cisco The bridge to possible

Model-Driven Streaming Telemetry and Programmability on Cisco NX-OS

Nick Mortari
Technical Marketing Engineer
Cloud Networking Team

BRKDCN-2904

Cisco Webex App

Questions?

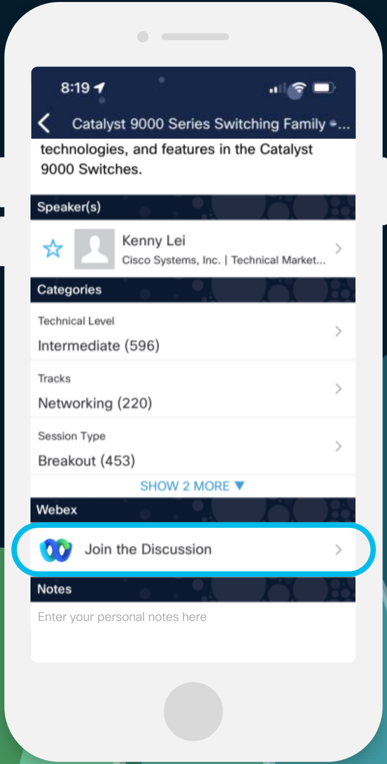
Use Cisco Webex App to chat
with the speaker after the session

How

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

Webex spaces will be moderated
by the speaker until June 7, 2024.

https://ciscolive.ciscoevents.com/
ciscolivebot/#BRKDCN-2904



8:19

Catalyst 9000 Series Switching Family -...

technologies, and features in the Catalyst 9000 Switches.

Speaker(s)

Kenny Lei
Cisco Systems, Inc. | Technical Market...

Categories

Technical Level
Intermediate (596)

Tracks
Networking (220)

Session Type
Breakout (453)

SHOW 2 MORE ▼

Webex

Join the Discussion

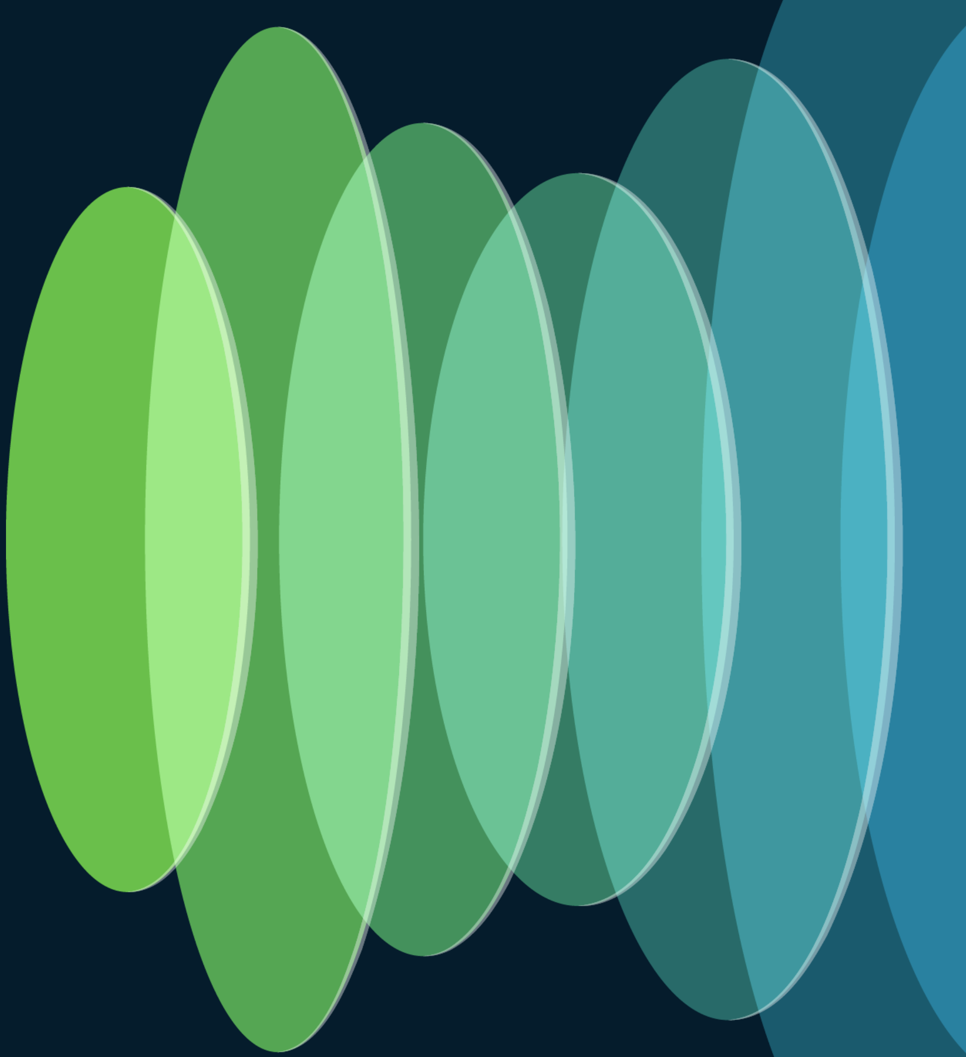
Notes

Enter your personal notes here

Agenda

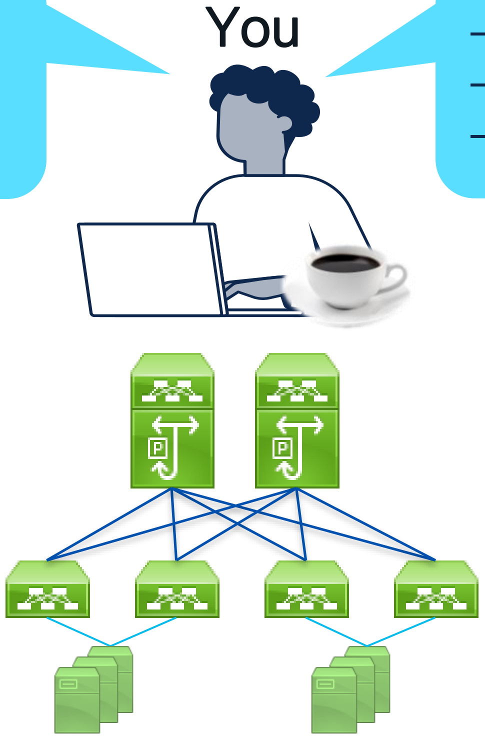
- Why Are We Here Today?
- What Does “Model-Driven” Mean?
- Building Blocks of Programmability
- Building Blocks of Telemetry
- Open-Source Telemetry Stack
- Demonstrations Throughout

Why Are We Here Today?



Why Are We Here Today?

- I need to change port VLANs
- I need to change MTU values
- I need to configure routes



- Are my BGP neighbors up?
- Are my links fully utilized?
- Are my devices overheating?

Why Are We Here Today?

Data

- Can’t access the information I need
- Can’t push or pull from different devices

Performance

- My tools don’t scale well
- Can’t push or pull data fast enough

Why Are We Here Today?

Model-driven methods save the day!

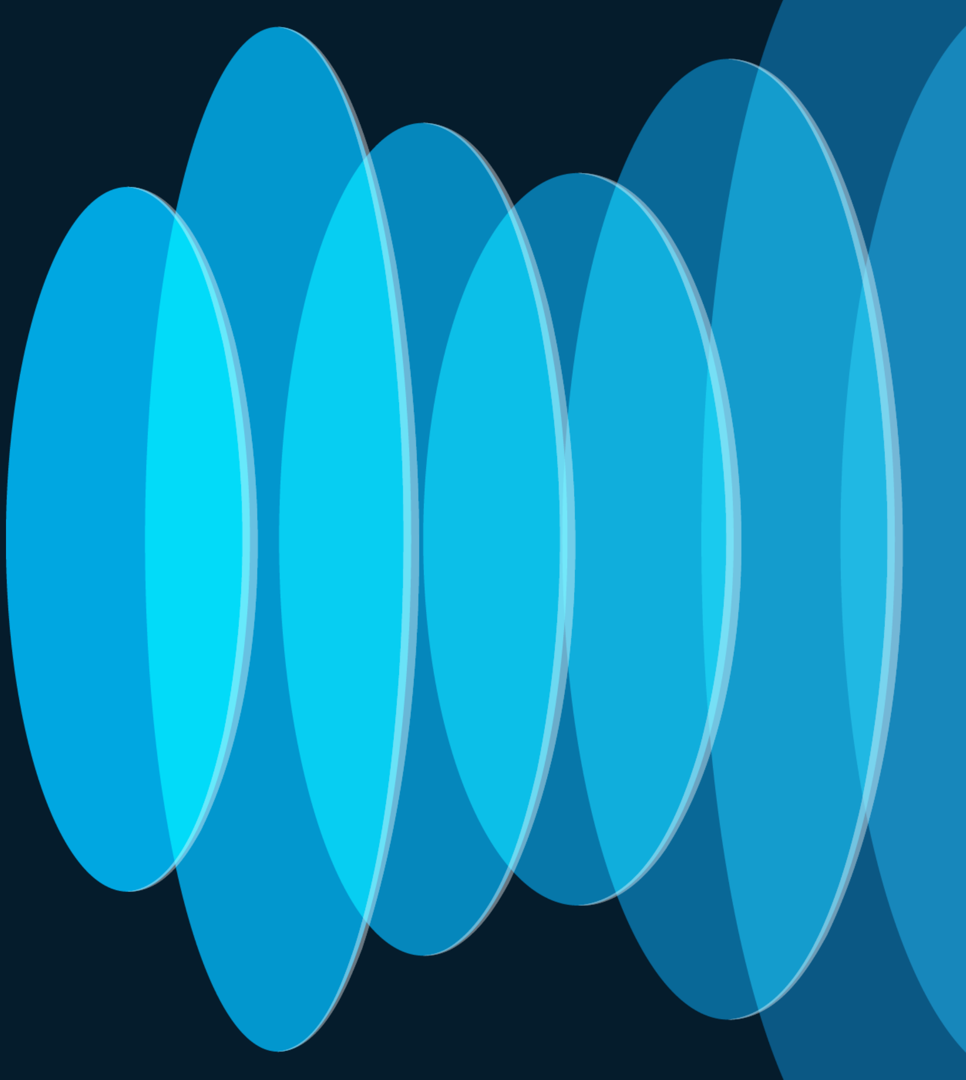
Why Are We Here Today?

Access Detailed Information

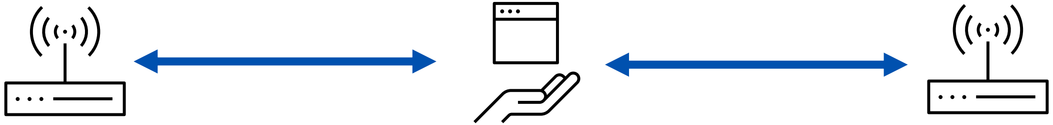
Scalability

Performance

What Does “Model-Driven” Mean?



What Is Model-Driven?



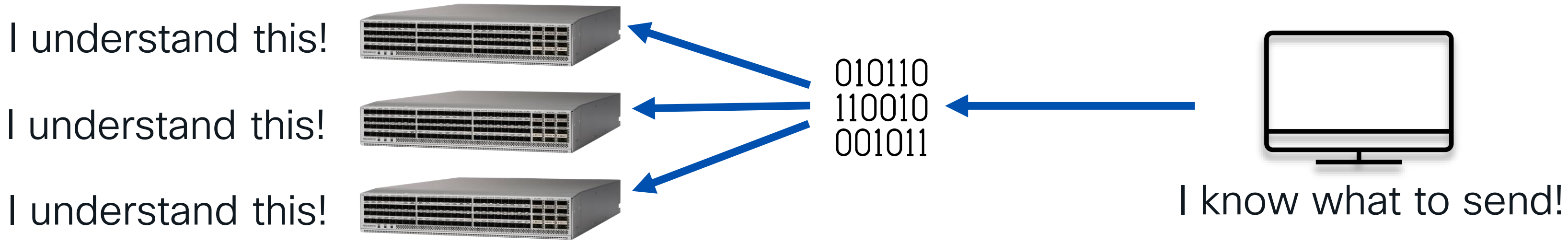
Using a mutually agreed upon method for structuring information

Example: Standard address format must be used when sending mail

To: Nick Mortari
Address: 123 Street,
City, State Zip Code

What Is Model-Driven Programmability?

Sending configuration information to a device using a common model



I understand this!

I understand this!

I understand this!

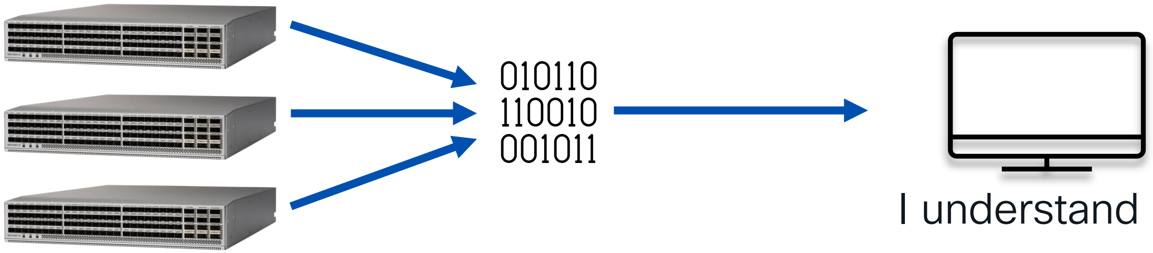
010110
110010
001011

I know what to send!

