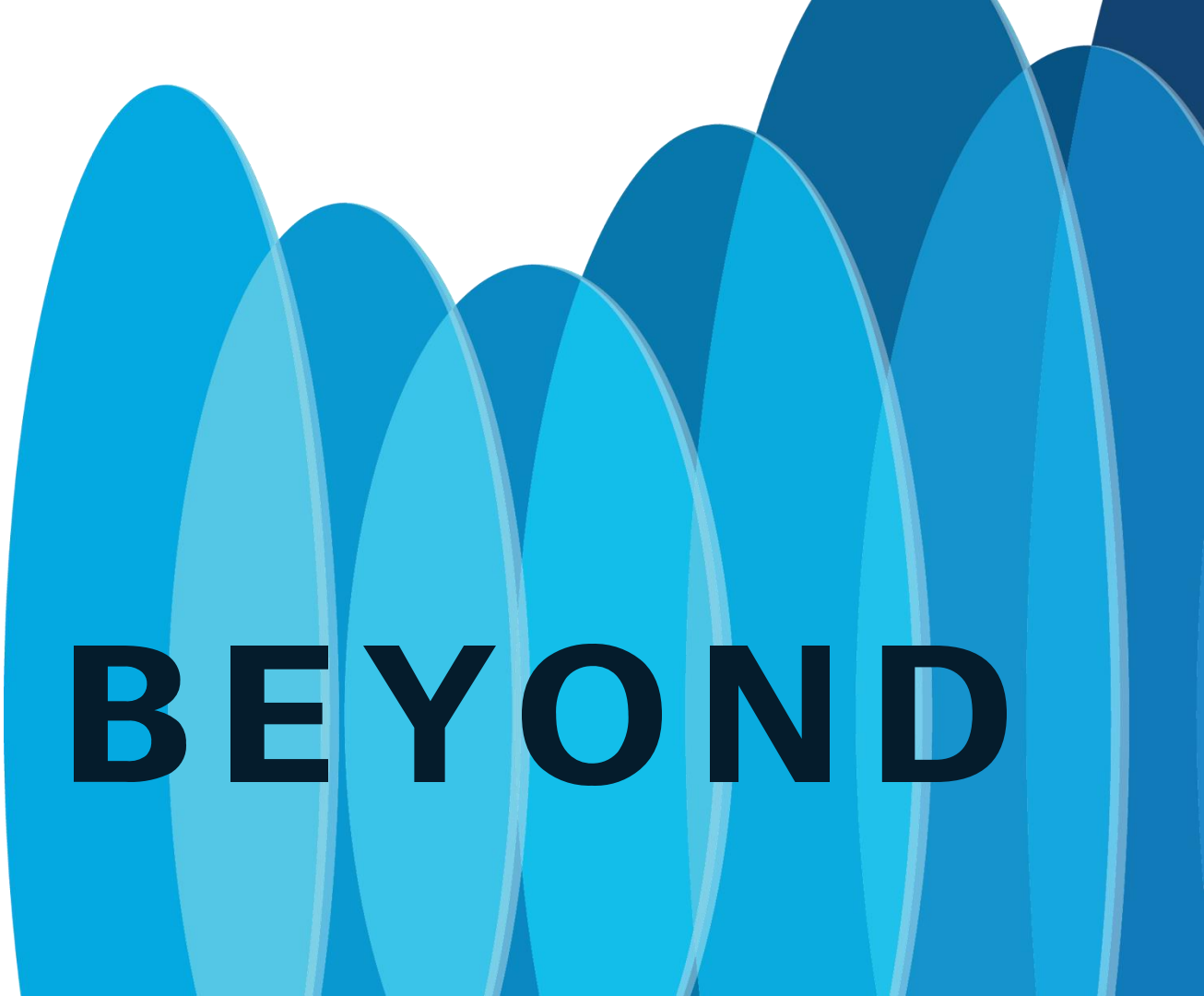
cisco Live!



GO BEYOND

Automation and Management with Catalyst Center APIs

Technical Workshop

Hector Morales
SE

Anton Inniss
BDM

Silvia Reis
SAM

cisco Live!

Agenda

Day 1		
9:00-9:30	Introduction and workshop objectives	ALL
9:30-10:30	Catalyst Center deep dive	Cisco
10:30-11:00	Lab introduction	Cisco
11:00-11:30	Break and email	ALL
11:30-13:30	CatC APIs 100-200 level with Postman collections	ALL
13:30-14:30	Lunch	ALL
14:30-16:30	CatC APIs 300-400 level using jupyter notebooks and python	ALL
16:30-17:00	Wrap up day 1	Cisco

Day 2		
9:00-9:30	Review Computacenter use cases	CC
9:30-10:30	Managed Campus Operations Guide	Cisco
10:30-11:00	Challenge: Use Case implementation	CC
11:00-11:30	Break and email	ALL
11:30-13:30	Challenge implementation	CC
13:30-14:00	Challenge presentation	CC
14:00-14:30	Wrap up day 2 and next steps	ALL

cisco Live!

Catalyst Center Headless Deployments

cisco Live!

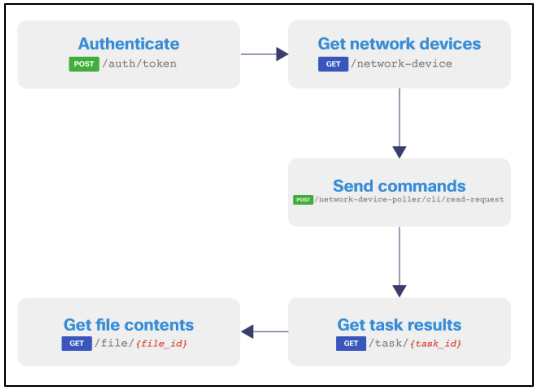
The complexity to operate large networks


```
!
Router# show running-config
Building configuration...


Current configuration : 5980 bytes
!
! Last configuration change at 13:56:48 PST Fri Nov 3 2017 by admin
!
version 16.6
service timestamps debug datetime msec
service timestamps log datetime msec
service call-home
platform qfp utilization monitor load 80
no platform punt-keepalive disable-kernel-core
platform shell
!
hostname Router
!
boot-start-marker
boot system tftp /auto/tftp-sjc-users5/c1100-universalk9_ias.16.06.02.SPA.bin 223.255.254.254
boot-end-marker
!
!
vrf definition VRF-example
 description VRF-example
!
no logging console
!
aaa new-model
!
!
!
aaa login success-track-conf-time 1
!
!
!
!
aaa session-id common
```

IOS XE CLI

```
- name: Check the running-config against master config
  cisco.ios.ios_config:
    diff_against: intended
    intended_config: "{{ lookup('file', 'master.cfg') }}"

- name: Check the startup-config against the running-config
  cisco.ios.ios_config:
    diff_against: startup
    diff_ignore_lines:
      - ntp clock .*

- name: Save running to startup when modified
  cisco.ios.ios_config:
    save_when: modified
```

IOS XE Ansible Collection



Cisco Catalyst Center APIs

The complexity to operate large networks

Pros and Cons

	IOS XE CLI	IOS XE Ansible Collection	Cisco Catalyst Center APIs
Day 0	✔✔✔✔✔ Fine configuration Prone to human errors	✔✔✔✔✔ IaC, CI/CD pipelines Requires more expertise	✔✔✔✔✔ API Platform Single tenant
Day 1	✔✔✔✔ Data rich Requires additional infra	✔✔✔ Leverage community Prone to errors	✔✔✔✔✔ Extensive integration Requires more coding
Day 2	✔✔✔✔ Information rich Complex to troubleshoot	✔✔ Leverage community Not ready for Day 2	✔✔✔✔✔ Event management Requires more coding
Complexity [1]	Very complex	Mid complexity	Low complexity
IDEAL Team size [2]	10-15 engineers	5-7 engineers	3-5 engineers

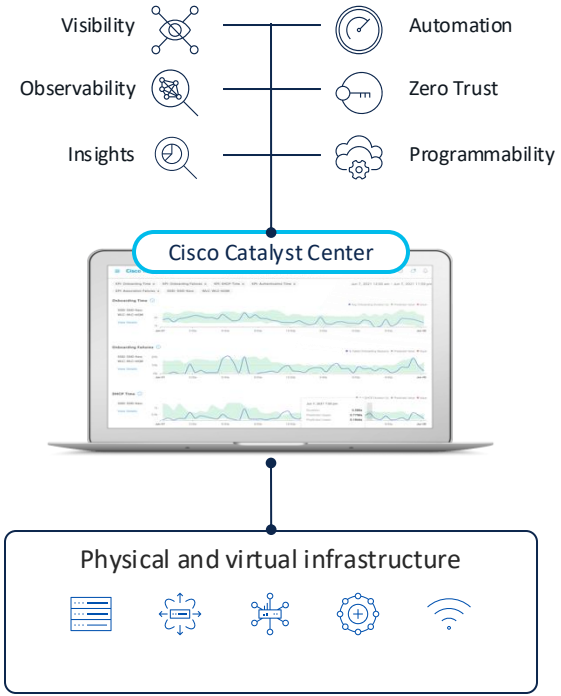
REALITY CHECK: We operate with reduced teams

Source: Own research with public data from top 100 customers and service providers, 2024
1 - Network size: 20,000 devices in a single campus
2 – Necessary team to support single campus

cisco Live!

Catalyst Center simplifies network operations

Enabling IT admin to deliver business outcomes, smarter and faster at scale



Small teams' productivity increase through automation

NetOps — Increase Scale

- Providing customers with business resiliency, continuity, and quick time to value
- Enabling customers to achieve Compliance of network with config policies
- Automation to simplify the creation and maintenance of customers networks

SecOps — Improved Security

- AI-driven security to classify endpoints and enforce security policies for a complete zero trust workplace solution
- Automate end point visibility, classification, and grouping

DevOps — Improved Service Delivery

- Mature APIs, SDKs, and closed-loop integrations to simplify and streamline ecosystem integration
- Faster service delivery using API-based automation workflows
- Early issue detection and integration with 3rd party platforms through enhanced notification channels

AIOps — Improved Performance Insights

- Reduced proactive problem resolution through faster Root Cause Analysis
- AI-driven visibility, observability, insights, and troubleshooting to ensure the health your customers applications, infrastructure and user experience

cisco Live!

