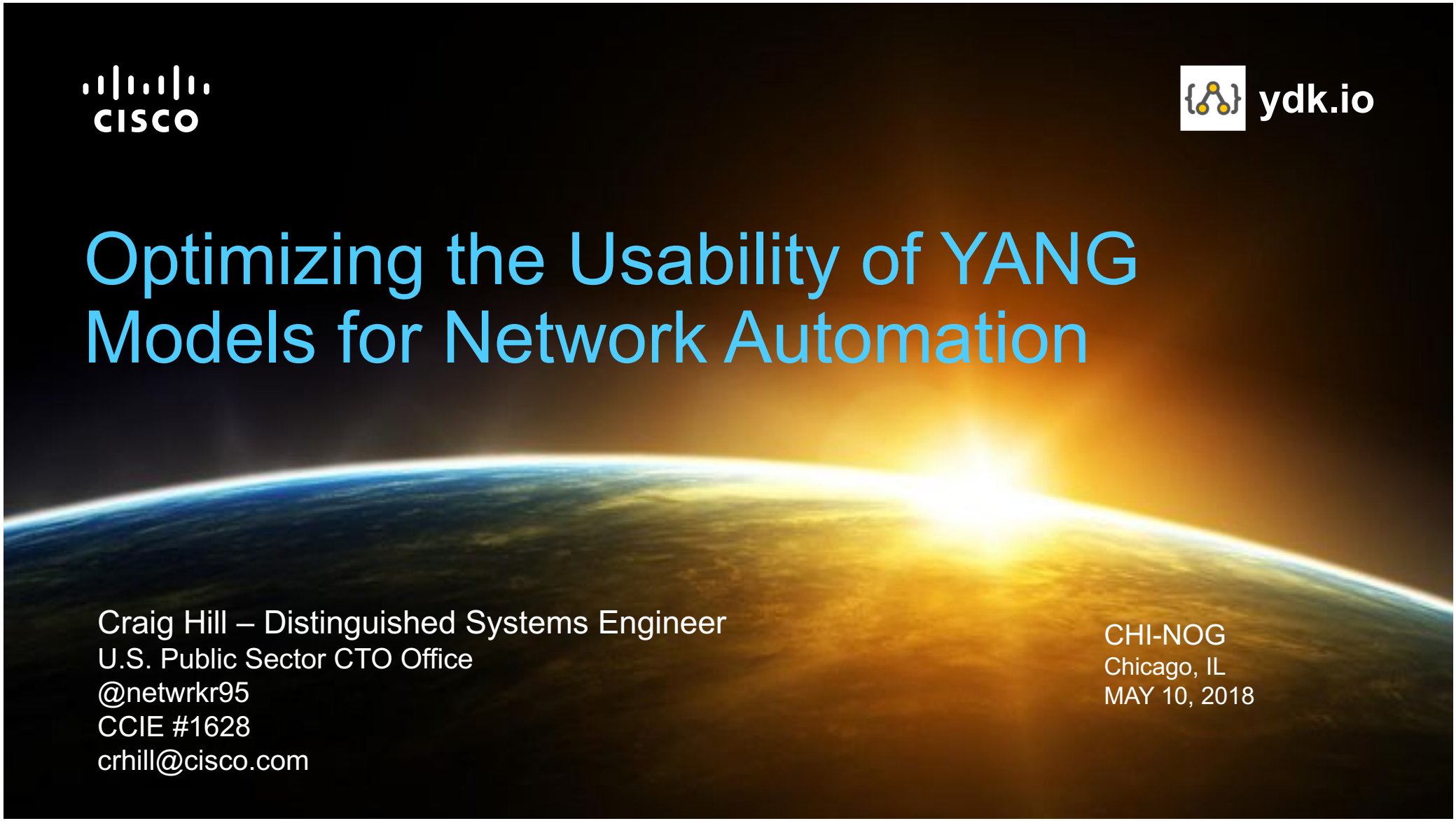


Optimizing the Usability of YANG Models for Network Automation

Craig Hill – Distinguished Systems Engineer
U.S. Public Sector CTO Office
@netwrkr95
CCIE #1628
crhill@cisco.com

CHI-NOG
Chicago, IL
MAY 10, 2018

Session Outline

- Open SDN Choices…
- The Shift to Model-driven programmability
- Moving Towards Model-Driven API's… What is YDK?
- Demo
- Summary and YDK Resource Locations