Example: Set CDP hold time on my switches

What Is Model-Driven Telemetry?

Receiving state information from a device using a common model



I know what to send!

I know what to send!

I know what to send!

010110
110010
001011

I understand this!

Example: Collect current power consumption from my devices

Model-Driven Programmability

Building Blocks of Programmability

- **Data Structure**
- Data Encoding
- Data Transport

Data Structure

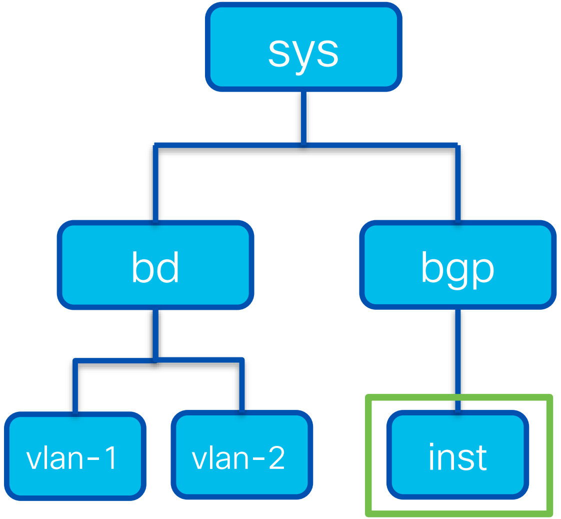
Two Options

DME (Data Management Engine)

YANG Models

Data Structure

DME - Programmability

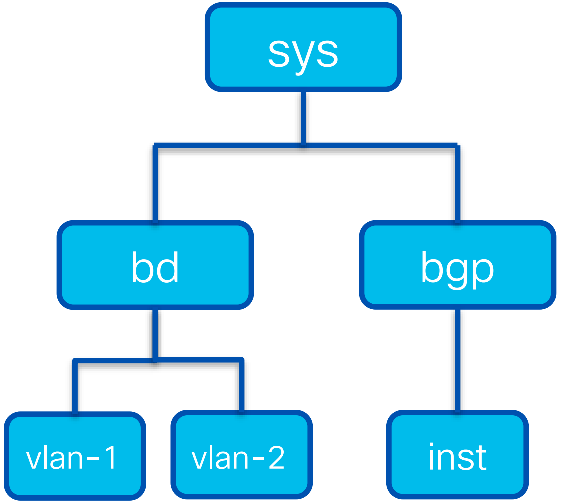


- Configuration and operational data is stored in DME
- Tree data structure
- DN (Distinguished Name) is in .../.../.../... format
- Configuration data can be accessed with the DN as a sensor path

- sys/bgp/inst represents configuration and state data for BGP process

Data Structure

DME - Programmability

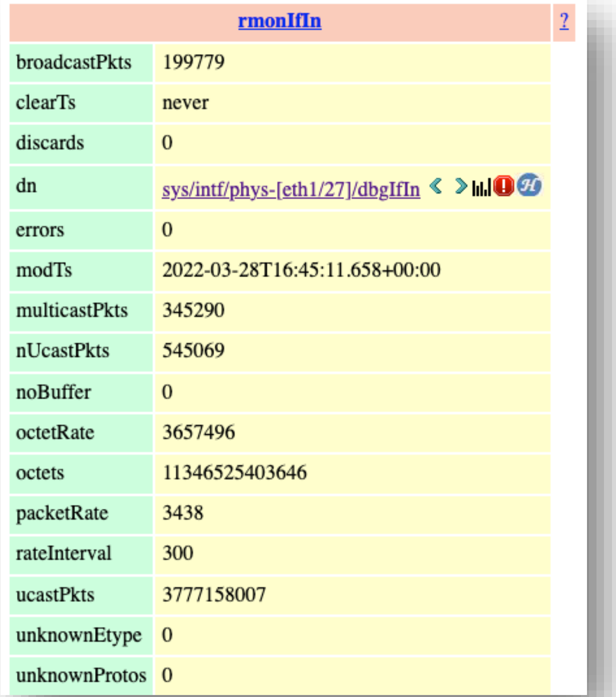


- Almost entire OS is available
- As of 10.4(2)F, almost all commands are DMEized
- We can use DME paths for programmability

Data Structure

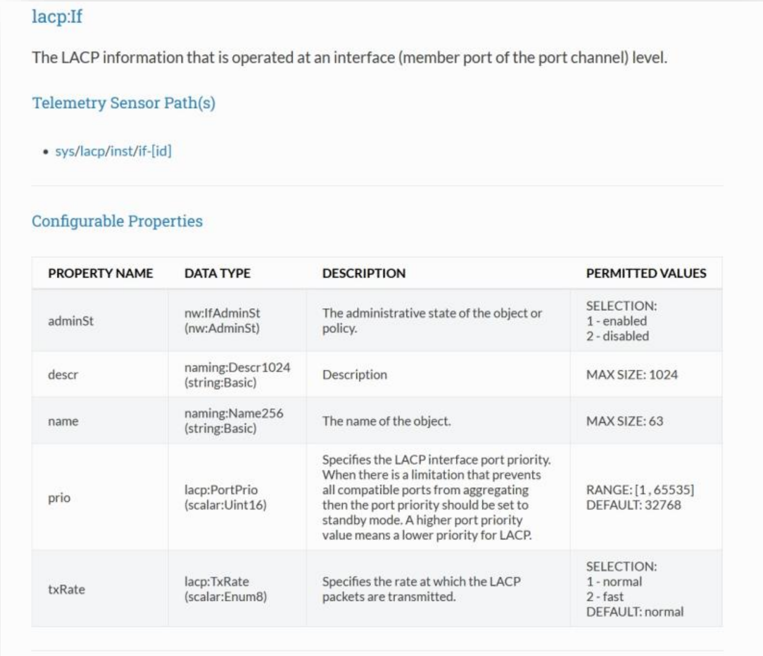
DME - Programmability Paths

Visore is a built-in DME browser of NX-OS, navigate to https://[switch_ip]/visore.html



rmonIfIn		?
broadcastPkts	199779	
clearTs	never	
discards	0	
dn	sys/intf/phys-[eth1/27]/dbgIfIn	
errors	0	
modTs	2022-03-28T16:45:11.658+00:00	
multicastPkts	345290	
nUcastPkts	545069	
noBuffer	0	
octetRate	3657496	
octets	11346525403646	
packetRate	3438	
rateInterval	300	
ucastPkts	3777158007	
unknownEtype	0	
unknownProtos	0	

API reference is also available: https://developer.cisco.com/site/nxapi-dme-model-reference-api/?version=10.3(2)



lacp:If

The LACP information that is operated at an interface (member port of the port channel) level.

Telemetry Sensor Path(s)

- sys/lacp/inst/if-[id]

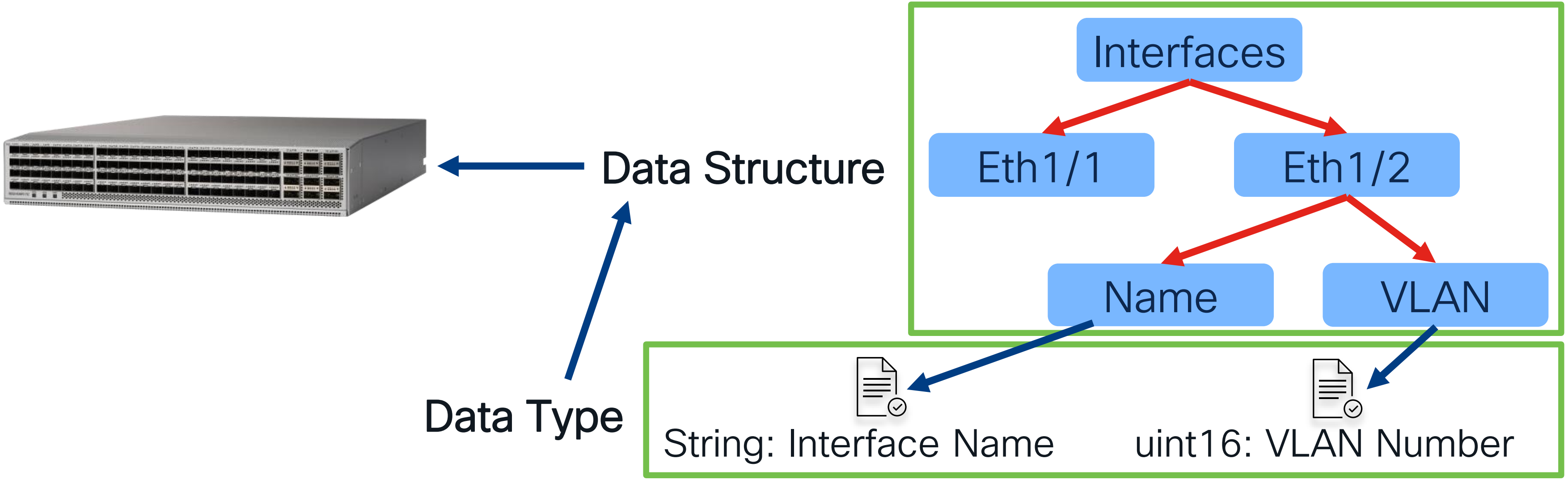
Configurable Properties

PROPERTY NAME	DATA TYPE	DESCRIPTION	PERMITTED VALUES
adminSt	nw:IfAdminSt (nw:AdminSt)	The administrative state of the object or policy.	SELECTION: 1 - enabled 2 - disabled
descr	naming:Descr1024 (string:Basic)	Description	MAX SIZE: 1024
name	naming:Name256 (string:Basic)	The name of the object.	MAX SIZE: 63
prio	lacp:PortPrio (scalar:Uint16)	Specifies the LACP interface port priority. When there is a limitation that prevents all compatible ports from aggregating then the port priority should be set to standby mode. A higher port priority value means a lower priority for LACP.	RANGE: [1 , 65535] DEFAULT: 32768
txRate	lacp:TxRate (scalar:Enum8)	Specifies the rate at which the LACP packets are transmitted.	SELECTION: 1 - normal 2 - fast DEFAULT: normal

Data Structure

YANG Models

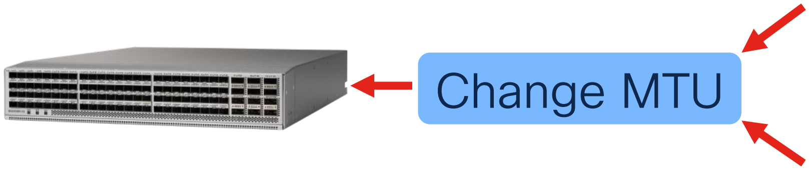
- YANG (Yet Another Next Generation) is a data modeling language
- Defines the data structure and data type for the model we use



Data Structure

YANG Models - Programmability

- NX-OS supports two YANG models for telemetry
  - OpenConfig YANG model
  - Cisco native YANG model
- Configuration elements can be accessed with YANG model paths



OpenConfig path:
/interfaces/interface/config/mtu

Cisco Native path:
/System/intf-items/phys-items/PhysIf-list/mtu

Data Structure

YANG Models – Choosing Your Model

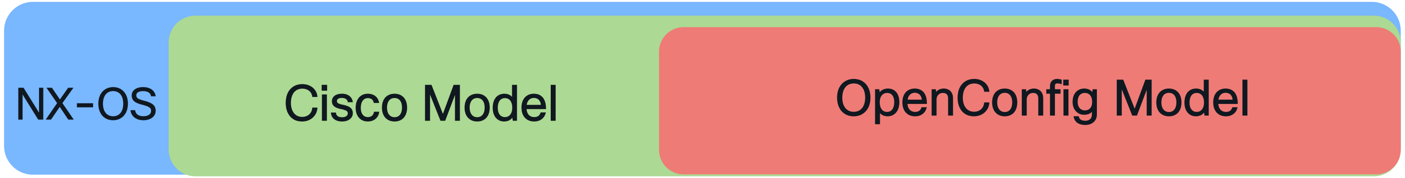
Cisco Native Model

- Vendor specific
- Created by Cisco
- Supports almost every feature on NX-OS

You can use both!

