Modern approaches for IOS XE network device management on Cat9K

with Cisco IOS XE

Jeremy Cohoe
Technical Marketing Engineer
@jeremycohoe

Story DeWeese
Technical Marketing Engineer
@storydeweese

Ashil Parekh
Product Management
@Ashil34571977

DEVNET-1110

Who We Are

Story DeWeese

Technical Marketing Engineer

Jeremy Cohoe

Technical Marketing Engineer

Ashil Parekh

Product Management

Who you are:

1. Public sector
2. Financial
3. Education
4. Defense
5. MSP/Self
6. Others

We see “web scalers” as leaders in Programmability & Automation adoption

Closely followed by: Financial, Education, Defense, Government, MSP… etc

Agenda

11:30 – 11:50: Introduction / YANG & API / Tooling

10 mins Ashil – Why we are here

10 mins - Jeremy - Intro to the API's & Tooling

11:50 – 12:05: Device Onboarding (SZTP)

5 mins – Ashil

10 mins – Story

12:05 - 12:35: Configuration Management

10 mins - Ashil

20 mins - Jeremy

12:35 - 12:55: Monitoring and Reporting

5 mins - Ashil

15 mins – Story – Telemetry & Sustainability

12:55 – 13:05: Optimization and Troubleshooting

10 mins – Story Livetools

13:05 - 13:15: Conclusion, Resources and Questions

10 mins – Story

01 Intro to Programmability

02 YANG, API, Tooling

03 Device Onboarding

04 Device Configuration

05 Device Monitoring

06 Device Optimizations

07 Resources, Closing & Q&A

Cisco Webex App

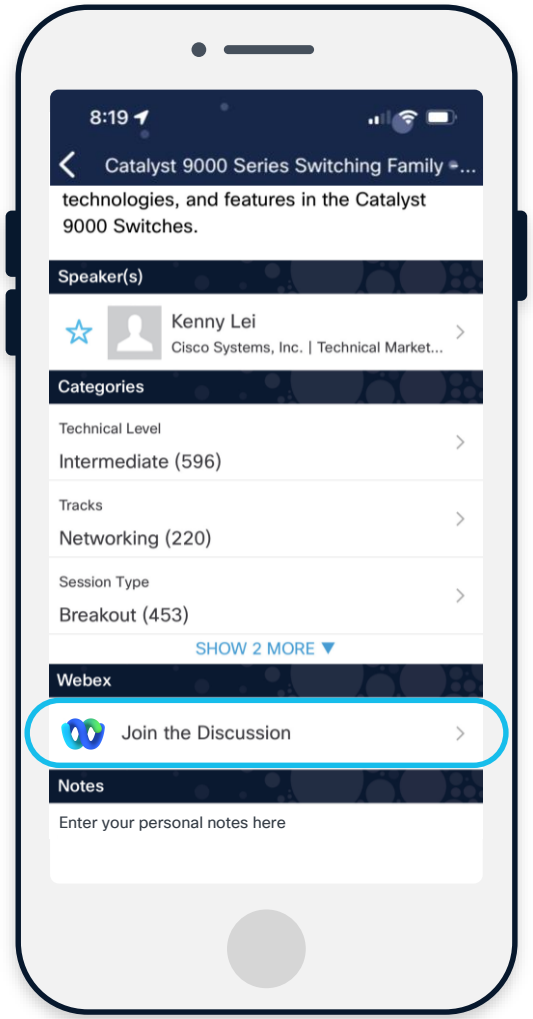
- **Please ask any questions or share any comments you have within the Webex Space**
- **We may have time for Q&A towards the end of the session**
- **We are happy to discuss with you 1:1 after the seminar is over as well**

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



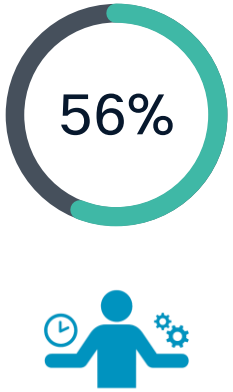
Webex spaces will be moderated by the speaker until June 13, 2025.

Intro to Programmability

AI Generated Images
Prompt: **Happy network engineer vs sad network engineer - same person**

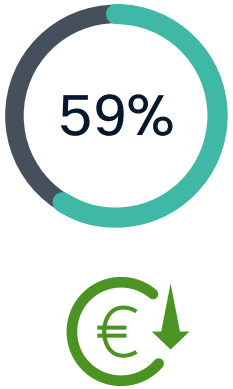
- Fewer Sites
- Less Endpoints
- Simple Network Architecture

From Networks to Business Impact: The Automation Advantages



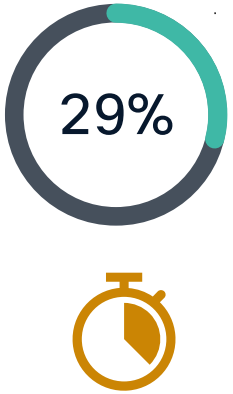
Improved Operational Efficiency

More than 80% organizations were able to achieve process standardization, reduce Human errors and improve Mean Time to Resolution(MTTR) - [1]



Operating Cost Reduction

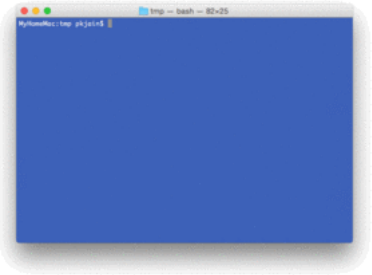
More than 80% organizations were able reduce network outages and improve security and compliance - [1]



Faster Service Deployment

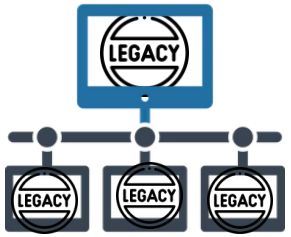
95% Respondents were able to improve the expedite the service deployment time and align with their business transformation goals - [1]

State of Network Automation Adoption



Manual Network Management

65% activities performed manually which causes 80% of outage and 22% of data breach[1]



Legacy Network Infrastructure

46% IT Leaders say that Legacy systems are difficult to automate [2]

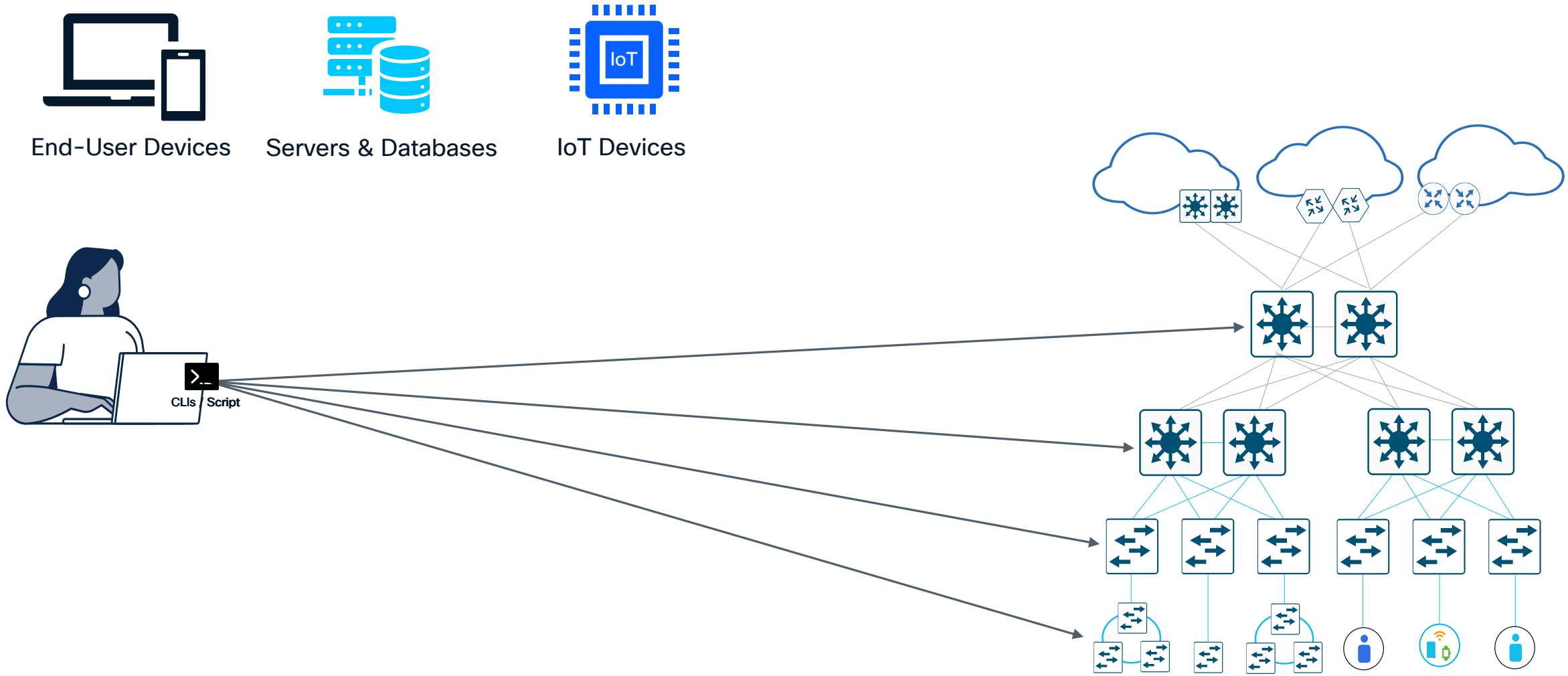
Lack of Skills

42% of IT Leaders claim their organizations lack network automation skills [2]

[1] - Enterprise Network Automation, Why do you need it? – Deloitte

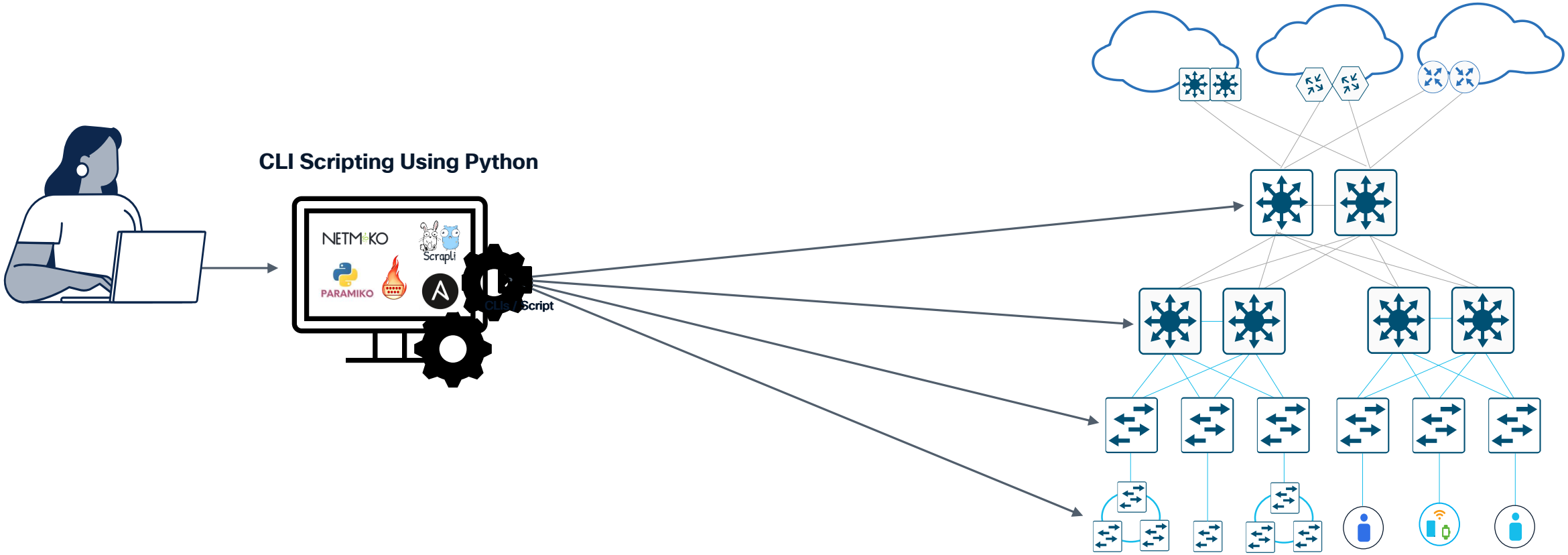
[2] – Global Network Automation Report - IDC

If It Feels Repetitive, It’s Not Automation



The First Generation of Network Automation:

Scripts and Struggles



Structured Data: The Fuel for Reliable Automation

Feature / Capability	Structured Data (YANG, XML/JSON)	Unstructured Data (CLI)
Designed For	Machine-to-machine communication	Human-to-machine interaction
Data Format	Structured (XML/JSON following a schema)	Free-form text, tables, headings
Parsing & Automation	Easy to parse programmatically	Requires fragile screen-scraping
Validation	Schema-based (YANG) validation before commit	No built-in validation; errors caught only at runtime
Error Handling	Structured errors (e.g., <rpc-error> with tags and messages)	Textual errors, inconsistent messages across vendors
Transactions	Supports atomic commits (all-or-nothing changes)	Commands executed line-by-line; no rollback capability
Consistency Across Vendors	Standard models (OpenConfig, IETF) enable multi-vendor support	Vendor-specific CLI syntax and outputs
Change Safety	Supports rollback, confirmed commits	No native rollback; must script it manually
Monitoring Integration	Works with model-driven telemetry (push-based)	Relies on CLI polling or SNMP
Scalability	Efficient for managing 100s or 1000s of devices at once	Slow, sequential; complex to scale manually
Extensibility	Models can evolve with versioning and modularization	Hard to adapt; even minor format changes break scripts
Feedback Loop	Clean input/output model for closed-loop automation	No structured input/output contract

Structured Data: The Fuel for Reliable Automation

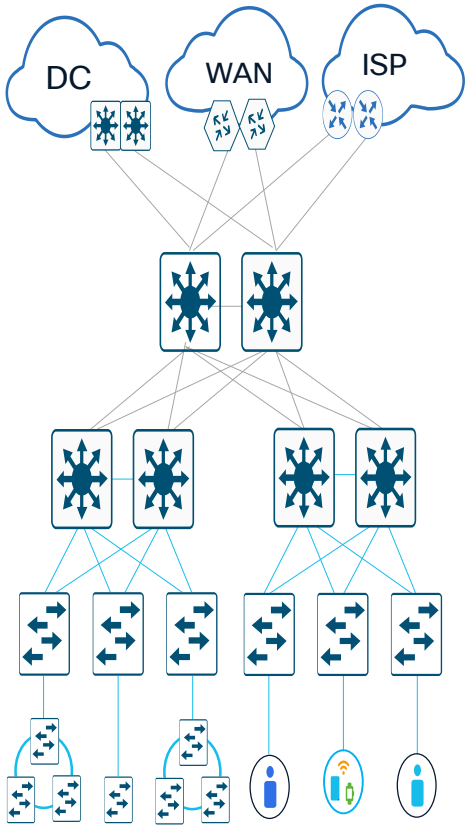
Unstructured Data (CLI)		Structured Data (YANG, XML/JSON)
Human-Oriented, Unstructured Text	Data Format & Parsing	Machine-Oriented, Structured Data
Low-Level, Procedural Logic, No intent abstraction, just commands	Abstraction & Intent	Intent-Based and Declarative, Describe what you want (intent)
Siloed and Manual, Poor integration with ITSM, CI/CD, observability tools	IT Ecosystems Integration	API-Driven and Event-Aware, Easily integrates with ITSM, GitOps, telemetry pipelines
Syntax differs by vendor, OS, and version, Scripts are bloated with conditionals	Multi-Vendor Complexity	Standard YANG models across vendors, Unified logic for multi-vendor networks

What’s the Difference?

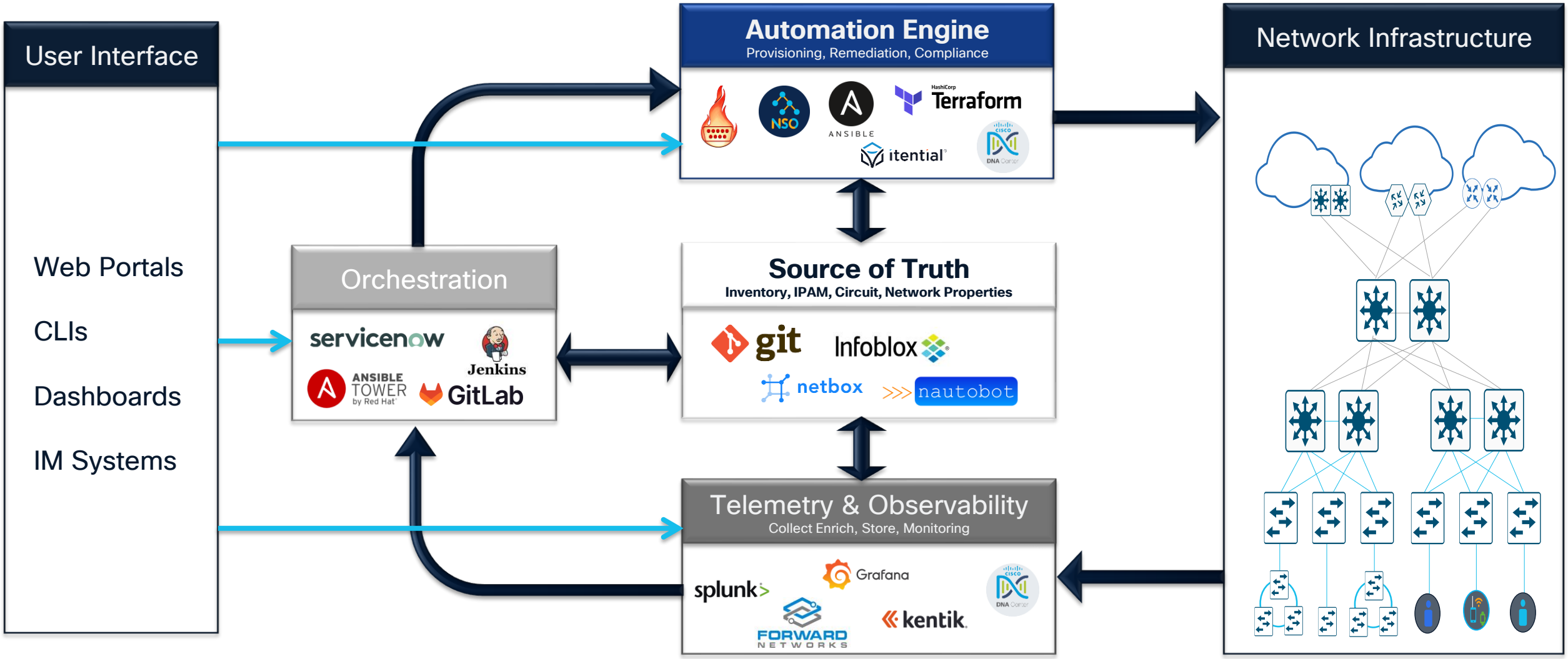
Tools

Protocol Stack

	Open Source or Infrastructure Automation Tools	Mang Protocol or Operation	Data Models	Resources	Data Modeling Language	Encoding	Transport
Using Unstructured Data	NETMIKO, Scrapli, PARAMIKO, Fabric, ANSIBLE	Vendor Specific	-	CLI	-	-	SSH
Using Structured Data	Cisco Yang Suite, HashiCorp Terraform, CHEF, ANSIBLE	NETCONF RESTCON gNMI gRPC	Vendor IETF OpenConfig	xPaths	YANG	XML JSON PROTOBUF	SSH HTTPS TLS

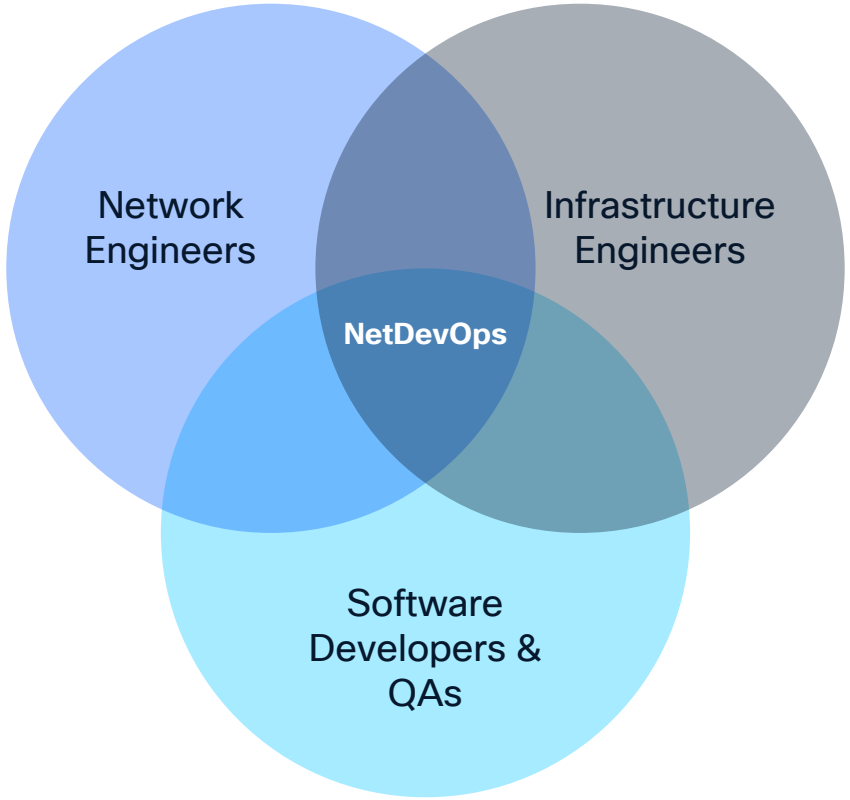


Modern Network Automation Architecture



From CLI to CI/CD: Enter NetDevOps

You can’t fully realize the benefits of network automation unless you embrace NetDevOps.



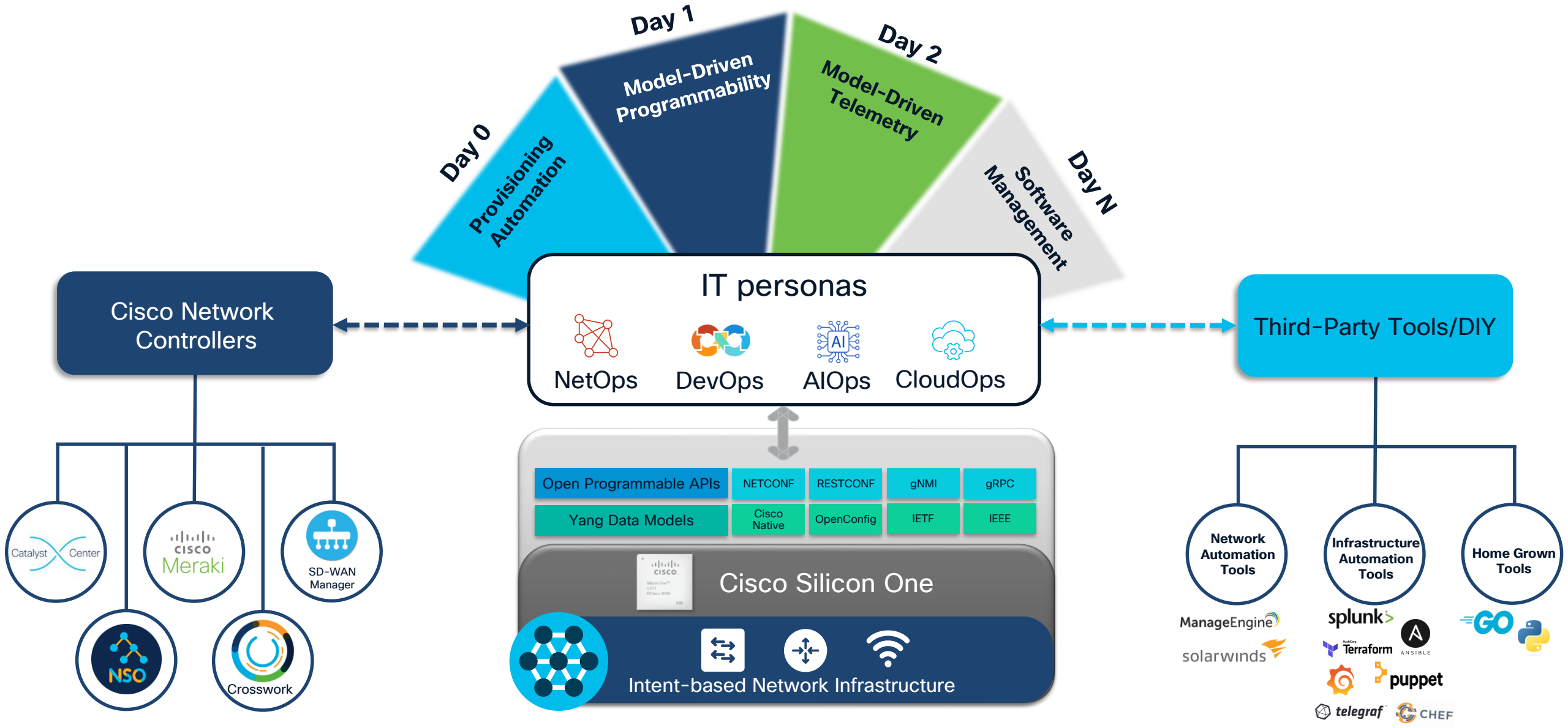
Infrastructure as Code

Version Control

CI/CD Pipelines

Industry's most open network OS

Receptive to a wide range of management and orchestration strategies



Automate Enterprise Networks With Cisco IOS-XE Programmability

Provision. Configure. Monitor. Optimize.

Day 0

Provisioning Automation

Bring up network devices into a functional state with minimal to no-touch

Network Plug-N-Play
Secure Zero Touch Provisioning (ZTP/SZTP)
Preboot eXectuion Environment (PXE)

Day 1

Model-Driven Programmability

Best in class open Programmability to empower your NetOps and DevOps

NETCONF, RESTCONF and gNMI
YANG Data models - Cisco Native, IETF, OpenConfig, IEEE

Software Image management

Manage OS, certificates and third-party Linux applications

gNOI
Guest Shell and application hosting

Day N

Model-Driven Telemetry

Real-time access to operational statistics

NETCONF Dial-In gNMI Dial-In & Dial-out
gRPC Dial-out
On-Change and periodic telemetry

Day 2

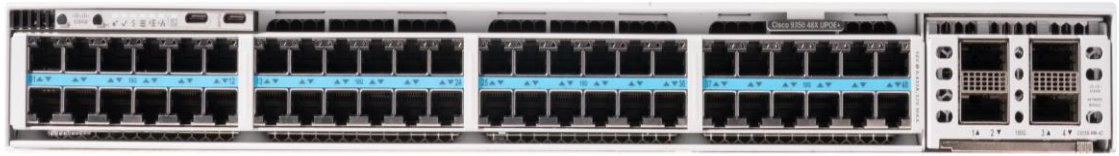
YANG

Lets talk... about data models

About Data Models

YANG = Yet Another Next Generation

"A Data-Model Explicitly and precisely defines Data Structure, Syntax and Semantics"



The YANG modules are included inside of Cisco IOS XE

They can be 'exported' from the software

They can be browsed using online/offline tools

Interface Model **definition**

ietf-interfaces@2014-05-08.yang

```
/*
 * Configuration data nodes
 */
container interfaces {
  description
    "Interface configuration parameters.";
  list interface {
    key "name";
    description
   leaf name {
      type string;
    }
    leaf description {
      type string;
    }
    leaf type {
      type identityref {
        base interface-type;
      }
      mandatory true;
    }
    leaf enabled {
      type boolean;
      default "true";
```

Feature Configuration via CLI or YANG

via CLI

```
telemetry ietf subscription 101
    encoding encode-kvgpb
    filter xpath /memory-ios-xe-oper:memory-
    statistics/memory-statistic
    stream yang-push
    update-policy periodic 6000
    source-vrf Mgmt-intf
    receiver ip address 10.10.1.45 575555
    protocol grpc-tcp
```

Human-Oriented Interface

via YANG Data Model

```
"mdt-config-data": {
    "mdt-subscription":[ {
        "subscription-id": "101",
        "base": {
            "stream": "yang-push",
        "encoding": "encode-kvgpb",
            "period": "6000",
            "xpath": "/memory-ios-xe-oper:memory-
                statistics/memory-statistic"
            }
        "mdt-receivers": {
            "address": ”10.10.1.45"
            "port": "57555" }
        }
    ]
}
```

Machine-Oriented Interface

YANG Models Example

YANG Models

ietf-interfaces@2014-05-08.yang

```
/*
 * Configuration data nodes
 */
container interfaces {
  description
    "Interface configuration parameters.";
  list interface {
    key "name";
    description
   leaf name {
      type string;
    }
    leaf description {
      type string;
    }
    leaf type {
      type identityref {
        base interface-type;
      }
      mandatory true;
    }
    leaf enabled {
      type boolean;
      default "true";
```

Data

Gig 1/0/1
“CL rocks!”
enabled

XML Payload

interface.xml — MyPythonScripts

```
<config>
  <interfaces
xmlns="urn:ietf:params:xml:ns:yang:ietf-int
erfaces">
    <interface>
      <name>GigabitEthernet0/0</name>
      <description>CL Rocks!</description>
      <enabled>true</enabled>
    </interface>
  </interfaces>
</config>
```

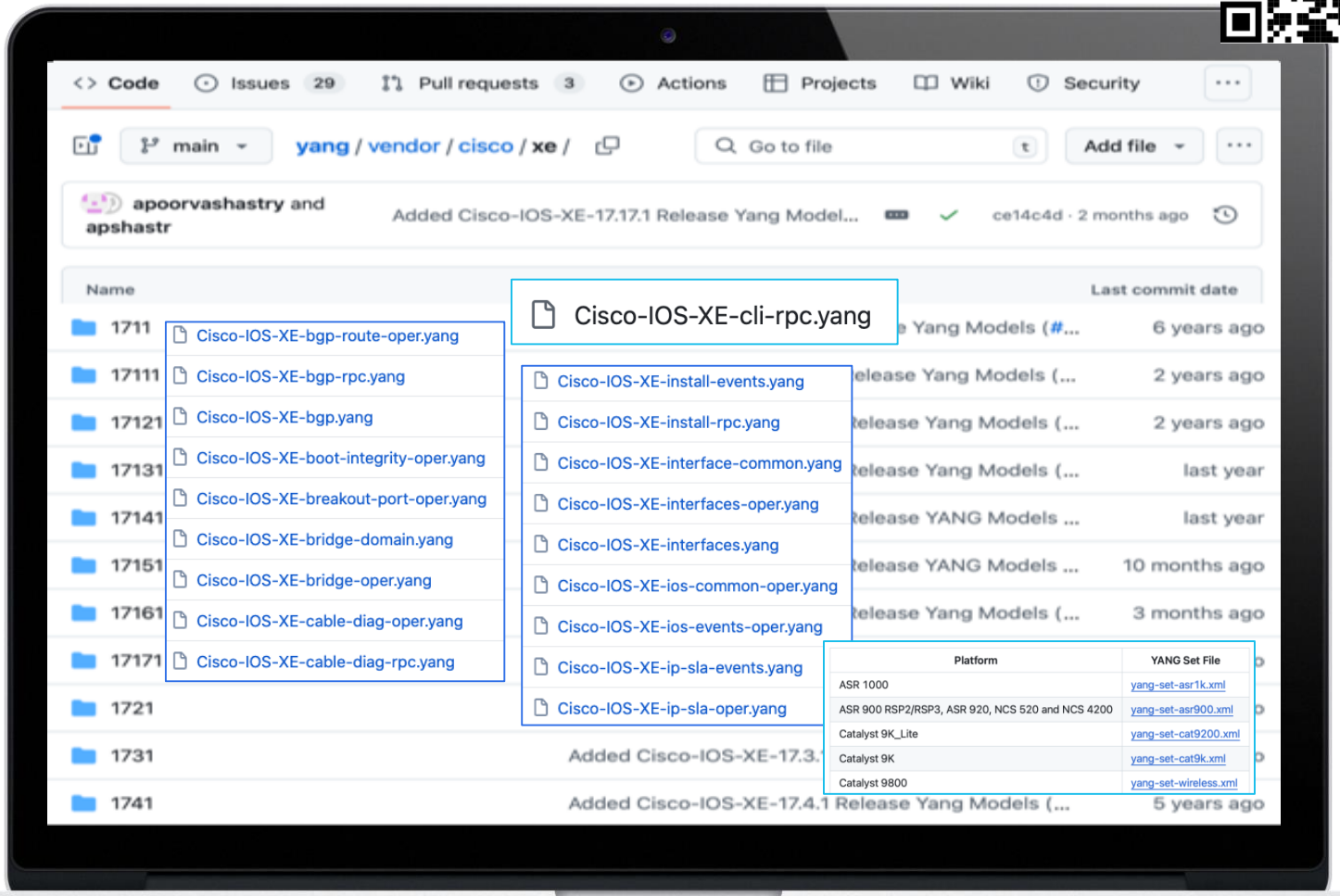
Line: 9:10 | XML | Tab Size: 4

YANG Models → **Data Models defined using the YANG language**

Cisco IOS XE - YANG model API documentation

1. There are 10 types of data models including config, oper, actions, and deviations
2. RFC 7950 YANG data modelling language are the API definitions for IOS XE
3. The YANG modules are available for download from the API and are also published on Github.com
4. Notable modules are listed below for the running-config, feature oper, actions and event notifications
5. XML “set” file list which modules are supported on the various hardware platforms

	TYPE	YANG module name.yang	Description
1	Native configuration model	Cisco-IOS-XE-native	running-config
2	Operation	Cisco-IOS-XE-{feature}-oper	Feature operational data
3	Configuration	Cisco-IOS-XE-{feature}-cfg	Feature configuration, independent of “native” model
4	Events	Cisco-IOS-XE-{feature}-events	Telemetry Events that can be triggered
5	RPC	Cisco-IOS-XE-{feature}-rpc	Actions that can be performed
6	Deviation	Cisco-IOS-XE-{feature}-deviation	Device implementation deviation from module
7	Types	Cisco-IOS-XE-{feature}-types	Types – Imported by other modules
8	Obsolete	Cisco-IOS-XE-{feature}-obsolete	Obsolete should not be implemented
9	Common	Cisco-IOS-XE-{feature}-common	Common – Imported by other modules
10	Abstractions	OpenConfig-{feature} & Cisco-evpn-service	abstraction for EVPN, OpenConfig config & oper



Since 17.10, YANG 1.1 modules are not advertised in NETCONF hello/capabilities but are available by reading the ietf-yang-library module

The YANG models are available for download directly from the running IOS XE device’s NETCONF, RESTCONF, or gNMI API, and from: https://github.com/YangModels/yang/tree/main/vendor/cisco/xe
Find the model for your usecase: https://www.yangcatalog.org

Cisco IOS XE – YANG model innovations

Deep dive into YANG models and FAQ, Common Models, etc

https://github.com/jeremycohoe/cisco-ios-xe-yang-model-innovations

YANG Model Innovations

1 Introduction & Overview
Model Types Overview
Summaries per Release

2 Config and Oper Models

3 PIN Specific
PIN Common
WNCD, PUBD
PIN Switching
PIN Routing
PIN Wireless

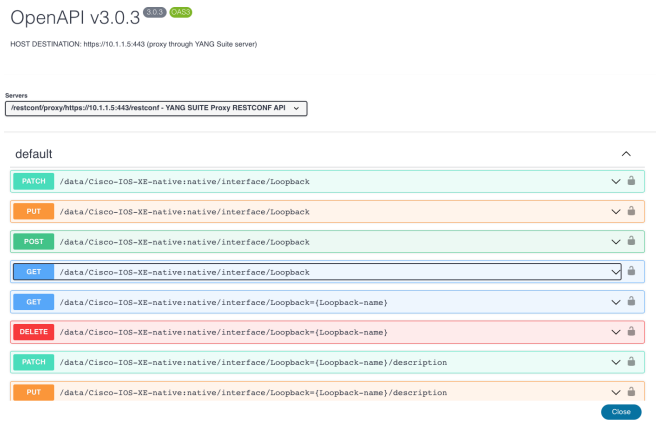
4 Model Abstractions
EVPN Service Model
OpenConfig
IETF

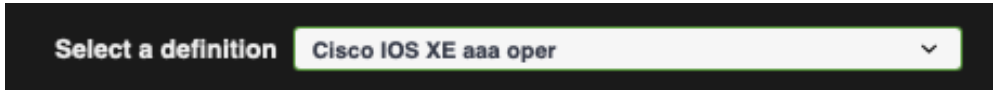
5 Custom Cisco
EXEC Actions
Telemetry & Telemetry Events
CLI RPC

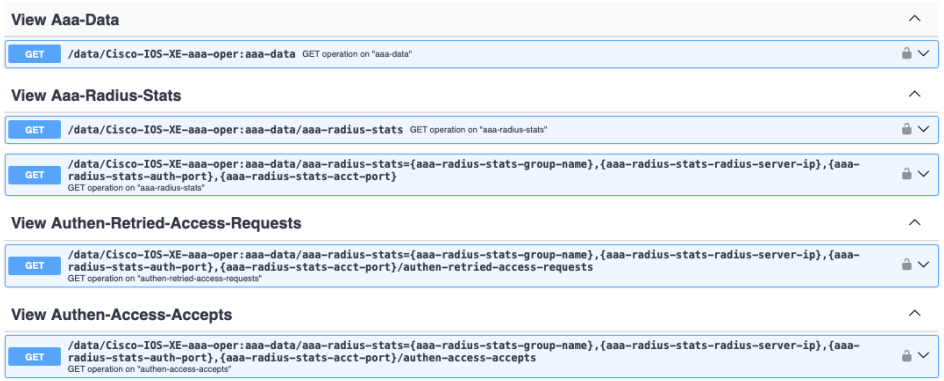
6 Release Deep Dive
17.13, 17.14, 17.15

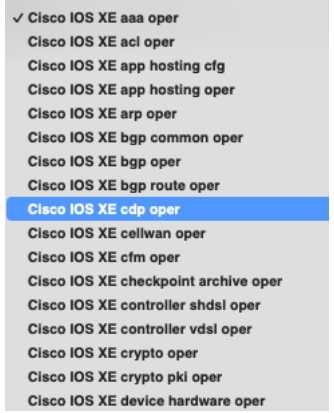
Swagger API Documentation

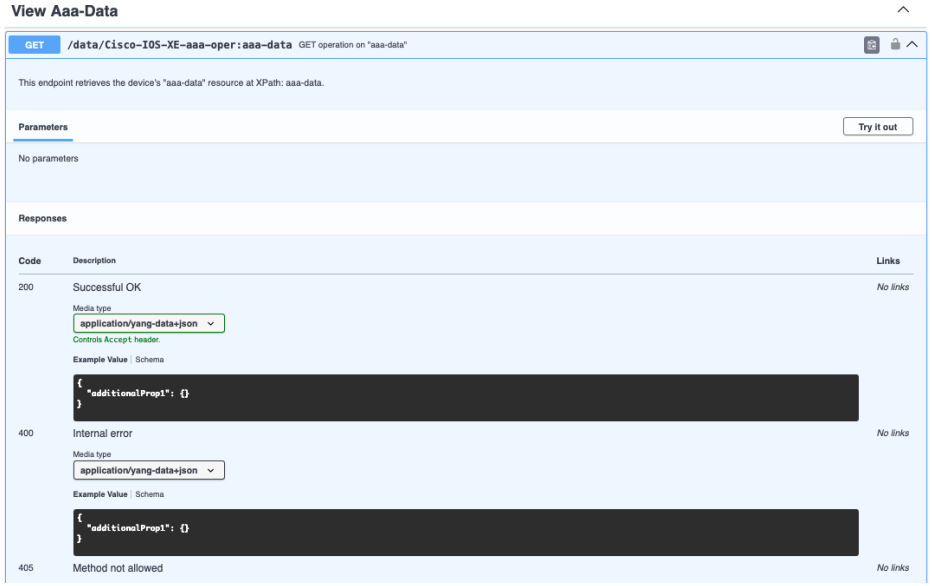
IOS XE release 17.15 API definitions posted to Github:
https://jeremycohoe.github.io/cisco-ios-xe-openapi-swagger/
Being moved into CiscoDevNet repository
Additional API's being added – currently all "oper" modules included
Native config model, openconfig, ietf, etc to be included
Swagger API generated from YANG Suite plugin !