Choices are a Good Thing…

Open SDN Approach? Offer IT Organizations Choices

Prescriptive Turn-key SDN Solutions	"Open" Programmable Solutions with Vendor HW	"Open" Programmable Solutions for Multi-Vendor
Prescriptive Solution	**"Do it Yourself" Solution**	
• Targets less experience in-house	• Wants open-source options, with vendor specific hardware	• Customer desires mixed-vendor SDN and network environment
• Requires much less operational expertise.	• Leverage open standard solutions (models, protocols)	• Leverage open standard solutions (models, protocols)
• May require vendor specific HW in some areas	• Require in-depth operational programmability skills in-house	• Also requires in-house programming skills, and open standard data/control network
Mass Market (commercial, enterprises, public sector)	More skilled DevOps IT Org's (SP's, Hyper-Scale Cloud Providers)	More skilled DevOps IT Org's (SP's, Hyper-Scale Cloud Providers)

Open SDN Approach? Offer IT Organizations Choices

Prescriptive Turn-key SDN Solutions	"Open" Programmable Solutions with Vendor Specific HW	"Open" Programmable Solutions for Multi-Vendor

Prescriptive Solution

"Do it Yourself" Solution

- ***Examples (Vendor Solutions):***
  - *VMware - NSX*
  - *Cisco ACI (DC)*
  - *Cisco SD-WAN (Viptela/Meraki)*
  - *Juniper Contrail*

- ***Examples (open source):***
  - *Model-driven approach using YANG models (native, open)*
  - *Python (protocol libraries)*
  - *REST API*
  - *Other Tools: Ansible, Puppet, Chef, etc…*

- ***Examples (open source):***
  - ***Same as Column #2***

  +

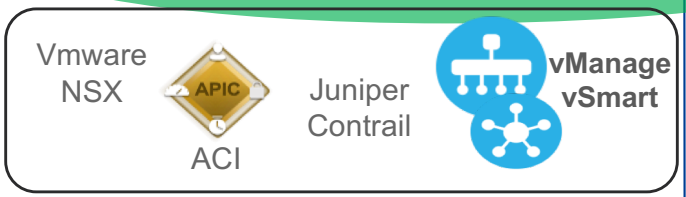
  - *IP/MPLS / Segment Routing*
  - *E-VPN (BGP) / VXLAN*
  - *OpenFlow*

Open SDN Approach? Offer IT Organizations Choices

Prescriptive Turn-key SDN Solutions	"Open" Programmable Solutions with Vendor Specific HW	"Open" Programmable Solutions for Multi-Vendor

Prescriptive Solution

"Do it Yourself" Solution



- ***Examples (Vendor Solutions):***
  - *VMware - NSX*
  - *Cisco ACI (DC)*
  - *Cisco SD-WAN (Viptela/Meraki)*
  - *Juniper Contrail*

- ***Examples (open source):***
  - *Model-driven approach using YANG models (native, open)*
  - *Python (protocol libraries)*
  - *REST API*
  - *Other Tools: Ansible, Puppet, Chef, etc…*

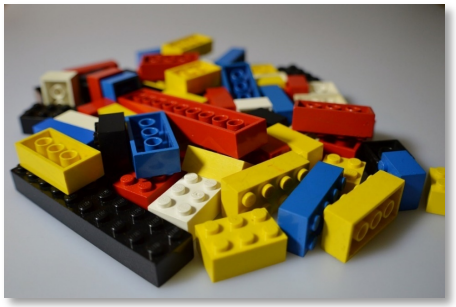
- ***Examples (open source):***
  - ***Same as Column #2***

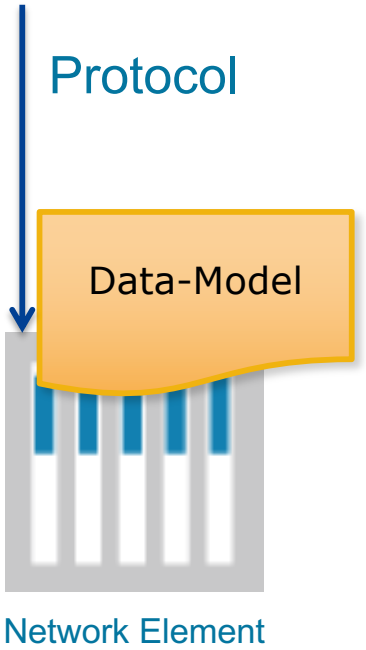
  +

  - *IP/MPLS / Segment Routing*
  - *E-VPN (BGP) / VXLAN*
  - *OpenFlow*

The Shift to Model-Driven Programmability

What is a Data Model?