OpenConfig Model

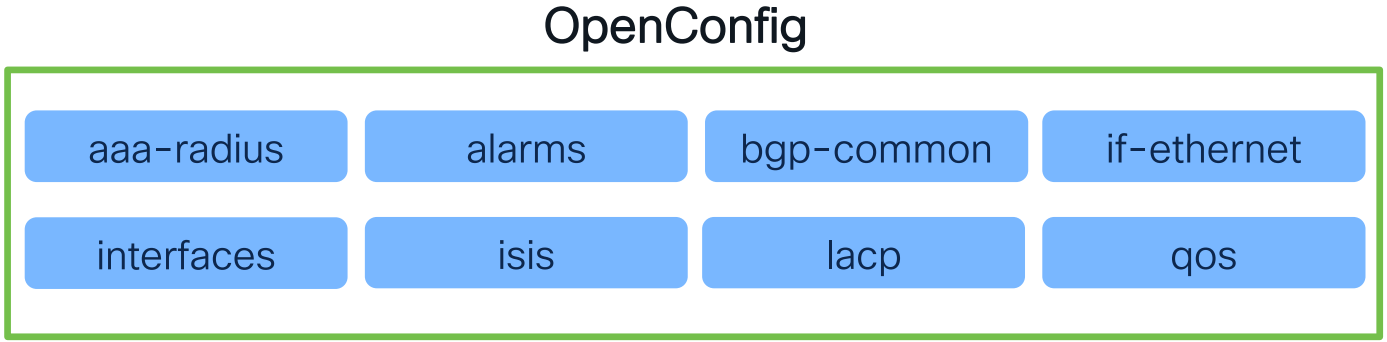
- Vendor agnostic
- Created by many networking companies (open-source)
- Does not support every feature on NX-OS



Data Structure

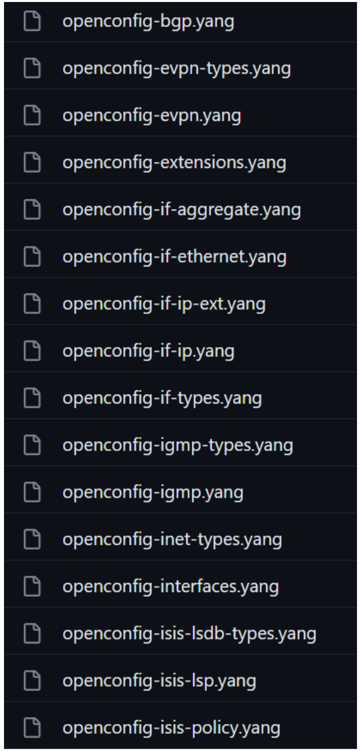
YANG Models – Modules

- YANG models are be broken down into modules
- Allows for easier changes when working with others
- The Cisco model is technically "one large module"



Data Structure

YANG Models – OpenConfig

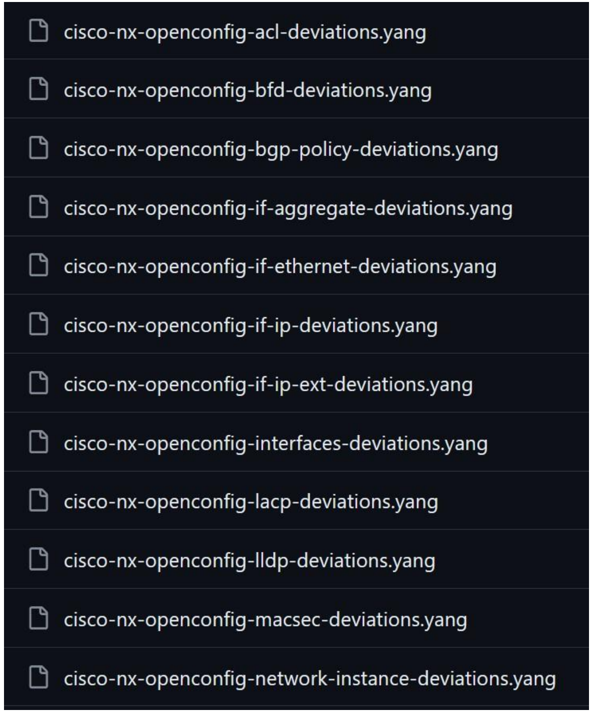


openconfig-bgp.yang
openconfig-evpn-types.yang
openconfig-evpn.yang
openconfig-extensions.yang
openconfig-if-aggregate.yang
openconfig-if-ethernet.yang
openconfig-if-ip-ext.yang
openconfig-if-ip.yang
openconfig-if-types.yang
openconfig-igmp-types.yang
openconfig-igmp.yang
openconfig-inet-types.yang
openconfig-interfaces.yang
openconfig-isis-lsdb-types.yang
openconfig-isis-lsp.yang
openconfig-isis-policy.yang

- 100+ OpenConfig modules supported
- To enable OC YANG Model:
  - Before 10.2(2), `mtx-openconfig-all` rpm needs to be installed on switch
  - After 10.2(2), run `feature openconfig`

Data Structure

YANG Models – OpenConfig



cisco-nx-openconfig-acl-deviations.yang

cisco-nx-openconfig-bfd-deviations.yang

cisco-nx-openconfig-bgp-policy-deviations.yang

cisco-nx-openconfig-if-aggregate-deviations.yang

cisco-nx-openconfig-if-ethernet-deviations.yang

cisco-nx-openconfig-if-ip-deviations.yang

cisco-nx-openconfig-if-ip-ext-deviations.yang

cisco-nx-openconfig-interfaces-deviations.yang

cisco-nx-openconfig-lacp-deviations.yang

cisco-nx-openconfig-lldp-deviations.yang

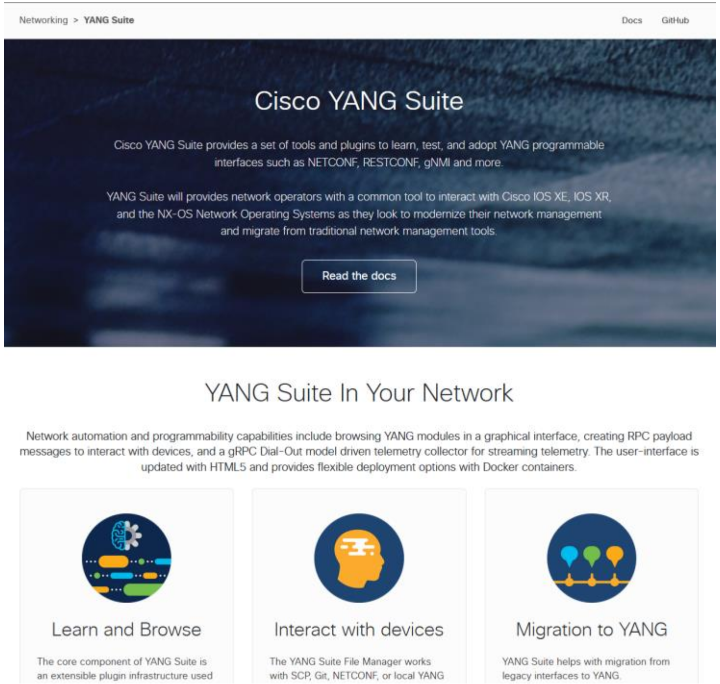
cisco-nx-openconfig-macsec-deviations.yang

cisco-nx-openconfig-network-instance-deviations.yang

- Be aware of deviations, modules can be partially supported
- Deviation can mean:
  - Path does not follow the original YANG module definition
  - Path is not supported
- A full list of supported modules and deviations can be found at: https://github.com/YangModels/yang/tree/master/vendor/cisco/nx

Data Structure

YANG Suite



Cisco YANG Suite

Cisco YANG Suite provides a set of tools and plugins to learn, test, and adopt YANG programmable interfaces such as NETCONF, RESTCONF, gNMI and more.

YANG Suite will provides network operators with a common tool to interact with Cisco IOS XE, IOS XR, and the NX-OS Network Operating Systems as they look to modernize their network management and migrate from traditional network management tools.

YANG Suite In Your Network

Network automation and programmability capabilities include browsing YANG modules in a graphical interface, creating RPC payload messages to interact with devices, and a gRPC Dial-Out model driven telemetry collector for streaming telemetry. The user-interface is updated with HTML5 and provides flexible deployment options with Docker containers.

Learn and Browse

The core component of YANG Suite is an extensible plugin infrastructure used

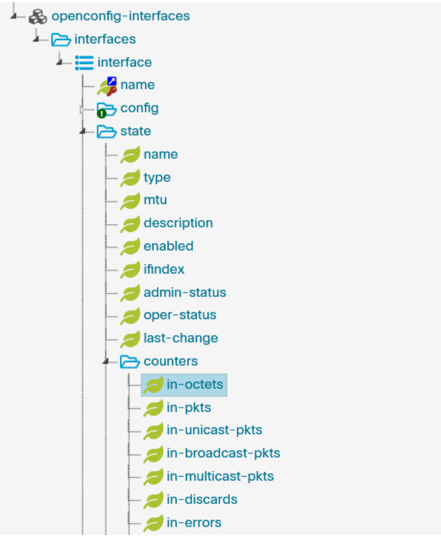
Interact with devices

The YANG Suite File Manager works with SCP, Git, NETCONF, or local YANG

Migration to YANG

YANG Suite helps with migration from legacy interfaces to YANG.

- Tool to assist with YANG model exploration and testing
- Includes YANG browser for both models



```
openconfig-interfaces
  interfaces
    interface
      name
      config
      state
        name
        type
        mtu
        description
        enabled
        ifindex
        admin-status
        oper-status
        last-change
        counters
          in-octets
          in-pkts
          in-unicast-pkts
          in-broadcast-pkts
          in-multicast-pkts
          in-discards
          in-errors
```

https://developer.cisco.com/yangsuite

Building Blocks of Programmability

- Data Structure
- **Data Encoding**
- Data Transport

Data Encoding

Two Options

XML (eXtensible Markup Language)

JSON (JavaScript Object Notation)

Data Encoding

XML

```
<interface>
      <name>eth1/49</name>
      <config>
            <access-vlan>200</access-vlan>
            <interface-mode>ACCESS</interface-mode>
            <description>To server</description>
            <duplex>auto</duplex>
            <admin-state>up</admin-state>
            <speed>50000</speed>
            <mtu>1500</mtu>
      </config>
</interface>
```

- Every element must have an opening and closing tag
- Value is placed between the tags
- Low transfer efficiency

🟩 *Tag*

🟨 *Value*

Data Encoding

JSON

```
{
      "interface": {
            "name": "eth1/49",
            "config": {
                  "access-vlan": “200”,
                  "interface-mode": "ACCESS",
                  "description": "To server",
                  "duplex": "auto",
                  "admin-state": "up",
                  "speed": "50000",
                  "mtu": "1500"
            }
      }
}
```

- Every element must have a key
- Value is placed after the key
- Better transfer efficiency

🟩 *Key*

🟨 *Value*

Building Blocks of Programmability

- Data Structure
- Data Encoding
- **Data Transport**

Data Transport

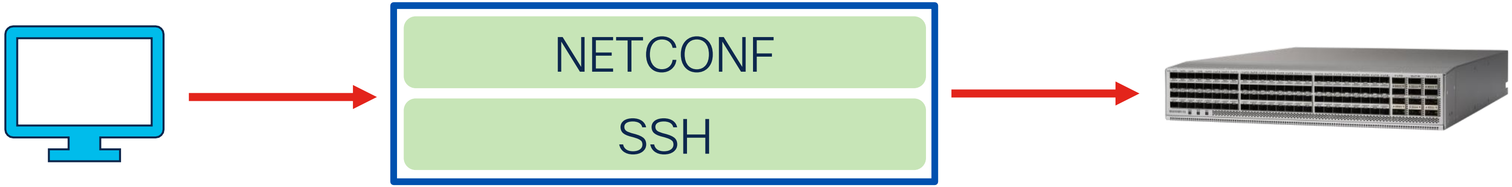
Four Options

- NETCONF (Network Configuration)
- RESTCONF (RESTful Configuration)
- NX-API REST
- gNMI (Google Remote Procedure Call Network Management Interface)

Data Transport

NETCONF

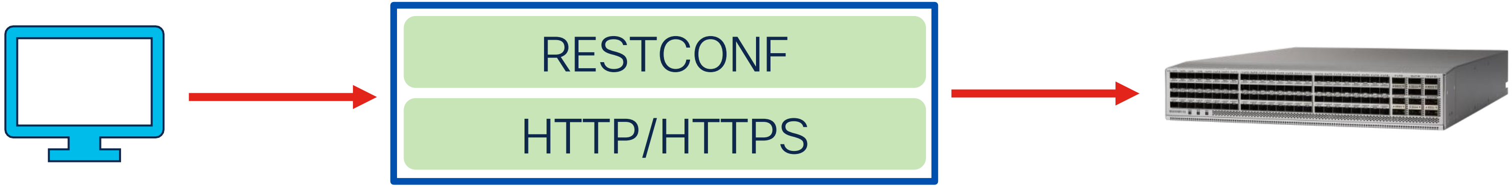
- Protocol that connects over SSH to leverage a set of RPCs
- Supports configuration management with `edit-config`
- Supports YANG as the data structure
- Supports XML for data encoding
- Enabled with `feature netconf` command



Data Transport

RESTCONF

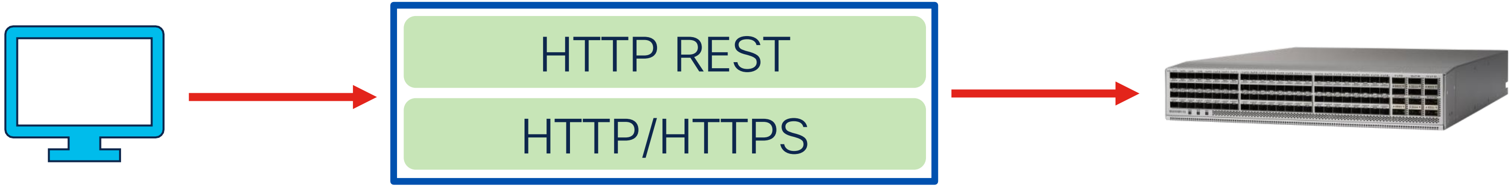
- Protocol that connects over HTTP/HTTPS to pass data
- Supports configuration management through RESTful methods
- Supports YANG as the data structure
- Supports XML and JSON for data encoding
- Enabled with `feature restconf` command