Catalyst Center as a platform

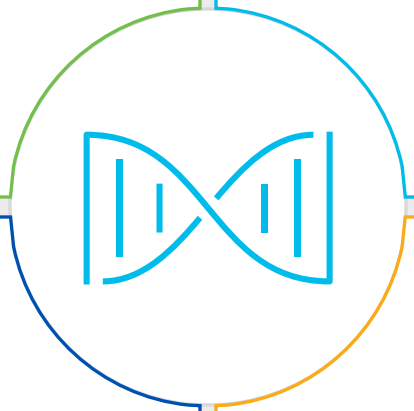
Powered for headless integration

Event Notifications

- Assurance Issues
- AI/ML Insights
- System Health
- Integration Connectivity
- License Management
- Webhooks
- PagerDuty
- Email
- Syslog
- SNMP

Northbound REST APIs

- Network Inventory
- Network Topology
- Network Design
- Provisioning
- SWIM, PnP
- Path Trace
- Assurance
- SDA
- Templates
- RMA
- Config Archive
- Sensors



IT Ecosystem Integrations

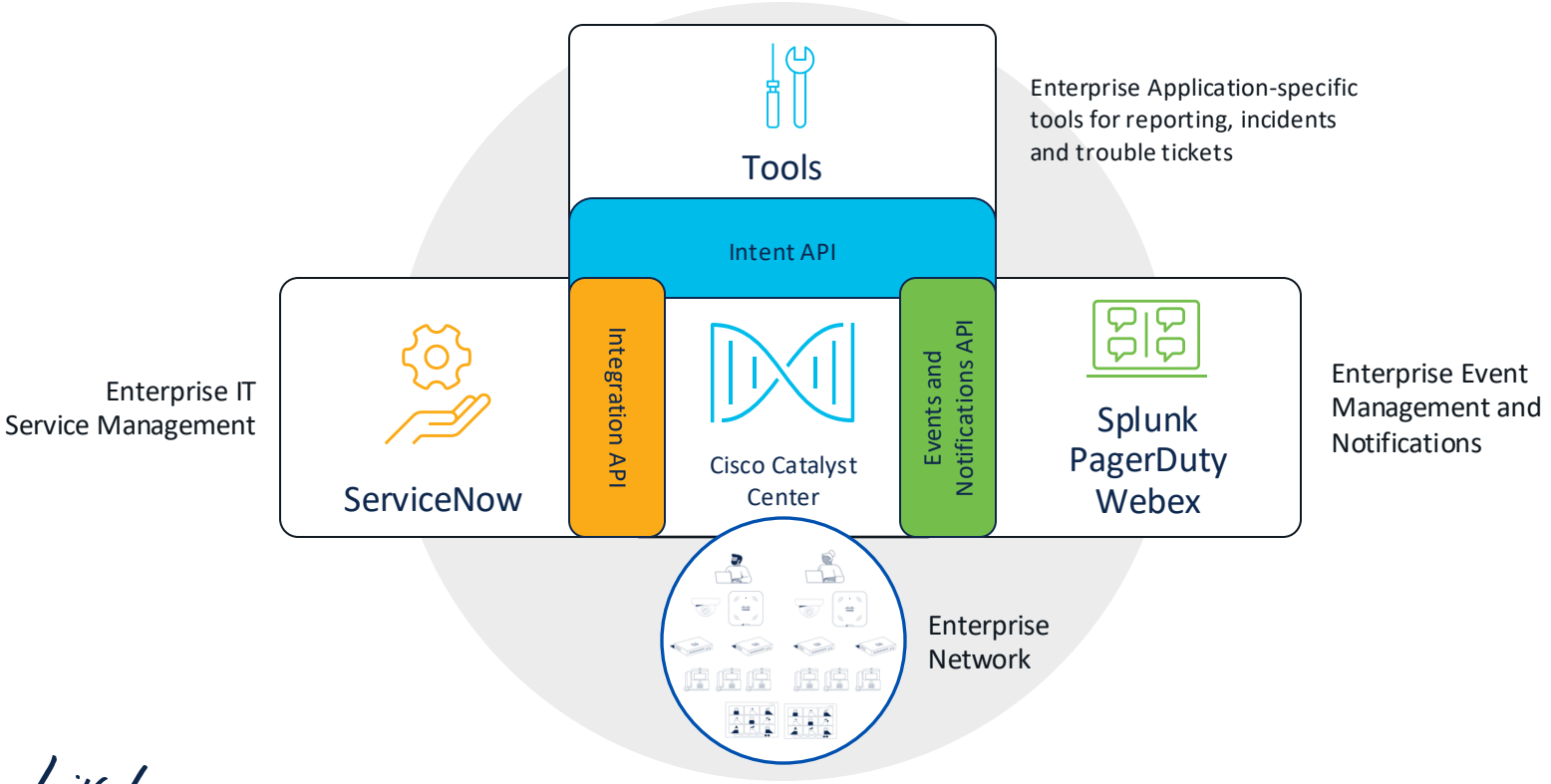
- IT Service Management
- IP Address Management
- Reporting
- Wireless Planning
- Alerting
- SIEM

Developer Resources

- Sample Code, Videos
- Python SDK, Ansible, Terraform
- Cisco DevNet
  - Sandboxes, Learning Labs
  - Developer Guides
  - Sample Code

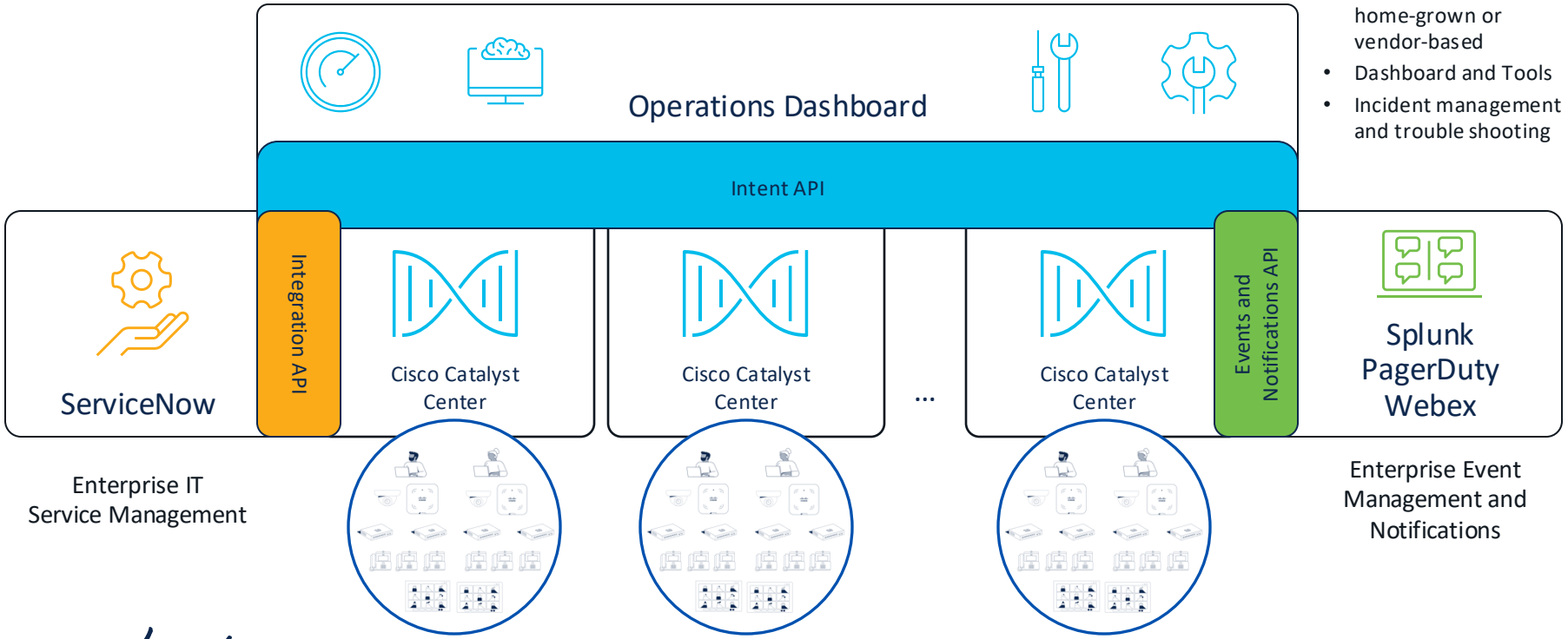
Cisco Catalyst Center enterprise architecture

Align to ITIL framework



Cisco Catalyst Center architecture

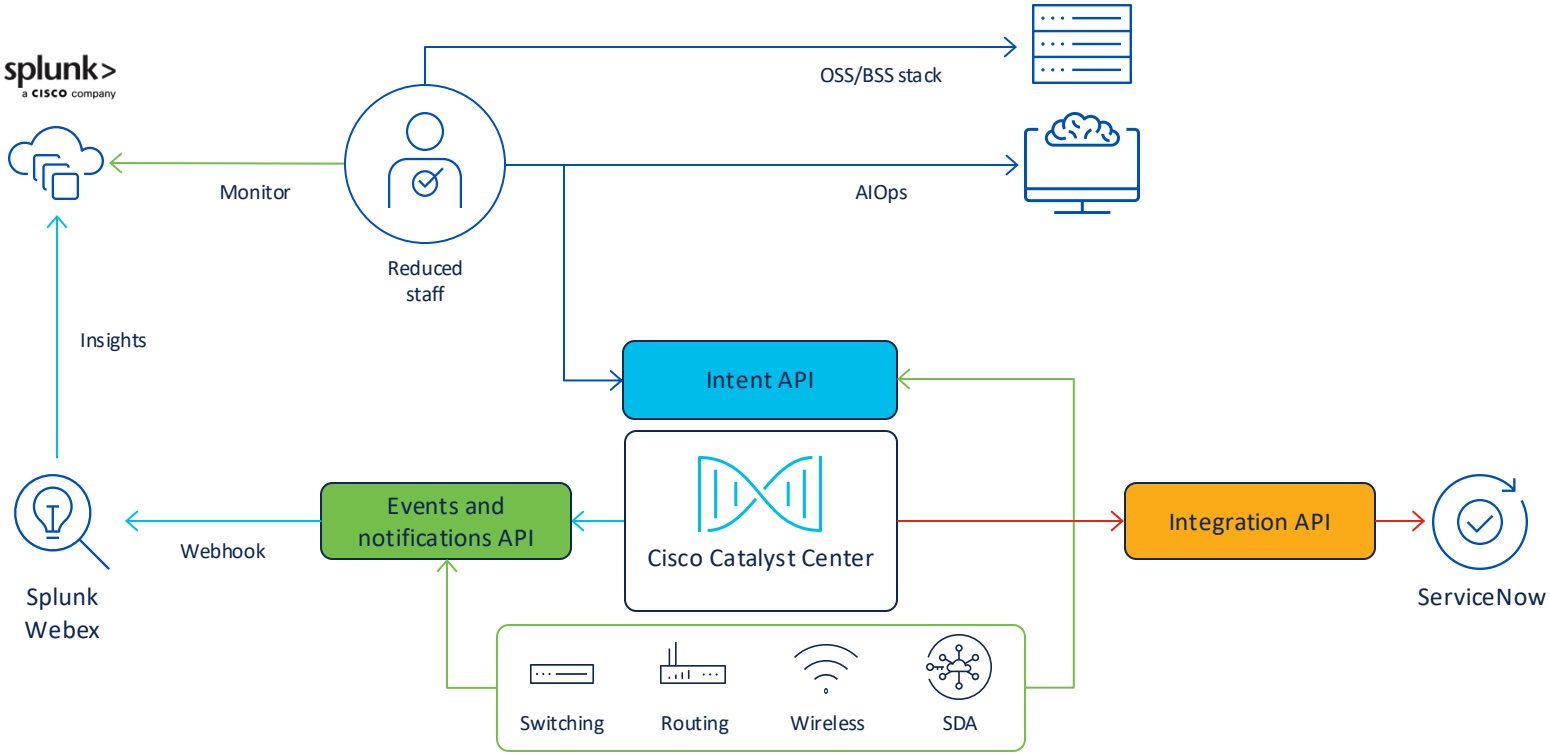
Headless operations: operate Catalyst Center at scale



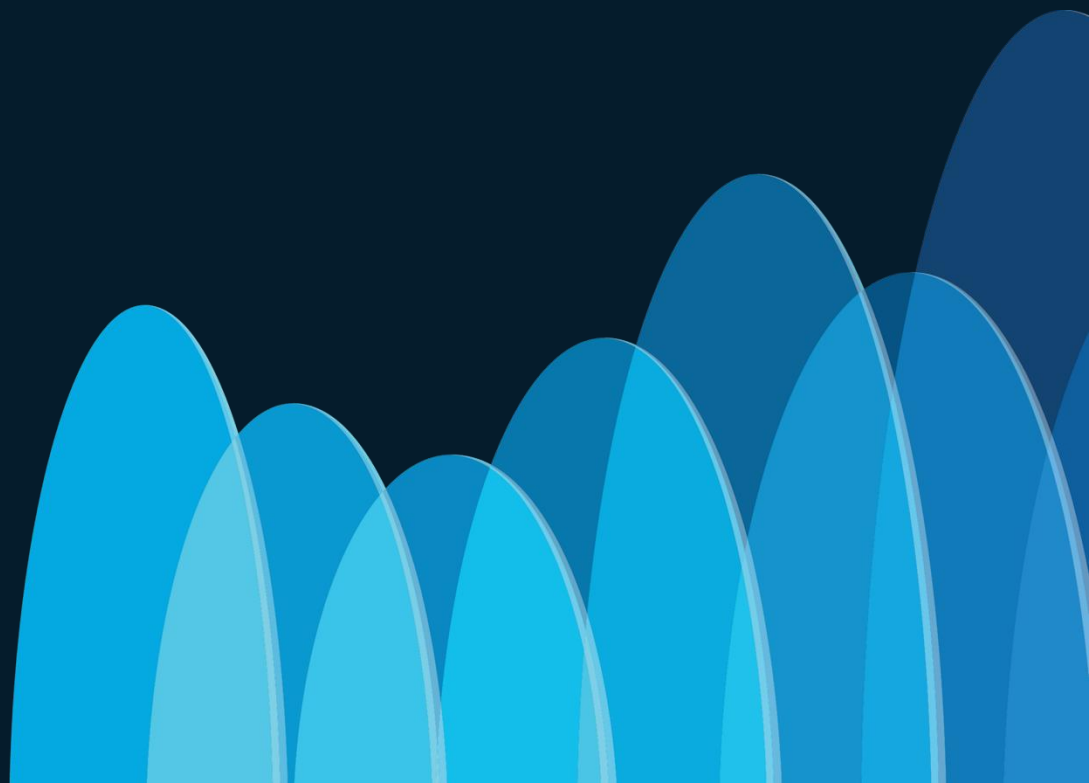
- OSS/BSS stack – home-grown or vendor-based
- Dashboard and Tools
- Incident management and trouble shooting

