

ALL
IN

CISCO *Live!*

DevNet Zone

#CiscoLiveAPJC



The bridge to possible

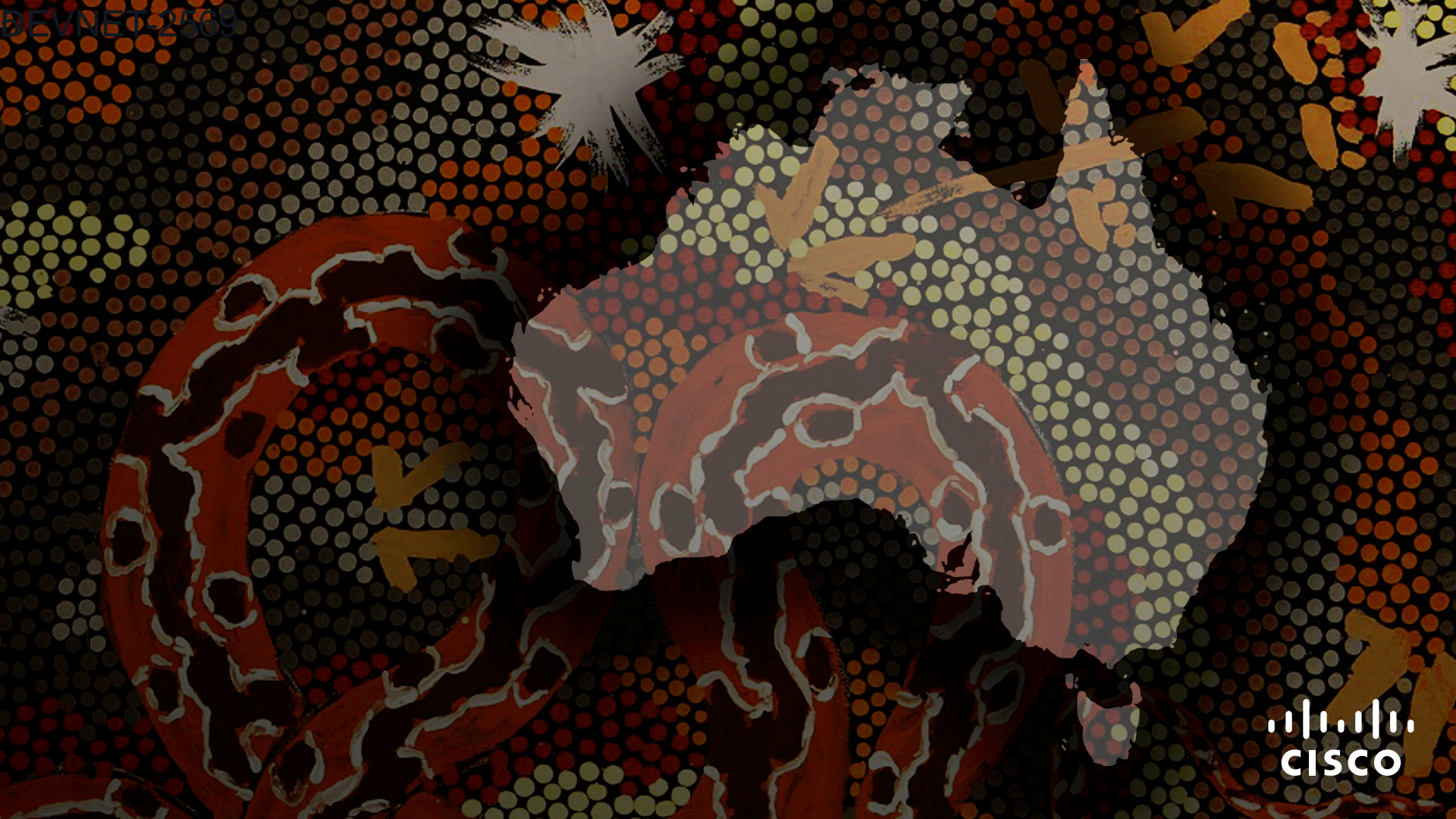
Automating Enterprise Deployments with Zero Touch Provisioning

Story DeWeese, Technical Marketing Engineer
@StoryDeWeese
DEVNET-2569



DevNet Zone

#CiscoLiveAPJC



Cisco Webex App

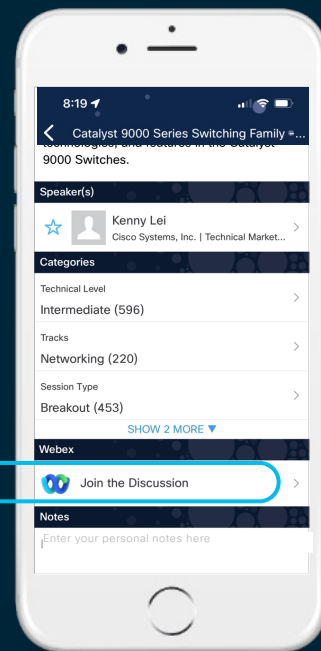
Questions?

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

How

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

Webex spaces will be moderated by the speaker until Thursday 22 December, 2022.