API

Programmable Interfaces

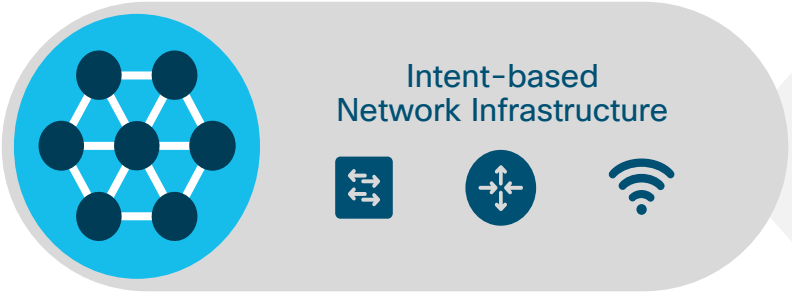
CLI

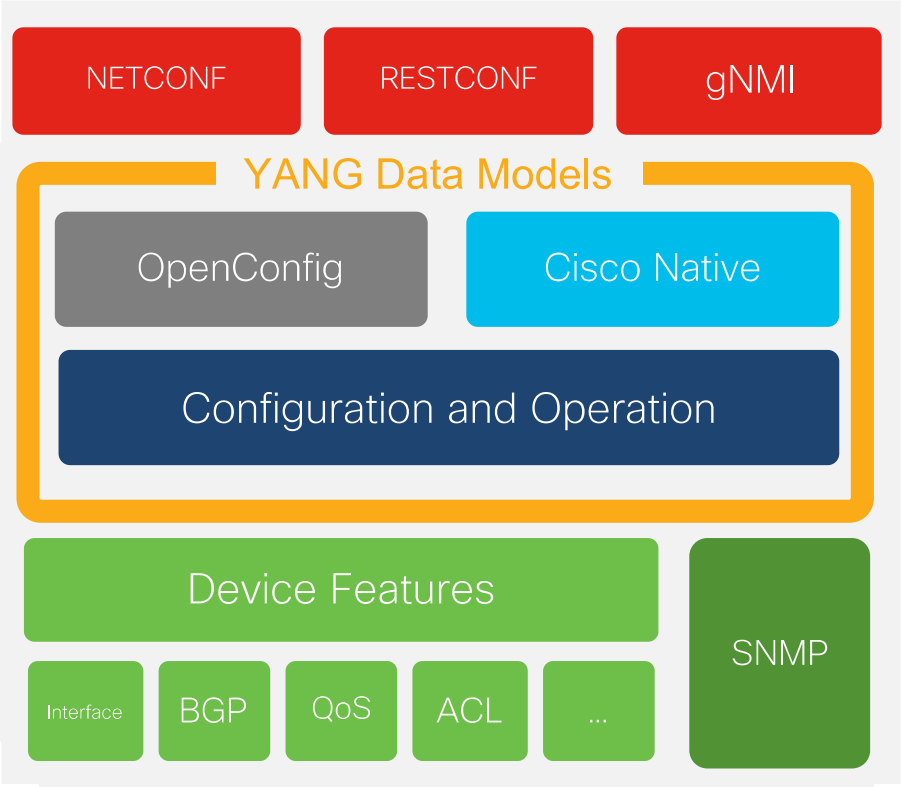
SNMP

WebUI

The NETCONF, RETCONF 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





Model Driven Programmability Interface Comparison

Protocol Benefits	NETCONF	RESTCONF	gNMI
Minimum IOS XE	16.6 (2017)	16.7 (2017)	16.8 (2018)
Default Port	830	443	9339
Operations	<get>,<get-config>,<edit-config>,<establish-subscription>	GET, POST, PUT, PATCH, DELETE	GET, SET, SUBSCRIBE
Encoding	XML	XML or JSON	RFC7951JSON_IETF + Proto
Security	SSH + PKI certificate or password	HTTPS user/pass	mTLS certificate with user authentication
Transport Protocol	SSH	HTTPS	HTTP/2
Tooling	YANG Suite, ncclient, Netconf-console	YANG Suite, Postman, python, curl	YANG Suite, gnmic, gnmi_cli
Content	YANG	YANG	YANG + Protobuf
Benefits/Pros	Mature API with candidate datastores, validation, rollback	REST is very common API used across industry, well know operations	Single secure API for config + Telemetry
Caveats/Cons	XML can be slow & difficult to use	No telemetry support, limited datastore support	“New” API, inconsistent implementations across vendors, Go lang, no rollback etc

Multiple API’s, same YANG data: the choice is yours

NETCONF

Request Interfaces Data

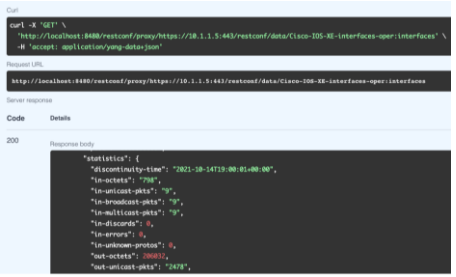
```
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:af602e50-6490-4b84-abff-5461a8e37502">
  <nc:get>
    <nc:filter>
      <interfaces xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-interfaces-oper"/>
    </nc:filter>
  </nc:get>
</nc:rpc>
```

Reply from Device

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:af602e50-6490-4b84-abff-5461a8e37502">
  <data>
    <interfaces xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-interfaces-oper">
      <interface>
        <name>GigabitEthernet0/0</name>
        <interface-type>iana-iftype-ethernet-csmacd</interface-type>
        <admin-status>if-state-up</admin-status>
        <oper-status>if-oper-state-ready</oper-status>
        <last-change>2025-06-04T16:48:26.731+00:00</last-change>
        <if-index>1</if-index>
        <phys-address>00:50:56:bf:77:ea</phys-address>
        <speed>1000000000</speed>
        <statistics> ... </statistics>
...
</interface>
    </interfaces>
  </data>
</rpc-reply>
```

RESTCONF

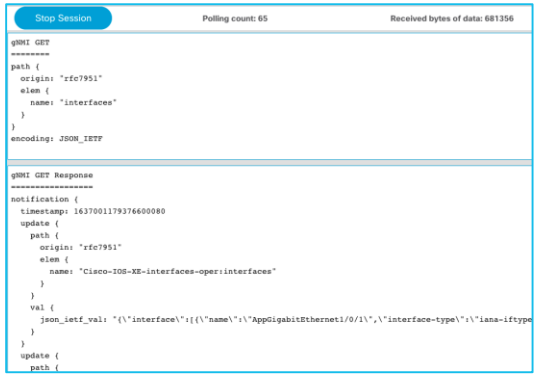
```
curl -X 'GET' \
'http://0.0.0.0:8480/restconf/proxy/https://devnetsandboxiosxec9k.cisco.com:443/restconf/data/Cisco-IOS-XE-interfaces-oper:interfaces/interface' \
  -H 'accept: application/yang-data+json'
```




```
{
  "Cisco-IOS-XE-interfaces-oper:interface": [
    {
      "name": "GigabitEthernet0/0",
      "interface-type": "iana-iftype-ethernet-csmacd",
      "admin-status": "if-state-up",
      "oper-status": "if-oper-state-ready",
      "last-change": "2025-06-04T16:48:26.49+00:00",
      "if-index": 1,
      "phys-address": "00:50:56:bf:77:ea",
      "speed": "1000000000",
      "statistics": {
      ...
      }.
...
    }
  ]
}
```

gNMI

```
{
  "path": [
    {
      "origin": "rfc7951",
      "elem": [
        {
          "name": "Cisco-IOS-XE-interfaces-oper:interfaces"
        },
        {
          "name": "interface"
        }
      ]
    }
  ],
  "encoding": "JSON_IETF"
}
```



CLI

```
show interfaces
```

```
CAT9k_AO# show interfaces
Vlan1 is up, line protocol is down , Autostate E
  Hardware is Ethernet SVI, address is 0050.56bf
  MTU 1500 bytes, BW 1000000 Kbit/sec, DLY 10 us
     reliability 255/255, txload 1/255, rxload 1
  Encapsulation ARPA, loopback not set
  Keepalive not supported
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input never, output never, output hang ne
  Last clearing of "show interface" counters nev
  Input queue: 0/375/0/0 (size/max/drops/flushes
  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
     0 packets input, 0 bytes, 0 no buffer
     Received 0 broadcasts (0 IP multicasts)
     0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun,
     0 packets output, 0 bytes, 0 underruns
     Output 0 broadcasts (0 IP multicasts)
     0 output errors, 2 interface resets
     0 unknown protocol drops
     0 output buffer failures, 0 output buffers
```

CLI to YANG: show run format netconf/restconf

1. This CLI addition to “show run | format” brings additional visibility into the YANG modelled configuration
2. NETCONF with XML and JSON with RESTCONF for Cisco-IOS-XE native models only
3. **Easily convert CLI into YANG** to re-use in tooling, scripts, and automation and orchestration systems

show run | format netconf-xml
show run | format restconf-json

```
C9300#show run | format netconf-xml
<config xmlns="http://tail-f.com/ns/config/1.0">
  <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
    <version>17.7</version>
    <memory>
      <free>
        <low-watermark>
          <processor>131752</processor>
        </low-watermark>
```

```
C9300#show run | format restconf-json
{
  "data": {
    "Cisco-IOS-XE-native:native": {
      "version": "17.7",
      "memory": {
        "free": {
          "low-watermark": {
            "processor": 131923
```

```
C9300#
C9300#show run | i netconf-yang
netconf-yang
C9300#
```

- Requires netconf-yang Data Model Interfaces to be enabled
- CLIs with corresponding native YANG and modeled in show run are returned
- No support for “show run all” or additional parameters
- NETCONF “Get-Config” RPC can also get used to get the YANG modelled configuration

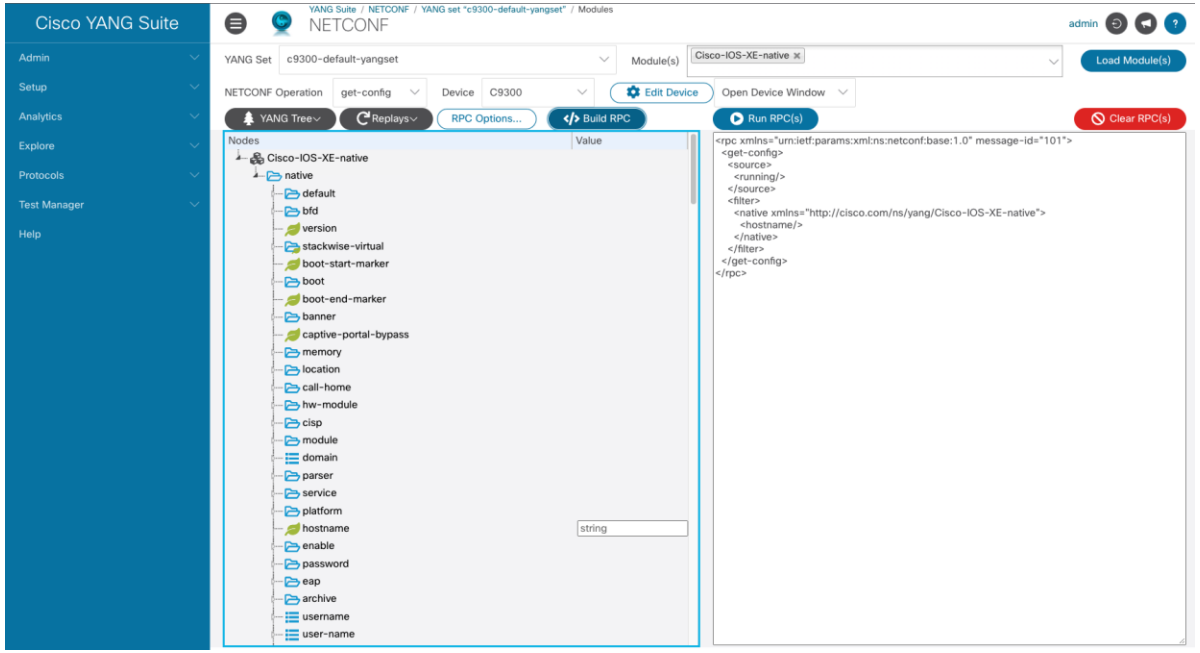
About NETCONF

- NETCONF provides a programmatic interface based on standard mechanisms for accessing configuration data, state data, data-model-specific Remote Procedure Call (RPC) operations and events, defined in the YANG model.
- NETCONF uses XML formatting for machine-friendly data transfers and supports the operations described below.
- The YANG Suite NETCONF plugin provides options to execute and visualize the YANG data model.

Attend the ACR Session tomorrow to learn more!

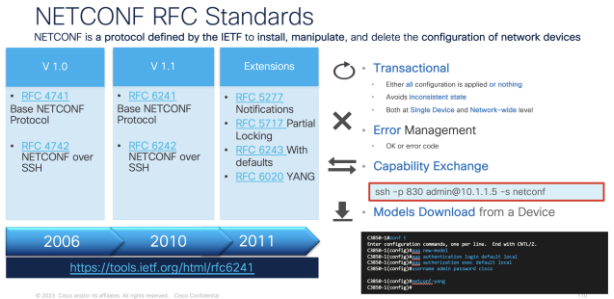
1:00 PM – 2:00 PM

BRKENS-2604
Atomic Config Replace with Cisco Catalyst 9000



Why use NETCONF?

- Candidate datastore
- Easily convert CLI to YANG using format | Netconf-xml
- NETCONF Access Control Model (NACM)
- Commonly used with Ansible
- Confirm Commit
- Mature RFC standard



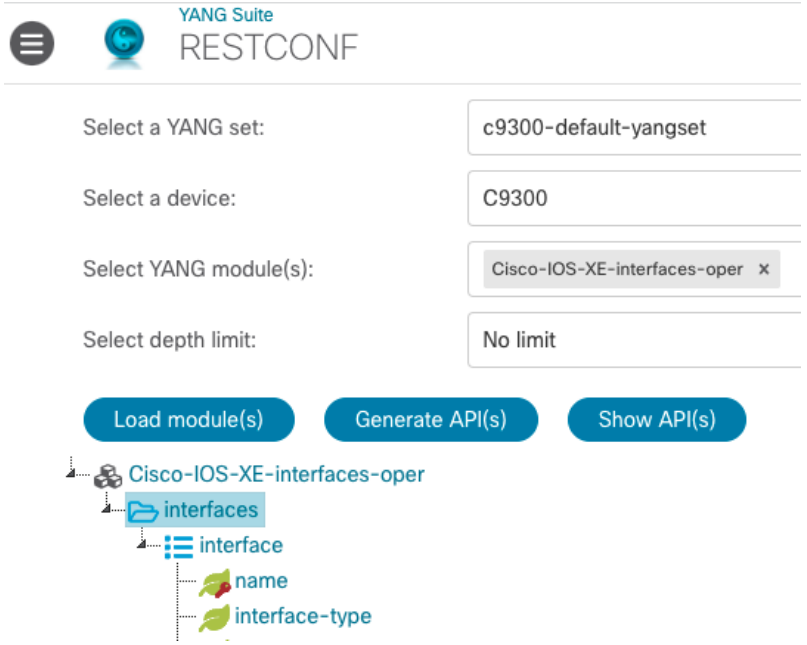
NETCONF Operations

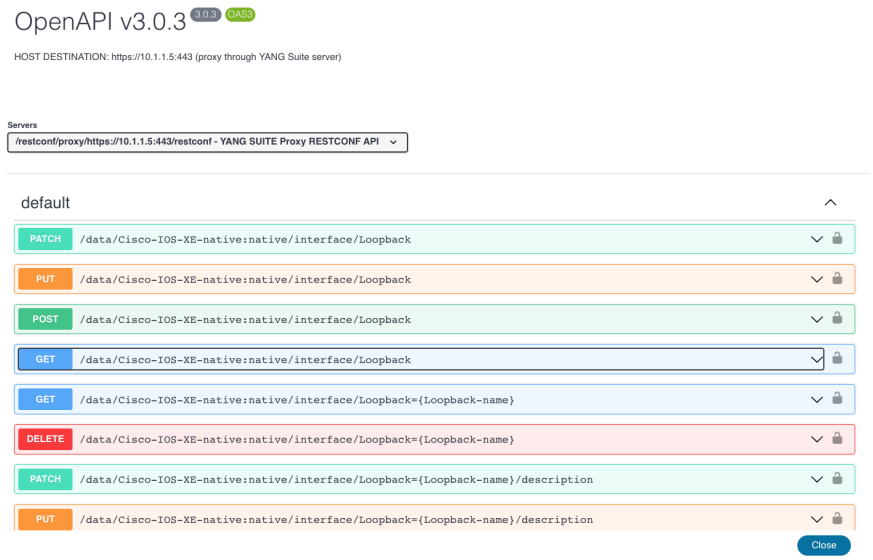
Main Operations		Description
<get>	(close to 'show ?')	Retrieve running configuration and device state information
<get-config>	(close to 'show run')	Retrieve all or part of specified configuration datastore
<edit-config>	(close to 'conf t')	Loads all or part of a configuration to the specified configuration datastore

Other Operations	Description
<copy-config>	Replace an entire configuration datastore with another
<delete-config>	Delete a configuration datastore
<commit>	Copy candidate datastore to running datastore
<lock> / <unlock>	Lock or unlock the entire configuration datastore system
<close-session>	Graceful termination of NETCONF session
<kill-session>	Forced termination of NETCONF session

About RESTCONF

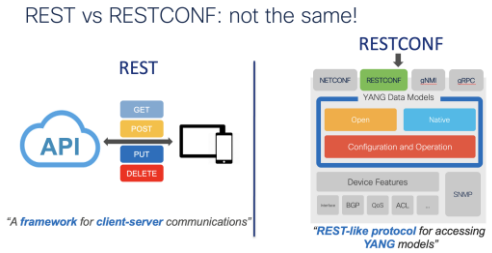
- RESTCONF provides a programmatic interface based on standard mechanisms for accessing configuration data, state data, data-model-specific Remote Procedure Call (RPC) operations and events, defined in the YANG model.
- The YANG Suite RESTCONF plugin provides Swagger UI and execution visualization of the YANG data model.
- Use YANG Suite or Postman tool





Why use RESTCONF?

- REST-like API
  - Easy to use with edit & get operations
  - Swagger documentation
- Commonly used with cURL commands, JSON-formatting and Python scripting
- Terraform enablement

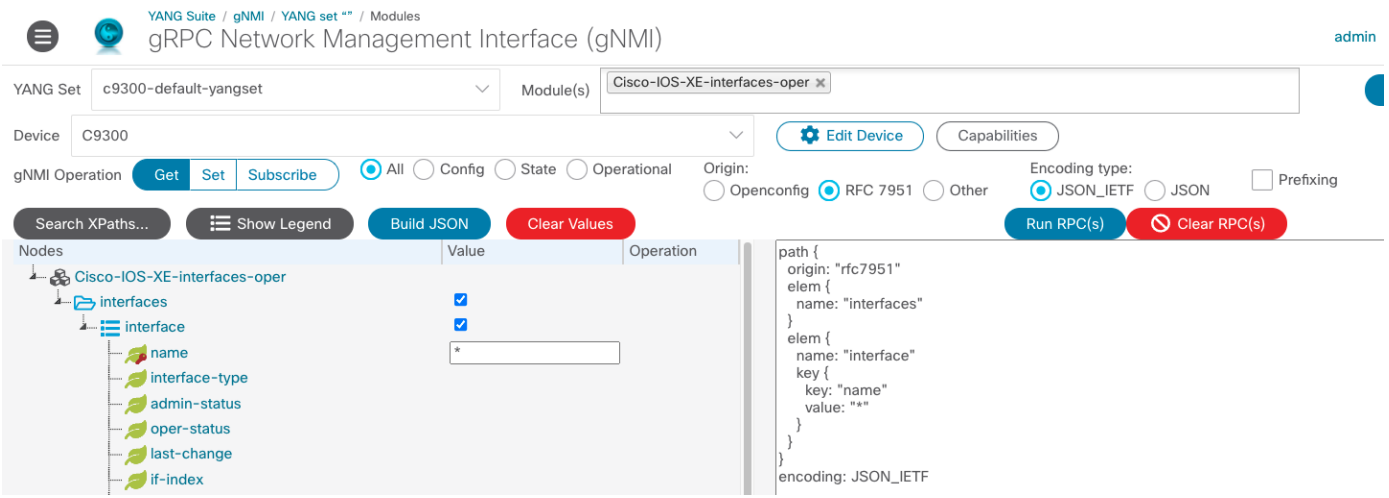


About gNMI...

1:30 PM – 2:30 PM

BRKDEV-2017
gRPC, gNMI, gNOI... Oh My! An Enterprise Network Automation Journey

gNMI provides a programmatic interface based on standard mechanisms for accessing configuration data, state data, data-model-specific Remote Procedure Call (RPC) operations and events, defined in the YANG model. gNMI supports GET, SET and SUBSCRIBE operations. The YANG Suite gNMI plugin provides execution and visualization of the YANG data model.



There are 100+ "actions" in YANG that are modelled for NETCONF/RESTCONF operations, but **NOT** for gNMI !

TLS certificate management, OS version upgrades, factory reset

100 other actions: "clear", reset, save, copy, ping, etc NOT with gNIMI

Why use gNMI?

- gNMI API supports config management through the "SET" operation and includes a "Subscribe" for model-driven telemetry. It also has some features for TLS certificate management, OS version upgrades, and factory-reset API's
- gNMI is quickly becoming the standard API used across network devices because it supports device configuration and telemetry all while its encoding mechanism is just faster and more efficient

So which API does Cisco recommend ?

It depends! On the usecase, tooling maturity, NetDevOps investment, the complexity of the problem, etc
This is a brief introduction... API's will be covered in more detail after Day0/Onboarding

API	Pros	Cons
NETCONF	• Candidate datastore capability: Option to work with Running datastore or candidate datastore (a sandbox-like environment for testing and validation on-box before pushing code to the running config) • Supports config, retrieving data and telemetry • Telemety data comes faster than SNMP • Confirm Commit • Standards-based	• XML is slower than gNMI / gRPC • Machine friendly (not human friendly) • Requires installing libraries for ease of use (ncclient)
RESTCONF	• Works like other REST-based APIs with standard operations like GET, PUT, POST, etc • No additional libraries to install for use (REST is widely supported) • Standards-based • Swagger documentation for use of use	• No telemetry support. If RESTCONF is the only option, aggressive poling through GET requests can be used in place of telemetry • HTML/JavaScript is commonly used for REST, which is slower than gNMI / gRPC
gNMI	• Capabilities for GET, SET, and Subscribe operations • Telemetry data comes faster than other APIs or SNMP because gRPC uses protobuf encoding which is more efficient and faster to serialize & deserialize the data	• Non-standards-based (rather it is operator-led), meaning YANG not always at parity (actions,events,on-change,etc don't work) • Machine friendly (not human friendly)
gRPC	• Fast Push-based telemetry	• Only telemetry support, meaning no device config or single gets (use gNMI)

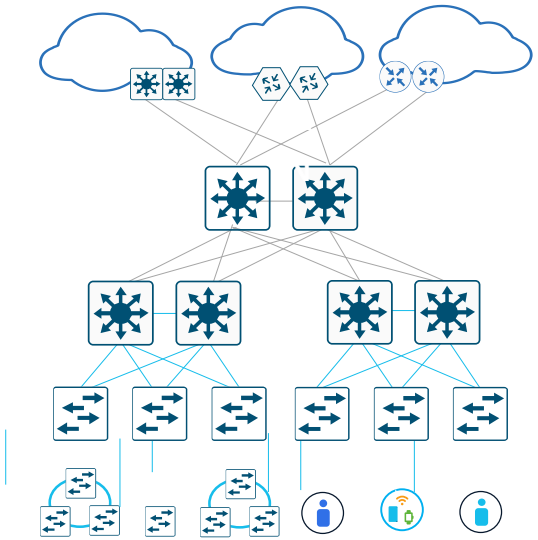
Tooling

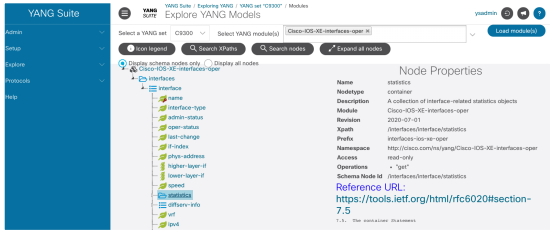
YANG Suite
Ansible
Terraform

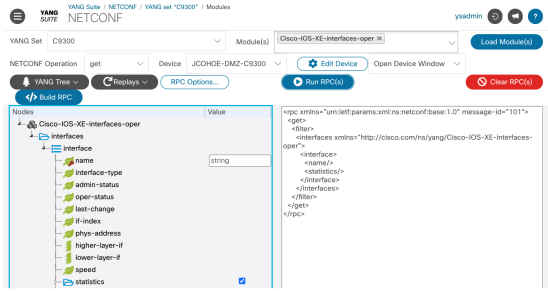
Empower your NetOps with Cisco YANG Suite
Best in class tooling for open programmability

Free and publicly available!

Complex Network Operations | **Cisco YANG Suite** | **Customer Benefits** | **Use Cases**







Simplified migration from CLI and SNMP to YANG

Toolkit to integrate with existing workflows

Guided workflow for all things YANG

API Testing and Validation Environment for NETCONF, RESTCON, gNMI & gRPC

Legacy Protocol Migration

Device Automation

Network Monitoring

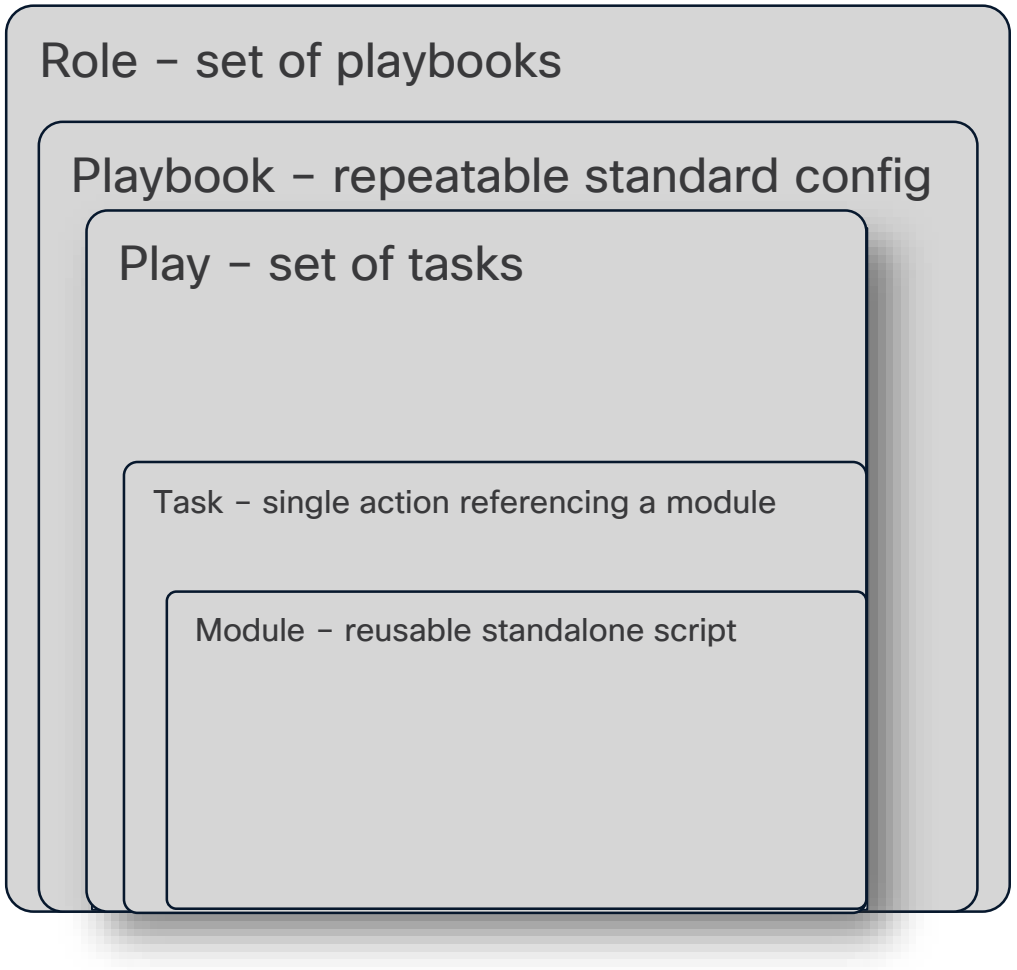
Compliance and Coverage

Get hands-on using the new learning lab!
https://developer.cisco.com/learning/labs/intro-yangsuite/
Docker container innovation 1 container

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

Simplify your device programmability journey with Cisco YANG Suite

What is Ansible?



- Open-source configuration management tool
- Commercially supported by Red Hat
- Declarative and idempotent
- Ansible can be Imperative when needed
- Can manage a wide range of systems:

  VMs, network devices, cloud instances, etc.
- Agentless: no requirement for installation of application software to run
- Has Python server-side dependencies

http://docs.ansible.com/ansible/latest/YAMLSyntax.html

Ansible with Cisco Catalyst & IOS XE

Ansible has full support for configuration management of Cisco IOS XE and Catalyst 9000 using plugins for CLI, SNMP, and for the YANG based API's including NETCONF, RESTCONF, and gNMI.

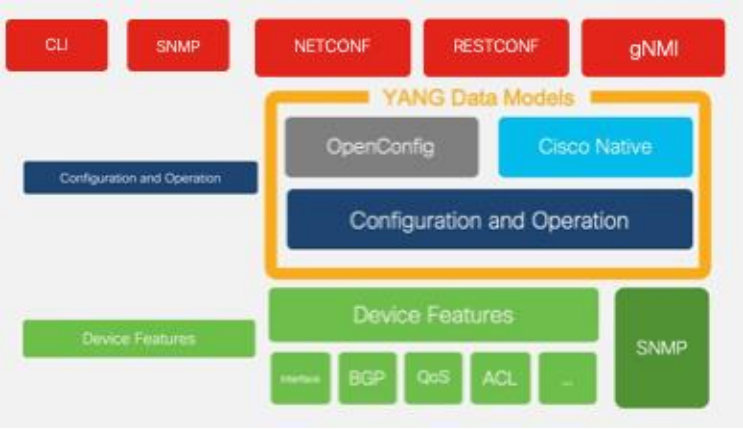
Integrations with Cisco IOS XE & Catalyst 9000:
2 CLI modules are most common:

ios_config – config CLI
ios_command – show CLI

Various CLI feature modules have been created to support specific usecases including: bgp, l2_interface, ntp, vlan, vrf, etc

The programmatic API interfaces of NETCONF/ RESTCONF/GNMI are also supported with a variety of modules and playbooks

DevNet has code samples, learning labs, and sandbox



Ios

- ios_banner – Manage multiline banners on Cisco IOS devices
- ios_bgp – Configure global BGP protocol settings on Cisco IOS

Netconf

- netconf_config – netconf device configuration
- netconf_get – Fetch configuration/state data from NETCONF enabled network devices

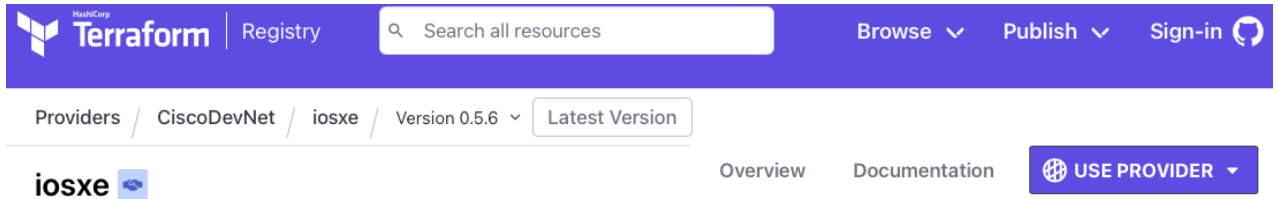
Restconf

- restconf_config – Handles create, update, read and delete of configuration data on RESTCONF enabled devices
- restconf_get – Fetch configuration/state data from RESTCONF enabled devices

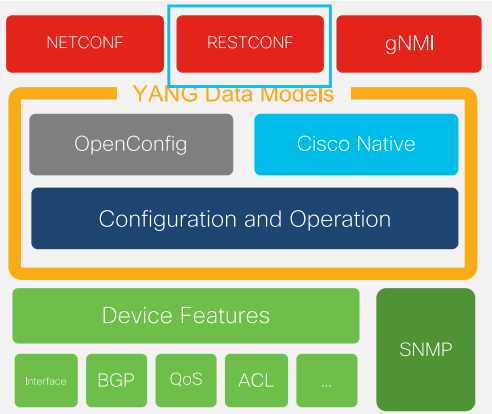
https://github.com/ansible-collections/cisco.ios
https://docs.ansible.com/ansible/latest/collections/ansible/netcommon/netconf_config_module.html

Terraform with Cisco Catalyst & IOS XE

The 1 ”iosxe” terraform provider manages configuration declaratively across 19 features using 98 resources & data sources
There are 2 additional resources for imperative configuration using both CLI and YANG
The imperative resources take YANG modelled inputs and the Cisco-IOS-XE-cli-rpc.YANG is used to abstract CLI over RESTCONF



Terraform interfaces with the IOS XE RESTCONF API



Declarative support

- iosxe provider
- Guides
- AAA
- BFD
- BGP
- CTS
- Crypto
- EVPN
- General
- Interface
- MDT
- MPLS
- Management
- Multicast
- OSPF
- QoS
- Routing
- Switching
- System
- VRF

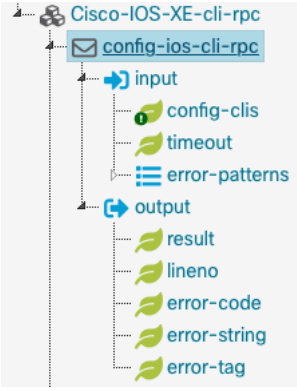
Imperative support

- General
  - Resources
    - iosxe_cli
    - iosxe_restconf
    - iosxe_save_config
  - Data Sources
    - iosxe_restconf

iosxe_cli