A high-level view

Catalyst Center headless deployment example

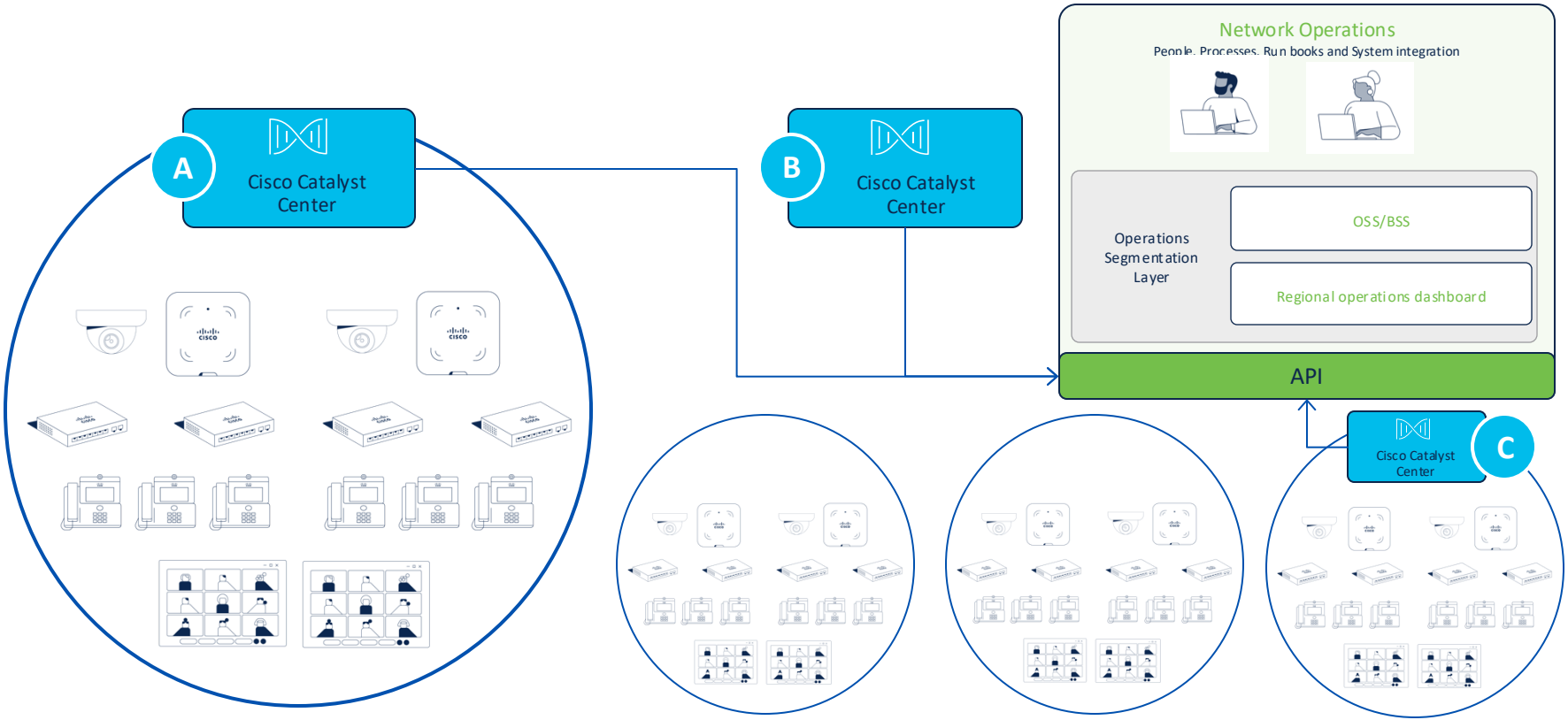


Catalyst Center APIs



cisco Live!

What a headless scenario looks like



Cisco Catalyst Center API platform

Event and Notifications API

Provides a notification handler when specific events are triggered, such as Network Assurance and Automation (SWIM) events.

Enables external systems to take actions in response to an event: execute a software upgrade action in response to an out of compliance notification of network devices

Notifications may also be triggered by internal Catalyst Center events. For example, Assurance events can be customised for IT Service Management incidents.

Intent API

The Intent API is a REST API that exposes specific capabilities of the Cisco Catalyst Center platform.

Provides policy-based abstraction of business intent, allowing focus on an outcome rather than struggling with individual mechanisms steps.

The RESTful Cisco Catalyst Center Intent API uses HTTPS verbs (GET, POST, PUT, and DELETE) with JSON structures to discover and control the network.

Integration API

Provides mechanisms for integrating Network Assurance end-to-end workflows and data with third-party IT Service Management (ITSM) solutions.

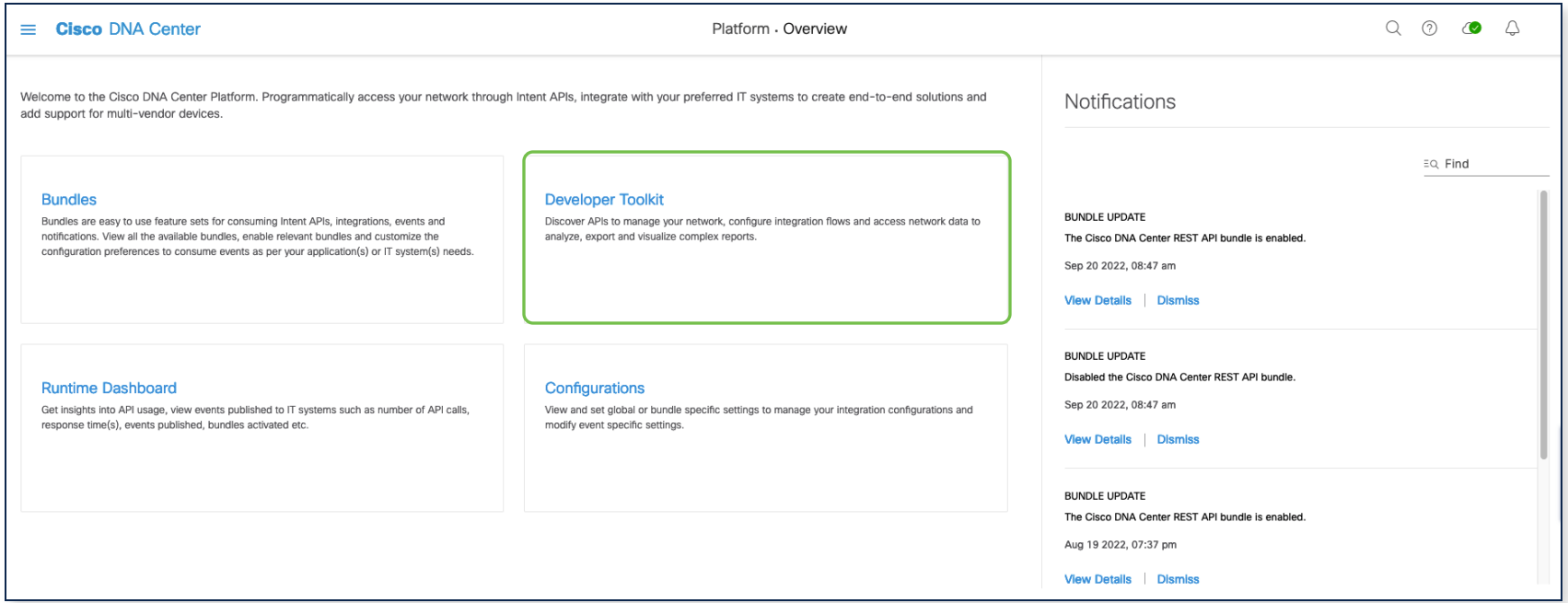
ITSM integration minimises handoffs, reduces duplication of issues, and optimizes processes such as approval- and pre-approval chains,

Cisco Catalyst Center also Integrates with Reporting and Analytics capabilities for capacity planning, asset management, compliance control, and auditing.

cisco Live!

Cisco Catalyst Center

Platform overview



Options for CatC Labs

- DevNet Catalyst Center Sandbox
- dCloud

CatC API documentation

- In CatC in the Platform overview
- In DevNet
  - https://developer.cisco.com/docs/dna-center/cisco-dna-center-2-3-7-api-overview/