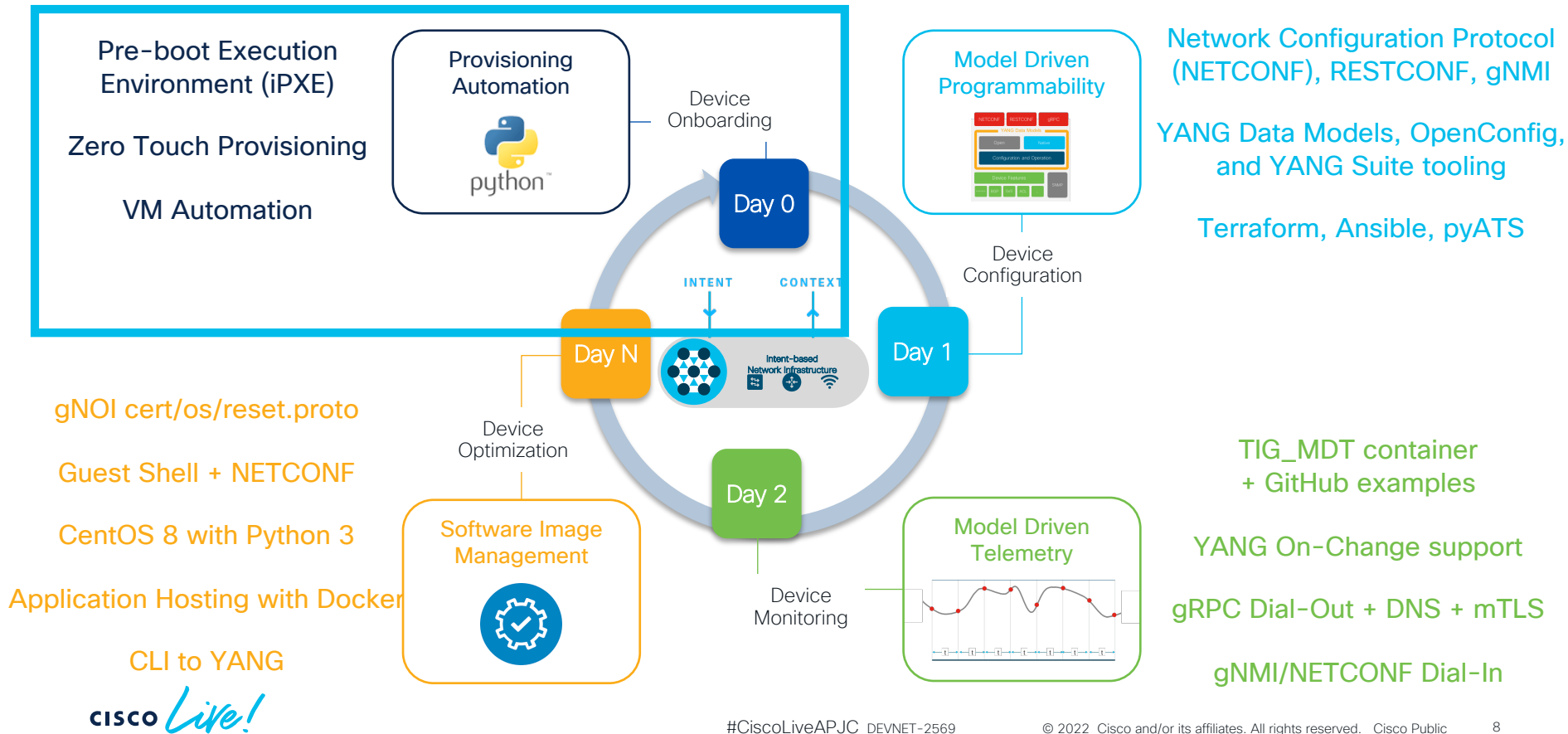
<https://ciscolive.ciscoevents.com/ciscolivebot/#DEVNET-2569>



Agenda

- YANG API Introduction
- ZTP in Action
- What's ZTP?
- ZTP Demo
- Resources

Cisco IOS XE Programmability & Automation Lifecycle



Programmable Interfaces

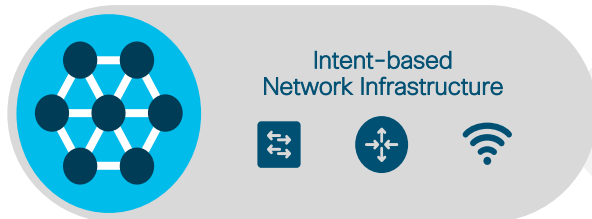
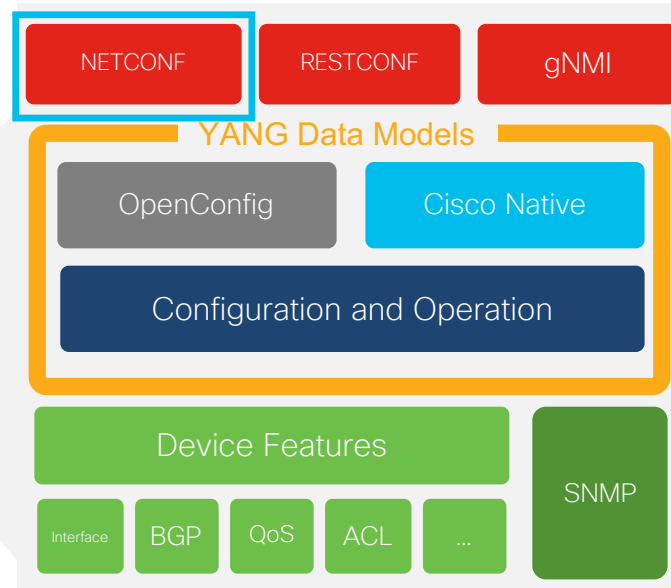
CLI

