



# Programmabilité Cisco Catalyst / IOS-XE

Community Live Webinar - <http://reseauxblog.cisco.fr/>

Jérôme Durand - Technical Solutions Architect  
Antoine Orsoni - Systems Engineer

6 décembre 2022

# Des nouveaux sujets tous les mois

La Communauté francophone Cisco grandit

Grâce à la participation de tous nos collaborateurs, notre communauté grandit pour vous offrir des nouveaux webinaires sur différentes technologies.

Inscrivez-vous et participez !

[Prochains événements](#)

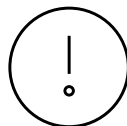
[Événements précédents](#)

[Webex et le RGPD \(2022\)](#)

le mardi 13 décembre 2022

[Redécouvrez Cisco SD-WAN](#)

le mardi 17 janvier 2023



# Connectez, Engagez, Collaborez !

## Solutions

Acceptez les solutions qui sont correctes et complimentez ceux qui vous ont aidé ! Aidez autres utilisateurs à trouver les réponses correctes dans la fenêtre de recherche.

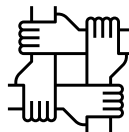
Accepter comme solution

## Compliments

Mettez en évidence les autres membres. Les votes utiles motivent les membres enthousiastes en leur offrant un signe de reconnaissance !



0 Compliments



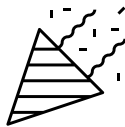
# Community Helping Community

Aidez-nous à franchir un nouveau challenge !

Cette année la communauté a choisi deux associations au profit des enfants, mais aussi pour soutenir l'approvisionnement des hôpitaux et des réfugiés en Ukraine.

Nous avons besoin de votre aide pour parvenir à 13'000 interactions et être en mesure d'atteindre notre objectif.

Si vous souhaitez collaborer vous n'avez pas besoin de verser de l'argent, votre activité lorsque vous vous connectez à la communauté sera comptabilisée pour nous aider. [Cliquez ici](#)



Community  
Helping  
Community



Améliorer notre  
monde ensemble



# Jérôme Durand

Présentateur



Jérôme Durand a rejoint Gip Renater en 2002. Il a commencé par travailler sur des projets de R&D, mais est rapidement devenu responsable d'exploitation en 2006. En 2009, il prend la responsabilité de l'équipe services. Jérôme a rejoint Cisco en 2011 en tant qu'expert des technologies de routage et de commutation. Actuellement, il est très impliqué dans la programmation et l'automatisation des réseaux, notamment les solutions SD-WAN et SD-Access. Il est également l'auteur de la RFC 7454 - BGP Operations and Security.

# Antoine Orsoni

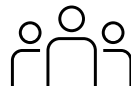
Présentateur



Antoine Orsoni a rejoint Cisco en 2017. Après un rôle de spécialiste Enterprise Networking, il accompagne depuis 2022 des clients opérateurs télécoms français. Il travaille avec ses clients sur des projets de SP Routing, Data Center et Automatisation.

## Téléchargez la présentation !

<https://bit.ly/WEB1sld-dec22>



## Agenda



- Introduction à la programmabilité YANG
- YANG
- Les modèles YANG en pratique



- NETCONF
- RESTCONF
- GNXI



- Télémétrie
- Démo YANG Suite
- Pour poursuivre l'aventure...

# Introduction à la programmabilité

# Do it Yourself or Integrated strategy



NETCONF  
RESTCONF  
gRPC  
gNMI  
gNOI



DNA Automation  
DNA Assurance  
SD-Access



DIY



Integrated





# Do it Yourself or Integrated strategy



NETCONF  
RESTCONF  
gRPC  
gNMI



DNA Automation  
DNA Assurance  
SD-Access



DIY

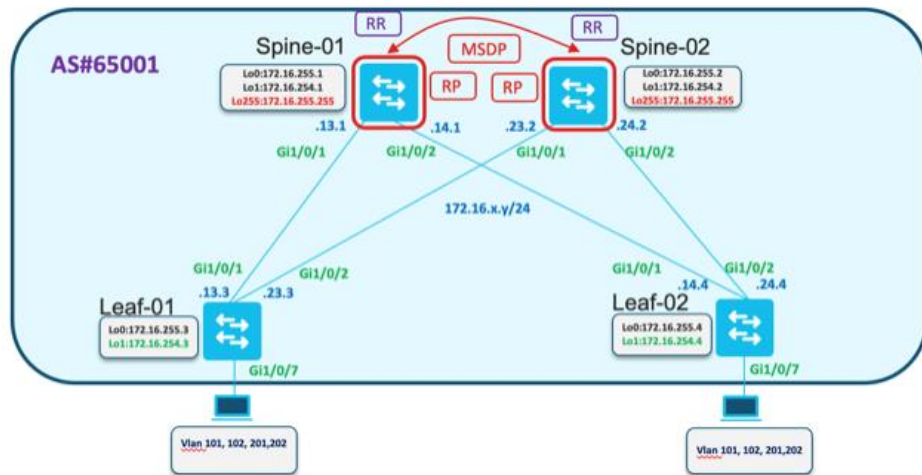


Integrated



# BGP EVPN VXLAN solution management with Ansible

The Ansible BGP EVPN solution is publicly available now on GitHub with step-by-step examples of getting started with DAG, L2VNI and L3VNI.



Try it out yourself and engage directly on GitHub:  
<https://github.com/Cat9kEVPN/cat9k-evpn-ansible>

## Quick start

For the quick start with DAG provisioning next steps have to be executed:

### Step 1

On this step Underlay is provisioned.

#### Step 1a

Edit `inventory.yml` and set proper name and management ip address.

```
all:
  children:
    leaf:
      hosts:
        Leaf-01:
          ansible_host: 10.1.1.1
<...snip...>
```

Detailed information could be found [here](#)

#### Step 1b

Edit `group_vars/all.yml` and set proper login and password

```
ansible_connection: ansible.netcommon.network_cli
ansible_network_os: cisco.ios.ios
ansible_python_interpreter: "python"
ansible_user: cisco
ansible_ssh_pass: cisco123
```

`ansible_user`: must have privilege level 15. Example of the configuration is below

```
username cisco privilege 15 password 8 cisco123
```

If enable password should be used, check the [Enable Mode](#) documentation.

Detailed information could be found [here](#)

#### Step 1c

Edit `host_vars/<hostname>.yml` and set required parameters for underlay

# Deployment of BGP EVPN VXLAN with Ansible

The screenshot displays a Cisco Modeling Labs environment with two hosts and a central Ansible terminal window.

**Host 1 (Left):** `cisco@ubuntu-1:/etc/netplan$`

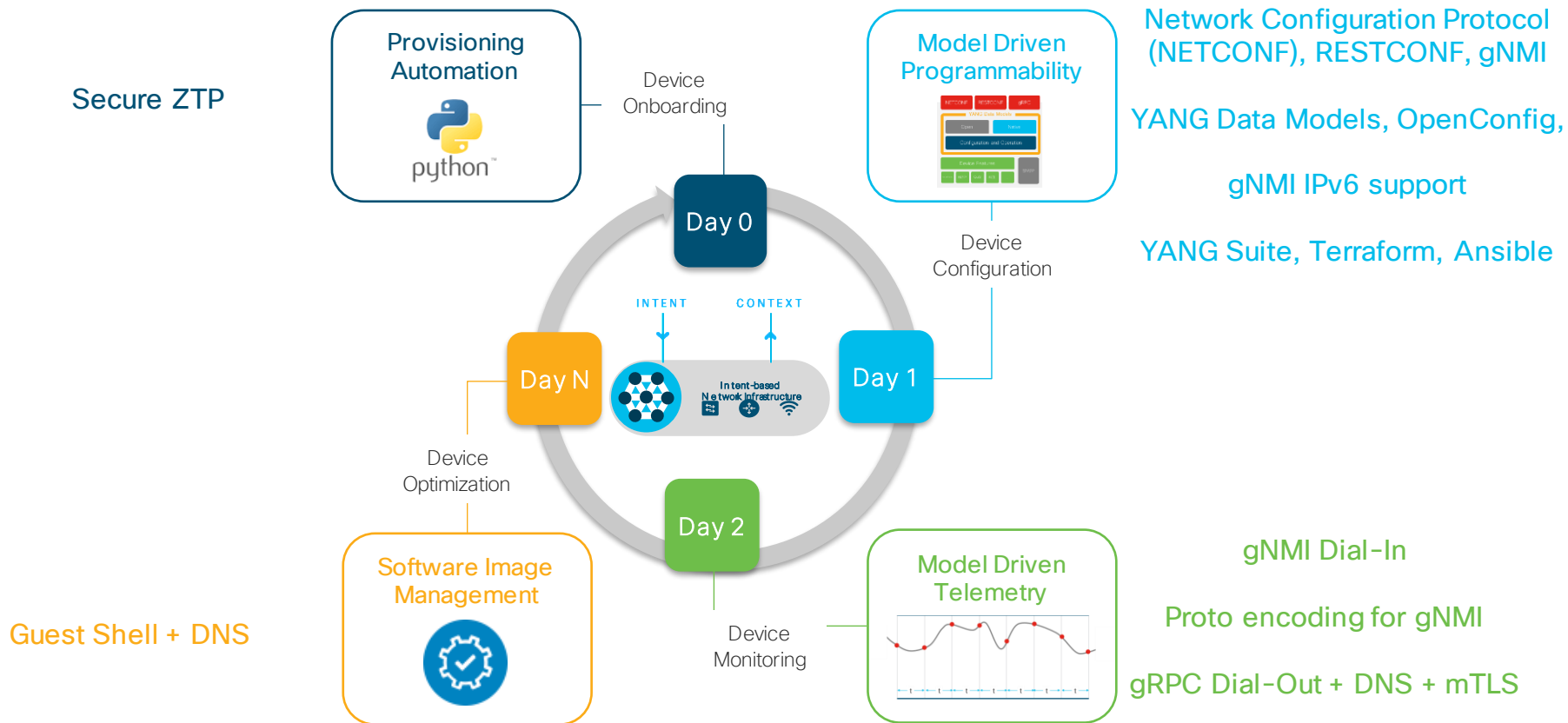
**Host 2 (Bottom Left):** `cisco@ubuntu-2:/etc/netplan$`

**Terminal (Right):** Shows the execution of an Ansible playbook for `inventory_raj.yml`. The terminal output includes the following configuration for `sw-access-leaf-1`:

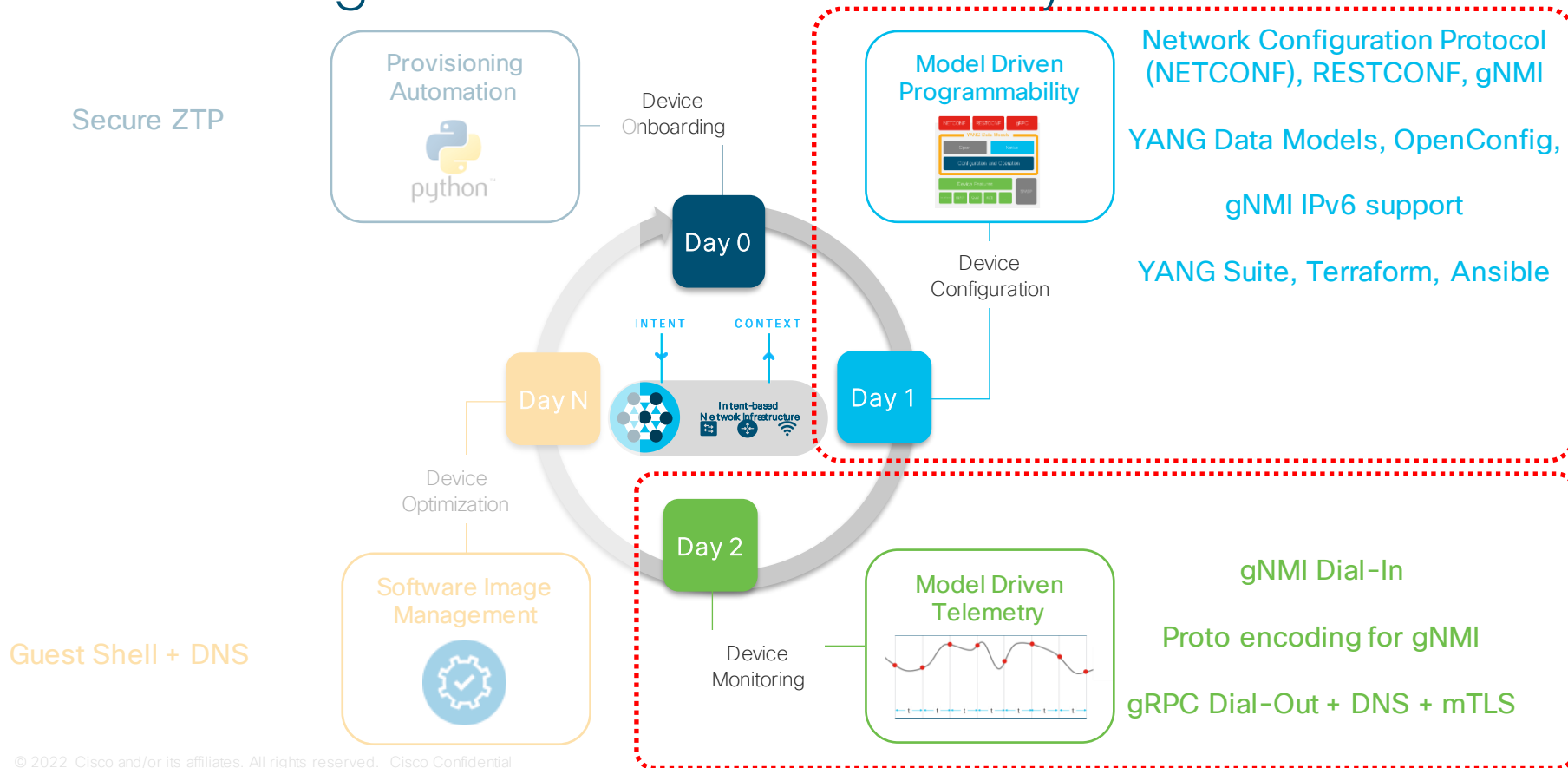
```
---
#
# leafs:
#
# hosts:
#
sw-access-leaf-1:
  ansible_host: 198.18.1.31
  role: leaf
  rid: 100.65.0.2
  vrfns:
    - vrf_name: VRF_OVERLAY_2010
      vrf_core_vlan_id: 2010
      vrf_l3_vni_id: 20010
      vrf_edge_vlan:
        - id: 10
          ip: 10.10.10.1
          mask: 255.255.255.0
        - id: 20
          ip: 10.20.10.1
          mask: 255.255.255.0
        - id: 30
          ip: 10.30.10.1
          mask: 255.255.255.0
    - vrf_name: VRF_OVERLAY_2020
      vrf_core_vlan_id: 2020
      vrf_l3_vni_id: 20020
      vrf_edge_vlan:
        - id: 40
          ip: 20.10.10.1
```

The terminal also shows the execution of `ansible-playbook -i inventory_raj.yml` and the resulting output for the `sw-access-leaf-1` host, indicating successful execution of the `Layer_3_Design_1X` design.