Lab introduction

Lab description

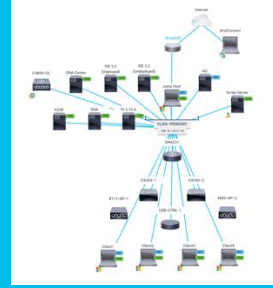
Prerequisites

Laptop with Cisco Anyconnect client

Internet

Jump windows host

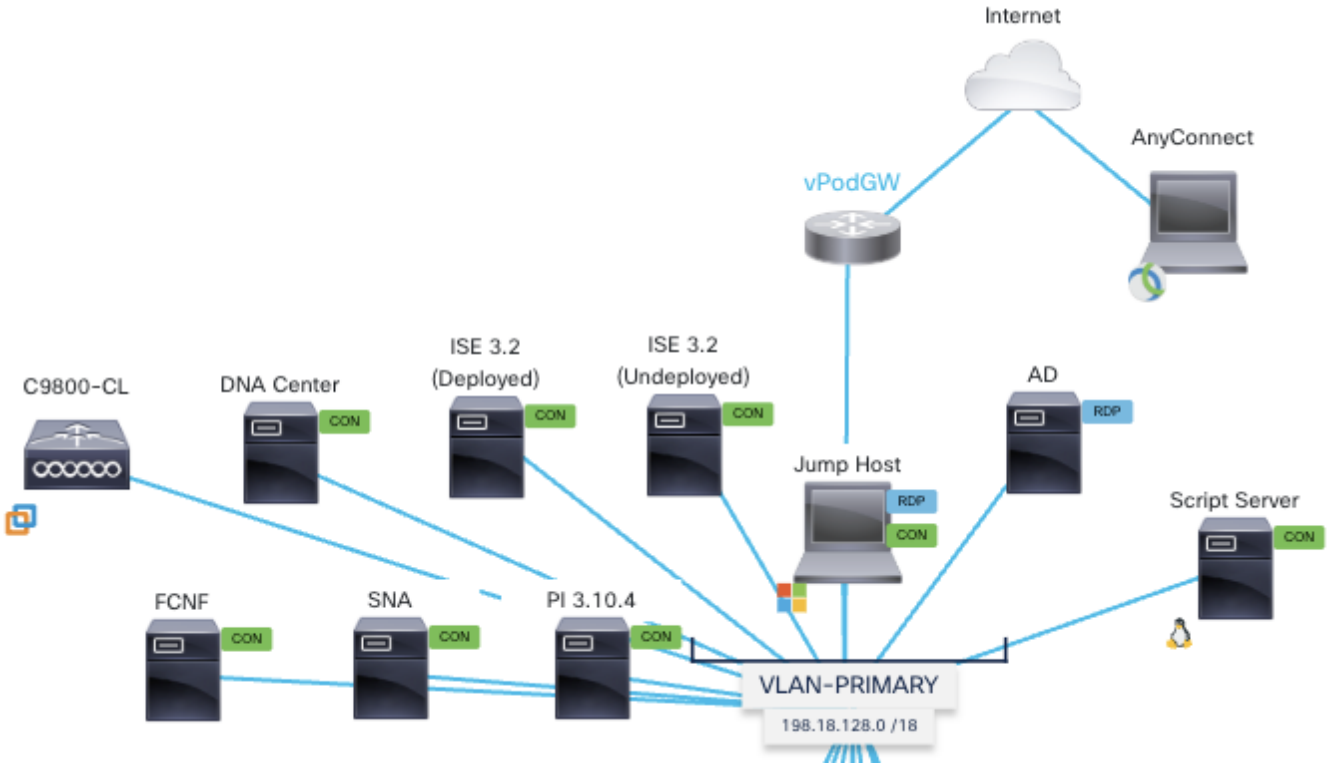
User preferable with Postman account



Sandbox

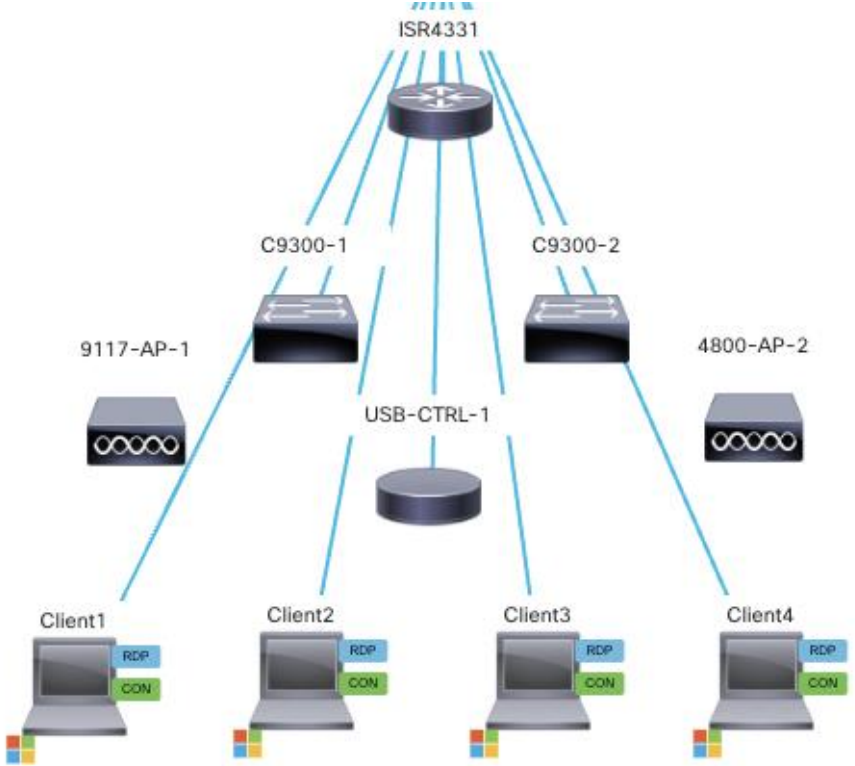
Cisco Enterprise Networks Sandbox v4.1

Jump host and servers including CatC



Cisco Enterprise Networks Sandbox v4.1

Network devices and clients



What is in this environment?

Virtual Machines

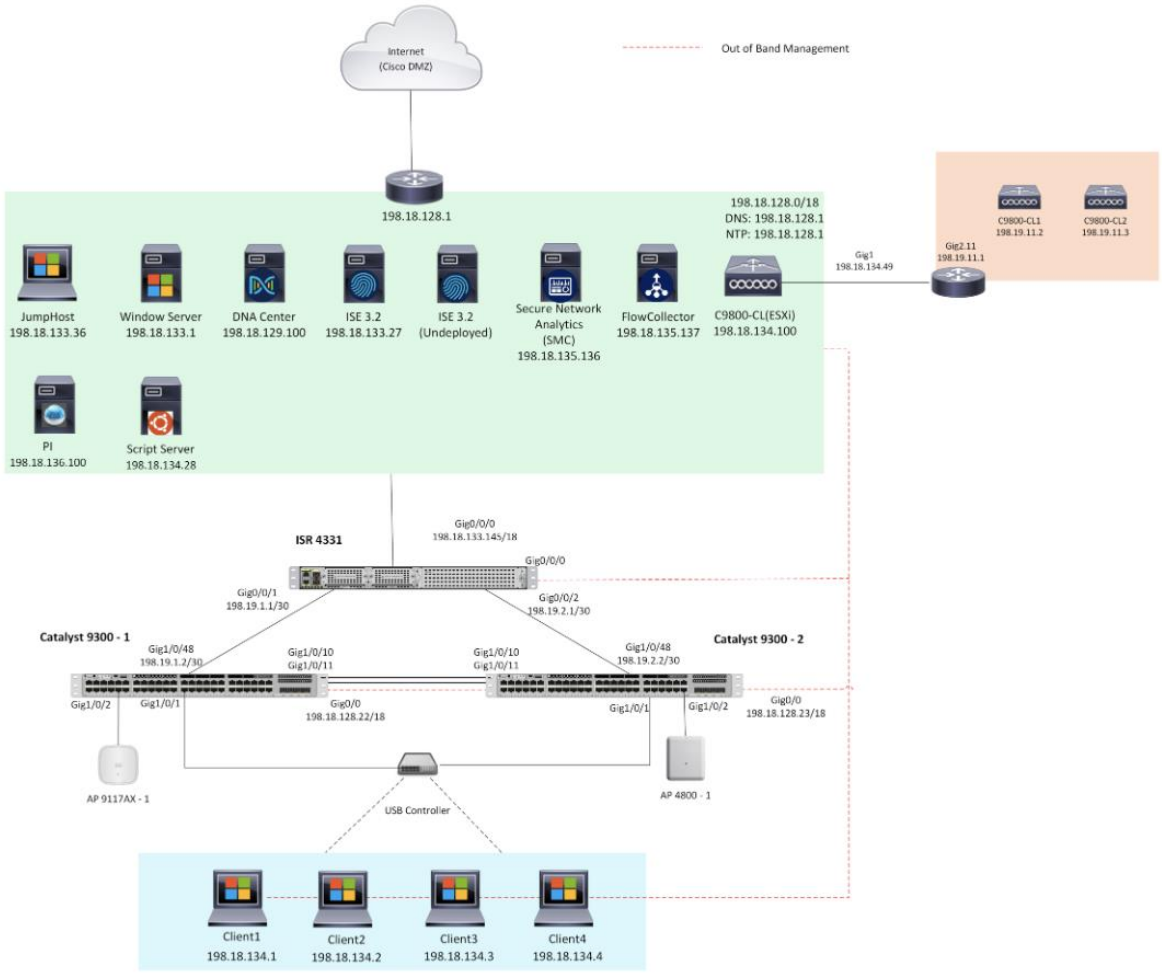
- DNA Center 2.3.5.5
- Identity Services Engine (ISE) 3.2 Patch 5 – (Deployed)
- Identity Services Engine (ISE) 3.2 – (Undeployed)
- Secure Network Analytics (Stealthwatch) 7.4.2
- FlowCollector 7.4.2
- Cisco Prime Infrastructure 3.10.4
- Wireless LAN Controllers – C9800 running IOS-XE 17.12.3 code
- Windows 10 Jump Host – Contains links to common URLs needed to view and configure the environment. Can also be used to TFTP files to/from the hardware devices. Can be used to pull files from Box.
- Ubuntu 20.04.3 – Can be used as a script server and for programing tools.
- Windows Server 2019 – Can be configured to provide identity, DHCP, DNS, etc.
- Windows 10 Clients – Used to simulate network clients participating in the environment. Can be used to test segmentation, host onboarding, policy implementation, etc.

What is in this environment?

Hardware devices

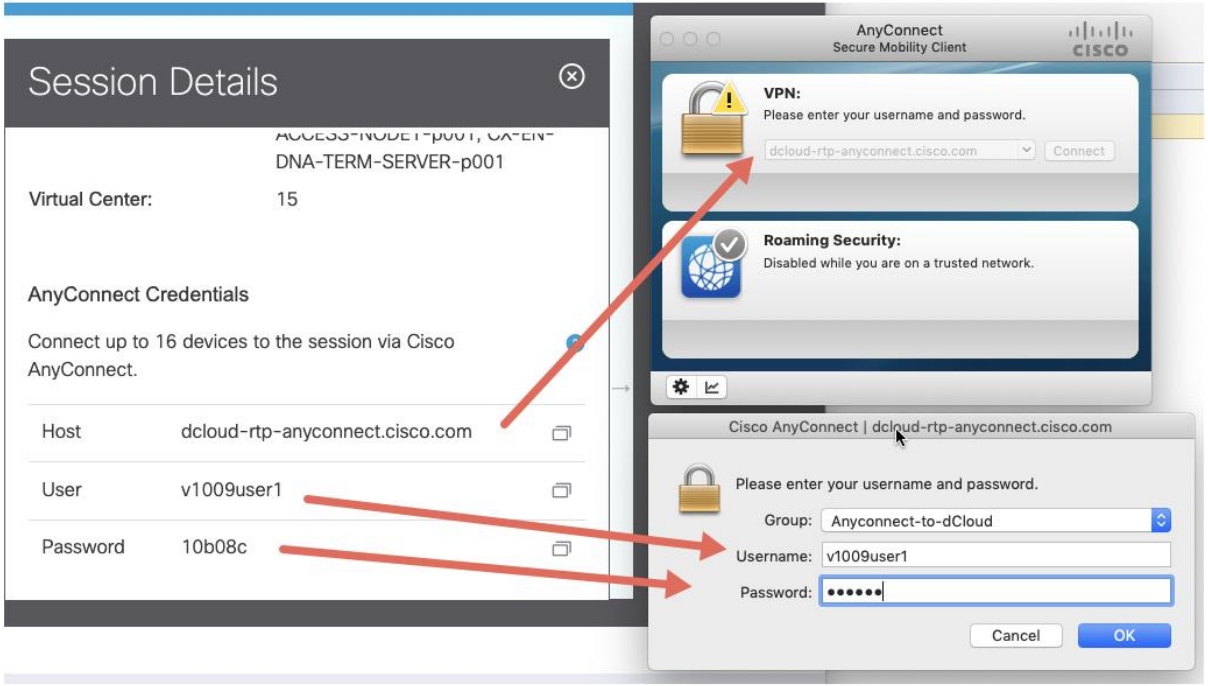
- ISR 4331 Router - 17.09.02a IOS-XE Code
- Catalyst 9300 Switches - 17.09.02 IOS-XE Code with Embedded Wireless Controller (EWC) and ThousandEyes Enterprise Agent
- 9117 Access Point
- 4800 Access Points
- Silex Controllers (2 Wired NICs)

Topology



How to access your environment?