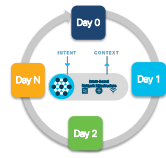
SNMP

WebUI

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

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

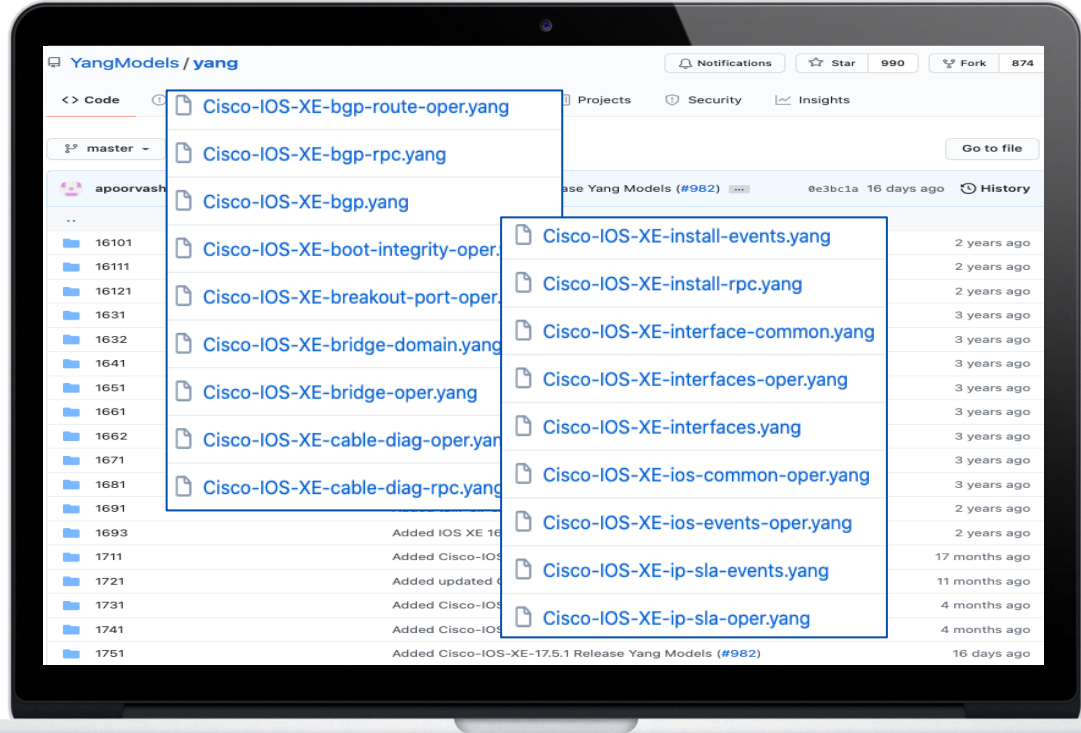




IOS XE - YANG Model Coverage on GitHub

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

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



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

ZTP in Action

ZTP in Action

Start the hands-on lab for Day 0



Module: Zero Touch Provisioning

In this module, you will verify and confirm the prerequisites for Zero Touch Provisioning (ZTP), the feature of IOS XE on the Catalyst 9300 switch. At the end of this module, you will issue the 'write erase' command, reload the switch, and watch as the ZTP process completes and the switch is configured programmatically and automatically.

What is ZTP? When a device that supports Zero-Touch Provisioning boots up, and does not find the startup configuration (during initial installation), the device enters the Zero-Touch Provisioning mode. The device searches for an IP from a DHCP server and bootstraps itself by enabling the Guest Shell. The device then obtains the IP address or URL of an HTTP/TFTP server, and downloads a Python script from an server to configure the device.

```
write erase  
reload
```

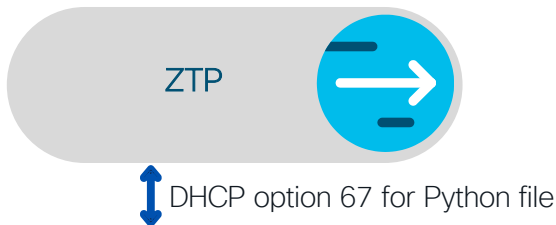
ping

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

What's ZTP?

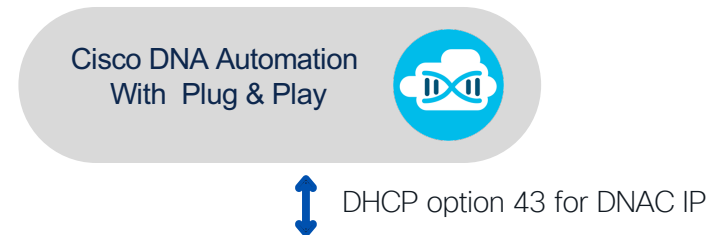
Controller Based Day 0 with Cisco DNA Center

Zero Touch Provisioning controller-less

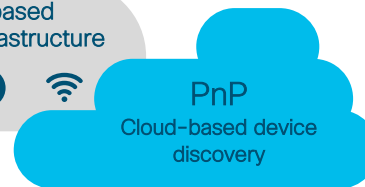


Python Configuration File + Guest Shell
Available with TFTP or HTTP

Plug and Play with Cisco DNA Center



Claim and Manage in DNAC GUI



ZTP Overview

1. When a Cisco IOS XE device boots and no configuration is present, the device will issue a DHCP request on the management port and on the front panel port.
2. If the DHCP response contains option 67 then ZTP is initiated and the device will retrieve and execute the Python script from within the Guest Shell
3. Guest Shell is started and networking is automatically configured

<https://www.youtube.com/watch?v=EAXnftG6odg>

<https://blogs.cisco.com/developer/device-provisioning-with-ios-xe-zero-touch-provisioning>

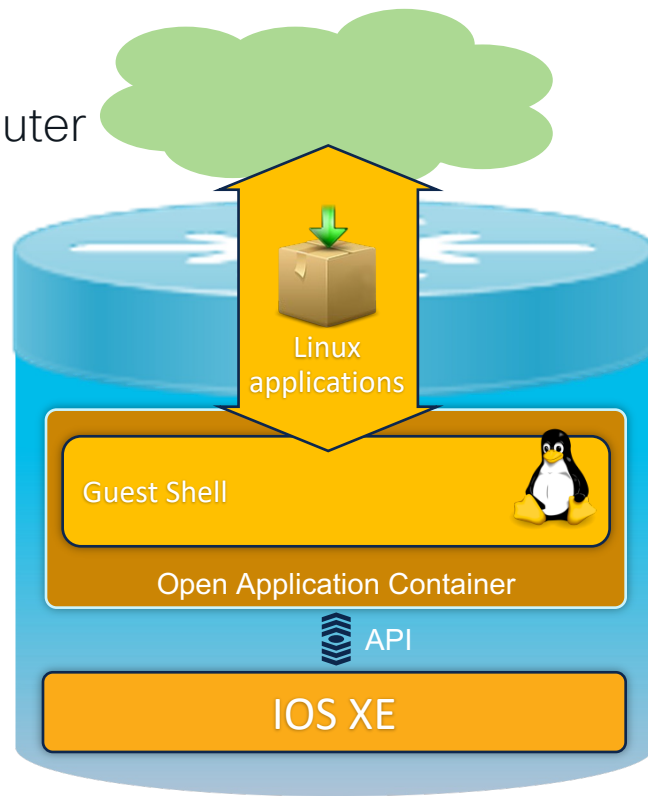
<https://devnetsandbox.cisco.com/RM/Diagram/Index/f2e2c0ad-844f-4a73-8085-00b5b28347a1?diagramType=Topology>