```
resource "iosxe_cli" "example" {
  cli = <<-EOT
  interface Loopback123
  description configured-via-restconf-cli
  EOT
}
```

CLI RPC YANG Model



iosxe_restconf

```
resource "iosxe_restconf" "nested_list" {
  path = "Cisco-IOS-XE-native:native/ip"
  attributes = {
    source-route = "true"
  }
  lists = [{
    name = "vrf"
    key  = "name"
    items = [
      {
        name = "VRF1"
      }
    ]
  }]
```

Documentation on registry https://registry.terraform.io/providers/CiscoDevNet/iosxe/latest
Source code in Cisco DevNet GitHub repo: https://github.com/CiscoDevNet/terraform-provider-iosxe/
RESTCONF guide https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1715/b_1715_programmability_cg/m_1715_prog_restconf.html
CLI RPC YANG API https://github.com/YangModels/yang/blob/main/vendor/cisco/xe/17121/Cisco-IOS-XE-cli-rpc.yang

DEMO: Terraform Add Device Config

Use Terraform to manage the gRPC Dial-Out Telemetry subscriptions
https://registry.terraform.io/providers/CiscoDevNet/iosxe/latest/docs/resources/mdt_subscription
Example files: https://github.com/jeremycohoe/cisco-ios-xe-mdt/tree/master/sustainability

auto@pod27-xelab: ~

```
auto@pod27-xelab:~$
auto@pod27-xelab:~$
auto@pod27-xelab:~$
auto@pod27-xelab:~$
auto@pod27-xelab:~$
auto@pod27-xelab:~$
auto@pod27-xelab:~$
auto@pod27-xelab:~$
auto@pod27-xelab:~$
auto@pod27-xelab:~$
auto@pod27-xelab:~$ docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
auto@pod27-xelab:~$
auto@pod27-xelab:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
auto@pod27-xelab:~$
auto@pod27-xelab:~$
```

```
c9300-pod27# sh run | s tel
telemetry ietf subscription 6041337
 encoding encode-kvgpb
 filter xpath /process-cpu-ios-xe-oper:cpu-usage/cpu-utilization/five-seconds
 stream yang-push
 update-policy periodic 30000
 receiver ip address 10.1.1.3 57500 protocol grpc-tcp
c9300-pod27#
c9300-pod27#
Oct 16 21:55:56.082: %HA_EM-6-LOG: catchall: show running-config
```

1. Ensure Docker is installed
2. Run the following commands
   1. docker pull jeremycohoe/tig_mdt
   2. docker run -ti -p 3000:3000 -p 57500:57500 jeremycohoe/tig_mdt
3. Identify the container ID
   1. docker ps
4. Enter into the container
   1. docker exec -it CONTAINER_ID /bin/bash
      1. Note: replace CONTAINER_ID with the ID found in step 3
5. Navigate to the correct folder
   1. cd
   2. cd cisco-ios-xe-panda-lab-terraform
6. Modify the device credentials in the header.tf file
   provider "iosxe" {
   username = "admin" ← replace with Cisco IOS XE device username
   password = "XXXXXXXX" ← replace with Cisco IOS XE device password
   url = "https://your-switch-hostname-or-ip" ← replace with Cisco IOS XE device hostname or I
   }
7. Configure the Cisco IOS XE device using Terraform

They can coexist… it's not an either/or story

Terraform

Manage infrastructure as code

https://developer.hashicorp.com/terraform

- Terraform keeps state locally
- Terraform knows what is configured vs desired end-state
- Terraform can automatically destroy/recreate resources

Red Hat
Ansible Automation Platform

https://www.redhat.com/en/ansible-collaborative

- Ansible mutates the infrastructure
- Ansible does not keep state (mostly)
- Terraform can call Ansible to perform tasks post resource deployment (e.g., on VMs deployed)

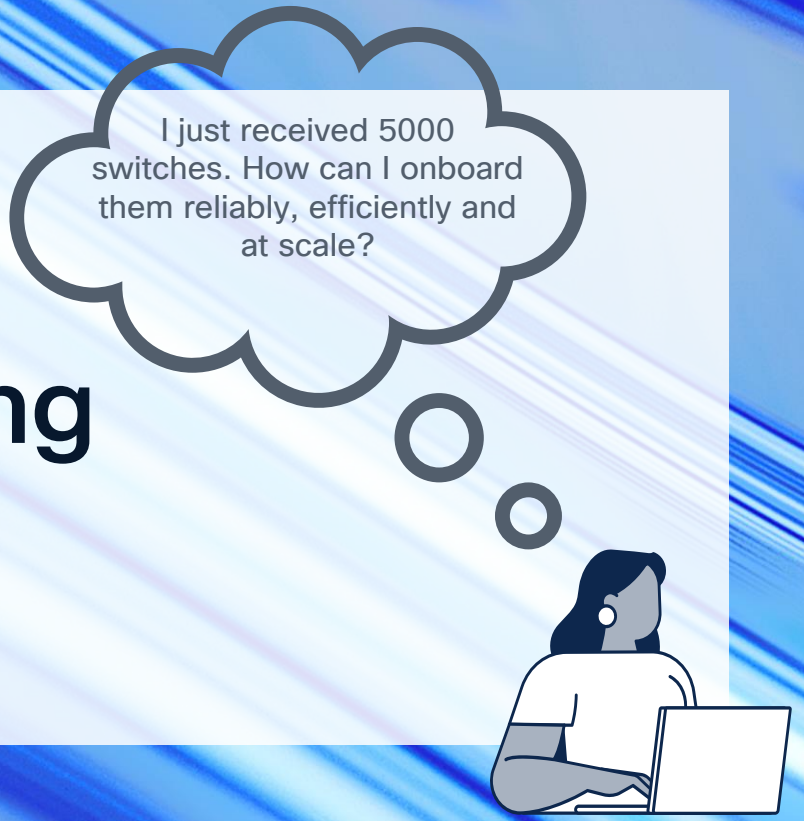
Tooling Comparison

This is not a complete list!

Other common tools: Postman(REST), gNMIc, Python, GoLang, https://httpie.io

Tool	YANG Suite	Ansible	Terraform
Use Case	To understand how YANG models work To help build a payload	Task-based tooling	Infrastructure-as-Code (IaC) tooling
Supported Protocols	NETCONF, RESTCONF, gNMI, gRPC (and more capabilities such as diffs, SNMP to YANG, etc)	NETCONF, RESTCONF	RESTCONF
Benefits	Getting started to understand YANG models and build payloads	Imperative – procedural (step-by-step)	Declarative
Interface	GUI web tool	CLI / YAML files	CLI / HCL (similar to JSON) files

Device Onboarding



Why It’s Time for a Change?

Limitations of Manual Network Provisioning

AI Generated Image
Prompt: **A real person in a server room, surrounded by racks of switches and servers. The person has connected their laptop to one of the racks with a wire and is intently working on their laptop. The server room is well-lit with a cool, blue-toned lighting, reflecting off the metallic surfaces of the servers. The person is focused, with a look of concentration on their face, as they navigate through their laptop. The background shows rows of servers, with blinking lights indicating active connections and data transfer.**

Time Consuming & Error prone

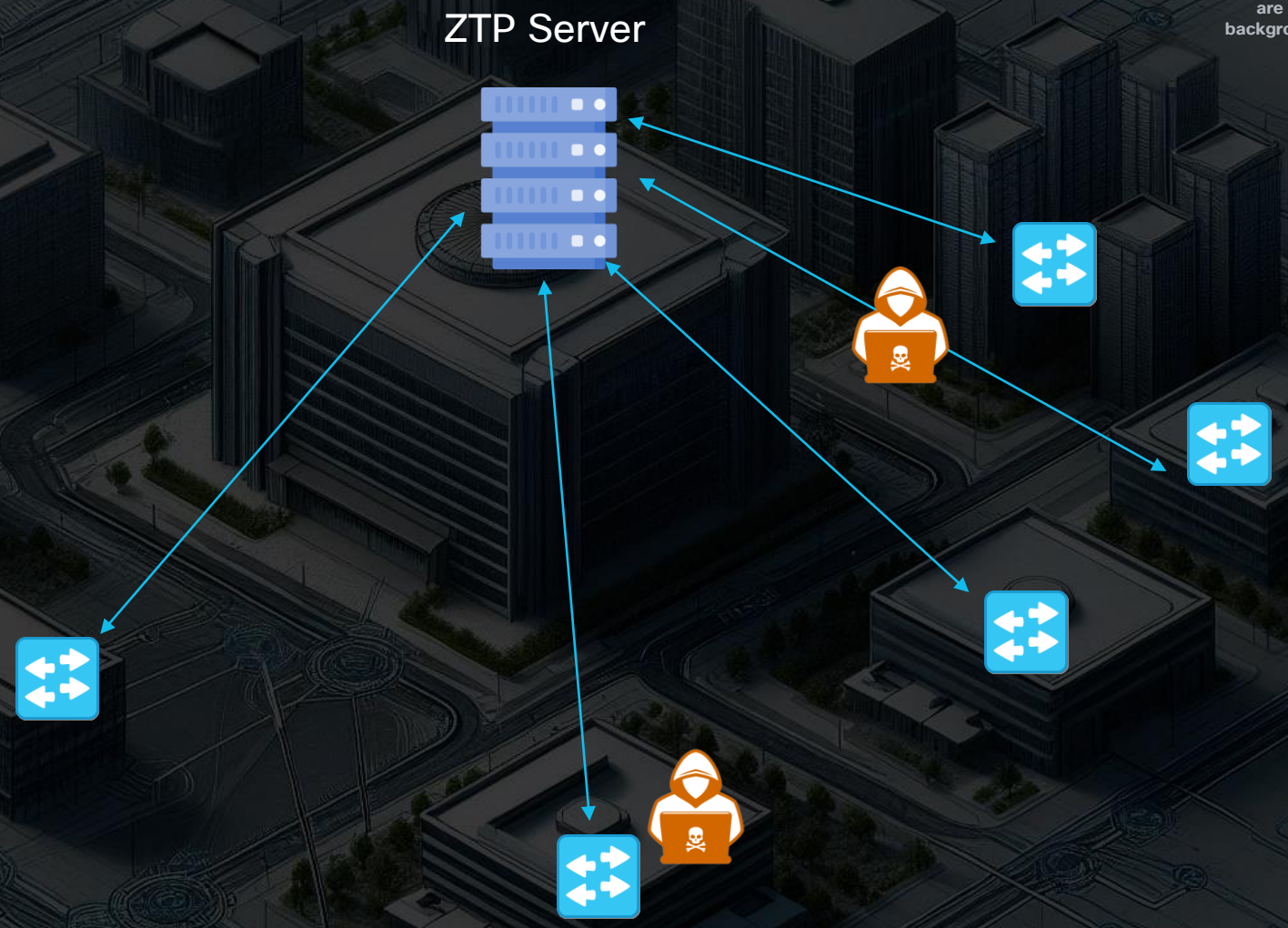
Limited Scalability with High Operational Costs

Security Risks

Day – 0 Network Automation

Zero-Touch Provisioning

AI Generated Image
Prompt: **A detailed illustration of an enterprise network, featuring a large main campus building at the center, surrounded by only 4 to 5 smaller branch office buildings. The network connections between the main campus and branch offices are clearly depicted, with lines representing data flow. The main campus is modern and sleek, with glass windows and a well-maintained exterior. The branch offices are smaller but share a similar architectural style. The background includes a subtle cityscape, with a clear blue sky and a few scattered clouds.**



15%

Organizations identified a supply chain compromise as the source of a data breach [1]

$4.8 Million

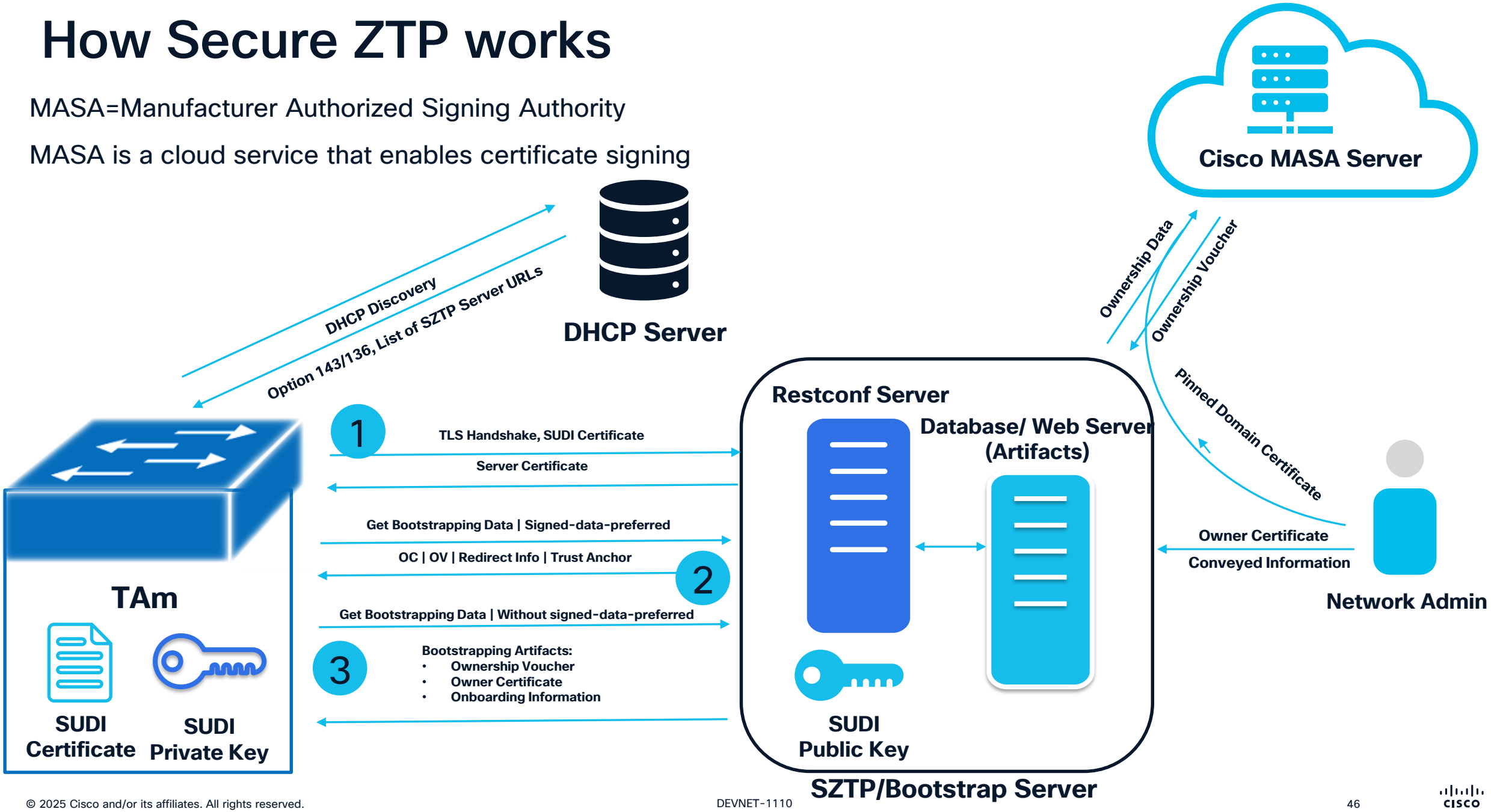
Is the global average cost of a data breach, increased by 10% compared to the previous year [1]

[1] - Cost of a Data Breach Report 2023/24 - IBM

How Secure ZTP works

MASA=Manufacturer Authorized Signing Authority

MASA is a cloud service that enables certificate signing

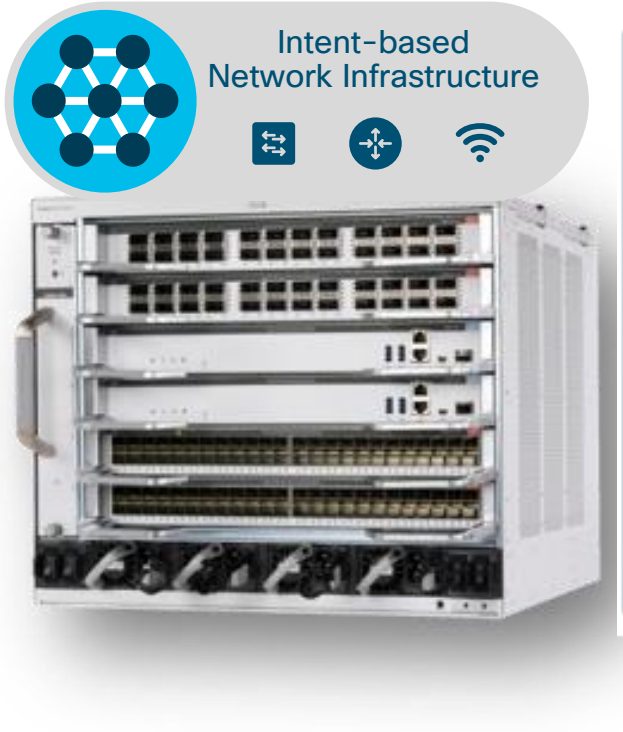


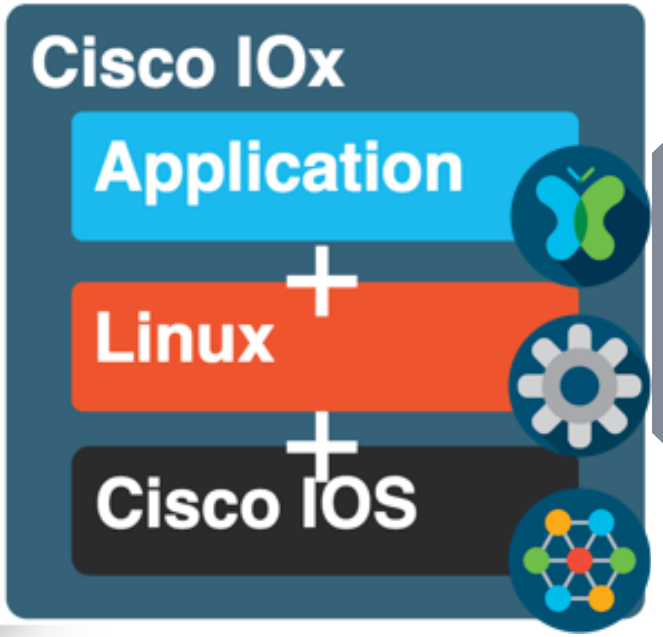
Manual Provisioning vs ZTP vs SZTP

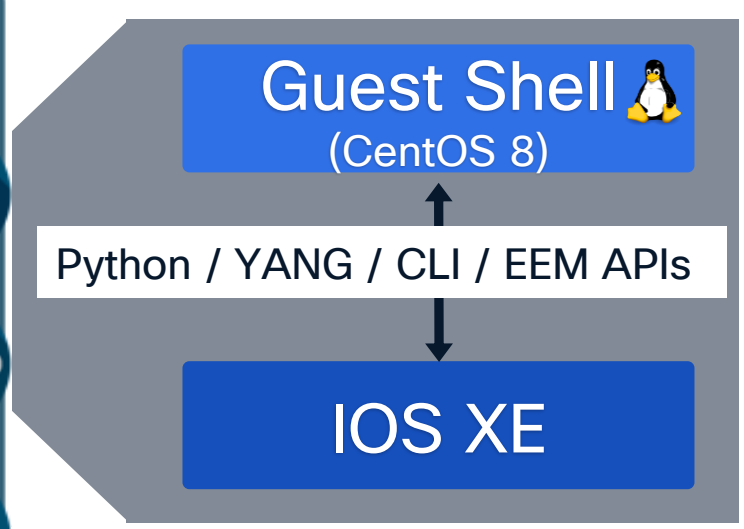
	Manual Provisioning	ZTP	Secure - ZTP
Provisioning Automation	✗	✓	✓
Configuration Consistency	✗	✓	✓
Scalability	✗	✓	✓
End to End Security	✗	✗	✓
Audit Logs	✗	✗	✓
Easy Image Upgrade	✗	✗	✓

ZTP, Gust Shell with CentOS 8 and Python 3

- ZTP runs when there is no config and DHCP Option 43 is set
- Embedded CentOS linux container with API's into IOS XE
- Disabled by default, enable with interface config + "guestshell enable" CLI







CentOS

https://www.centos.org/

Example Python ZTP Script (ztp-simple.py)

```
1  print "\n\n *** Sample ZTP Day0 Python Script *** \n\n"
2  # Importing cli module
3  import cli
4
5  print "Configure vlan interface, gateway, aaa, and enable netconf-yang\n\n"
6  cli.configurep(["int vlan 1", "ip address 10.5.123.27 255.255.255.0", "no shut", "end"])
7  cli.configurep(["ip default-gateway 10.5.123.1", "end"])
8  cli.configurep(["username admin privilege 15 secret 0 XXXXXXXXXXXXX"])
9  cli.configurep(["aaa new-model", "aaa authentication login default local", "end"])
10 cli.configurep(["aaa authorization exec default local", "aaa session-id common", "end"])
11 cli.configurep(["netconf-yang", "end"])
12
13 print "\n\n *** Executing show ip interface brief  *** \n\n"
14 cli_command = "sh ip int brief"
15 cli.executep(cli_command)
16
17 print "\n\n *** ZTP Day0 Python Script Execution Complete *** \n\n"
```

https://github.com/jeremycohoe/IOSXE-Zero-Touch-Provisioning

NETCONF API

The NETCONF interface on Cisco IOS XE is accessible from within the Guest Shell, which can be used at Day 0. No interface configuration or connectivity is required.

The **ncclient** Python library can be used to connect to the NETCONF interface when there is no IP connectivity, similar to the Python CLI modules and API. This can be used by ZTP at Day 0 to programmatically configure the device using either CLI or YANG.

```
C9300(config)#netconf-yang ssh port ?
  <1-65535>  Port number range (default port number is 830)
  disable    Disable external NETCONF SSH connectivity

C9300(config)#netconf-yang ssh local-vrf guestshell ?
  enable  Enable NETCONF access
  port    Configure port number for the NETCONF ssh connection
```

```
[guestshell@guestshell ~]$ ssh localhost -p 830 -l admin
admin@localhost's password:
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
<capability>urn:ietf:params:netconf:base:1.0</capability>
<capability>urn:ietf:params:netconf:base:1.1</capability>
```

ZTP

CLI API
Enable NETCONF
Enable AAA

NETCONF API
Preform RPC Actions
Programmatic Configuration of device features

```
[guestshell@guestshell ~]$ python3 ./get_hostname.py
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:20d75dd3-60b7-4b1e-9260-0438282d37c8" xmlns="urn:
        <data>
                <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
                        <hostname>C9300</hostname>
                </native>
        </data>
</rpc-reply>
```

Authentication from Guest Shell to NETCONF is still required, both credentials and certificates are supported
get_hostname.py example at https://github.com/jeremycohoe/ncclient-get-hostname
ztp-Netconf.py example at. https://github.com/jeremycohoe/IOSXE-Zero-Touch-Provisioning/blob/master/ztp-netconf.py

Admin Setup for Secure ZTP Workflow

1. 

   Order 5000 new switches

2. 

   Receive 5000 new switches with serial numbers for each

3. 

   Add each serial number (+PDC) to MASA

4. 

   Generate OV per device from MASA

5. 

   Put OVs into bootstrapping service

Day 0 SZTP onboarding complete!

6. 

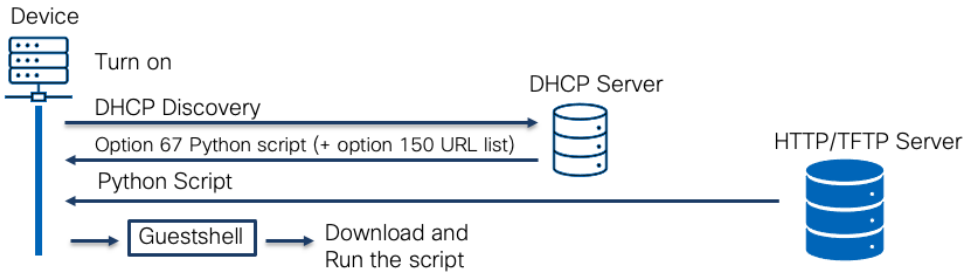
   Devices onboard using OV with its serial number

RFC8572 Secure ZTP

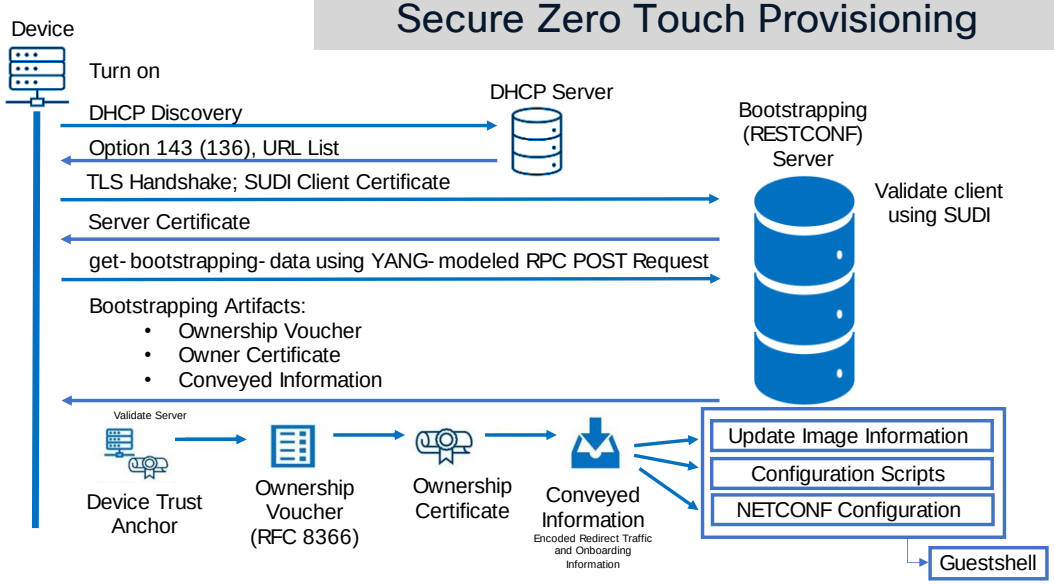
RFC details: https://www.rfc-editor.org/rfc/rfc8572.html

1. Conveyed Information: used to encode the redirect information and onboarding information (switch config)
2. Ownership Certificate: used by a device to verify the signature over the conveyed information
3. Ownership Voucher: used to verify a device owner as defined by the manufacturer (from the MASA)

Classic Zero Touch Provisioning



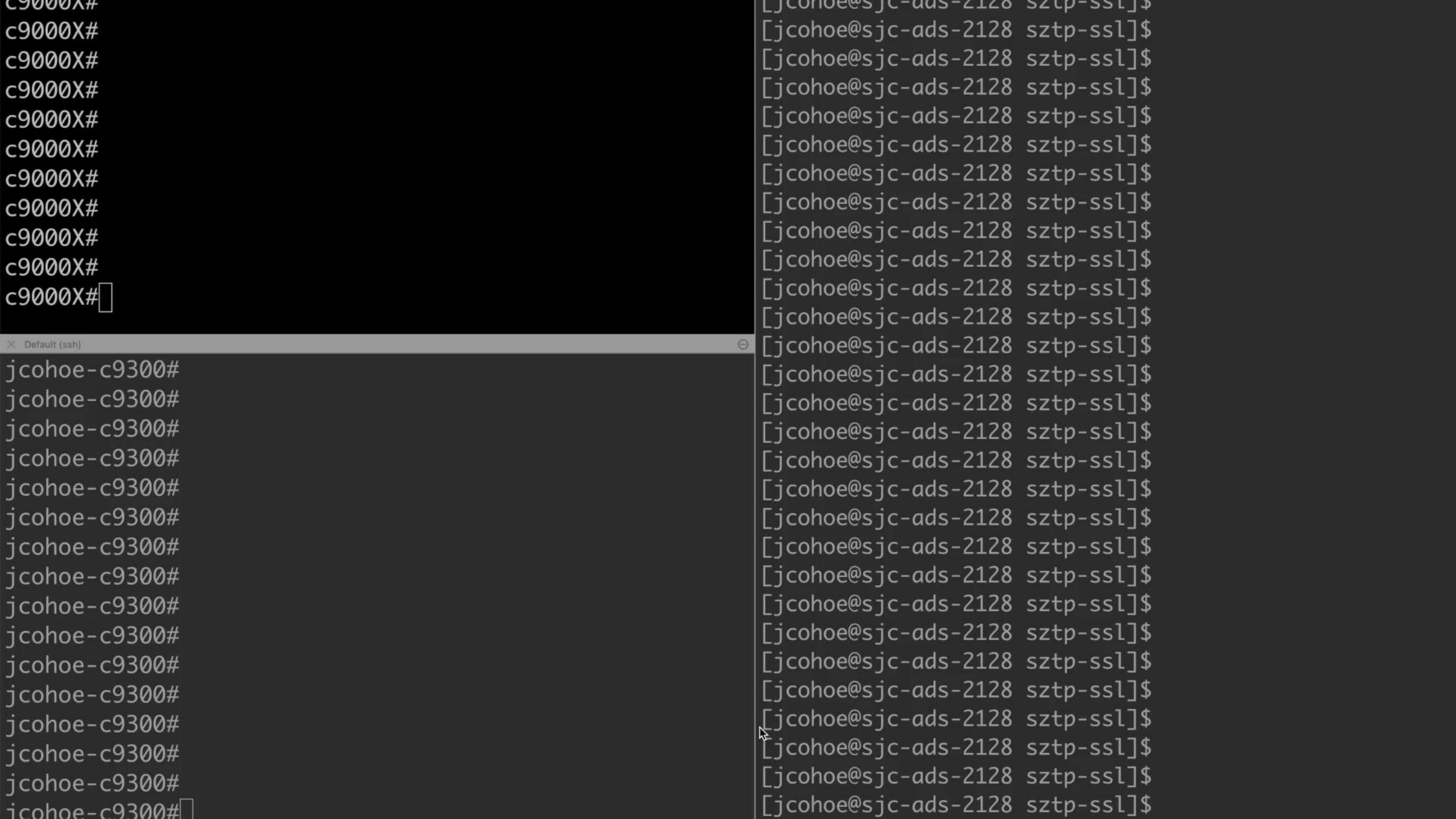
Secure Zero Touch Provisioning



Some security requirements for classic ZTP are resolved using Secure ZTP:

- Management system needs to validate the device
- Device needs to validate the server
- Device must validate the data is what server sent

As part of the SZTP RFC, the device supports image upgrade as part of the conveyed information



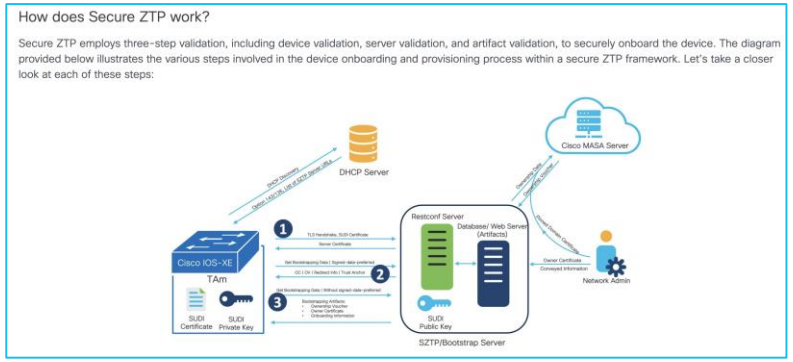
```
c9000X#
c9000X#
c9000X#
c9000X#
c9000X#
c9000X#
c9000X#
c9000X#
c9000X#
c9000X#
c9000X#
```

```
jcohoe-c9300#
jcohoe-c9300#
jcohoe-c9300#
jcohoe-c9300#
jcohoe-c9300#
jcohoe-c9300#
jcohoe-c9300#
jcohoe-c9300#
jcohoe-c9300#
jcohoe-c9300#
jcohoe-c9300#
jcohoe-c9300#
jcohoe-c9300#
jcohoe-c9300#
jcohoe-c9300#
jcohoe-c9300#
```