Print session details the day before

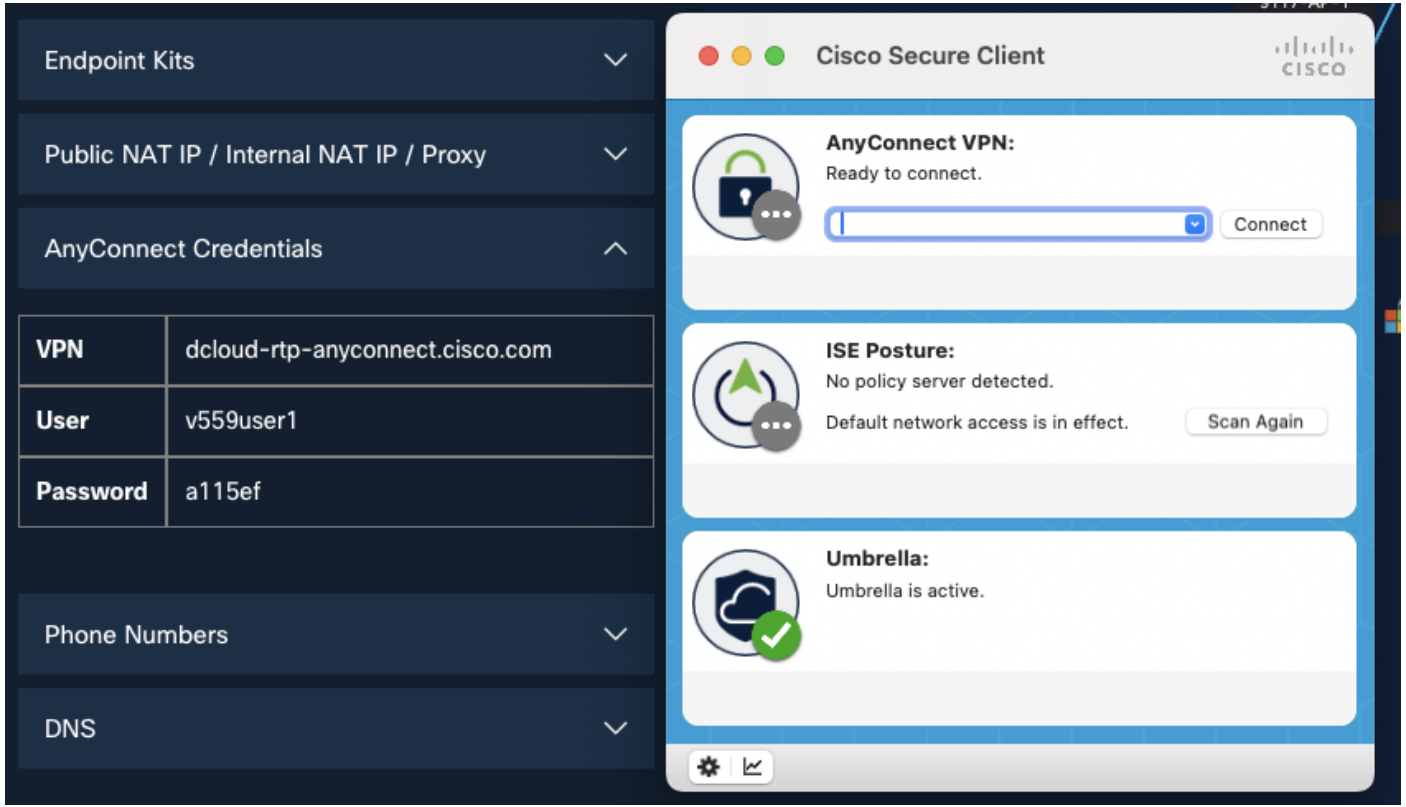


1. Connect to the demo with Cisco AnyConnect VPN. Find your AnyConnect information in the Session Details for your dCloud session.
2. Use your local browser to connect to URLs as outlined in the table in the Virtual Machines section that follows.
3. Use your local RDP client to connect to RDP-capable Virtual Machines as outlined in Virtual Machines section that follows.

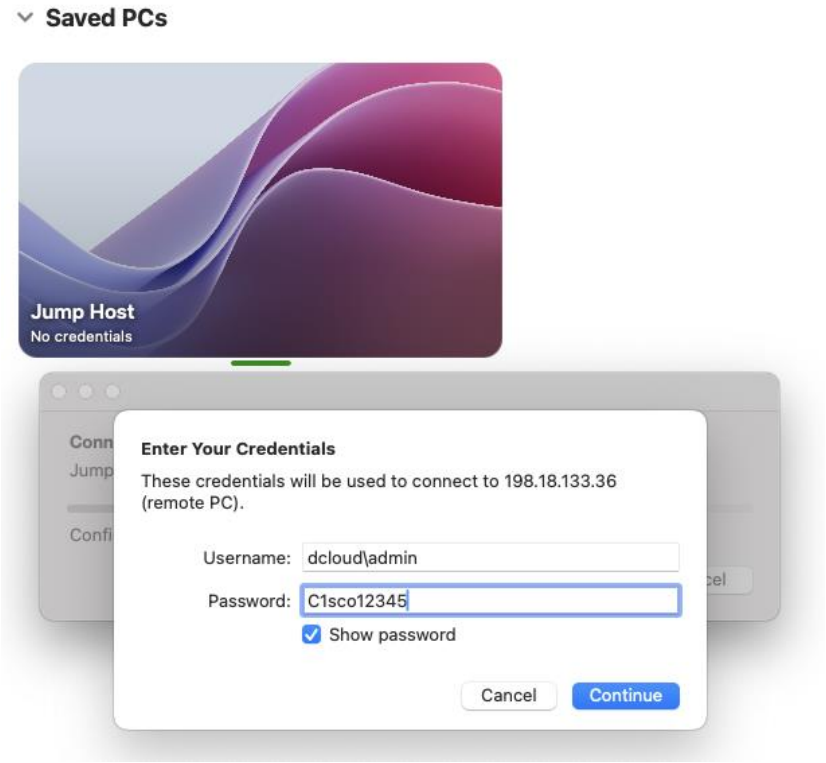
Virtual Machine Addressing and Credentials

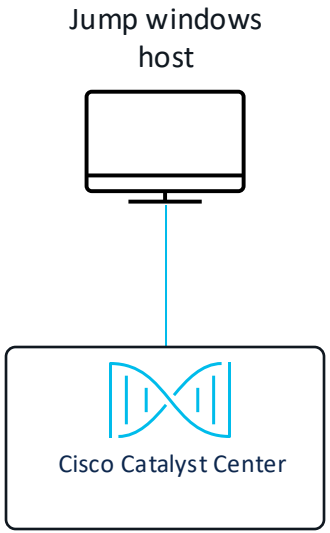
IP Address	Name	Username	Password	Preferred Access Method
Entry	ISE 3.0 Undeployed	Entry	Entry	VM Console via dCloud UIChrome or Firefox (once configured)
198.18.133.27	ISE 3.0 (Deployed)	admin	C1sco12345	Chrome or Firefox
198.18.129.100	DNA Center	admin	C1sco12345	Chrome or Firefox
198.18.135.136	Secure Network Analytics (Stealthwatch)	admin	C1sco12345	Chrome or Firefox
198.18.135.137	FlowCollector	admin	C1sco12345	Chrome or Firefox
198.19.11.2	C9800-CL1	admin	C1sco12345	Chrome or FirefoxSSH or Telnet
198.19.11.3	C9800-CL2	admin	C1sco12345	Chrome or FirefoxSSH or Telnet
198.18.134.49	ESXi-Router	admin	C1sco12345	SSH or Telnet
198.18.136.100	Cisco Prime Infrastructure	admin	C1sco12345	Chrome or Firefox
198.18.134.28	Script Server	root	C1sco12345	Chrome or FirefoxSSH
198.18.133.36	Jump Host	admin	C1sco12345	RDP
198.18.133.1	AD	admin	C1sco12345	RDP
198.18.134.1	Client1	admin	C1sco12345	RDP
198.18.134.2	Client2	admin	C1sco12345	RDP
198.18.134.3	Client3	admin	C1sco12345	RDP
198.18.134.4	Client4	admin	C1sco12345	RDP

Cisco Secure Client

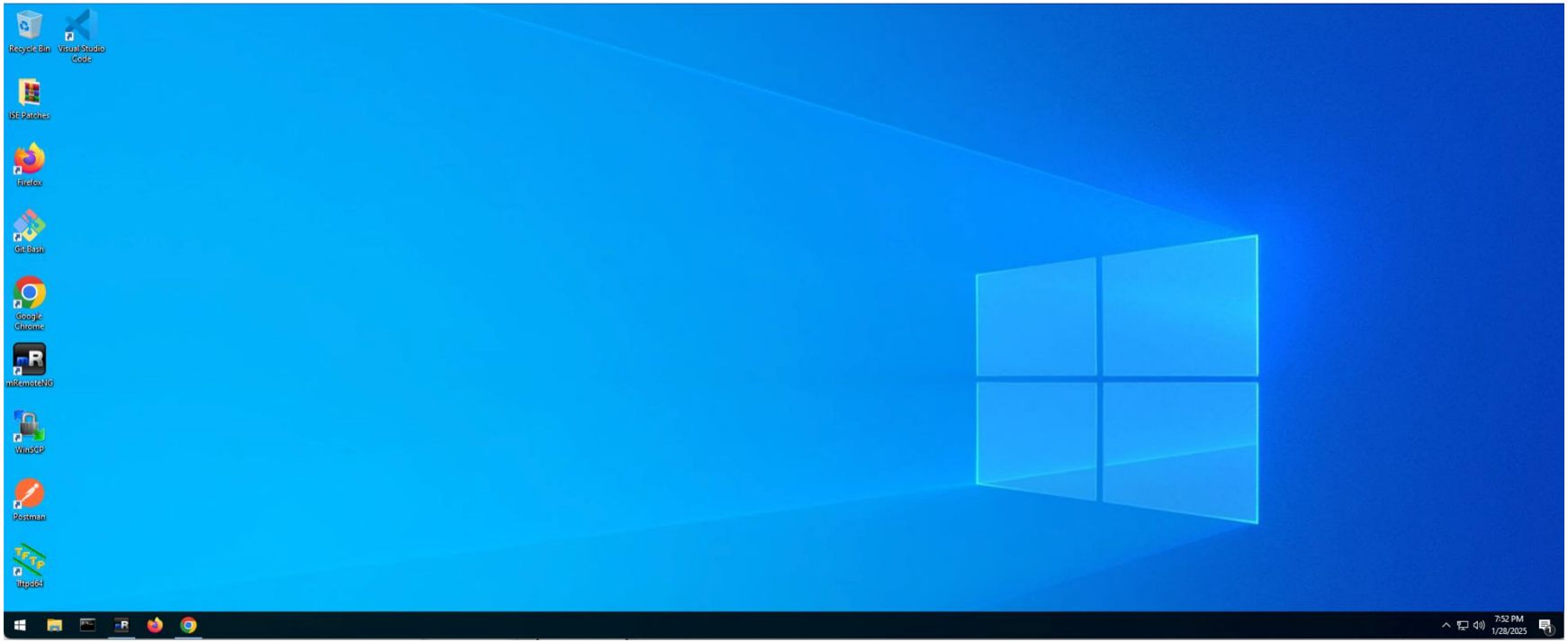


RDC to jump host





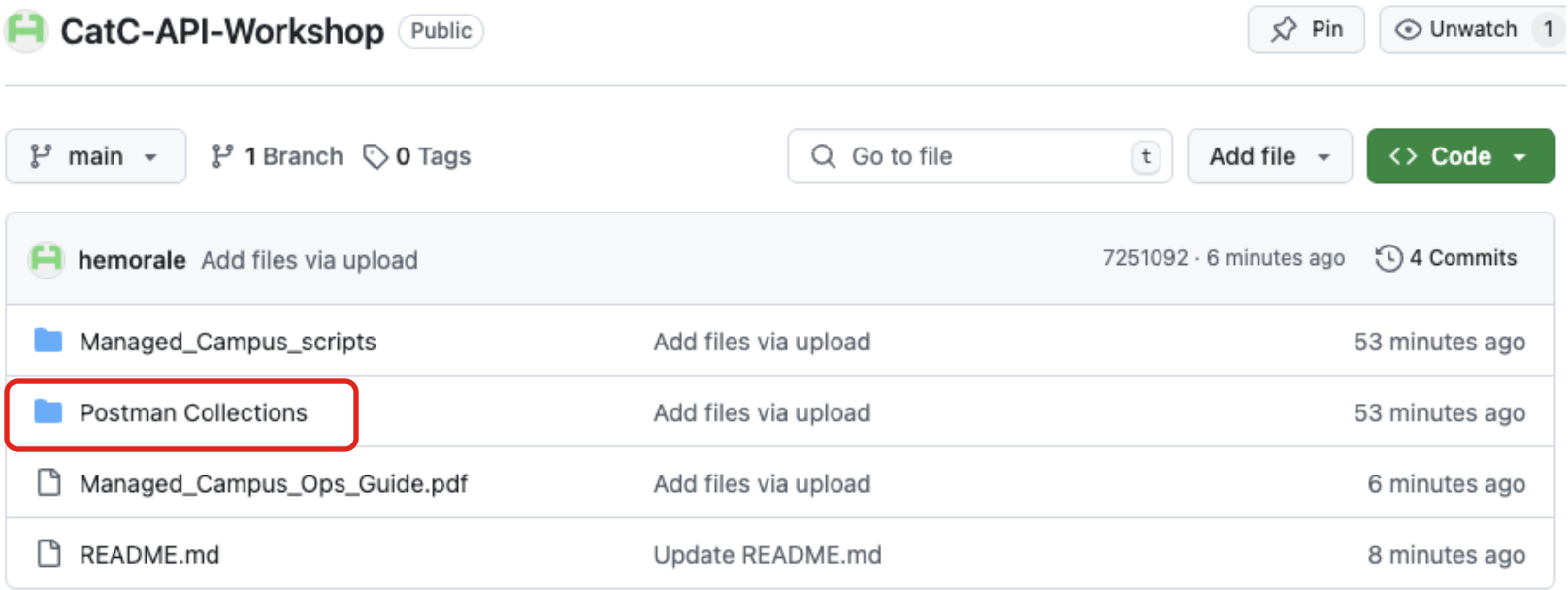
Windows Jump Host



CatC APIs 100-200 level Postman collections

cisco Live!