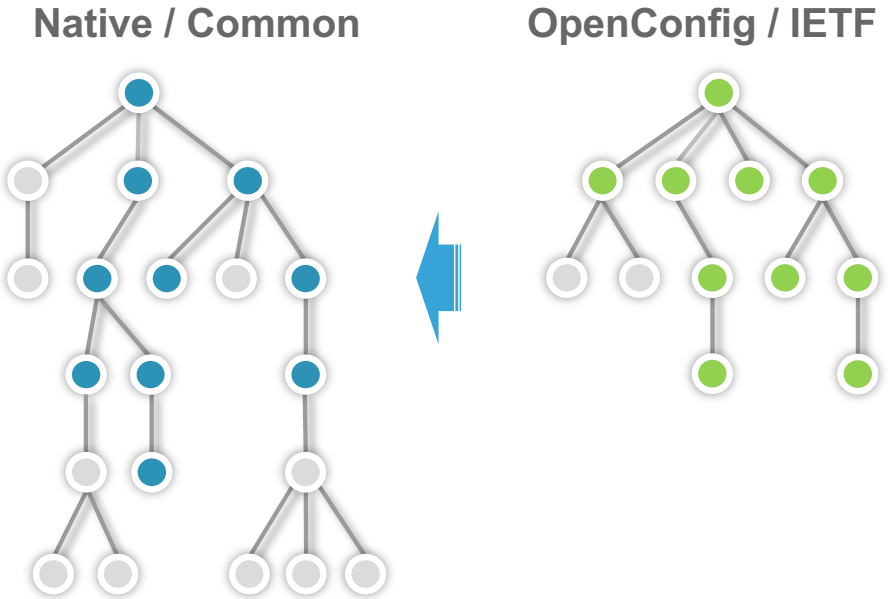




Data Model (YANG the focus here)

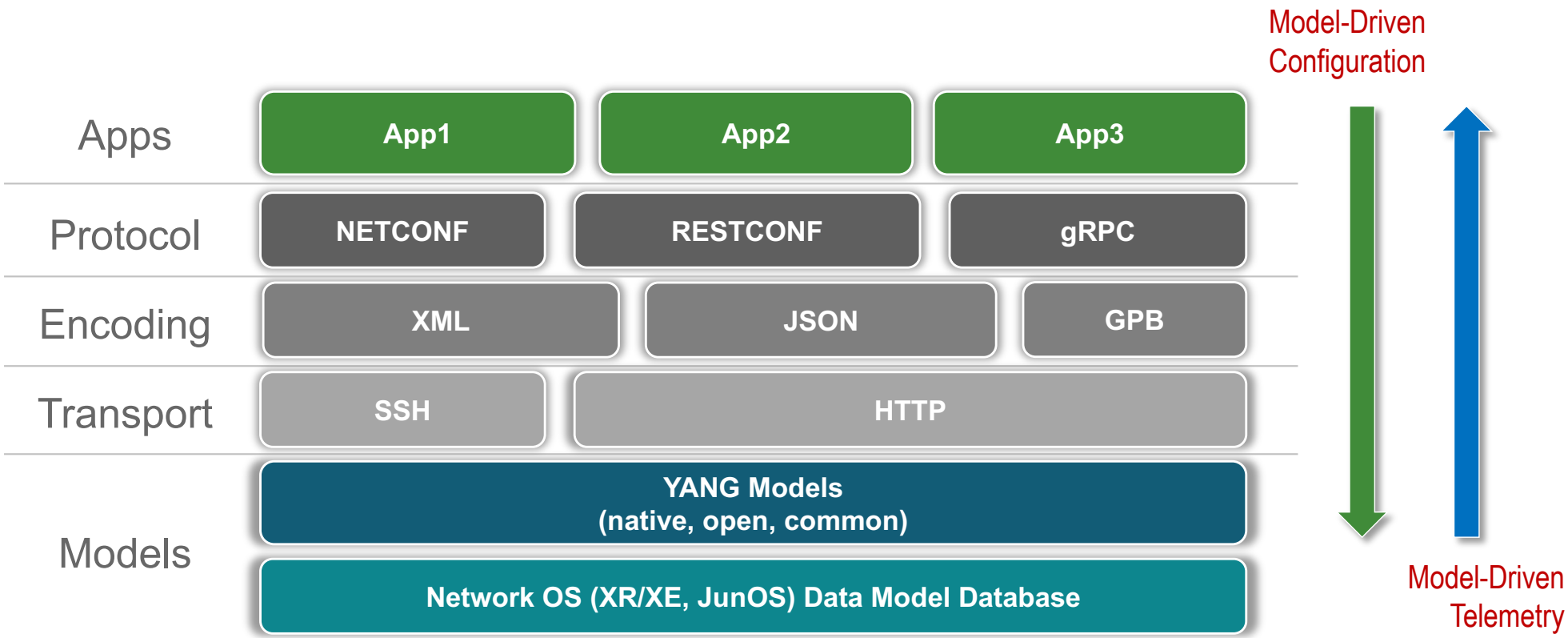
- A data model explicitly and precisely determines the structure, syntax and semantics of the data…
- Consistent and complete
- The data model is highly structured (tree structure)
- The data model has no understanding of hardware, device (physical/virtual), transport
- Associated configuration management protocol (NETCONF, gRPC) to encode the data as defined by the model

