You make **possible**

Model-Driven Programmability for Cisco IOS XR

Santiago Álvarez
Technical Marketing Engineer
@111pontes

BRKSPG-2303

Barcelona | January 27-31, 2020

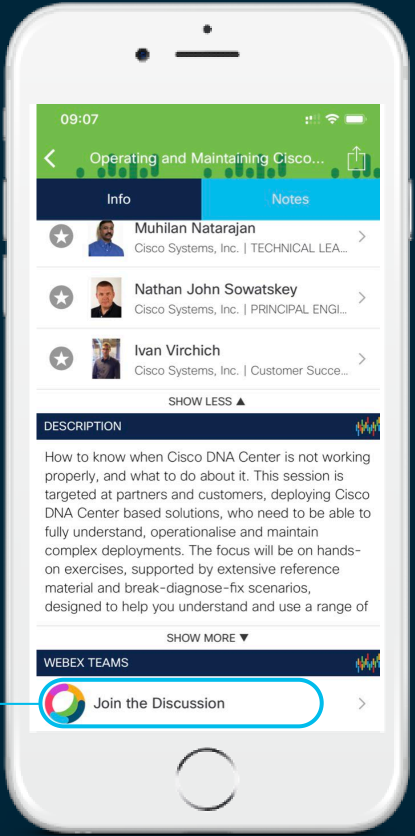
Cisco Webex Teams

Questions?

Use Cisco Webex Teams to chat
with the speaker after the session

How

1. Find this session in the Cisco Events Mobile App
2. Click “Join the Discussion”
3. Install Webex Teams or go directly to the team space
4. Enter messages/questions in the team space



09:07

Operating and Maintaining Cisco...

Info | Notes

Muhilan Natarajan
Cisco Systems, Inc. | TECHNICAL LEA...

Nathan John Sowatskey
Cisco Systems, Inc. | PRINCIPAL ENGI...

Ivan Virchich
Cisco Systems, Inc. | Customer Succe...

SHOW LESS ▲

DESCRIPTION

How to know when Cisco DNA Center is not working properly, and what to do about it. This session is targeted at partners and customers, deploying Cisco DNA Center based solutions, who need to be able to fully understand, operationalise and maintain complex deployments. The focus will be on hands-on exercises, supported by extensive reference material and break-diagnose-fix scenarios, designed to help you understand and use a range of

SHOW MORE ▼

WEBEX TEAMS

Join the Discussion

Agenda

- Introduction
- Data Models
- Management Protocols
- Model-Driven SDK
- Telemetry

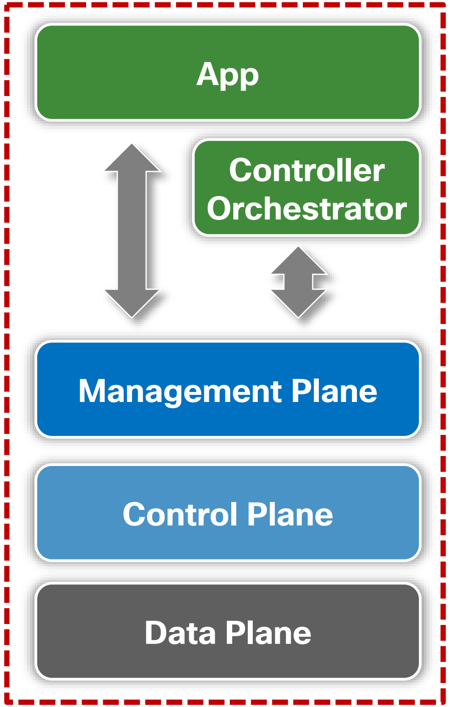
Introduction

Motivations for Network Programmability

- **Speed** and **scale** demand software automation and data analytics
- **Rapid innovation** as competitive advantage
- One network operator per 1000s / 10000s of complex network devices

