



We're ready.
Are you?

Introduction to RESTCONF

Matthew DeNapoli - Systems Manager

Agenda and Housekeeping

Housekeeping

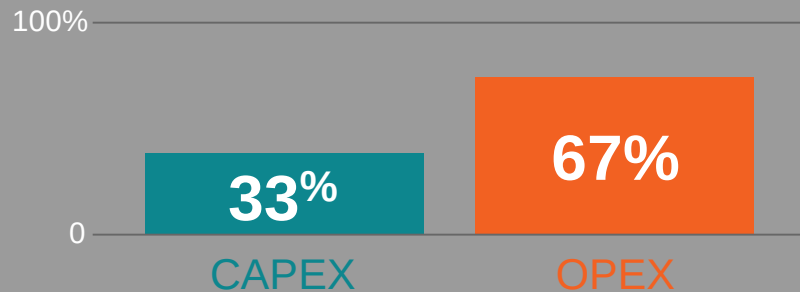
- Lecture Part
- Hands-On Part
 - Chrome / Postman
 - Python
- Ask questions right away

Agenda

- Overview
- RESTCONF and YANG
- URI Construction
- Configuration Snippets

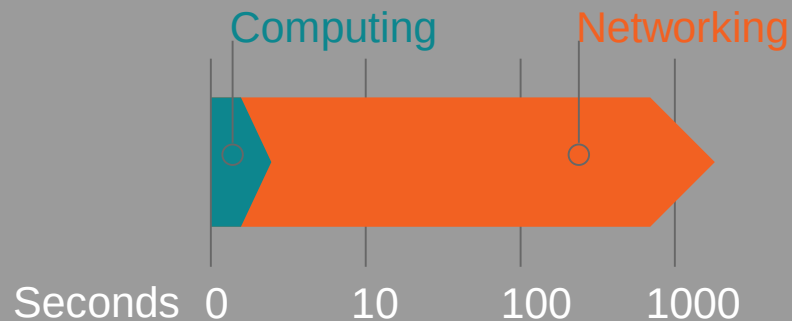
Why Network Programmability Matters

Network Expenses



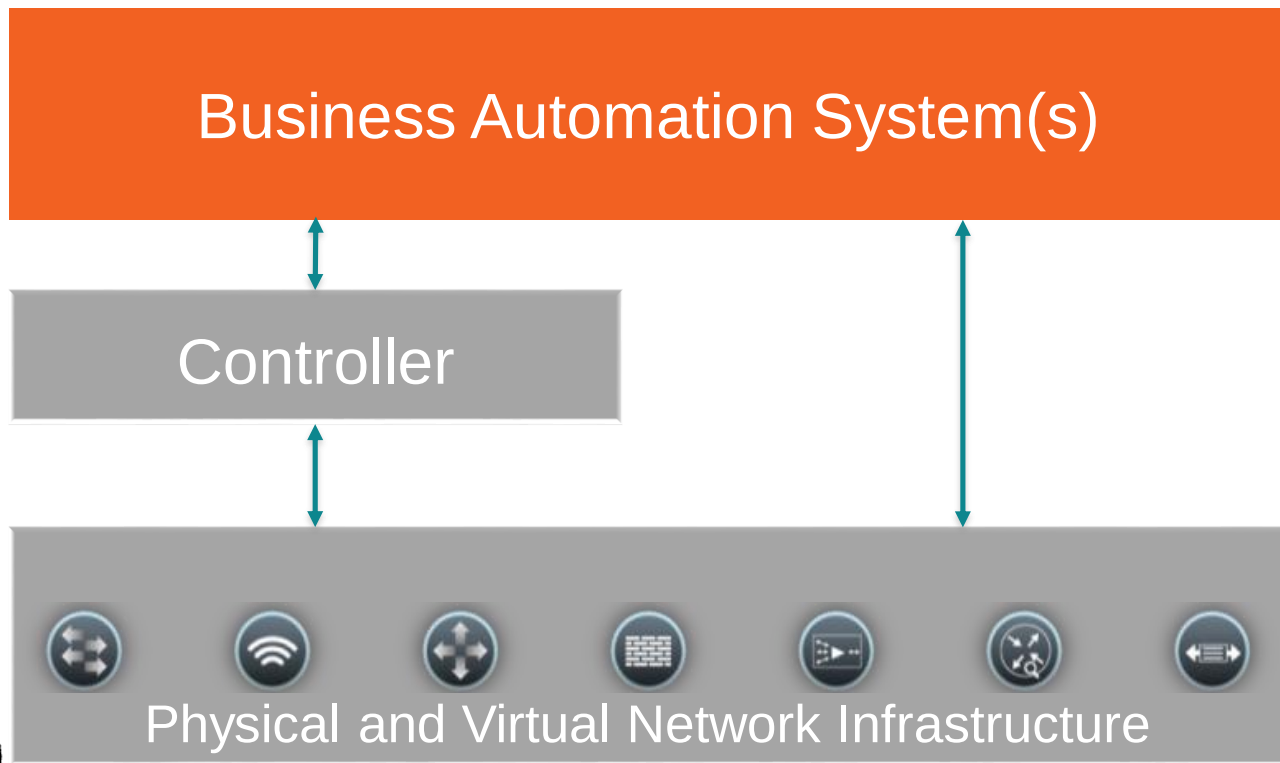
Source: Forrester

Deployment Speed

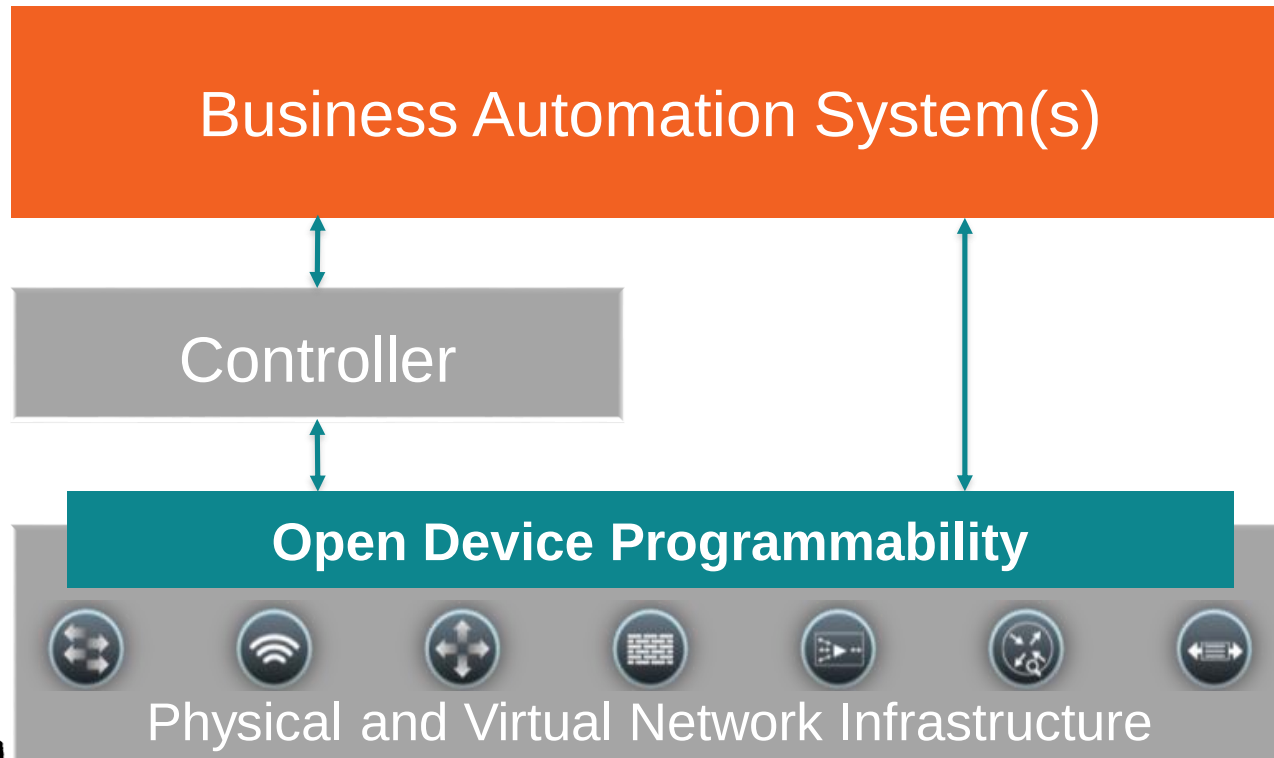


Source: Open Compute Project

Network Programmability



Network Programmability



Network Programmability

**Configuration
Management**

**Operational
State**

**Traffic
Engineering**

Programmatic
Interfaces

Open
Protocols

Open Device Programmability



Physical and Virtual Network Infrastructure

Implementation



International
Telecommunication
Union

OpenConfig



Other vendors

Open Device Programmability



Physical and Virtual Network Infrastructure

Cisco *live!*
DevNet Program

RESTconf

NETCONF

gRPC

Data Model

Configuration

Operational

Standard

Device
Specific

Standard

Device
Specific

Device Features

Interface

BGP

QoS

ACL

...

Context: Programmatic Interfaces

RESTCONF

- IETF draft
- Designed for web applications (REST “like”, based on NETCONF)
- Session oriented
- Runs over HTTPS

NETCONF

- IETF standard
- Designed for network programmability
