

CISCO *Live!*

GO BEYOND

#CiscoLiveAPJC



Secure Zero Touch Provisioning (SZTP) with Cisco Catalyst Switches

DEVNET-1249

Story DeWeese, Technical Marketing
@StoryDeWeese
DEVNET-1249

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 November 15, 2024.



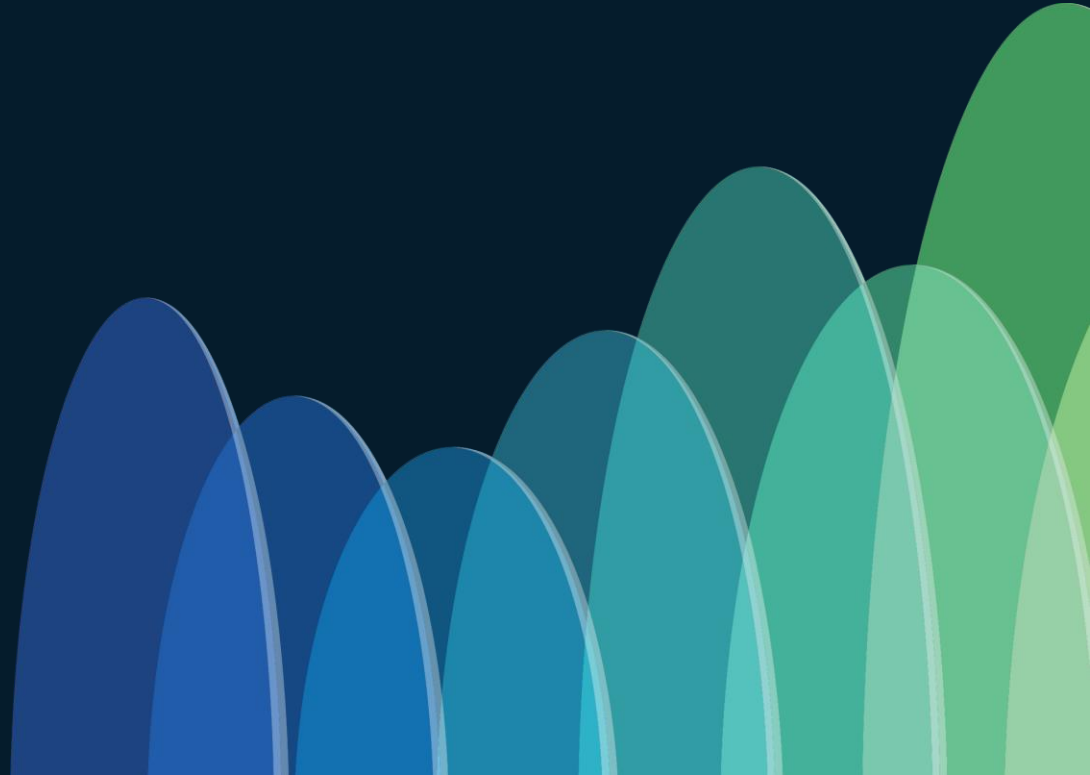
<https://ciscolive.ciscoevents.com/ciscolivebot/#DEVNET-1249>



Agenda

- Intro to Programmability & Automation
- Day 0 Onboarding
- What is ZTP?
- ZTP vs Secure ZTP
- Use case: upgrading image
- Demo
- Resources

Intro to Programmability & Automation



Programmability Solutions for Complex Networks



Efficient

Managing infrastructure as code while lowering OPEX



Consistent

Repeatable and precise intent-based outcomes



Scalable

Reduce time and effort with large network automation requirements



Secure

Secure APIs and integrations



Seamless

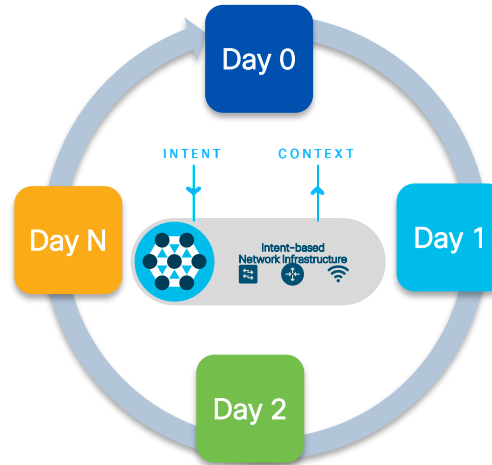
Create value integrating with 3rd party platforms



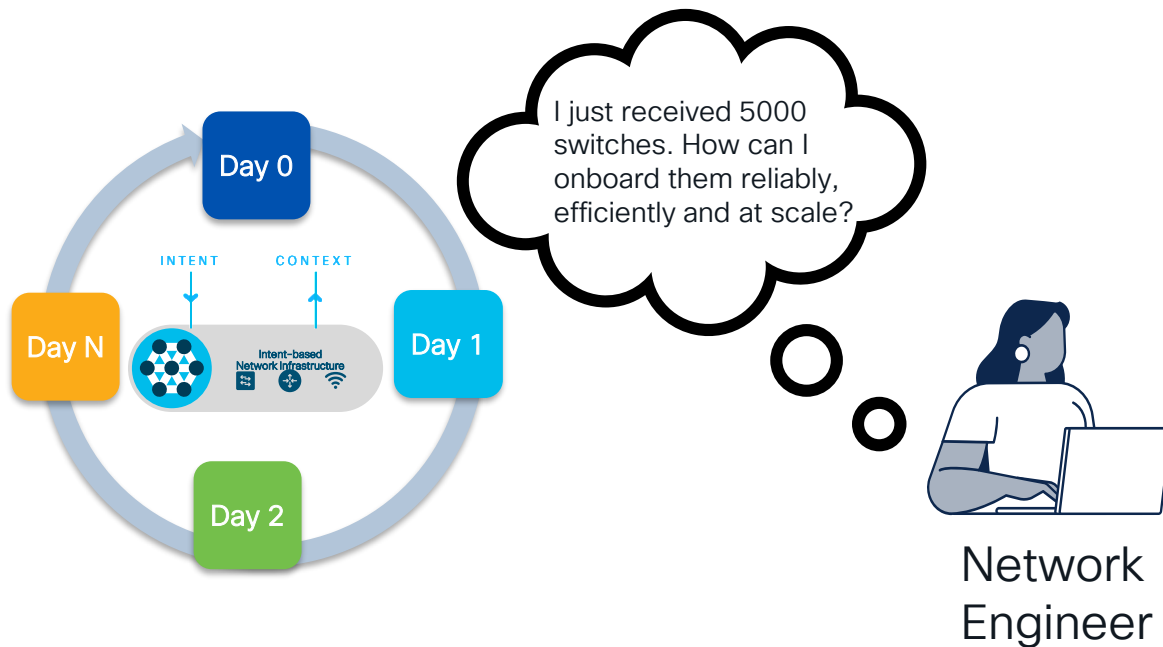
Smart

Real time event notifications, AI/ML capabilities, and streaming telemetry

Cisco IOS XE Programmability & Automation Lifecycle



Cisco IOS XE Programmability & Automation Lifecycle



Cisco IOS XE Programmability & Automation Lifecycle

Pre-boot Execution Environment (iPXE)