Cisco IOS XR Device Programmability

Model-Driven Manageability



App

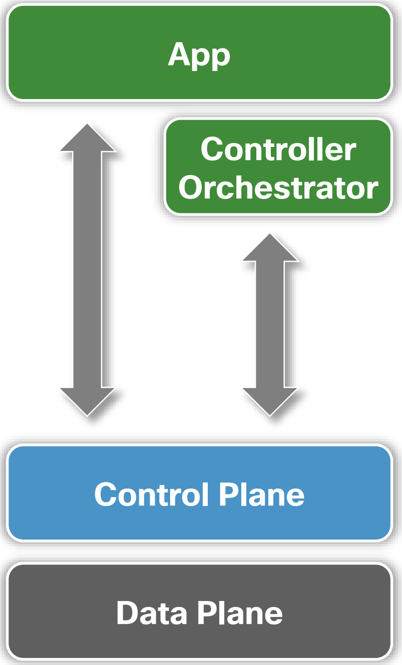
Controller Orchestrator

Management Plane

Control Plane

Data Plane

Service Layer API



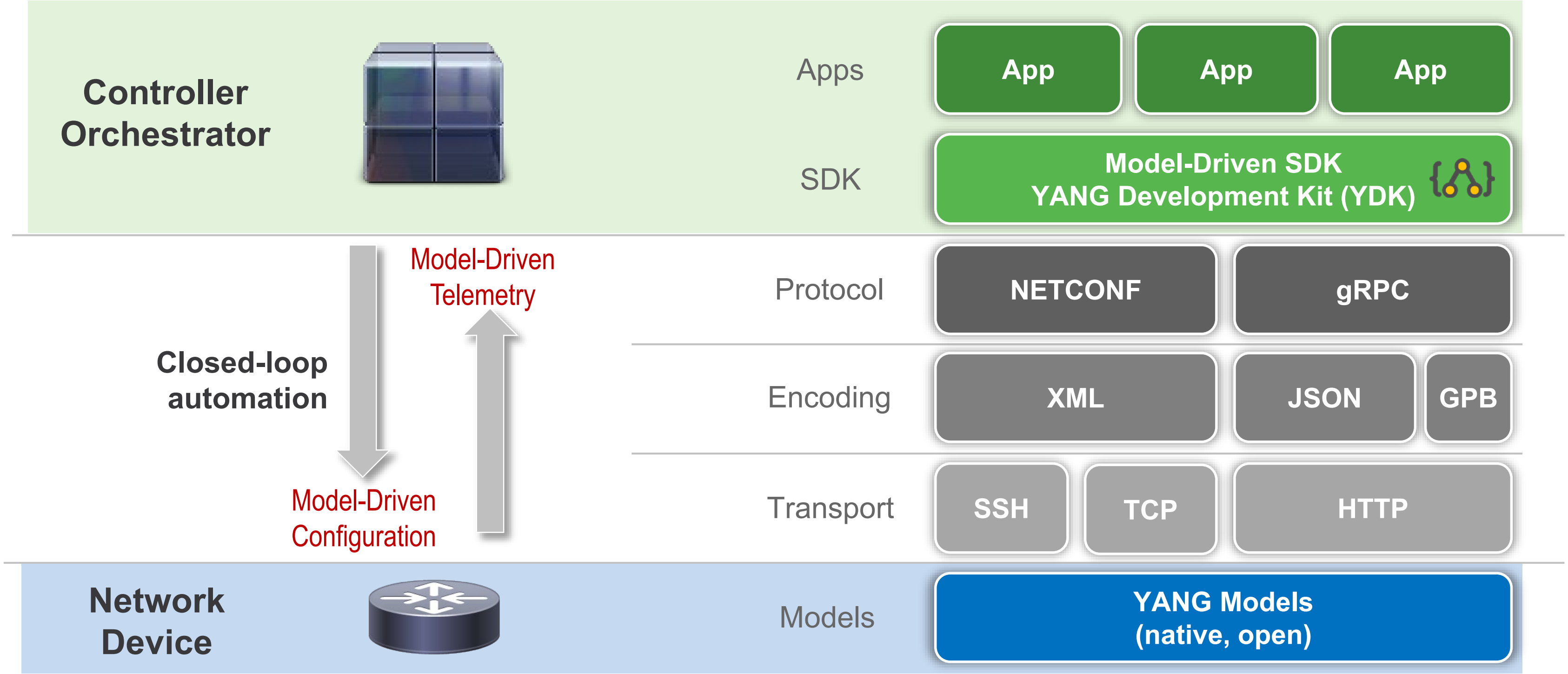
App

Controller Orchestrator

Control Plane

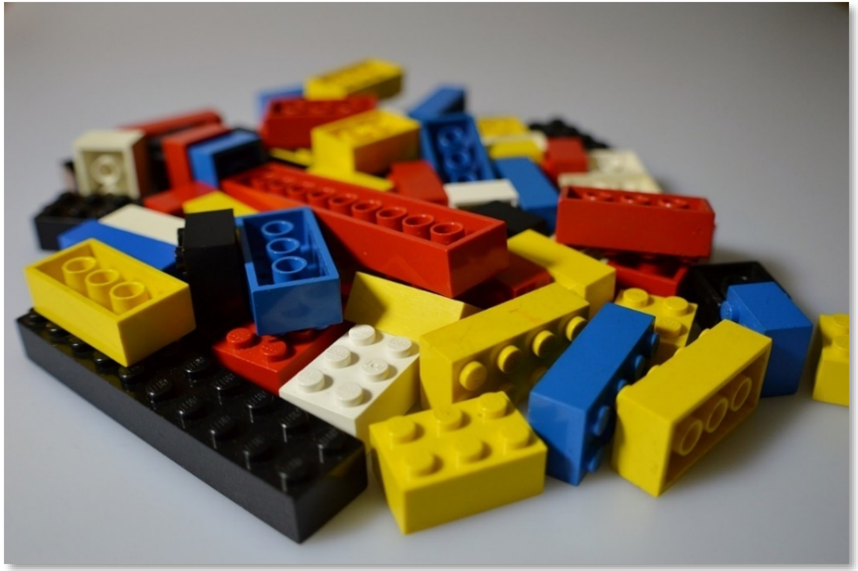
Data Plane

Model-Driven Manageability



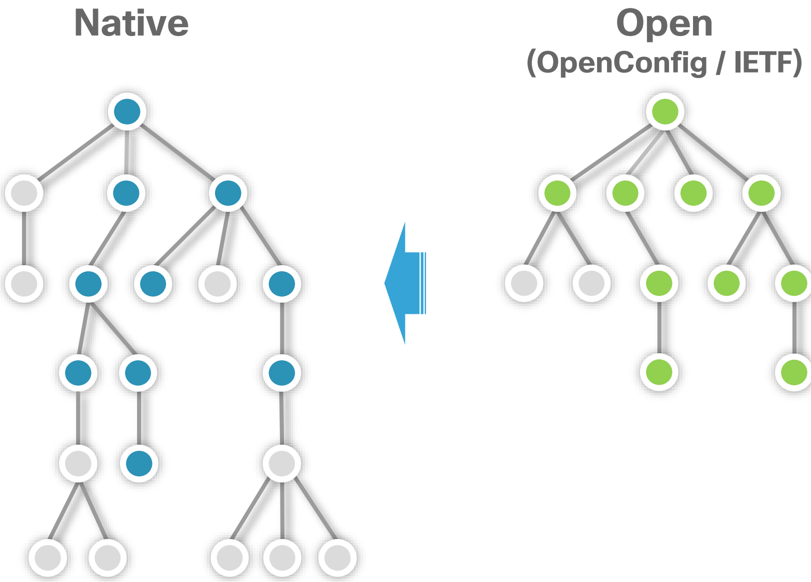
Benefits of Model-Driven Manageability

- Model based, 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 SDKs for **abstraction** and **simplification**
- Wide standard support while leveraging open source



Data Models

Data Models in Cisco IOS XR

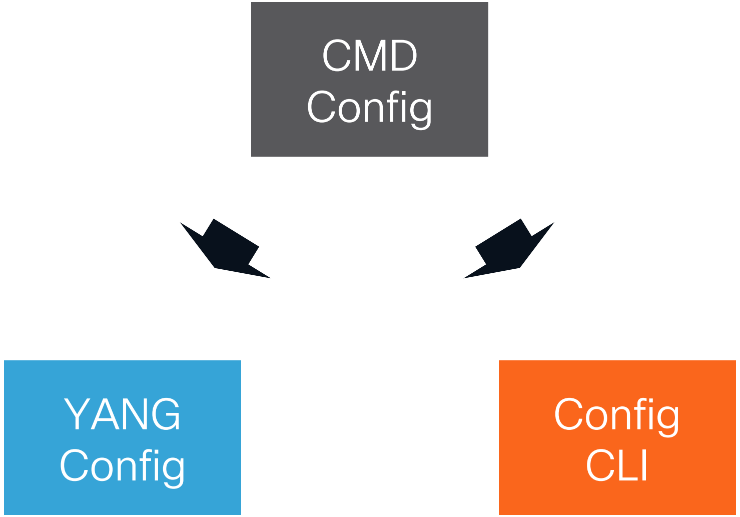


- Data (config and operational) and actions/commands (RPCs) in a tree structure
- Self-documented and shipped with devices
- Native (XR specific) and open (vendor neutral) models
- Native models provide most coverage
- Open (OpenConfig and IETF) provide reduced coverage
- Open models internally mapped to native models

Cisco IOS XR Native Data Models

- Provide most comprehensive coverage for device functionality
- Approximately ~500 models in XR 7.0.1 (1000+ YANG files)
- A single model defines either configuration (cfg), operational state (oper) or an action/command (act)
  - Cisco-IOS-XR-um-router-bgp-cfg
  - Cisco-IOS-XR-ipv4-bgp-oper
  - Cisco-IOS-XR-ipv4-bgp-act
- Models posted at
  - https://github.com/YangModels/yang/tree/master/vendor/cisco/xr