- Session and connection oriented
- Runs over SSH

gRPC

- Open source initiative
- Designed for generic client / service communications
- Session oriented
- Runs over HTTPS

RESTCONF: What is it?



- IETF driven
- Extends the idea of NETCONF
- **RESTCONF promises a standardised / vendor agnostic management approach** to network device management
- Using REST as a popular access method
- Modular and extensible
- Model definition in YANG
- JSON or XML are used for data representation

Current State of Draft / RFC

- Not standardised as of today (February 2016)
- Latest draft revision -09 (expires June 2016)
- <https://tools.ietf.org/html/draft-ietf-netconf-restconf-09>

[Docs] [txt|pdf|xml|html] [Tracker] [WG] [Email] [Diff1] [Diff2] [Nits]

Versions: ([draft-bierman-netconf-restconf](#)) [00](#)
[01](#) [02](#) [03](#) [04](#) [05](#) [06](#) [07](#) [08](#) [09](#)

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 17, 2016

A. Bierman
YumaWorks
M. Bjorklund
Tail-f Systems
K. Watzen
Juniper Networks
December 15, 2015

RESTCONF Protocol
draft-ietf-netconf-restconf-09

Abstract

This document describes an HTTP-based protocol that provides a programmatic interface for accessing data defined in YANG, using the datastores defined in NETCONF.

Use Cases

SDN Controller Integration



Open SDN
Controller



Configuration Mgmt

Access Control

Inventory / Topology

Cisco *live!*

DevNet Program

Application Integration