```
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
[jcohoe@sjc-ads-2128 sztp-ssl]$
```

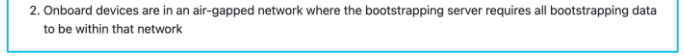
Learn more about Secure Zero Touch Provisioning

SZTP Blog



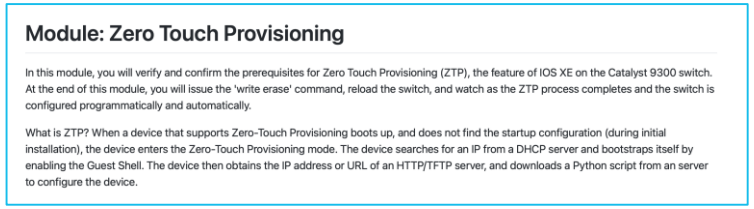
https://blogs.cisco.com/developer/secureztp01

Cisco SZTP Guide



https://github.com/sdeweese/sztp

ZTP guide for hands-on lab

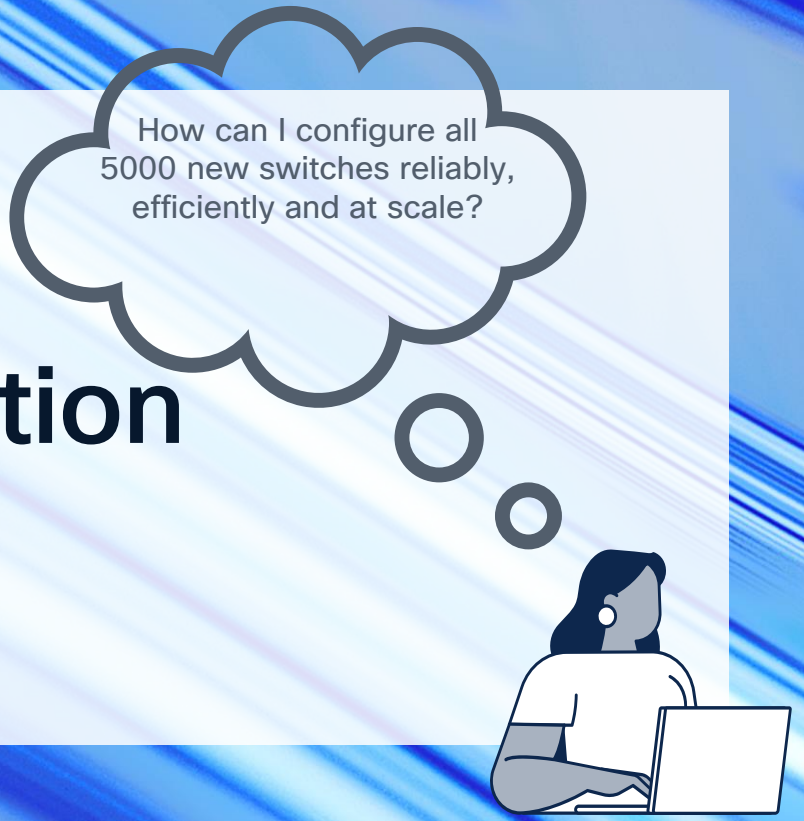


https://github.com/sdeweese/CLUS22-LTROPS-1836-programmabilty-and-automation/blob/main/ZTP2.md

ZTP Script repository

https://github.com/jeremycohoe/IOSXE-Zero-Touch-Provisioning

Device Configuration



Why It’s Time for a Change?

- The Complexity and Struggles of CLI-Based Automation

Creating Configurations That Just Work

Error Handling & Rollback

Scalability & Operational Efficiency

Why It’s Time for a Change?

Creating Configurations That Just Work

1. Declarative Config Management

Terraform

2. Two-Phase Commit

Verify before apply

Error Handling & Rollback

1. Atomic Config Replace

Transactional Config Change

2. Multiple Rollback Options

Instant Rollback
Post Deployment Rollback

Scalability & Operational Efficiency

1. Programmable Interfaces

NETCONF RESTCONF gNMI gRPC OPENCONFIG

APIs & Yang Models

2. Infrastructure as Code

NetDevOps

IOS XE programmability and Telemetry “Stack”

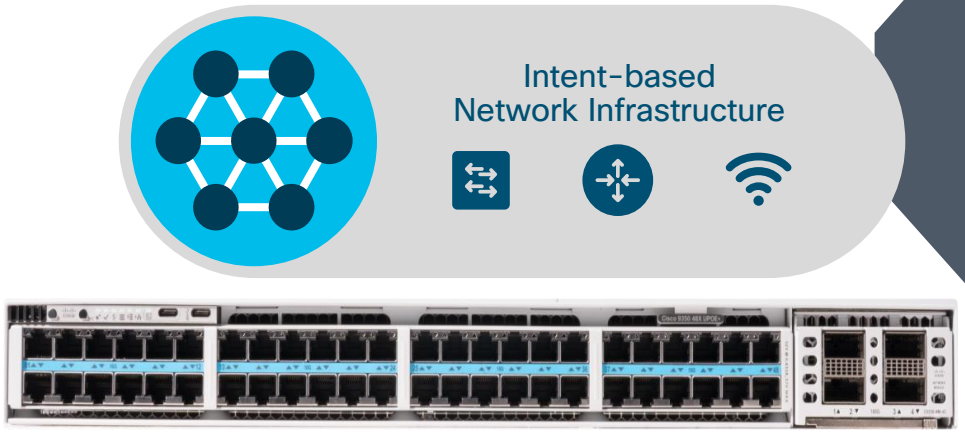
CLI

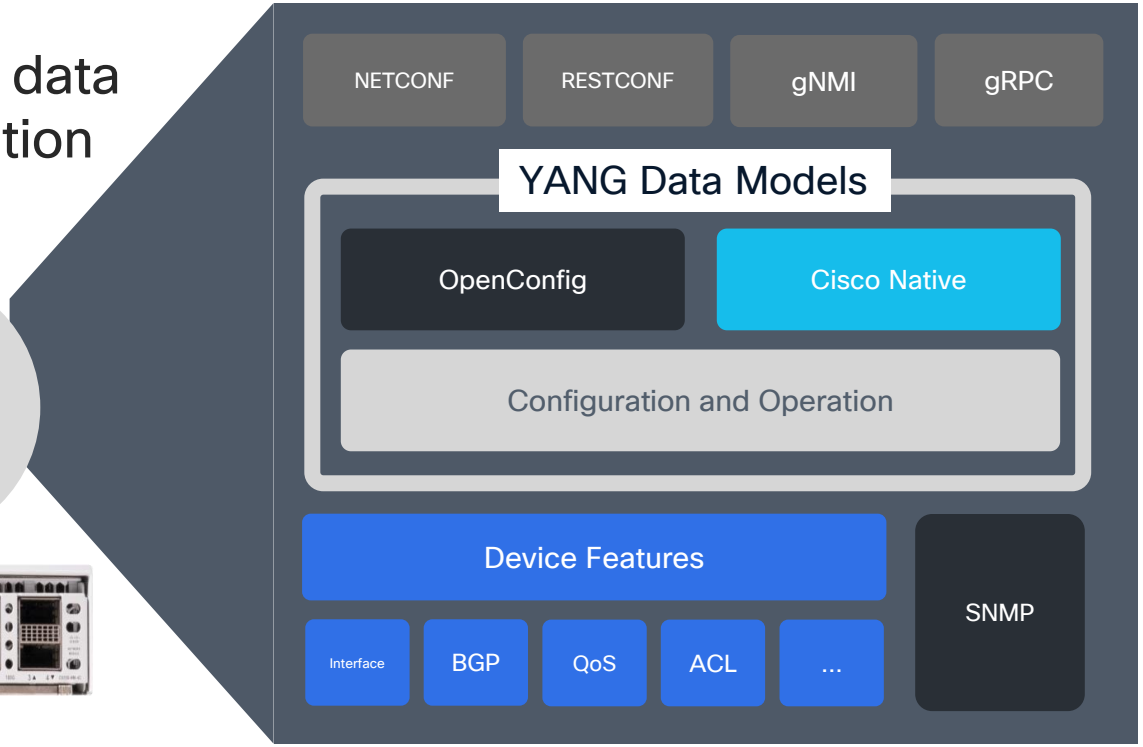
SNMP

WebUI

The NETCONF, RESTCONF, gNMI and gRPC 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, gNMI, and gRPC.

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





NETCONF

NETCONF over SSH

- NETCONF uses SSH port 830 but is <u>independent</u> from the SSH service
- ACL can ben applied for this interface as necessary

- Laptop
- Client
- Workstation
- Orchistrator
- Etc

Client connects to NETCONF SSH sub-system →

← Server responds with Hello that includes NETCONF supported capabilities

Client responds with supported capabilities →

Client issues NETCONF request (rpc/operation/content) →

← Server issues response / performs operation

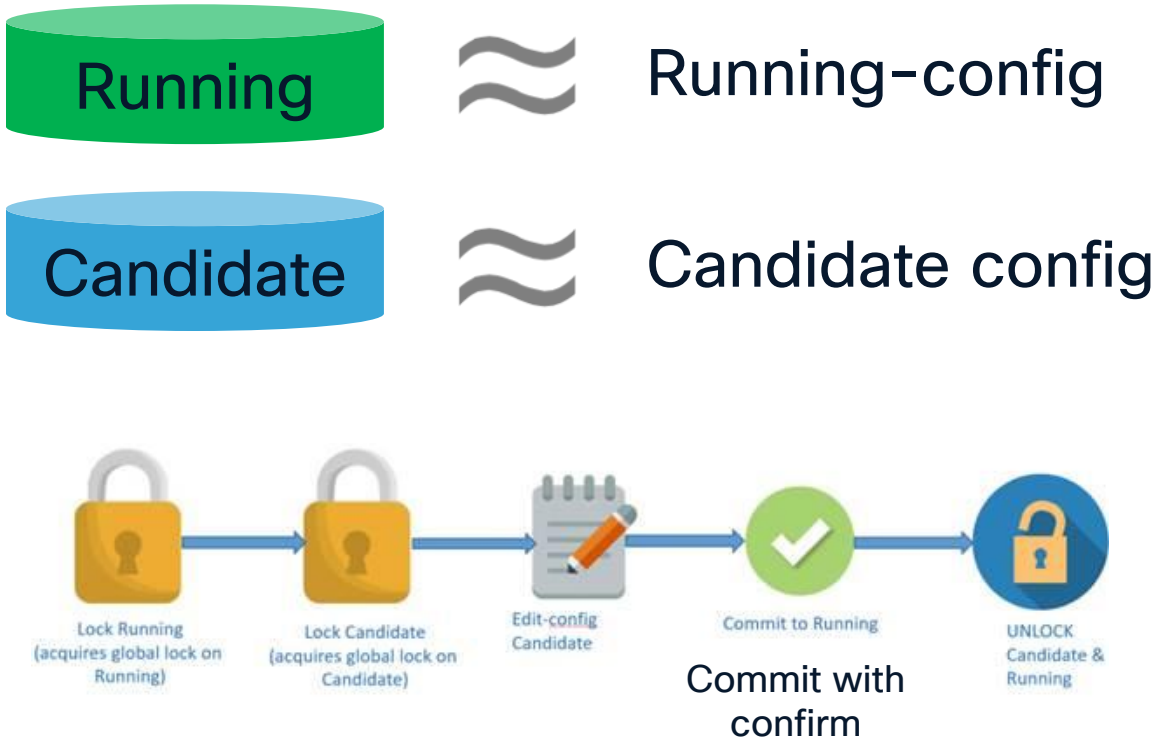
IOS XE
NETCONF/SSH
port 830

IOS XE NETCONF Datastores

"A Datastore holds a copy of the configuration data that is required to get a device from its initial default state into a desired operational state"

Running is the default and only mandatory Datastore

The Candidate Configuration feature enables support for candidate capability by implementing RFC 6241 with a simple commit option.
The candidate datastore provides a temporary workspace in which a copy of the device's running configuration is stored.
The candidate configuration supports the confirmed commit capability



https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1717/b_1717_programmability_cg/m_1717_prog_yang_netconf.html#id_78218

How to enable NETCONF ?

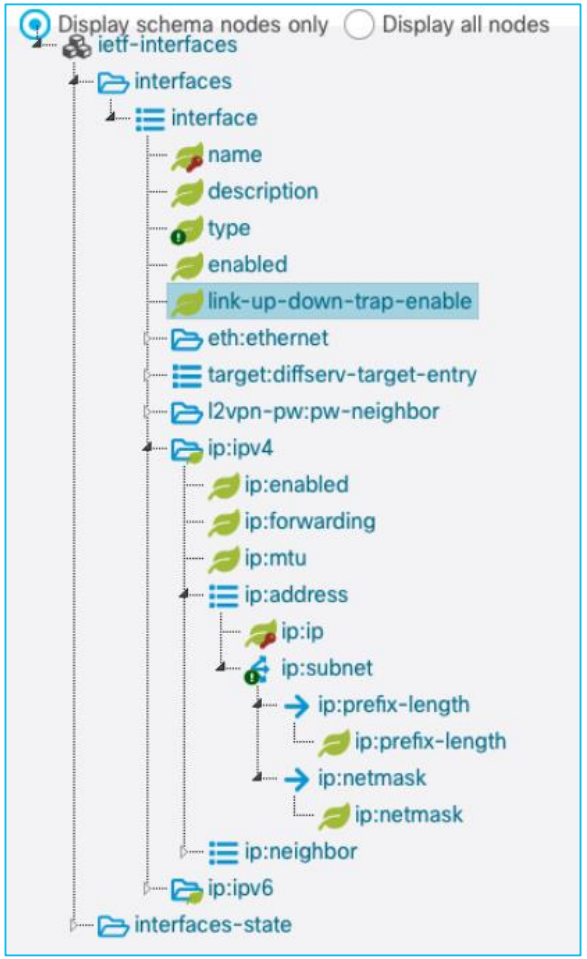
```
Cat9k-1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Cat9k-1(config)#aaa new-model
Cat9k-1(config)#aaa authentication login default local
Cat9k-1(config)#aaa authorization exec default local
Cat9k-1(config)#username admin privilege 15 password cisco

Cat9k-1(config)#netconf-yang
Cat9k-1(config)#
```

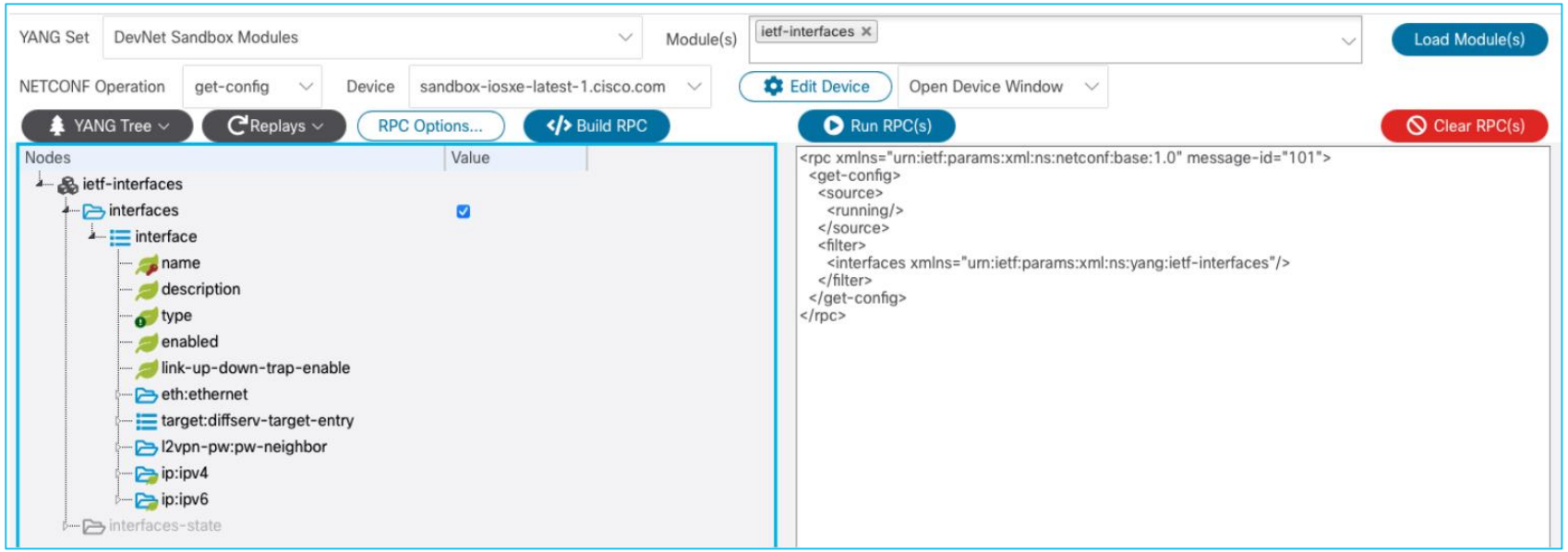
Enable AAA

Enable NETCONF

Example NETCONF GET IETF-Interfaces

IETF-interfaces.YANG GET-config operation





Demo: Generate Python from NETCONF

Access YANG Suite > Protocols > NETCONF

Select YANG model then Build RPC payload

Select Replays

Select Generate Python script

Cisco YANG Suite

Admin
Setup
Explore
Protocols
Help

YANG Suite / NETCONF / YANG set "c9300-default-yangset" / Modules

NETCONF

admin

YANG Set: c9300-default-yangset
Module(s): Cisco-IOS-XE-native
Load Module(s)

NETCONF Operation: edit-config
Device: C9300
Edit Device
Open Device Window

YANG Tree | Replays | RPC Options | Build RPC | Run RPC(s) | Clear RPC(s)

Load Replay...
Save as New Replay...
Edit Replay Information...
Save Changes
Generate Python script

Nodes | Value | Operation

- ipv6
- vlan
  - ios-vlan:access-log
  - ios-vlan:group
  - ios-vlan:vlan-list
    - ios-vlan:id — 28 — create
    - ios-vlan:remote-span
    - ios-vlan:private-vlan
    - ios-vlan:name
    - ios-vlan:state
    - ios-vlan:lldp
    - ios-vlan:uni-vlan
    - ios-vlan:shutdown
  - ios-vlan:device-tracking
- mvrp
- avb
- ptp
- cdp
- avc
- policy
- interface
- route-map
- route-tag
- table-map

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
        <vlan>
          <vlan-
            <id>2
          </vlan
        </vlan>
      </native>
    </config>
  </edit-confi
</rpc>
```

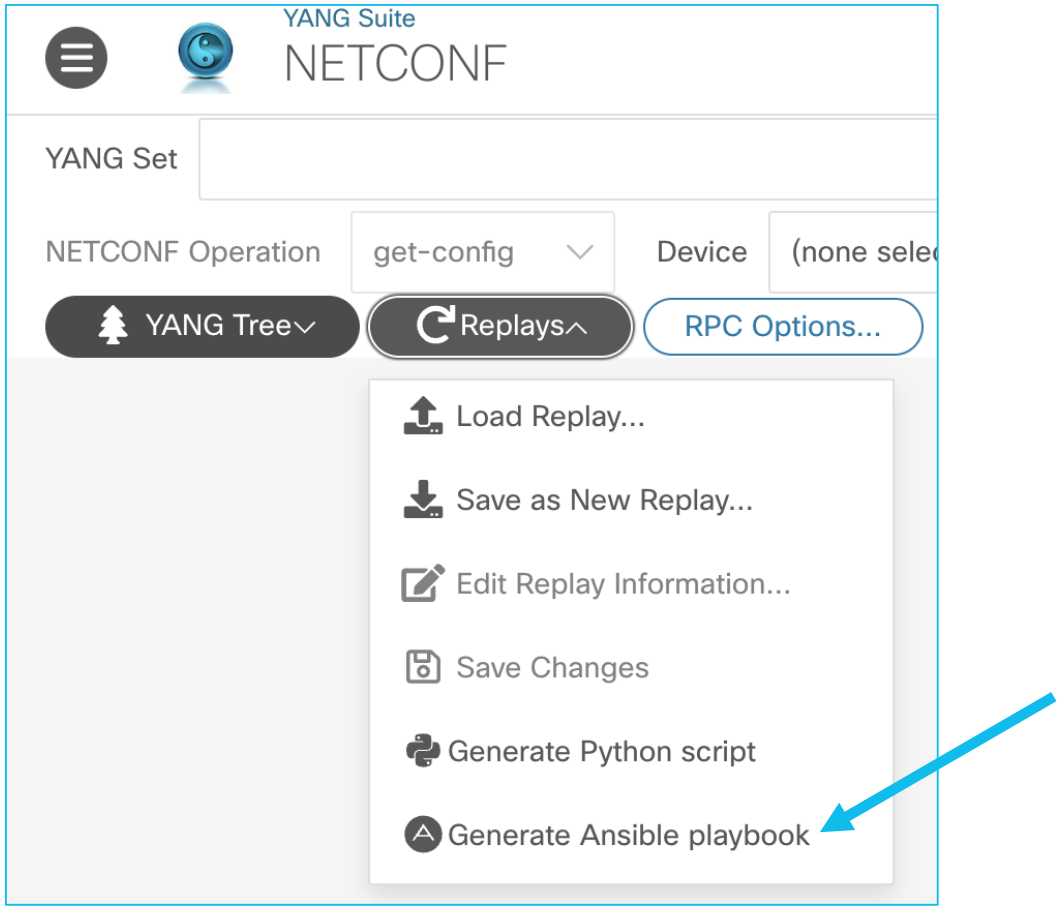
```
#! /usr/bin/env python
import lxml.etree as et
from argparse import ArgumentParser
from ncclient import manager
from ncclient.operations import RPCError

payload = [
'''
<get xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <filter>
    <lag-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-lacp-o
  </filter>
</get>
''',
]

if __name__ == '__main__':

    parser = ArgumentParser(description='Usage:')

    # script arguments
    parser.add_argument('-a', '--host', type=str, required=True,
                        help="Device IP address or Hostname")
    parser.add_argument('-u', '--username', type=str, required=True,
                        help="Device Username (netconf agent username)
    parser.add_argument('-p', '--password', type=str, required=True,
                        help="Device Password (netconf agent password)
    parser.add_argument('--port', type=int, default=830,
                        help="Netconf agent port")
    args = parser.parse_args()
```

NETCONF + Ansible

Similar to the “Generate Python” functionality, a new functionality generates YAML formatted for Ansible.

Requirements
1. Install Ansible
2. Install NETCONF collection:
   ansible-galaxy collection install NETCONF

```
- name : conf-host
  hosts: c9300
  connection: netconf
  gather_facts: no

  tasks:
    - name: hostname-conf
      netconf_config:
        xml: |
          <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
            <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
              <hostname>c9300-pod29</hostname>
            </native>
          </config>
```



NETCONF + Ansible Update Interface Description Demo

Cisco YANG Suite

YANG Suite / Help

Help: Welcome to Cisco YANG Suite!

admin

- Admin
- Setup
- Analytics
- Explore
- Protocols
- Test Manager
- Help

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 NETCONF (RFC 6241).

It provides a modular infrastructure which various YANG application plugins can be used.

Please check developer.cisco.com/yangsuite for information, learning labs, and announcements.

Docker-based Installation

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

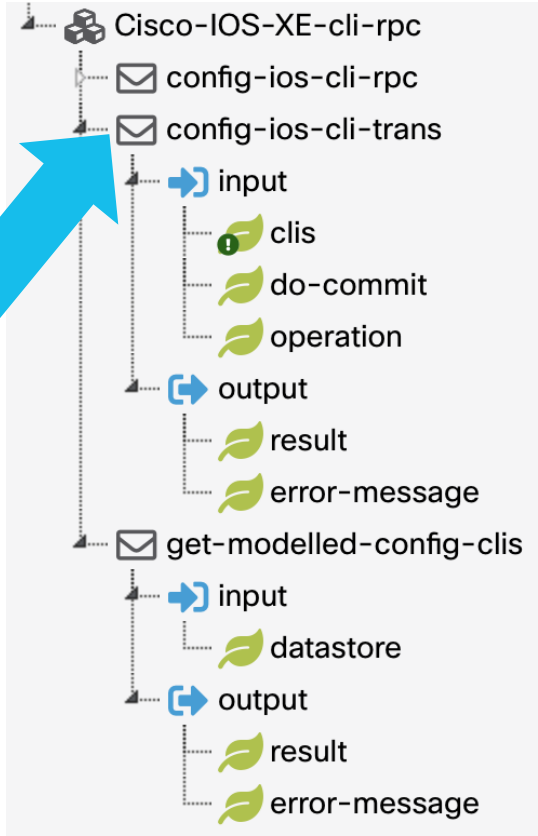
- Prompts for username, password, and email which will be the superuser to yangsuite.
- Gives the choice of creating test SSL/TLS certificate and key.
- Creates an environment file needed for the yangsuite docker container install.
- Runs docker-compose up.

YANG Suite Documentation

Search docs

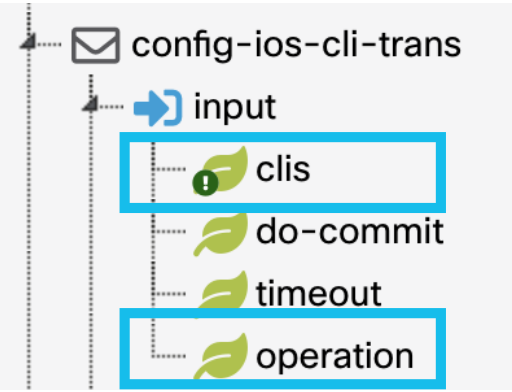
Cisco-IOS-XE-CLI-RPC.YANG

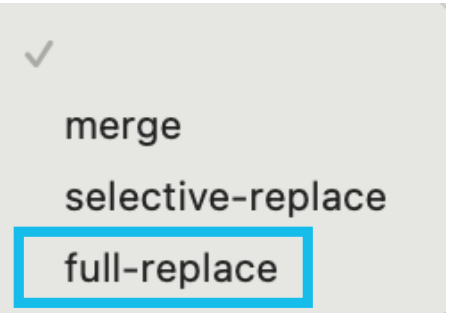
This YANG data model allows sending CLI through the YANG API interfaces
Previously only YANG modelled data was supported



Name	config-ios-cli-trans
Nodetype	rpc
Description	Configure using modelled config CLI. Wireless, app-hosting, telemetry features are not supported through this RPC.
Module	Cisco-IOS-XE-cli-rpc
Revision	2023-11-01
Xpath	/config-ios-cli-trans
Prefix	cli-ios-xe-rpc
Namespace	http://cisco.com/ns/yang/Cisco-IOS-XE-cli-rpc
Schema Node Id	/config-ios-cli-trans
Access	write
Operations	• "rpc"

Name	get-modelled-config-clis
Nodetype	rpc
Description	Retrieve configuration in CLI format from Running or Candidate datastore. Wireless, app-hosting, telemetry features are not supported through this RPC.
Module	Cisco-IOS-XE-cli-rpc
Revision	2023-11-01
Xpath	/get-modelled-config-clis
Prefix	cli-ios-xe-rpc
Namespace	http://cisco.com/ns/yang/Cisco-IOS-XE-cli-rpc
Schema Node Id	/get-modelled-config-clis
Access	write
Operations	• "rpc"

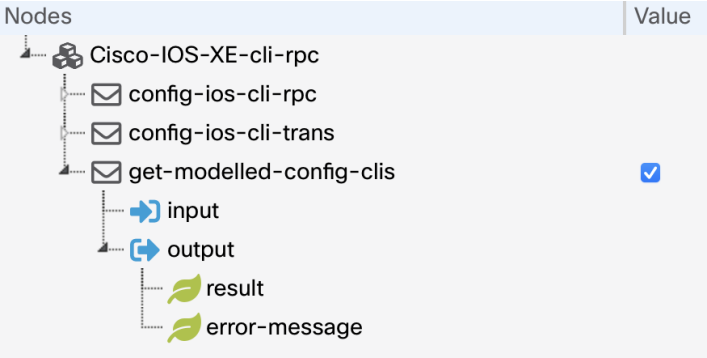




https://github.com/YangModels/yang/blob/main/vendor/cisco/xe/1791/Cisco-IOS-XE-cli-rpc.yang

Get Modelled Config CLI RPC

- Sending the “get-modelled-config-clis” RPC returns the modelled running-config in CLI format
- Anything not modelled will not be returned (AppH)
- Unsupported model config will be ignored (AppH)
- This is used as the template to update the device with after being modified as needed




```
Sending:
#246
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:afff
  <get-modelled-config-clis xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-cli-rpc"/>
</nc:rpc>


##
```

```
Received message from host
```

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params
<result xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-cli-rpc">version 17.14
memory free low-watermark processor 130582
service timestamps debug datetime msec
service timestamps log datetime msec
service call-home
no service tcp-small-servers
no service udp-small-servers
hostname JCOHOE-C9300-2
control-plane
 service-policy input system-cpp-policy
!
clock summer-time PDST recurring
clock timezone pacific -8 0
login on-success log
license boot level network-advantage addon dna-advantage
transceiver type all
 monitoring
!
iox
call-home
 contact-email-addr sch-smart-licensing@cisco.com
 profile CiscoTAC-1
  active
  destination transport-method http
```

RPC:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <get-modelled-config-clis xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-cli-rpc"/>
</rpc>
```

Enable IPv6 on IOS XE using YANG models

Automate with **Python**

Configuration payload

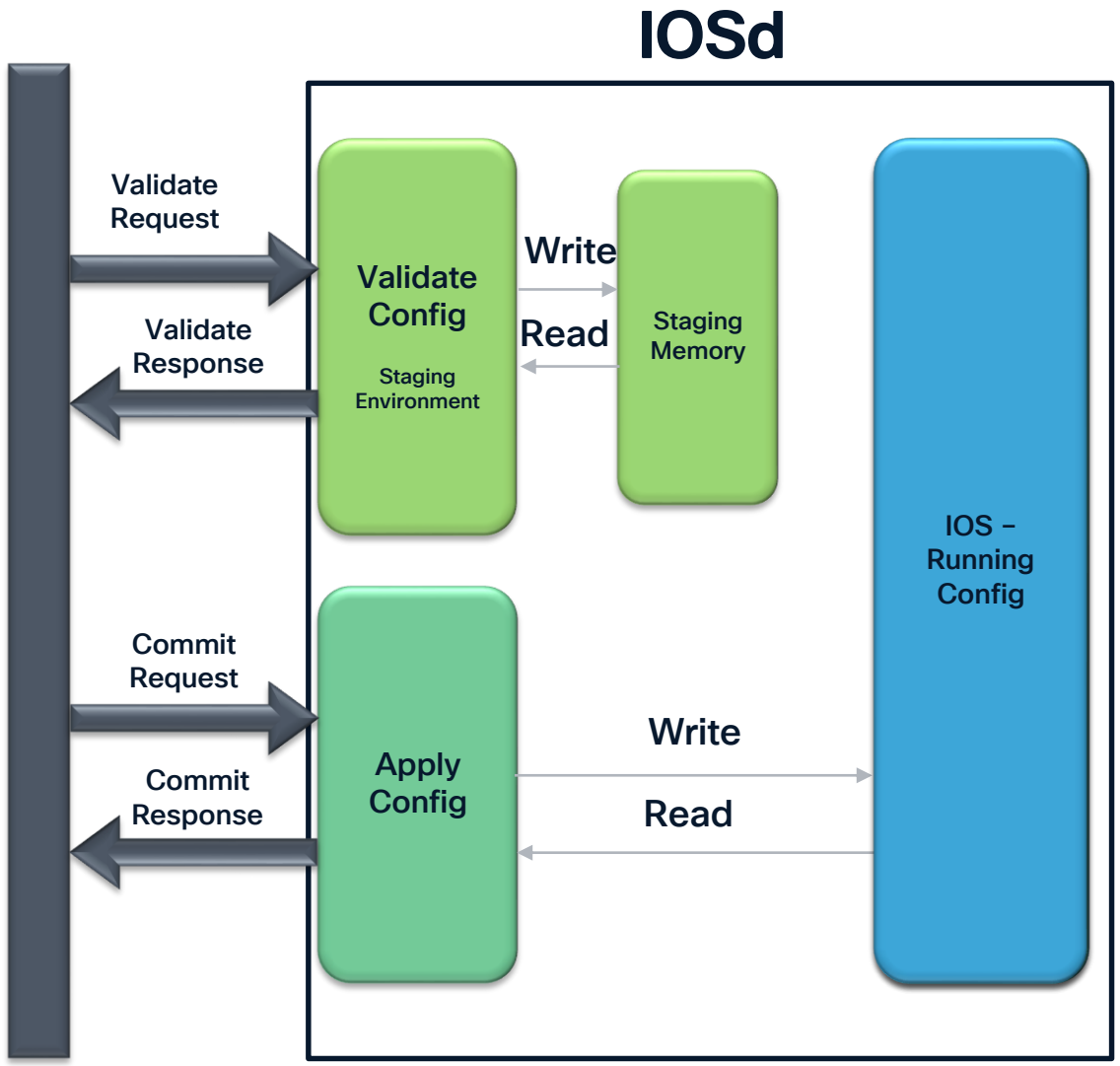
```
<native
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
  <ipv6>
    <unicast-routing/>
    <router>
      <ospf>
        <id>1</id>
      </ospf>
    </router>
  </ipv6>
</native>
```

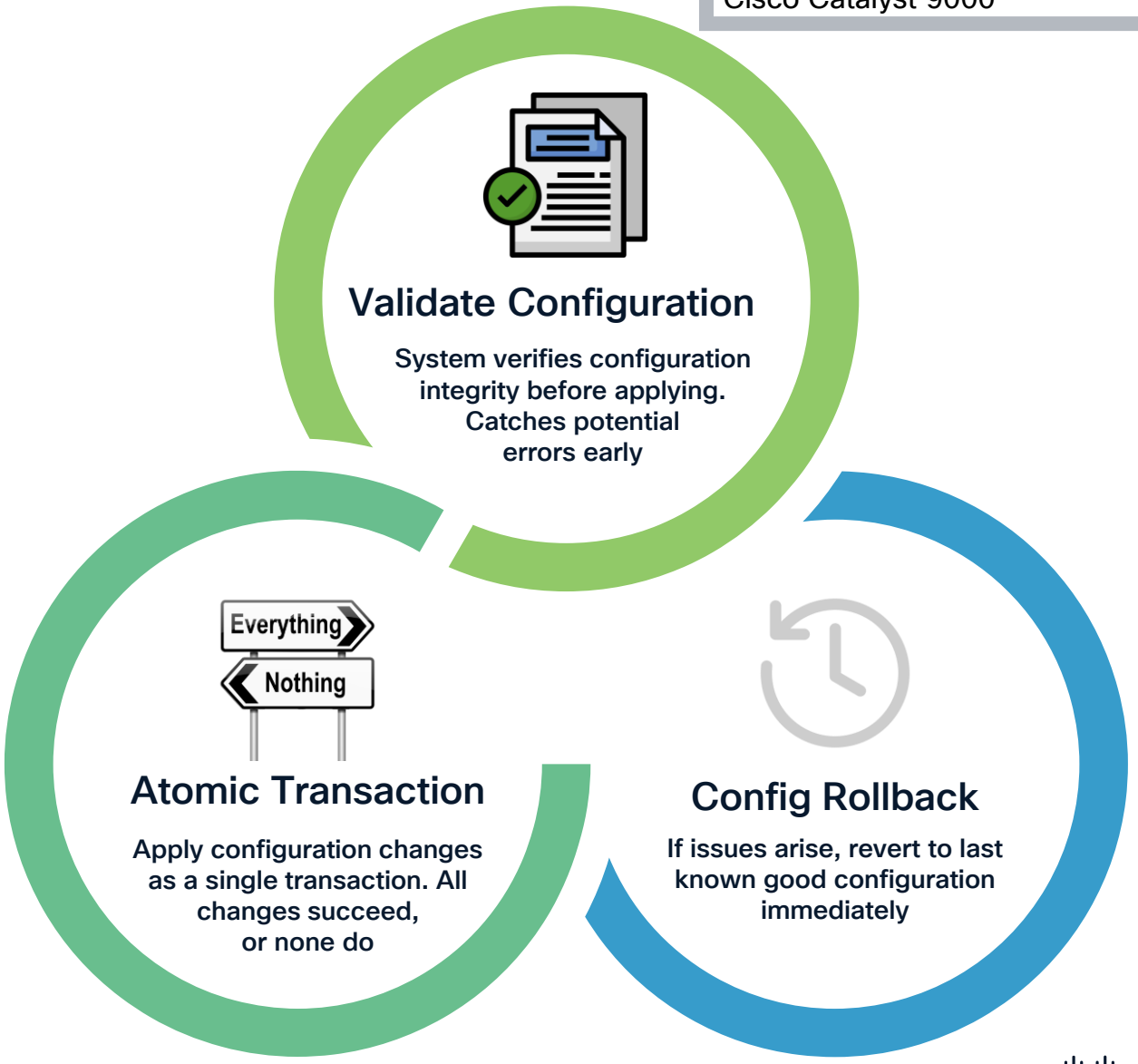
```python
from ncclient import manager

DEVICE = {
    "host": "198.18.11.2",
    "username": "developer",
    "password": "C1sco12345",
    "hostkey_verify": False
}

payload = """
    <config>
        <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
            <ipv6>
                <unicast-routing/>
                <router>
                    <ospf><id>1</id></ospf>
                </router>
            </ipv6>
        </native>
    </config>
"""

with manager.connect(**DEVICE, device_params={"name":"iosxe"}) as connection:

    response = connection.edit_config(target="running", config=payload)
```

Send to device over a NETCONF session

Introducing Atomic Configuration Replace

Watch the recording:
BRKENS-2604
Atomic Config Replace with Cisco Catalyst 9000

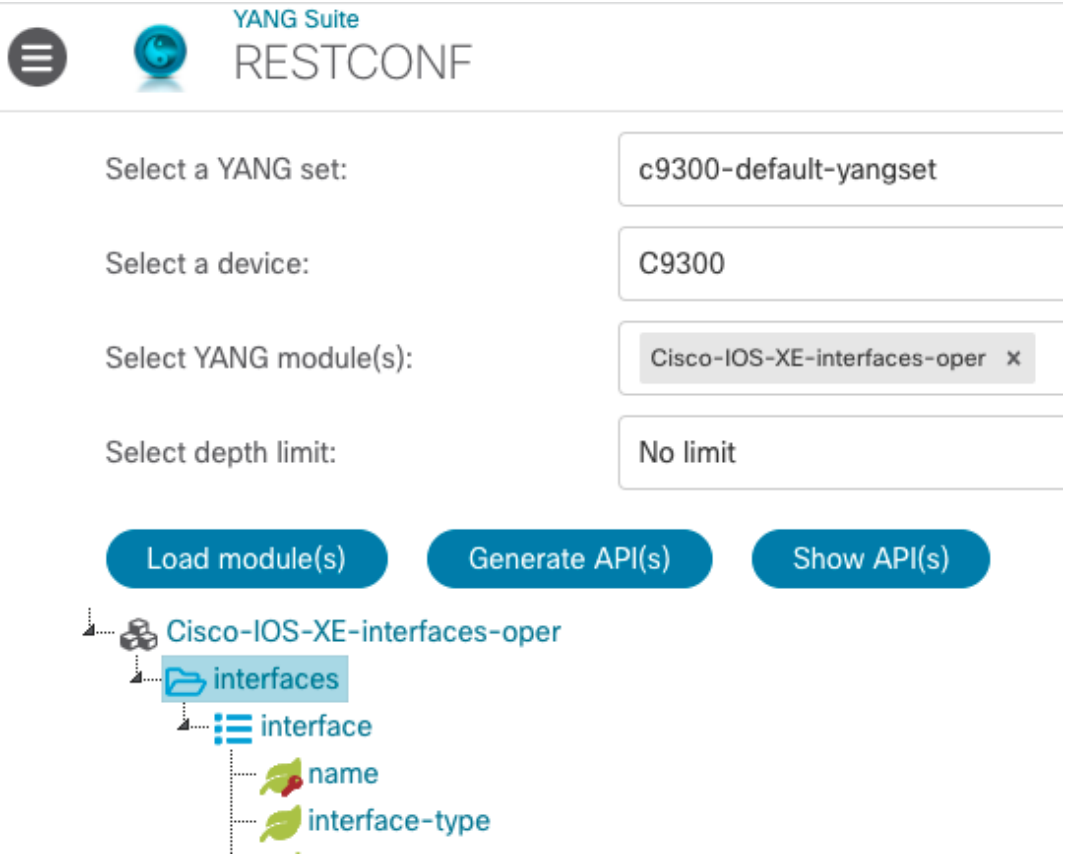




RESTCONF

YANG Suite RESTCONF

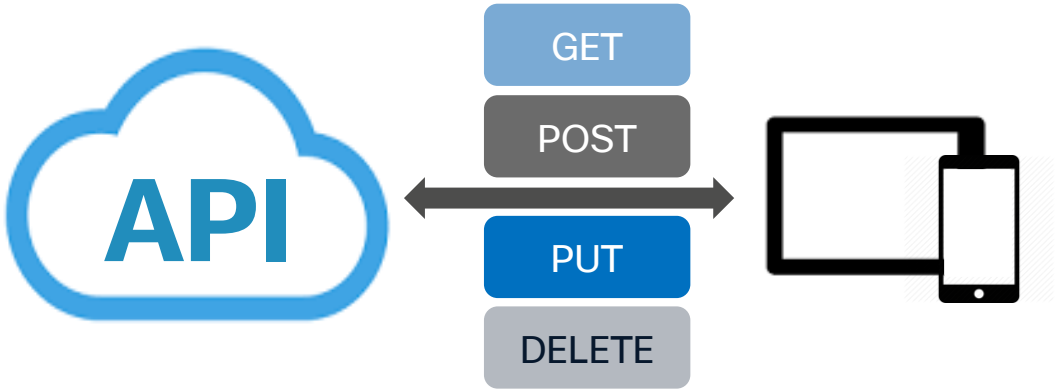
RESTCONF provides a programmatic interface based on standard mechanisms for accessing configuration data, state data, data-model-specific Remote Procedure Call (RPC) operations and events, defined in the YANG model. The YANG Suite RESTCONF plugin provides Swagger UI and execution visualization of the YANG data model.