RFC8572 Secure Zero Touch

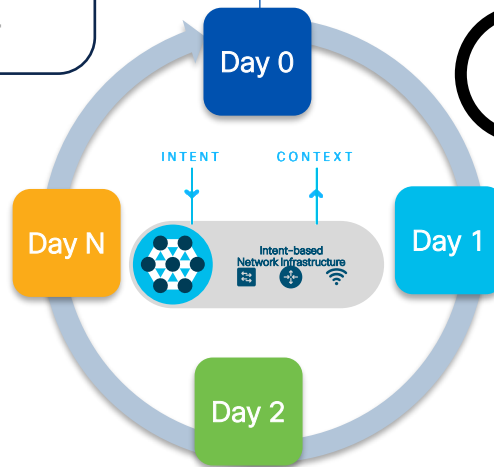
Provisioning ZTP

VM Automation

Provisioning Automation



Device Onboarding



Nice! All 5000 switches are onboarded using Secure Zero Touch Provisioning



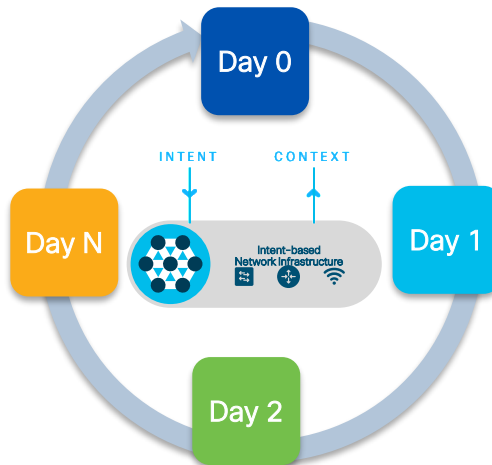
Network Engineer

Cisco IOS XE Programmability & Automation Lifecycle

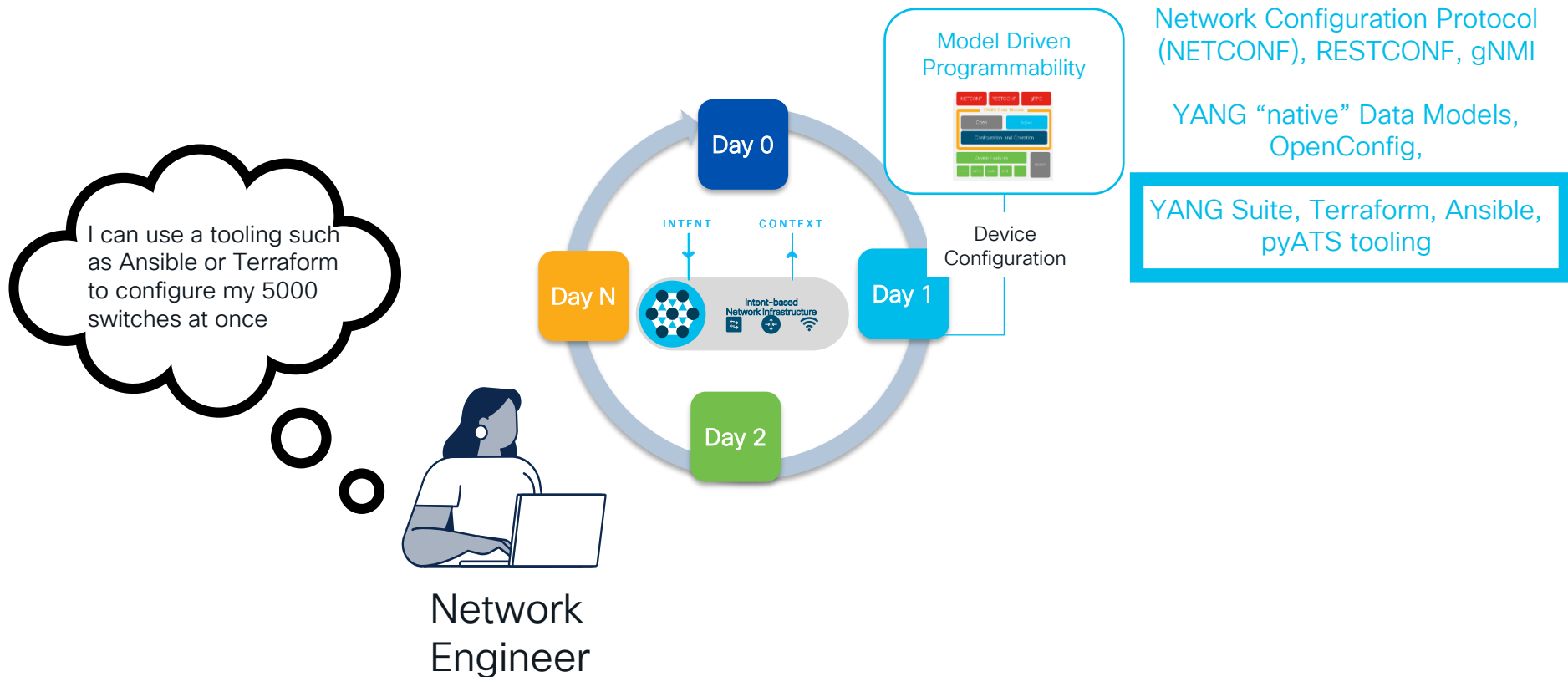
Now that the 5000 new switches are onboarded, how can I configure them reliably, efficiently and at scale?



Network Engineer



Cisco IOS XE Programmability & Automation Lifecycle

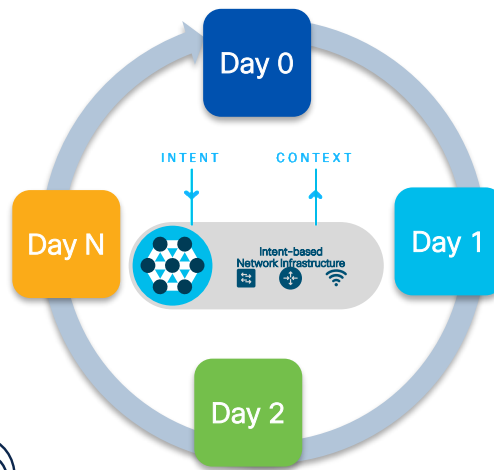


Cisco IOS XE Programmability & Automation Lifecycle

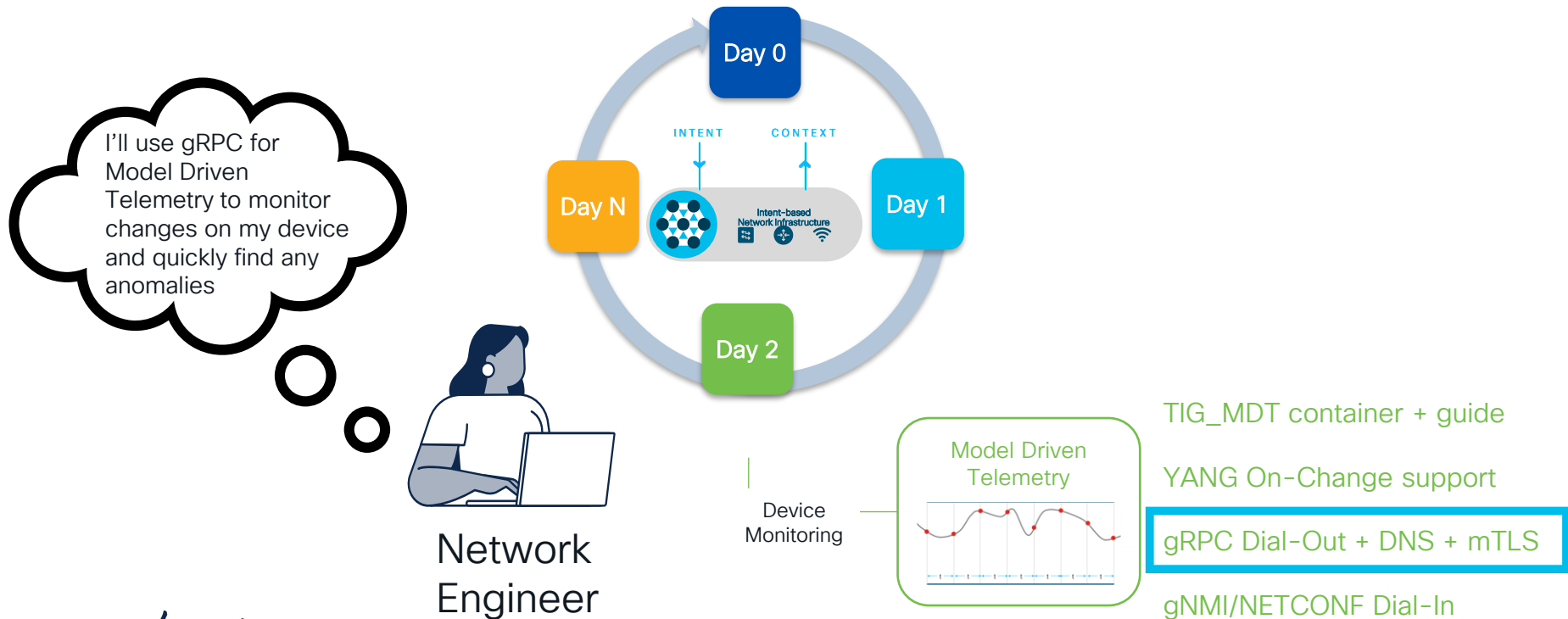
Now that the 5000 new switches are configured, how can I monitor them reliably, efficiently and at scale?