# IOS XE Programmable Device Lifecycle

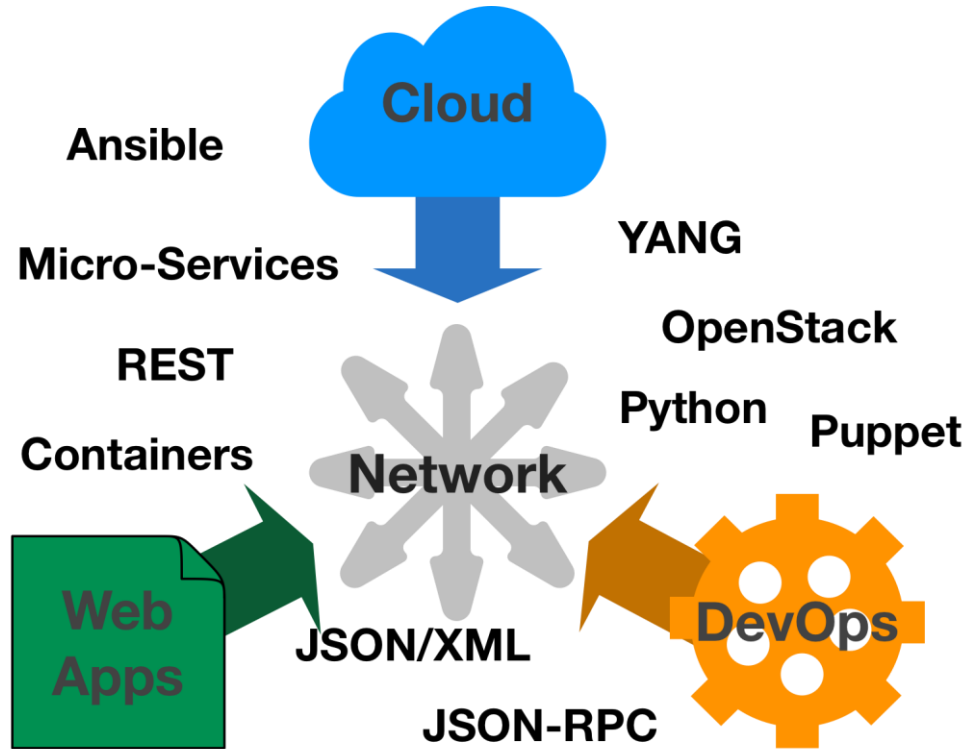


# IOS XE Programmable Device Lifecycle



Why Model Driven  
Programmability ?

# The Network is No Longer Isolated





Congrats !  
Your configuration has  
been successfully deployed  
on 217 / 276 devices



# What about SNMP?

*SNMP works  
“reasonably well for  
device monitoring”*

RFC 3535: Overview of the 2002 IAB  
Network Management Workshop – 2003  
<https://tools.ietf.org/html/rfc3535>



Standard MIB?

**Scalability?**

**NO**

Retrieve full device config?

Filtered output?

**Configuration?**

# RFC 3535: What is Needed?

- A programmatic interface for device configuration
- Separation of Configuration and State Data
- Ability to configure "services" NOT "devices"
- Integrated error checking and recovery



# Programmable Interfaces

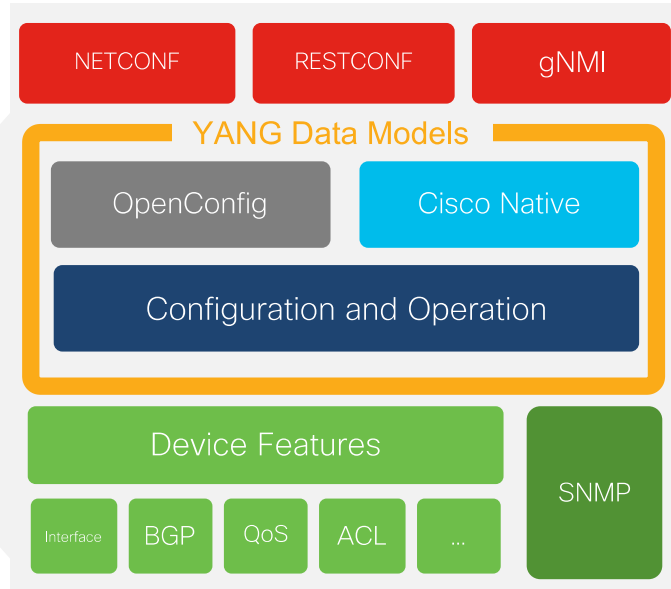
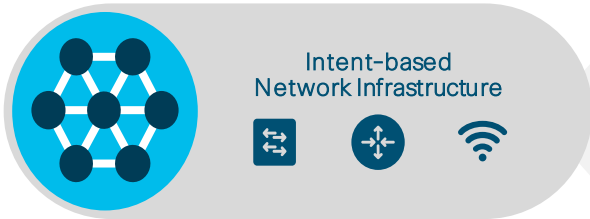
CLI

SNMP

WebUI

The NETCONF, RESTCONF and gNMI are **programmatic** interfaces that provide **additional** methods for interfacing with the IOS XE device – Just like the CLI, SNMP, and WebUI is used for configuration changes and operational metrics so can the programmatic interfaces of NETCONF, RESTCONF and gNMI

YANG data models define the data that is available for configuration and streaming telemetry




YANG

# Why models ?

```
SA-Edge-1#sh int gi 1/0/17
GigabitEthernet1/0/17 is up, line protocol is up (connected)
  Hardware is Gigabit Ethernet, address is 70d3.7984.0f11 (bia 70d3.7984.0f11)
  MTU 9100 bytes, BW 1000000 Kbit/sec, DLY 10 usec,
    reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Full-duplex, 1000Mb/s, media type is 10/100/1000BaseTX
  input flow-control is on, output flow-control is unsupported
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/2000/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
    23530484 packets input, 3412544575 bytes, 0 no buffer
    Received 3871554 broadcasts (1816130 multicasts)
    0 runts, 0 giants, 0 throttles
    0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
    0 watchdog, 1816130 multicast, 0 pause input
    0 input packets with dribble condition detected
  24346526 packets output, 5625088240 bytes, 0 underruns
  Output 281128 broadcasts (0 multicasts)
```

# Why models ?

```
SA-Edge-1#sh int gi 1/0/17
GigabitEthernet1/0/17 is up, line protocol is up (connected)
  Hardware is Gigabit Ethernet, address is 70d3.7984.0f11 (bia 70d3.7984.0f11)
  MTU 9100 bytes, BW 1000000 Kbit/sec, DLY 10 usec,
    reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Full-duplex, 1000Mb/s, media type is 10/100/1000BaseTX
  input flow-control is on, output flow-control is unsupported
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/2000/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
    23530484 packets input, 3412544575 bytes, 0 no buffer
    Received 3871554 broadcasts (1816130 multicasts)
    0 giants, 0 throttles
    0 errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
    watchdog, 1816130 multicast, 0 pause input
    0 input packets with dribble condition detected
  24346526 packets output, 5625088240 bytes, 0 underruns
  Output 281128 broadcasts (0 multicasts)
```



# What is a Data Model?

A data model is simply a well understood and agreed upon method to describe "something". As an example, consider this simple "data model" for a person.

- *Person*
  - **Gender** – male, female, other
  - **Height** – Feet/Inches or Meters
  - **Weight** – Pounds or Kilos
  - **Hair Color** – Brown, Blond, Black, Red, other
  - **Eye Color** – Brown, Blue, Green, Hazel, other

# Where do YANG models come from ?



Industry  
Standard

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



Vendor  
Specific

- **Vendor definition**  
(Cisco, Juniper, etc.)
- **Unique to Vendor Platforms**  
`cisco-memory-stats.yang`  
`cisco-flow-monitor`  
`cisco-qos-action-qlimit-cfg`

<https://github.com/YangModels/yang>  
<https://github.com/openconfig/public>



# What is OpenConfig?

Models Designed by Operators for Operators

*“OpenConfig’s initial focus is on compiling a consistent set of vendor-neutral data models based on actual operational needs from use cases and requirements from multiple network operators.”*

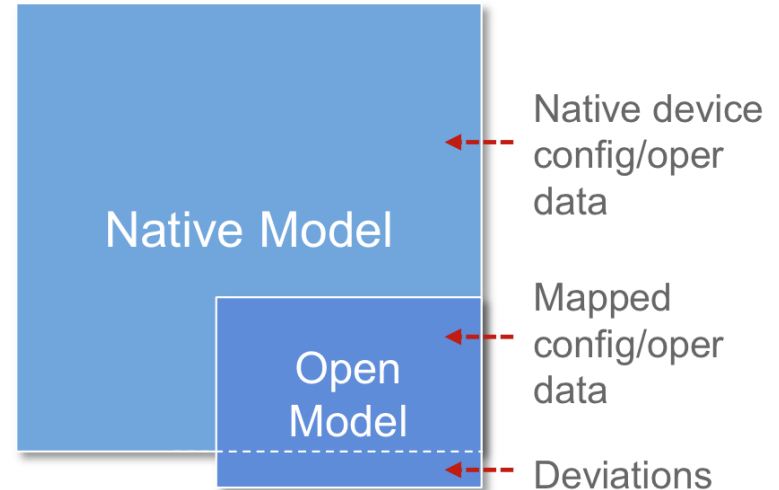
OpenConfig FAQ:  
[www.openconfig.com](http://www.openconfig.com)



# What is OpenConfig?

## Models Designed by Operators for Operators

- Focused on creating vendor-neutral data models written in YANG
- Models combine both configuration and operational data
- Model coverage still limited with an active development community
- Support from multiple routing vendors (e.g. Cisco, Juniper, Arista)



# A look at a Standard YANG Model for Interfaces

```
DevNet$ pyang -f tree 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 admin-status       enumeration {if-mib}?
    |   +--ro oper-status        enumeration
    |   +--ro last-change?       yang:date-and-time
    |   +--ro if-index           int32 {if-mib}?
    |   +--ro phys-address?      yang:phys-address
```

*Example output edited for simplicity and brevity*

[dna-dne-code/intro-mdp/yang/models/ietf-interfaces.yang](https://github.com/dna-dne-code/intro-mdp/yang/models/ietf-interfaces.yang)

# Programmable Interfaces

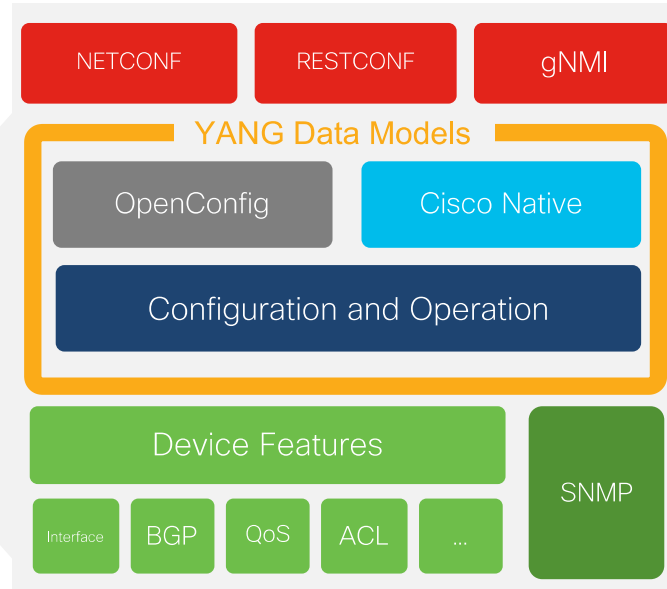
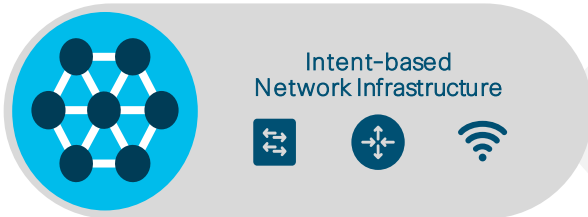
CLI