Data Models



- Data (config and operational) and actions (RPCs) in a tree structure
- Self-documented and shipped with devices
- Native models provide most coverage
- OpenConfig and IETF models are mapped to native models
- Details of YANG as a modeling language is in RFC 6020, 7950 (YANG 1.1)

Model-Driven Programmability Stack



© 2018 Cisco and/or its affiliates. All rights reserved. - Cisco Public

Benefits of Model-Driven Programmability

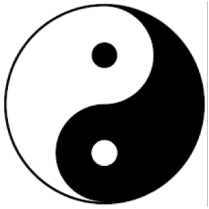
- Model based, highly structured, computer friendly
- Multiple model types (native, OpenConfig, IETF, etc.)
- Models decoupled from transport, protocol and encoding
- Choice of transport, protocol and encoding
- Model-driven APIs for abstraction and simplification
- Wide standard support while leveraging open source

http://ydk.io/

Shift to Model-Driven APIs

What is the YANG Development Kit (YDK)?

Two User Profiles for Network Programmability



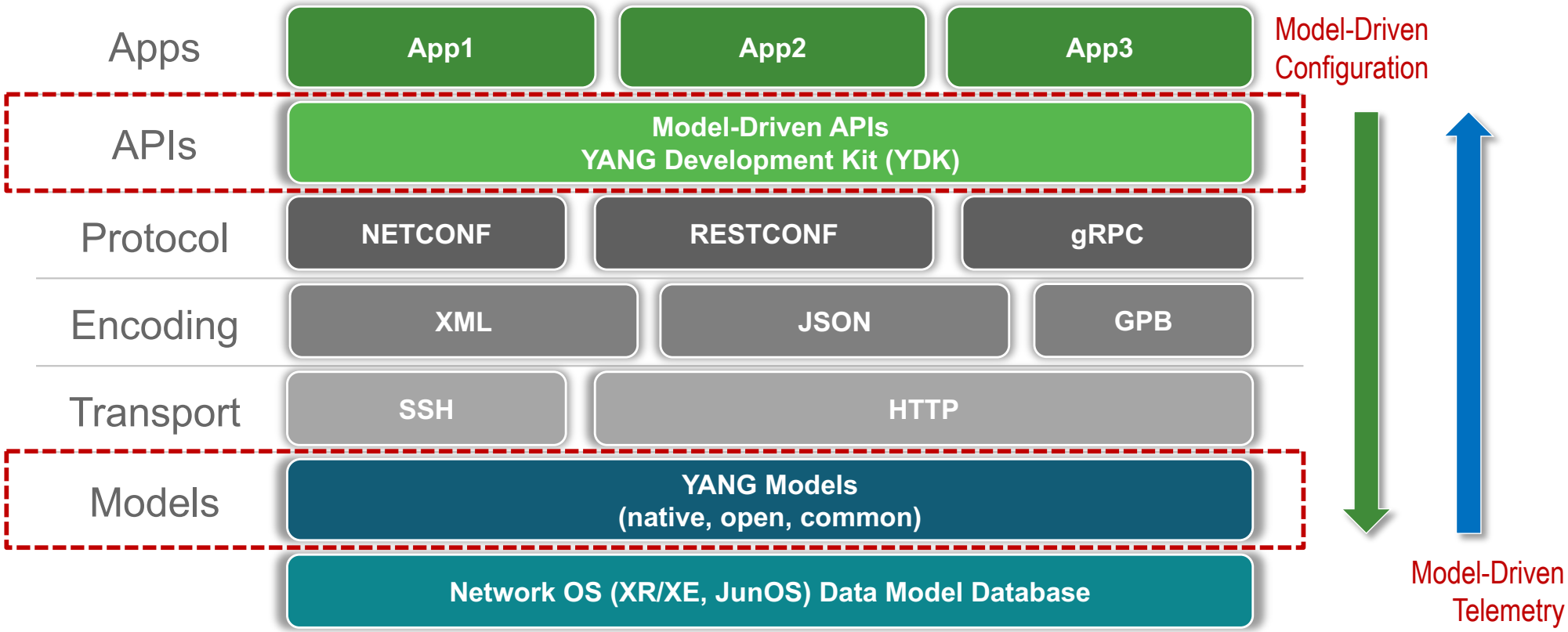
Network Engineer

- **Skills**
  - Proficient in network protocols and network management
  - No or minimal programming experience
- **Requires**
  - Simple programming abstractions
  - Avoid programming complexities of management protocols, encodings, transport and YANG