Custom
Application



ANSIBLE

CFEngine

Script Automation

DevOps

OSS / BSS Integration



SIGMA

NETWORKMINING

Configuration Mgmt

Service Provisioning

Fault Mgmt

Data Model Life Cycle Management



Industry Standard

- **Standard definition** (IETF, ITU, OpenConfig, etc.)
- Compliant with standard, i.e. “Policy”
`ietf-diffserv-policy.yang`
`ietf-diffserv-classifier.yang`
`ietf-diffserv-target.yang`



Cisco Common

- **Cisco definition**
- Common across Cisco platforms,

i.e. “OTV” on IOS-XE and NX-OS



Cisco Platform Specific

- **Cisco definition**
- Unique to specific Cisco platform,

i.e. “BGP” extensions on IOS-XE

Reality Check: What's Available?

Implementations available in

- IOS XR 6.0
- IOS XE 3.17

Both are in controlled availability

RESTCONF and YANG

So... Where's the API Reference Guide?

- **Well, there is none.**

RESTCONF

... is about the process, a formalised way to talk to a device

... knows a few 'verbs'

... uses data models to describe device capabilities and functions

... provides well known entry points to 'discover' those

Think 'SNMP and MIBs'...

RESTCONF and SNMP

“Framework”

Definition Language:
YANG



Definition Language:
SMIV2

“Content”

Information Model:
YANG modules



Information Model:
MIBs

“Payload”

Instantiated / transfer syntax:
XML / JSON



Instantiated / transfer syntax:
ASN.1 BER

Management Service:
RESTCONF



Management Service:
SNMP

What is YANG?