Unified Config Definition

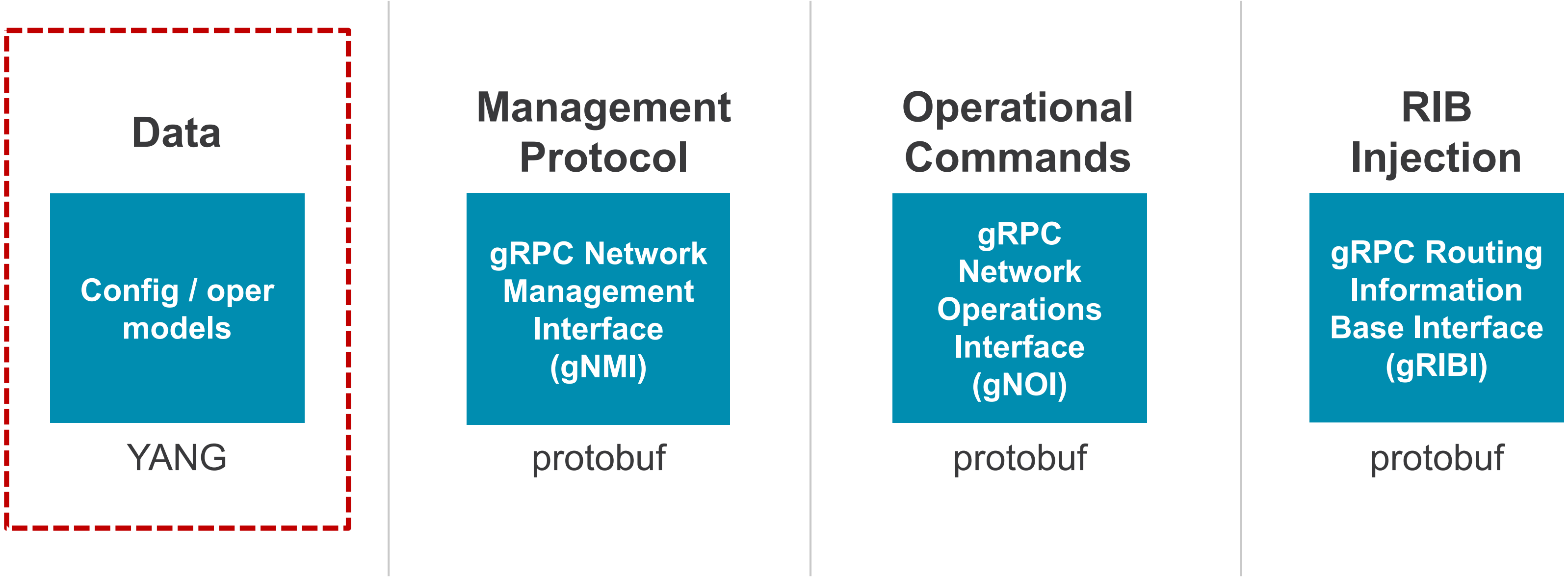


- Single config definition
- Same abstraction for YANG and CLI
- Full parity and deterministic coverage
- Same help/doc strings
- Simpler translation between config abstractions
- YANG file names start with “Cisco-IOS-XR-um”

Unified Configuration Models (Phase 1)

7.0.1	7.1.1	7.2.1
Interfaces Bundles ARP LACP VRF Static routing RIB MPLS (LDP, LSD, L3VPN) Telemetry NETCONF gRPC SNMP	BGP ISIS OSPF (v2/v3) MPLS (TE) RSVP	QoS ACL (IPv4, IPv6, Ethernet, prefix list, object group) Multicast (AMT, IGMP, MLD, MSDP, PIM)

OpenConfig Major Components



Data

Config / oper models

YANG

Management Protocol

gRPC Network Management Interface (gNMI)

protobuf

Operational Commands

gRPC Network Operations Interface (gNOI)

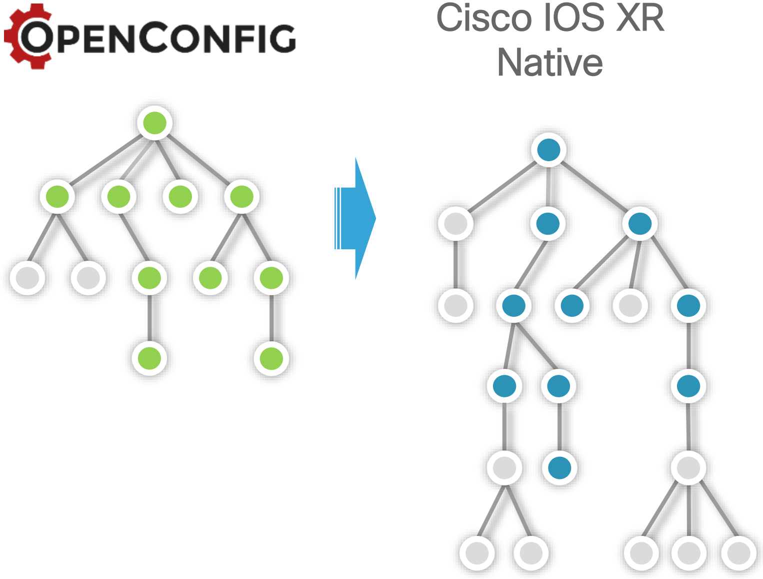
protobuf

RIB Injection

gRPC Routing Information Base Interface (gRIBI)

protobuf

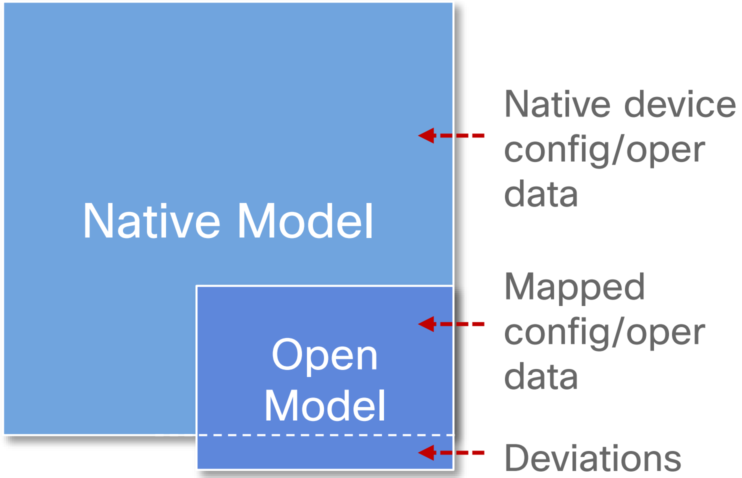
OpenConfig Data Models In Cisco IOS XR (7.0.1)



openconfig-acl
openconfig-bgp-policy
openconfig-bgp
openconfig-channel-monitor
openconfig-interfaces
openconfig-if-aggregate
openconfig-if-ethernet
openconfig-if-ip
openconfig-vlan
openconfig-lacp
openconfig-lldp
openconfig-network-instance
openconfig-local-routing
openconfig-isis
openconfig-isis-policy