The image shows two overlapping web browser windows. The background window is a Cisco Blog post titled 'Automate Device Provisioning with Cisco IOS XE Zero Touch Provisioning' by a developer. It features an illustration of a person in a server room. The foreground window is the 'DEVNET' interface, specifically the 'SANDBOX LABS' section. It has tabs for 'INSTRUCTIONS', 'COMMANDS', and 'NAVIGATOR'. Under the 'INSTRUCTIONS' tab, there are buttons for 'IOS XE on CSR (Latest)', 'VPN Access', and 'DevBox Info'. The 'IOS XE on CSR (Latest)' button is selected, displaying text that describes the sandbox environment: 'The IOS XE on CSR Latest Code Sandbox offers developers access to an IOS XE device running the latest IOS XE code release (currently 17.3.1). This lab allows you to provision and test a programmability feature, ZTP (Zero Touch Provisioning). A Devbox (running KVM) provides serial interface to a CSR1000v for validation and testing ZTP. In addition, a standalone CSR1000v with SSH access has been provided to validate other features.'

Guest Shell Application

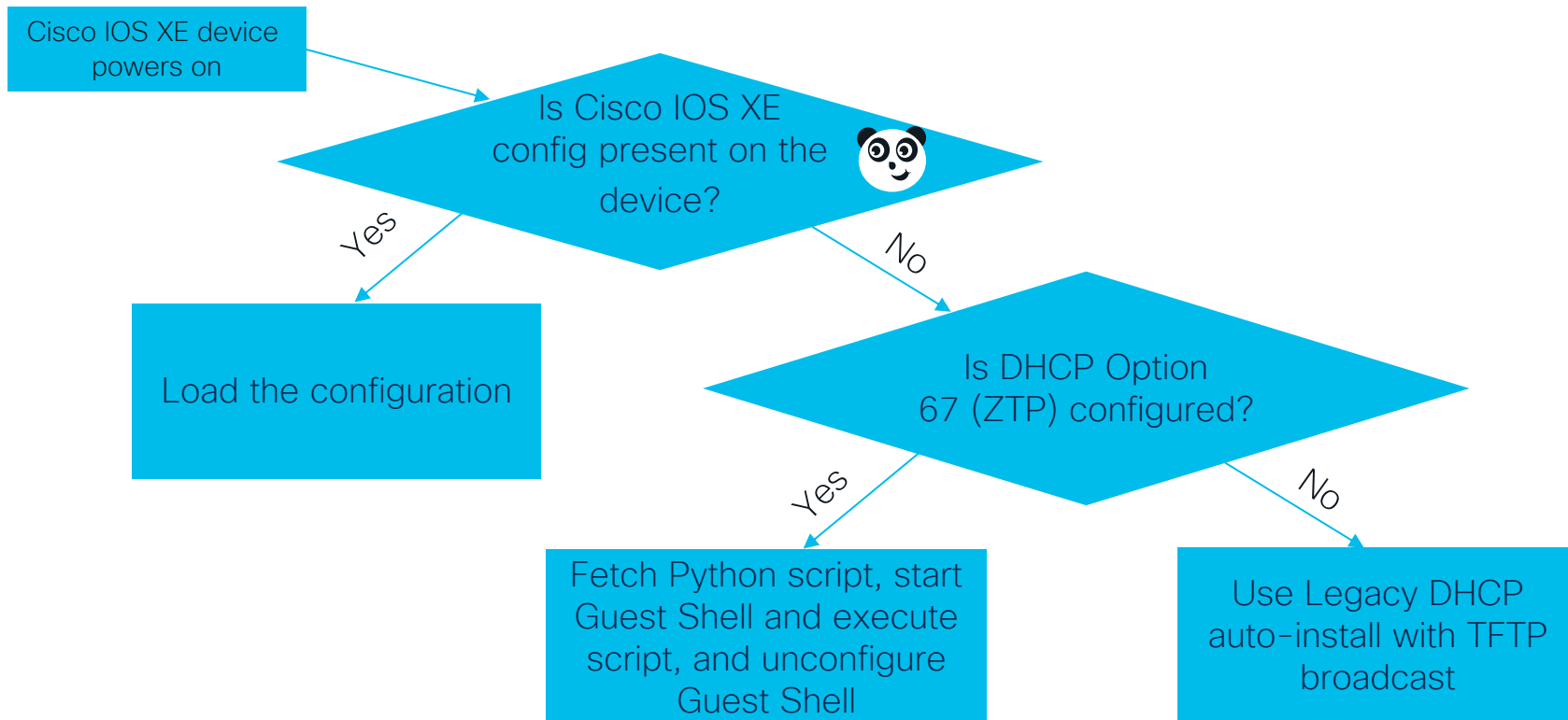
Linux Shell Environment On Your Switch or Router

- Maintain IOS-XE system integrity
 - Isolated User Space
 - Fault Isolation
 - Resource Isolation
- On-box rapid prototyping
 - Device-level API Integration
 - Scripting (Python)
 - Linux Commands
- Application Hosting
- Integrate into your Linux workflow
- Integrated with IOS-XE



The Python API within Guest Shell
enables complete device configuration

Day 0 device onboarding workflow



Python API between Guest Shell and Cisco IOS XE

Each Python module below takes a Cisco IOS XE command as an argument

Function	Description
<code>cli.cli(command)</code>	<u>runs the command</u> through the IOS parser and returns the resulting text.
<code>cli.clip(command)</code>	<u>runs the command</u> through the IOS parser and prints the resulting text to <i>stdout</i> (rather than returning it).
<code>cli.execute(command)</code>	<u>executes a single EXEC command</u> and returns the output. No semicolons or newlines are allowed as part of this command. Use a Python list with a for-loop to execute this function more than once.
<code>cli.executep(command)</code>	<u>executes a single EXEC command</u> and prints the resulting text to <i>stdout</i> (rather than returning it). No semicolons or newlines are allowed as part of this command. Use a Python list with a for-loop to execute this function more than once.
<code>cli.configure(command)</code>	<u>configures the device</u> and returns a list of named tuples that contains the command and its result
<code>cli.configurep(command)</code>	<u>configures the device</u> and prints a list of named tuples that contains the command and its result to <i>stdout</i> (rather than returning it).