- YANG is a modeling language defined in RFC 6020
- Used by RESTCONF to define the objects and data in requests and replies
- Analogous to XML schema and SMI for SNMP (but more powerful)
- Models configuration, operational, and RPC data
- Provides semantics to better define RESTCONF data
 - Constraints (i.e., “MUSTs”)
 - Reusable structures
 - Built-in and derived types
- YANG is extensible and modular
- YANG modules are for RESTCONF what MIBs are for SNMP

Working with YANG Models

- RESTCONF defines how a YANG model is mapped to a RESTful interface
- Specifically
 - How to modify the data by applying using REST verbs (GET / PUT / PATCH / ...)
 - How to construct URIs to access the model / data
- Tools help to understand YANG models

Where to get the Models?

- <https://github.com/YangModels/yang>

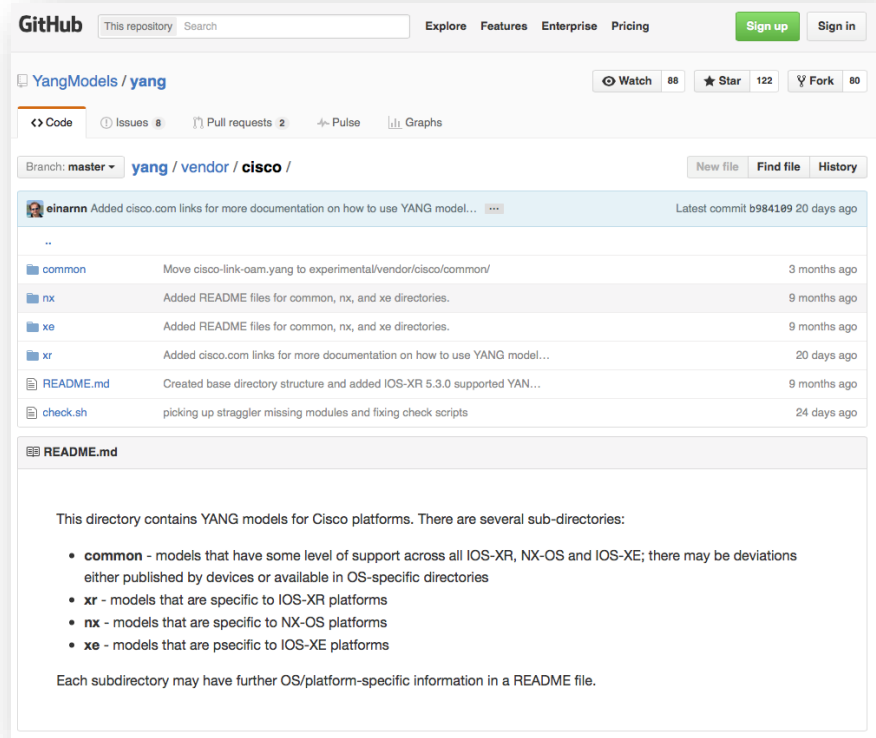
Note: No XE models have been published so far!

“YANG modules from standard organisations such as the IETF, open source such as Open Daylight or vendor specific modules”

- <https://github.com/CiscoDevNet/xenetcnf-yang> **
- During a lab, like here 😊

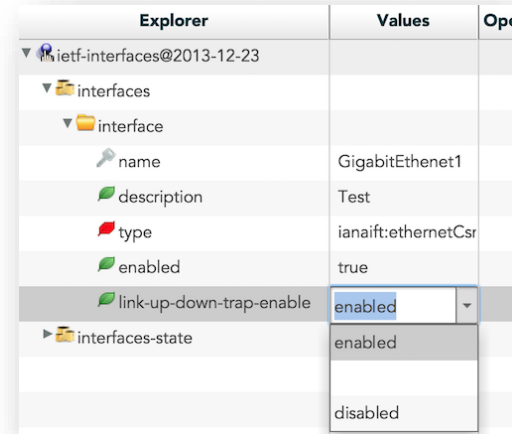
Cisco *live!*
DevNet Program

** private, add required



Tools to work with YANG Models

- **PYANG** ‘An extensible YANG validator and converter in python’
 - <https://github.com/mbj4668/pyang>
 - Via PyPi: `pyang` - A YANG (RFC 6020) validator and converter
 - Mandatory tool 😊
- **YANG Explorer** ‘An open-source YANG Browser and RPC Builder Application’
 - <https://github.com/CiscoDevNet/yang-explorer>
 - Web Based GUI
 - More difficult to get started with