SNMP

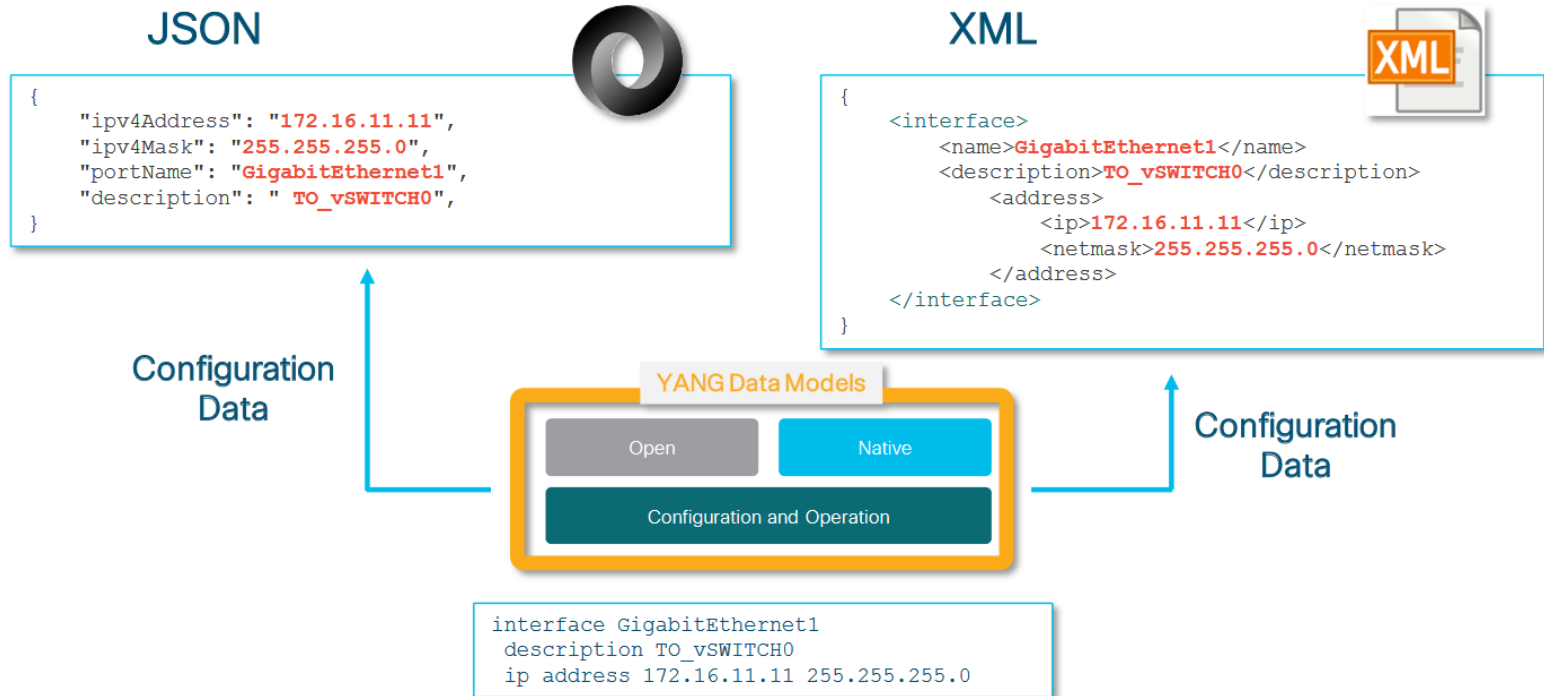
WebUI

The NETCONF, RESTCONF and gNMI are programmatic interfaces that provide additional methods for interfacing with the IOS XE device – Just like the CLI, SNMP, and WebUI is used for configuration changes and operational metrics so can the programmatic interfaces of NETCONF, RESTCONF and gNMI

YANG data models define the data that is available for configuration and streaming telemetry



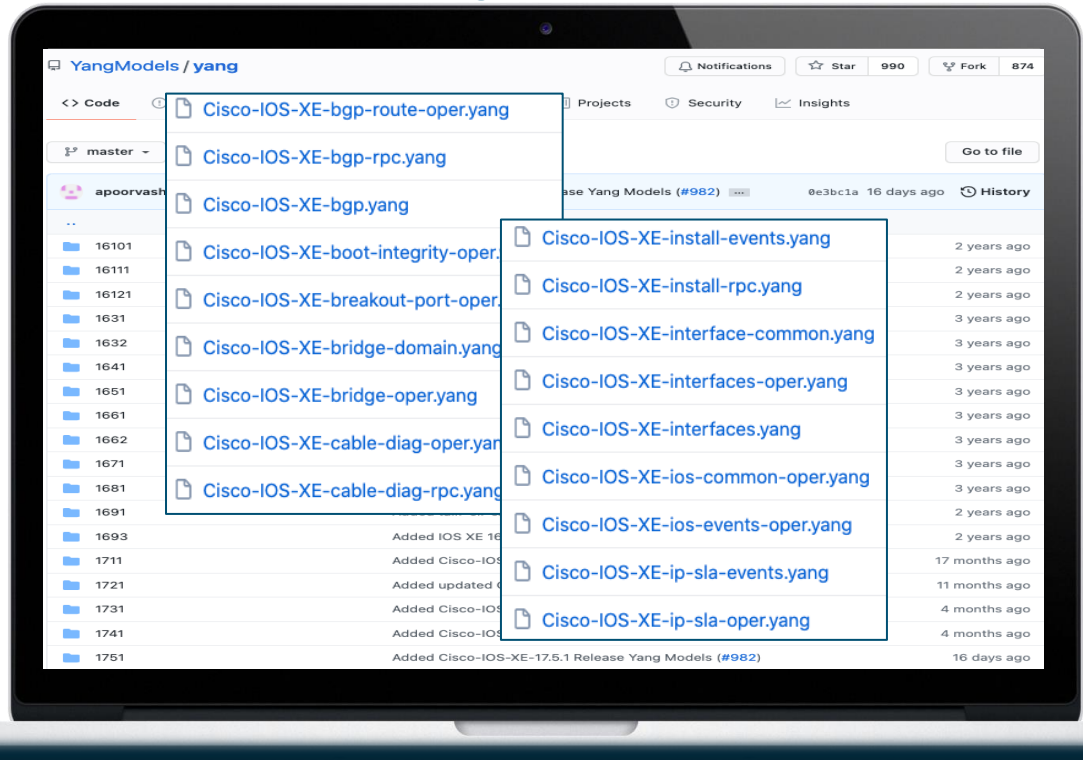
# YANG Relationship to JSON and XML



# IOS XE - YANG Model Coverage on GitHub

RFC7950 states that “YANG is a data modeling language used to model configuration data, state data, Remote Procedure Calls, and notifications for network management protocols”

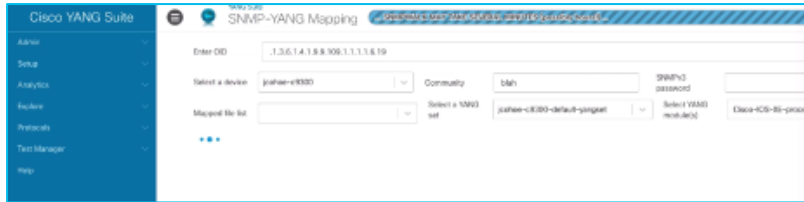
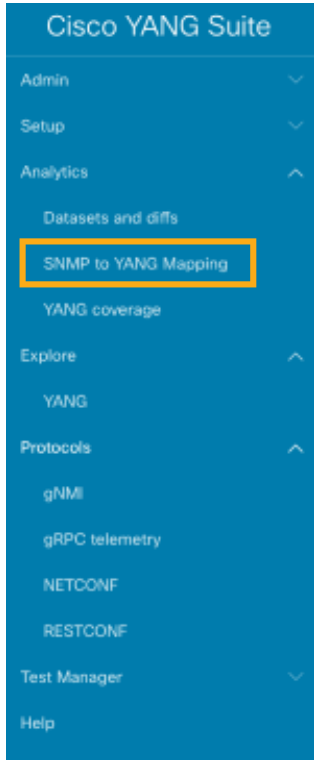
YANG module name.yang	Description
Cisco-IOS-XE-native	running-config
Cisco-IOS-XE-{feature}-cfg	Feature configuration
Cisco-IOS-XE-{feature}-oper	Feature operational data
Cisco-IOS-XE-{feature}-rpc	Actions
Cisco-evpn-service	EVPN service abstraction
OpenConfig-{feature}	abstraction for config & oper



<https://github.com/YangModels/yang/tree/master/vendor/cisco/xe>

# SNMP to YANG migration mapping

Ease the transition from SNMP OID to YANG Xpath and easily verify the responses from both.



OID: .1.3.6.1.4.1.9.9.109.1.1.1.1.6.19

**MIB Path** .iso.org.dod.internet.private.enterprises.9.9.276.1.6.1.1.2.7.84.101.49.47.48.47.52

**YANG Path** /cpu-usage/cpu-utilization/five-seconds

**YANG Suite**

SNMPGET:  
.iso.org.dod.internet.private.enterprises.9.9.276.1.6.1.1.2.7.84.101.49.47.48.47.52 = INTEGER: 12

YANG GET:  
/cpu-usage/cpu-utilization/five-seconds  
VALUES:  
['1']

Run  
Save  
Delete

Right click > Run to retrieve from SNMP and NETCONF simultaneously.

This solution utilizes the Python library for “fuzzy matching” of OID and XPATH values to identify most accurate match.

Modèles YANG en pratique



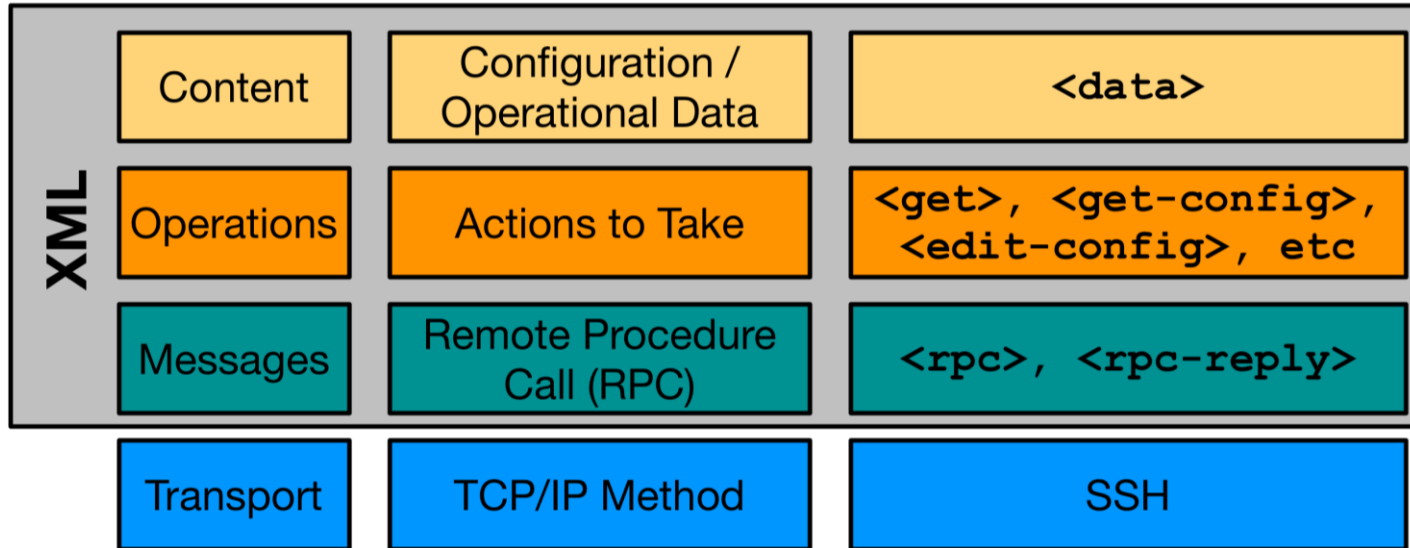
# Polling Question 1

Où puis-je trouver les modèles YANG pour IOS-XE ?

- 1) Cisco.com
- 2) GitHub
- 3) Contact TAC

NETCONF

# NETCONF protocol



# Understanding the Capabilities List

```
urn:ietf:params:netconf:base:1.0  
urn:ietf:params:netconf:base:1.1
```

```
urn:ietf:params:xml:ns:yang:ietf-interfaces?  
module=ietf-interfaces&revision=2014-05-08&  
features=pre-provisioning,if-mib,arbitrary-names&  
deviations=ietf-ip-devs
```

*Example edited for simplicity and brevity*

## Two General Types

- Base NETCONF capabilities (= NETCONF version)
- Data Models Supported

# YANG 1.0 to 1.1 transition – YANG advertisement

Legacy YANG 1.0 capabilities exchange and NETCONF  
"hello" message will soon be unsupported