ZTP Example using Python Modules

```
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 XXXXXXXXXXXXXXXX"])
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"
```

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.

ZTP

CLI API
Enable NETCONF
Enable AAA

NETCONF API
Preform RPC Actions
Programmatic Configuration
of device features

```
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>
```

```
[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>

ZTP Demo

Summary ZTP Workflow



1. DHCP Option 67

```
$ cat /etc/dhcp/dhcpd.conf

# ZTP option 67 example
option bootfile-name "http://10.1.1.3/ztp-simple.py";
# *** This maps to /var/www/html/ztp-simple.py
```

2. ZTP.py File

```
$ cat /var/www/html/ztp-simple.py

hostname = 'c9300-pod22'
cli.configurep(['hostname {}'.format(hostname)])
```

3. Trigger ZTP using the following commands

```
Device# write erase
Device# reload
```

4. Success

```
$ ping 10.1.1.5
From 10.1.1.3 icmp_seq=261 Destination Host Unreachable
64 bytes from 10.1.1.5: icmp_seq=262 ttl=254 time=15.8 ms
```

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

Option 67 DHCP.conf

```
$ cat /etc/dhcp/dhcpd.conf
default-lease-time 600;
max-lease-time 7200;
ddns-update-style none;
authoritative;

# Pod Network
# interface has IP 10.1.1.x
subnet 10.1.1.0 netmask 255.255.255.0 {
range 10.1.1.150 10.1.1.159;
option domain-name "ios-xe-programmability-lab.cisco.com";
option domain-name-servers 128.107.212.175;
option subnet-mask 255.255.255.0;
option broadcast-address 10.1.1.255;
option routers 10.1.1.3;
option ntp-servers 10.1.1.3;
default-lease-time 600000;
max-lease-time 720000;

# ZTP option 67 example
option bootfile-name "http://10.1.1.3/ztp-simple.py";
# *** This maps to /var/www/html/ztp-simple.py
```

- ZTP requires DHCP option 67 to be configured
- Configure DHCP option 67 using “option bootfile-name” and specify where the Python bootfile can be found

ZTP.py example

```
$ cat /var/www/html/ztp-simple.py
```

```
hostname = 'c9300-pod22'
cli.configurep(['hostname {}'.format(hostname)])
ID = hostname[-2:]
vlan_num = str(int(ID) + 20)
cli.configurep(["vlan " + vlan_num, "end"])
cli.configurep(["int vlan 1", "no ip address", "end"])
cli.configurep(["int vlan " + vlan_num, "ip address 10.1.1.5 255.255.255.0", "end"])
cli.configurep(["int range gi1/0/1 - 24", "switchport access vlan " + vlan_num, "end"])
cli.configurep(["ip route 0.0.0.0 0.0.0.0 10.1.1.3", "end"])
# AAA
cli.configurep(["username admin privilege 15 secret 0 Cisco123"])
cli.configurep(["aaa new-model", "aaa authentication login default local", "end"])
cli.configurep(["aaa authorization exec default local", "aaa session-id common", "end"])
# TCP settings
cli.configurep(["ip tftp blocksize 8192", "end"])
cli.configurep(["ip tcp window-size 65535", "end"])
cli.configurep(["ip http client source-interface GigabitEthernet1/0/24", "end"])
# Create loopback0
cli.configurep(["interface Loopback0", "ip address 192.168.12.1 255.255.255.0", "end"])
# Enable NETCONF-YANG API
cli.configurep(["netconf-yang", "end"])
# NETCONF from Guestshell
cli.configurep(["netconf-yang ssh local-vrf guestshell enable", "end"])
# Enable gNMI secure API
cli.configurep(["gnxi", "gnxi secure-init", "gnxi secure-server", "end"])
```

Python file
containing
configuration
to apply to
device

Trigger ZTP with 'write erase' & 'reload'

```
Switch# write erase
```

```
*****  
Erasing Nvram will not clear license trust code.  
*****
```

```
Erasing the nvram filesystem will remove all configuration files! Continue? [confirm]  
[OK]
```

```
Erase of nvram: complete
```

```
*Dec  5 00:49:05.790: %SYS-7-NV_BLOCK_INIT: Initialized the geometry of nvram
```

```
Switch# reload
```

```
System configuration has been modified. Save? [yes/no]: no
```

```
Reload command is being issued on Active unit, this will reload the whole stack
```

```
Proceed with reload? [confirm]
```

```
*Dec  5 00:49:16.278: %SYS-5-RELOAD: Reload requested by console. Reload Reason: Reload Command.  
Chassis 1 reloading, reason - Reload command
```

```
<snip>
```

```
No startup-config, starting autoinstall/pnp/ztp...
```

```
Autoinstall will terminate if any input is detected on console
```

Verify ZTP.py Successfully ran