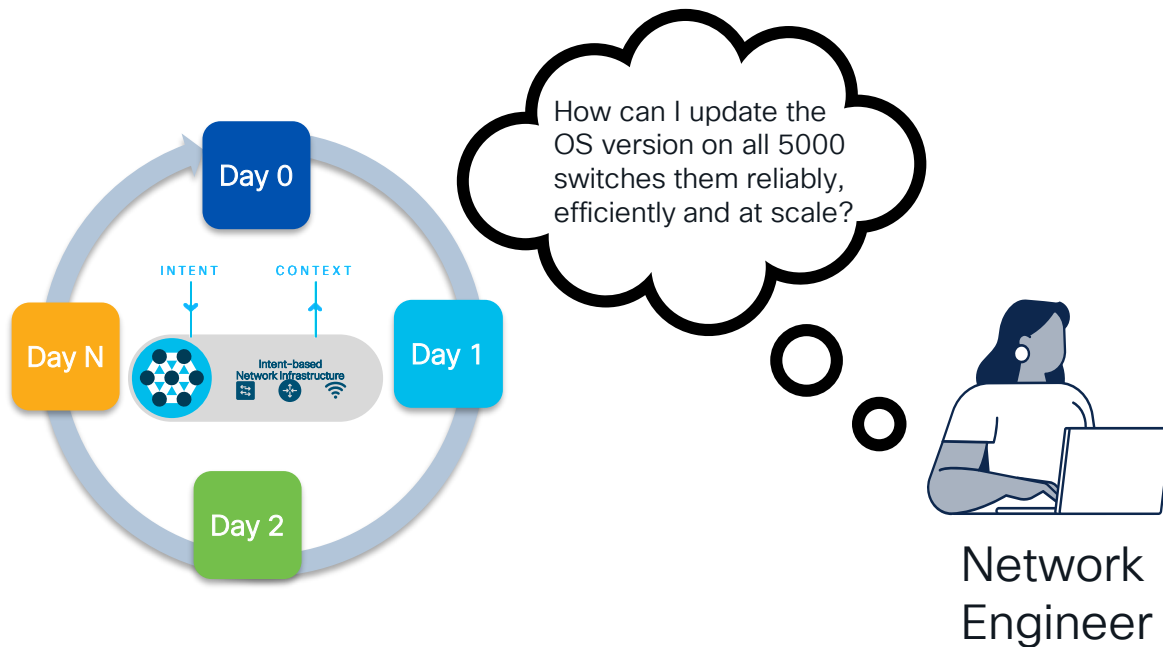
Network Engineer



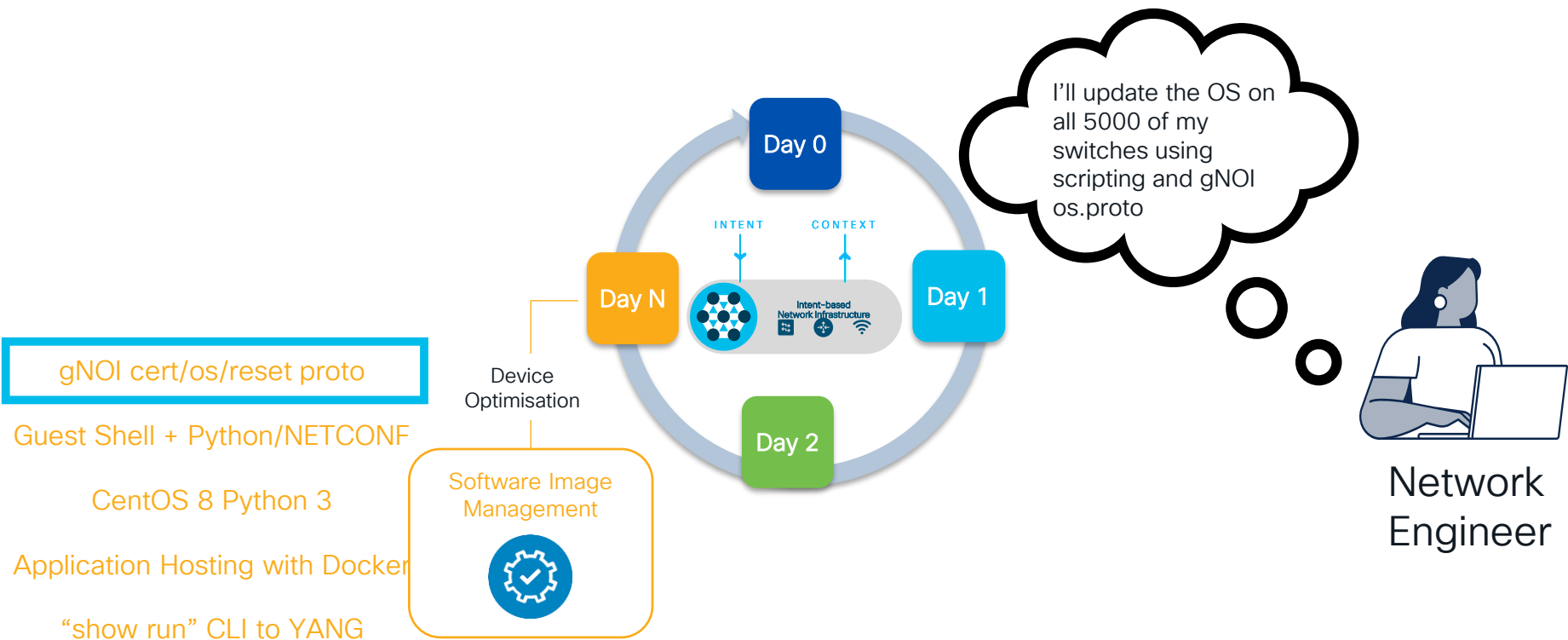
Cisco IOS XE Programmability & Automation Lifecycle



Cisco IOS XE Programmability & Automation Lifecycle



Cisco IOS XE Programmability & Automation Lifecycle

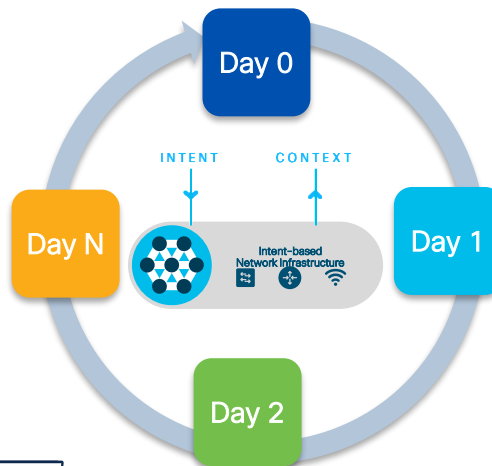


Cisco IOS XE Programmability & Automation Lifecycle

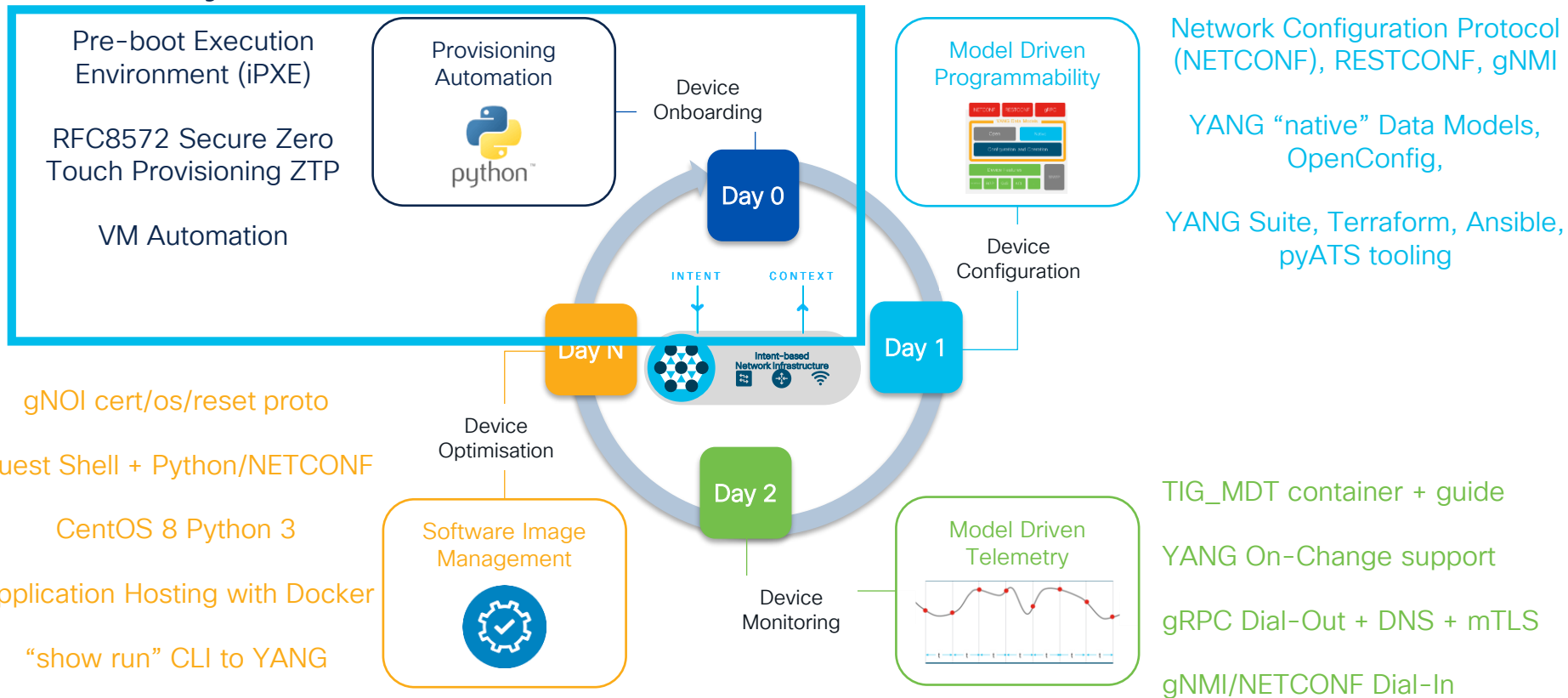
Whew! Now that programmability has me covered, I can focus on the important stuff!



Network Engineer



Cisco IOS XE Programmability & Automation Lifecycle



IOS XE Programmability and Telemetry “Stack”

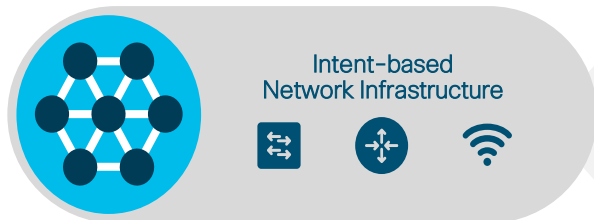
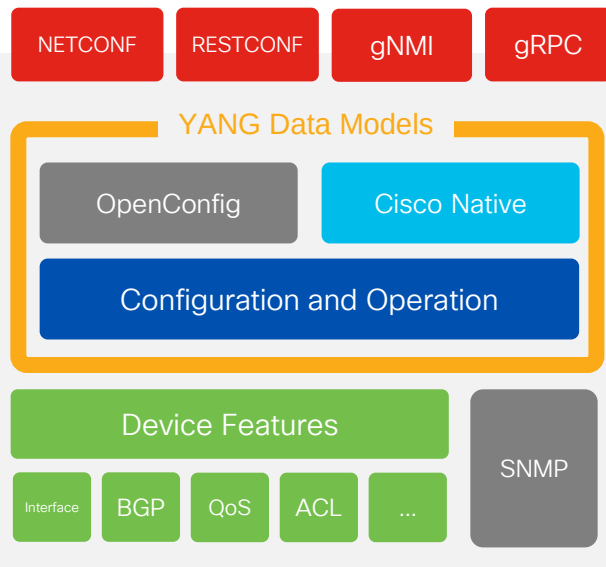
CLI