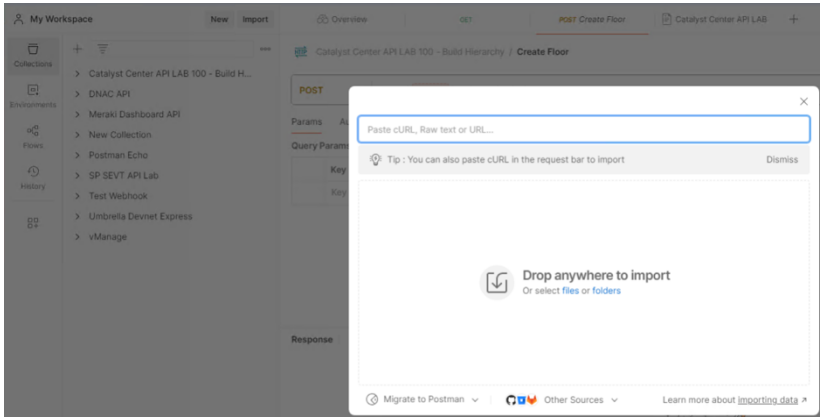
GitHub



https://github.com/hemorale/CatC-API-Workshop

Postman collections

- Connect to the dCloud lab using your assigned PoD via Cisco Anyconnect
- Using the Windows App, connect into the Jump Host
- Open Chrome and download the Postman collections in the jump host.
- Open Postman. If necessary login into your account or create a new account. You will need it to upload collections
- Click on Collections. Then click on Import. Select the json files.

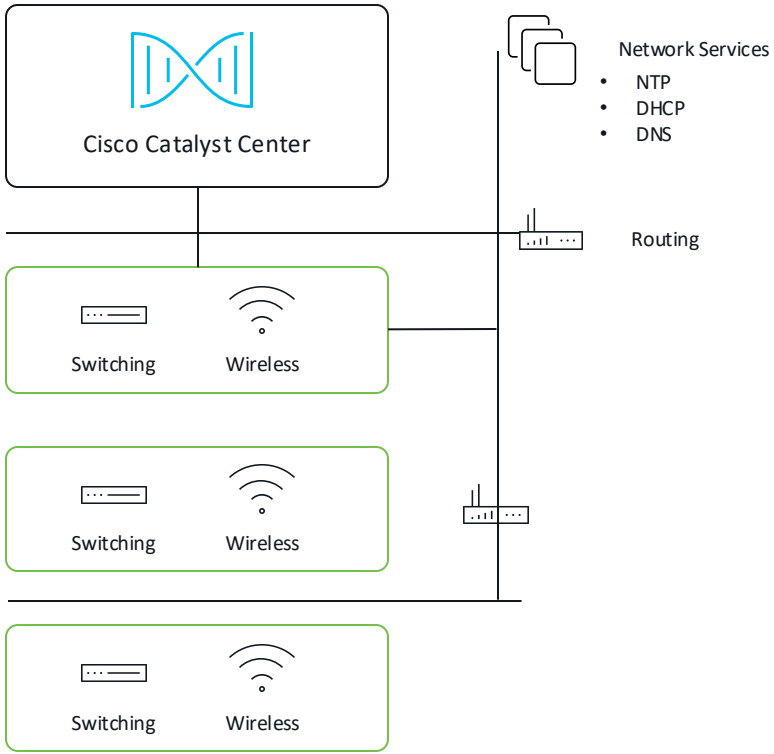


Postman collections

- **Catalyst Center API Environment.postman_environment.json** - provides all environment variables. This should be import in the Environments and this will be used for the rest of the collections
- **Catalyst Center API LAB 100 - Build Hierarchy.postman_collection.json** – builds the initial CatC network hierarchy
- **Catalyst Center API LAB 101 - Assign Settings Creds.postman_collection.json** - assign all CatC lab credentials
- **Catalyst Center API LAB 201 - Device Discovery.postman_collection.json** - device discovery
- **Catalyst Center API LAB 302 - Template Deployment.postman_collection.json** – deploys a switch template
- **Catalyst Center API LAB 304 - Configuration Archive.postman_collection.json** - configuration archive
- **Catalyst Center API LAB 401 - Device Inventory.postman_collection.json** – device inventory
- **Catalyst Center API LAB 402 - Command Runner.postman_collection.json** – command runner

Catalyst Center basic configuration

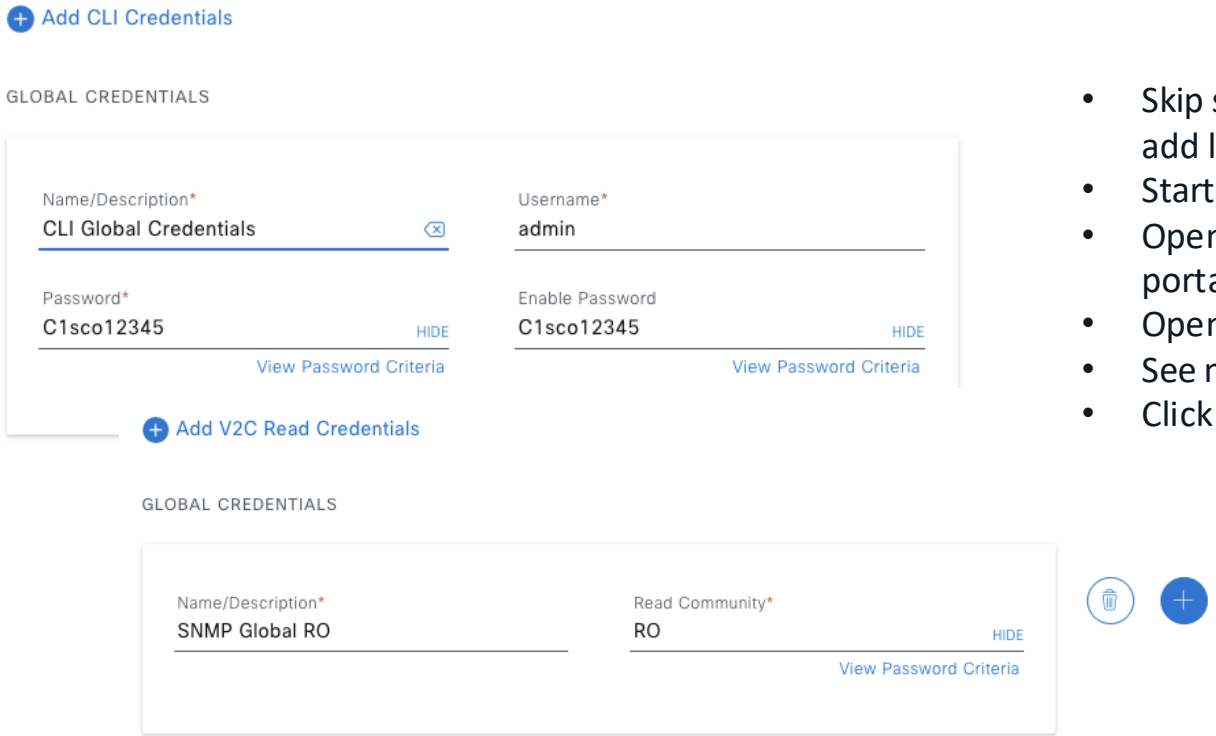
Topology discovery



Connect to Catalyst Center

1. In Jump host, open Chrome
2. Click on DNA Center
3. Login
4. Click on Add sites
5. Click on Discovery
6. Add Discovery
7. Click Next
8. Discovery type CDP
9. IP Address: 198.18.133.145
10. Use Loopback
11. Click Next

Catalyst Center basic configuration

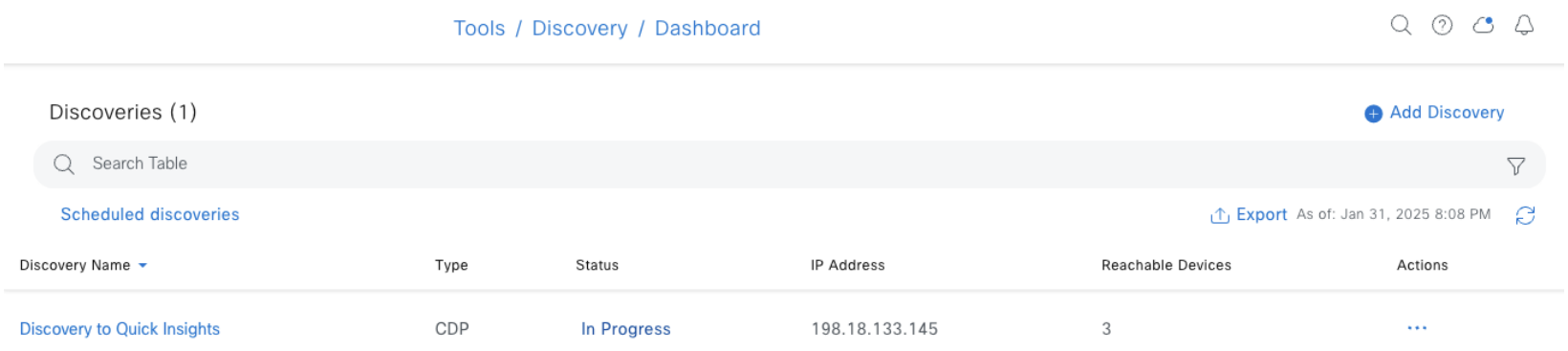


- Skip site assignment for now. We will add later from inventory.
- Start discovery
- Open Hardware console on dcloud portal
- Open ISR console
- See messages
- Click on view discovery