```
From 10.1.1.3 icmp_seq=255 Destination Host Unreachable
From 10.1.1.3 icmp_seq=256 Destination Host Unreachable
From 10.1.1.3 icmp_seq=257 Destination Host Unreachable
From 10.1.1.3 icmp_seq=258 Destination Host Unreachable
From 10.1.1.3 icmp_seq=259 Destination Host Unreachable
From 10.1.1.3 icmp_seq=260 Destination Host Unreachable
From 10.1.1.3 icmp_seq=261 Destination Host Unreachable
64 bytes from 10.1.1.5: icmp_seq=262 ttl=254 time=15.8 ms
64 bytes from 10.1.1.5: icmp_seq=263 ttl=254 time=0.919 ms
64 bytes from 10.1.1.5: icmp_seq=264 ttl=254 time=0.807 ms
64 bytes from 10.1.1.5: icmp_seq=265 ttl=254 time=0.639 ms
64 bytes from 10.1.1.5: icmp_seq=266 ttl=254 time=0.764 ms
64 bytes from 10.1.1.5: icmp_seq=267 ttl=254 time=0.772 ms
64 bytes from 10.1.1.5: icmp_seq=268 ttl=254 time=2.58 ms
64 bytes from 10.1.1.5: icmp_seq=269 ttl=254 time=0.669 ms
64 bytes from 10.1.1.5: icmp_seq=270 ttl=254 time=0.788 ms
```

Once the device is reachable, the ZTP process has been successfully completed!

Customer Use Cases

ZTP customer use case - workflow

- Customer problem statement: How can we reduce the manual work that takes time which increases costs and risk when deploying networks at various locations globally ?
- Zero Touch Provisioning and automated onboarding solution for multiple devices in multiple locations
 - (retail, warehouse, corporate branch and campus)
- DHCP connectivity required and new devices are upgraded, configured, and available for management without any manual interaction – 100's – 1000's of devices deployed and configured with zero touch



Day 0
workflow

option 67 for ZTP python script

Python3 script + Guest Shell
Software upgrade + EEM
Download and apply pre-generated config

option 150 for AutoInstall CLI config

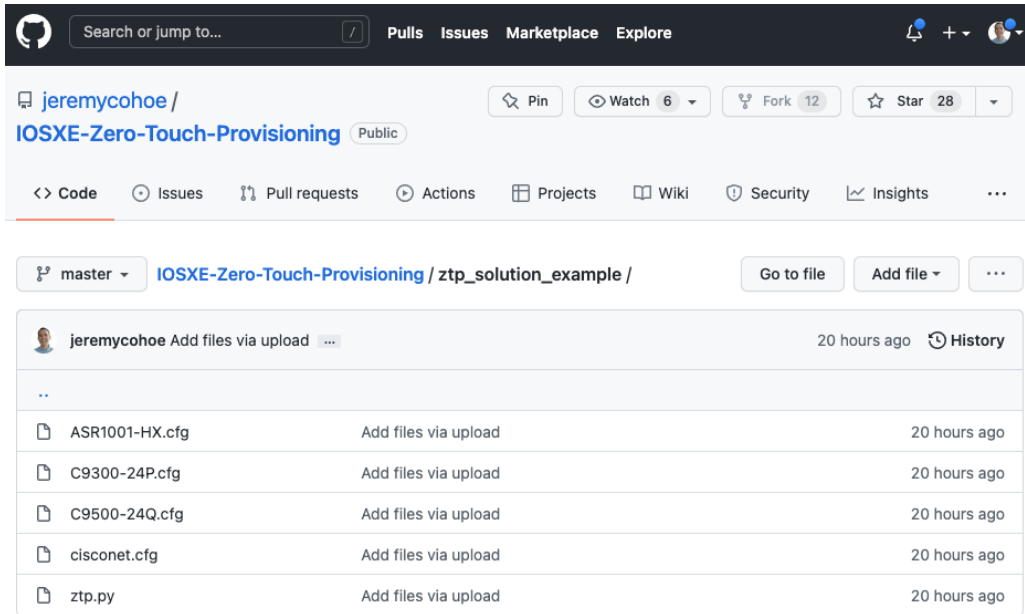
Auto Install CLI config
Set basic device config

IOS XE will use the option 67 for ZTP first then the option 150 for Auto Install

Auto Install is the “backup” for ZTP

ZTP customer use case – code samples

<https://developer.cisco.com/codeexchange/github/repo/jeremycohoe/IOSXE-Zero-Touch-Provisioning/>



The screenshot shows the GitHub repository page for 'jeremycohoe / IOSXE-Zero-Touch-Provisioning'. The repository is public and has 6 watchers, 12 forks, and 28 stars. The 'Code' tab is selected, showing the file structure. The 'master' branch is selected, and the file path is 'IOSXE-Zero-Touch-Provisioning / ztp_solution_example /'. A table lists the files in the directory:

File	Action	Time
ASR1001-HX.cfg	Add files via upload	20 hours ago
C9300-24P.cfg	Add files via upload	20 hours ago
C9500-24Q.cfg	Add files via upload	20 hours ago
cisconet.cfg	Add files via upload	20 hours ago
ztp.py	Add files via upload	20 hours ago