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



Model Driven Programmability Interface Comparison



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

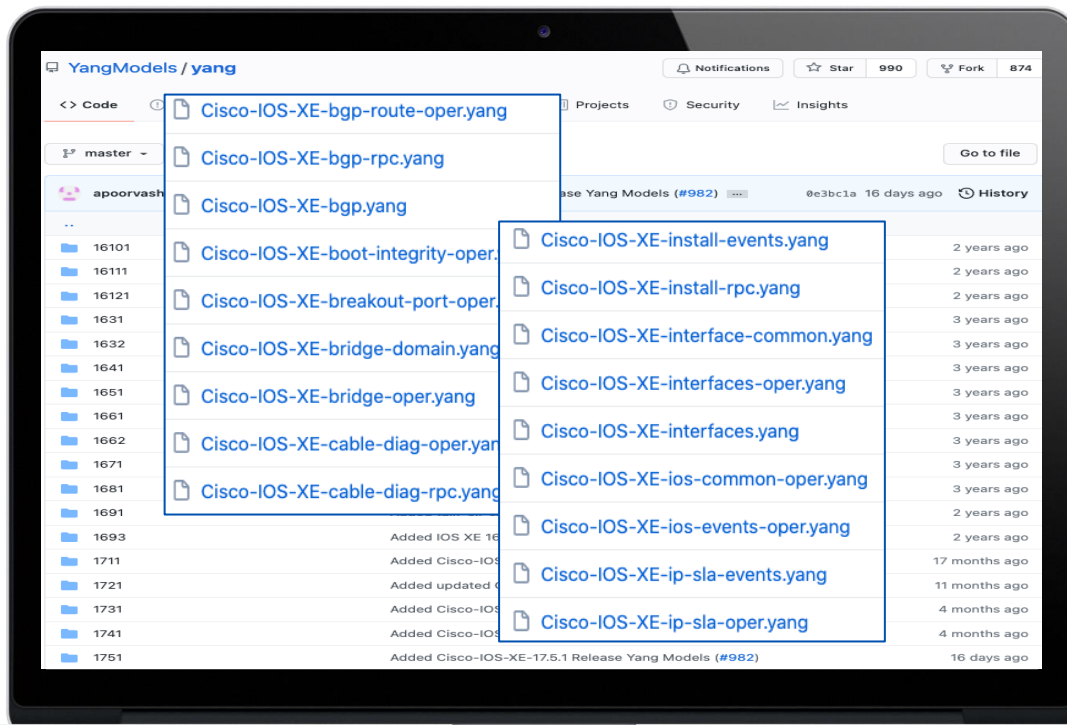
	NETCONF	RESTCONF	gNMI
Minimum IOS XE Version	16.6	16.7	16.8
Recommended Version	17.6	17.6	17.7
Default Port	830	443	9339
Operations	<get>,<get-config>,<edit-config>,<establish-subscription>	GET, POST, PUT, PATCH, DELETE	GET, SET, SUBSCRIBE
Encoding	XML	XML or JSON	RFC7951 JSON_IETF
Security	SSH + PKI certificate or password	HTTPS user/pass	TLS certificate with user authentication
Transport Protocol	SSH	HTTPS	HTTP/2
Tooling	YANG Suite, ncclient, Netconf-console	YANG Suite*, Postman, python	YANG Suite*, gnmic, gnmi_cli
Content	YANG	YANG	YANG + Protobuf

Cisco IOS XE - YANG models on GitHub



- RFC 7950 YANG data modelling language are the API definitions for IOS XE
- The YANG modules are available for download from the API and are also published on Github.com
- Notable modules are listed below for the running-config, feature oper, actions and event notifications

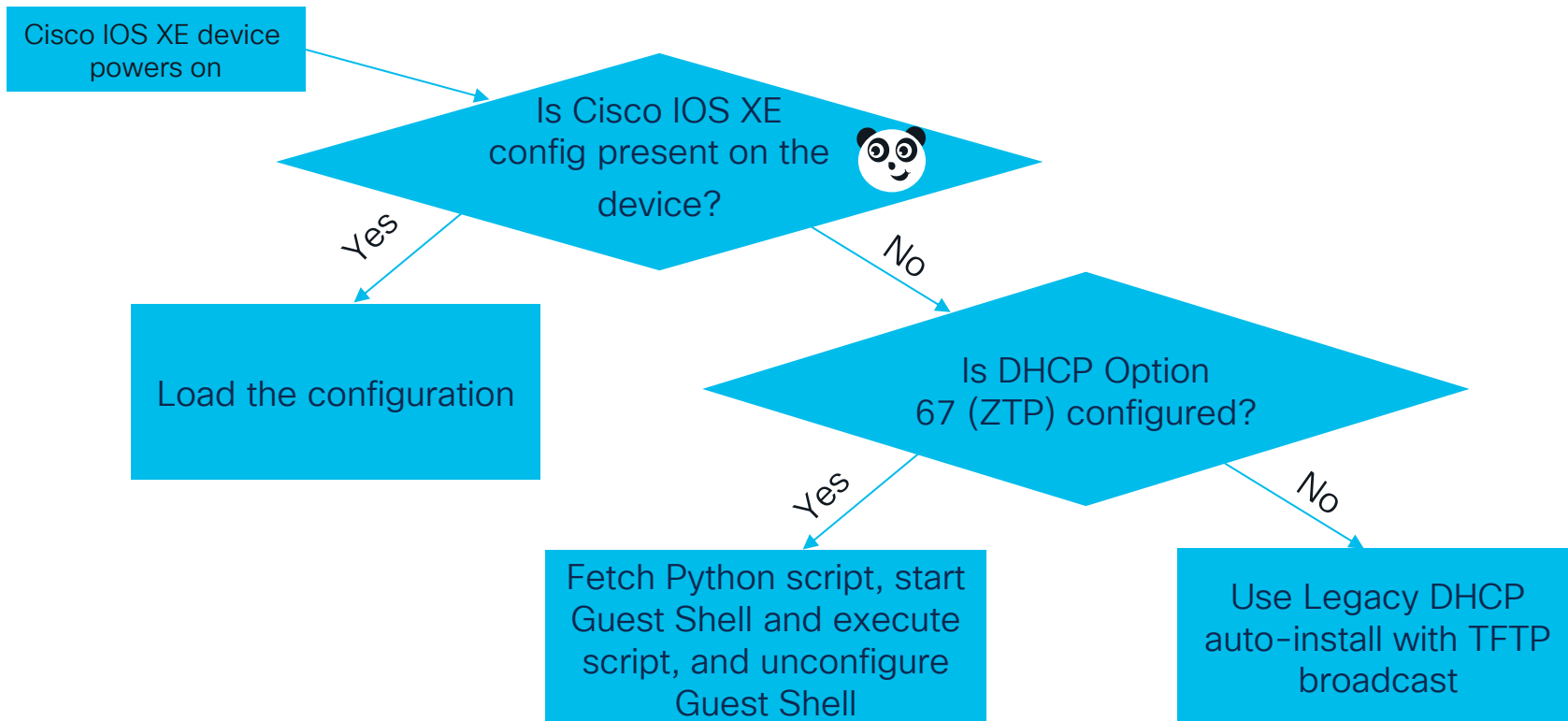
YANG module name.yang	Description
Cisco-IOS-XE-native	running-config
Cisco-IOS-XE-{feature}-cfg	Feature configuration
Cisco-IOS-XE-{feature}-oper	Feature operational data
Cisco-IOS-XE-{feature}-rpc	Actions
Cisco-IOS-XE-{feature}-events	Telemetry Events
Cisco-evpn-service	EVPN service abstraction
OpenConfig-{feature}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



The YANG models are available for download directly from the running IOS XE device's NETCONF, RESTCONF, or gNMI API, and from:

What is ZTP?