YANG Suite RETSCONF replaces the need for POSTMAN, which doesn't have integration with YANG models

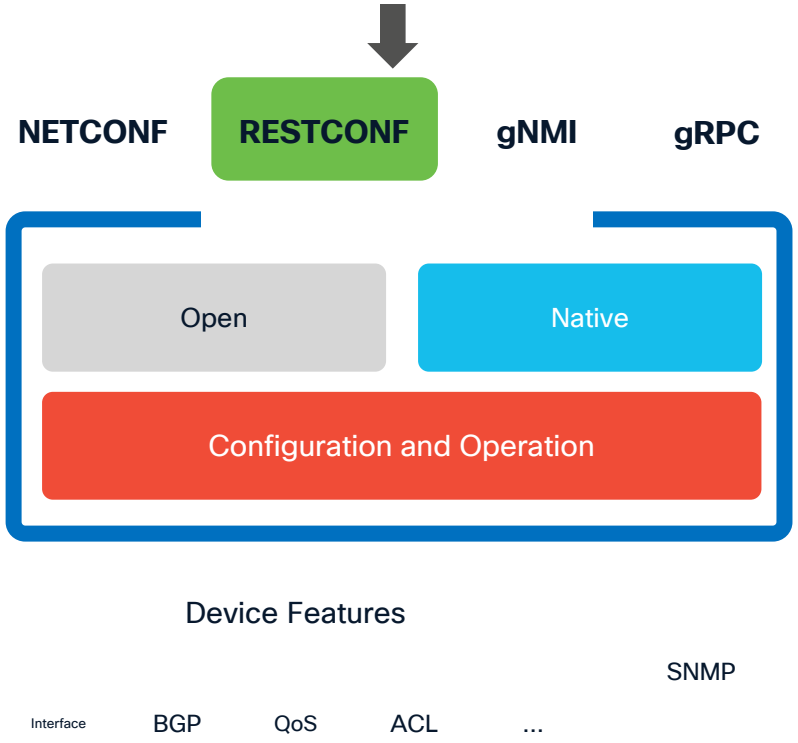
REST vs RESTCONF: not the same!

REST



*"A **framework** for **client-server** communications"*

RESTCONF



*"**REST-like protocol** for accessing **YANG** models"*

Enable RESTCONF

```
Cat9k-1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.

Cat9k-1(config)#restconf

Cat9k-1(config)#ip http secure-server
```

Enable RESTCONF

Enable HTTP server

YANG Suite RESTCONF Demo

Cisco YANG Suite

- Admin
- Setup
- Analytics
- Explore
- Protocols
- Test Manager
- Help

YANG Suite / Help

Help: Welcome to Cisco YANG Suite!

admin

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 NETCONF (RFC 6241).

It provides a modular infrastructure which various YANG application plugins can be used.

Please check developer.cisco.com/yangsuite for information, learning labs, and announcements.

Docker-based Installation

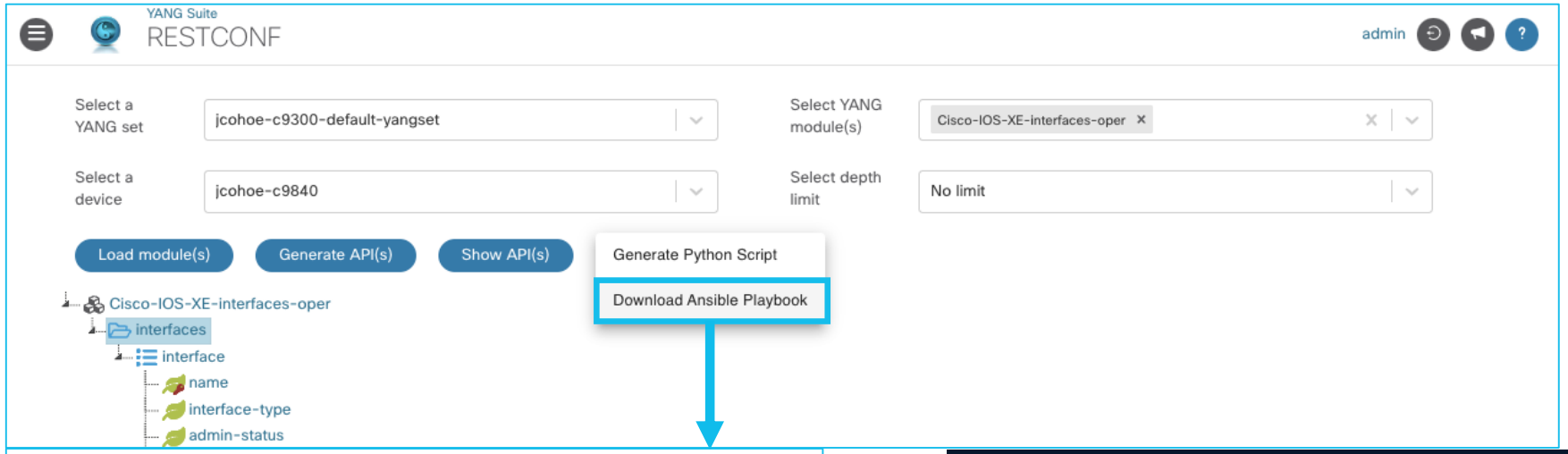
The yangsuite/docker/start_yang_suite.sh script performs the following:

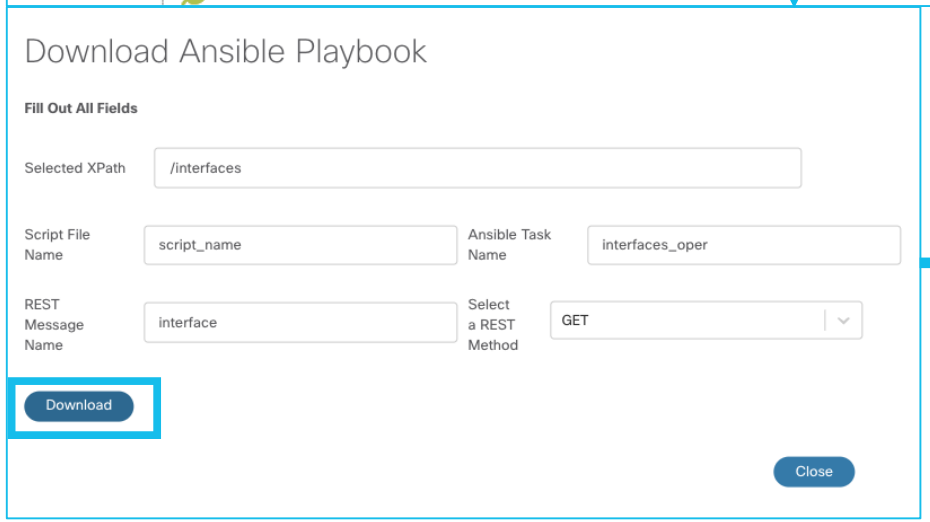
- Prompts for username, password, and email which will be the superuser to yangsuite.
- Gives the choice of creating test SSL/TLS certificate and key.
- Creates an environment file needed for the yangsuite docker container install.
- Runs docker-compose up.

YANG Suite Documentation

Search docs

RESTCONF + Ansible





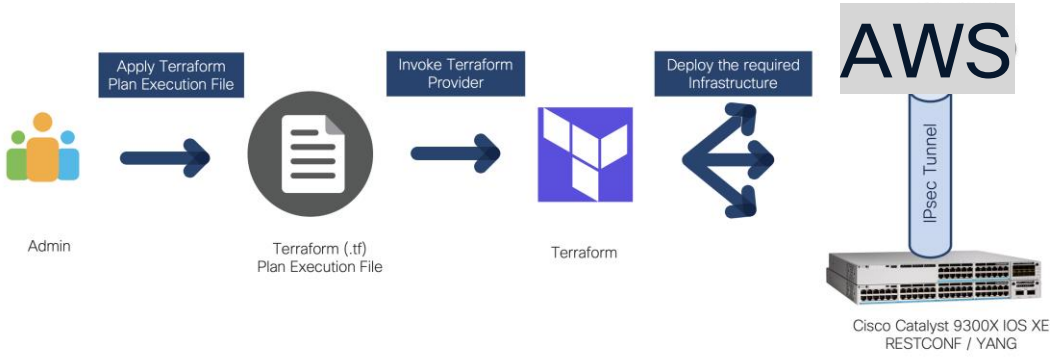

```
 - name: interface
hosts: HOST_NAME_HERE
gather_facts: no
tasks:
- name: interfaces_oper
ansible.netcommon.restconf_get:
# Output can either be json or xml
output: json
path: Cisco-IOS-XE-interfaces-oper:interfaces
```

Requirements

1. Install Ansible
2. Install the required collection: ansible-galaxy collection install ansible.netcommon

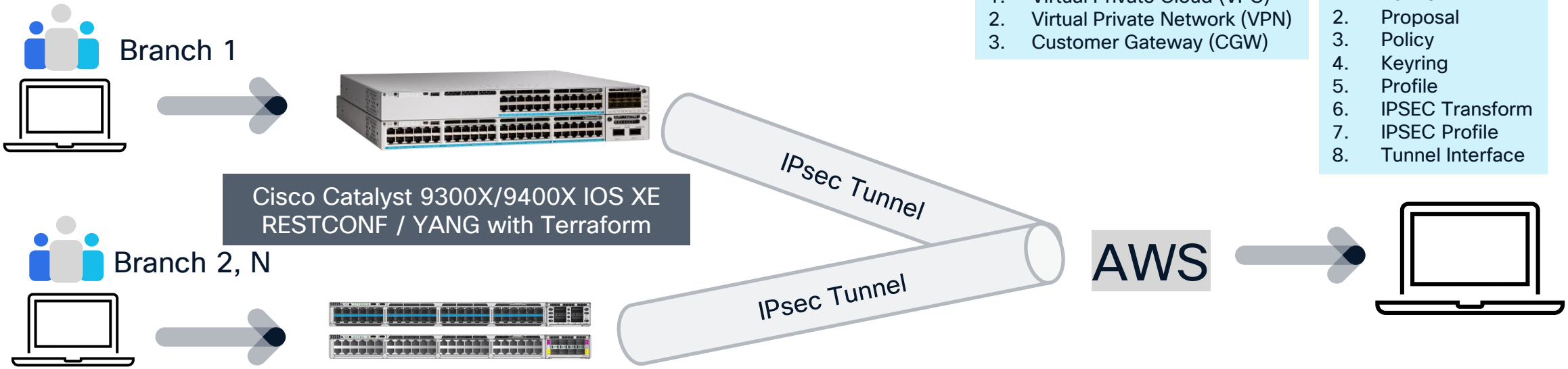
IPsec tunnel + cloud automation with Terraform

1. Terraform configures the IPsec tunnel between the 9300X and the cloud service where the internal resources are available
2. Terraform also manages the cloud-native resources including certificate key management and IP subnetting
3. Connections between VPC, VPN, CGW and device certificates, tunnels, and interfaces are created



1. Virtual Private Cloud (VPC)
2. Virtual Private Network (VPN)
3. Customer Gateway (CGW)

1. Tunnel
2. Proposal
3. Policy
4. Keyring
5. Profile
6. IPSEC Transform
7. IPSEC Profile
8. Tunnel Interface



Terraform + Crypto IPsec Demo

terraform.tf | variables.tf | 9300X.tfvars | crypto.txt

sevt > terraform.tf > ...

```
# See preview using: terraform plan -var-file="9300X.tfvars"
# Run using: terraform apply -var-file="9300X.tfvars" -auto-approve

terraform {
  required_providers {
    iosxe = {
      source  = "CiscoDevNet/iosxe"
      version = "0.1.1"
    }
  }
}

provider "iosxe" { # variables initialized in variables.tf and values stored in 9300X.tfvars
  host            = var.host_url
  insecure        = var.insecure
  device_username = var.device_username
  device_password = var.device_password
}

# crypto all
resource "iosxe_rest" "crypto_example_post" {
  method = "PATCH"
  path = "/data/Cisco-IOS-XE-native:native/crypto"
  payload = jsonencode(

{
    "Cisco-IOS-XE-native:crypto": {
      "Cisco-IOS-XE-crypto:ikev2": {
```

PROBLEMS 69 | OUTPUT | TERMINAL | DEBUG CONSOLE

```
sdeweese@SDEWEESE-M-KQC6 sevt %
sdeweese@SDEWEESE-M-KQC6 sevt %
sdeweese@SDEWEESE-M-KQC6 sevt %
sdeweese@SDEWEESE-M-KQC6 sevt %
sdeweese@SDEWEESE-M-KQC6 sevt %
sdeweese@SDEWEESE-M-KQC6 sevt %
sdeweese@SDEWEESE-M-KQC6 sevt %
sdeweese@SDEWEESE-M-KQC6 sevt %
sdeweese@SDEWEESE-M-KQC6 sevt %
sdeweese@SDEWEESE-M-KQC6 sevt %
sdeweese@SDEWEESE-M-KQC6 sevt %
sdeweese@SDEWEESE-M-KQC6 sevt %
sdeweese@SDEWEESE-M-KQC6 sevt %
sdeweese@SDEWEESE-M-KQC6 sevt %
sdeweese@SDEWEESE-M-KQC6 sevt %
sdeweese@SDEWEESE-M-KQC6 sevt %
sdeweese@SDEWEESE-M-KQC6 sevt %
```

```
9300X-Edge-2#
9300X-Edge-2#
9300X-Edge-2#
9300X-Edge-2#
9300X-Edge-2#
9300X-Edge-2#
9300X-Edge-2#
9300X-Edge-2#
9300X-Edge-2#
9300X-Edge-2#
9300X-Edge-2#
9300X-Edge-2#
9300X-Edge-2#
9300X-Edge-2#
9300X-Edge-2#
9300X-Edge-2#
```

gNMI

gRPC and gNMI

gRPC

Google
Remote Procedure Call

gNMI

gRPC
Network Management Interface

```
C9300#show run | sec telemetry
telemetry ietf subscription 101
 encoding encode-kvgpb
 filter xpath /process-cpu-ios-xe-oper:cpu-usage/cpu-
utilization/five-seconds
 source-address 10.1.1.5
 stream yang-push
 update-policy periodic 500
 receiver ip address 10.1.1.3 57500 protocol grpc-tls profile
myca
```

```
C9300#show run | i gnxi
gnxi
gnxi secure-trustpoint gnxi-tls-cert
gnxi secure-server
gnxi secure-port 9339

show gnxi state detail
```

Enable gNMI

```
Cat9k-1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.

Cat9k-1(config)# gnmi-yang
Cat9k-1(config)# gnmi-yang server

Cat9k-1(config)# gnmi-yang port 50052
```

Enable gNMI

Config server

Port

Enabling gNxI: secure-server

```
Cat9k-1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.

Cat9k-1(config)# gnxi secure-trustpoint <<trustpoint name>>
Cat9k-1(config)# gnxi secure-server
```

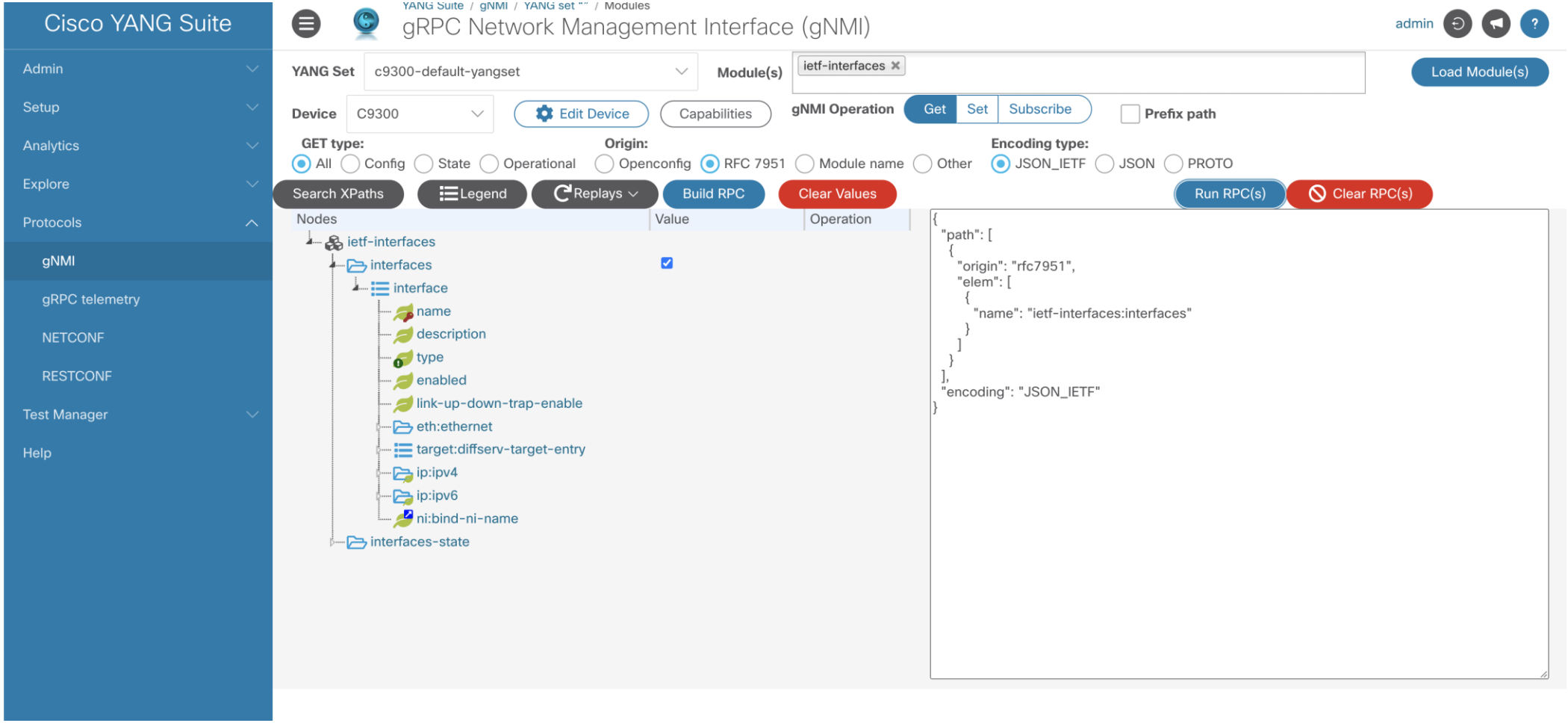
Create a trustpoint

Use trustpoint

```
Cat9k-1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.

Cat9k-1(config)# gnxi secure-init
```

Alias for above config

gNMI GET IETF Interfaces

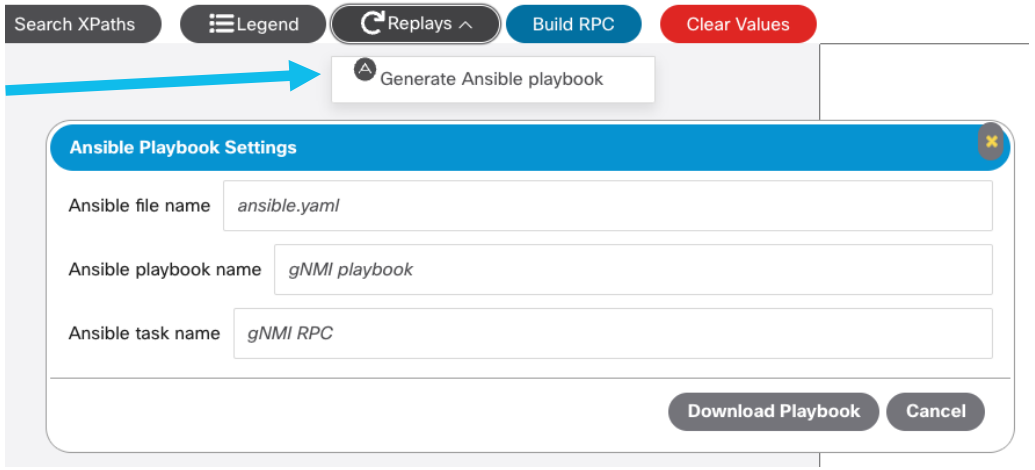


gNMI + Ansible

Requirements
1. Install Ansible
2. Install the required collection

ansible-galaxy collection install nokia.grpc

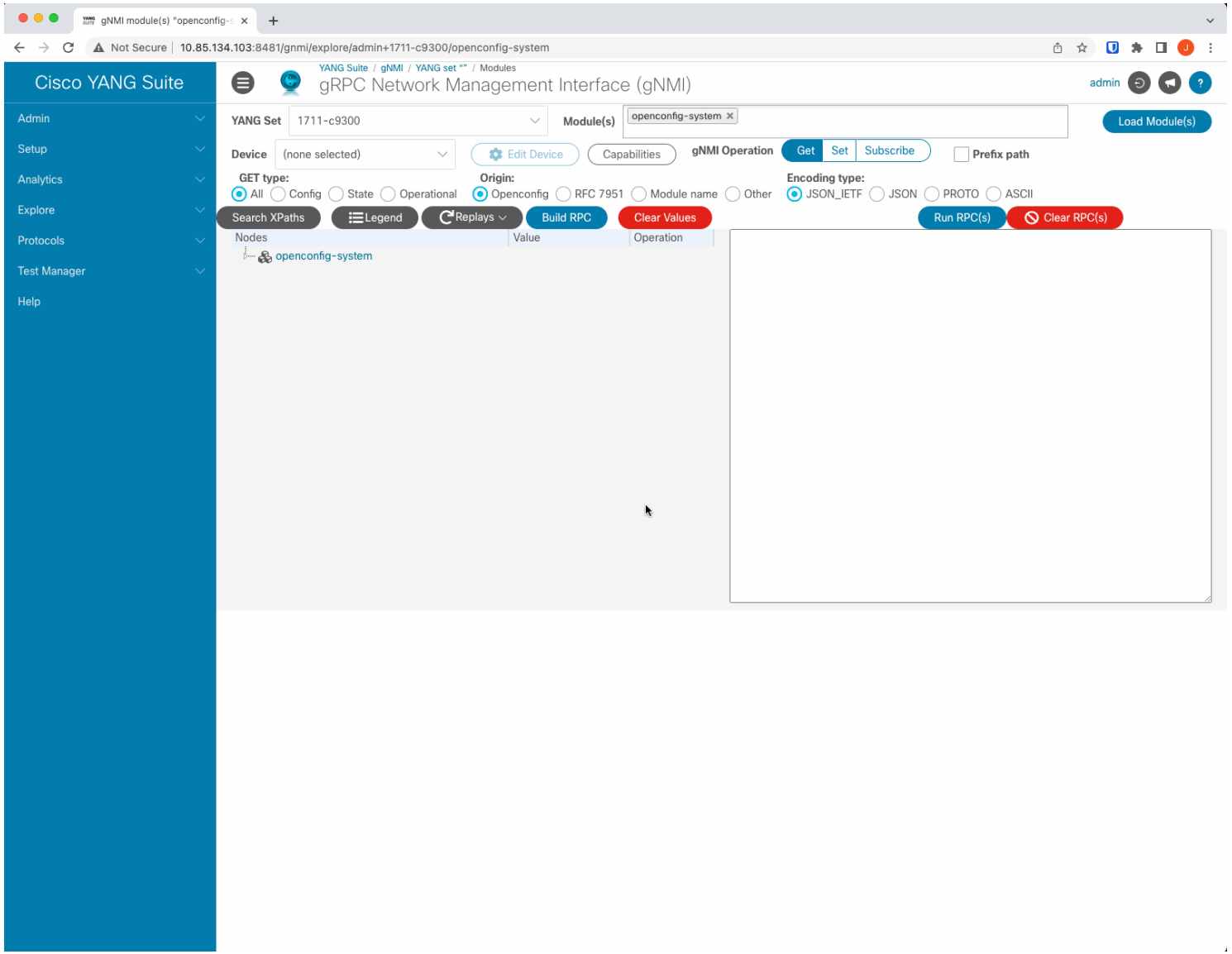
```
- name : gNMI playbook
  gather_facts: false
  hosts: MY_HOST_NAME

  collections:
  - nokia.grpc

  tasks:
  - name: gNMI RPC
    gnmi_config:
        update:
        - path: system/config/hostname
          val: set-by-ys-gnmi-ansible

    register: testout
  - name: dump test output
    debug:
        msg: '{{ testout.output }}'
```




```
auto@pod19-xelab:~/ansible/YANGSuite-ansible$ ansible-galaxy collection install nokia.grpc
Process install dependency map
Starting collection install process
Installing 'nokia.grpc:1.0.2' to '/home/auto/.ansible/collections/ansible_collections/nokia/grpc'
auto@pod19-xelab:~/ansible/YANGSuite-ansible$
```

Note: the ansible gnmi integration works only with OpenConfig model

gNMI + Ansible demo



Device Monitoring

Why It’s Time for a Change?

Challenges of SNMP Monitoring

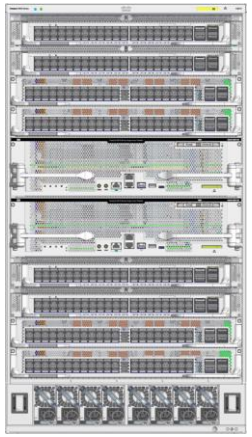
Lack of Real-Time and Granular Monitoring

Scalability and Performance Bottlenecks

Security Risks

Telemetry Data Flow

Cisco C9610-SUP-3 & 3XL
Powered by Cisco Silicon One **E100** & **K100**



Cisco C9350 Series
Powered by Cisco Silicon One **A100**

Cisco IOS XE Devices

Collector/Receiver
Decodes to text

telegraf™

Storage
Time Series Database

InfluxDB

Monitoring and Visualizations

Grafana

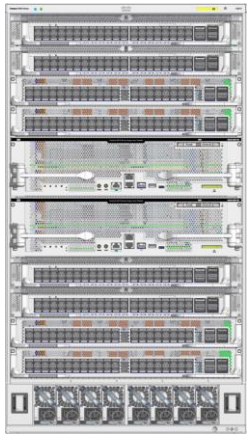
https://hub.docker.com/r/jeremycohoe/tig_mdt https://github.com/jeremycohoe/cisco-ios-xe-mdt
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1715/b_1715_programmability_cg/m_1715_prog_ietf_telemetry.html

Splunk integration

Cisco IOS XE Devices

Cisco C9350 Series
Powered by Cisco Silicon One A100

Cisco C9610-SUP-3 & 3XL
Powered by Cisco Silicon One E100 & K100



telegraf™

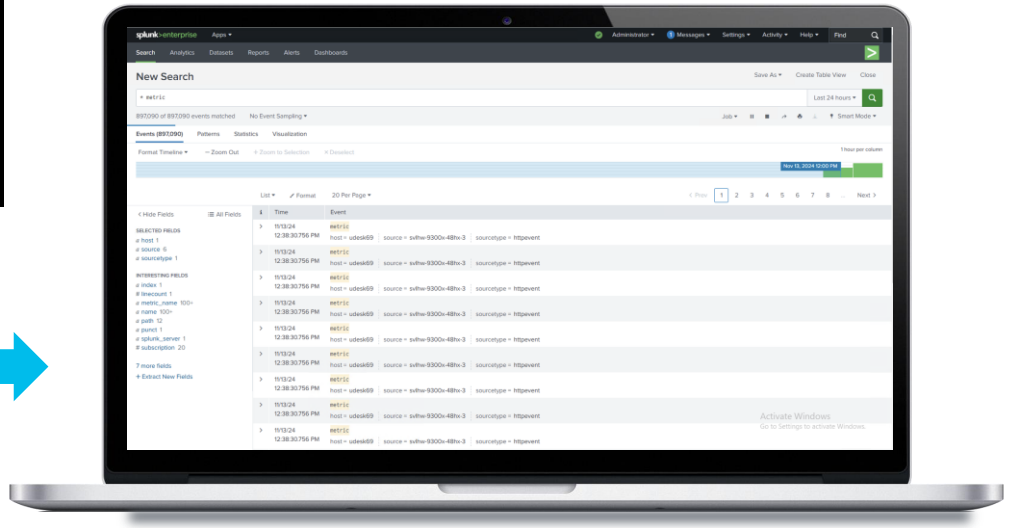
Collector/Receiver
Decodes to text

```
# Output to Splunk
[[outputs.http]]
 url = "https://172.26.202.150:8088/services/collector"
 insecure_skip_verify = true
 data_format = "splunkmetric"
 splunkmetric_hec_routing = true
 [outputs.http.headers]
 Content-Type = "application/json"
 Authorization = "Splunk 3042d3d2-c140-48e7-a7c7-2f60f4c54b05"
 X-Splunk-Request-Channel = "3042d3d2-c140-48e7-a7c7-2f60f4c54b05"
```

Storage
Time Series Database

Monitoring and Visualizations

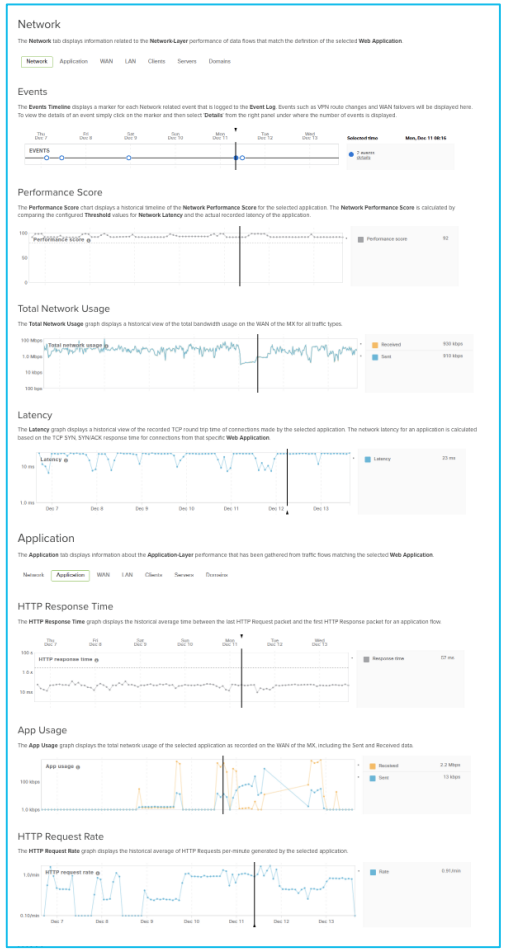
splunk>
a CISCO company



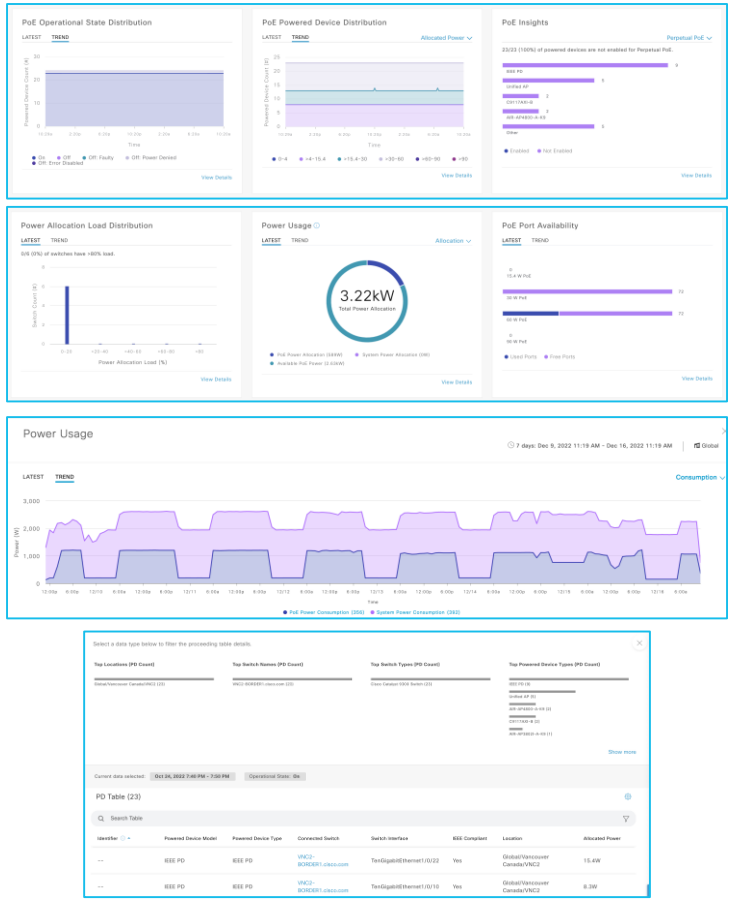
Cisco-on-Cisco Telemetry Integrations ?