Software Developer

- **Skills**
  - Proficient in software development and automation
  - No or minimal experience with network protocols and network management
- **Requires**
  - Software development kit
  - Avoid learning curb of management protocols, encodings, transport and YANG

Model-Driven Programmability Stack (w/ YDK)

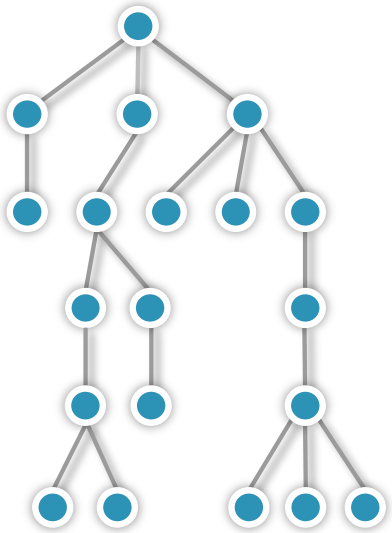


© 2018 Cisco and/or its affiliates. All rights reserved. - Cisco Public

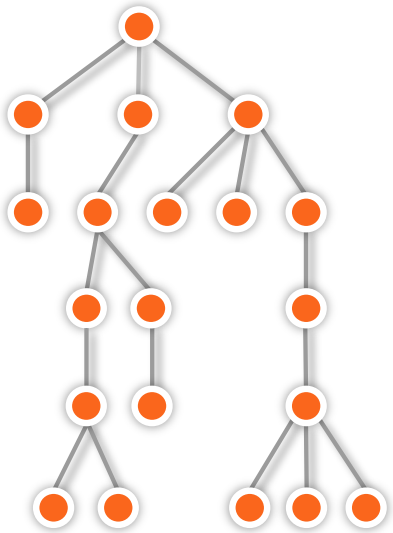
YDK – Offers Model-Driven APIs, Generated from the YANG Model

- Simplify app development
- Abstracts transport, protocol, encoding, modeling language
- API is generated directly from the YANG model
- Rich data validation
- One-to-one correspondence between model and class hierarchy
- Multi-language (Python, C++, Go, etc.)

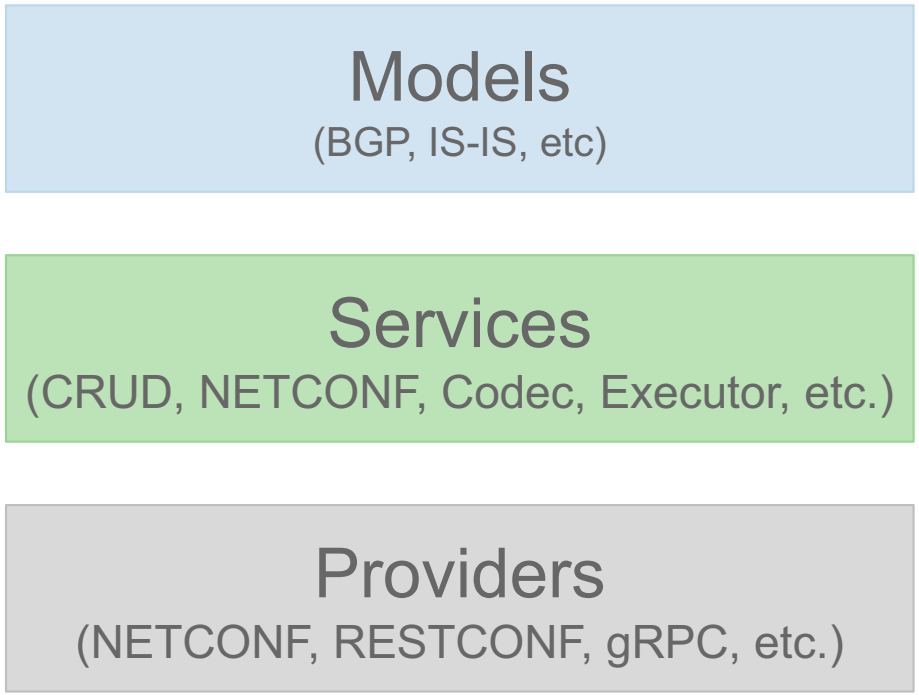
YANG Model



Class Hierarchy (Python, C++, Go)