Explorer	Values	Open
▼ ietf-interfaces@2013-12-23		
▼ interfaces		
▼ interface		
name	GigabitEthernet1	
description	Test	
type	ianaif:ethernetCsr	
enabled	true	
link-up-down-trap-enabled	<input type="text" value="enabled"/>	
▶ interfaces-state	enabled	
	disabled	

Display a YANG Module

```
$ pyang -f tree <yang-file>
```

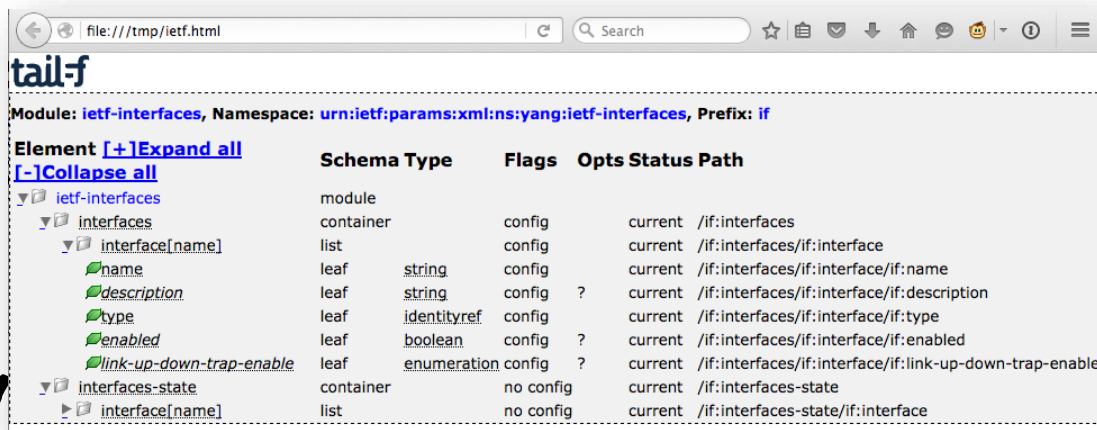
```
(restconf)$ pyang -f tree -p yang/standard/ietf/RFC yang/standard/ietf/RFC/ietf-interfaces.yang
module: ietf-interfaces
```

```
  +--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
```

```
[...]
```

pyang Tip

- JavaScript Tree Output really useful
- Use `pyang -f jstree -p <path-to-models> <model.yang>`
 `>/tmp/ietf.html`
- Produces collapsible Tree / HTML



The screenshot shows a web browser window with the address bar displaying `file:///tmp/ietf.html`. The page title is **tail-f**. Below the title, the module and namespace are specified: **Module: ietf-interfaces, Namespace: urn:ietf:params:xml:ns:yang:ietf-interfaces, Prefix: if**.

The main content is a tree view of YANG model elements. The tree is expanded to show the `ietf-interfaces` module, which contains the `interfaces` container. The `interfaces` container has a child `interface[name]` list. The `interface[name]` list has several leaf children: `name` (string), `description` (string), `type` (identityref), `enabled` (boolean), and `link-up-down-trap-enable` (enumeration). The `link-up-down-trap-enable` leaf has a child `interfaces-state` container. The `interfaces-state` container has a child `interface[name]` list.