```
1
2 # Importing cli module
3 from cli import configure, cli, configurep, executep
4 import re
5 import time
6 import urllib
7 import sys
8 import logging
9 import os
10 from logging.handlers import RotatingFileHandler
11 import subprocess
12
13 software_mappings = {
14     'C9300-24P': {
15         'software_image': 'cat9k_iosxe.17.06.01.SPA.bin',
16         'software_version': '17.06.01',
17         'software_md5_checksum': 'fdb9c92bae37f9138d0ee6761afe2919'
18     },
19     'C9500-24Q': {
20         'software_image': 'cat9k_iosxe.17.06.01.SPA.bin',
21         'software_version': '17.06.01',
22         'software_md5_checksum': 'fdb9c92bae37f9138d0ee6761afe2919'
23     },
24     'ASR1001-HX': {
25         'software_image': 'asr1000-universalk9.17.05.01a.SPA.bin',
26         'software_version': '17.05.01a',
27         'software_md5_checksum': '0e4b1fc1448f8ee289634a41f75dc215'
28     }
29 }
30
31 http_server = '10.85.134.66'
32 log_tofile = True
33 release_set = ['17.05', '17.06']
34
35 def main():
36
37     try:
38         print ('##### STARTING ZTP SCRIPT #####\n')
39         # switch to enable/disable persistent logger
40         if(log_tofile == True):
41             filepath = create_logfile()
42             configure_logger(filepath)
```

Resources

Catalyst Programmability and Automation White Paper

Learn more about Catalyst programmable features, tooling and examples covering Day 0 through Day N in the device programmability lifecycle.

Catalyst Programmability and Automation

Updated: October 19, 2022

Bias-Free Language

Table of Contents

Table of Contents

Programmability and automati...

- Day 0: Provisioning automation
- Day 1: Model-driven programmability
- Day 2: Model-driven telemetry
- Day N: Device optimization
- Cisco IOS XE operational consistency

Yet Another Next Generation (...)

- Day 1: Model-driven program...
- Tooling: Cisco YANG Suite
- Day 2: Model-driven telemetry
- Day N: Device optimization

Conclusion

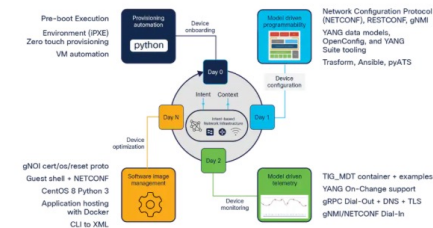
Additional resources

Blogs

Programmability and automation overview

The world of programmability has been evolving for years, and with the latest Cisco IOS XE releases, we've included new Yet Another Next Generation (YANG) models to bring additional automation to wireless technology. With the use of APIs, interacting with devices and retrieving data has gotten much easier. Back in the day, we used many commands sent from command-line interfaces (CLIs) to communicate with the software. In addition, the Simple Network Management Protocol (SNMP) was frequently used for network management. Fast forward to today: We now have a new way to interact with software, commonly called an application programming interface, or API. Even though CLIs and SNMP are widely used, they are not as efficient and scalable as APIs.

This document dives into the different programmable interfaces used to communicate with the various Cisco IOS XE devices, specifically the Cisco Catalyst 9300 Series Switches but also the Catalyst 9800 Series Wireless Controllers and the Catalyst 8000 Edge Platforms Family. We discuss the pros and cons of using Network Configuration Protocol (NETCONF), Representational State Transfer Configuration (RESTCONF), and the gRPC network management interface/Google remote procedure call (gNMI/gRPC) protocols, and the main differences between them.



<https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-9300-series-switches/catalyst-programmability-automation-wp.html#Conclusion?cachemode=refresh>

Cisco IOS XE Programmability – Booksprint Book

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

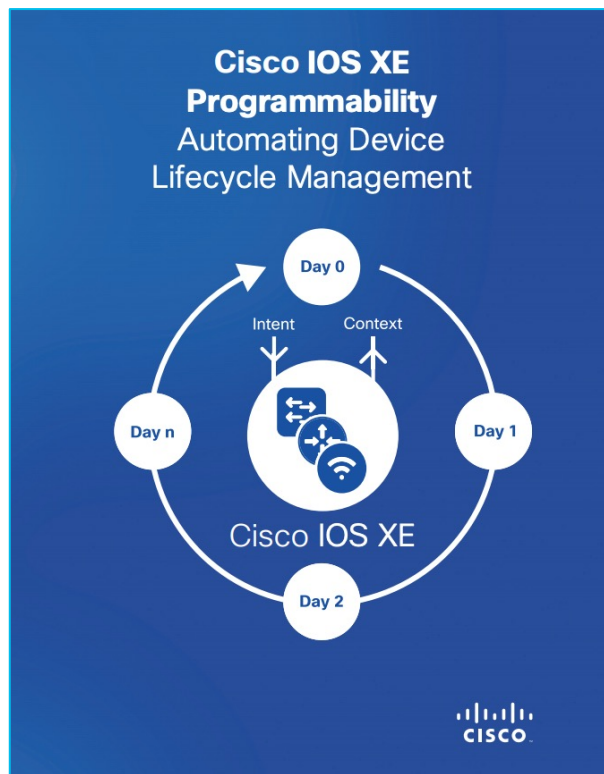


Table of Contents

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

Programmability Configuration Guide

Book Table of Contents

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



Programmability Configuration Guide, Cisco IOS XE Cupertino 17.9.x

First Published: 2022-08-01

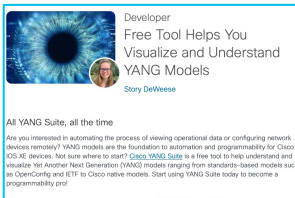
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/179/b_179_programmability_cg.html

YANG Suite Resources

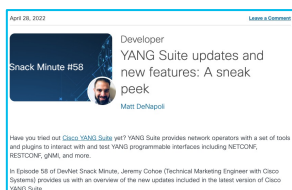
Blogs



<https://blogs.cisco.com/developer/363-yangsuite-01>



<https://blogs.cisco.com/developer/yangallthetime01>

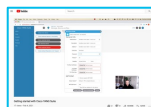


<https://blogs.cisco.com/developer/2022yangsuiteupdatesfeatures01>



<https://blogs.cisco.com/developer/leverageyangsuite01?dtd=os-scdc000283>

YouTube Videos



<https://youtu.be/smrhIL5AyZ0>



<https://www.youtube.com/watch?v=dTun33611JA>



<https://www.youtube.com/watch?v=soyWPf0fJ0s>



<https://www.youtube.com/watch?v=PkbAOzZ1vNk>



<https://www.youtube.com/watch?v=3zmNdn8b38>

Additional Resources

<https://github.com/CiscoDevNet/yangsuite/>

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

<https://eurl.io/#MaW78CeIS> YANG Suite General (external)

Blog and Resources: Terraform

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

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

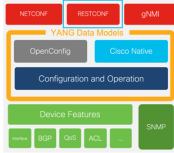
Terraform is...



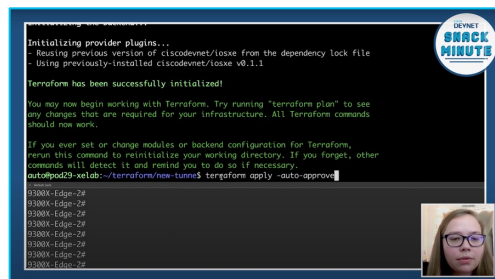
Open-source Infrastructure as Code (IaC) Software Tool providing a consistent CLI workflow to manage hundreds of cloud services. Terraform codifies cloud APIs into declarative configuration files.

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

Terraform uses the RESTCONF API