Q: How does Cisco consume telemetry from IOS XE ?
A: The Cisco software controllers also use the gRPC & NETCONF telemetry interfaces to collect data

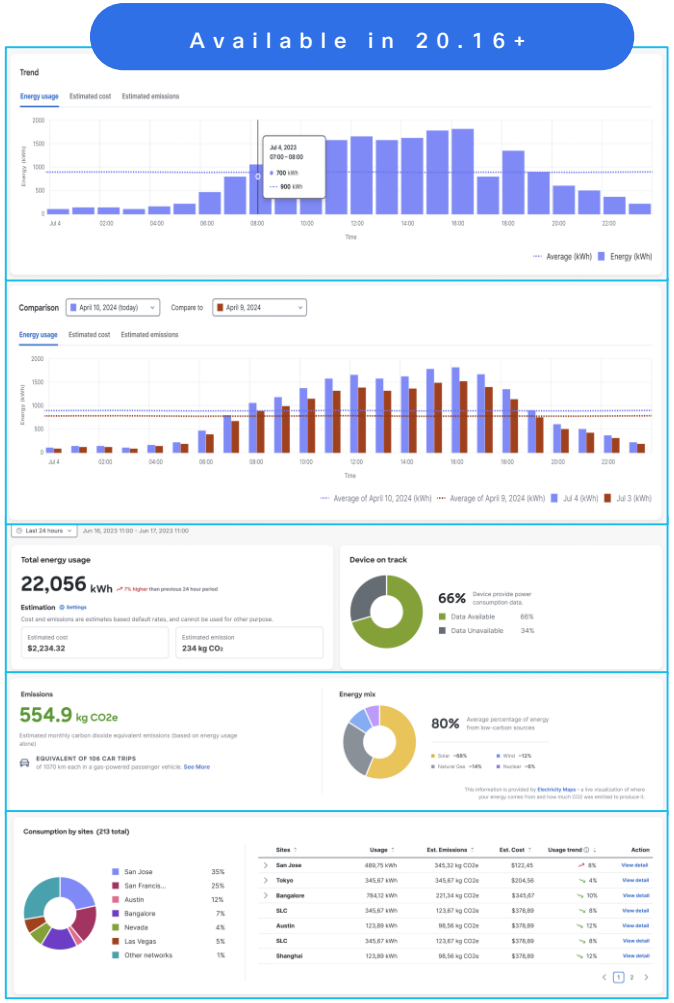
Meraki Dashboard



Catalyst Center



Catalyst SD-WAN

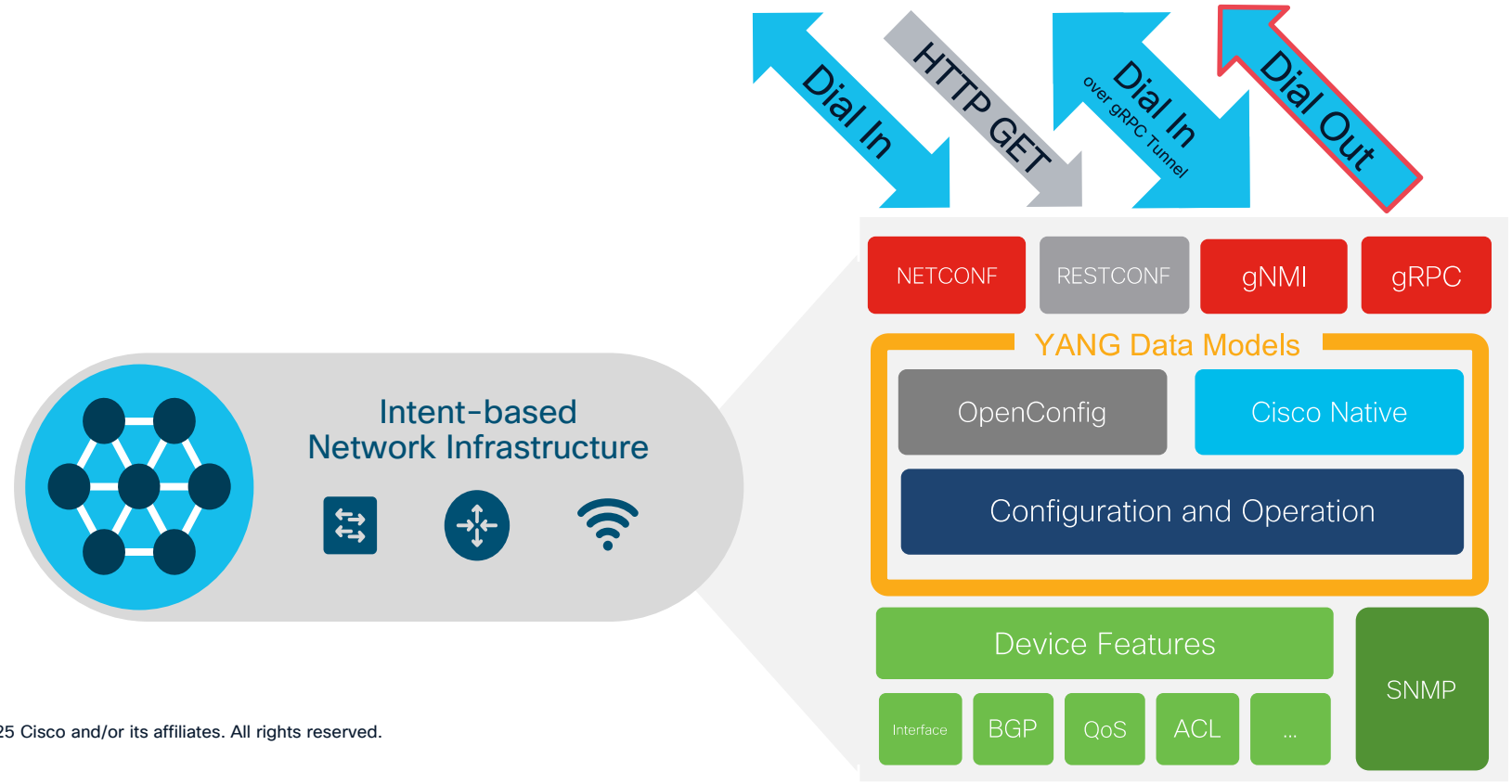


Model Driven Telemetry Interfaces

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

Dial 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

Model Driven Telemetry Interface Comparison

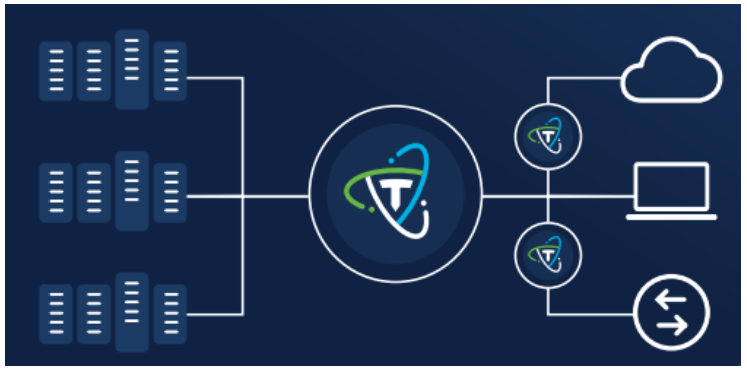
	NETCONF	gRPC (Dial-Out)	gNMI
Minimum IOS XE Version	16.6 (2017)	16.10 (2018)	Dial-In: 16.12 (2019) gRPC tunnel (2023)
Telemetry Direction	Dial-In, IOS XE is server	Dial-Out IOS XE is client	Dial-In IOS XE is server Dial-Out gRPC Tunnel
Configuration	Dynamic per session	Static per configuration	Dynamic per session
Telemetry Collector	Client	Server	Client
Encoding	XML	KV GPB	JSON_IETF + PROTO
Security	SSH + PKI certificate or password	mTLS or plain-text	mTLS certificates mTLS cert only or mTLS cert + user/pass authentication
Transport Protocol	SSH	HTTP2	HTTP2
Data Models	YANG	YANG	YANG

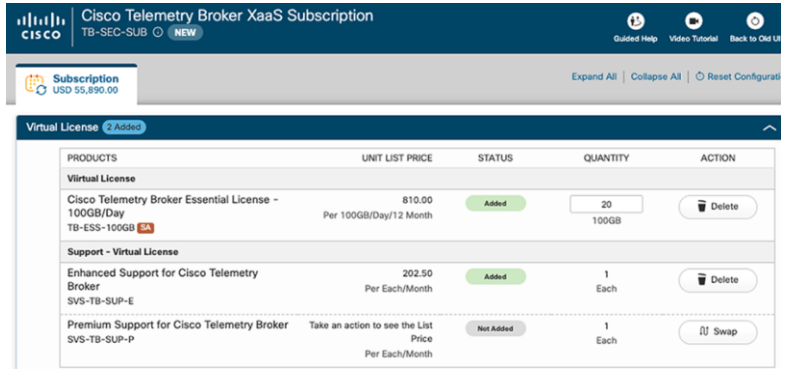
Network architecture, security posture and policy, YANG data modules, tools and language preferences, and <u>standards</u>, and software version, are some considerations when leveraging the various MDT interfaces

- NETCONF for RFP/compliance use only – not really seen in use in production
- gRPC for most enterprises: the NetOps configures the telemetry to push to the server
- gNMI for leading edge: NetDevOps accesses the network device

Cisco Telemetry Data Broker (Telegraf)

Cisco Telemetry Broker provides many benefits include brokering, filtering, and transforming data. It provides the ability to replicate telemetry data.





- **Brokering Data:**
  The ability to route and replicate telemetry data from a source location to multiple destination consumers.

  Quickly onboard new telemetry-based tools!

- **Filtering Data:**
  The ability to filter data that is being replicated to consumers for fine grain control over what consumers are able to see and analyze.

  Save money sending data to expensive tools!

- **Transforming Data:**
  The ability to transform data protocols from the exporter to the consumer's protocol of choice.

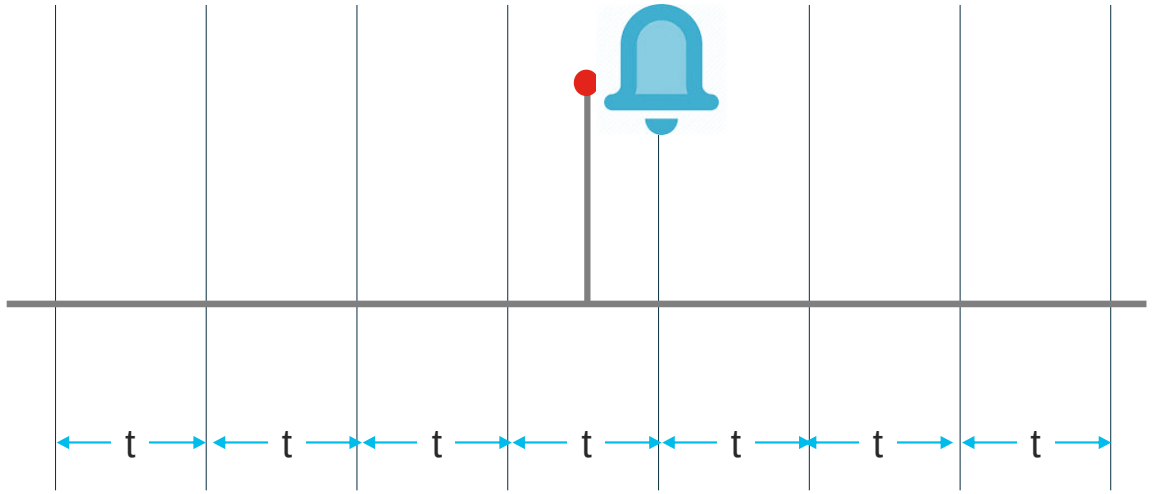
  Enable tools to consume multiple data formats!

https://cs.co/telemetrybroker aka https://www.cisco.com/c/en/us/products/security/telemetry-broker/index.html
https://blogs.cisco.com/security/taking-full-control-of-your-telemetry-with-the-intelligent-telemetry-plane

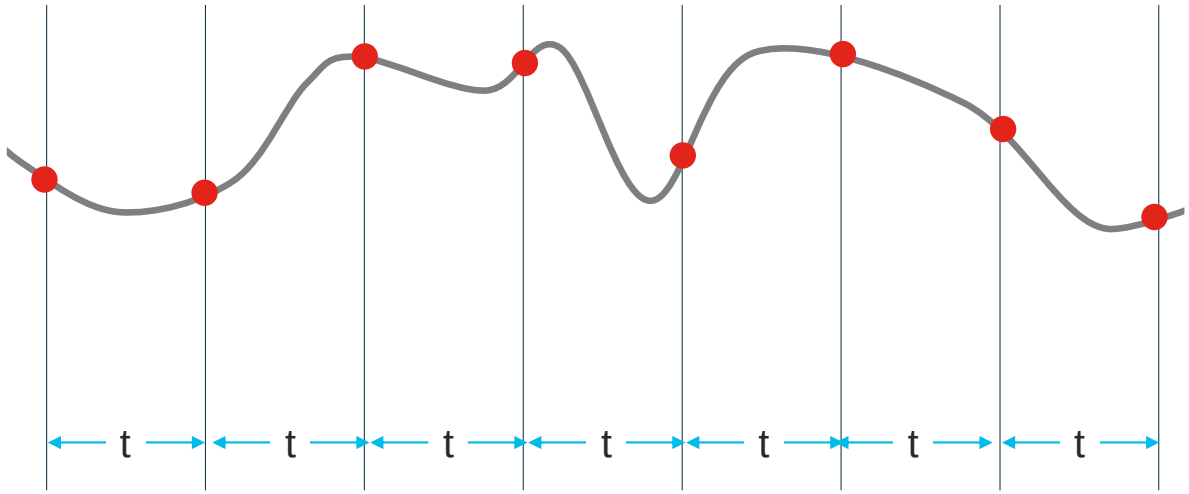
Publication options

We'll focus on periodic in this session because it is the most common!

On-Change



Periodic



Feature Model "On-Change" Notifications
Event Notifications (failed login, optic fault, etc)
State and Configuration

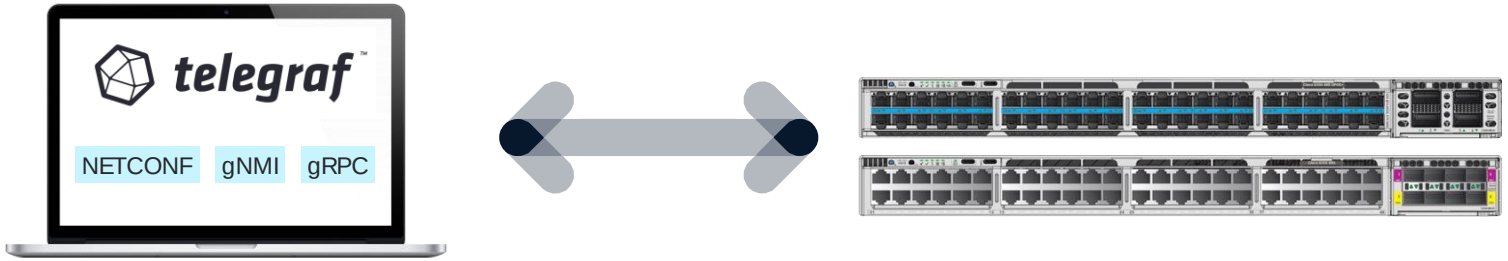
Feature Model "Periodic" Notifications
Time based publication
Minimum interval 100 centiseconds (1s)

OpenConfig YANG with GNXI, not NETCONF

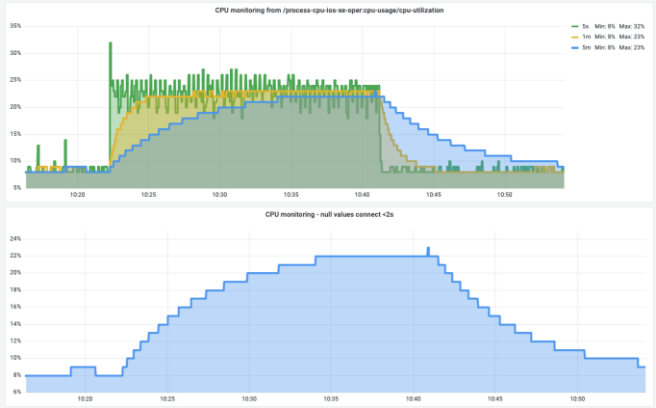
Model Driven Telemetry: usage comparison

60-minute collection sample with 60-second update interval

Interface	CPU Impact	PCAP file size/data size (MB)	Data byte Rate	Data bit rate	Average Packet Rate (sec)	Average Packet Size (bytes)
gNMI	+3%	23 MB	6 kBps	53 kbps	5	1180
gRPC	+3%	69 MB	19 kBps	155 kbps	58	333
NETCONF	+2%	83 MB	23 kBps	185 kbps	29	780
RESTCONF	+4%	200 MB	35 kBps	281 kbps	37	945
SNMP *	+6%	120 / 87	24 kBps	197 kbps	90	273



This demonstrates that even when SNMP is only measuring Interfaces the load is still significantly higher than YANG which is measuring significantly more YANG data



17 xpaths collected at 60 second update interval

/arp-ios-xe-oper:arp-data
/cdp-ios-xe-oper:cdp-neighbor-details
/environment-ios-xe-oper:environment-sensors
/if:interfaces-state
/interfaces-ios-xe-oper:interfaces/interface
/ios:native
/lldp-ios-xe-oper:lldp-entries
/matm-ios-xe-oper:matm-oper-data
/mdt-oper:mdt-oper-data/mdt-subscriptions
/memory-ios-xe-oper:memory-statistics/memory-statistic
/oc-if:interfaces/interface/state/counters
/oc-platform:components
/oc-sys:system
/platform-ios-xe-oper:components
/poe-ios-xe-oper:poe-oper-data/poe-switch
/process-cpu-ios-xe-oper:cpu-usage/cpu-utilization
/process-memory-ios-xe-oper:memory-usage-processes

+ Device-hardware-oper + Switch-stack-oper + more ?

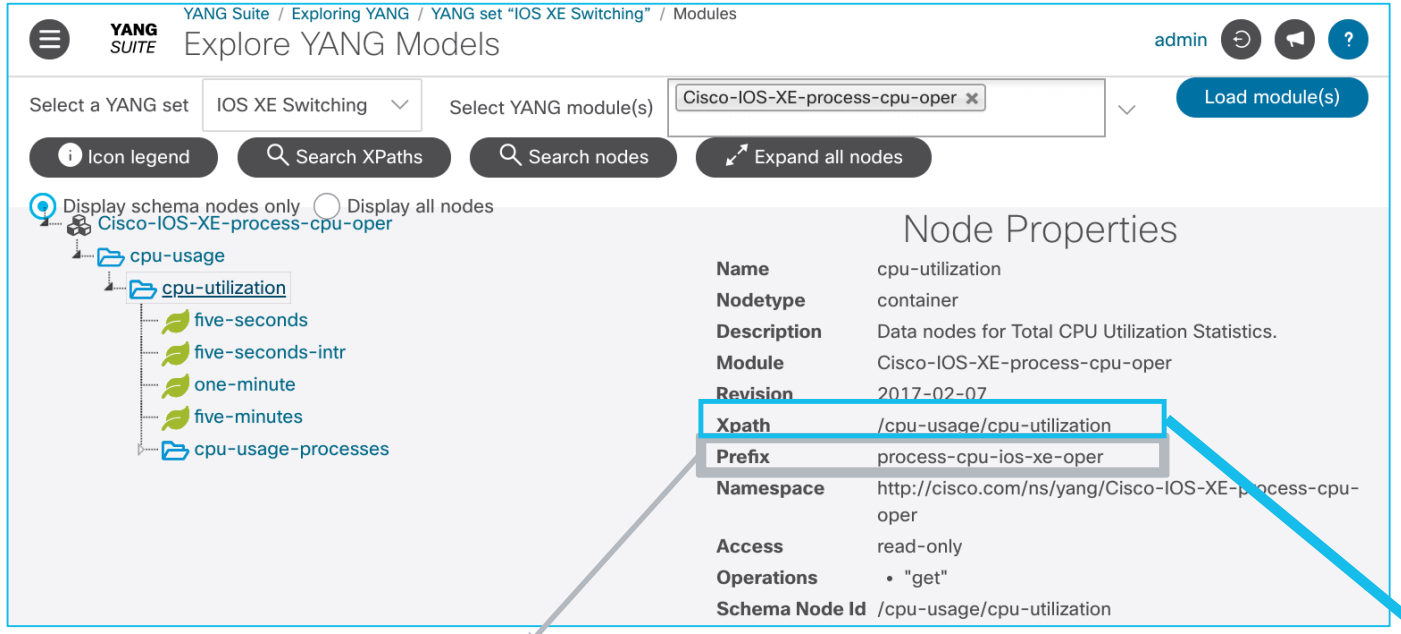
Model Metadata with YANG Suite

The CLI config to enable MDT requires the “filter xpath” which defines which YANG data to publish

Use YANG Suite to find the prefix and xpath from the YANG models to use in the configuration

Configuration Guide: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1610/b_1610_programmability_cg/model_driven_telemetry.html

filter xpath / PREFIX : XPATH



filter xpath /process-cpu-ios-xe-oper:cpu-usage/cpu-utilization

Configure a gRPC Telemetry Subscription

Configuring telemetry subscriptions like the following to collect CPU data over time

On Cisco IOS XE Device:

```
configure terminal
telemetry ietf subscription 1
 encoding encode-kvgpb
 filter xpath /process-cpu-ios-xe-oper:cpu-usage/cpu-utilization
 stream yang-push
 update-policy periodic 60000
 receiver ip address 10.1.1.3 57500 protocol grpc-tcp
```

See more examples at: https://github.com/jeremycohoe/cisco-ios-xe-mdt

Enhanced Energy Metering

IOS XE 17.15.1

What it is

Enables energy meter for Switch System Power and PoE switchport power consumption
Platforms: C9200, C9300, C9400

What it does

System Energy is power consumed by the system for a specific duration, measured in unit of Watt Second. Macro-metered window size of 3-hours with 12 x 15-minute micro-meters

Gives the visibility within the windows of consumption that can be further analyzed and considered for carbon intensity reporting, density-based usage analysis and so on.

What it receives

Ledger on Energy consumption for System Power and PoE Port with a bucketized data of 15 minutes each.

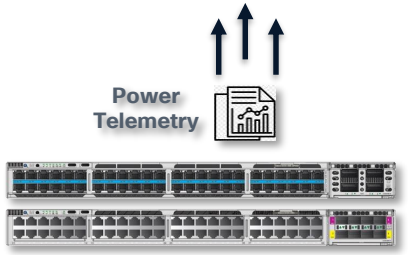
System Meter

```
Meter start time 2024-10-22 10:36:34 PST
Energy Data For Last 180 Minute

                    System Energy   System Meter Update
Mod  Model No          (MilliWattSec)  Time
---  ---------------------  ---------------  -------------------------

1    C9300-24H         234894600       2024-10-22 11:32:39 PST

---  ---------------------  ---------------  -------------------------
```

PoE Port Meter

```
SB-Salone1-C9324H#sho power inline meter
Module   Available    Used     Remaining
         (Watts)    (Watts)   (Watts)
------   ---------  --------  ---------
1        860.0      120.0     740.0

Interface   Meter Update          Hourly Metered       Metered Energy in MilliWattSec (15min Buckets)
            Time                  Value (MilliWattSec)
----------  ----------------------  -------------------  ---------------------------------------------------------
---------------------------------------------------------
Gi1/0/1     2024-10-22 11:21:09 PST                0      0-0-0-0-0-0-0-0-0-0-0-0

Gi1/0/2     2024-10-22 11:21:09 PST           10507380  1465368-3553224-3558320-1930468-0-0-0-0-0-0-0-0
```

15 minutes/bucket #



0	15 min
1	30 min
2	45 min
3	60 min
4	75 min
5	90 min
6	105 min
7	120 min
8	135 min
9	150 min
10	165 min
11	180 min

Hour-1

Hour-2

Hour-3

Subscription to the new platform data

```
telemetry ietf subscription 79007
 encoding encode-kvgpb
 filter xpath /platform-ios-xe-oper:components/component/platform-properties/platform-property
 source-address 172.26.202.111
 stream yang-push
 update-policy periodic 300000
 receiver ip address 172.26.202.69 57000 protocol grpc-tcp

telemetry ietf subscription 79008
 encoding encode-kvgpb
 filter xpath /poe-ios-xe-oper:poe-oper-data/poe-port-detail
 source-address 172.26.202.111
 stream yang-push
 update-policy periodic 30000
 receiver ip address 172.26.202.69 57000 protocol grpc-tcp
```


```
telemetry ietf subscription 79007
 encoding encode-kvgpb
 filter xpath /platform-ios-xe-oper:components/component/platform-properties/platform-property
 source-address 172.26.202.111
 stream yang-push
 update-policy periodic 300000
 receiver ip address 172.26.202.69 57000 protocol grpc-tcp
```

```
telemetry ietf subscription 79008
 encoding encode-kvgpb
 filter xpath /poe-oper-data/poe-port-detail
 source-address 172.26.202.111
 stream yang-push
 update-policy periodic 30000
 receiver ip address 172.26.202.69 57000 protocol grpc-tcp
```

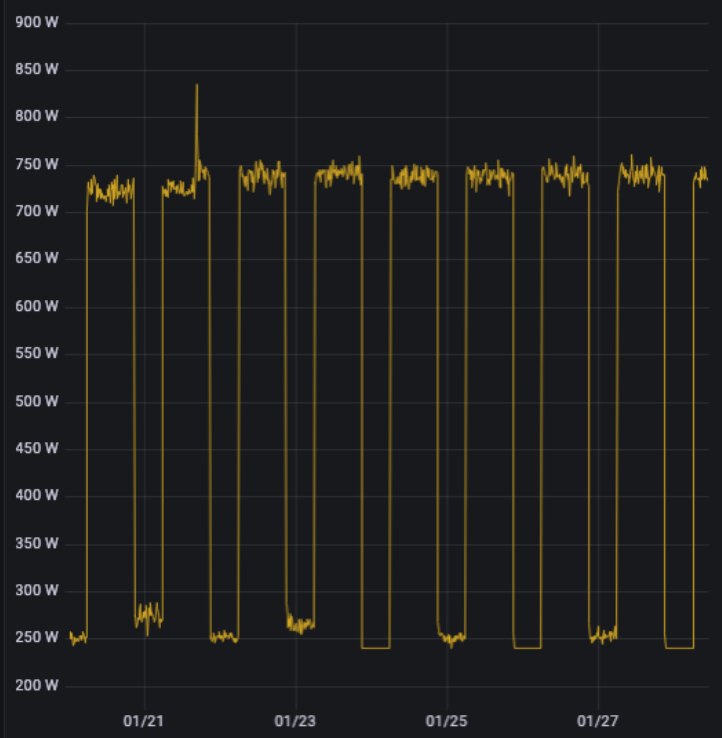
General / Cisco Catalyst IOS XE - POE - SmartBuilding

2023-01-20 00:24:10 to 2023-01-28 10:43:16

Interface All switch SB-Salone1-C9324H

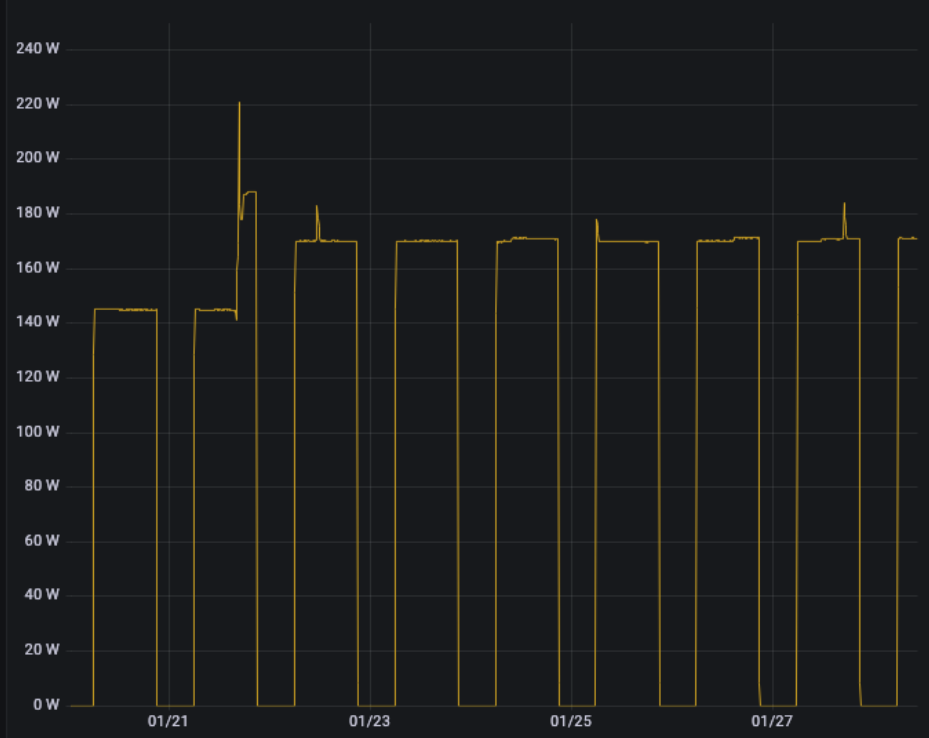
Total Power Allocated - POE + System

	Min	Max	Last *
Salone	240 W	836 W	733 W



Realtime POE Power Consumption

	Min	Max	Last *
Salone	0 W	221 W	171 W

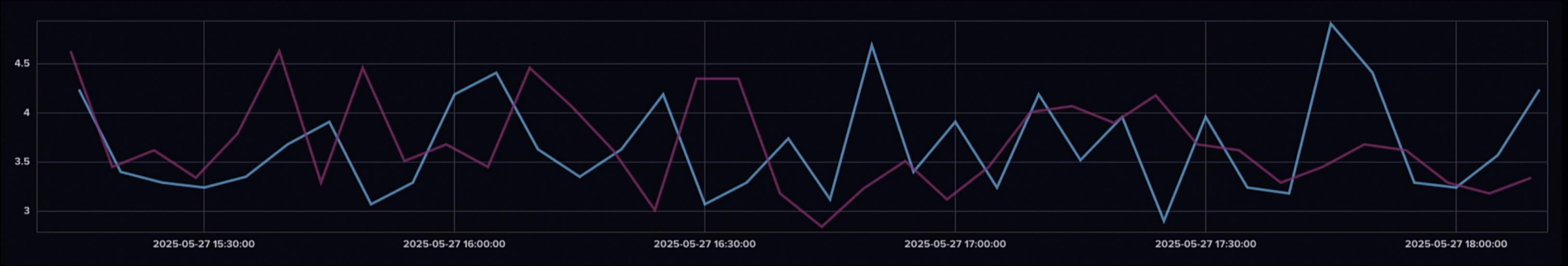


Per interface power consumption

GigabitEthernet1/0/12 Mean: 23.5 W
GigabitEthernet1/0/13 Mean: 54.7 W
GigabitEthernet1/0/15 Mean: 0 W
GigabitEthernet1/0/16 Mean: 1.03 W
GigabitEthernet1/0/19 Mean: 0 W
GigabitEthernet1/0/20 Mean: 1.23 W
GigabitEthernet1/0/21 Mean: 0 W
GigabitEthernet1/0/22 Mean: 0 W
GigabitEthernet1/0/23 Mean: 4.56 W
GigabitEthernet1/0/24 Mean: 8.87 W
GigabitEthernet1/0/3 Mean: 4.99 W
GigabitEthernet1/0/5 Mean: 14.2 W
GigabitEthernet1/0/6 Mean: 1.04 W
GigabitEthernet1/0/7 Mean: 1.91 W
GigabitEthernet1/0/9 Mean: 74.2 W

Data Explorer

Band | CUSTOMIZE | Local | SAVE AS



Query 1 (0.02s) | + | View Raw Data | Past 3h | SCRIPT EDITOR | SUBMIT

FROM

Search buckets

- SB-Bucket
- _monitoring
- _tasks
- + Create Bucket

Filter — source (3)

Search source tag values

- [x] C9350-48U_Stack_Edge
- [ ] SB-C9524Q-Core-SVL
- [x] svlhw-9300x-48hx-3
- [x] svlhw-93

Filter — subscription

Search subscription tag value

- 69007
- 69008
- 69009
- 69010
- 69011
- 69012
- 69013
- 69014
- 69015

Filter — _measurement (1)

Search _measurement tag va

- Cisco-IOS-XE-interfac…
- Cisco-IOS-XE-memory-o…
- Cisco-IOS-XE-platform…
- Cisco-IOS-XE-platform…
- Cisco-IOS-XE-poe-oper…
- [x] Cisco-IOS-XE-poe-oper…
- Cisco-IOS-XE-poe-oper…
- Cisco-IOS-XE-poe-oper…
- Cisco-IOS-XE-switchpo…

Filter — _field (2)

Search _field tag values

- module_id
- number_of_buckets
- oper_police
- [x] oper_power
- oper_priority
- oper_state
- over_current_counter
- pd_class

Filter — _field (1)

Search _field tag values

- device_detected
- value/string
- device_name
- [x] oper_power

Filter — intf_name

Search intf_name

- [x] TenGigabitEt
- TenGigabitEt
- TenGigabitEt
- TenGigabitEt
- TenGigabitEt
- TenGigabitEt
- TenGigabitEt

WINDOW PERIOD

CUSTOM | AUTO

auto (1m)