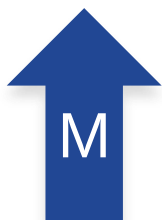
Element	Schema Type	Flags	Opts	Status	Path
ietf-interfaces	module				
interfaces	container	config		current	/if:interfaces
interface[name]	list	config		current	/if:interfaces/if:interface
name	leaf string	config		current	/if:interfaces/if:interface/if:name
description	leaf string	config ?		current	/if:interfaces/if:interface/if:description
type	leaf identityref	config		current	/if:interfaces/if:interface/if:type
enabled	leaf boolean	config ?		current	/if:interfaces/if:interface/if:enabled
link-up-down-trap-enable	leaf enumeration	config ?		current	/if:interfaces/if:interface/if:link-up-down-trap-enable
interfaces-state	container	no config		current	/if:interfaces-state
interface[name]	list	no config		current	/if:interfaces-state/if:interface

URI Construction

*(some slides credit:
Wojciech Dec's BRKSDN-1903)*

Request URI Structure

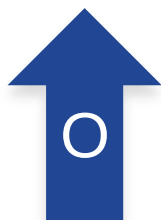
<OP> /<api-entry>/<path>?<query>#<fragment>



Method



Entry



Resource



Query



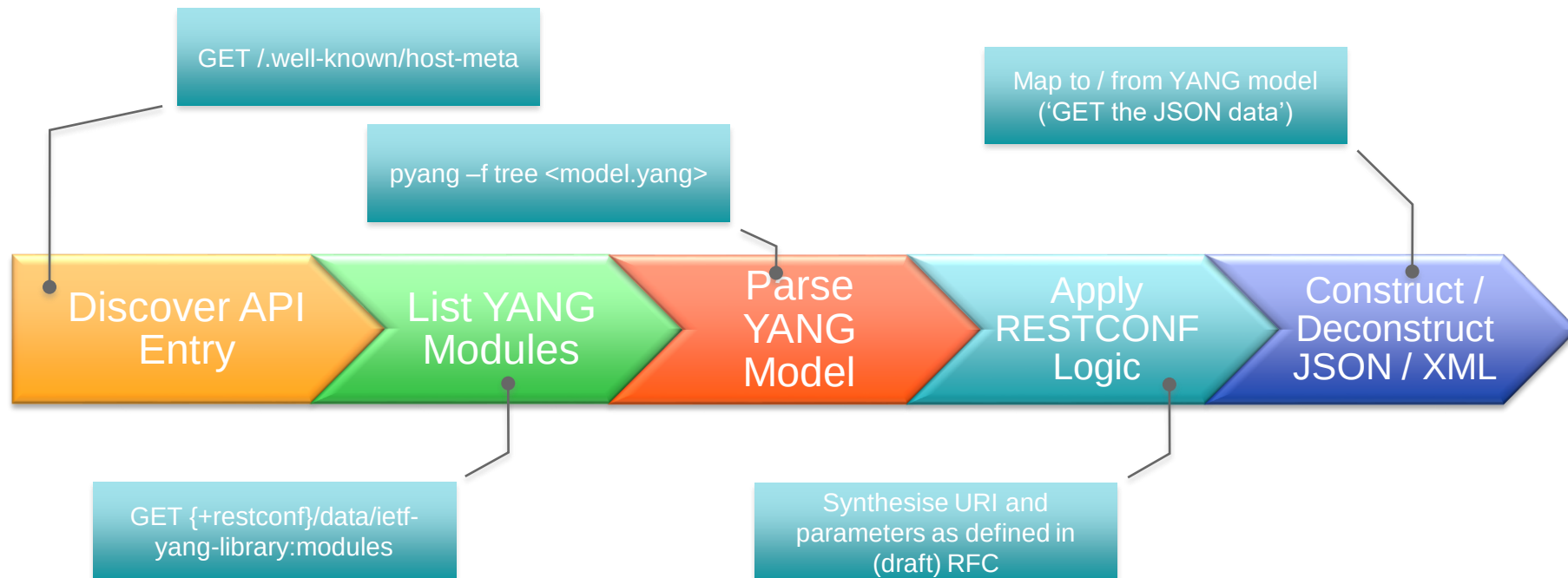
Fragment

M=mandatory, O=optional, I=ignored

CRUD Methods in RESTCONF

RESTCONF	As compared to NETCONF
OPTIONS	none
HEAD	none
GET	<get-config>, <get>
POST	<edit-config> (operation="create")
PUT	<edit-config> (operation="create/replace")
PATCH	<edit-config> (operation="merge")
DELETE	<edit-config> (operation="delete")