Day 0 device onboarding workflow

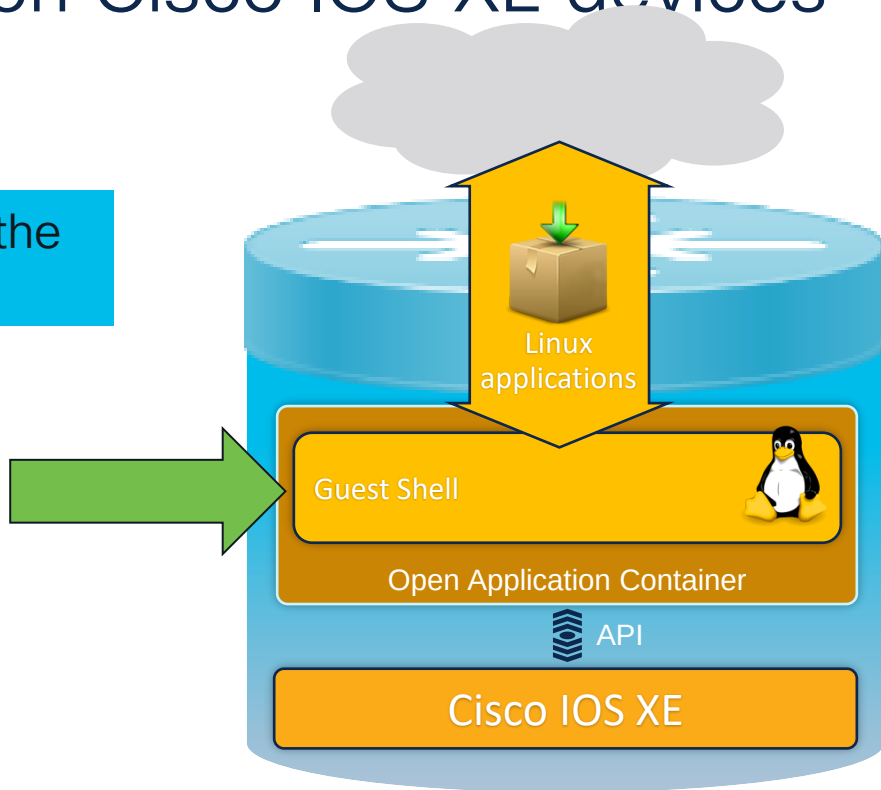


Linux Shell Environment on Cisco IOS XE devices

Cisco IOS XE devices can access the on-box Guest Shell using API calls

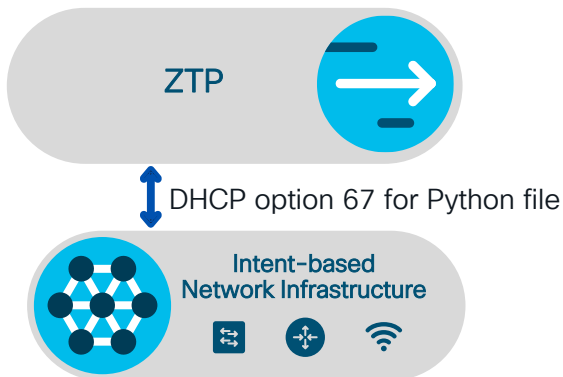
This Linux shell can be used for

- Scripting
- Hosting Applications
- Zero Touch Provisioning
- & so much more!



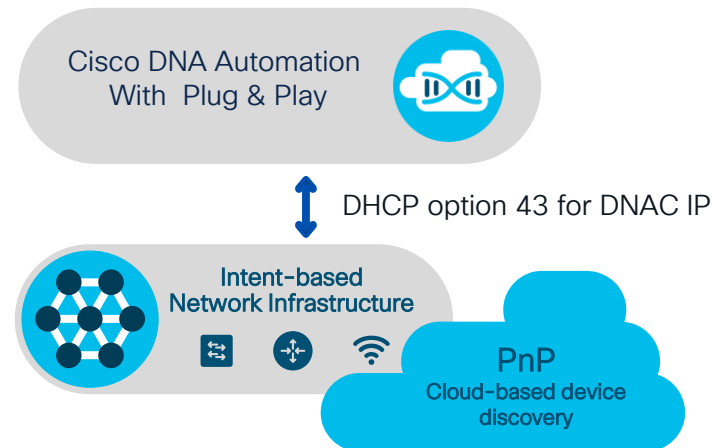
Controller Based Day 0 with Cisco DNA Center

Zero Touch Provisioning controller-less



Python Configuration File + Guest Shell
Available with TFTP or HTTP

Plug and Play with Cisco DNA Center

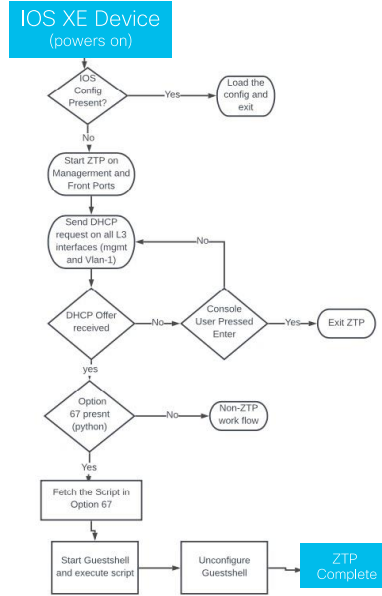
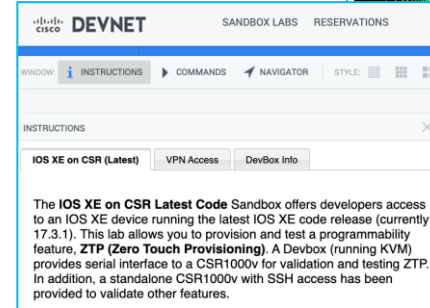
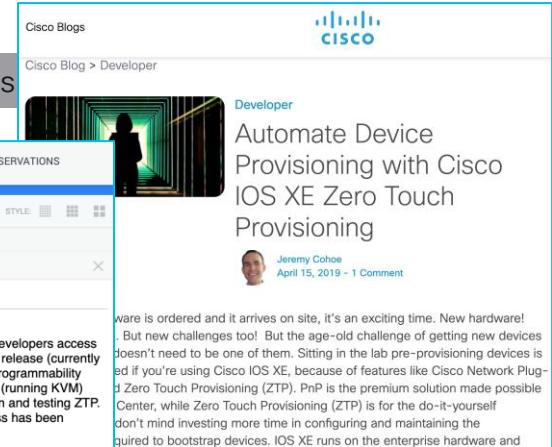
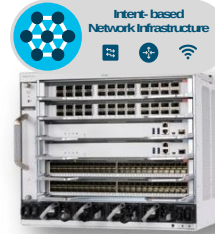
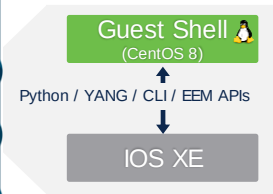


Claim and Manage in DNAC GUI

Day 0 - Classic ZTP Overview

The Day 0 ZTP features are used to automatically configure and provision network devices

1. When an 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/DevNet>
Secure ZTP blog <https://blogs.cisco.com/developer/secureztp01>

Python API between Guest Shell and Cisco IOS XE

FYI

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 customer use case – code samples

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

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 path '/ IOSXE-Zero-Touch-Provisioning / ztp_solution_example /' is shown. Below the path, there is a table of files:

File Name	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

The screenshot shows the code for the 'ztp.py' file. The code is a Python script that imports various modules and defines a main function. It includes a dictionary for software mappings and a function to create a log file.

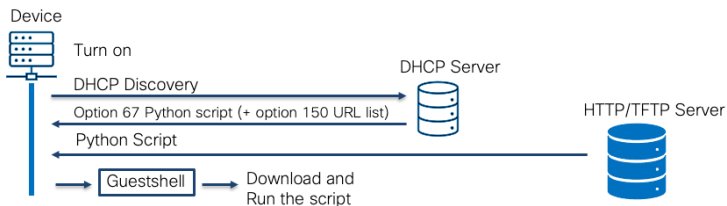
```
1 # Importing cli module
2 from cli import configure, cli, configurep, executep
3 import re
4 import time
5 import urllib
6 import sys
7 import logging
8 import os
9 from logging.handlers import RotatingFileHandler
10 import subprocess
11
12
13 software_mappings = {
14     'C9300-24P': {
15         '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_to_file = 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_to_file == True):
41             filepath = create_logfile()
42             configure_logger(filepath)
```

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)