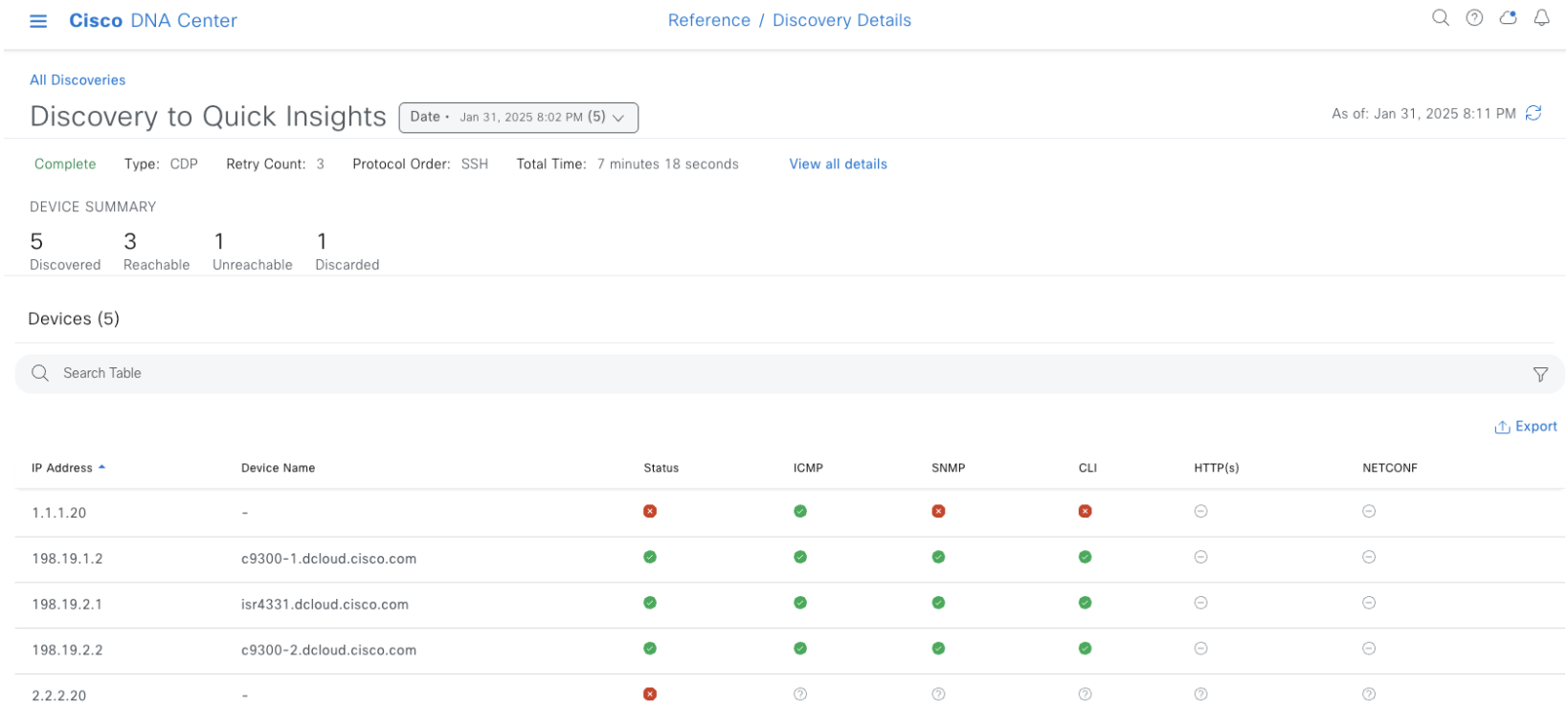
Discovery process

```
 login authentication CONSOLE
 stopbits 1
line aux 0
 --More--
*Feb  1 03:10:07.127: %CRYPTO_ENGINE-5-CSDL_COMPLIANCE_EXCEPTION_ADDED: Cisco PSB security compliance exception has been added by SSH Process for use
of MD5
*Feb  1 03:10:07.302: %SEC_LOGIN-5-LOGIN_SUCCESS: Login Success [user: admin] [Source: 198.18.129.100] [localport: 22] at 03:10:07 UTC Sat Feb 1 2025
*Feb  1 03:10:07.462: %SYS-6-LOGOUT: User admin has exited tty session 867(198.18.129.100)
*Feb  1 03:10:20.086: %SEC_LOGIN-5-LOGIN_SUCCESS: Login Success [user: admin] [Source: 198.18.129.100] [localport: 22] at 03:10:20 UTC Sat Feb 1 2025
*Feb  1 03:10:20.323: %SYS-5-CONFIG_I: Configured from console by admin on vty1 (198.18.129.100)
*Feb  1 03:10:27.214: %SYS-6-PRIVCFG_ENCRYPT_SUCCESS: Successfully encrypted private config file
*Feb  1 03:10:30.740: %CRYPTO_ENGINE-5-CSDL_COMPLIANCE_EXCEPTION_ADDED: Cisco PSB security compliance exception has been added by SSH Process for use
of MD5
*Feb  1 03:10:30.865: %SEC_LOGIN-5-LOGIN_SUCCESS: Login Success [user: admin] [Source: 198.18.129.100] [localport: 22] at 03:10:30 UTC Sat Feb 1 2025
*Feb  1 03:10:30.950: %SYS-6-LOGOUT: User admin has exited tty session 868(198.18.129.100)
*Feb  1 03:10:33.067: %SEC_LOGIN-5-LOGIN_SUCCESS: Login Success [user: admin] [Source: 198.18.129.100] [localport: 22] at 03:10:33 UTC Sat Feb 1 2025
*Feb  1 03:10:34.946: %SEC_LOGIN-5-LOGIN_SUCCESS: Login Success [user: admin] [Source: 198.18.129.100] [localport: 22] at 03:10:34 UTC Sat Feb 1 2025
*Feb  1 03:10:40.515: %SEC_LOGIN-5-LOGIN_SUCCESS: Login Success [user: admin] [Source: 198.18.129.100] [localport: 22] at 03:10:40 UTC Sat Feb 1 2025
*Feb  1 03:10:51.305: %SYS-6-LOGOUT: User admin has exited tty session 868(198.18.129.100)
*Feb  1 03:10:51.552: %SEC_LOGIN-5-LOGIN_SUCCESS: Login Success [user: admin] [Source: 198.18.129.100] [localport: 22] at 03:10:51 UTC Sat Feb 1 2025
*Feb  1 03:11:46.253: %SEC_LOGIN-5-LOGIN_SUCCESS: Login Success [user: admin] [Source: 198.18.129.100] [localport: 22] at 03:11:46 UTC Sat Feb 1 2025
*Feb  1 03:12:16.157: %SYS-6-TTY_EXPIRE_TIMER: (exec timer expired, tty 866 (198.18.133.36)), user admin
*Feb  1 03:12:16.157: %SYS-6-LOGOUT: User admin has exited tty session 866(198.18.133.36)
*Feb  1 03:13:31.206: %SYS-6-LOGOUT: User admin has exited tty session 868(198.18.129.100)
*Feb  1 03:13:53.733: %SYS-6-LOGOUT: User admin has exited tty session 867(198.18.129.100)
```

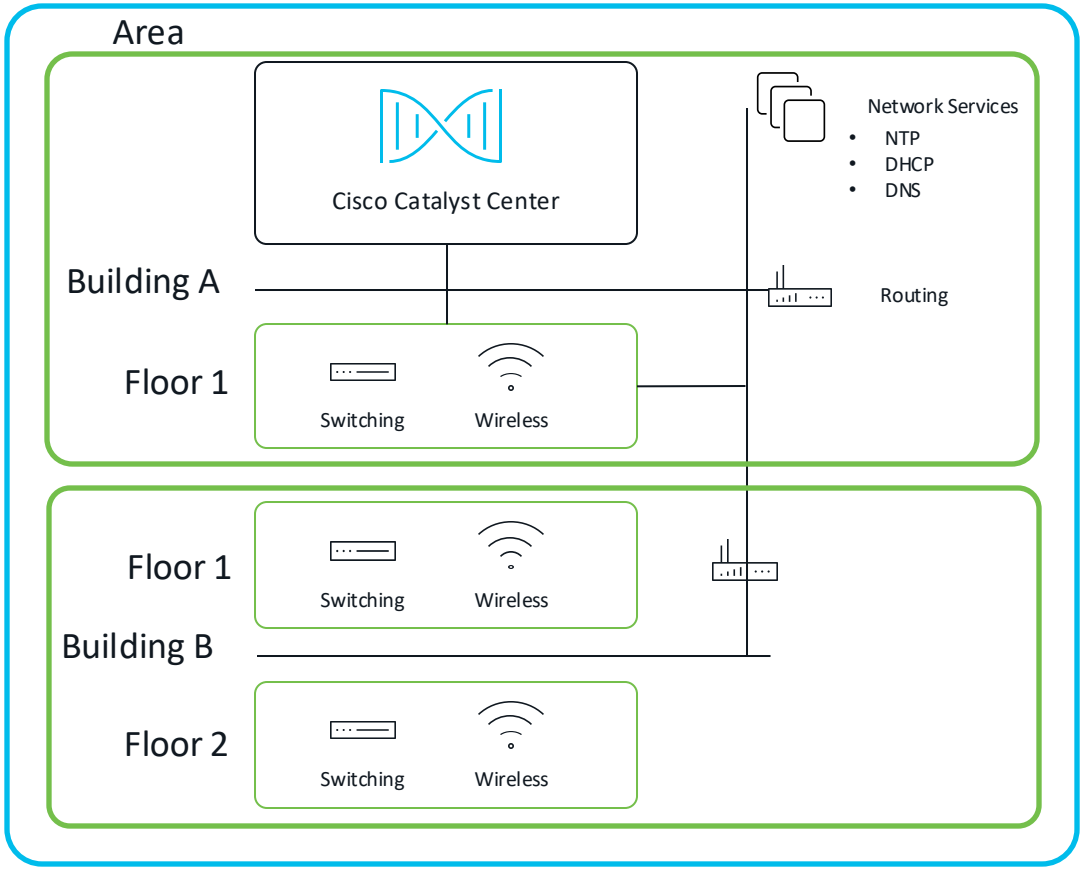
Discovery process



Discovery process



Catalyst Center basic configuration

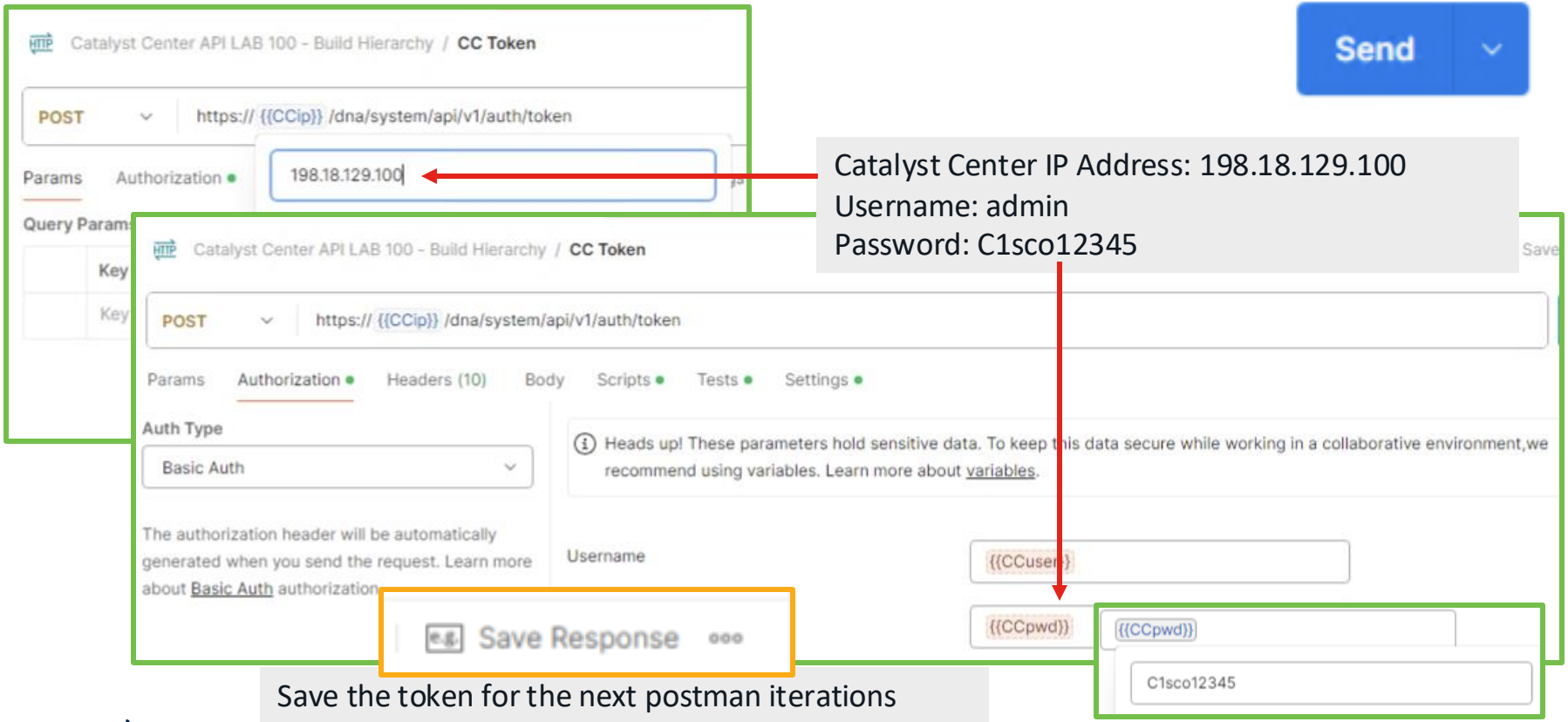


Using APIs to create Topology map

1. In Jump host, open Postman
2. Click on the collection Catalyst Center API LAB 100 - Build Hierarchy
3. This collection will create a hierarchy like this one