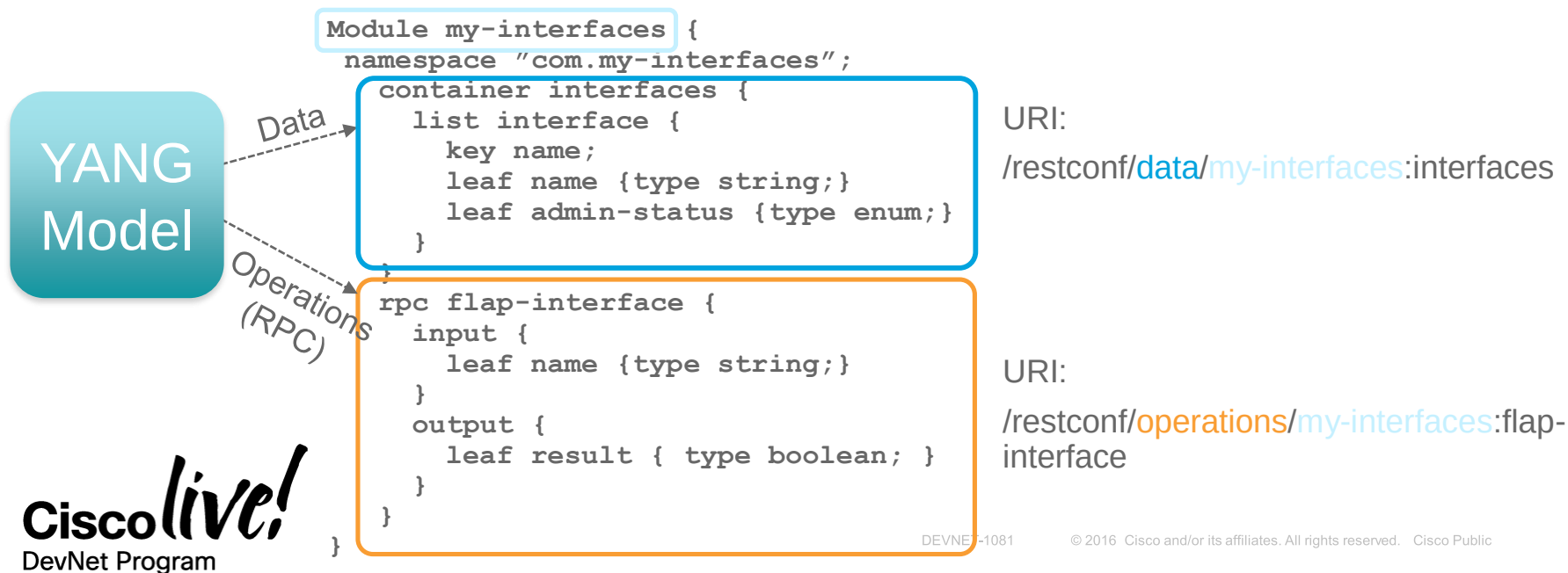
Flow: API Discovery and Use



RESTCONF API (1/4)

A RESTCONF URI is:

/<api-entry>/<resource-type>/<yang-module:resource>



RESTCONF API (2/4)

Containers and Lists

/<api-entry>/<resource-type>/<yang-module:resource>



Data

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

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

URI:

/restconf/data/my-interfaces:interfaces

URI:

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

RESTCONF API (3/4)

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>

RESTCONF API (4/4)

List Data

- RESTCONF requires that data passed includes module name-space
- Creation of List items is done using POST to resource: `/restconf/data/my-interfaces:interfaces/interface` with JSON or XML data expressing name

```
Module my-interfaces {  
  namespace "com.my-interfaces";  
  container interfaces {  
    list interface {  
      key name;  
      leaf name  
        { type string; }  
      leaf admin-status  
        { type enum; }  
    }  
  }  
}
```

```
{ "my-interfaces:interface": [  
  { "name": "GigabitEthernet0/0",  
    "admin-status": up  
  }  
]
```

```
<interface xmlns="com.my-interfaces">  
  <name>GigabitEthernet0/0</name>  
  <admin-status>up</admin-status>  
</interface>
```