Data Transport

NX-API REST

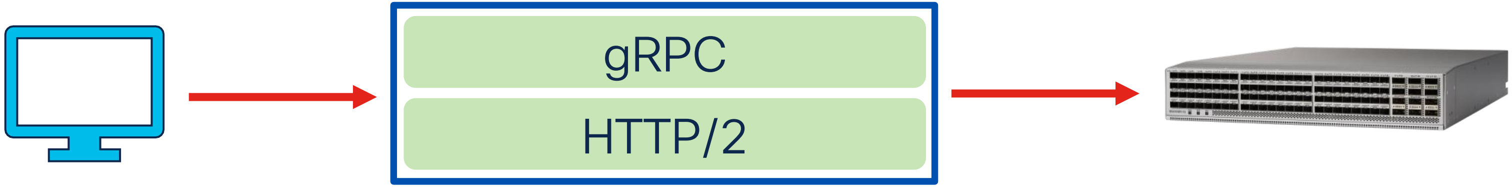
- Protocol that connects over HTTP/HTTPS to pass data
- Supports configuration management through RESTful methods
- Supports DME as the data structure
- Supports XML and JSON for data encoding
- Enabled with `feature nxapi` command



Data Transport

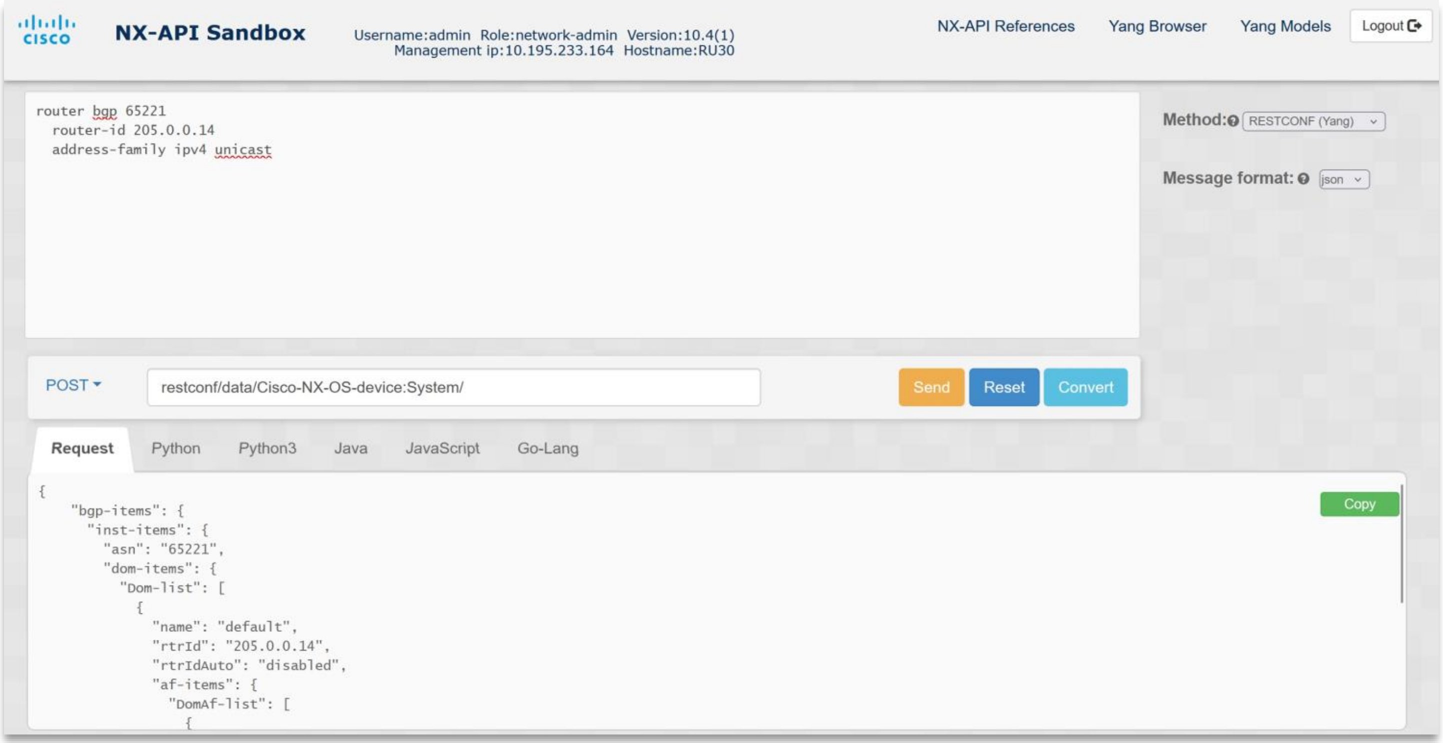
gNMI

- Protocol that connects over HTTP/2 to leverage a set of RPCs
- Supports configuration management with `gnmi set`
- Supports DME and YANG as the data structure
- Supports XML and JSON for data encoding
- Enabled with `feature grpc` command



Data Transport

NX-OS Sandbox



```
router bgp 65221
  router-id 205.0.0.14
  address-family ipv4 unicast
```

Method: RESTCONF (Yang)

Message format: json

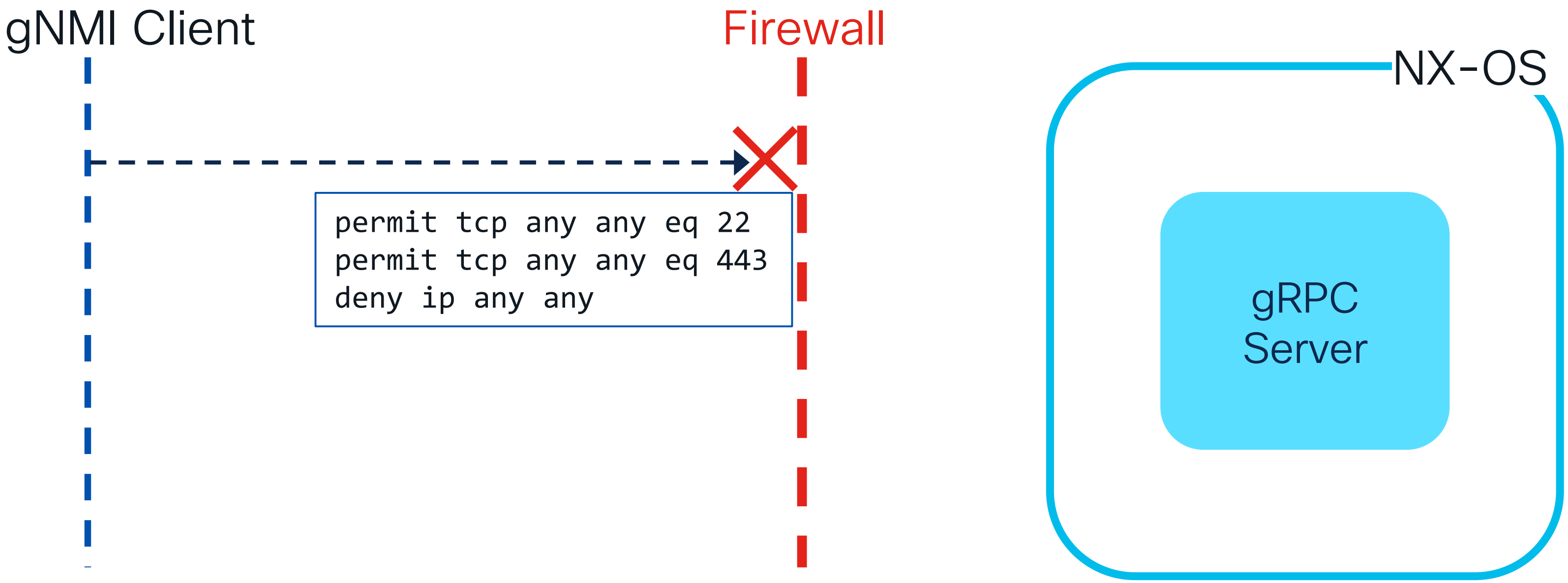
POST restconf/data/Cisco-NX-OS-device:System/

```
{
    "bgp-items": {
      "inst-items": {
        "asn": "65221",
        "dom-items": {
          "Dom-list": [
            {
              "name": "default",
              "rtrId": "205.0.0.14",
              "rtrIdAuto": "disabled",
              "af-items": {
                "DomAf-list": [
                  {
```

- Tool to construct payloads and scripts from CLI
- Navigate to https://[switch_ip]/

Data Transport

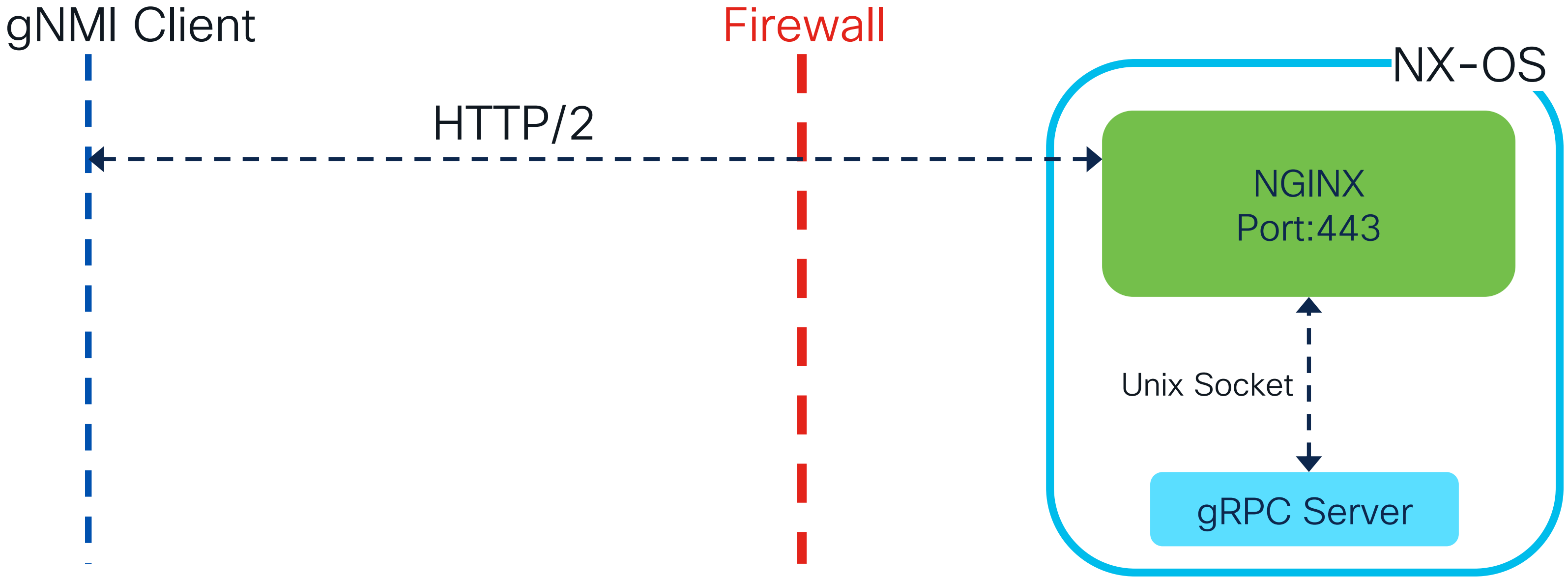
gNMI – Firewalls



```
permit tcp any any eq 22
permit tcp any any eq 443
deny ip any any
```