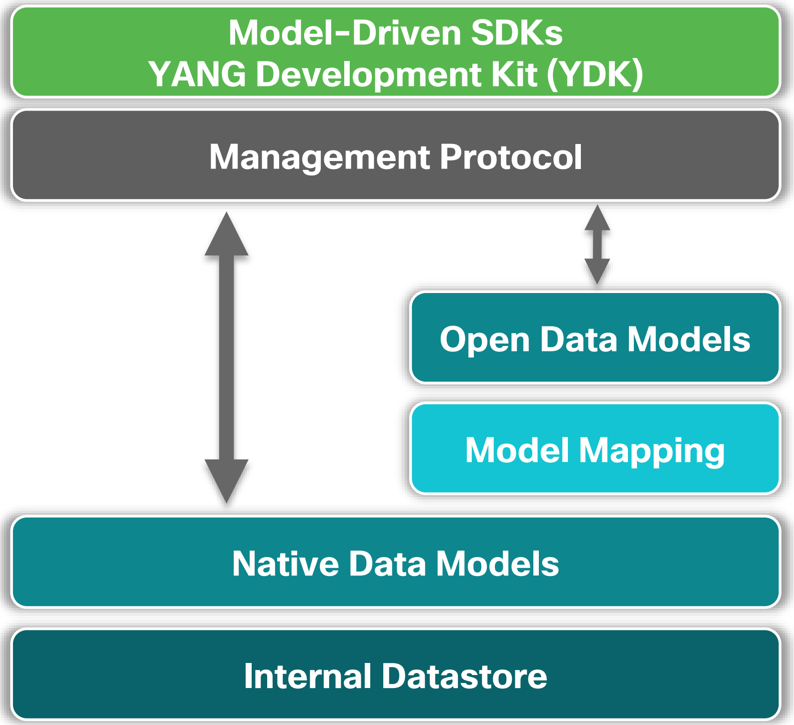
openconfig-rib-bgp
openconfig-routing-policy
openconfig-mpls
openconfig-rsvp-sr-ext
openconfig-aft
openconfig-aft-network-instance
openconfig-platform
openconfig-platform-port
openconfig-system
openconfig-telemetry
openconfig-terminal-device
openconfig-transport-line-common
openconfig-transport-line-protection
openconfig-optical-amplifier

Native vs Open Data Models



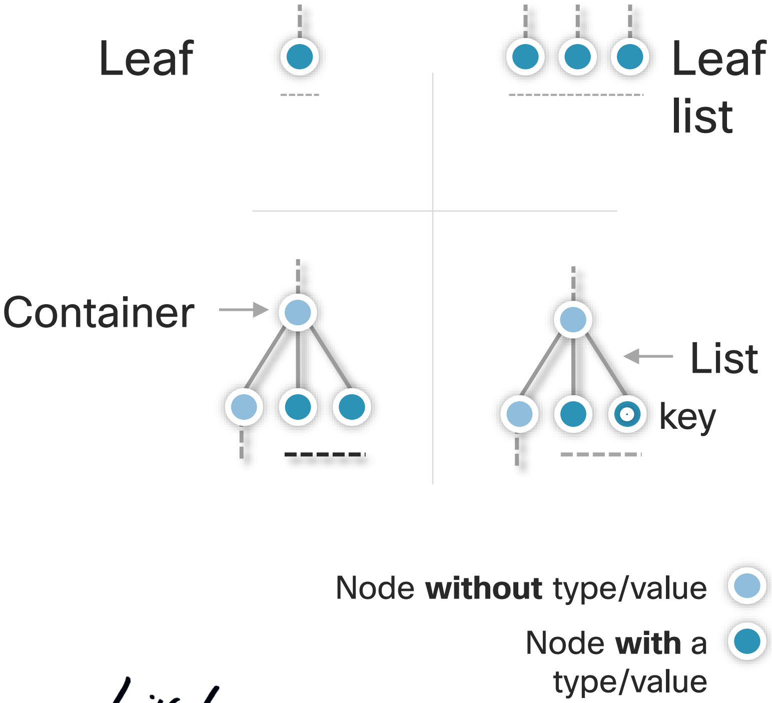
- Native data models provide most configuration and operational coverage
- Open models mapped to native data models
- Departures from open models specified as deviation module

Open Model Mapping in IOS XR



- Model mapping converts open model data to native model data and vice versa
- Support for mapping config and operational data (including telemetry)
- Single view of config and operational data in internal datastore
- Planning to enable user defined mapping (Model Mapping SDK)

YANG

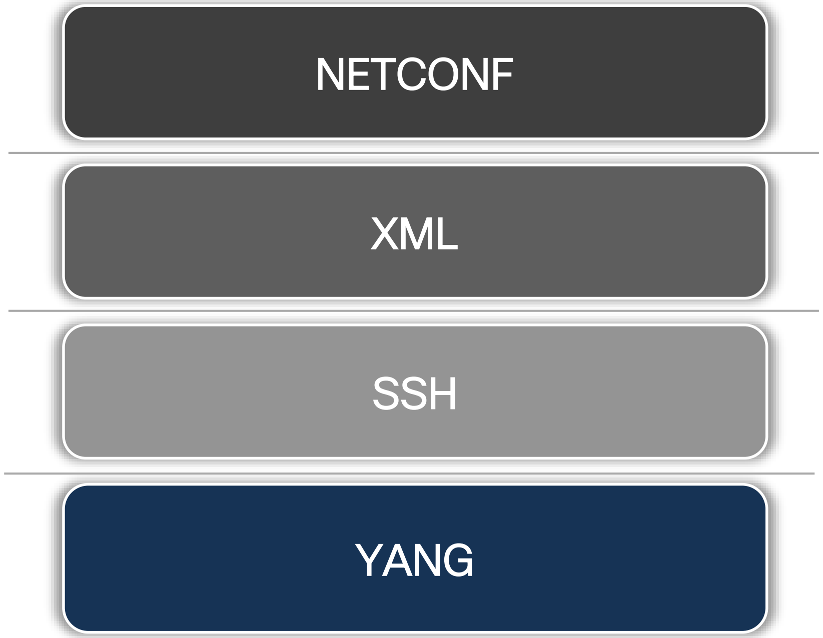


- Modeling language for networking
- Defines data hierarchy (config or oper), RPCs and notifications
- Main node types
  - **Leaf** – node with name, type and value (no children)
  - **Leaf list** – sequence of leafs (no children)
  - **Container** – node that groups nodes and has no type or value
  - **List** – Series of data instances generally with one or more keys
- Models extended through augmentations
- Unsupported nodes specified as deviations