YANG 1.1 example: "ietf-yang-library" to  
retrieve supported YANG modules

```

V> auto@pod24-xelab-7/nc5
V> auto@pod24-xelab-7/nc5 netconf-console --host c9300 --port 830 --user admin --password Cisco123 --hello
/home/autoncc/v/lib/python3.8/site-packages/paramiko/xx_eccdh_nist.py:39: CryptographyDeprecationWarning: encode_point has been deprecated on EllipticCurvePublicNumbers and will be
compressed and uncompressed point encoding.
  m_add_string(self.Q_C_public_numbers(), encode_point(C))
/home/autoncc/v/lib/python3.8/site-packages/paramiko/xx_eccdh_nist.py:91: CryptographyDeprecationWarning: Support for unsafe construction of public numbers from encoded data will be
self.Q_S = ec.EllipticCurvePublicNumbers.from_encoded_point(
/home/autoncc/v/lib/python3.8/site-packages/paramiko/xx_eccdh_nist.py:103: CryptographyDeprecationWarning: encode_point has been deprecated on EllipticCurvePublicNumbers and will be
compressed and uncompressed point encoding.
  m_add_string(self.Q_C_public_numbers(), encode_point(C))
<?xml version='1.0' encoding='UTF-8'?>
<hello xmlns:nc='urn:ietf:params:xml:ns:netconf:base:1.0'*>
  <nc:capability><
  <nc:capability>urn:ietf:params:netconf:base:1.0/nc:capability
  <nc:capability>urn:ietf:params:netconf:base:1.1/nc:capability
  <nc:capability>urn:ietf:params:netconf:capability:writeable-running:1.0/nc:capability
  <nc:capability>urn:ietf:params:netconf:capability:rollback-errors:1.0/nc:capability
  <nc:capability>urn:ietf:params:netconf:capability:validate:1.0/nc:capability
  <nc:capability>urn:ietf:params:netconf:capability:validate:1.1/nc:capability
  <nc:capability>urn:ietf:params:netconf:capability:ssh:1.0/nc:capability
  <nc:capability>urn:ietf:params:netconf:capability:notification:1.0/nc:capability
  <nc:capability>urn:ietf:params:netconf:capability:interleave:1.0/nc:capability
  <nc:capability>urn:ietf:params:netconf:capability:with-defaults:1.0/basic-mode=public&also-supported-report-all-tagged,report-all/nc:capability
  <nc:capability>urn:ietf:params:netconf:capability:yang-library:1.0?revision=2016-06-21&module-set-id=702f66e9d4c6b47ccf523dcfba47/nc:capability
  <nc:capability>http://tail-f.com/ns/netconf/actions/1.0/nc:capability
  <nc:capability>http://cisco.com/ns/cisco-xe-ietf-ip-deviation/nc:module-cisco-xe-ietf-ip-deviation&revision=2016-08-18/nc:capability
  <nc:capability>http://cisco.com/ns/cisco-xe-ietf-ip-v4-unicast-routing-deviation/nc:module-cisco-xe-ietf-ip-v4-unicast-routing-deviation&revision=2015-09-11/nc:capability
  <nc:capability>http://cisco.com/ns/cisco-xe-ietf-pv6-unicast-routing-deviation/nc:module-cisco-xe-ietf-pv6-unicast-routing-deviation&revision=2015-09-11/nc:capability
  <nc:capability>http://cisco.com/ns/cisco-xe-ietf-ospf-deviation/nc:module-cisco-xe-ietf-ospf-deviation&revision=2018-08-09/nc:capability
  <nc:capability>http://cisco.com/ns/cisco-xe-ietf-routing-deviation/nc:module-cisco-xe-ietf-routing-deviation&revision=2018-07-09/nc:capability
  <nc:capability>http://cisco.com/ns/cisco-xe-openconfig-ocl-deviation/nc:module-cisco-xe-openconfig-ocl-deviation&revision=2017-08-25/nc:capability
  <nc:capability>http://cisco.com/ns/cisco-xe-openconfig-oft-deviation/nc:module-cisco-xe-openconfig-oft-deviation&revision=2018-12-05/nc:capability
  <nc:capability>http://cisco.com/ns/cisco-xe-openconfig-isis-deviation/nc:module-cisco-xe-openconfig-isis-deviation&revision=2018-12-05/nc:capability
  <nc:capability>http://cisco.com/ns/cisco-xe-openconfig-lldp-deviation/nc:module-cisco-xe-openconfig-lldp-deviation&revision=2018-07-25/nc:capability
  <nc:capability>http://cisco.com/ns/cisco-xe-openconfig-mpis-deviation/nc:module-cisco-xe-openconfig-mpis-deviation&revision=2019-06-27/nc:capability
  <nc:capability>http://cisco.com/ns/cisco-xe-openconfig-segment-routing-deviation/nc:module-cisco-xe-openconfig-segment-routing-deviation&revision=2018-12-05/nc:capability
  <nc:capability>http://cisco.com/ns/cisco-xe-openconfig-system-management-deviation/nc:module-cisco-xe-openconfig-system-management-deviation&revision=2019-07-01/nc:capability
  </hello>
  
```

```

Sending:
#275
<nc:rpc xmlns:nc='urn:ietf:params:xml:ns:netconf:base:1.0' message-id='urn:uuid:b966c2ff-a59b-46a3-b1cc5e97e44'>
  <nc:get>
  <nc:filter>
    <modules-state xmlns='urn:ietf:params:xml:ns:yang:ietf-yang-library'/*>
  </nc:filter>
</nc:rpc>

##

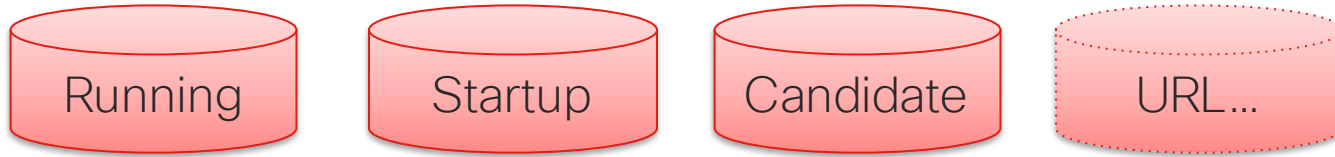
Received message from host

<?xml version='1.0' ?>
<rpc-reply message-id='urn:uuid:b966c2ff-a59b-46a3-b1cc5e97e44' xmlns='urn:ietf:params:xml:ns:netconf:base:1.0' xmlns:nc='urn:ietf:para

<data>
  <modules-state xmlns='urn:ietf:params:xml:ns:yang:ietf-yang-library'
  <module-set-id=4702f66e9d4c6b47ccf523dcfba47/module-set-id>
    <module>
      <name>BGPA-MIB</name>
      <revision>1994-05-05</revision>
      <schemas-http://localhost:9930/restconf/tailf/modules/BGPA-MIB/1994-05-05/</schemas>
      <namespace-urn:ietf:params:xml:ns:yang:smv2:BGPA-MIB</namespace>
      <conformance-type>implement</conformance-type>
    </module>
    <module>
      <name>BRIDGE-MIB</name>
      <revision>2005-09-19</revision>
      <schemas-http://localhost:9930/restconf/tailf/modules/BRIDGE-MIB/2005-09-19/</schemas>
      <namespace-urn:ietf:params:xml:ns:yang:smv2:BRIDGE-MIB</namespace>
      <conformance-type>implement</conformance-type>
    </module>
  
```

If the desired application previously parsed the NETCONF "hello" message to retrieve the supported YANG models, the parsing must be modified to reflect how version 1.1 advertises via "ietf-yang-library" instead of the NETCONF "hello" message.

# NETCONF Data Stores



- Data stores are named containers that *may* hold an entire copy of the configuration
- Not all data stores are supported by all devices
- **Running** is the only mandatory data store
- Not all data stores are writable
  - Check the device's capabilities
  - To make changes to a non-writable data store, copy from a writable one

# Operations – NETCONF Actions

Operation	Description
<get>	Retrieve running configuration and device state information
<get-config>	Retrieve all or part of specified configuration data store
<edit-config>	Loads all or part of a configuration to the specified configuration data store
<copy-config>	Replace an entire configuration data store with another
<delete-config>	Delete a configuration data store
<commit>	Copy candidate data store to running data store
<lock> / <unlock>	Lock or unlock the entire configuration data store system
<close-session>	Graceful termination of NETCONF session
<kill-session>	Forced termination of NETCONF session

# Challenges of most of tools out there



## Create

Convert service request into valid config

**Easy:** need to check resource availability



## Update

**Config change** from change in service

**Medium:** need to add/release resources



## Delete

**Remove service instance, release resources**

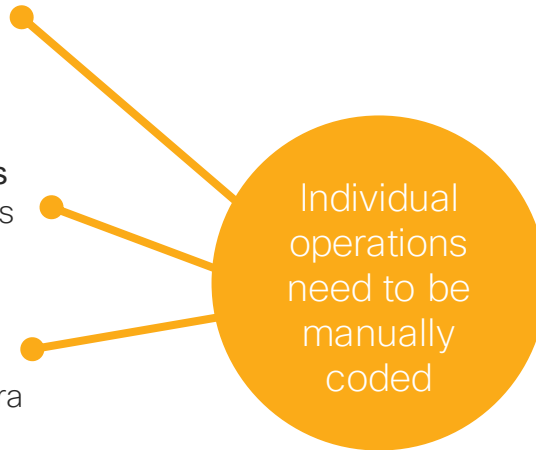
**Hard:** need reference counting of resources



## Repair

Recover from arbitrary resource changes

**Very Hard:** identify/fix or migrate to new infra





# NETCONF example with Ansible



ANSIBLE

# Playbook Ansible (module cli\_command)

```
- ios_config:  
  lines:  
    - ntp server 2.2.2.2
```

**What if there is already an NTP server ?**

You're lucky there is a module that handles better NTP  
on Cisco IOS: ios\_ntp\_global\_module

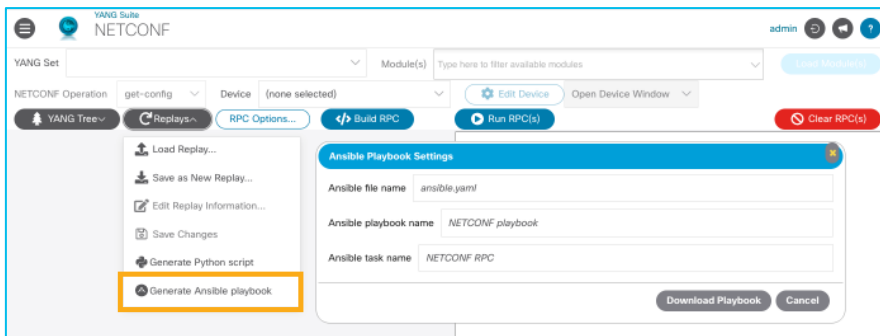
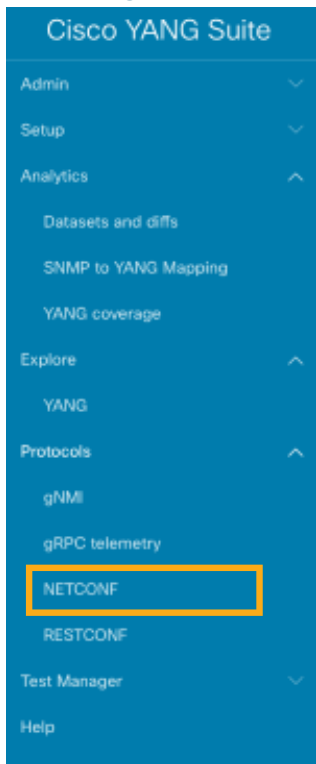
# Playbook Ansible (module netconf\_rpc)

```
- netconf_rpc:
  rpc: edit-config
  content: |
    <target>
      <running/>
    </target>
    <config>
      <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
        <ntp>
          <server xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ntp"
operation='replace'>
            <server-list>
              <ip-address>2.2.2.2</ip-address>
            </server-list>
          </server>
        </ntp>
      </native>
    </config>
```

# YANG Suite + Ansible integrations

## using NETCONF, RESTCONF & gNMI OpenConfig

Quickly and easily generate Ansible playbook for deployments to be used with the inventory, similar to the “Generate Python script” button.



```
30 Lines (29 slots) 1.06 KB
3 ---
4 - hosts: ios-xe
5 gather_facts: no
6 connection: netconf
7 remote_user: admin
8 tasks:
9   - name: establish subscription
10    netconf_config:
11      xslt |
12        <netconf xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
13          <net-conf-data xmlns="http://cisco.com/x/yang/Cisco-300-80-net-cfg">
14            <subscription-id>001</subscription-id>
15            <base>
16              <trk@yang-paths/xslt>
17                <encoding-keg@/encoding>
18                <source-address>10.50.0.10</source-address>
19                <source-url@get-url/>source-url
20            </base>
21          </net-conf-data>
22        </netconf>
23
```

RESTCONF

# RESTCONF Details

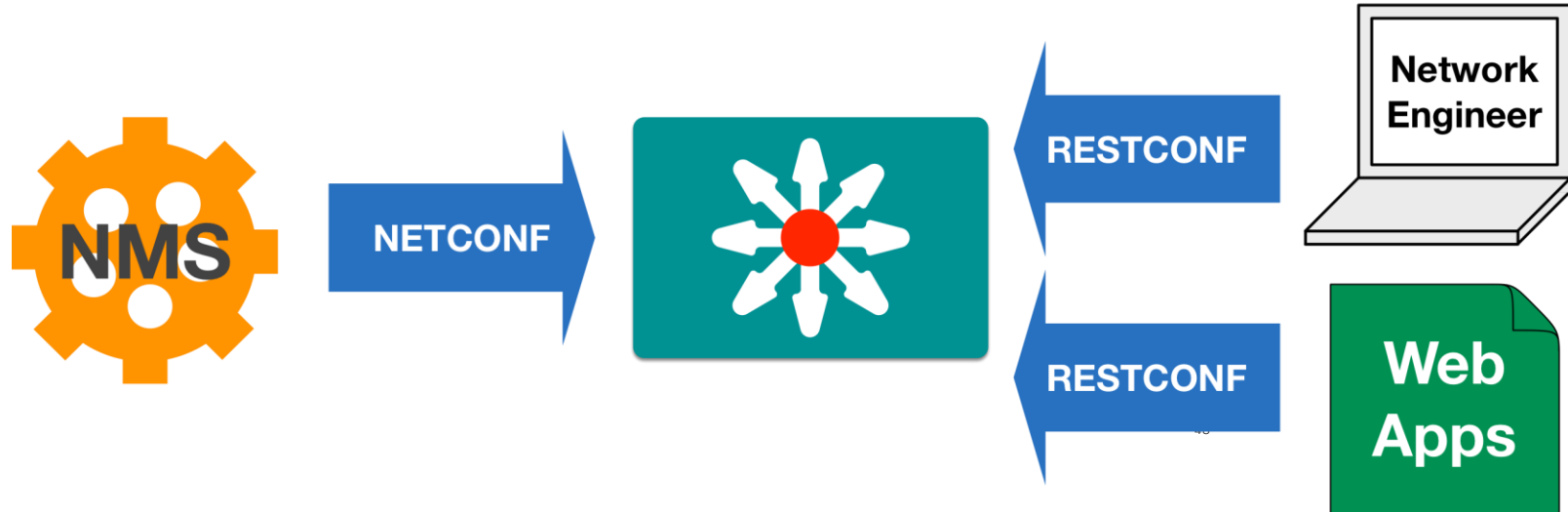
“an HTTP-based protocol that provides a programmatic interface for accessing data defined in YANG...”