Data Transport

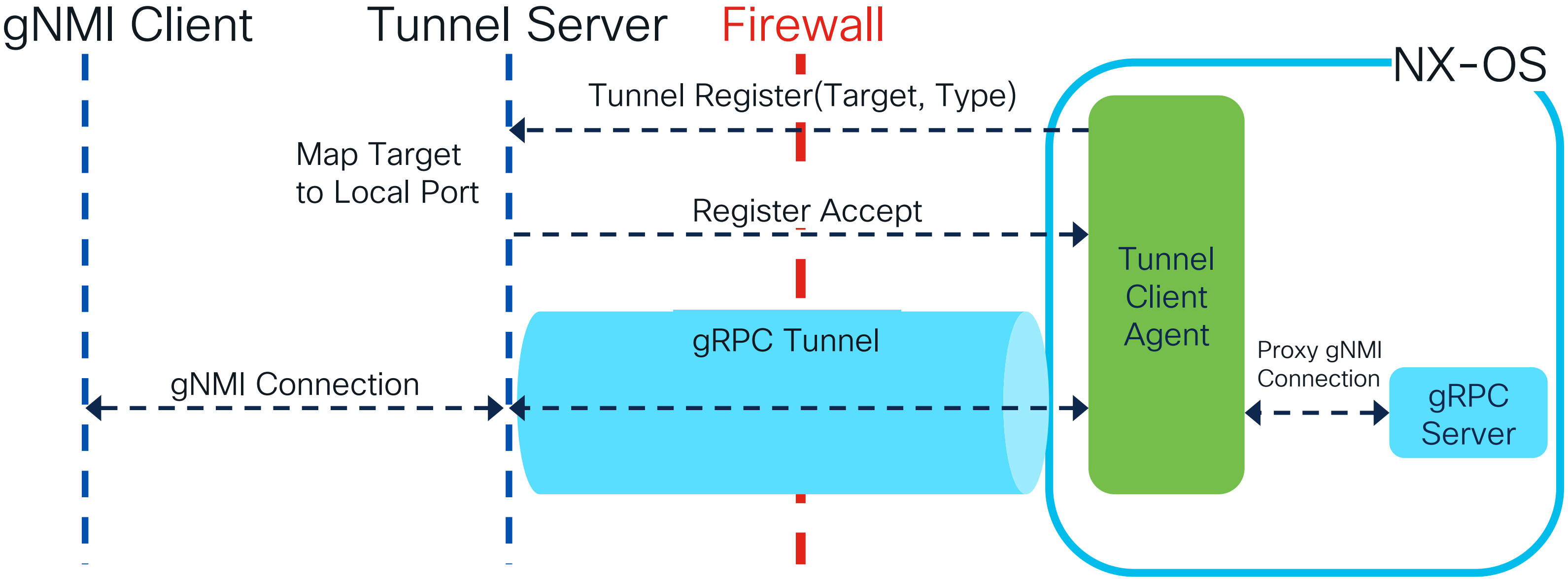
gNMI – NGINX Proxy



Shipping in 10.3(3)F

Data Transport

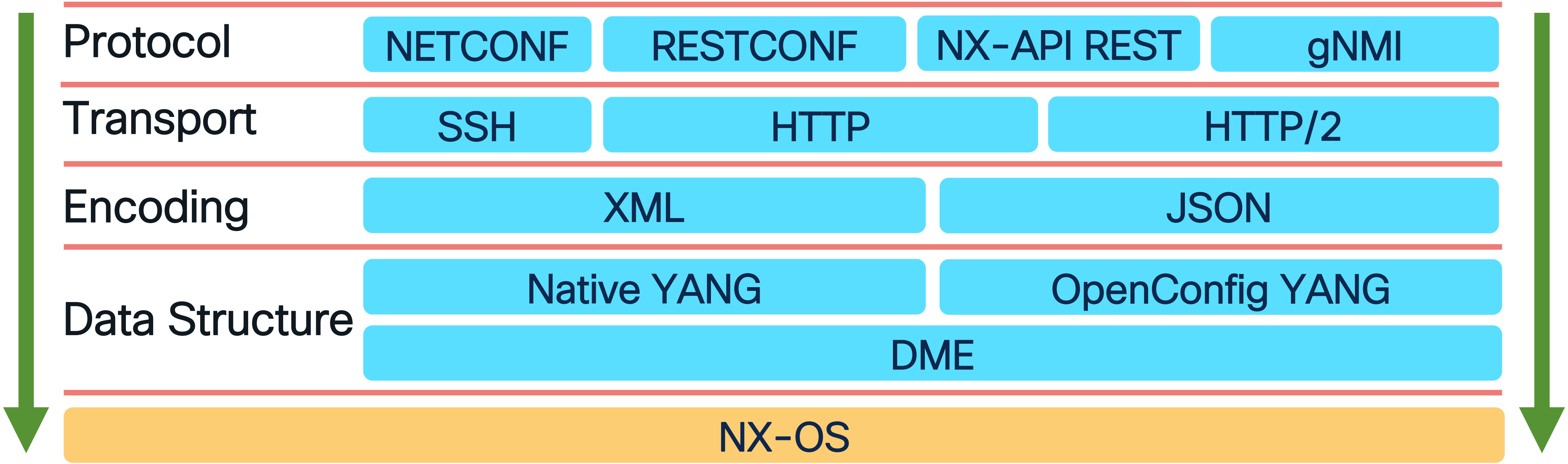
gNMI – gRPC Tunnel



Shipping in 10.3(2)F

Data Transport

Programmability Overview



Protocol: NETCONF | RESTCONF | NX-API REST | gNMI

Transport: SSH | HTTP | HTTP/2

Encoding: XML | JSON

Data Structure: Native YANG | OpenConfig YANG; DME

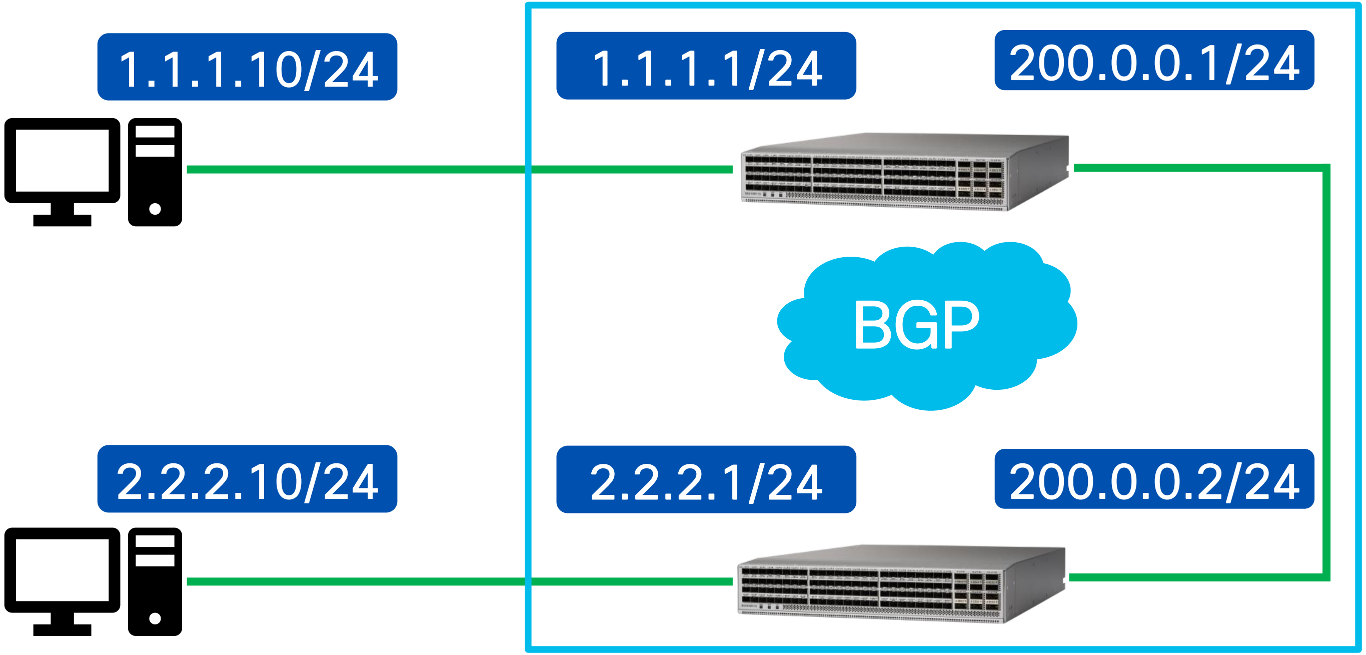
NX-OS

Data Transport

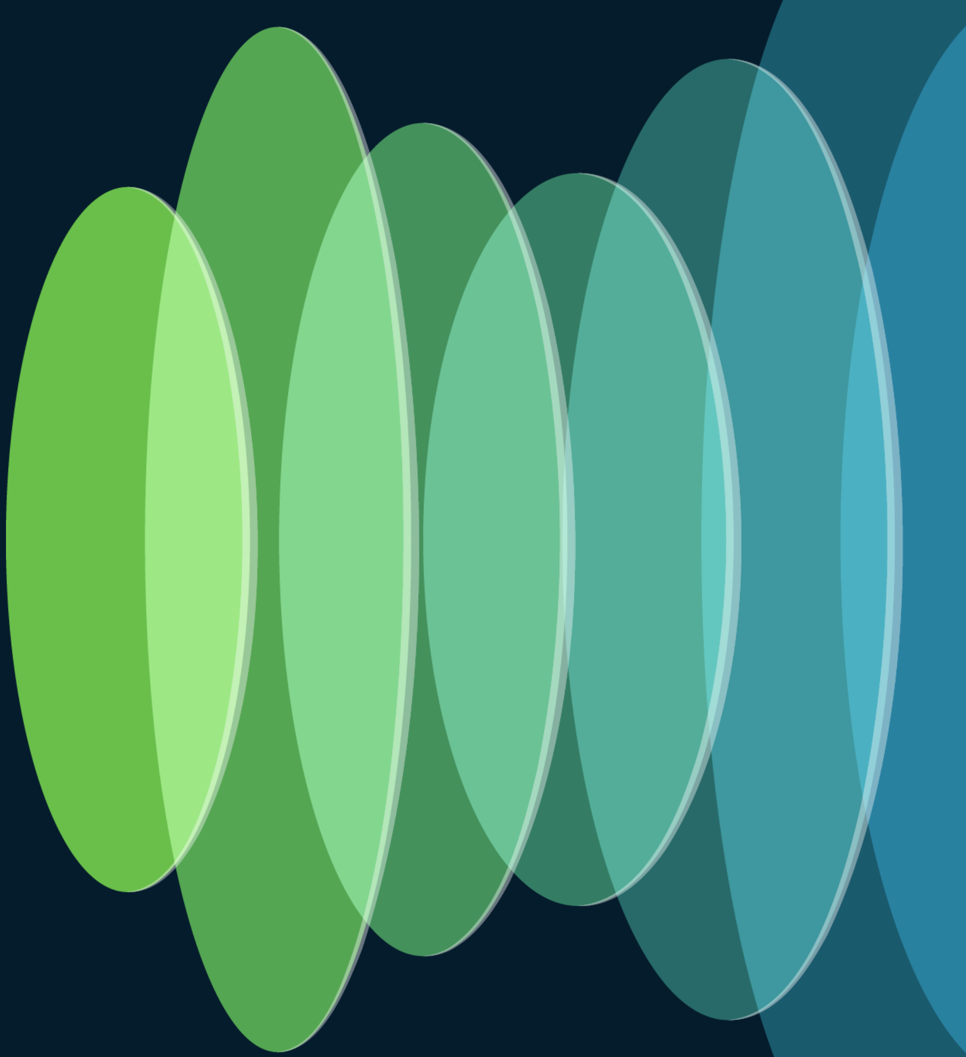
Programmability Options Compared

	NETCONF	RESTCONF	NX-API REST	gNMI
Data Model	YANG	YANG	DME	YANG / DME
Encoding	XML	JSON / XML	JSON / XML	JSON / XML
Transport Method	SSH	HTTP / HTTPS	HTTP / HTTPS	HTTP/2