Management Protocols

NETCONF Protocol Overview

- Rich functionality to manage configuration and operational (state) data
- Operations defined as RPCs (request / reply) in XML
- Client/app initiate request towards server/device
- Supports running, candidate and startup configurations
- Capability exchange during session initiation



Main NETCONF Protocol Operations

Operation	Description
get-config	Retrieve all or part of a specified configuration
edit-config	Loads all or part of a specified configuration (merge, replace, create, delete, remove)
copy-config	Create or replace an entire configuration datastore
get	Retrieve all or part of running configuration and device operational data
get-schema	Retrieve device schema (model)
lock	Lock entire configuration datastore (e.g. candidate)
unlock	Remove lock on entire configuration datastore (e.g. candidate)
close-session	Request graceful session termination

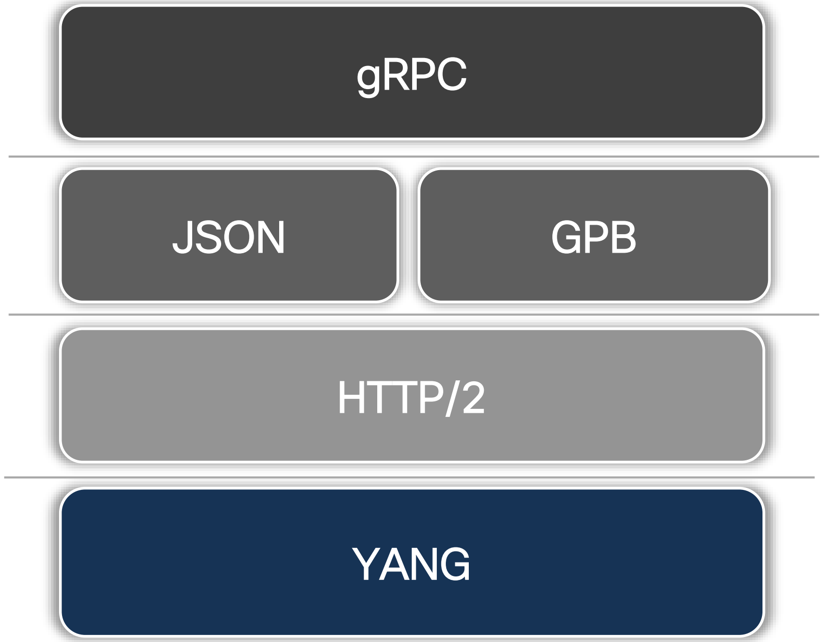
NETCONF Edit-Config Operations

Operation	Description
Merge	Merge configuration with existing configuration (default)
Replace	Replace configuration with existing configuration
Create	Create configuration if non-existent. Otherwise, return error. (non-idempotent*)
Delete	**Delete configuration if non-existent. Otherwise, return error. (non-idempotent)**
Remove	Remove configuration. Ignore if configuration non-existent.

* Cannot be applied multiple times without changing the result beyond the initial application

Overview of gRPC on Cisco IOS XR

- Google RPC provides a general (open source) RPC framework
- Two interface definitions
  - Cisco IOS XR
  - OpenConfig gNMI
- Combines configuration management and Telemetry
- Rich development toolchain
- High performance



Protocol Operations in Cisco IOS XR Interface

Operation	Description
GetConfig	Retrieve configuration
MergeConfig	Merge configuration
DeleteConfig	Delete configuration
ReplaceConfig	Replace configuration
CommitReplace	Replace entire configuration
GetOper	Retrieve operational data
CliConfig	Merge configuration data in CLI format
ShowCmdTextOutput	Retrieves CLI show-command output data

Protocol Operations in OpenConfig gNMI Interface

Operation	Description
capabilities	Discover device capabilities (models, encodings, version, extensions)
get	Retrieve device state (all, config, state or operational)
set	Modify device state (delete, replace, update)
subscribe	Subscribe to device update

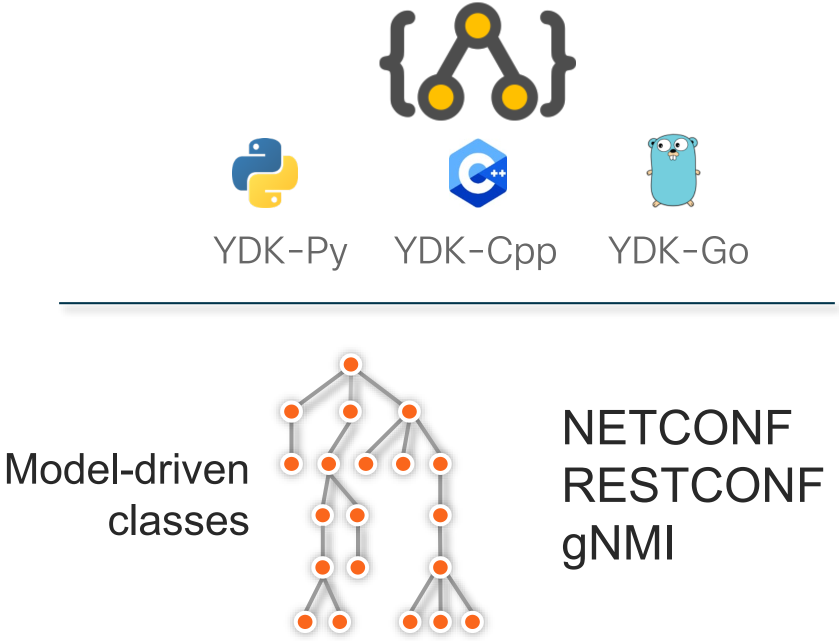
gNMI Implementation in Cisco IOS XR

- Based on gNMI v0.4.0
- Introduced in release 6.5.1
- Set and Get RPCs use JSON_IETF (RFC 7951) and ASCII (CLI) encoding
- Subscribe RPC
  - Paths must consider data aggregation points (no arbitrary paths)
  - No aliases

Model-Driven SDK

YANG Development Kit

- SDK simplifying client development for model-driven programmability
- Rich protocol support (NETCONF, RESTCONF, gNMI)
- Rich data model support (XR, XE, NX-OS, OC, IETF)
- Rich language support (Python, Go, C++)
- Built-in model data validation
- Open source