YDK - API Structure



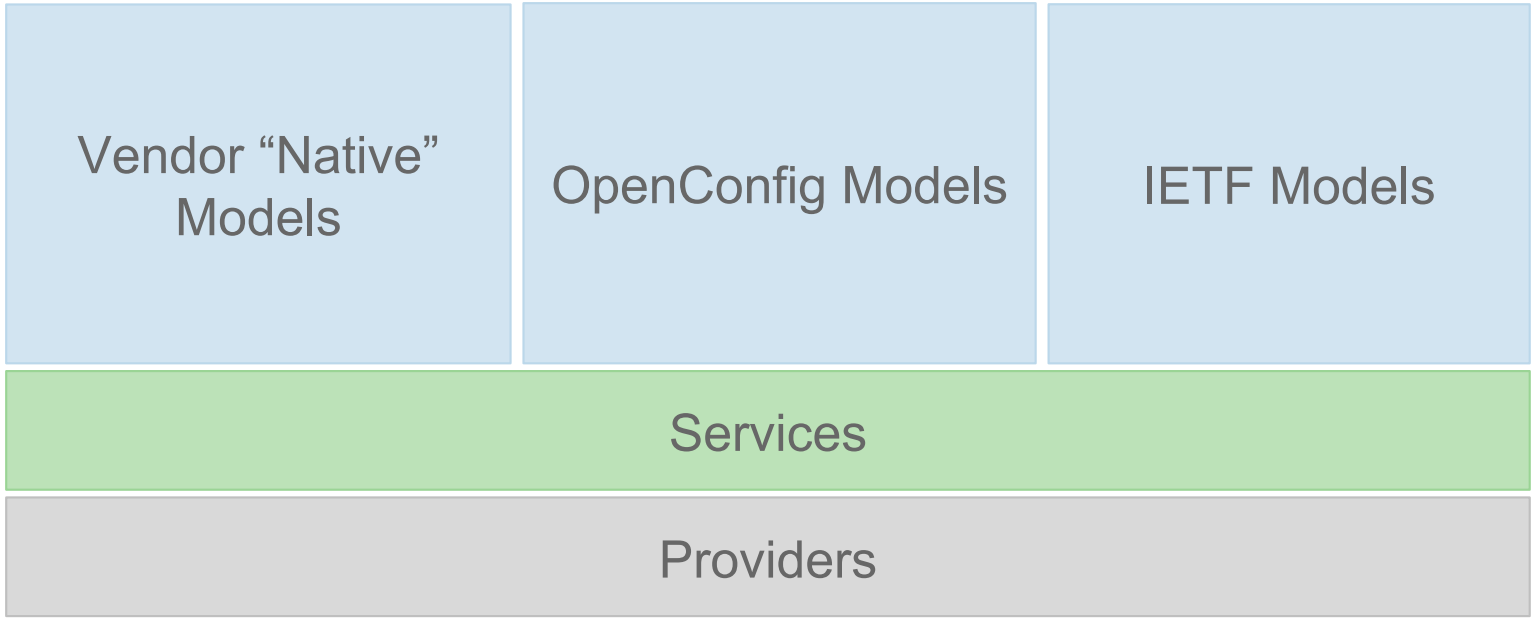
- **Models** group Python APIs created for each YANG model
- **Services** perform operations on the model objects (interface)
- **Providers** implement services (implementation)