Demo

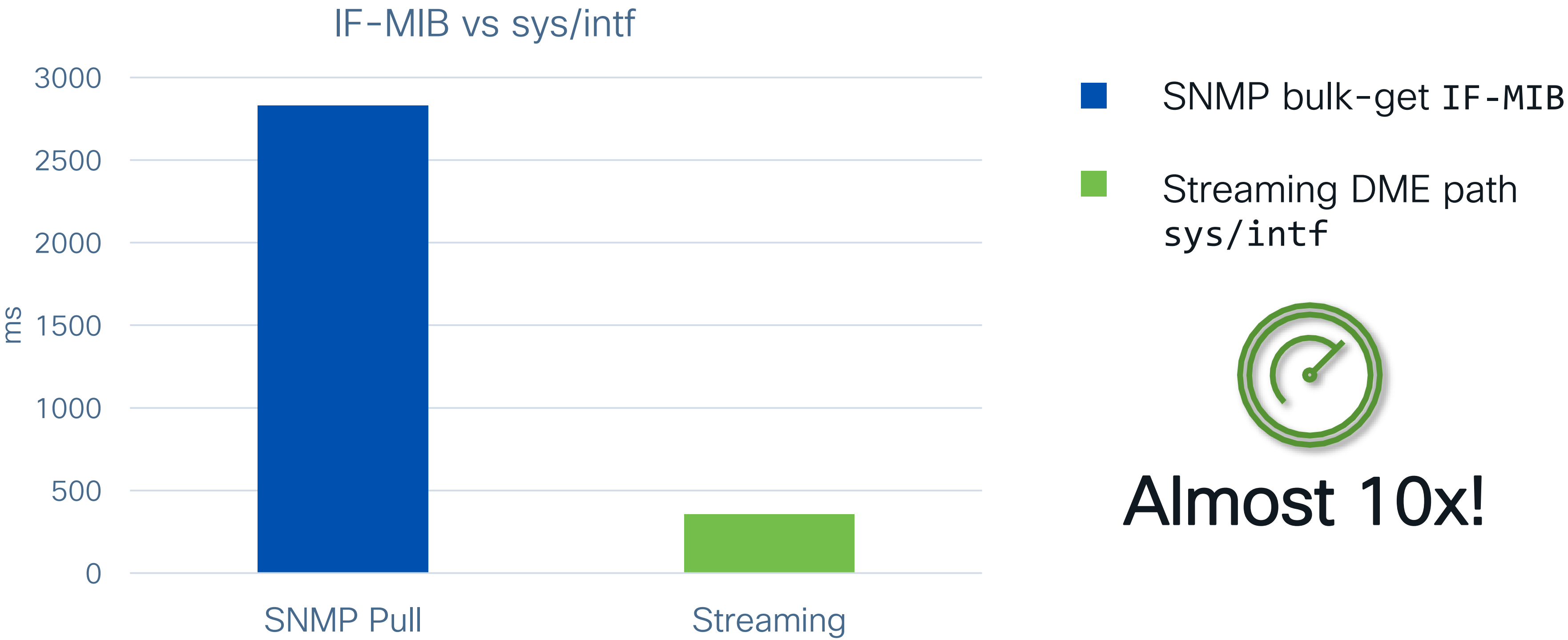


Model-Driven Streaming Telemetry



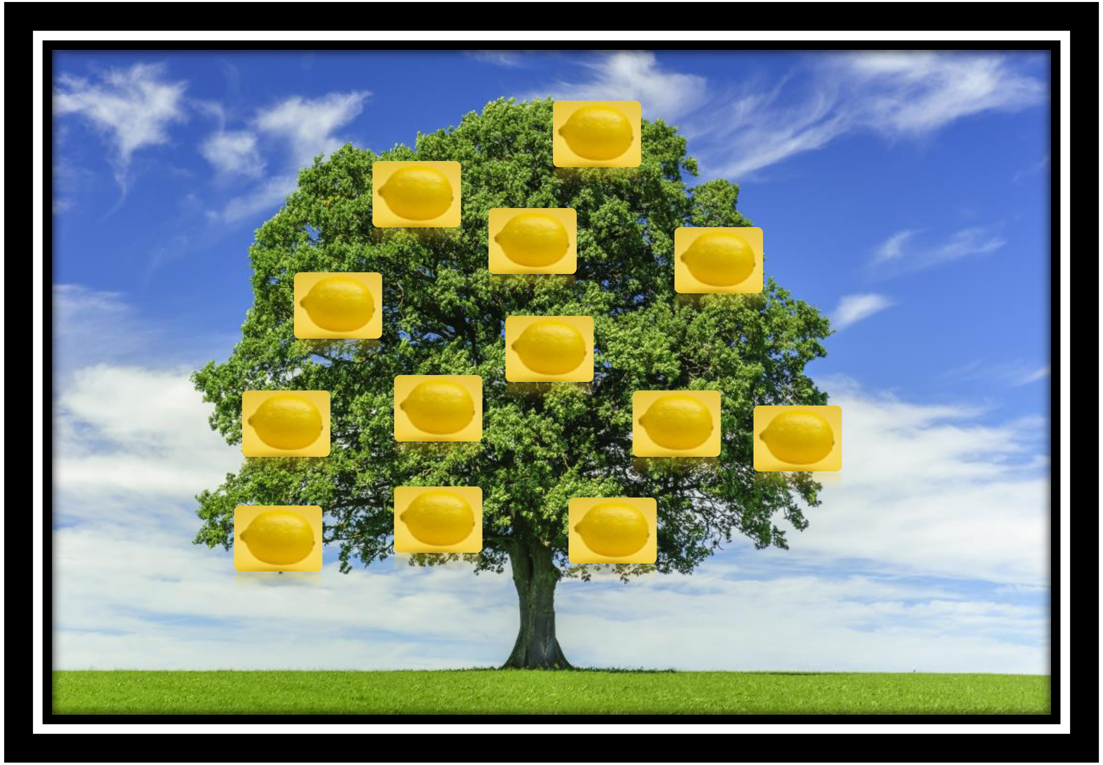
Model-Driven Streaming Telemetry

Performance



Almost 10x!

~~Telemetry~~ The Lemon Tree



Building Blocks of Streaming Telemetry

- **Data Structure**
- Data Frequency
- Data Encoding
- Data Transport

Data Structure

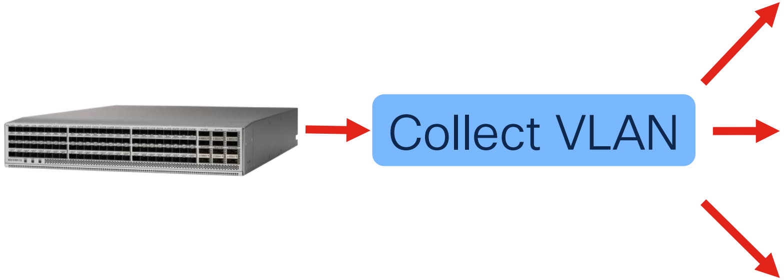
Three Options (well…actually two)

- DME
- YANG
- NX-API CLI

Data Structure

DME and YANG - Telemetry

- DME and YANG paths can also be used for telemetry
- We can “subscribe” to a path and receive a stream of structured data



Collect VLAN

OpenConfig path:
/interfaces/interface/ethernet/switched-vlan/state/access-vlan/

DME:
sys/intf/phys-eth/phys/accessVlan

Cisco Native path:
/System/intf-items/phys-items/PhysIf-list/accessVlan

Data Structure

NX-API CLI

```
93240YC-FX2-L02-S4# show nve vni  | json-pretty
{
    "TABLE_nve_vni": {
        "ROW_nve_vni": [
            {
                "if-name": "nve1",
                "vni": "30000",
                "mcast": "239.1.1.1",
                "vni-state": "Up",
                "mode": "CP",
                "type": "L2 [2300]",
                "flags": null,
                "dci-mcast": "Unconfigured"
            },
...
```

✓

- 100% of customer-facing show commands of NX-OS have output

⚠

- Only supports sample-based telemetry
- CLI doesn‘t have data types, all values are strings
    - The collector will need to parse the result and “guess” data type

Data Structure

Platform Support

Nexus Platform	DME	CLI/NX-API	YANG	Release
3000 with 8G+ RAM	✓	✓	✓ *	7.0(3)I7(1)
9300	✓	✓	✓ *	7.0(3)I5(1)
9500/9400/9800	✓	✓	✓ *	7.0(3)I7(1)
7000/7700	✗	✓	✗	8.3(1)

* Streaming YANG models starting from 9.2(1)

Building Blocks of Streaming Telemetry

- Data Structure
- **Data Frequency**
- Data Encoding
- Data Transport

Data Frequency

Two Options

Sample-Based Collection

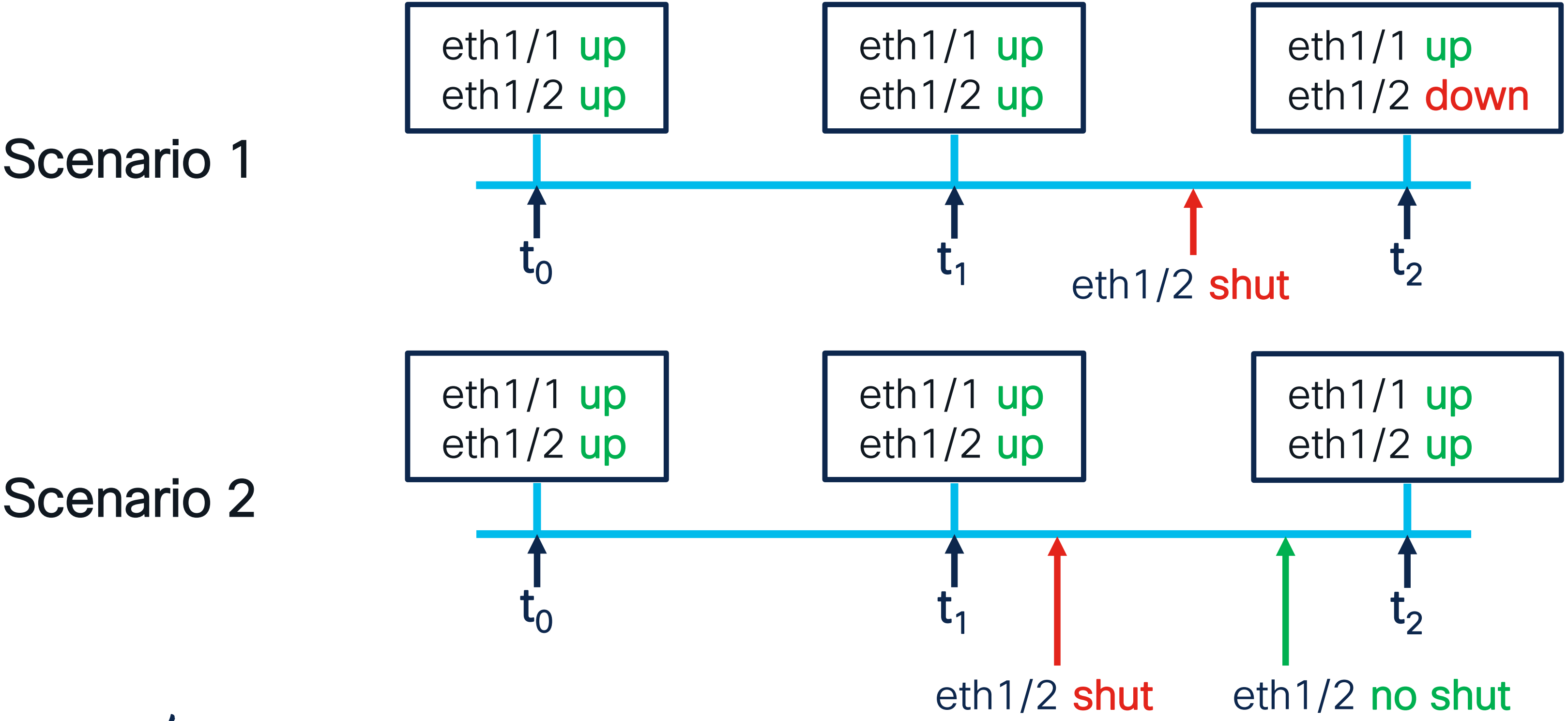
Collect information at every sample interval