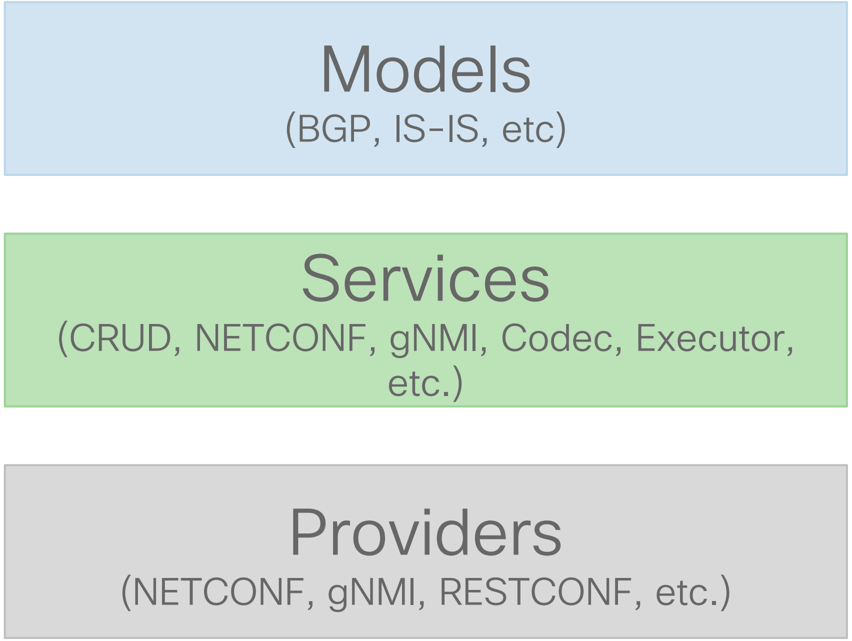
YDK-Py YDK-Cpp YDK-Go

Model-driven classes

NETCONF
RESTCONF
gNMI

ydk.io

YDK API Structure



Models

(BGP, IS-IS, etc)

Services

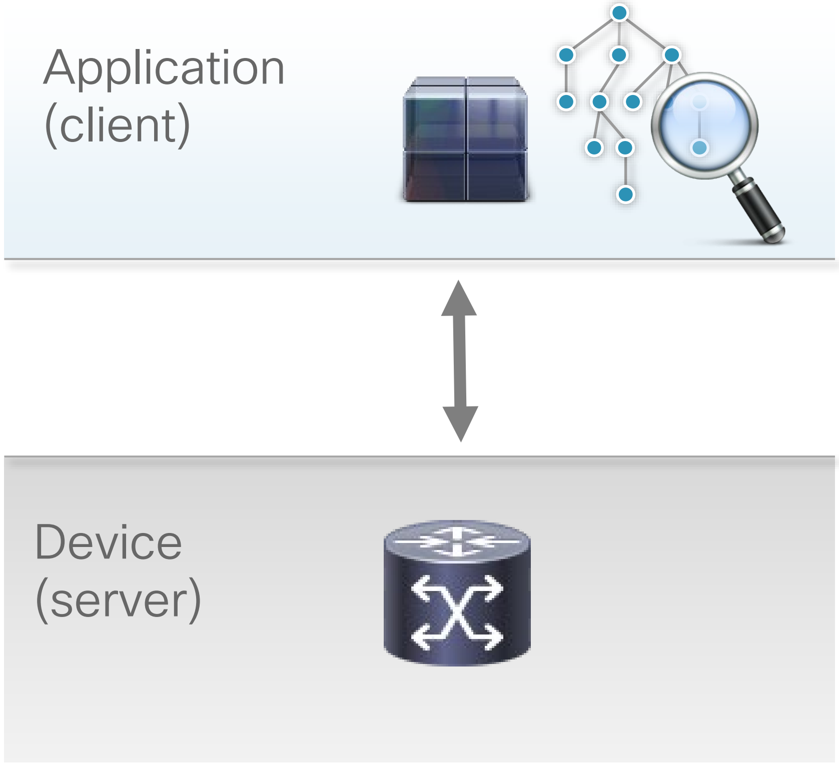
(CRUD, NETCONF, gNMI, Codec, Executor, etc.)

Providers

(NETCONF, gNMI, RESTCONF, etc.)

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

YDK Client-Side Validation



Application (client)

Device (server)

- 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.)

A YDK-Py “Hello World” Using OpenConfig BGP

```
# Cisco YDK-Py OC-BGP “Hello world”
from ydk.services import CRUDService
from ydk.providers import NetconfServiceProvider
from ydk.models.openconfig import openconfig_bgp as oc_bgp

if __name__ == "__main__":
    provider = NetconfServiceProvider(address=10.0.0.1,
                                      port=830,
                                      username=“admin”,
                                      password=“admin”,
                                      protocol=“ssh”)
    crud = CRUDService()  # create CRUD service
    bgp = oc_bgp.Bgp()  # create oc-bgp object
    bgp.global_.config.as_ = 65000  # set local AS number
    crud.create(provider, bgp)  # create on NETCONF device
    exit()
# End of script
```

```
module: openconfig-bgp
   +--rw bgp
      +--rw global
      |  +--rw config
      |  |  +--rw as
      |  |  +--rw router-id?
      |  +--ro state
      |  |  +--ro as
      |  |  +--ro router-id?
      |  |  +--ro total-paths?
      |  |  +--ro total-prefixes?
...
```