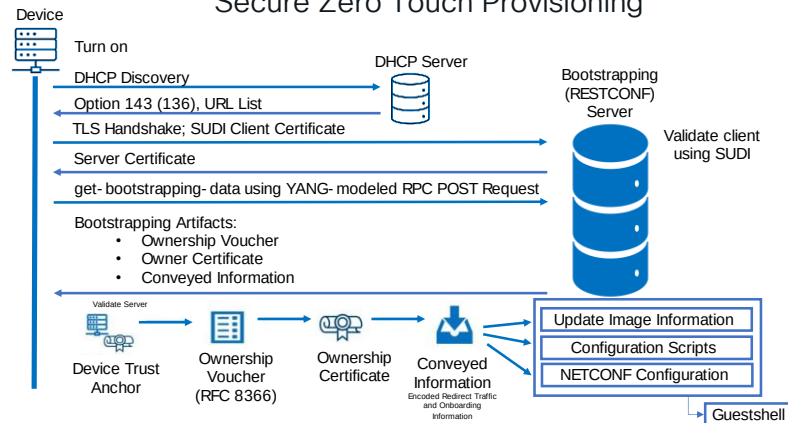
Classic Zero Touch Provisioning



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

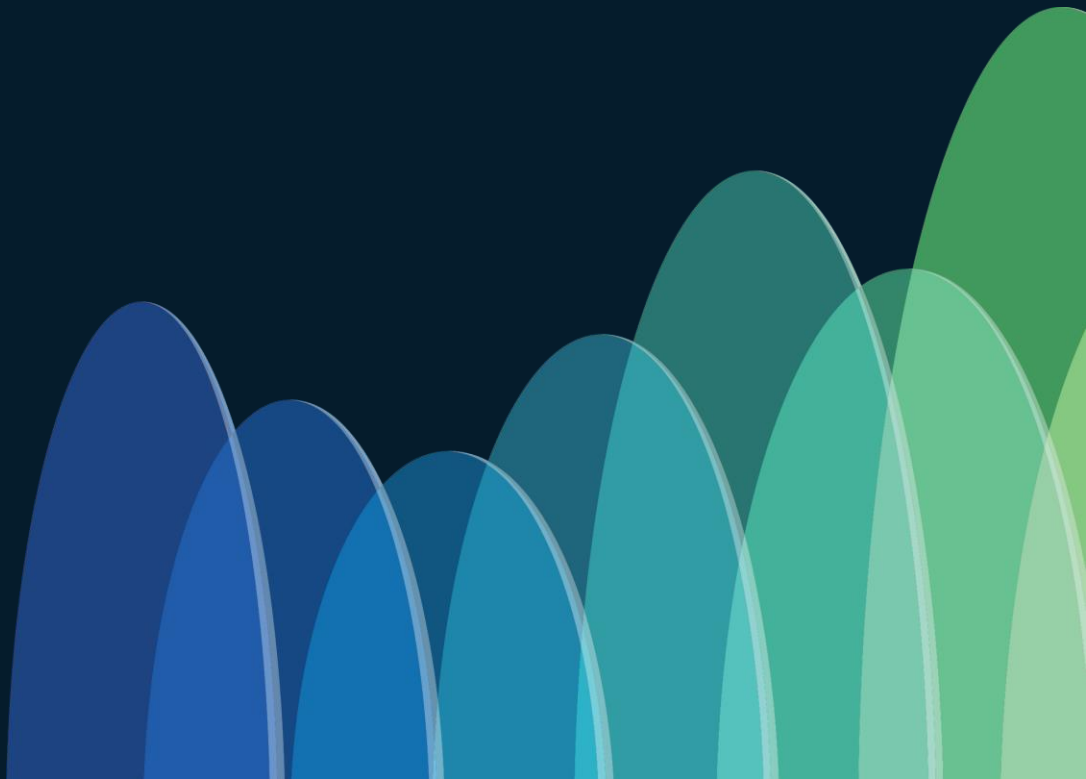
- Management system needs to validate the device
- Device needs to validate the server
- Device must validate the data is what server sent

Secure Zero Touch Provisioning



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

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.configure(['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 authorisation 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: Initialised 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

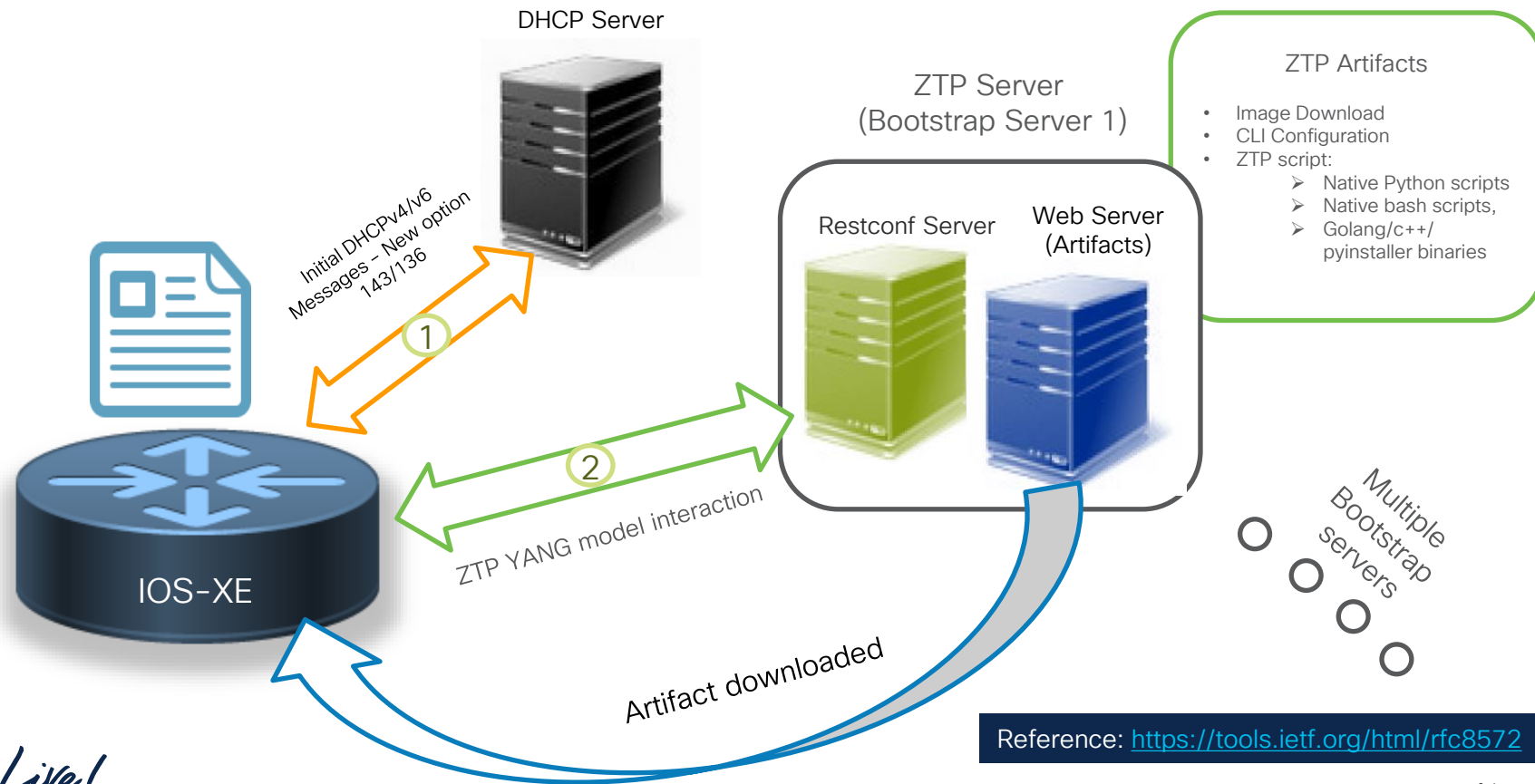
```
From 10.1.1.3 icmp_seq=255 Destination Host Unreachable
From 10.1.1.3 icmp_seq=256 Destination Host Unreachable
From 10.1.1.3 icmp_seq=257 Destination Host Unreachable
From 10.1.1.3 icmp_seq=258 Destination Host Unreachable
From 10.1.1.3 icmp_seq=259 Destination Host Unreachable
From 10.1.1.3 icmp_seq=260 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!

ZTP vs Secure ZTP

RFC8572

Secure ZTP (SZTP) workflow

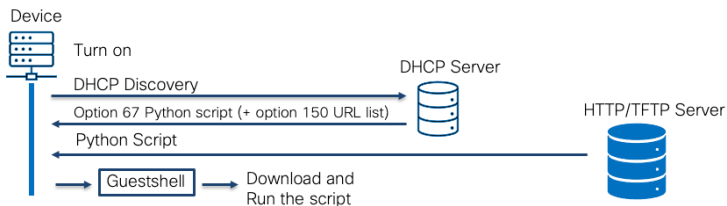


RFC8572 Secure ZTP

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)

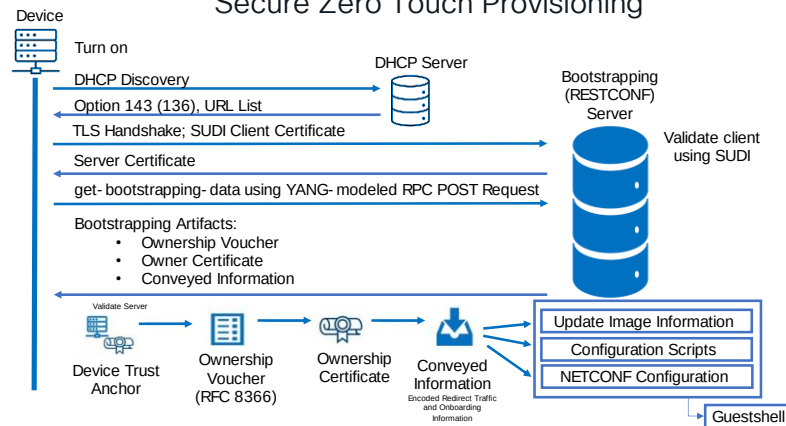
Classic Zero Touch Provisioning



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

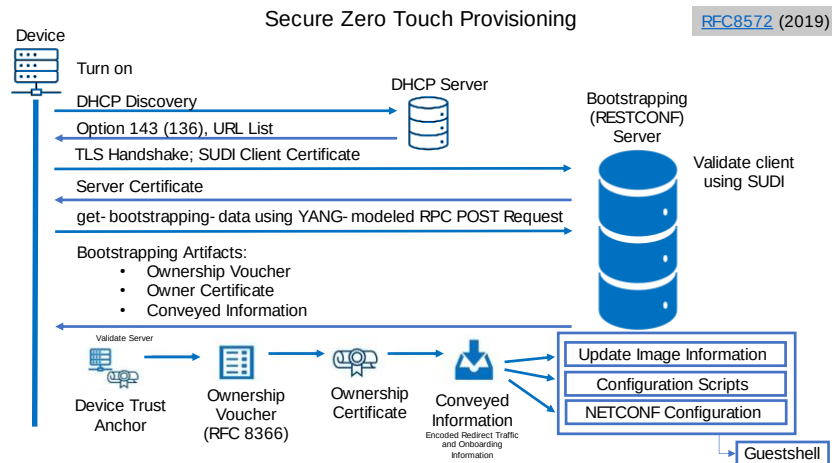
Secure Zero Touch Provisioning



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

Bootstrapping Server

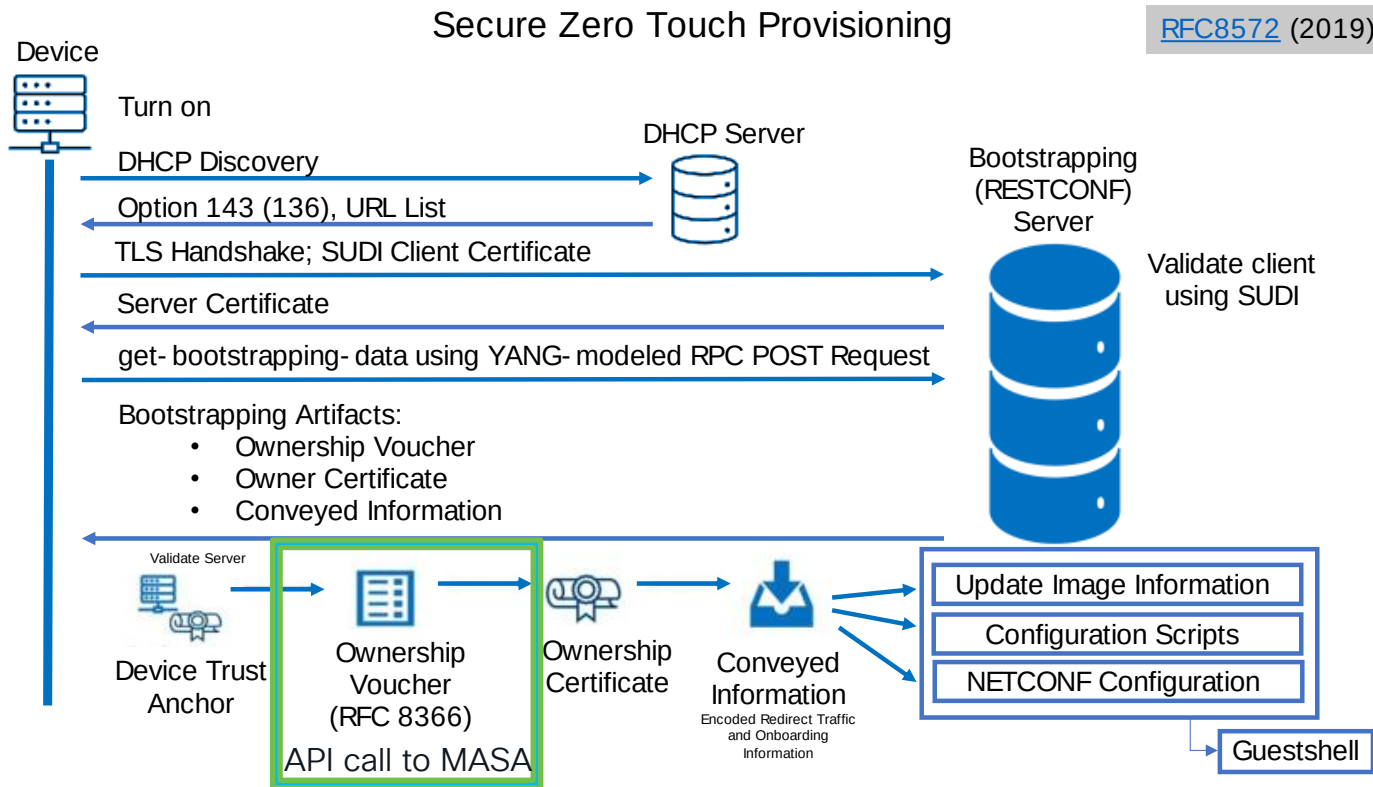
1. Currently there are no open-source bootstrapping servers freely available, but one can easily be written in Python by following the RFC and API documentation
2. YANG Suite + Secure ZTP plugin on road map
3. We will soon be publishing a bootstrapping server for testing and validation purposes



Workflows

Single bootstrap server without validation.

Redirect to bootstrap server with trust anchor validation.



MASA and Certificate Signing for OV

(Manufacturer Authorised Signing Authority)

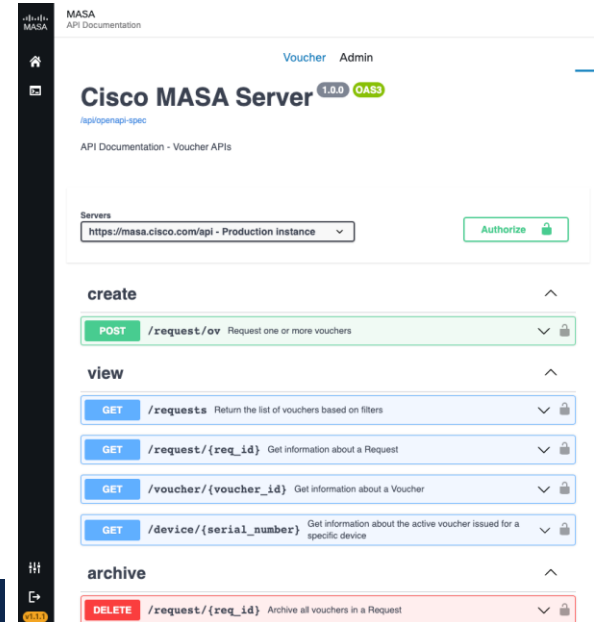
The upstream cloud-based *certificate verification service* at <https://masa.cisco.com> is available for IOS XE Ownership Voucher (OV) signing workflows



Serial Number	Requested By	Requested	Expires	Assertion	Status	Request ID	Voucher ID	PDC Organization	Actions
FOC22362ENG	kmuthus2@cisco.com	Aug 30 2022, 4:14 PM	Jun 2 2023, 12:27 PM	LOGGED	COMPLETED	7f1c8b26-28b9-11...	7227d980-28b9-11...	Cisco Systems Inc.	[Info] [Edit]
FOC2549R1U7	mbrenels@cisco.com	Aug 24 2022, 5:36 AM	Aug 24 2023, 5:36 AM	LOGGED	COMPLETED	6b42b7d4-23a9-11...	6b48bc7e-23a9-11...	CISCO SYSTEMS INC	[Info] [Edit]
FOC2237R0NK	kmuthus2@cisco.com	Jun 2 2022, 3:08 PM	Jun 2 2023, 12:27 PM	LOGGED	COMPLETED	9272640a-e2c0-11...	927d1a2c-e2c0-11...	Cisco Systems Inc.	[Info] [Edit]

MASA API Docs: <https://masa.cisco.com/docs>

Reference XR: <https://xrdocs.io/automation/tutorials/setting-up-crosswork-for-sztp/>



cisco Live!

Classic vs Secure ZTP Example Output

Classic ZTP

Acquired IPv4 address 10.1.1.157 on Interface Vlan1

Received following DHCPv4 options:

domain-name : ios-xe-programmability-
lab.cisco.com

bootfile : http://10.1.1.3/ztp-simple.py

dns-server-ip : 128.107.212.175

si-addr : 10.1.1.3

Secure ZTP

Acquired IPv4 address 192.168.1.61 on Interface
GigabitEthernet0/0

Received following DHCPv4 options:

bootstrap-server-list : https://192.168.1.10:11601

See additional info here: <https://github.com/jeremycohoe/IOSXE-Zero-Touch-Provisioning/blob/master/logs/pod08-xelab-vlan1.txt>

Use Case: Image Upgrade from Conveyed Info