Fill missing values

AGGREGATE FUNCTION

CUSTOM | AUTO

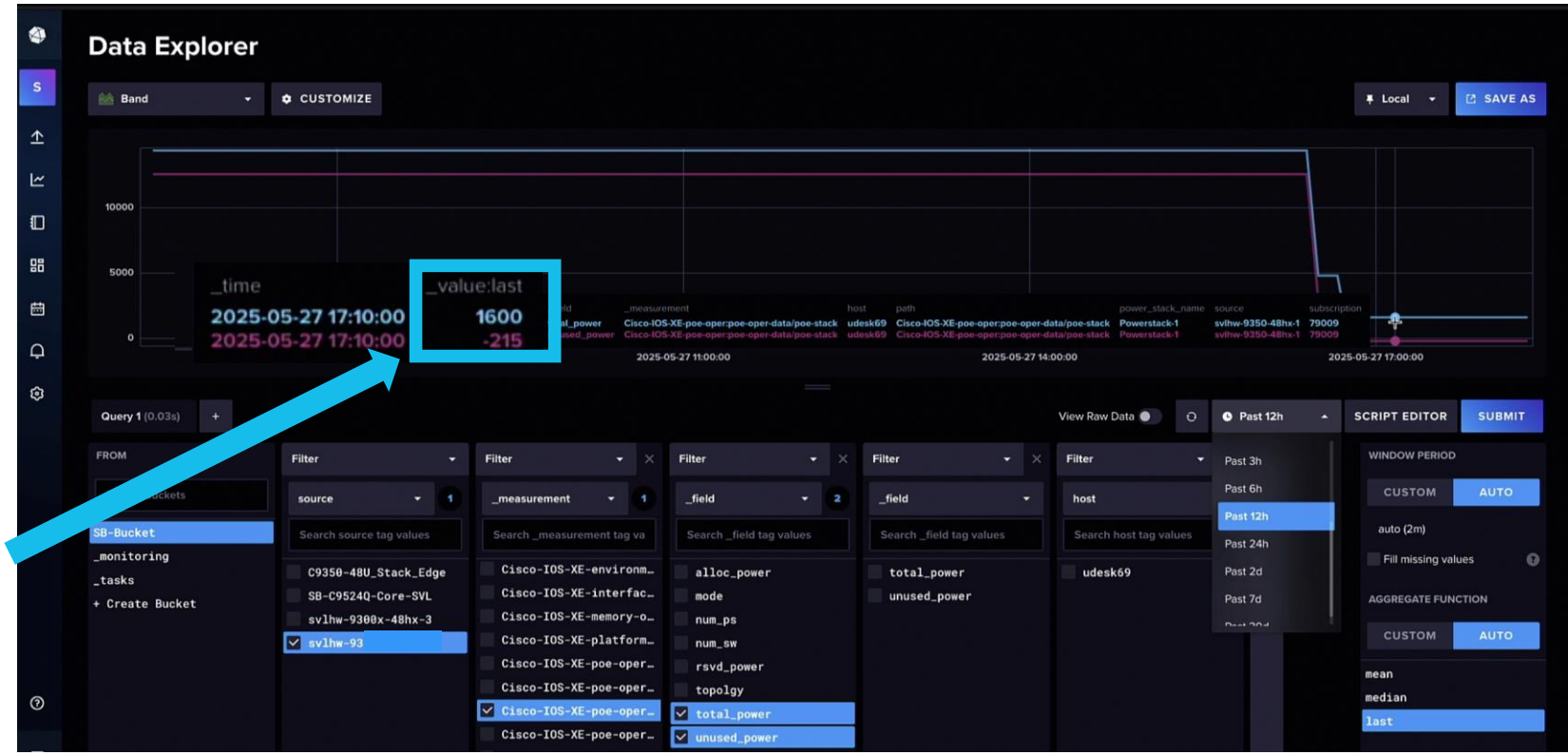
- mean
- median
- last

CLI

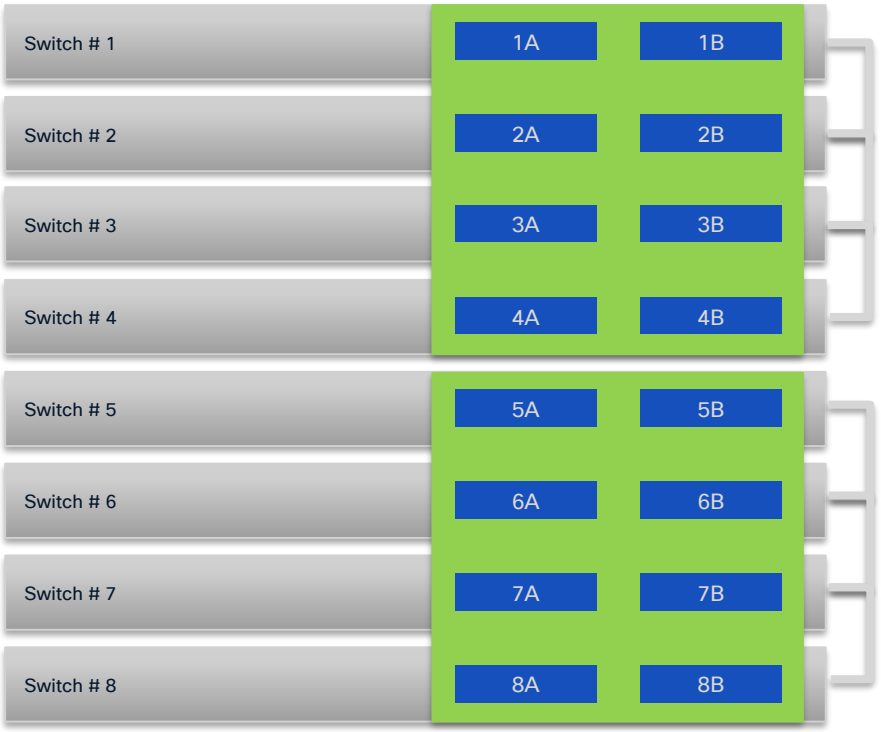
```
show power detail
```

Power Summary (in Watts)	Allocated	Consumed	Maximum Available
System Power	1440	518	1470
POE Power	345	67	130
Total	1785	585	1600

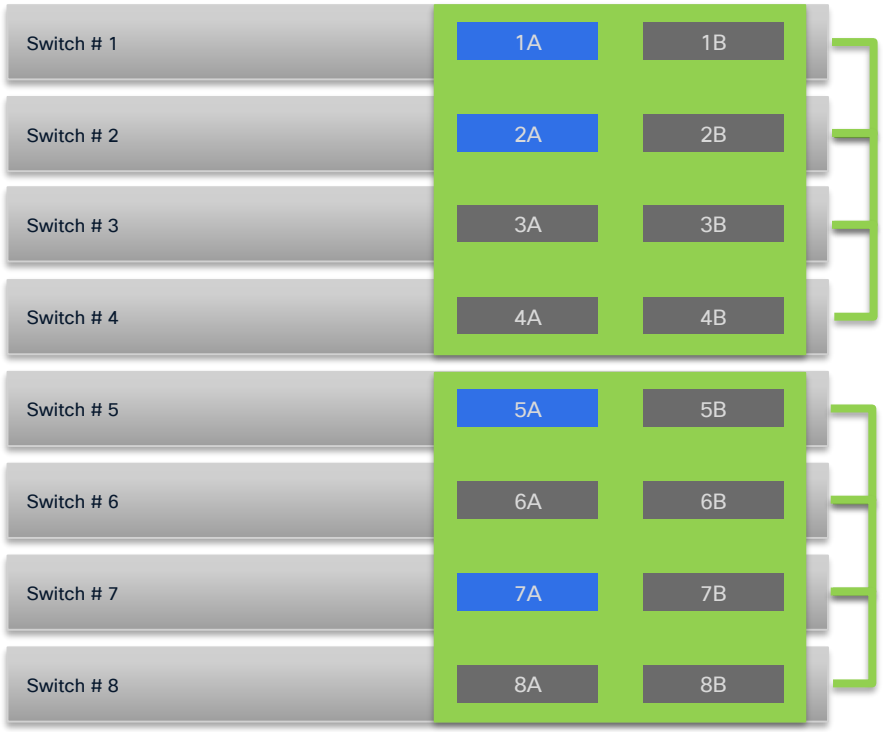
Telemetry



Auto-off StackPower PSU



8 x C9300X switch stack
2 StackPower Domains
16 Active PSUs



8 x C9300X switch stack
2 StackPower Domains
4 Active PSUs

- PSU Active
- PSU auto-off
- Stack-power group

StackPower Power draw with auto-off

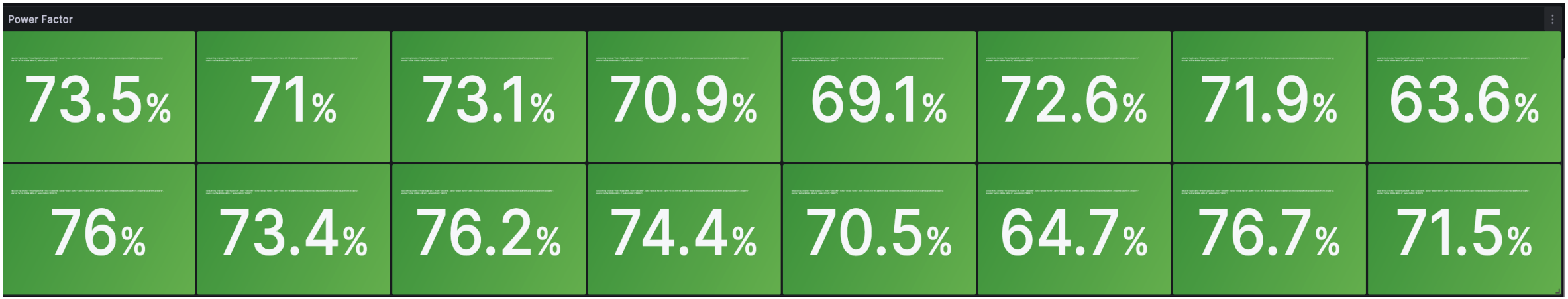
Stack Power Supplies Output Power

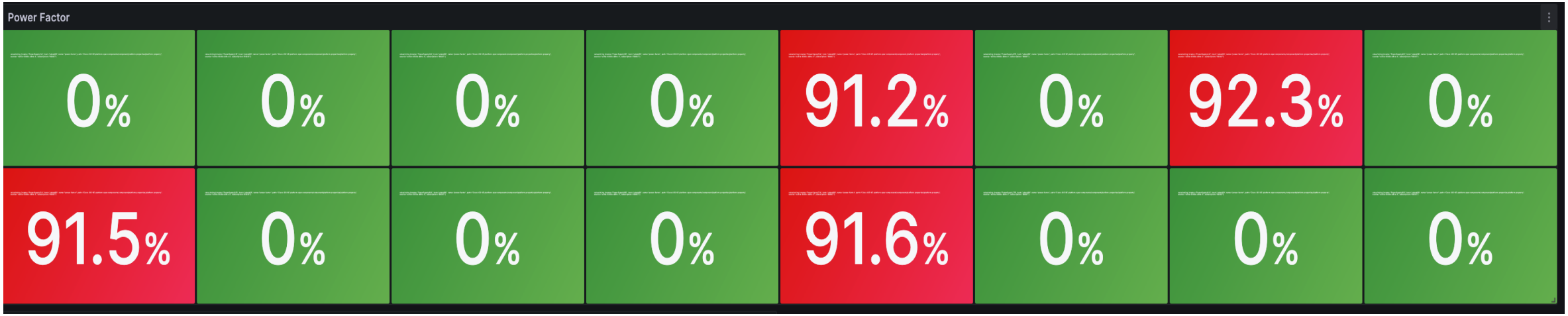
91	88.8	90.3	86.8	85.3	94.8	83.3	48
86.8	89.5	93	89.3	90.3	51.5	98	84

Stack Power Supplies Input vs Output Power

0	0	0	0	362	0	331	0
329	0	0	0	370	0	0	0

StackPower Efficiency with auto-off



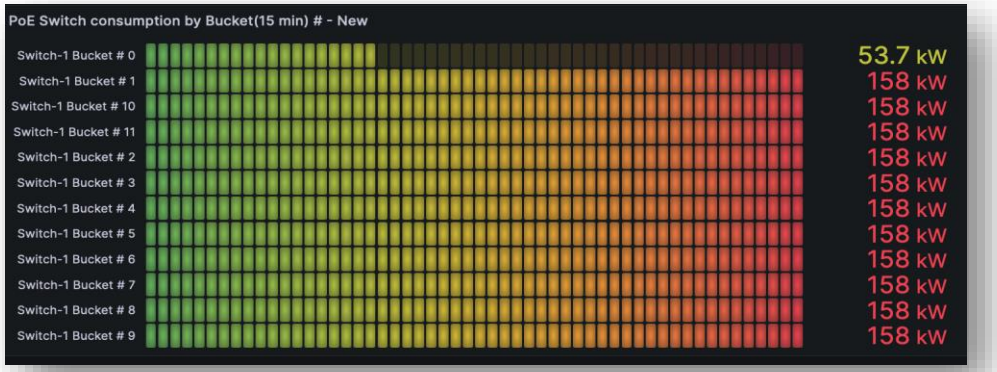


What can we do with Power Telemetry

Grafana

InfluxDB

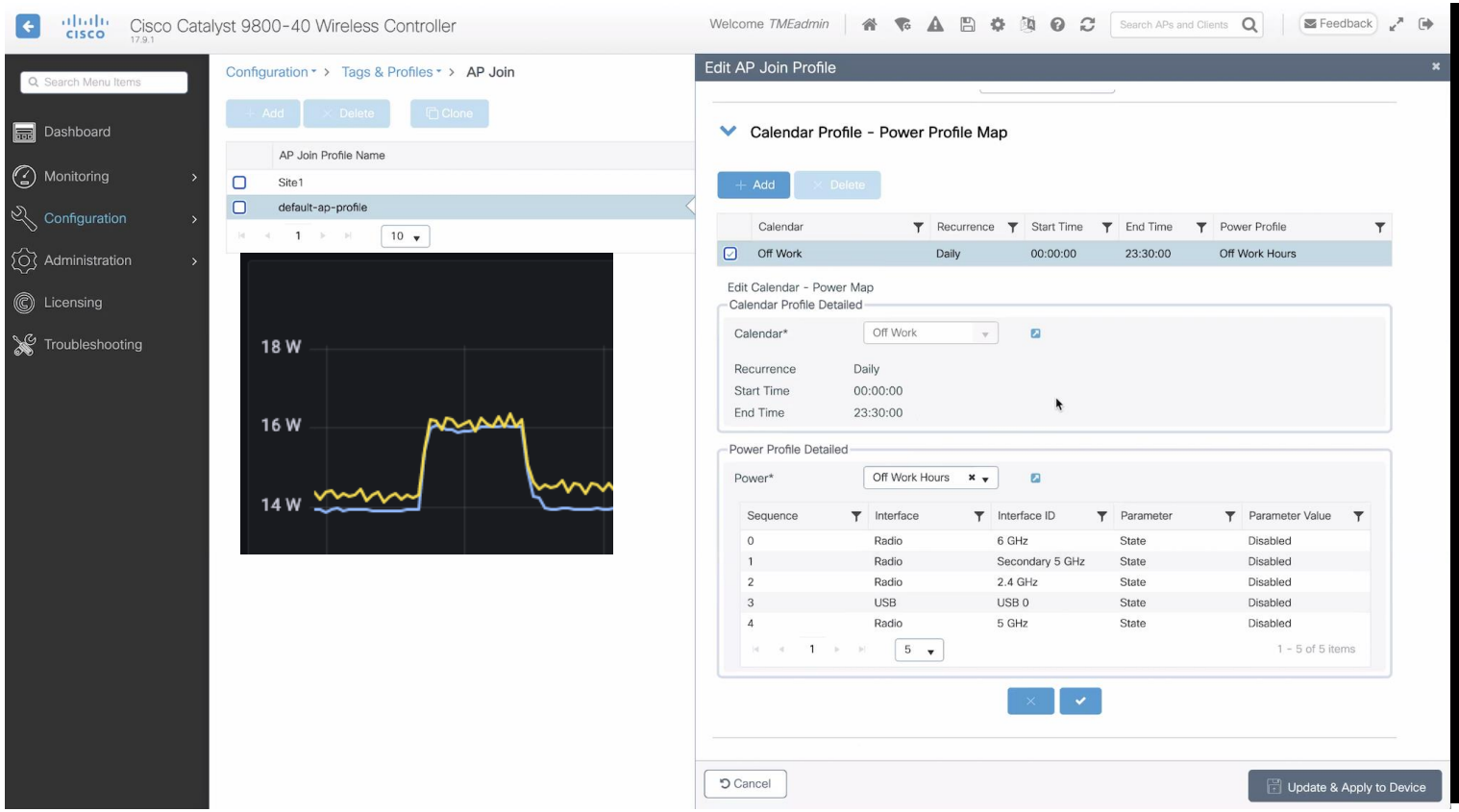
Telegraf



Power Telemetry

Catalyst 9800 WLC Calendar Template scheduling

To enable **power save mode** on Cisco Catalyst Access Points
https://github.com/jeremycohoe/c9800-ap-power-save/




```
wireless profile power "Off Work Hours"
 0 radio 6ghz state shutdown
 1 radio secondary-5ghz state shutdown
 2 usb 0 state disable
 3 radio 5ghz state shutdown

wireless profile calender-profile name "Off Work 5PM to Midnight"
 day monday
 day tuesday
 day wednesday
 day thursday
 day friday
 recurrance weekly
 start 17:00:00 end 23:59:59

wireless profile calender-profile name "Off Work Midnight to 8AM"
 day monday
 day tuesday
 day wednesday
 day thursday
 day friday
 recurrance weekly
 start 00:00:00 end 08:00:00

ap profile default-ap-profile
 calendar-profile "Workday 5pm to Midnight"
  action power-saving-mode power-profile "Off Work Hours"
 calendar-profile "Workday Midnight to 8am"
  action power-saving-mode power-profile "Off Work Hours"
```

General / Cisco Catalyst IOS XE - POE

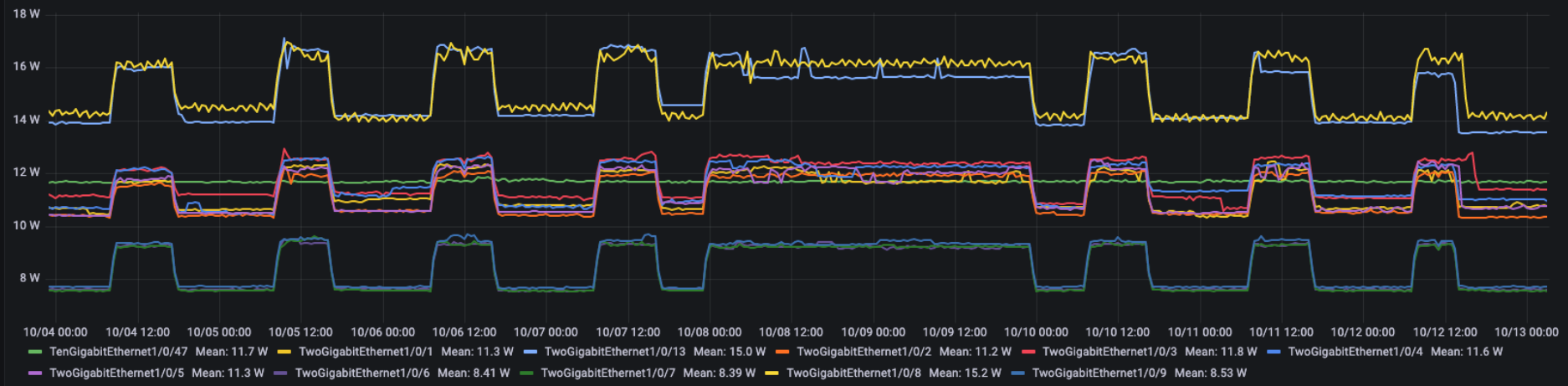
2022-10-03 22:58:16 to 2022-10-13 03:15:15

Interface: TenGigabitEthernet1/0/47 + TwoGigabitEthernet1/0/1 + TwoGigabitEth...

switch: SJC14F1-WTME-C9K-48UXM

Total device power consumption

Per interface power consumption



POE Interface

device	intf_name	power_admin_max	max_power_drawn	power_consumption	power_to_pd
C9130AXE-B	TenGigabitEthernet1/0/47	60	13.3	11.6	33.3
CW9164I-B	TwoGigabitEthernet1/0/1	60	15.6	11.9	31.4
C9136I-B	TwoGigabitEthernet1/0/13	60	21.3	17.0	41.9
C9136I-B	TwoGigabitEthernet1/0/16	60	6.25	0.950	41.9
CW9164I-B	TwoGigabitEthernet1/0/2	60	13.3	12.5	31.4
CW9166I-B	TwoGigabitEthernet1/0/3	60	15.7	12.6	34.1

Monitoring & Automation Impact

Up to 25% ⬇ of Energy usage **reduction** on POE switchports with IOT endpoints turned off ~ 9hrs/Day

Minimum of 12% ⬇ Energy usage **reduction** for controller-based Cisco Catalyst Access Points

100 % ⬆ Visibility on POE Power Consumption

*based on use case and data presented in demos

Device Optimization

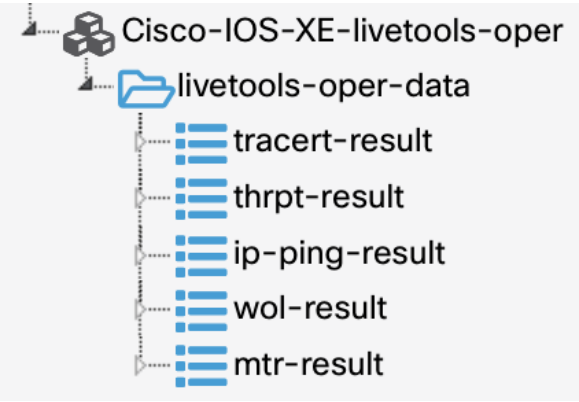
Livetools: YANG support for common networking operations

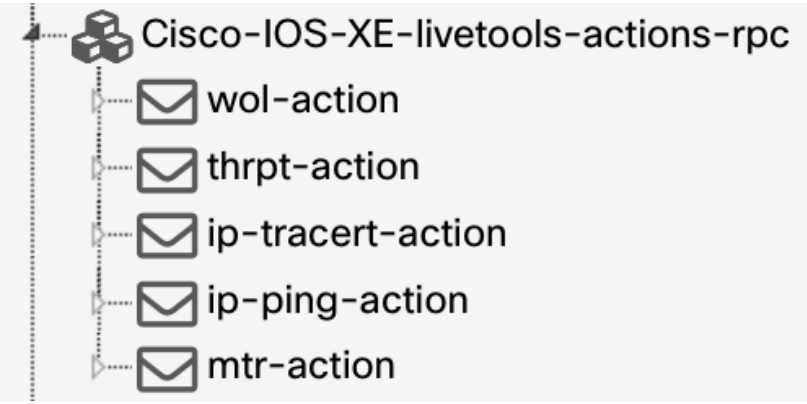
1. MTR
2. Ping
3. Throughput
4. Traceroute
5. Wake-On-LAN

Cisco-IOS-XE-livetools: Ping, Traceroute, MTR, etc

- 5 common network management tasks including ping and traceroute are now available through the YANG API (mtr, ping, tracerout, throughput, WOL)
- They are called from the “RPC” model with variable inputs like IP address, etc
  - A “JobID” is returned to programmatically retrieve the results
- The Results are available by querying the “oper” model

1. MTR
2. Ping
3. Throughput
4. Traceroute
5. Wake-On-LAN

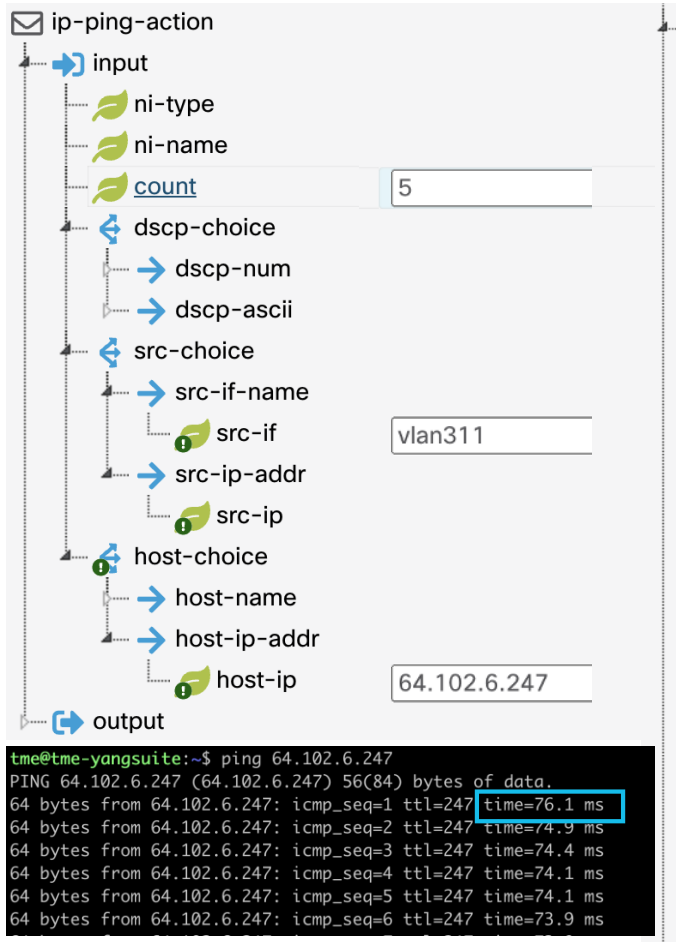


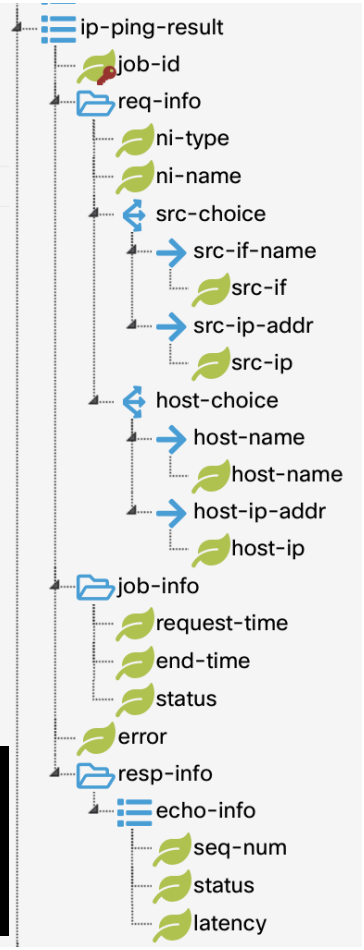


These “live tools” enable additional programmatic network connectivity troubleshooting and validation

Ping RPC

The ping RPC measures the network latency in milliseconds to the specified IP or hostname





RPC TO EXEC PING

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <ip-ping-action xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-livetools-actions-rpc">
    <count>5</count>
    <src-if>vlan311</src-if>
    <host-ip>64.102.6.247</host-ip>
  </ip-ping-action>
</rpc>
```

RPC TO GET PING RESULTS

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <get>
    <filter>
      <livetools-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-livetools-oper">
        <ip-ping-result>
          <job-id>1</job-id>
        </ip-ping-result>
      </livetools-oper-data>
    </filter>
  </get>
</rpc>
```

RESPONSE FROM API

```
<livetools-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-livetools-oper">
  <ip-ping-result>
    <job-id>1</job-id>
    <req-info>
      <ni-type>net-inst-default</ni-type>
      <ni-name/>
      <src-if>vlan311</src-if>
      <host-ip>64.102.6.247</host-ip>
    </req-info>
    <job-info>
      <request-time>2024-09-11T23:34:01.976+00:00</request-time>
      <end-time>2024-09-11T23:34:02.348+00:00</end-time>
      <status>job-completed</status>
    </job-info>
    <error>ip-ping-no-err</error>
    <resp-info>
      <echo-info>
        <seq-num>1</seq-num>
        <status>echo-success</status>
        <latency>75</latency>
      </echo-info>
      <echo-info>
        <seq-num>2</seq-num>
        <status>echo-success</status>
        <latency>74</latency>
      </echo-info>
```

Resources

Cisco Live US Programmability Learning Map

Sunday–8th

TECOPS-2314 9:00A
Automating All Things YANG, All the time - Programmability and Automation 101 with Cisco IOS XE

Monday–9th

BRKOPS-2256 10-11:30
Exploring Practical AIOps Use Cases for Enterprise Networks with Splunk 1-2PM

BRKENS-2604
Atomic Config Replace with Cisco Catalyst 9000 3:30-5PM

BRKOPS-1401
Cisco IOS XE: Telemetry, Automation, and YANG–Oh My!

Tuesday–10th

DEVNET-1110 11:30-1:15
Modern approaches for IOS XE network device management on Cat9k

BRKDEV-2017 1:30-2:30
gRPC, gNMI, gNOI... Oh My! An Enterprise Network Automation Journey 2-3:30

BRKOPS-2223
Network of the Future is Here - Let's Automate your IPv6 deployment with Python!

Wednesday–11th

CISCOU-1059 4-4:30PM
Observability at TorIX: custom telemetry solutions on next-gen campus switching

Thursday–12th


May 16, 2025 Leave a Comment

Developer

Explore Cisco IOS XE Automation at Cisco Live US 2025

7 min read

Story DeWeese

Cisco IOS XE offers advanced programmability and automation capabilities that simplify network management and reduce complexity. By utilizing open, standards-based APIs such as NETCONF, RESTCONF, and gNMI, network administrators can efficiently provision and configure devices, minimizing manual errors. The integration of Zero-Touch Provisioning with DevOps toolkits accelerates device onboarding, while model-driven telemetry provides real-time data collection for enhanced monitoring and troubleshooting. Cisco's automation solutions are extensible across the entire device lifecycle, supporting seamless software upgrades and patching. These features allow our customers to drive business outcomes and be at the cutting edge of network automation, enabling faster, more secure, and efficient network operations.

https://blogs.cisco.com/developer/cisco-ios-xe-automation-clus25

Walk-in labs (open Monday-Thursday)

- ✓ Efficiently monitoring device statistics in real-time using gRPC Dial-out with IOS XE [LABPRG-2004]
- ✓ Explore and test YANG models for model driven telemetry on IOS XE with Cisco's YANG Suite [LABOPS-2000]
- ✓ Hands-On Lab: Monitoring Cisco IOS-XE Devices with RESTCONF [LABDEV-2001]

Cisco Live US Catalyst 9000 Learning Map

Sunday–8th

TECENS-2620 9:00AM
Catalyst 9000 Switching Architecture and Software Innovations

TECENS-2680 2:00PM
Cisco Catalyst 9000 Switching Family Architecture

TECARC-2446 2:00PM
BGP EPVN in Enterprise Campus with Catalyst 9000 Switching Platforms

Monday–9th

BRKENS-1500 8:00AM
Introduction to Campus Network Design and Multilayer Architectures

BRKENS-2092 8:00AM
BGP EVPN in Enterprise Campuses with Catalyst 9000 Series Switches

BRKENS-2095 9:30AM
Designing Highly Available Networks using Catalyst 9000 Series Switches

BRKENS-2652 11:00AM
Connecting Beyond Fabric: Catalyst 9000 BGP EVPN Handoff Scenarios

BRKENS-2604 1:00PM
Atomic Config Replace with Cisco Catalyst 9000

BRKENS-2608 2:30PM
Future-proofing Campus Switching for WiFi7

BRKARC-1012 2:30PM
Investment Protection with Catalyst 9000 Series Switching & Wireless: A Competitive Edge

BRKENS-2099 4:00PM
Innovations on Cisco Campus Switching for Sustainability and Energy Management

Tuesday–10th

LTRARC-3001 8:00AM
Mastering Catalyst 9000 Switches: Architectural Insights and Troubleshooting Strategies

LTRENS-2429 8:00AM
AI/ML in Cisco Catalyst Center: Transforming Network Operations!

BRKARC-2092 11:00AM
Unlocking the Automation Power in Catalyst Center for Wired and Wireless Networks

BRKENS-2609 11:00AM
Deploy Cisco Catalyst Center with Rest-API's

BRKENS-2655 1:30PM
Catalyst Center Network Operations Essentials using UI and APIs

BRKENS-1402 3:00PM
Deploying Cisco Catalyst Center with CICD

BRKARC-2039 4:00PM
Cisco Catalyst 9000 Switching QoS with Silicon One ASICs Deep Dive

BRKENS-2603 4:00PM
Catalyst Switching enabled Smart Buildings : Beyond PoE Connectivity

Wednesday–11th

BRKENS-2610 10:30AM
Catalyst Center Network Operations Essentials using UI and APIs

CIUG-1109 10:30AM
Catalyst 9000 Switching Innovations & Roadmap

LTRENS-2256 1:00PM
Cisco Catalyst Switching Innovations Lab

BRKENS-2500 1:30PM
Advanced Campus Network Design: Multilayer Architectures and Next-Gen Protocols

BRKARC-2668 3:30PM
Campus Switching Architecture for Future Proofed Workspaces

Thursday–12th

BRKENS-2094 9:30AM
Media & Time Sensitive Networking with C9K Switches: Converging Time Sensitive Applications & Devices onto Ethernet

BRKARC-2099 10:30AM
Catalyst 9000 Series Switching Family: Core and Distribution

BRKARC-2098 10:30AM
Open-Source GenAI Bot for Catalyst Center

BU-led sessions

Cisco Live US Catalyst Center Learning Map

Sunday–8th

TECOPS-2001 9:00AM
The Ultimate Guide to Install, Onboard, and Operate Your Campus Network with Cisco Catalyst Center

LTRSEC-2005 9:00AM
Building Cisco SD-Access with Cisco Catalyst Center & ISE

TECENS-2680 2:00PM
BGP EPVN in Enterprise Campus with Catalyst 9000 Switching Platforms

Monday–9th

BRKOPS-2698 8:00AM
Choosing the Right Cisco Catalyst Center Deployment Model for Your Network

CIUG-1100 10:00AM
Cisco Catalyst Center: AI-Driven Switching: Revolutionizing Automation and Assurance

LTRXAR-3783 1:00PM
Cross-Architecture Integration Experience Lab

BRKENS-1601 1:30PM
Catalyst Center and Meraki Cloud: The Right Choice for your Catalyst 9000 Switch Management!

BRKOPS-2609 1:30PM
Cisco Catalyst Center: Built-In Integrations for Streamlined Network Operations

LTROPS-2341 2:00PM
Build a Flexible Network Automation Workflow with GitLab CI/CD, Catalyst Center, NetBox, and Ansible

IBOENS-1100 2:30PM
Cisco Catalyst Center and SD-Access Design Fundamentals

BRKEWN-2029 4:00PM
Separating hype from reality, real world use cases of AIOps and Assurance for wireless within Catalyst Center

BRKCOC-2483 4:00PM
Cisco IT: Streamlining Network Management and Decisions with Catalyst Center Automation and Splunk

CISCOU-3004 5:00PM
Configuring and Troubleshooting Catalyst Center Templates

Tuesday–10th

BRKEWN-2306 1:30PM
Wireless Network Automation and Assurance with Cisco Catalyst Center

IBOOPS-2391 1:30PM
AI/ML in Cisco Catalyst Center: Transforming Network Operations!

BRKOPS-2697 2:00PM
Unlocking the Automation Power in Catalyst Center for Wired and Wireless Networks

DEVWKS-1004 2:30PM
Deploy Cisco Catalyst Center with Rest-API's

Wednesday–11th

DEVNET-2660 10:00AM
Catalyst Center Network Operations Essentials using UI and APIs

DEVNET-2176 10:30AM
Deploying Cisco Catalyst Center with CICD

BRKTRS-2821 2:30PM
Troubleshooting Strategies for Cisco Catalyst Center & SD-Access

BRKXAR-1013 2:30PM
4 Ways to Streamline Your Licensing with Cisco's Networking Subscription Across Your Portfolio

BRKOPS-2379 3:30PM
Automate Catalyst Center with Cisco Workflows

BRKOPS-2835 4:00PM
5 new things you need to know about Catalyst Center licensing

Thursday–12th

BRKIOT-2016 8:30AM
Streamline Your Success: Automating OT Services with Cisco Catalyst Center Best Practices

BRKOPS-2442 8:30AM
Leveraging Digital Twin for Advanced Network Management with Cisco Catalyst Center

DEVNET-3000 9:30AM
Open-Source GenAI Bot for Catalyst Center

BRKOPS-2570 10:30AM
AI-Powered Automation: Building Smarter Apps for Cisco Catalyst Center Operations

BRKOPS-2492 10:30AM
Let's Deploy Catalyst Center Global Manager (CCGM): Single Pane of Glass for Multiple Catalyst Centers

BRKOPS-2343 10:30AM
Decoding Site Reliability Engineering Through Catalyst Center

○ BU-led sessions

Catalyst Center

Cisco Live US SD-Access Fabric Learning Map

Sunday–8th

TECENS-2820 9:00AM
Cisco Software-Defined Access LISP: Architecture Overview

LTRENS-2509 9:00AM
Mastering Cisco SD-Access: LISP Pub/Sub and its Benefits Made Simple

TECENS-2850 2:00PM
Security in Enterprise - A cross domain security primer across LAN, wLAN and WAN

Monday–9th

BRKENS-2810 10:00AM
Cisco Software-Defined Access LISP Solution Fundamentals

LTRENS-3751 1:00PM
SD-Access as Code with Cisco Catalyst Center and ISE Automation

IBOENS-1100 2:30PM
Cisco Catalyst Center and SD-Access Design Fundamentals

BRKENS-1804 3:30PM
The Power of Cisco SD-Access LISP Fabric: Simplified Deployment to Advanced Use Cases - Part 1

BRKENS-1851 4:00PM
Zero Trust: Secure the Workplace with Cisco Software-Defined Access

Tuesday–10th

BRKENS-1805 11:00AM
SD-Access in Action: Trusted Outcomes Across Education and Finance-Featuring UC Riverside & CIBC Bank

BRKENS-2824 2:00PM
Deploying Your First Cisco SD-Access Project

BRKENS-2804 4:00PM
The Power of Cisco SD-Access LISP Fabric: Simplified Deployment to Advanced Use Cases - Part 2

IBOENS-2828 4:30PM
Network Quest: Exploring Campus Fabrics and Secure Segmentation

Wednesday–11th

BRKENS-2816 10:30AM
Cisco SD-Access Transit: Advanced Design Principles

IBOENS-2826 10:30AM
Cisco SD-Access Design and Deployment Best Practices

BRKENS-2836 10:30AM
Endpoint profiling and segmentation using AI endpoint Analytics and Cyber Vision for next generation SD Access manufacturing plants

BRKENS-1806 1:00PM
Transforming Enterprise Networks with Cisco SD-Access: Real-World Strategies from CDW

BRKENS-3826 3:30PM
Advanced LISP SD-Access Forwarding Architecture

Thursday–12th

BRKENS-2650 8:30AM
Designing and Deploying Cisco SD-Access with BGP EVPN

BRKENS-2700 8:30AM
Fabric Networking in the Campus: What's the fuss and what are the choices?