Getting Started with gNMI in YDK 0.8.0

Native

Install Python
Install YDK
Download ydk-py-samples

Virtual

VAGRANT VirtualBox

Install Vagrant
Install Virtualbox
Download ydk-py-samples

Install docker
Download from Docker Hub

dCloud

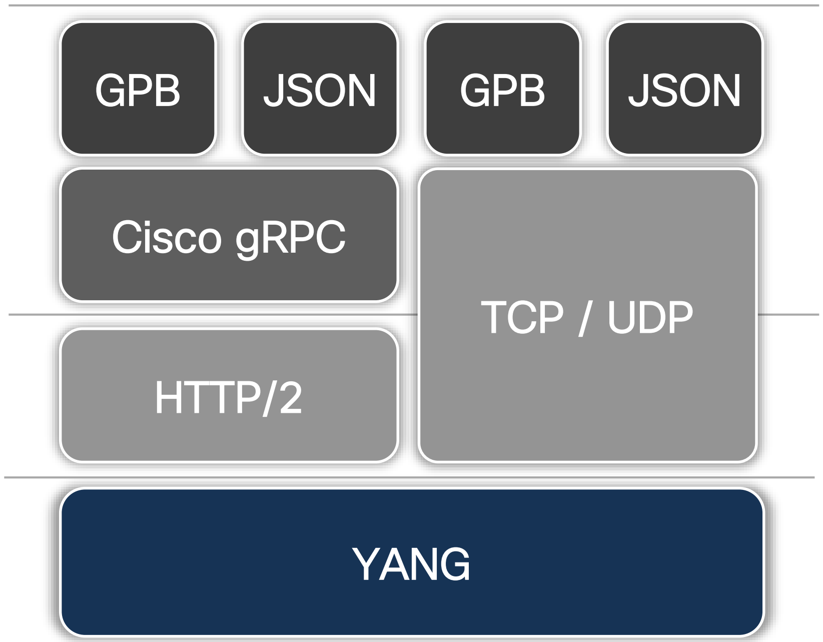
cisco dCloud

YANG Development Kit Sandbox 3.0
dCloud.cisco.com

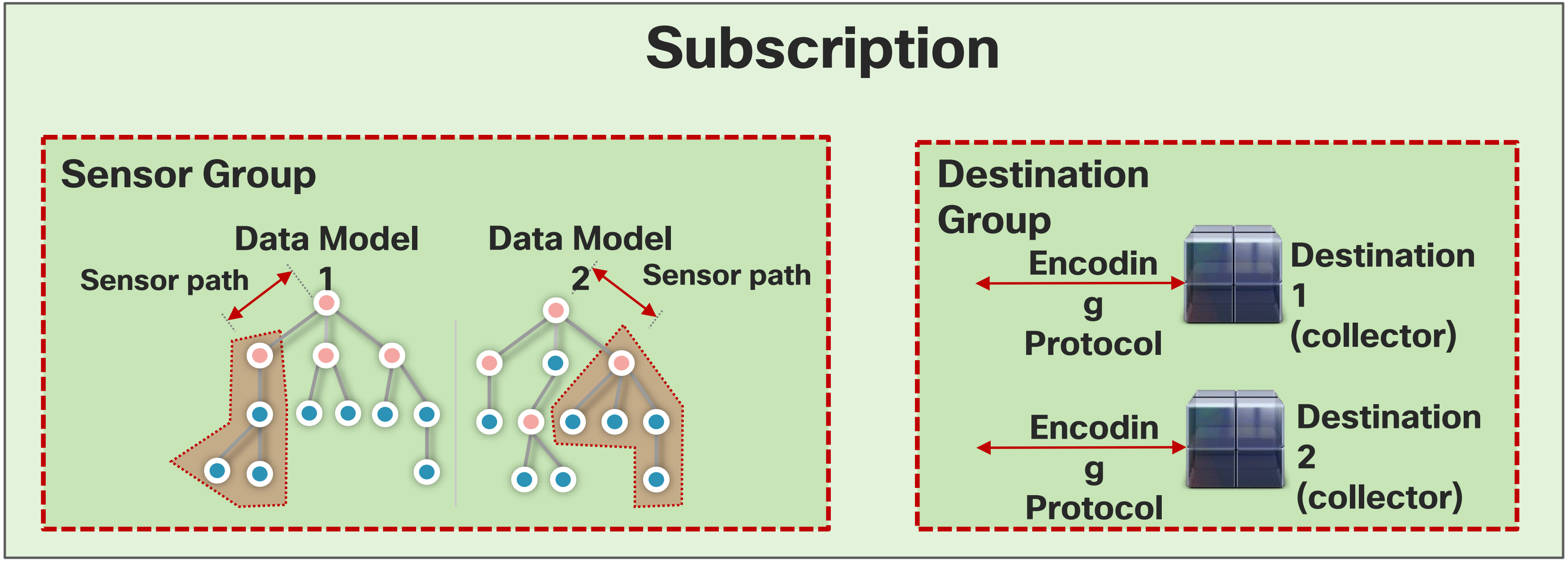
Telemetry

Overview of Telemetry on Cisco IOS XR

- Loosely-coupled stack
  - Data encoding (JSON vs GPB)
  - Transport (HTTPv2 vs TCP vs UDP)
  - Data model (native vs open)
- Session initiation
  - Dial-in (transient destination)
  - Dial-out (persistent destination)
- Flexible data streaming modes (frequency vs event driven)



Telemetry Subscriptions



Subscription

Sensor Group

Data Model 1

Sensor path

Data Model 2

Sensor path

Destination Group

Encoding Protocol

Destination 1 (collector)

Encoding Protocol

Destination 2 (collector)

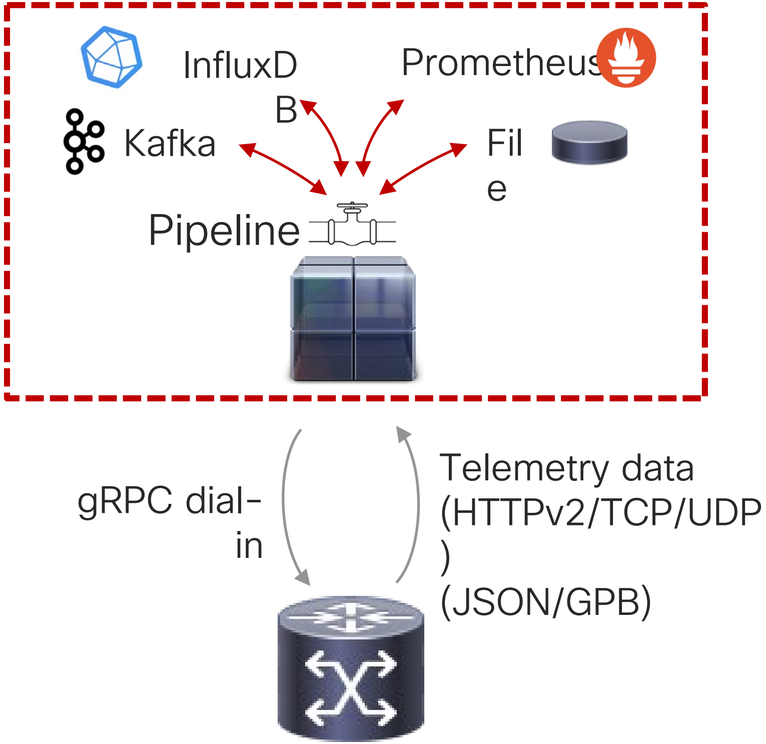
● Aggregation point

Encoding Options

Compact Google Protocol Buffers (GPB)	Self-describing (key-value) Google Protocol Buffers (GPB)	JSON
Most efficient	Medium efficiency	Least efficient
Binary encoding	Hybrid text/binary encoding	Text-based encoding
Data definition required to decode data stream	No data definition required to decode data stream	No data definition required to decode data stream