<https://tools.ietf.org/html/rfc8040>

- [RFC 8040](#) - January 2017
- Uses HTTPS for transport
- Tightly coupled to the YANG data model definitions
- Provides JSON or XML data formats

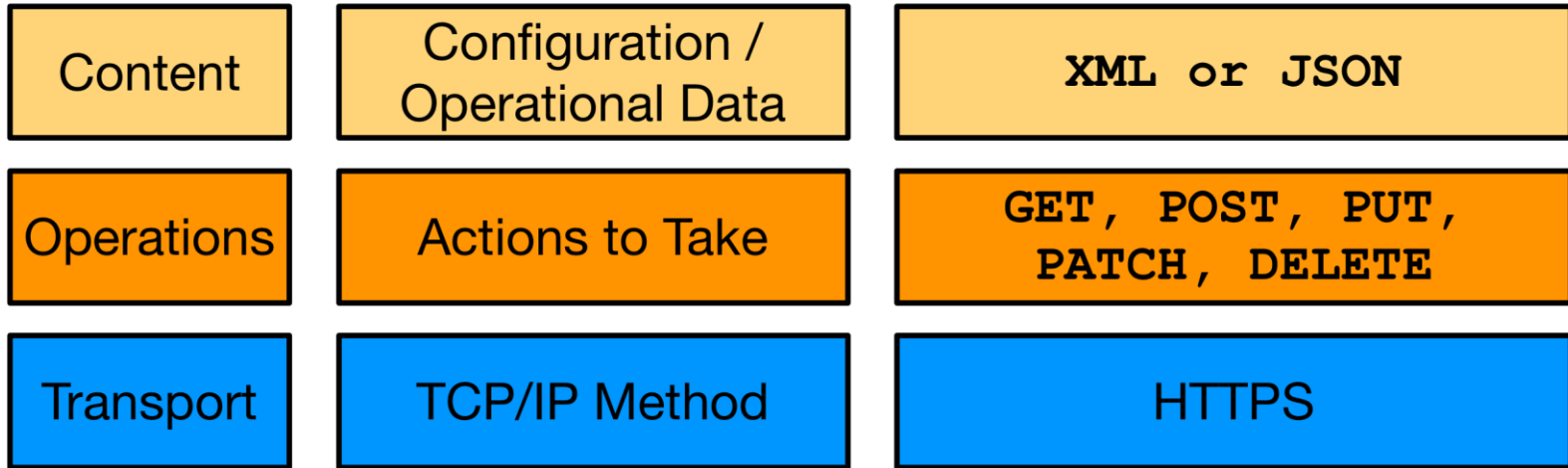
# What about NETCONF?

## Standard Network Management



# RESTCONF Protocol Stack & Transport

## RESTCONF Protocol Stack





# Content – XML or JSON

## HTTP Headers

- **Content-Type:** Specify the type of data being sent from the client
- **Accept:** Specify the type of data being requested by the client

## RESTCONF MIME Types

- `application/yang-data+json`
- `application/yang-data+xml`

# Operations - HTTP CRUD

RESTCONF	NETCONF
GET	<get> , <get-config>
POST	<edit-config> (operation="create")
PUT	<edit-config> (operation="create/replace")
PATCH	<edit-config> (operation="merge")
DELETE	<edit-config> (operation="delete")

# URL Creation Review

`https://<ADDRESS>/restconf/data/ietf-interfaces:interfaces/interface=GigabitEthernet1?depth=unbounded`

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

## Options Examples:

- `depth=unbounded`  
Follow nested models to end. Integer also supported
- `content=[all, config, nonconfig]`  
Query option controls type of data returned.
- `fields=expr`  
Limit what leaves are returned

Key:  
`https://<ADDRESS>/<ROOT>/data/<[YANG MODULE]:<CONTAINER>/<LEAF>[?<OPTIONS>]`

# Postman collections (BGP EVPN VXLAN)

A Postman collection contains examples for managing EVPN configurations

<https://www.postman.com/ciscodevnet/workspace/cisco-devnet-s-public-workspace>

The screenshot displays a Postman workspace with a collection of RESTCONF endpoints. The endpoints are organized into folders: Cisco BGP EVPN RESTCONF, IOS-XE, BGP, EVPN, and VRF. The endpoints are color-coded: green for GET, red for PATCH, and red with a slash for DELETE. The right-hand side of the image shows a detailed view of a GET endpoint for a loopback interface configuration, including its URL, headers, and a JSON response.

Folder	Method	Endpoint
Cisco BGP EVPN RESTCONF	GET	Native BGP
IOS-XE	GET	Native BGP Neighbor id
	GET	Native BGP EVPN Neig...
	GET	Native BGP VRF Unicas...
	GET	Native BGP VRF Unicas...
	PATCH	Native BGP
	PATCH	Native BGP System
	PATCH	Native BGP System - fai...
	PATCH	Native BGP Neighbor
	PATCH	Native BGP VRF Unicas...
	DEL	Native BGP Neighbor
DEL	Native BGP ID	
DEL	Native BGP EVPN Neig...	
DEL	Native BGP VRF Unicas...	
BGP	GET	L2VPN EVPN
BGP	GET	L2VPN
BGP	PATCH	L2VPN
BGP	PATCH	L2VPN EVPN
BGP	DEL	L2VPN EVPN_CONST
EVPN		
VRF		

```
GET https://postman/restconf/data/Cisco-IOS-XE-native/native/interface/Loopback
Headers (9)
Authorization: Basic cm9vdjEyEjZheSFMFTAu
Content-Type: application/yang-data+json
Accept: application/yang-data+json
Body
200 OK 1118 ms 125 KB Save Response
```

```
1
2
3
4   "name": 1,
5   "ip": {
6     "primary": {
7       "address": "192.168.28.1",
8       "address": "192.168.28.1",
9       "mask": "255.255.255.0"
10    }
11  }
12
13
14
15   "name": 12,
16   "ip": {
17     "primary": {
18       "address": "172.16.18.8",
19       "address": "172.16.18.8",
20       "mask": "255.255.0.0"

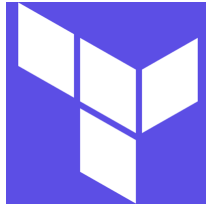
```

RESTCONF

Example with Terraform



# Terraform is...



Open-source Infrastructure as Code (IaC) Software Tool providing a consistent CLI workflow to manage hundreds of cloud services. Terraform codifies cloud APIs into declarative configuration files.

- Cloud Native Tooling circa 2014 from HashiCorp
- Agentless, single binary file
- Zero server-side dependencies

## Resources:

Ask IOS XE Terraform Provider Webex space: <https://eur1.io/#PtsT8eJFI>

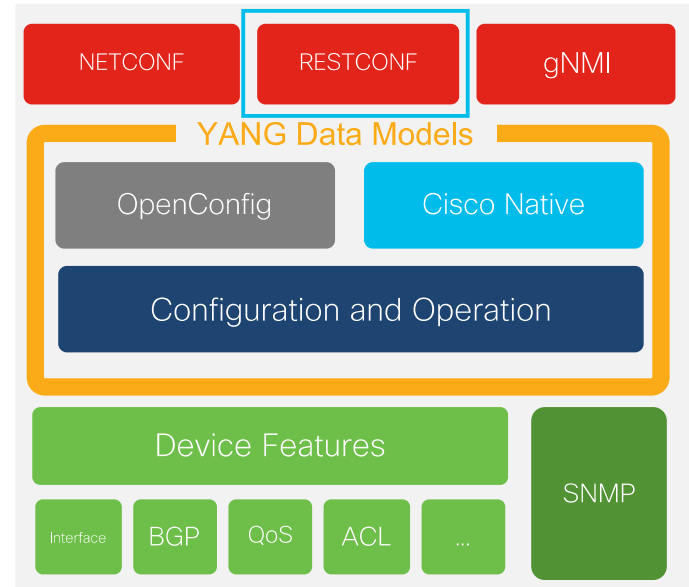
GitHub Provider Examples: <https://github.com/CiscoDevNet/terraform-provider-iosxe/>

Provider Binary: <https://registry.terraform.io/search/providers?namespace=CiscoDevNet>

Go Client: <https://github.com/CiscoDevNet/iosxe-go-client>

Blogs at <https://blogs.cisco.com/tag/terraform>

Terraform uses the RESTCONF API



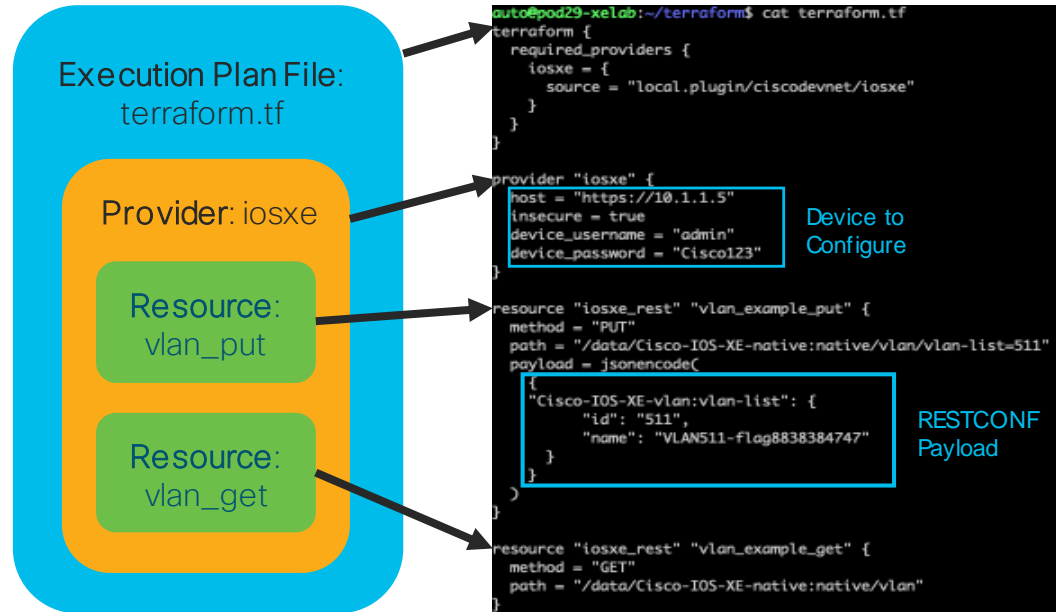
# Terraform Terminology

Terraform uses an execution plan file with a provider and resource definitions

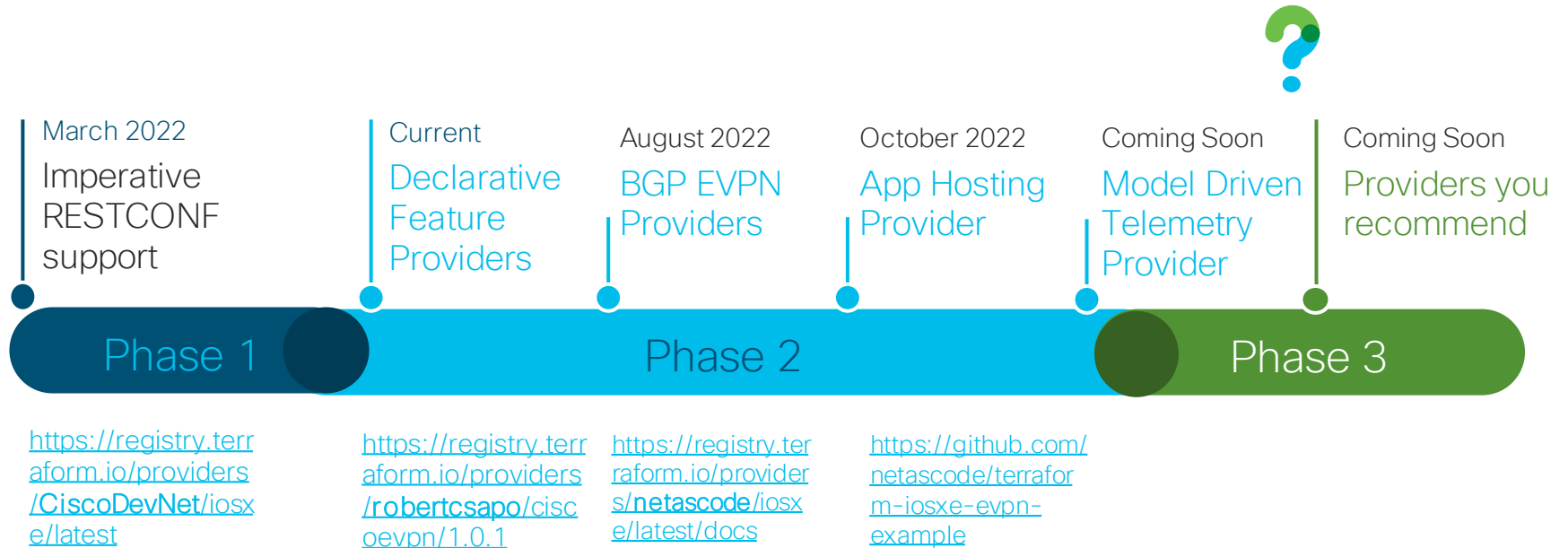
An **execution plan file** defines the provider and resources. It is written in HashiCorp Configuration Language (HCL), similar to JSON, and stored with a .tf extension