```
{
  "ietf-sztp-conveyed-info:onboarding-information" : {
    "boot-image" : {
      "os-name" : "VendorOS",
      "os-version" : "17.2R1.6",
      "download-uri" : [ https://example.com/path/to/image/file ],
      "image-verification" : [
        {
          "hash-algorithm" : "ietf-sztp-conveyed-info:sha-256",
          "hash-value" : "ba:ec:cf:a5:67:82:b4:10:77:c6:67:a6:22:ab:\
7d:50:04:a7:8b:8f:0e:db:02:8b:f4:75:55:fb:c1:13:b2:33"
        }
      ]
    },
    "configuration-handling" : "merge",
    "pre-configuration-script" : "base64encodedvalue==",
    "configuration" : "base64encodedvalue==",
    "post-configuration-script" : "base64encodedvalue=="
  }
}
```

Conveyed Information

Pre-Config

```
import subprocess

import datetime

with open('/bootflash/guest-share/pre-
config-script-output.txt', 'w') as f:

    f.write('This file was created by the
pre-configuration script\n')

    now = datetime.datetime.now()

    f.write('Current time: ' + str(now))
```

Post-Config

```
import datetime

with open('/bootflash/guest-share/post-
config-script-output.txt', 'w') as f:

    f.write('This file was created by the
post-configuration script\n')

    now = datetime.datetime.now()

    f.write('Current time: ' + str(now))
```

Classic vs Secure ZTP Example Output

Classic ZTP

Acquired IPv4 address 10.1.1.157 on Interface Vlan1

Received following DHCPv4 options:

domain-name : ios-xe-programmability-lab.cisco.com

bootfile : http://10.1.1.3/ztp-simple.py

dns-server-ip : 128.107.212.175

si-addr : 10.1.1.3

Secure ZTP

Acquired IPv4 address 192.168.1.61 on Interface
GigabitEthernet0/0

Received following DHCPv4 options:

bootstrap-server-list : https://192.168.1.10:11601

See additional info here: <https://github.com/jeremycohoe/IOSXE-Zero-Touch-Provisioning/blob/master/logs/pod08-xelab-vlan1.txt>

Use Case: Image Upgrade from Conveyed Info