BRKENS-3834 10:30AM
1 to 100: Master All Steps of Automated and Seamless Deployment, Integration, and Migration of Large SDA and SD-WAN Networks

BRKENS-3810 2:30PM
How to Adopt Zero Trust using SD-Access and Default-Deny without Tears

BU-led sessions

Cisco Catalyst Programmability Sessions at Cisco Live San Diego 2025

Sunday June 8

9:00 AM – 1:00 PM

TECOPS-2314
Automating All Things YANG, All the time - Programmability and Automation 101 with Cisco IOS XE

Monday June 9

1:00 PM – 2:00 PM

BRKENS-2604
Atomic Config Replace with Cisco Catalyst 9000

3:30 PM – 5:00 PM

BRKOPS-1401
Cisco IOS XE: Telemetry, Automation, and YANG—Oh My!

Tuesday June 10

11:30 AM - 1:15 PM

DEVNET-1110
Modern approaches for IOS XE network device management on Cat9k

1:30 PM – 2:30 PM

BRKDEV-2017
gRPC, gNMI, gNOI... Oh My! An Enterprise Network Automation Journey

2:00 PM – 3:30 PM

BRKOPS-2223
Network of the Future is Here - Let's Automate your IPv6 deployment with Python!

Wednesday June 11

4:00 PM - 4:30 PM

CISCOU-1059
Observability at TorIX: custom telemetry solutions on next-gen campus switching


May 16, 2025 Leave a Comment

Developer
Explore Cisco IOS XE Automation at Cisco Live US 2025
7 min read
Story DeWeese

Cisco IOS XE offers advanced programmability and automation capabilities that simplify network management and reduce complexity. By utilizing open, standards-based APIs such as NETCONF, RESTCONF, and gNMI, network administrators can efficiently provision and configure devices, minimizing manual errors. The integration of Zero-Touch Provisioning with DevOps toolkits accelerates device onboarding, while model-driven telemetry provides real-time data collection for enhanced monitoring and troubleshooting. Cisco's automation solutions are extensible across the entire device lifecycle, supporting seamless software upgrades and patching. These features allow our customers to drive business outcomes and be at the cutting edge of network automation, enabling faster, more secure, and efficient network operations.

Session Levels:
Beginner
Intermediate

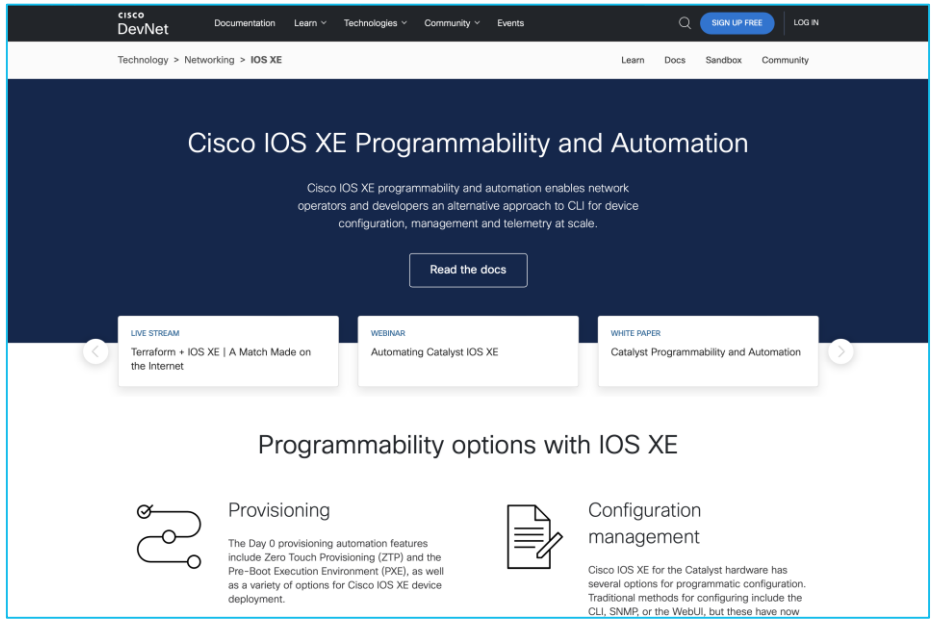
https://blogs.cisco.com/developer/cisco-ios-xe-automation-clus25

Walk-in labs (open Monday-Thursday)!

- ✓ Efficiently monitoring device statistics in real-time using gRPC Dial-out with IOS XE [LABPRG-2004]
- ✓ Explore and test YANG models for model driven telemetry on IOS XE with Cisco's YANG Suite [LABOPS-2000]
- ✓ Hands-On Lab: Monitoring Cisco IOS-XE Devices with RESTCONF [LABDEV-2001]

Programmability Website

The one-stop-shop for Cisco IOS XE Programmability resources including videos, white papers, labs and more!



- Community Forum
- IOS XE FAQ
- White Papers
- Code Exchange
- IOS XE Docs & Guide
- Learning Tracks and Labs
- Sandboxes
  - ... and more !

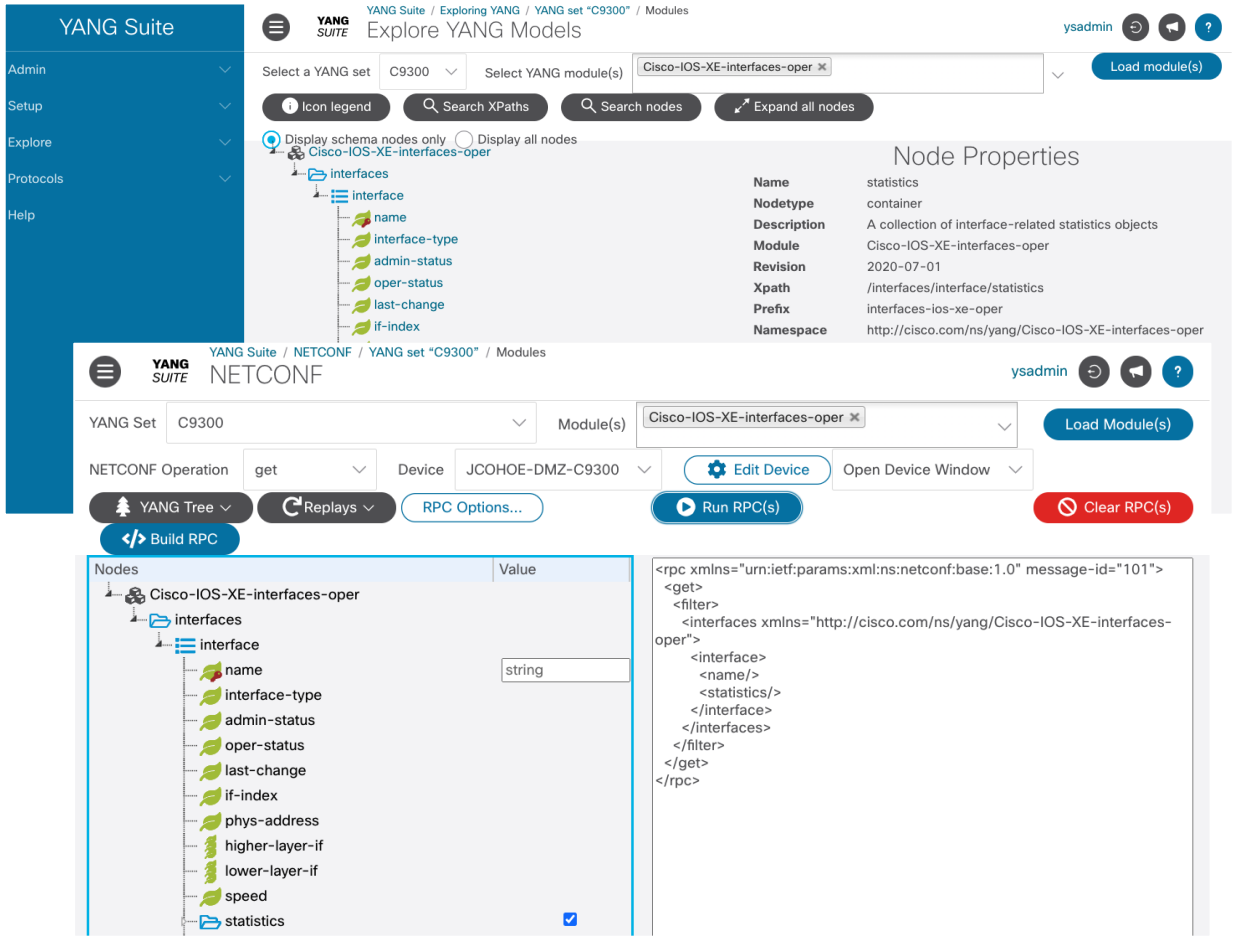
https://developer.cisco.com/iosxe/

Cisco YANG Suite

YANG API Testing and Validation Environment

Construct and test YANG based APIs over NETCONF, RESTCONF, gRPC and gNMI

IOS XE / IOS XR / NX OS platforms



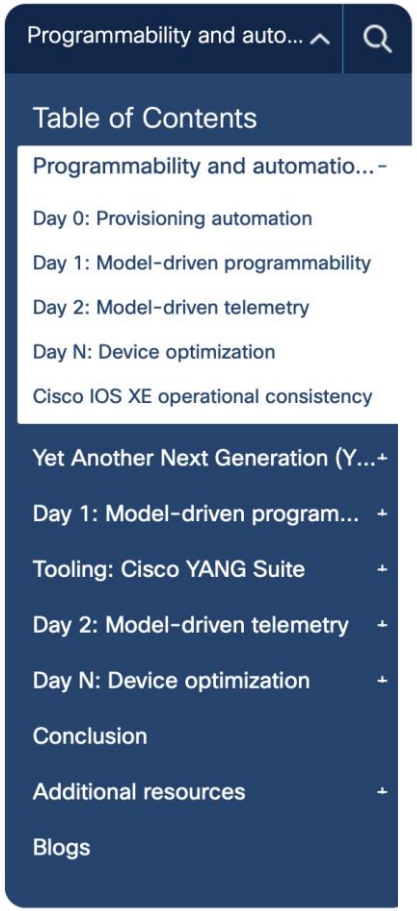
developer.cisco.com/yangsuite

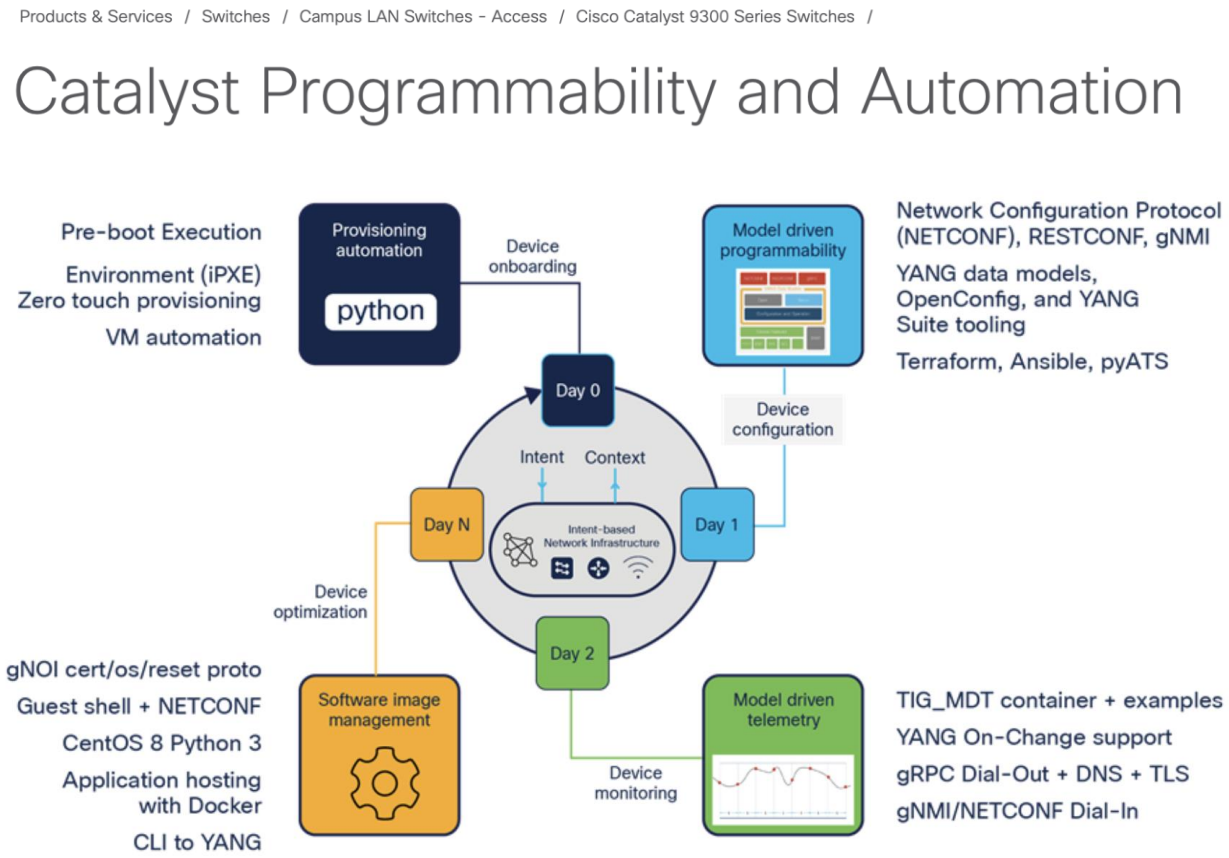
github.com/CiscoDevNet/yangsuite

On-Demand Learning Lab with YANG Suite in Docker
https://developer.cisco.com/learning/labs/intro-yangsuite/

API White Paper

http://cs.co/apiwp





http://cs.co/apiwppdf

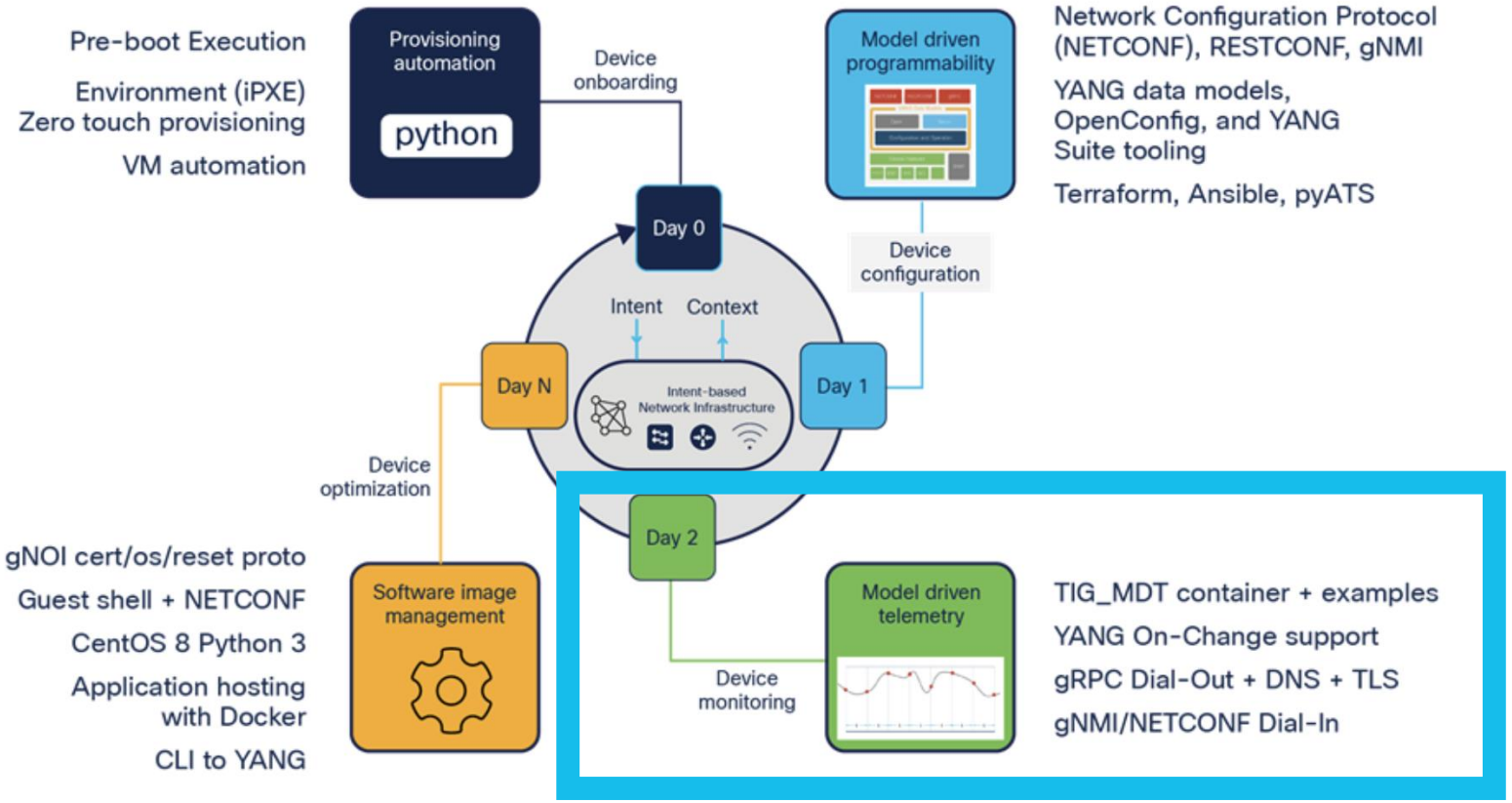
https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-9300-series-switches/nb-06-catalyst-programmability-automation-wp.html

https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-9300-series-switches/nb-06-catalyst-programmability-automation-wp.pdf

https://www.youtube.com/watch?v=LdcK5PnPu2I

Model Drive Telemetry (MDT) White Paper

The Model Driven Telemetry White Paper includes examples, use cases and tooling related to telemetry. This paper is now available online and in PDF form!



View online: https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-9300-series-switches/model-driven-telemetry-wp.html
View as PDF: https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-9300-series-switches/model-driven-telemetry-wp.pdf

Introduction to Cisco IOS ...

Table of Contents

dCloud Programmability

https://dcloud.cisco.com
"Cisco Catalyst 9000 IOS XE Programmability & Automation Lab v1"
https://dcloud2.cisco.com/demo/catalyst-9000-ios-xe-programmability-automation-lab-v1

Use Cases:

EVPN:

Ansible with CLI deployment of EVPN solutions
EVPN management over RESTCONF/YANG with Postman
Declarative EVPN fabric management with Terraform

Model Driven Telemetry

Telemetry configuration with CLI and YANG Suite
Collection with TIG_MDT container and tooling

YANG Programmability

YANG Suite tooling and integrations to YANG API's
Ansible integrations

Tooling and Integrations

YANG Suite
- NETCONF/RESTCONF/gNMI API
  - Ansible integration
- NETCONF/gNMI Dial-In Telemetry
- gRPC Dial-Out Telemetry receiver

Telemetry
- TIG stack in Docker
- Grafana dashboard for device health

Postman / RESTCONF
- EVPN fabric API calls

Terraform/RESTCONF
- Declarative EVPN fabric management

Ansible
- EVPN solution enablement using CLI

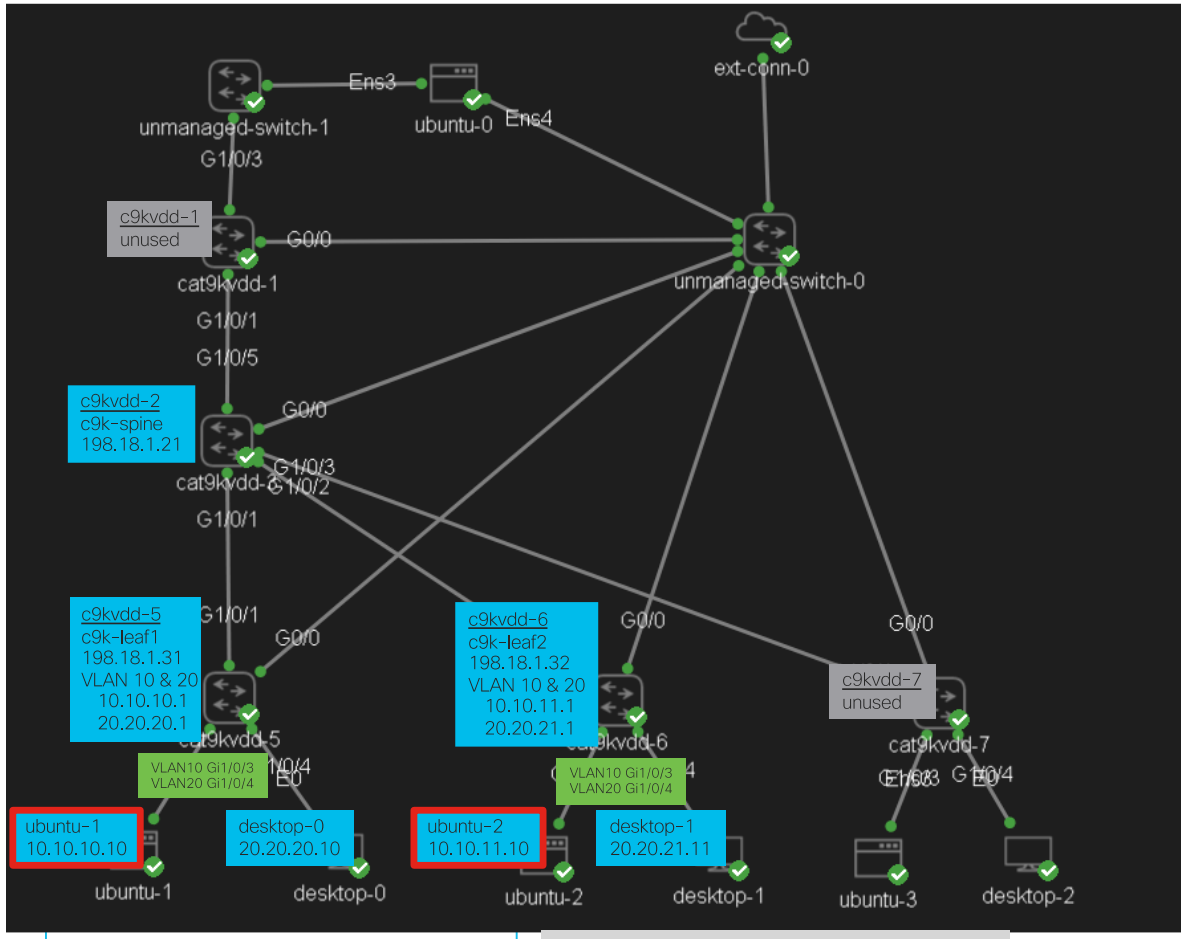
Ubuntu VM Details:

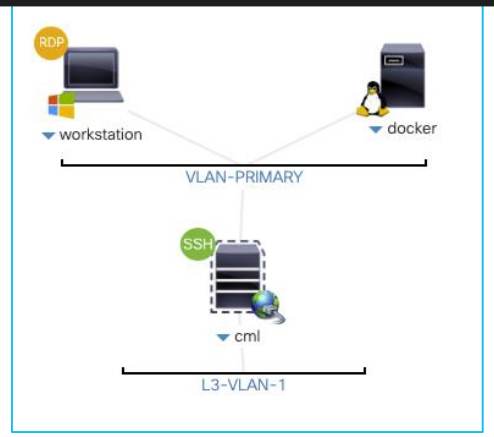
Syslog receiver from all switches
TFTP config backup
See slide

Windows VM Details

VS Code
Terraform @ folder
Ansible @ folder
Chrome browser
YANG Suite, Grafana
Bash/PS/Cmd shells
SSH into C9K or Ubuntu
Postman
Workspace for EVPN

C9K VM's





VLAN1
c9k-spine
IP: 198.18.1.21
developer /
C1sco12345
c9k-leaf1
IP: 198.18.1.31
developer /
C1sco12345
c9k-leaf2
IP: 198.18.1.32
developer /
C1sco12345
c9kvdd-1 - unconfigured

c9kvdd-7 - unconfigured

DevNet Sandbox – overview for Campus and Enterprise

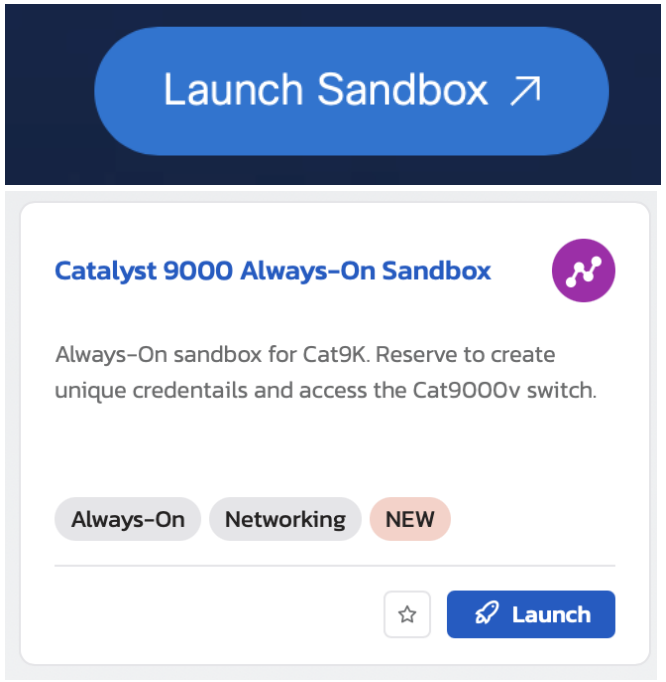
https://developer.cisco.com/site/sandbox/

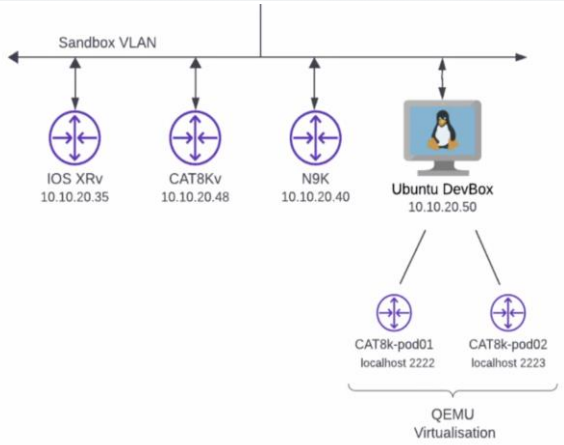
1. Reservable Physical: C9200, C9300, C9300X including stacks
   About to go into production April 2025
   Usecases: Application Hosting, Power telemetry, etc
2. Reservable Virtual: C8KV Router + NX + XR + Ubuntu VM
   Usecases: Enterprise topology, dual-ZTP
3. Always-On: C9KV
   Usecase: Virtual switch for basic config validation usecases
   DNS: devnetsandboxiosxec9k.cisco.com
   "Launch Sandbox" to get login credentials
4. Always-On: C8KV
   DNS is devnetsandboxiosxe.cisco.com
5. Reservable Catalyst Center (Physical & Virtual)
   CML and C9KV, ISE for SDA

C9KV user is priv15 and has full permissions, there is reset automation for when you break it now too ☺

Additional enablement labs:

1. YANG Suite "Learning Lab 2.0"
   Interactive guide with tool running in Docker container
   "YANG Suite as a service"
2. dCloud Programmability Lab
   EVPN topology with all programmability features enabled





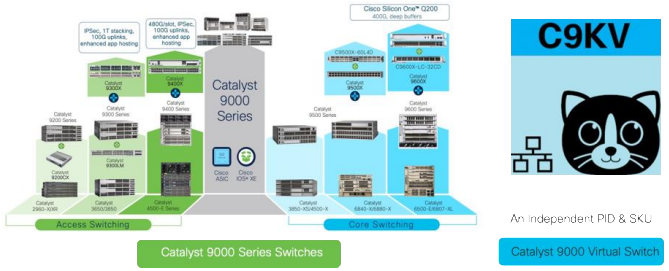
C9KV Always-On with reservation

New C9kV Sandbox now available!

Sandbox access: https://devnetsandbox.cisco.com/DevNet/catalog/Cat9k-Always-On_cat9k-always-on
Hostname: devnetsandboxiosxec9k.cisco.com

C9KV Sandbox Capabilities:

- Cisco IOS XE 17.15.1 Virtual Catalyst 9000 UADP 8 port switch
- “Always-On” outside of the Cisco Network DMZ in a colocation on hardware in VMWARE/OVA
- Accessible with reservation though Cisco’s Developer Enablement platform
- Reservable by anybody including customers, partners and external
- Enabled for **read-only usecases** with SSH/CLI and API: NETCONF/YANG, RESTCONF, gNMI
- Support for Model-Driven Telemetry and basic configuration changes through the API
- Supports 40 concurrent sessions




```
sdeweese@SDEWEESE-M-C20V ~ % ssh sdeweese@devnetsandboxiosxec9k.cisco.com
(sdeweese@devnetsandboxiosxec9k.cisco.com) Password:

CAT9k_AO#
CAT9k_AO#
CAT9k_AO#
CAT9k_AO#
```

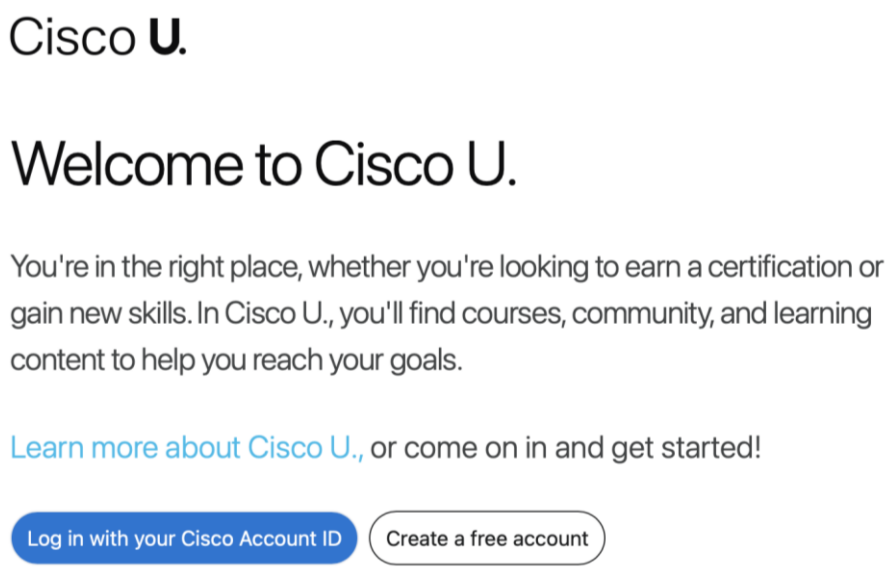
```
CAT9k_AO#sh inv
NAME: "Switch 1", DESCR: "Catalyst 9000 UADP 8 Port Virtual Switch"
PID: C9KV-UADP-8P      , VID: V01  , SN: 98DVJUONW1X

NAME: "Switch 1", DESCR: "Catalyst 9000 UADP 8 Port Virtual Switch"
PID: C9KV-UADP-8P      , VID: V01  , SN: 98DVJUONW1X


CAT9k_AO#sh ver
Cisco IOS XE Software, Version 17.15.01
```

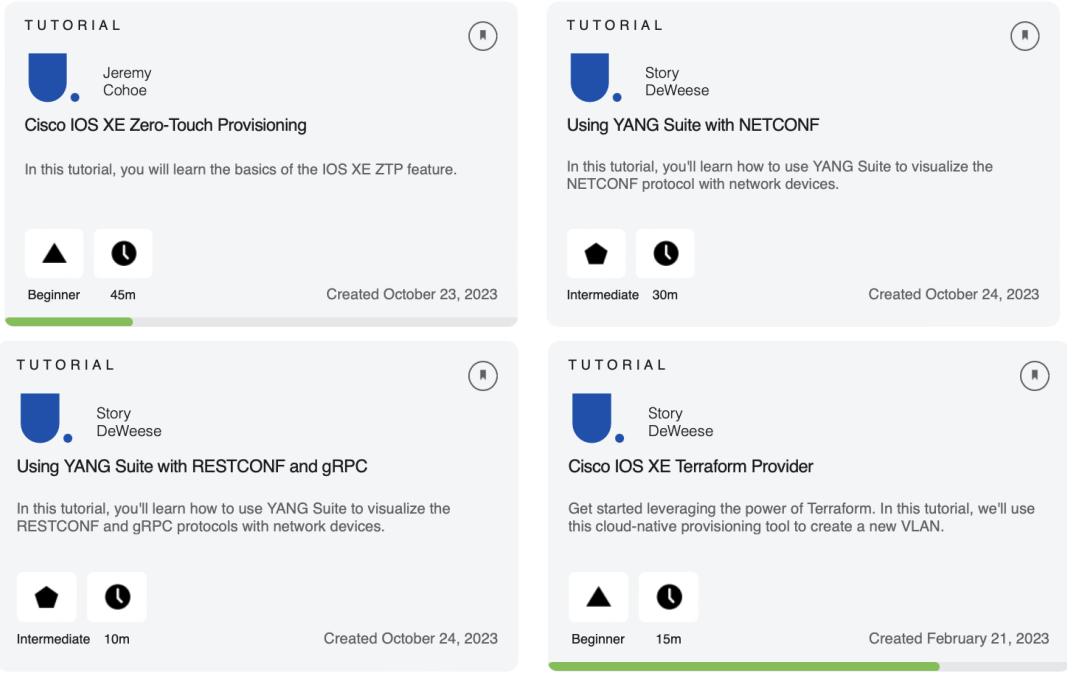
Cisco University (Cisco U) part of L&D

https://u.cisco.com
https://u.cisco.com/search/tutorial?query=Story%20DeWeese,%20Jeremy%20Coho,%20not%20berry

Direct link to Tutorial, requires login to u.cisco.com first:

1. https://ondemandelearning.cisco.com/apollo-alpha/tc-iosxe-ztp/pages/1
2. https://ondemandelearning.cisco.com/apollo-alpha/tc-terraform-ios-xe/pages/1
3. https://ondemandelearning.cisco.com/apollo-alpha/tc-yangsuite-netconf/pages/1
4. https://ondemandelearning.cisco.com/apollo-alpha/tc-yangsuite-restconf/pages/1





Complete Your Session Evaluations

Complete a minimum of 4 session surveys and the Overall Event Survey to be entered in a drawing to win 1 of 5 full conference passes to Cisco Live 2026.

Earn 100 points per survey completed and compete on the Cisco Live Challenge leaderboard.

Level up and earn exclusive prizes!

Complete your surveys in the Cisco Live mobile app.

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

Questions

Thank you

cisco Live !

CISCO