A **provider** is a plugin to make a collection of resources accessible

A **resource** (or infrastructure resource) describes one or more infrastructure objects managed by Terraform. With the IOS XE Terraform provider, resources can be considered the same as a configurable feature



# Evolution of Terraform Provider



Declarative providers leverage the SDK from the Phase 1 imperative provider



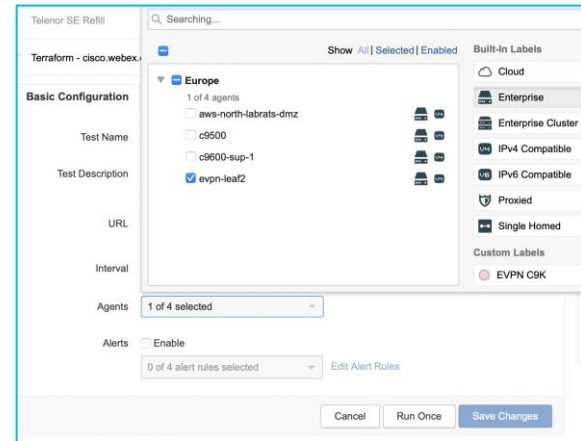
# Terraform ThousandEyes lifecycle management

1. Deploy TE agent on switch catalyst9000
2. Pass variables including the the Agent ID to the ThousandEyes API
3. Create test and attach the C9K TE Agent ID to the test
4. Trigger test to run

```
terraform {
  required_providers {
    ciscoapphosting = {
      source = "robertcsapo/ciscoapphosting"
      version = "1.0.0"
    }
  }
}

provider "ciscoapphosting" {
  username = var.username
  password = var.password
  insecure = var.insecure
  timeout = var.timeout
}

resource "ciscoapphosting_app" "app" {
  host = "127.0.0.1"
  image = "https://downloads.thousandeyes.com/enterprise-agent/thousandeyes-enterprise-agent-4.2.2.cisco.tar"
  app_gigabit_ethernet = "1/0/1"
  vlan_trunk = false
  vlan = 1
  env = {
    TEAGENT_ACCOUNT_TOKEN = "token"
  }
}
```



<https://github.com/robertcsapo/terraform-provider-ciscoapphosting>  
<https://registry.terraform.io/providers/robertcsapo/ciscoapphosting/>

GNMI

# gRPC and gNMI

## gRPC

### Google Remote Procedure Call

- gRPC's are used when interfacing with the gNMI interface
- Implemented in IOS XE for Model Driven Telemetry (MDT)
- Dial-Out / Configured Telemetry
- Low-latency and scalable
- HTTP/2 transport

## gNMI

### gRPC Network Management Interface

- Network management protocol
- Manage configuration and view operational data of network devices
- Developed by Google
- Modeled using YANG
- Programmability and Dial-In / Dynamic Telemetry (Dial-out soon)

# gNMI Operations

CAP

Capabilities exchange – list all supported YANG modules

GET

Prefix	Shortcut if Paths share root paths
Paths	Defines what info is gathered
Data Type	OPER, CONFIG, ALL

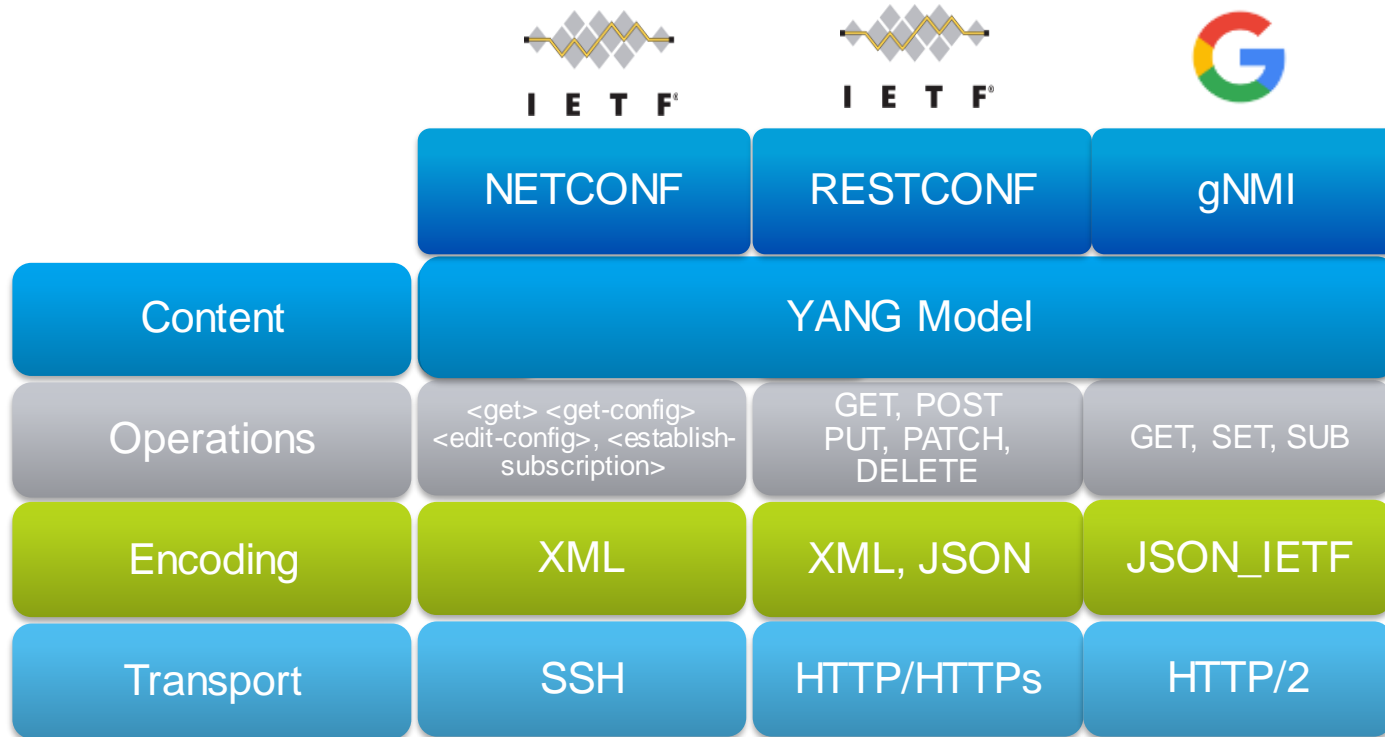
SET

Like GET	Prefix, Paths
Type	Update, Replace or Delete

SUBSCRIBE

Subscribe	Prefix, Path
-----------	--------------

# API Interfaces



# gNOI

## gRPC Network Operations Interface

1. gRPC Network Operations Interface, or gNOI, is a set of gRPC-based microservices, used for executing operational commands on network devices
2. gNOI operations are executed against the gNMI API interface
3. gNOI is defined and implemented on a per proto basis
4. There are many protos defined - some are more mature and evolve and different pace

Protobuf RPC	Use	Related CLI	Release
Cert.proto	TLS Certificate management	crypto pki ...	17.3
Os.proto	Network Operating System management	install add file ...	17.5
Reset.proto	Factory Reset and wipe	factory-reset	17.7
File.proto	Not implemented	copy, delete	N/A
System.proto	Not implemented	reload, set boot	N/A

<https://github.com/openconfig/gnoi>

Search or jump to... 7 Pulls Issue

openconfig / gnoi

<> Code ⓘ Issues 9 🔄 Pull requests 5

master

Go to file

aashaikh Updates to cert service. (#41)

- docs
- factory\_reset
- file
- interface
- layer2
- mpls
- os
- otdr
- system

## Polling Question 2

Quel API de programmation est à privilégier pour upgrader un équipement ?

- 1) CLI
- 2) RESTCONF
- 3) GNOI

Téléométrie

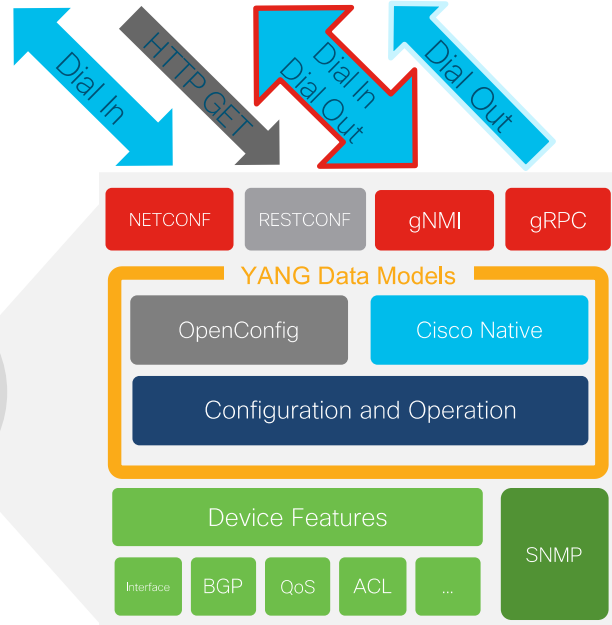


# Model Driven Telemetry Interfaces

↔ Dial In: Collector establishes a connection to the device then subscribes to telemetry (pub/sub)

↔ Out: Telemetry is pushed from the device to the collector based off configuration (push)

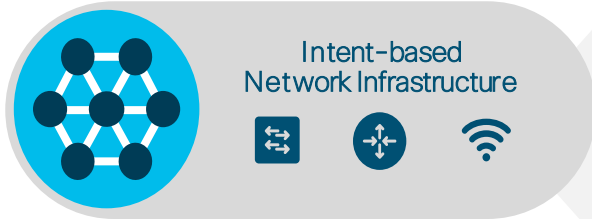
## Publication / Subscription



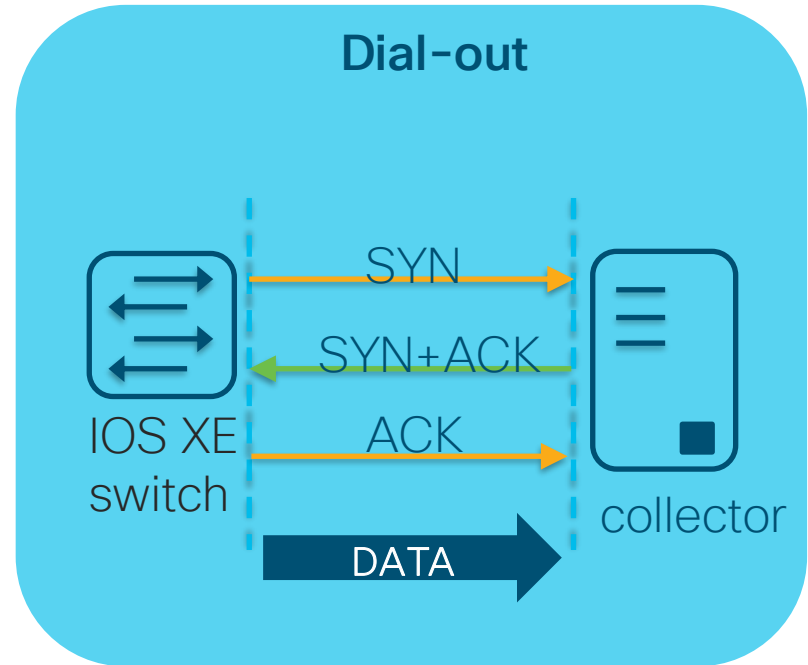
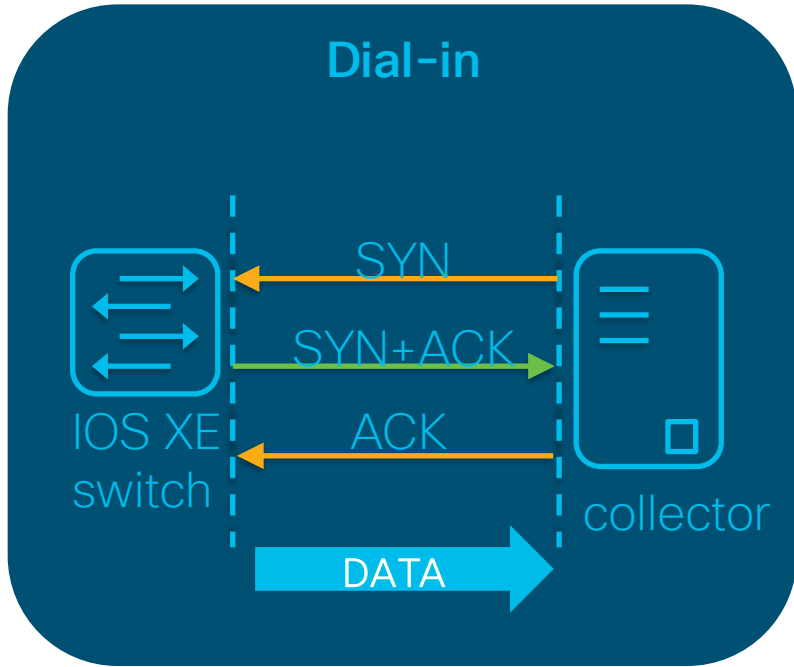
XML, JSON, proto and kvGPB encoding

Consistent YANG data models between interfaces

On-change event and time-based publication options

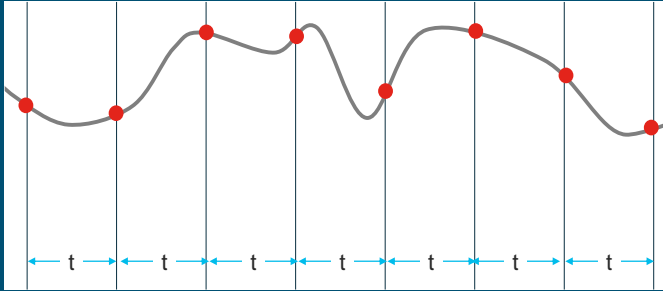


# Telemetry Dial-in vs Dial-out

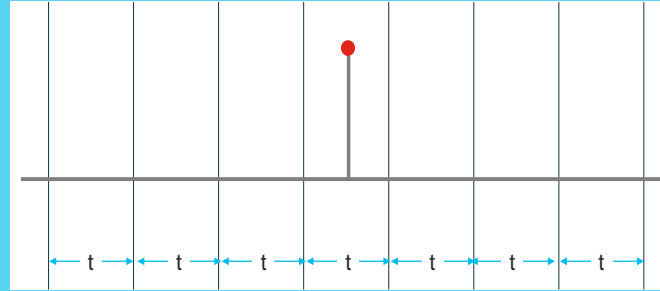


# Telemetry Periodic vs On-Change

## Periodic



## On-Change





# Model Driven Telemetry Interface Comparison

	NETCONF	gRPC Dial-Out	gNMI Dial-In	gNMI Dial-Out
Minimum IOS XE Version	16.6	16.10	16.12	17.11
Recommended Version	17.6	17.6	17.6	17.11
Telemetry Direction	Dial-In, IOS XE is server	Dial-Out, IOS XE is client	Dial-In, IOS XE is server	Dial-Out
Configuration	Dynamic per session	Static per configuration	Dynamic per session	Static
Telemetry Collector	Client	Server	Client	Server
Encoding	XML	KV GPB	JSON_IETF	PROTO + JSON_IETF
Security	SSH + PKI certificate or password	TLS or plain-text	TLS certificate with user authentication	TLS certificate with user authentication
Transport Protocol	SSH	HTTP2	HTTP2	HTTP2
Data Models	YANG	YANG	YANG	YANG

Network architecture, security posture and policy, YANG data modules, tools and language preferences are some considerations when leveraging the various MDT interfaces

# IOS XE Model Driven Telemetry



Cisco IOS XE



CLI

...or with...