YANG Development Kit – Multi-Language

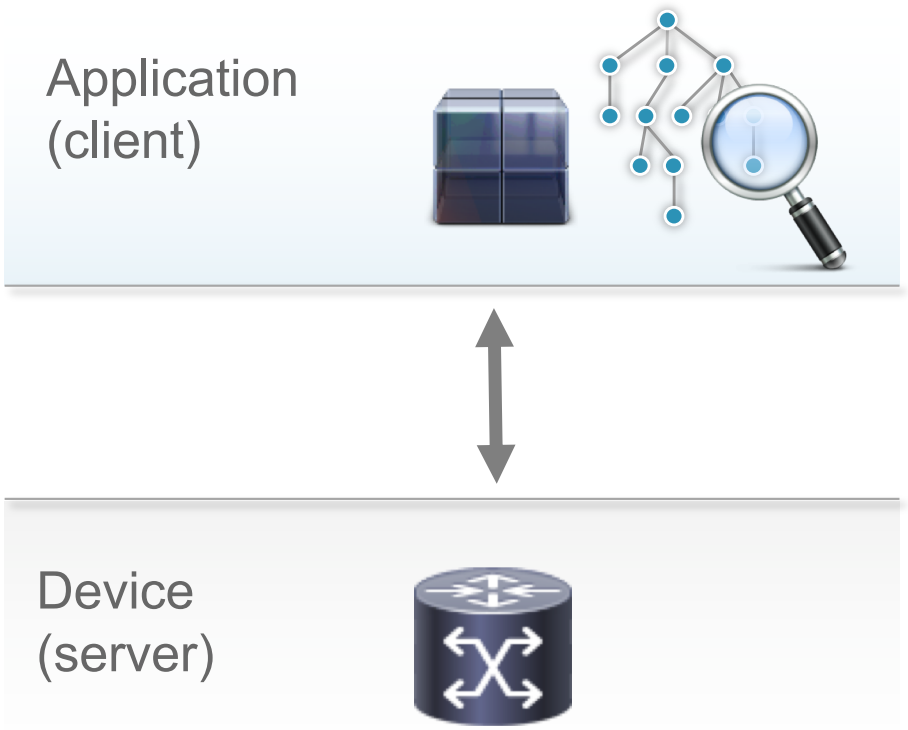
YDK-Py
Python

YDK-Cpp
C++

YDK-Go
GoLang



YDK Client-Side Validation



- Client will automatically perform **local validation** based on model constraints
  - Check between **type of data**: config (read-write) and state (read-only)
  - **Type** check (enum, string, etc.)
  - **Value** check (range, pattern, etc.)
  - **Semantic** check (key uniqueness/presence, mandatory leafs, etc.)
  - Model **deviation** check (unsupported leafs, etc.)
- Validation done BEFORE transaction is sent to box

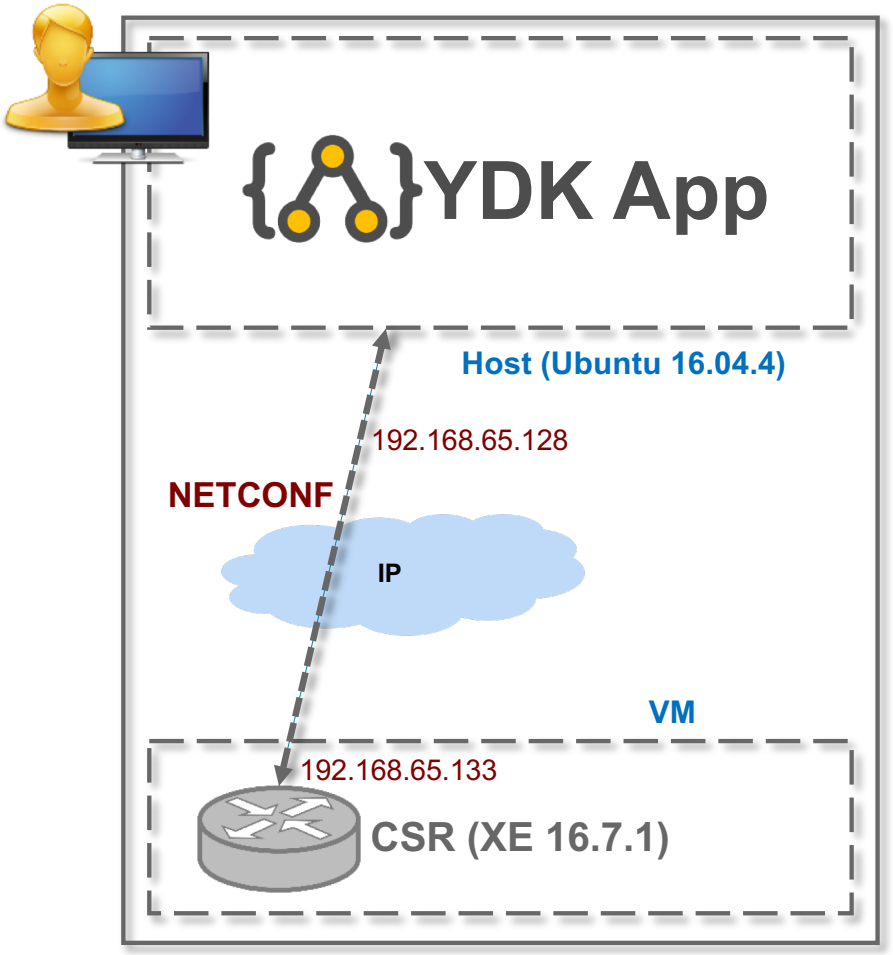
Demo

YDK.io

YDK (Demo)

Adding Interface IPv4 with VRF Forwarding

- YDK 0.7.1 (Linux Ubuntu 14.04)
- Cisco CSR (XE 16.7.1)
- Demo executed against the YANG model in IOS-XE for “native/interface”
- YDK Application generated for “interface” run from Host, adding “interface”:
  - Interface name
  - Description
  - IP address / mask
  - VRF forwarding name
  - No shut interface