<https://salesconnect.cisco.com/#/content-detail/fa072157-b099-494b-8ec5-2522c6ab2bf6>



Initializing provider plugins...

- Raising previous version of cisco/devnet/iosxe from the dependency lock file
- Using previously-installed cisco/devnet/iosxe v0.1.1

Terraform has been successfully initialized!

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

If you ever set or change modules or backend configuration for Terraform, run this command to re-initialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.

auto@pod29-xelab:~/terraform/new-tunnel\$ terraform apply -auto-approve

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

Demo Create a Crypto Tunnel Video:

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



Terraform with IOS XE

Story DeWeese
Technical Marketing Engineer (TME)
February, 2022

Intro to IOS XE Terraform Provider Video:
https://www.youtube.com/watch?v=GEY_hyXimBA

Questions? Join the Ask IOS XE
Terraform Provider Webex space:
<https://eurl.io/#PtsT8eJFI>



Introducing Terraform with IOS XE

Code Included

Developer
Automation with Any Tooling on Any Interface

Story DeWeese

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



Users → Apply Terraform Plan Execution File → Provision Terraform Provider → Deploy the required Infrastructure → Cisco Catalyst 9300X IOS XE RESTCONF / YANG

IOS XE's vast, programmable feature set

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

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

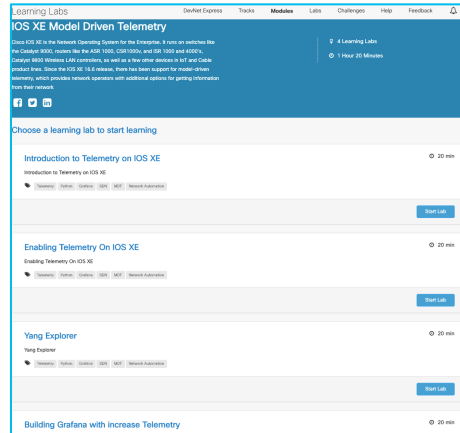
Learning Lab and Blogs: Telemetry

https://developer.cisco.com/learning/modules/iosxe_telemetry

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

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

<https://youtu.be/QwwZakkWBng>



The screenshot shows the Cisco Learning Labs interface. The main heading is 'IOS XE Model Driven Telemetry'. Below it, there's a description: 'Click IOS XE in the Network Operating System for the Enterprise. It runs on switches like the Catalyst 9000, routers like the ASR 1000, CSR1000v, and ISR 1000 and 4000's, Catalyst 9800 Wireless LAN controllers, as well as a few other devices in IoT and Cable product lines. Since the IOS XE 16.6 release, there has been support for model-driven telemetry, which provides network operators with additional options for getting information from their network.' There are 4 Learning Labs listed: 'Introduction to Telemetry on IOS XE' (20 min), 'Enabling Telemetry On IOS XE' (20 min), 'Yang Explorer' (20 min), and 'Building Grafana with Ingested Telemetry' (20 min). Each lab has a 'Start Lab' button.



Developer Explore Model-Driven Telemetry

Stuart Clark

New learning labs and sandbox

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

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

Cisco Blogs



Cisco Blog > Developer



Developer

Enterprise Streaming Telemetry and You: Getting Started with Model Driven Telemetry



Jeremy Cohoe
July 8, 2019 - 3 Comments

Why Streaming Telemetry?

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



Start Now



Videos and
Tutorials



Sandbox
Learning Lab



Automation and
Code Exchange



Learning and
Certifications



Community and
Study Groups

developer.cisco.com

Session Surveys

We would love to know your feedback on this session!

- Complete the session surveys in the Cisco Events mobile app. You'll earn some points in the Cisco Live Game and potentially win a prize.
- Complete a minimum of four session and the overall event surveys to claim a Cisco Live cable bag.

Continue your education



Visit the Cisco Showcase for related demos



Book your one-on-one
Meet the Expert 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

Cisco Learning and Certifications

From technology training and team development to Cisco certifications and learning plans, let us help you empower your business and career. www.cisco.com/go/certs

Pay for Learning with Cisco Learning Credits

(CLCs) are prepaid training vouchers redeemed directly with Cisco.



Learn



Cisco U.

IT learning hub that guides teams and learners toward their goals

Cisco Digital Learning

Subscription-based product, technology, and certification training

Cisco Modeling Labs

Network simulation platform for design, testing, and troubleshooting

Cisco Learning Network

Resource community portal for certifications and learning



Train



Cisco Training Bootcamps

Intensive team & individual automation and technology training programs

Cisco Learning Partner Program

Authorized training partners supporting Cisco technology and career certifications

Cisco Instructor-led and Virtual Instructor-led training

Accelerated curriculum of product, technology, and certification courses



Certify



Cisco Certifications and Specialist Certifications

Award-winning certification program empowers students and IT Professionals to advance their technical careers

Cisco Guided Study Groups

180-day certification prep program with learning and support

Cisco Continuing Education Program

Recertification training options for Cisco certified individuals



The bridge to possible

Programmability and Automation with Catalyst IOS XE Platforms

Wednesday @ 4pm ☺

Story DeWeese, Technical Marketing Engineer
@StoryDeWeese
DEVNET-2745



DevNet Zone

#CiscoLiveAPJC



The bridge to possible

Thank you

CISCO *Live!*

DevNet Zone

#CiscoLiveAPJC

CISCO *Live!*



#CiscoLiveAPJC