YANG

gNMI Dial-In/Dynamic NETCONF Dial-In   gRPC Dial-Out/Configured



Collector/Receiver  
Decodes to text



Storage  
Time Series Database



Monitoring  
and Visualizations



# Migrating from SNMP to gRPC Dial-Out Telemetry

What is the expected increase in CPU/Memory when using the gRPC Dial-Out telemetry interface, compared to SNMP ?

gRPC adds 2% for each telemetry collector  
SNMP adds 6% and an additional 4% for each collector

Testbed:

Ubuntu Linux VM  
Telegraf Tooling  
SNMP + gRPC

C9300-48

Spirent 48 port  
traffic generator

## CPU impact with multiple gRPC receivers

1. No interface collection, CPU @ 1%
2. Add gRPC 1 receiver, CPU @ 3% +2
3. Add 2<sup>nd</sup> gRPC receiver, CPU @ 5% +2
4. Add 3<sup>rd</sup> gRPC receiver, CPU @ 8% +3

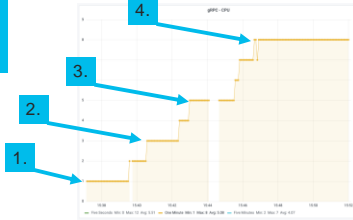
### gRPC configuration

```
telemetry leaf subscription 1000
  encoding json-vevops
  filter xpath /interface-ios-ww-oper/interfaces/interface
  source-address 128.107.223.252
  stream yang-push
  update-policy periodic 500
  receiver ip address 10.10.10.200 17500 protocol grpc-ntp

telemetry leaf subscription 1001
  encoding json-vevops
  filter xpath /interface-ios-ww-oper/interfaces/interface
  source-address 128.107.223.252
  stream yang-push
  update-policy periodic 500
  receiver ip address 10.10.10.200 17500 protocol grpc-ntp

telemetry leaf subscription 1002
  encoding json-vevops
  filter xpath /interface-ios-ww-oper/interfaces/interface
  source-address 128.107.223.252
  stream yang-push
  update-policy periodic 500
  receiver ip address 10.10.10.200 17500 protocol grpc-ntp
```

© 2020 Cisco and/or its affiliates. All rights reserved. Cisco Confidential



## CPU impact multiple SNMP pollers

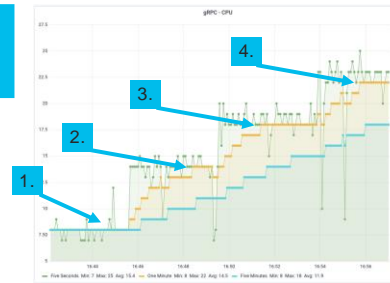
1. Baseline CPU @ 8 %
2. Add 1 SNMP CPU @ 14% +6
3. Add 2<sup>nd</sup> SNMP CPU @ 18% +4
4. Add 3<sup>rd</sup> SNMP CPU @ 22% +4

### SNMP collection configuration

```
telemetry process snmp
  snmpwalk -v 1 -c EN-TMR-Cisco123 128.107.223.252 [0x%1.1.1.1:161]

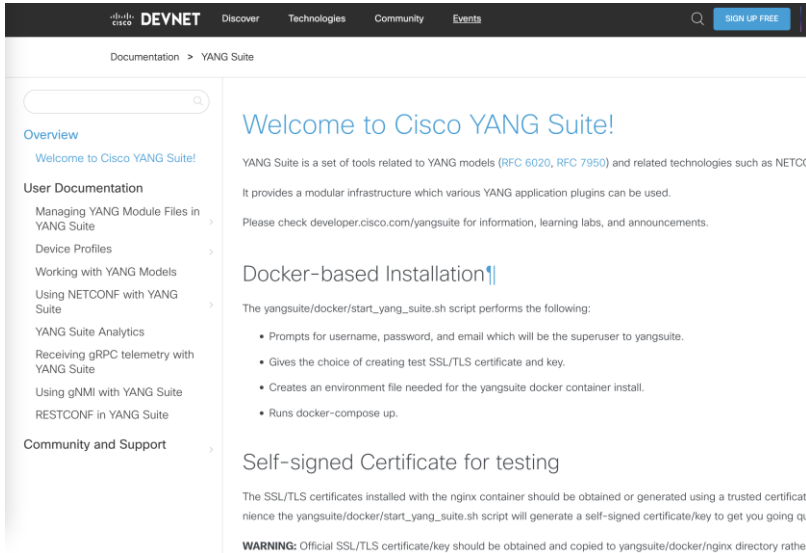
telemetry process snmp
  snmpwalk -v 1 -c EN-TMR-Cisco123 128.107.223.252 [0x%1.1.1.1:161]

telemetry process snmp
  snmpwalk -v 1 -c EN-TMR-Cisco123 128.107.223.252 [0x%1.1.1.1:161]
```



Démo Yang Suite

# Install YANG Suite



The screenshot shows the Cisco DevNet documentation page for the YANG Suite. The page has a dark header with the DevNet logo and navigation links for Discover, Technologies, Community, and Events. A search bar and a 'SIGN UP FREE' button are also present. The main content area is titled 'Welcome to Cisco YANG Suite!' and includes a search bar, a table of contents, and a 'Docker-based Installation' section. The table of contents lists sections like Overview, User Documentation, YANG Suite Analytics, and Community and Support. The 'Docker-based Installation' section describes the yangsuite/docker/start\_yang\_suite.sh script and lists its actions: prompting for user details, offering SSL/TLS certificate options, creating an environment file, and running docker-compose up. A warning at the bottom states that official SSL/TLS certificates should be used instead of self-signed ones.

Documentation > YANG Suite

Overview

- Welcome to Cisco YANG Suite!

User Documentation

- Managing YANG Module Files in YANG Suite
- Device Profiles
- Working with YANG Models
- Using NETCONF with YANG Suite

YANG Suite Analytics

- Receiving gRPC telemetry with YANG Suite
- Using gNMI with YANG Suite
- RESTCONF in YANG Suite

Community and Support

## Welcome to Cisco YANG Suite!

YANG Suite is a set of tools related to YANG models (RFC 6020, RFC 7950) and related technologies such as NETCO. It provides a modular infrastructure which various YANG application plugins can be used. Please check [developer.cisco.com/yangsuite](https://developer.cisco.com/yangsuite) for information, learning labs, and announcements.

## Docker-based Installation¶

The `yangsuite/docker/start_yang_suite.sh` script performs the following:

- Prompts for username, password, and email which will be the superuser to yangsuite.
- Gives the choice of creating test SSL/TLS certificate and key.
- Creates an environment file needed for the yangsuite docker container install.
- Runs `docker-compose up`.

## Self-signed Certificate for testing

The SSL/TLS certificates installed with the `nginx` container should be obtained or generated using a trusted certificate hence the `yangsuite/docker/start_yang_suite.sh` script will generate a self-signed certificate/key to get you going quickly.

**WARNING:** Official SSL/TLS certificate/key should be obtained and copied to `yangsuite/docker/nginx` directory rather

```
git clone
https://github.com/CiscoDevNet/yangsuite

yangsuite/docker/start_yang_suite.sh
```

<https://developer.cisco.com/docs/yangsuite/>



## Polling Question 3

Pour quelle raison  
n'installeriez-vous pas  
Yang Suite après ce  
Webinar ?

- 1) Cela prend trop de temps à installer
- 2) C'est trop cher
- 3) Aucune de ces 2 réponses

Pour poursuivre vos  
recherches...

# IOS XE Operational Consistency

	Switching										Wireless				Routing					IOT				SPAG		Cable							
	CAT 3850/3850	CAT 9200CX	CAT 920DL	CAT 9200	CAT 9300L	CAT 9300/9400	CAT 9500	CAT 9500H	CAT 9600	CAT 9100-EWC	CAT 9800-CL	CAT 9800-L	CAT 9800-40/80	ISR 1000	ISR 4000	C8300 C8200	CSR 1000v	C8KV	ASR 1000 Fixed	ASR 1000 Modular	C8500 / C8500L	IR 1100	E9R 6300	IE 3x00	E5S 3300	ASR 900 / 920	NCS 520	NCS 4200	cBR-8				
Provisioning Automation																																	
ZTP	16.5+	17.8		16.12+	16.12+	16.6+	16.8+	16.12+																									
PXE	16.5+	17.8		16.9+	16.9+	16.6+	16.8+	16.11+																									
Model Driven Configuration Management																																	
NETCONF	16.5+	17.8	16.9+	16.9+	16.9+	16.6+	16.8+	16.11+	16.12+	16.10+	16.12+	16.10+	16.8+	16.3+	17.3+	16.3+	17.4+	16.3+	16.3+	17.3+	16.10+	17.1+	16.11+	16.11+	16.8+	16.10+	16.8+	16.8+	16.8+	16.8+	16.8+		
RESTCONF	16.7+	17.8	16.9+	16.9+	16.9+	16.7+	16.8+	16.11+	16.12+	16.11+	16.12+	16.11+	16.8+	16.6+	17.3+	16.6+	17.4+	16.6+	16.6+	17.3+	16.10+	17.1+	16.11+	16.11+	16.8+	16.10+	16.8+	16.8+	16.8+	16.8+	16.8+		
gNMI		17.8	16.12+	16.12+	16.12+	16.8+	16.10+	16.11+		17.8+	17.8+	17.8+	17.8+	17.8+	17.8+		17.12+	17.2	17.2	17.3+	17.2+	17.2+	16.12+	16.12+	17.1	17.1	17.1	16.12+	16.12+	16.12+			
Model Driven Telemetry																																	
NETCONF Dial-In	16.6+	17.8	16.9+	16.9+	16.9+	16.6+	16.8+	16.11+	16.12+ ***				16.8+	16.7+	17.3+	16.10+	17.4+	16.7+	16.8+	17.3+	16.10+	17.1+	16.12+	16.12+	16.9+	16.10+	16.9+	16.9+	16.9+	16.9+			
gRPC Dial-Out		17.8	16.10+	16.10+	16.10+	16.10+	16.10+	16.11+	N/A	17.6 ***	17.6 ***		16.10+	16.10+	17.3+	16.10+	17.4+	16.10+	16.10+	17.3+	16.10+	17.1+	16.12+	16.12+	16.10+	16.10+	16.10+	16.10+	16.10+	16.10+	16.10+		
gNMI Dial-In		17.8	16.12+	16.12+	16.12+	16.12+	16.12+	16.12+	N/A	17.6 ***	17.6 ***		17.8+	17.8+	17.8+		17.12+	17.2+	17.2+	17.3+	17.2+	17.2+	17.1+	17.1+	17.1+	17.1+	17.1+	17.1+	17.1+	16.12+	16.12+		
gNMI Dial-out		17.11+	17.11+	17.11+	17.11+	17.11+	17.11+	17.11+																							17.11+	17.11+	
gNMI explicit wildcard		17.8	17.5+	17.5+	17.5+	17.5+	17.5+	17.5+																									
Software Image Management																																	
GuestShell (On Box Python)	16.5+ *	17.8		16.12+	16.12+	16.6+	16.8+	16.12+			17.8+	16.11+	16.9+ **	16.5+	17.3+	16.7+	17.4+	16.7+	16.8+	17.3+					17.1+	17.1+							
gRPC Network Operations Interface (gNOI)																																	
cert.proto		17.8	17.3+	17.3+	17.3+	17.3+	17.3+	17.3+		17.8+	17.8+	17.8+	17.8+	17.8+	17.8+	17.8+		17.12+	17.8+	17.8+	17.8	17.8+	17.8+	17.8+	17.8+	17.8+	17.8+	17.8+	17.8+	17.8+	17.8+	17.8+	
os.proto		?		17.5+	17.5+	17.5+	17.5+																										
reset.proto		?	17.7+	17.7+	17.7+	17.7+	17.7+	17.7+				17.7+													17.8+	17.8+	17.8+	17.8+					

Not Planned | Planned | Shipping

Source of truth: [https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/179/b\\_179\\_programmability\\_cg.html](https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/179/b_179_programmability_cg.html)



