>Apps</th>
<th>App1</th>
<th>App2</th>
<th>App3</th>
</tr>
</thead>
<tbody>
<tr>
<td>APIs</td>
<td>Model-Driven APIs</td>
<td>YANG Development Kit (YDK)</td>
<td></td>
</tr>
<tr>
<td>Network</td>
<td>Protocol</td>
<td>Encoding</td>
<td>Transport</td>
</tr>
<tr>
<td>Models</td>
<td>XR Data Models (native, open)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td>IOS-XR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Configuration

Streaming Telemetry
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 primarily 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 training: [pyang](http://www.yang-central.org))
  - 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](https://github.com/mbj4668/pyang)

- 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
  - API docs feature provides a way to explore both local and mounted YANG models