Event-Based Collection

Collect information at every event

Data Frequency

Sample-Based Collection



Scenario 1

eth1/1 up
eth1/2 up

eth1/1 up
eth1/2 up

eth1/1 up
eth1/2 down

t_0

t_1

eth1/2 shut

t_2

Scenario 2

eth1/1 up
eth1/2 up

eth1/1 up
eth1/2 up

eth1/1 up
eth1/2 up

t_0

t_1

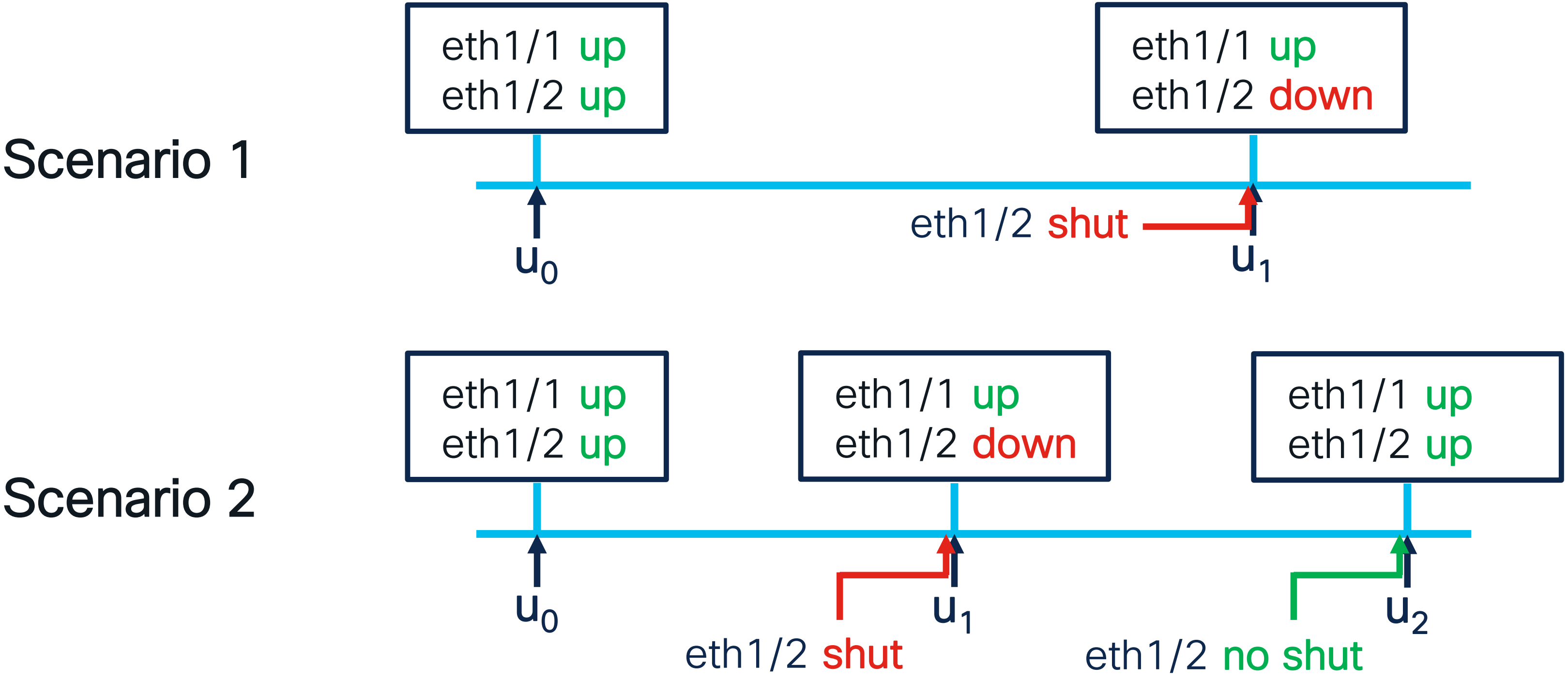
t_2

eth1/2 shut

eth1/2 no shut

Data Frequency

Event-Based Collection



Building Blocks of Streaming Telemetry

- Data Structure
- Data Frequency
- **Data Encoding**
- Data Transport

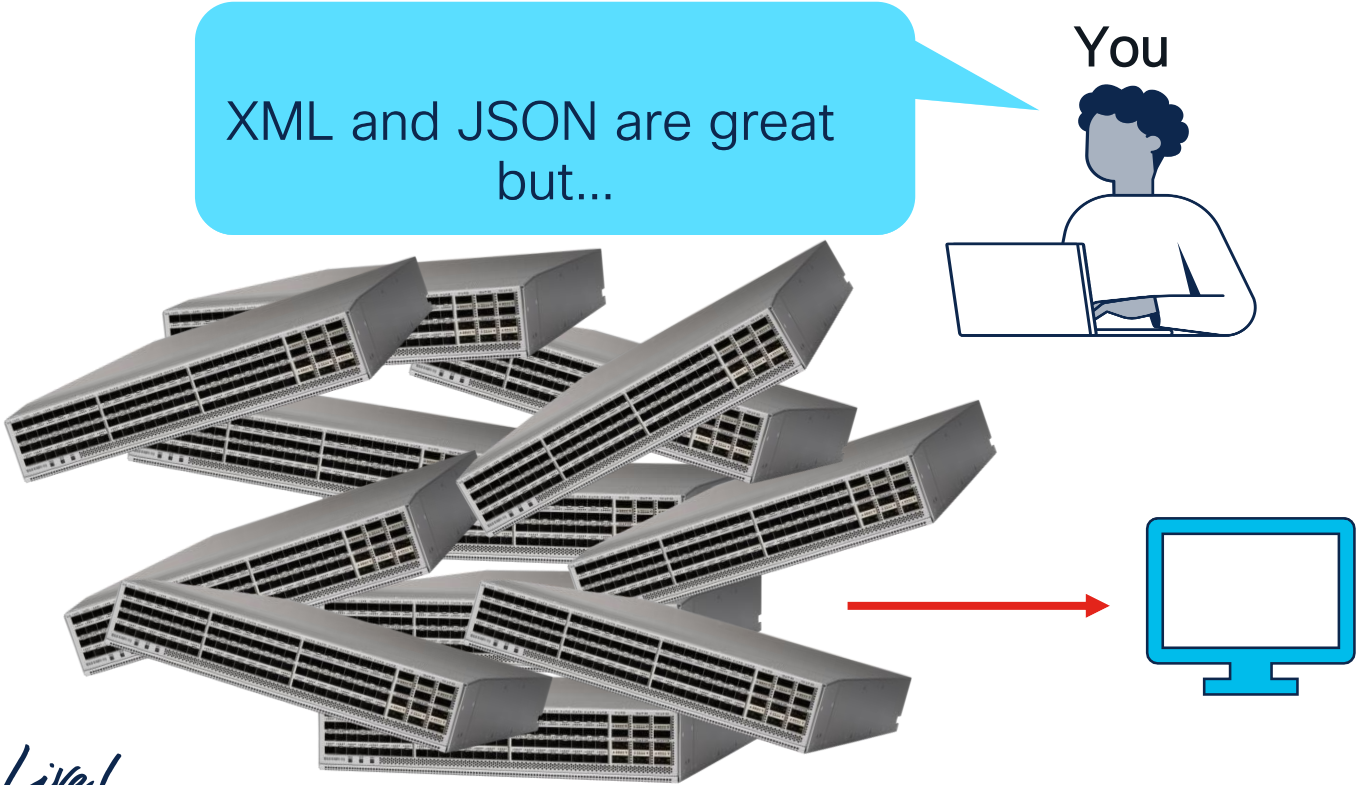
Data Encoding

Three Options

- XML (eXtensible Markup Language)
- JSON (JavaScript Object Notation)
- GPB (Google Protocol Buffers)

Data Encoding

GPB - The Need for Speed



XML and JSON are great but...

Data Encoding

GPB (Google Protocol Buffers)

```
<interface>
  <name>eth1/49</name>
  <state>
    <counters>
      <in-broadcast-pkts>2</in-broadcast-pkts>
      <in-discards>0</in-discards>
      <in-errors>0</in-errors>
      <in-fcs-errors>0</in-fcs-errors>
      <in-multicast-pkts>30543</in-multicast-pkts>
      <in-octets>13320913920</in-octets>
      <in-unicast-pkts>5406026</in-unicast-pkts>
      <in-unknown-protos>0</in-unknown-protos>
      <out-broadcast-pkts>3</out-broadcast-pkts>
      <out-discards>0</out-discards>
      <out-errors>0</out-errors>
      <out-multicast-pkts>26070</out-multicast-pkts>
      <out-octets>143144868</out-octets>
      <out-unicast-pkts>1424051</out-unicast-pkts>
    </counters>
  </state>
</interface>
```

Variable Length Integers!
(varints)

```
1:"eth1/49"
2:{
  1:{
    1:2
    2:0
    3:0
    4:0
    5:30543
    6:13320913920
    7:5406026
    8:0
    9:3
    10:0
    11:0
    12:26070
    13:143144868
    14:1424051
  }
}
```

Data Encoding

GPB (Google Protocol Buffers)

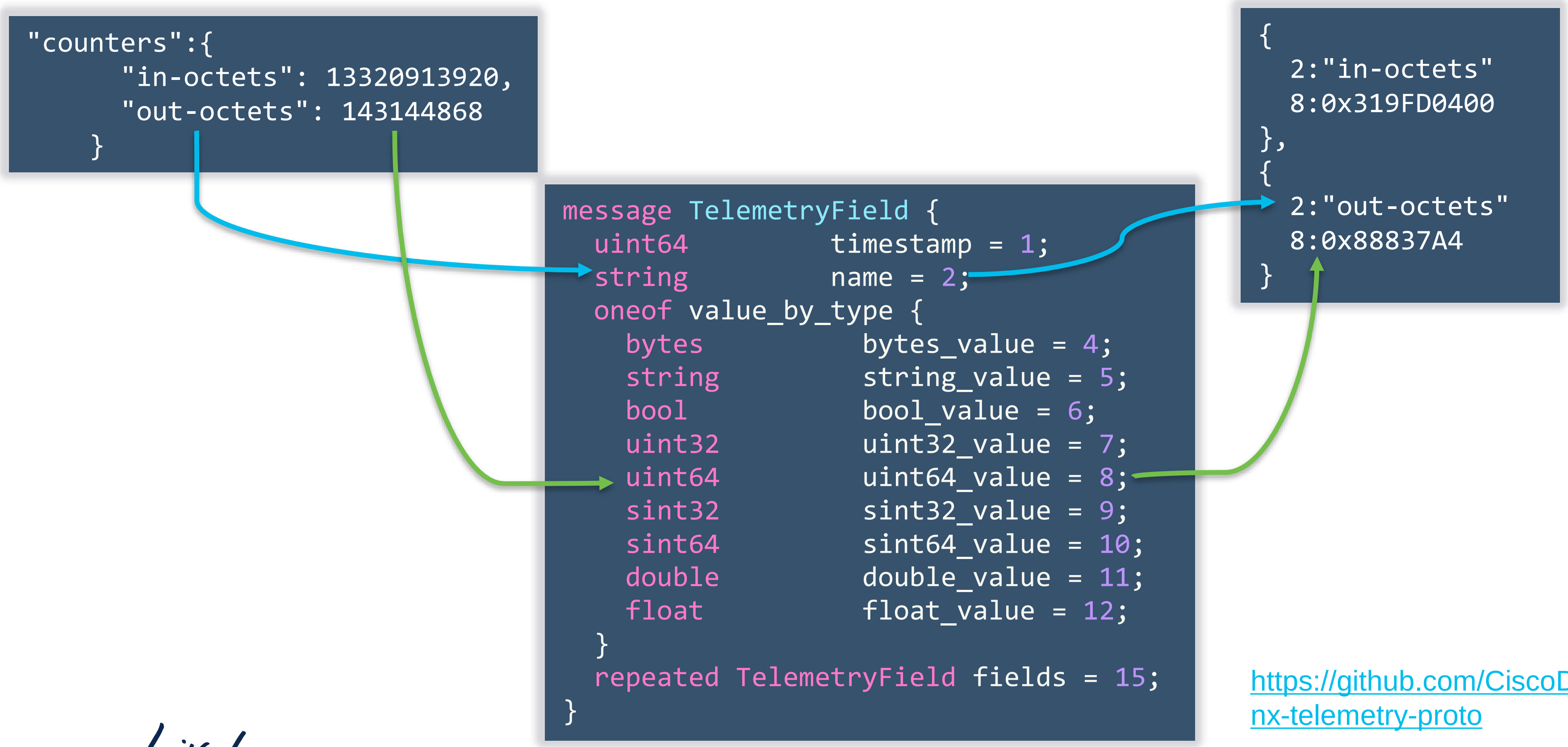
```
<interface>
  <name>eth1/49</name>
  <state>
    <counters>
      <in-broadcast-pkts>2</in-broadcast-pkts>
      <in-discards>0</in-discards>
      <in-errors>0</in-errors>
      <in-fcs-errors>0</in-fcs-errors>
      <in-multicast-pkts>30543</in-multicast-pkts>
      <in-octets>13320913920</in-octets>
      <in-unicast-pkts>5406026</in-unicast-pkts>
      <in-unknown-protos>0</in-unknown-protos>
      <out-broadcast-pkts>3</out-broadcast-pkts>
      <out-discards>0</out-discards>
      <out-errors>0</out-errors>
      <out-multicast-pkts>26070</out-multicast-pkts>
      <out-octets>143144868</out-octets>
      <out-unicast-pkts>142405
    </counters>
  </state>
</interface>
```

```
1:"eth1/49"
2:{
  1:{
    1:2
    2:0
    3:0
    4:0
    5:30543
    6:13320913920
    7:5406026
    8:0
    9:3
    10:0
    11:0
    12:26070
    13:143144868
    14:1424051
```

High wire efficiency,
but hard to develop the encoder and decoder

Data Encoding

GPB-KV (Key-Value)



```
"counters":{
      "in-octets": 13320913920,
      "out-octets": 143144868
    }
```

```
message TelemetryField {
  uint64         timestamp = 1;
  string         name = 2;
  oneof value_by_type {
    bytes          bytes_value = 4;
    string         string_value = 5;
    bool           bool_value = 6;
    uint32         uint32_value = 7;
    uint64         uint64_value = 8;
    sint32         sint32_value = 9;
    sint64         sint64_value = 10;
    double         double_value = 11;
    float          float_value = 12;
  }
  repeated TelemetryField fields = 15;
}
```

```
{
  2:"in-octets"
  8:0x319FD0400
},
{
  2:"out-octets"
  8:0x88837A4
}
```

https://github.com/CiscoDevNet/nx-telemetry-proto

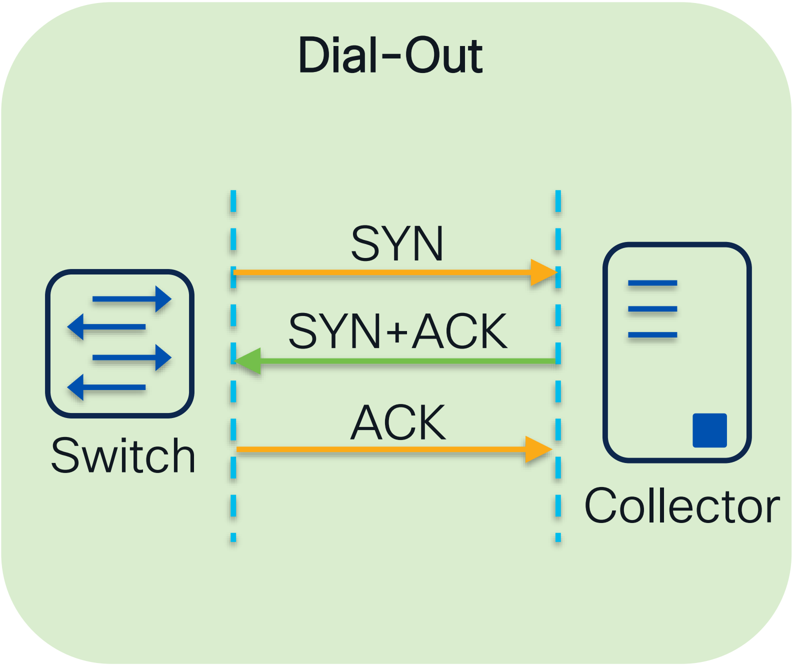
Building Blocks of Streaming Telemetry

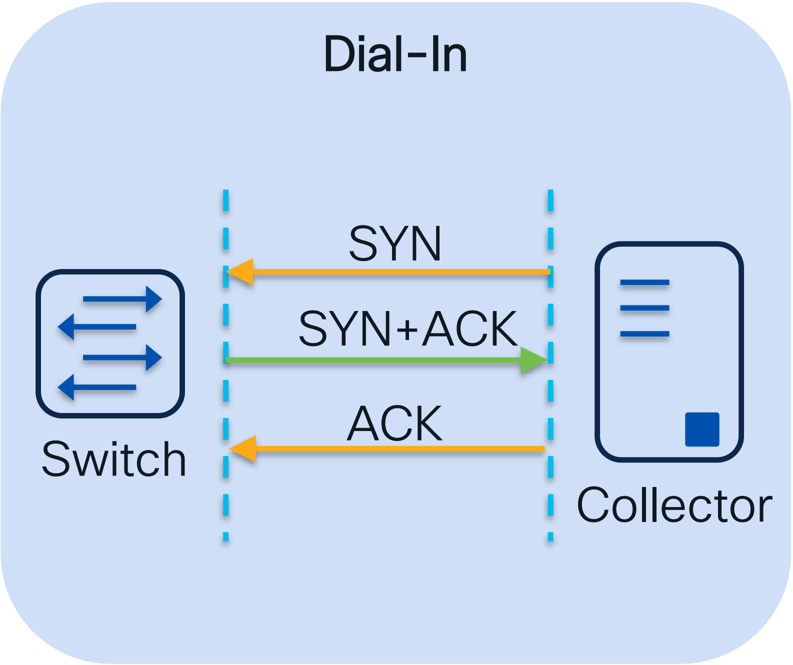
- Data Structure
- Data Frequency
- Data Encoding
- **Data Transport**