```
{
  "ietf-sztp-conveyed-info:onboarding-information" : {
    "boot-image" : {
      "os-name" : "VendorOS",
      "os-version" : "17.2R1.6",
      "download-uri" : [ https://example.com/path/to/image/file ],
      "image-verification" : [
        {
          "hash-algorithm" : "ietf-sztp-conveyed-info:sha-256",
          "hash-value" : "ba:ec:cf:a5:67:82:b4:10:77:c6:67:a6:22:ab:\
7d:50:04:a7:8b:8f:0e:db:02:8b:f4:75:55:fb:c1:13:b2:33"
        }
      ]
    },
    "configuration-handling" : "merge",
    "pre-configuration-script" : "base64encodedvalue==",
    "configuration" : "base64encodedvalue==",
    "post-configuration-script" : "base64encodedvalue=="
  }
}
```

Conveyed Information

Pre-Config

```
import subprocess

import datetime

with open('/bootflash/guest-share/pre-config-
script-output.txt', 'w') as f:

    f.write('This file was created by the
pre-configuration script\n')

    now = datetime.datetime.now()

    f.write('Current time: ' + str(now))
```

Post-Config

```
import datetime

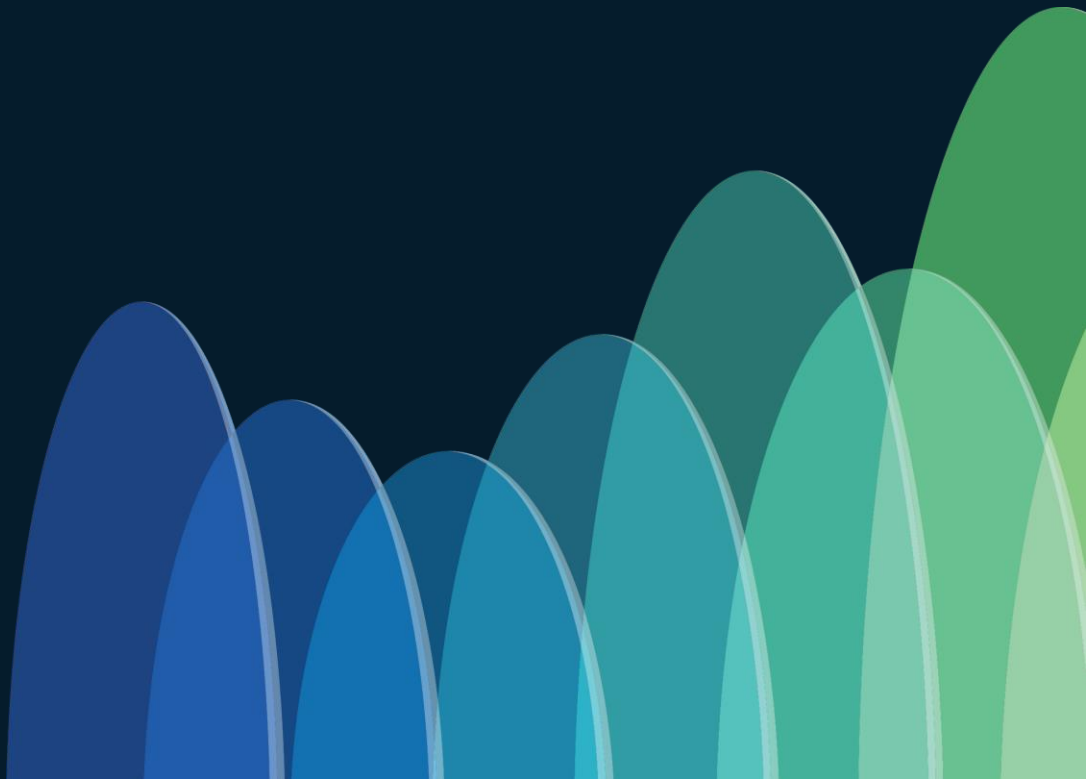
with open('/bootflash/guest-share/post-
config-script-output.txt', 'w') as f:

    f.write('This file was created by the
post-configuration script\n')

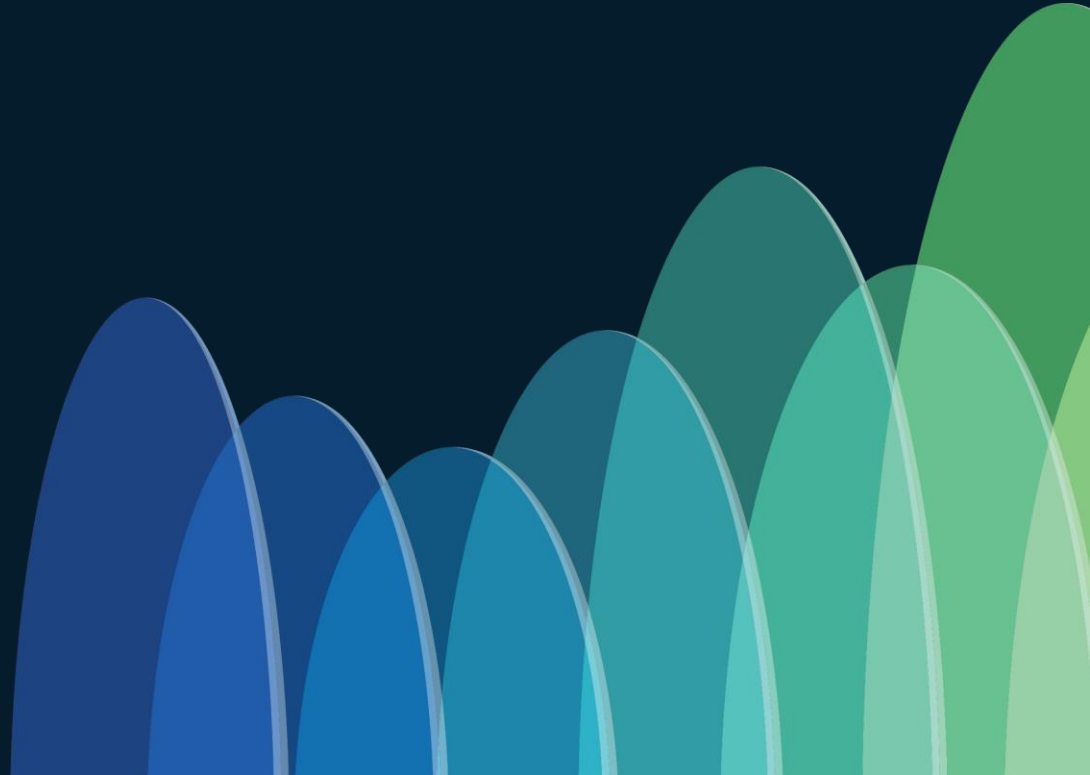
    now = datetime.datetime.now()

    f.write('Current time: ' + str(now))
```

Demo



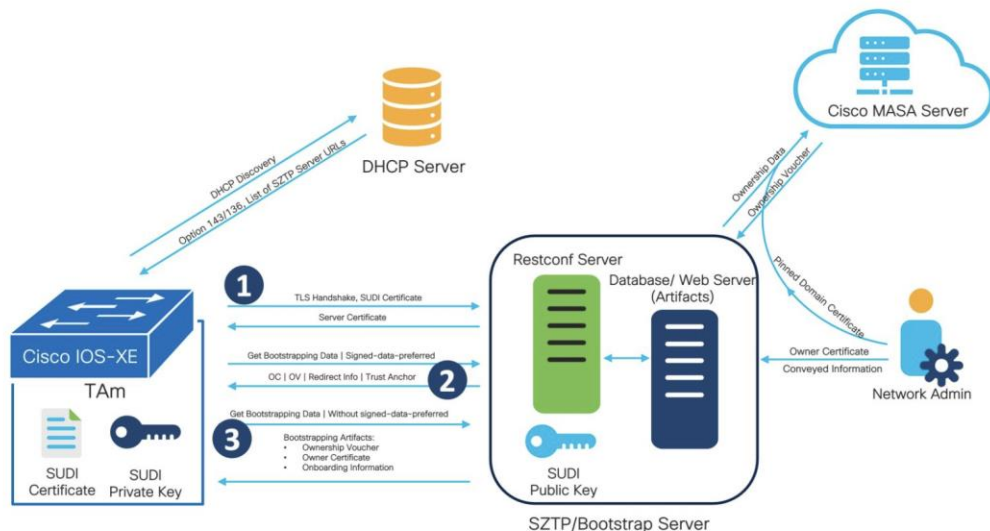
Resources



Slide on Secure ZTP Blog

How does Secure ZTP work?

Secure ZTP employs three-step validation, including device validation, server validation, and artifact validation, to securely onboard the device. The diagram provided below illustrates the various steps involved in the device onboarding and provisioning process within a secure ZTP framework. Let's take a closer look at each of these steps:



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

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>

Keynote Deep Dives

Wednesday
10:30am – 11:30am



Experiences Amplified:
How AI Can Fuel Better Employee and Customer Experiences

Level 1
Room 106



Smart, Secure, Seamless:
Transforming Experiences with Next-Generation Networking

Level 2
Room 204



Harness a Bold New Era:
Transform Data Centre and Service Provider Connectivity

Level 2
Room 203



Securing User to Application and Everything in Between

Level 2
Melbourne Room 2



Unlocking Digital Resilience through Unified Observability

The HUB
Centre Stage

Complete Your Session Evaluations



Complete a minimum of 4 session surveys and the Overall Event Survey to claim a **Cisco Live T-Shirt**.



Complete your surveys in the **Cisco Live mobile app**.



Continue your education

- Visit the Cisco Stand 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



Thank you

CISCO *Live!*

#CiscoLiveAPJC

CISCO *Live!*

GO BEYOND

#CiscoLiveAPJC