Configuration Snippets

IOS XE (3.17) Configuration Snippet

```
remote-management
  no restful-api
  dmi
!
interface VirtualPortGroup0
  ip unnumbered GigabitEthernet1
!
virtual-service csr_mgmt
  vnic gateway VirtualPortGroup0
  guest ip address 172.16.1.32
  activate
!
ip route 172.16.1.32 255.255.255.255 VirtualPortGroup0
```

Ports:
TCP/8008 (HTTP)

**In the lab: 9443, port
translated**

IOS XE (3.17) with VRF Configuration Snippet

```
remote-management
  no restful-api
  dmi
!
interface GigabitEthernet1
  vrf forwarding <VRF>
  ip address <MGMT_IP AND MASK>
!
interface VirtualPortGroup0
  vrf forwarding <VRF>
  ip unnumbered GigabitEthernet1
!
virtual-service csr_mgmt
  vnic gateway VirtualPortGroup0
  guest ip address <CONTAINER-IP>
  activate
!
ip route VRF <VRF> <CONTAINER-IP> 255.255.255.255 VirtualPortGroup0 <CONTAINER-IP>
```

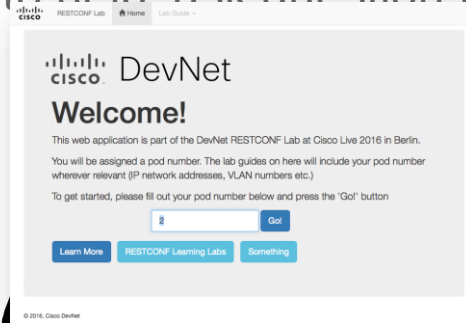
IOS XR (6.0) Configuration Snippet

```
interface Loopback1
  ipv4 address 128.0.0.1 255.0.0.0
!
web server
  service restconf
    http-port 80
    https-port 443
    http-enable
  !
!
restconf agent
  open-request-limit 10
  per-user-request-limit 10
!
```

RESTCONF Hands On

How to Access the Lab

- A shared CSR 1000v hosted in the DevNet Sandbox will be used
- Lab Instructions (access, credentials, exercises etc.) are here
<http://virl-dev-innovate.cisco.com:8300>
- We're here to help ask questions!



Resources

Interface Protocols

- RESTCONF- <https://tools.ietf.org/html/draft-ietf-netconf-restconf-09>
- NETCONF - <https://tools.ietf.org/html/rfc6241>
- gRPC - <http://www.grpc.io>

Data Models

- IETF - <https://tools.ietf.org/html/rfc6020>
- ITU - <https://www.itu.int/oth/T1C02000002>
- OpenConfig - <http://www.openconfig.net/data-models>

Resources (cont.)

- Wojciech's CL deck 'A Model-driven Approach to Software Defined Networks with Yang, NETCONF/RESTCONF', BRKSDN-1903
https://www.ciscolive.com/online/connect/sessionDetail.ww?SESSION_ID=84149&tclass=popup
- YANG Explorer (open source)
<https://github.com/CiscoDevNet/yang-explorer>
- SDN Tutorials: What is RESTCONF?
<http://sdntutorials.com/what-is-restconf/>
- Postman and self-signed Certs
<http://blog.getpostman.com/2014/01/28/using-self-signed-certificates-with-postman/>

Resources (cont.)

- YANG Module Repository (IETF, Open, Vendors)
<https://github.com/YangModels/yang>
- Pyang tool
<https://github.com/mbj4668/pyang>

Conclusion

Conclusion

RESTCONF defines how a YANG model is mapped to a RESTful interface

There is no API guide, but tools to deal with models

It uses a well understood transport (REST)

Get Ready to RESTCONF!

Q & A

Complete Your Online Session Evaluation

Give us your feedback and receive a **Cisco 2016 T-Shirt** by completing the Overall Event Survey and 5 Session Evaluations.

- Directly from your mobile device on the Cisco Live Mobile App
- By visiting the Cisco Live Mobile Site
<http://showcase.genie-connect.com/ciscolivemelbourne2016/>
- Visit any Cisco Live Internet Station located throughout the venue

T-Shirts can be collected Friday 11 March at Registration

Cisco *live!*
DevNet Program



Learn online with Cisco Live!
Visit us online after the conference for full access to session videos and presentations.

www.CiscoLiveAPAC.com

Thank you