Data Transport

Dial-Out vs Dial-In

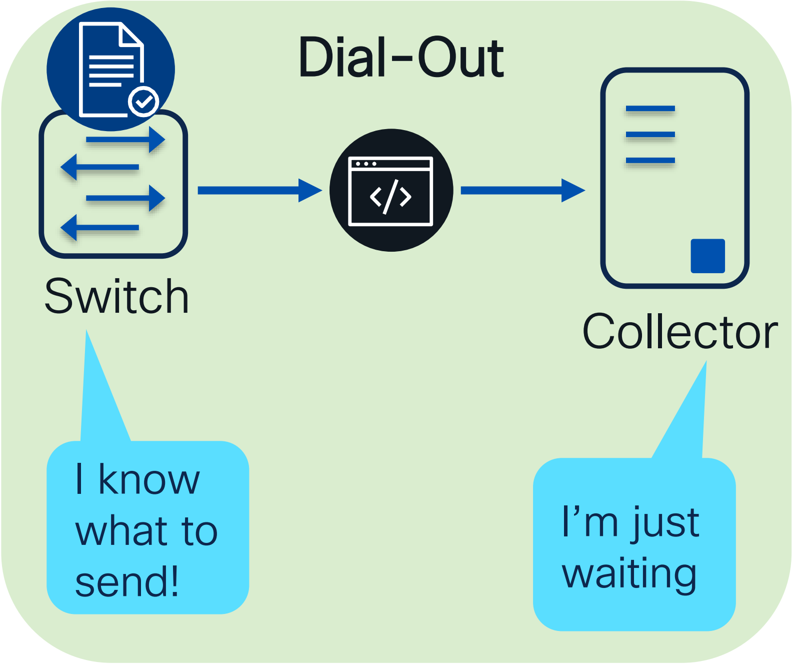
- TCP connection is always persistent in telemetry
- The difference is which part initializes the connection

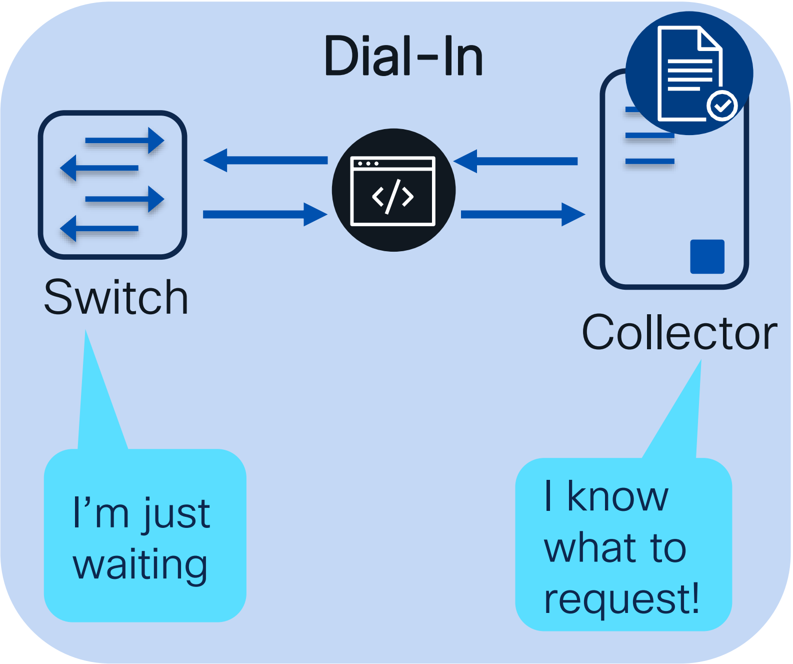




Data Transport

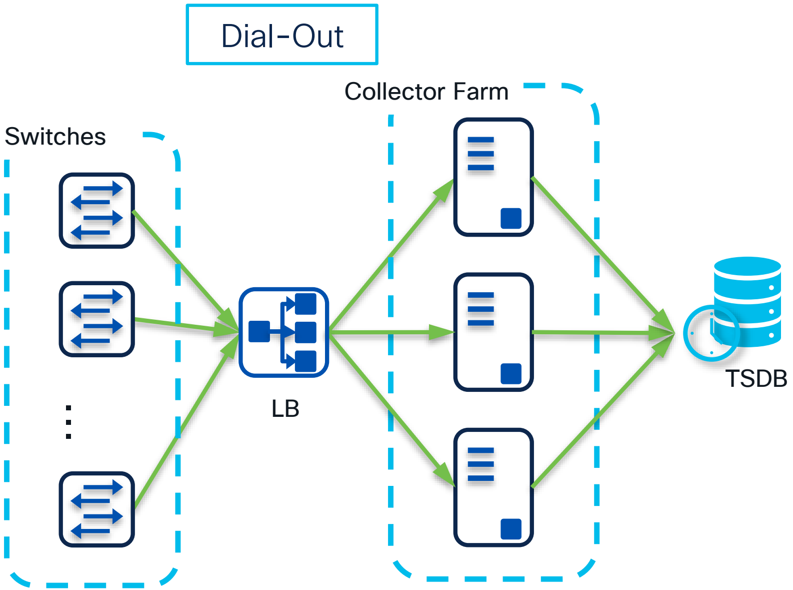
Dial-Out vs Dial-In: Configuration



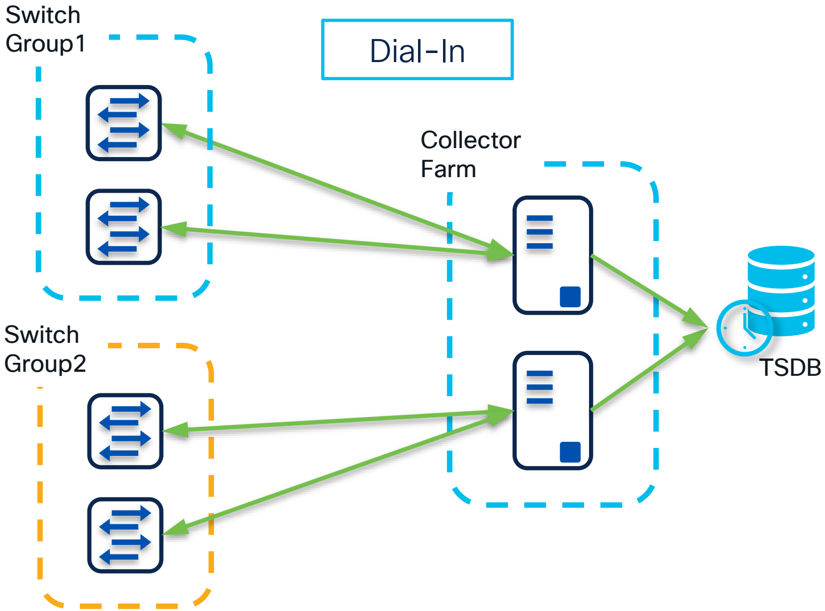


Data Transport

Design Considerations



Collectors can be set up behind load balancer, all switches stream to the same VIP of collector



To distribute workload, the collectors need to dial-in to different switch groups, need to keep the sensor configuration synchronized across the cluster

Data Transport

Dial-Out vs Dial-In

	Dial-Out	Dial-In
Protocol	Supports gRPC, HTTP, UDP as transport protocols	Only gNMI protocol supported
Configuration	Telemetry configuration must be setup on the switch	Single channel for both subscription and data transport
Access	No need to open a specific port to the management interface	External firewalls must allow ingress connection to the gNMI service
Load Balancing	Easier to load balance with collectors behind VIP	gNMI clients need to be distributed to handle entire fabric

Data Transport

gNMI - Features

Capabilities

Collect capabilities from device

Get

Collect current data values from device

Set

Modify data on the device

Subscribe

Subscribe to a data stream for a path in the data model

Data Transport

gNMI - Features

`sample`	Sample-based collection
`on_change`	Event-based collection
`target_defined`	Switch decides which frequency type
`suppress_redundant`	Don’t send redundant data
`heartbeat_interval`	Every X interval, give me an update

Data Transport

gNMI – NX-OS Implementation

RPC

- gNMI fully supported from NX-OS 9.3(5)
- NX-OS 10.4(x) updated to gNMI version 0.8.0

Security

- gNMI leverages TLS
- Mutual TLS supported

Data Encoding

- Native and OpenConfig models supported
- Supports Google Protobuf and JSON as encoding
- Wild card supported from NX-OS 10.2(2)

Open-Source Telemetry Stack

Open-Source Telemetry Stack

Requires Three Pieces

1 Collection Agent

A service that understands the data collected from the device

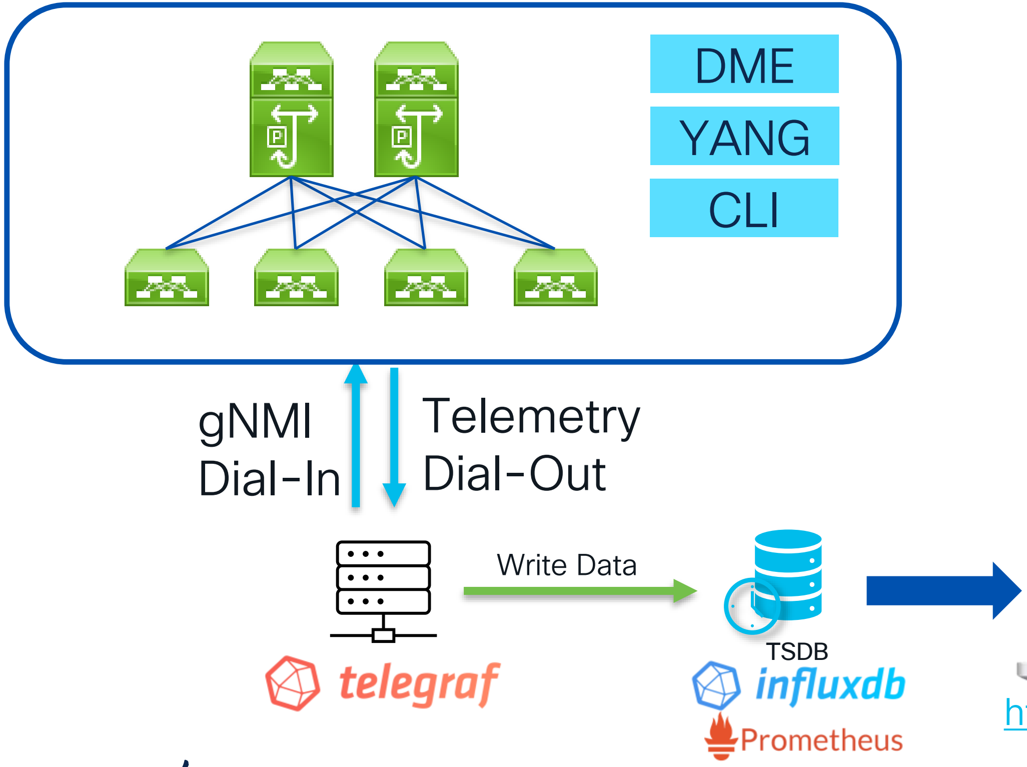
2 Time Series Database

A database with very precise time stamping that stores the collected data

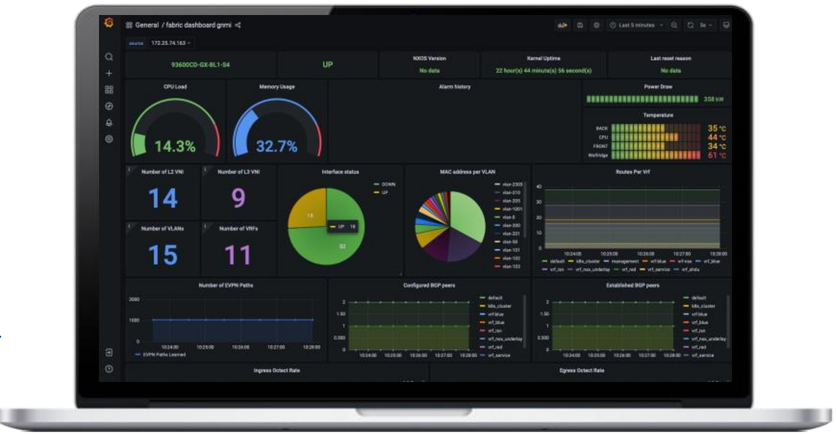
3 Using Stored Data

Integrating the data with an automation system, or graphically displaying the data

Open-Source Telemetry Stack



Grafana



https://github.com/dsx1123/telemetry_collector

Takeaways

- NX-OS has several choices for the data model, encoding, and transport options.
  - Customers can choose based on business requirements
- Most customers are interested in gNMI dial-in
  - Pros and cons between dial-out and dial-in
- Use OpenConfig model first, fall back to native model and DME

Continue Your Journey

- DEVNET-2135: Mastering Network Automation: Unleashing the Power of gNMI (Wednesday)
- DEVNET-2235: Industry Standard Streaming Telemetry with Cisco NX-OS (Thursday)
- Explore the NX-OS sandbox
- Install YANG suite to explore the models
- Install the TIG stack to get started with telemetry

Links

- API guide

https://developer.cisco.com/site/nxapi-dme-model-reference-api/?version=10.3(2)

- Supported YANG modules

https://github.com/YangModels/yang/tree/main/vendor/cisco/nx

- Cisco YANG Suite

https://developer.cisco.com/yangsuite/

- TIG dashboard

https://github.com/dsx1123/telemetry_collector

- BGP Python script

https://github.com/nmortari/gNMI-Programming/blob/main/gNMI-BGP-Setup.py

Continue your education

- Visit the Cisco Showcase for related demos
- Book your one-on-one Meet the Engineer meeting
- Attend the interactive education with DevNet, Capture the Flag, and Walk-in Labs
- Visit the On-Demand Library for more sessions at www.CiscoLive.com/on-demand

The bridge to possible

Thank you

#CiscoLive