Pipeline - An Open-Source Telemetry Collector

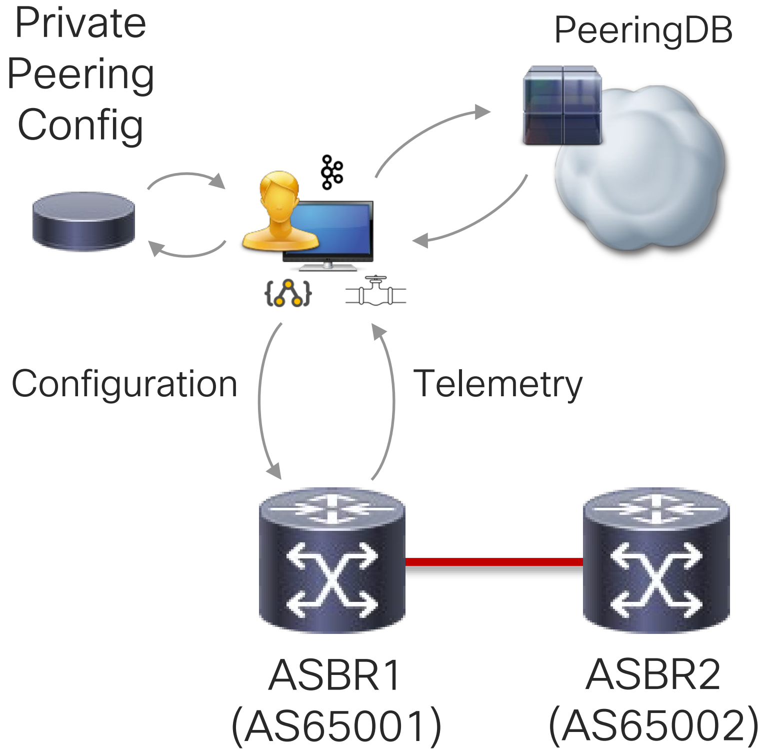
- Collector for telemetry data
- Performs basic encoding transformation
- Data producer for Kafka, InfluxDB, Prometheus, etc.
- Supports dial-in and dial-out sessions



Demonstration

Peering Use Case
Configure and Validate Peering on ASBR1

- Load peer configuration
- Configure interface and validate operation
- Configure BGP neighbor and validate operation



Open Source Tool Chain

YDK

(ydk.io)

- Python/C++/Go bindings for OpenConfig models
- Detailed client-side data validation
- Protocol / transport / encoding abstraction

Pipeline

(git.io/vdnnT)

- Collector for router streaming telemetry
- Performs basic encoding transformation
- Data producer for Kafka, InfluxDB, Prometheus, etc.

Kafka

(kafka.apache.org)

- Distributed streaming platform (message bus)
- Producer, consumer, stream and connector APIs
- Rich client support (Python, Java, etc)

Resources

Resources

Model-driven programmability @ Cloud-Scale Networking

- Model-Driven Programmability (http://goo.gl/x3GZDB)

Programmability @ XR Docs

- Tutorials (https://xrdocs.github.io/programmability/tutorials)
- Blogs (https://xrdocs.github.io/programmability/blogs)

Configuration guide

- Cisco IOS XR programmability configuration guide for ASR 9000 series router (http://goo.gl/8dYUeK)
- Cisco IOS XR programmability configuration guide for NCS 5500 series router (http://goo.gl/cnYPw7)

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!

Sandboxes

- dCloud YANG Development Kit sandbox (https://goo.gl/kaYJ3R)
- Ubuntu YDK Vagrant box (https://git.io/vaw1U)
- Docker YDK-Py (https://hub.docker.com/r/ydkdev/ydk-py)

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

GitHub

- YDK Python SDK – YDK-Py (https://github.com/CiscoDevNet/ydk-py)
- YDK Go SDK – YDK-Cpp (https://github.com/CiscoDevNet/ydk-go)
- YDK C++ SDK – YDK-Cpp (https://github.com/CiscoDevNet/ydk-cpp)
- YDK-Py sample apps (https://github.com/CiscoDevNet/ydk-py-samples) – Over 700 apps!

Resources (cont.)

Conferences

- MPLS+SDN+NFV World Congress 2019: Device Programmability Using gRPC (https://youtu.be/KEdNPFU2vLs)
- MPLS+SDN+NFV World Congress 2018: Getting started with OpenConfig (http://youtu.be/B43PRZV-CD8)
- 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)
- Tech Field Day: gNMI Programmatic Configuration (http://youtu.be/8zAebRr6Pg4)

Conclusion

Let’s Recap

- Model-Driven Programmability
  - Speed and scale through automation
  - Rich and flexible in terms of models, transports and encodings
- Data Models
  - Native
  - Open (OpenConfig / IETF)
- NETCONF
  - Rich, mature protocol
  - Relies on XML encoding
- Google RPC
  - Cisco IOS XR and gNMI interface definition
  - Rich development toolchain
- Model-Driven SDK
  - Simplify app development
  - Abstract transport and encoding
  - Automatic data validation
- Telemetry
  - Loosely-coupled stack
  - Session initiation (dial-in vs dial-out)
  - Flexible data streaming modes

Complete your online session survey

- Please complete your session survey after each session. Your feedback is very important.
- Complete a minimum of 4 session surveys and the Overall Conference survey (starting on Thursday) to receive your Cisco Live t-shirt.
- All surveys can be taken in the Cisco Events Mobile App or by logging in to the Content Catalog on ciscolive.com/emea.

Cisco Live sessions will be available for viewing on demand after the event at ciscolive.com.

Continue your education

Demos in the Cisco Showcase

Walk-In Labs

Meet the Engineer 1:1 meetings

Related sessions

Thank you

cisco

You make **possible**

Backup

YANG Model Example

YANG

```
container community-sets {
       description “Container for community sets";
       list community-set {
         key community-set-name;
         description "Definitions for community sets";
         leaf community-set-name {
           type string;
           description "name of the community set";
         }
         leaf-list community-member {
           type string {
             pattern '([0-9]+:[0-9]+)';
           }
           description "members of the community set";
         }
       }
     }
```

CLI

```
community-sets
  community-set C-SET1
    65172:1,
    65172:2,
    65172:3
  !
  community-set C-SET10
    65172:10,
    65172:20,
    65172:30
  !
!
```

Model Data Example

XML

```
<community-sets>
  <community-set>
    <community-set-name>C-SET1</community-set-name>
    <community-member>65172:1</community-member>
    <community-member>65172:2</community-member>
    <community-member>65172:3</community-member>
  </community-set>
  <community-set>
    <community-set-name>C-SET10</community-set-name>
    <community-member>65172:10</community-member>
    <community-member>65172:20</community-member>
    <community-member>65172:30</community-member>
  </community-set>
</community-sets>
```

CLI

```
community-sets
  community-set C-SET1
    65172:1,
    65172:2,
    65172:3
  !
  community-set C-SET10
    65172:10,
    65172:20,
    65172:30
  !
!
```

Model Data Example

JSON

```
{   "community-sets": {
        "community-set": [
            {   "community-set-name": "CSET1",
                "community-member": [
                    "65172:1",
                    "65172:2",
                    "65172:3" ]
            },
            {   "community-set-name": "CSET10",
                "community-member": [
                    "65172:10",
                    "65172:20",
                    "65172:30" ]
            }
        ]
    }
}
```

CLI

```
community-sets
  community-set C-SET1
    65172:1,
    65172:2,
    65172:3
  !
  community-set C-SET10
    65172:10,
    65172:20,
    65172:30
  !
!
```

You make **possible**