Start Now



Videos and  
Tutorials



Sandbox  
Learning Lab



Automation and  
Code Exchange



Learning and  
Certifications



Community and  
Study Groups

[developer.cisco.com](https://developer.cisco.com)

# Cisco IOS XE Programmability – Booksprint Book

<http://cs.co/programmabilitybook> OR <https://www.cisco.com/c/dam/en/us/products/collateral/enterprise-networks/nb-06-ios-xe-prog-ebook-cte-en.pdf>

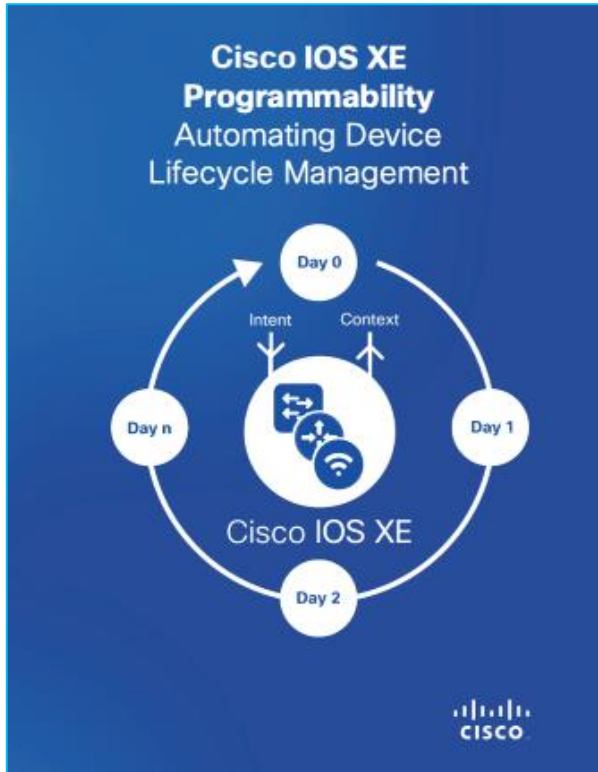


Table of Contents	
Authors	Telemetry
Acknowledgments	Overview
About this Book	Operational Data
Introduction	Flow Data
Why Programmability Matters	Use Cases
Lifecycle of Network Device Operations	Subscription Tools
Use Cases	Data Collectors
Operational Approaches	Python
Next Steps	Overview
General Concepts	Python WebUI Sandbox
Cisco IOS XE	On-Box Python
What is Programmability?	Advanced On-Box Python
Application Programming Interfaces (APIs)	Common Issues
Programming Languages	Guest Shell
Structured Data	Introduction
Data Encoding Formats	Security
Day 0 Device Onboarding	Configuration and Updates
Introduction	Resource Allocation
Zero-Touch Provisioning (ZTP) Scenarios	Use Cases
Basic ZTP Workow	Next Steps
Advanced ZTP Workows	Application Hosting
Considerations	Introduction
Next Steps	Cisco Application-Hosting Framework
YANG	Containers and Virtual Machines
Overview	Use Case
YANG Concepts	Next Steps
YANG Native vs Open Data Models	Controllers
YANG Data Model Highlights	Introduction
YANG Tools	Common Controllers
Network Device APIs	Why Use a Controller?
Overview	DevOps and NetDevOps
NETCONF	Introduction
RESTCONF	Continuous Integration and Delivery
Comparison of NETCONF and	DevOps Tools
RESTCONF	Next Steps
Next Steps	Appendices
	Additional Resources
	Acronyms

# Programmability Configuration Guide

## Book Table of Contents

- Preface
- New and Changed Information
- ▼ Provisioning
  - Zero-Touch Provisioning
  - IPXE
- ▼ Shells and Scripting
  - Guest Shell
  - Python API
  - EEM Python Module
- ▼ Model-Driven Programmability
  - NETCONF Protocol
  - RESTCONF Protocol
  - NETCONF and RESTCONF Service-Level ACLs
  - gNMI Protocol
  - gRPC Network Operations Interface
  - Model Based AAA
  - Model-Driven Telemetry
  - In-Service Model Update
- ▼ Application Hosting
  - Application Hosting
- ▼ OpenFlow
  - OpenFlow
  - High Availability in OpenFlow Mode



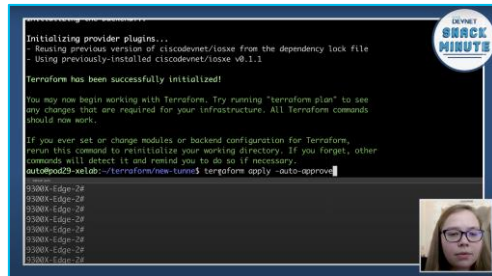
## **Programmability Configuration Guide, Cisco IOS XE Cupertino 17.9.x**

**First Published:** 2022-08-01

[https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/179/b\\_179\\_programmability\\_cg.html](https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/179/b_179_programmability_cg.html)

# Blog and Resources: Terraform

<https://github.com/CiscoDevNet/terraform-provider-iosxe/>  
<https://registry.terraform.io/search/providers?namespace=CiscoDevNet>



```
Initializing provider plugins...
  - Reusing previous version of cisco/devnet:iosxe from the dependency lock file
  - Using previously-installed cisco/devnet:iosxe v0.1.1

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

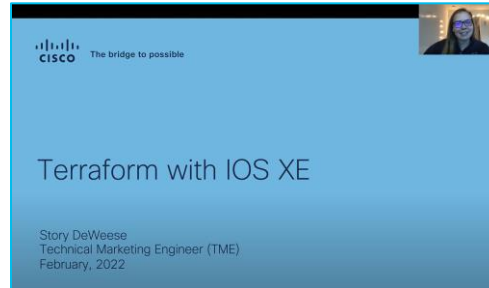
If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
aut@pod29-xelab:~/terraform/new-tunnel$ terraform apply -auto-approve
```

3300X-Edge-2F  
3300X-Edge-2F  
3300X-Edge-2F  
3300X-Edge-2F  
3300X-Edge-2F  
3300X-Edge-2F  
3300X-Edge-2F

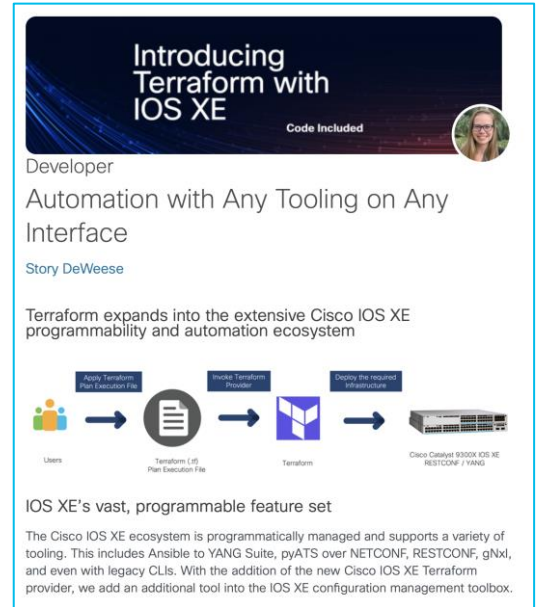
DEVNET  
SHACK  
MINUTE

Demo Create a Crypto Tunnel Video:

<https://www.youtube.com/watch?v=bPS0bhPacDw>




Intro to IOS XE Terraform Provider Video:  
[https://www.youtube.com/watch?v=GEY\\_hyXimbA](https://www.youtube.com/watch?v=GEY_hyXimbA)



**Introducing Terraform with IOS XE**  
Code Included

Developer  
Automation with Any Tooling on Any Interface  
Story DeWeese

Terraform expands into the extensive Cisco IOS XE programmability and automation ecosystem



```
graph LR
    Users[Users] --> TerraformCLI[Terraform CLI  
Plan Execution File]
    TerraformCLI --> Terraform[Terraform]
    Terraform --> IOSXE[Cisco Catalyst 9300X IOS XE  
RESTCONF / YANG]
```

IOS XE's vast, programmable feature set

The Cisco IOS XE ecosystem is programmatically managed and supports a variety of tooling. This includes Ansible to YANG Suite, pyATS over NETCONF, RESTCONF, gNxi, and even with legacy CLIs. With the addition of the new Cisco IOS XE Terraform provider, we add an additional tool into the IOS XE configuration management toolbox.

<https://blogs.cisco.com/developer/terraformiosxe01>

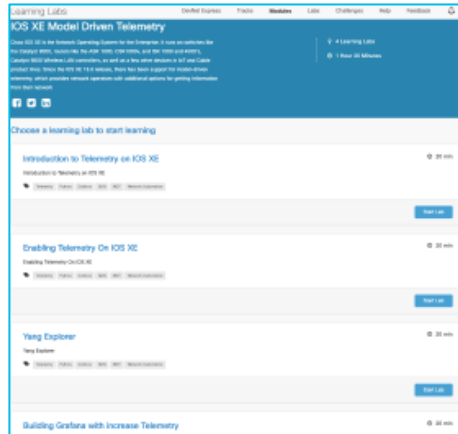
# Learning Lab and Blog: Telemetry

[https://developer.cisco.com/learning/modules/iosxe\\_telemetry](https://developer.cisco.com/learning/modules/iosxe_telemetry)

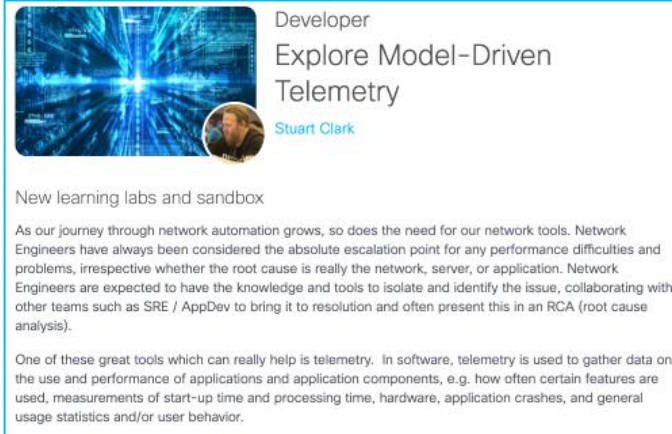
<https://blogs.cisco.com/developer/model-driven-telemetry-sandbox>

<https://blogs.cisco.com/developer/getting-started-with-model-driven-telemetry>


<https://youtu.be/QwwZakkWBng>



The screenshot shows the Cisco Learning Labs interface. At the top, it says 'IOS XE Model Driven Telemetry' with a '4 Learning Lab' badge. Below this, there's a section 'Choose a learning lab to start learning' with a list of labs: 'Introduction to Telemetry on IOS XE' (20 min), 'Enabling Telemetry on IOS XE' (20 min), 'Using Explorer' (20 min), and 'Building Grafana with Ingested Telemetry' (20 min). Each lab has a 'Start Lab' button.



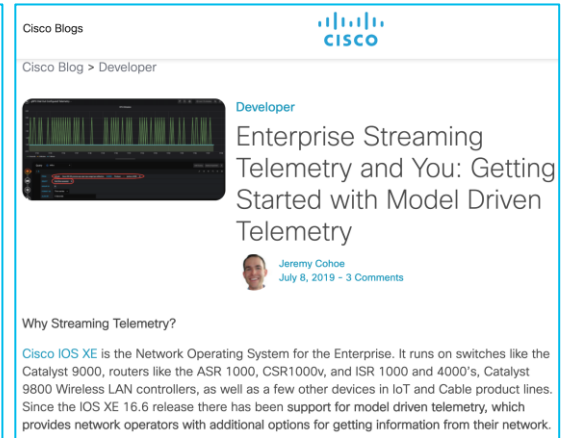
**Developer**  
**Explore Model-Driven Telemetry**  
Stuart Clark



**New learning labs and sandbox**


As our journey through network automation grows, so does the need for our network tools. Network Engineers have always been considered the absolute escalation point for any performance difficulties and problems, irrespective whether the root cause is really the network, server, or application. Network Engineers are expected to have the knowledge and tools to isolate and identify the issue, collaborating with other teams such as SRE / AppDev to bring it to resolution and often present this in an RCA (root cause analysis).

One of these great tools which can really help is telemetry. In software, telemetry is used to gather data on the use and performance of applications and application components, e.g. how often certain features are used, measurements of start-up time and processing time, hardware, application crashes, and general usage statistics and/or user behavior.



**Cisco Blogs**

Cisco Blog > Developer



**Developer**  
**Enterprise Streaming Telemetry and You: Getting Started with Model Driven Telemetry**  
Jeremy Cohoe  
July 8, 2019 - 3 Comments

**Why Streaming Telemetry?**

Cisco IOS XE is the Network Operating System for the Enterprise. It runs on switches like the Catalyst 9000, routers like the ASR 1000, CSR1000v, and ISR 1000 and 4000's, Catalyst 9800 Wireless LAN controllers, as well as a few other devices in IoT and Cable product lines. Since the IOS XE 16.6 release there has been support for model driven telemetry, which provides network operators with additional options for getting information from their network.





[reseauxblog.cisco.fr](http://reseauxblog.cisco.fr)

# Avez-vous encore des questions ?

Nos experts vous répondent

Si vous avez posé une question sur le panneau de Q&R (Q&A en anglais) ou que vous revenez sur la communauté dans les jours qui suivent notre webinaire, nos experts peuvent encore vous aider !

Participez dans le forum de Ask Me Anything (AMA) avant le 16 décembre 2022.

<https://bit.ly/AMA-dec22>



# Nos réseaux sociaux



LinkedIn

[Cisco Community](#)

Twitter

[@cisco\\_support](#)

YouTube

[CiscoSupportChannel](#)

Facebook

[CiscoSupportCommunity](#)





Faites valoir votre opinion  
en répondant à notre enquête !

---

Cliquez sur le lien

<https://bit.ly/WEBenq-dec22>



The bridge to possible