GitHub Repository to Example:
https://git.io/vp681

Model Data Example

XML

```
<Loopback>
  <name>99</name>
  <description> IPv4 VRF with Route Target</description>
  <vrf>
   <forwarding>VRF1</forwarding>
  </vrf>
  <ip>
   <address>
    <primary>
     <address>10.99.99.99</address>
     <mask>255.255.255.255</mask>
    </primary>
   </address>
  </ip>
</Loopback>
```

Corresponding YANG Model

```
module: Cisco-IOS-XE-native
   +--rw native
      +--rw interface
         +--rw Loopback* [name]
            +--rw name
            +--rw description?
            +--rw ip
               +--rw address
                  …

            +--rw vrf
               +--rw forwarding
                  +--rw word?
```

(Output from 'pyang -f tree… ')

YDK-Py Interface with VRF Example

Python Code for Populating Parameters

```
. . .

def config_native(native):
    """Add config data to native object."""
    loopback = native.interface.Loopback()
    loopback.name = 99
    loopback.description = " IPv4 VRF with Route Target "
    loopback.ip.address.primary.address = "10.99.99.99"
    loopback.ip.address.primary.mask = "255.255.255.255"

    # Adding VRF to configuration
    loopback.vrf.forwarding = "VRF1"
    native.interface.loopback.append(loopback)

. . .
```

Corresponding YANG Model

```
module: Cisco-IOS-XE-native
   +--rw native
      +--rw interface
         +--rw Loopback* [name]
            +--rw name
            +--rw description?
            +--rw ip
               +--rw address
                  …

            +--rw vrf
               +--rw forwarding
                  +--rw word?
```

(Output from 'pyang -f tree…')

YDK-Py Interface with VRF (output)

Python Code for Populating Parameters

```
. . .

def config_native(native):
    """Add config data to native object."""
    loopback = native.interface.Loopback()
    loopback.name = 99
    loopback.description = " IPv4 VRF with Route Target "
    loopback.ip.address.primary.address = "10.99.99.99"
    loopback.ip.address.primary.mask = "255.255.255.255"

    # Adding VRF to configuration
    loopback.vrf.forwarding = "VRF1"
    native.interface.loopback.append(loopback)

. . .
```

CLI Output (IOS-XE)

```
!
interface Loopback99
 description IPv4 VRF with Route Target
 vrf forwarding VRF1
 ip address 10.99.99.99 255.255.255.255
!
```

Recap and Key Takeaways

Summary and Key Takeaways

- Model driven approach offers highly structured, machine-friendly approach to device configuration (and model-driven telemetry)

Yang Development Kit (YDK)…

- Is open source
- Targets the Simplification of app development
- Abstracts protocol, transport, encoding, and modeling language
- Generates the API's from YANG model (native / open)
- Offers rich "local" data validation
- Offers a rich set of services and providers
- Multi-language capable (Python, C++, Go, etc.)

http://ydk.io/

YDK

Resources

How to Get YDK-Py

Native

Install Python
Install YDK
Download ydk-py-samples

Virtual

Install Vagrant
Install Virtualbox
Download ydk-py-samples

Cloud

YANG Development Kit Sandbox

Resources

YDK Portal

- YDK at DevNet (http://ydk.io)

YDK Sample Apps

- YDK-Py sample apps (https://github.com/CiscoDevNet/ydk-py-samples) - Over 700 apps!
- YDK-Cpp sample apps (https://github.com/CiscoDevNet/ydk-cpp-samples) - Coming soon

Sandboxes

- dCloud YANG Development Kit sandbox (https://goo.gl/RPpBvL)
- Ubuntu YDK Vagrant box (https://git.io/vaw1U)

Support

- Cisco support community (https://communities.cisco.com/community/developer/ydk)

Resources (cont.)

YDK Documentation

- YDK-Py docs (http://ydk.cisco.com/py/docs)
- YDK-Cpp docs (http://ydk.cisco.com/cpp/docs)
- YDK-Go docs (http://ydk.cisco.com/go/docs/)

GitHub

- YDK Python API – YDK-Py (https://git.io/vaWsg)
- YDK-Py sample apps (https://git.io/vaw1U)
- YDK C++ API – YDK-Cpp (https://git.io/v1Cst)
- YDK-Cpp sample apps (https://git.io/v14Qh)
- YDK-Go API – YDK-Go (https://git.io/vp6lu)

Resources (cont.)

Conferences

- NANOG 68: Ok, We Got YANG Data Models. Now What? (http://youtu.be/2oqkiZ83vAA)
- NANOG 71: Getting started with OpenConfig (https://youtu.be/L7trUNK8NJI)
- LinuxCon NA 2016: Simplifying Network Programmability Using Model-Driven APIs (https://goo.gl/W6tH2X)

CISCO