Creating hierarchy using Postman

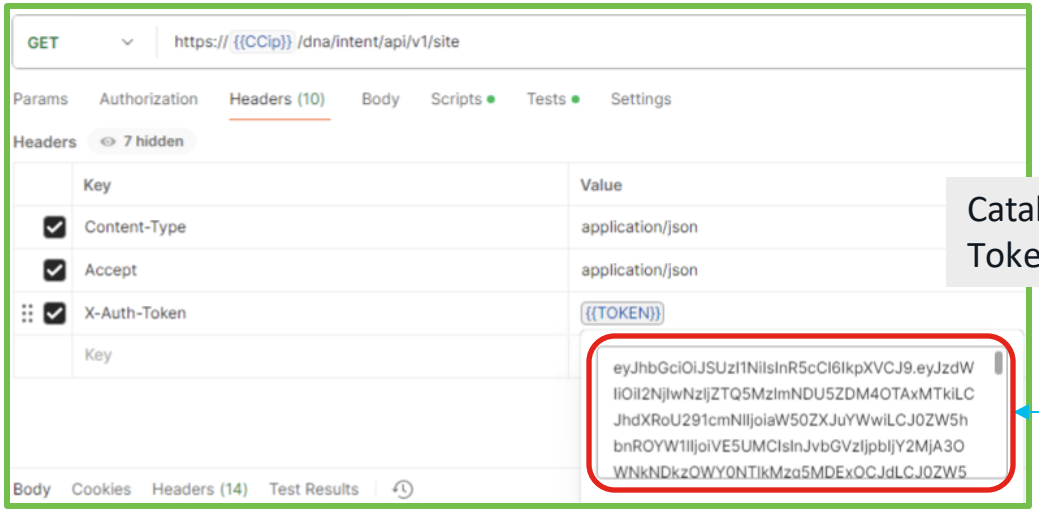
Get the Token



Save the token for the next postman iterations

Creating hierarchy using Postman

Get SiteIDs



Send

Catalyst Center IP Address: 198.18.129.100
Token: the text part of the token

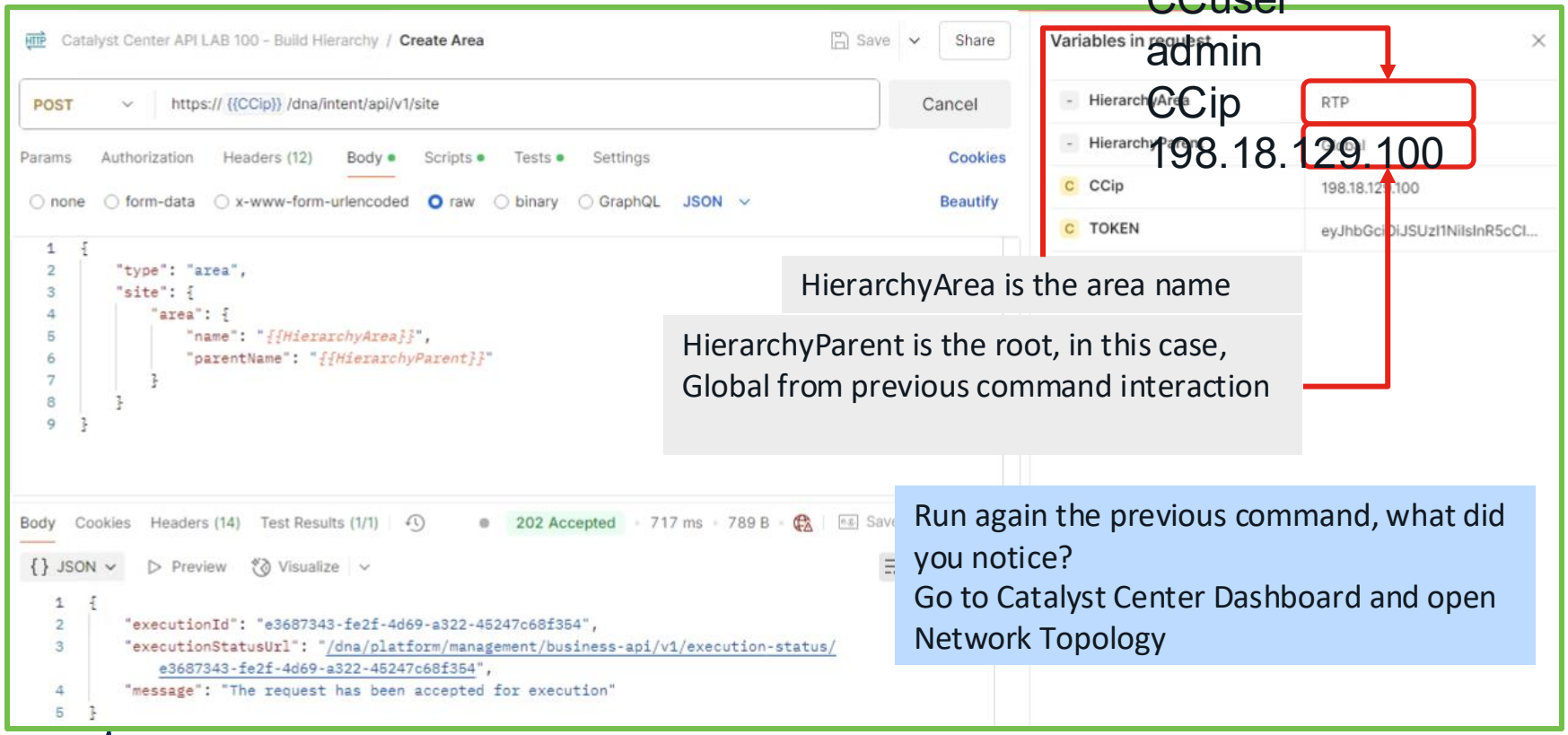
Creating hierarchy using Postman

Get SiteIDs

Creating hierarchy using Postman

Create Area

CCCCpwd
C1sco12345
CCuser
admin
CCip
198.18.129.100



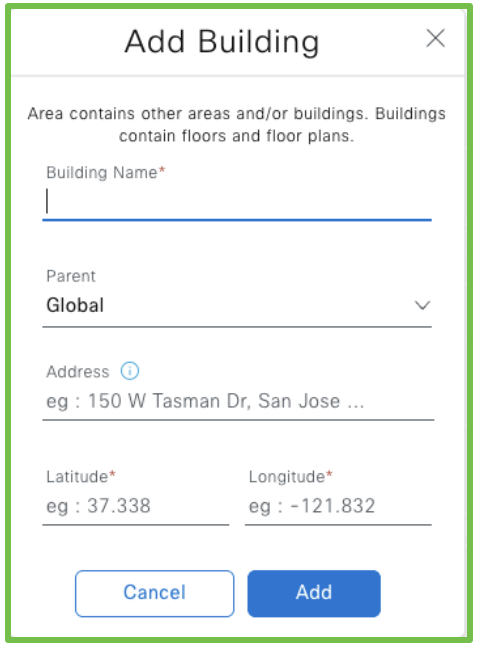
Creating hierarchy using Postman

Add Building

Using Postman collection, try on your own

Creating hierarchy using Postman

Add Building



What is wrong?

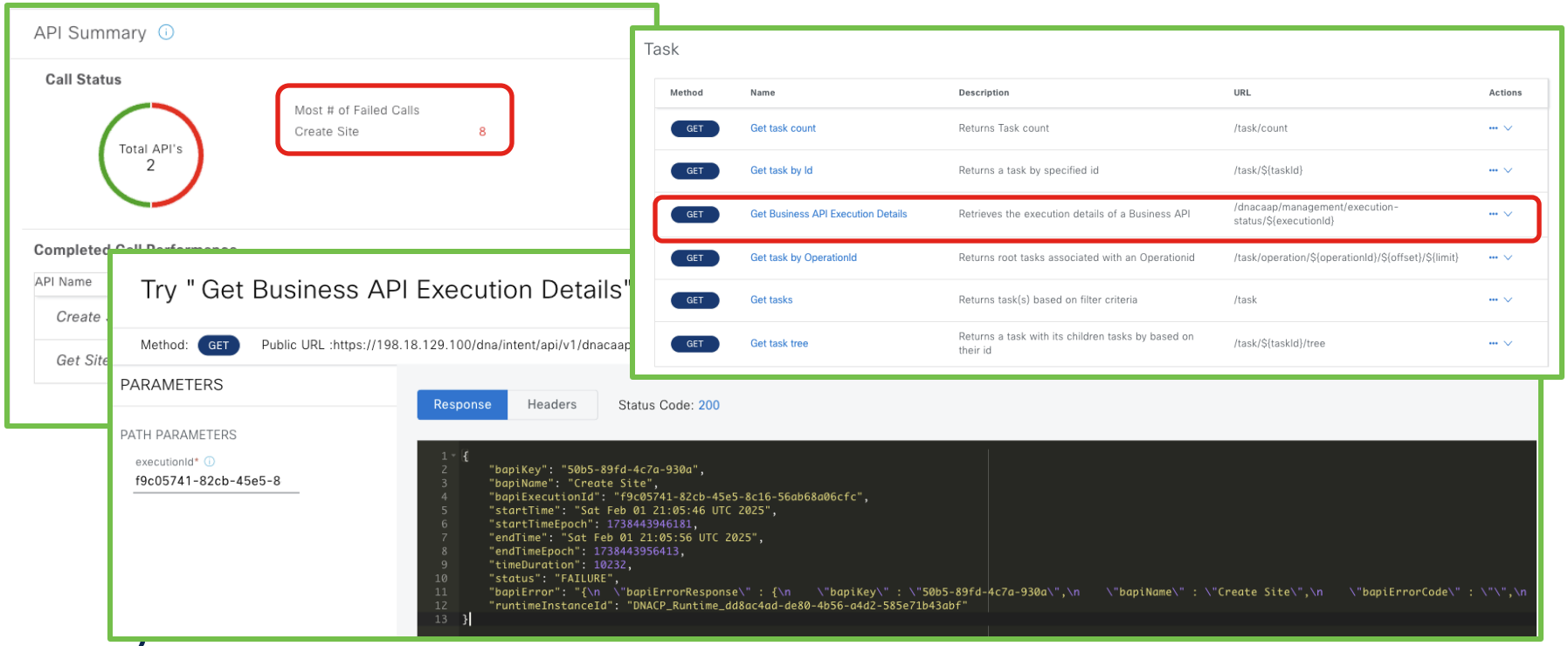
Asynchronous operations

When the Catalyst Center platform returns a 202 (Accepted) HTTP status code, the result body includes a task ID and a URL that you can use to query for more information about the asynchronous task that your original request spawned. For example, you can use this information to determine whether a lengthy task has completed.

Use CatC API Platform

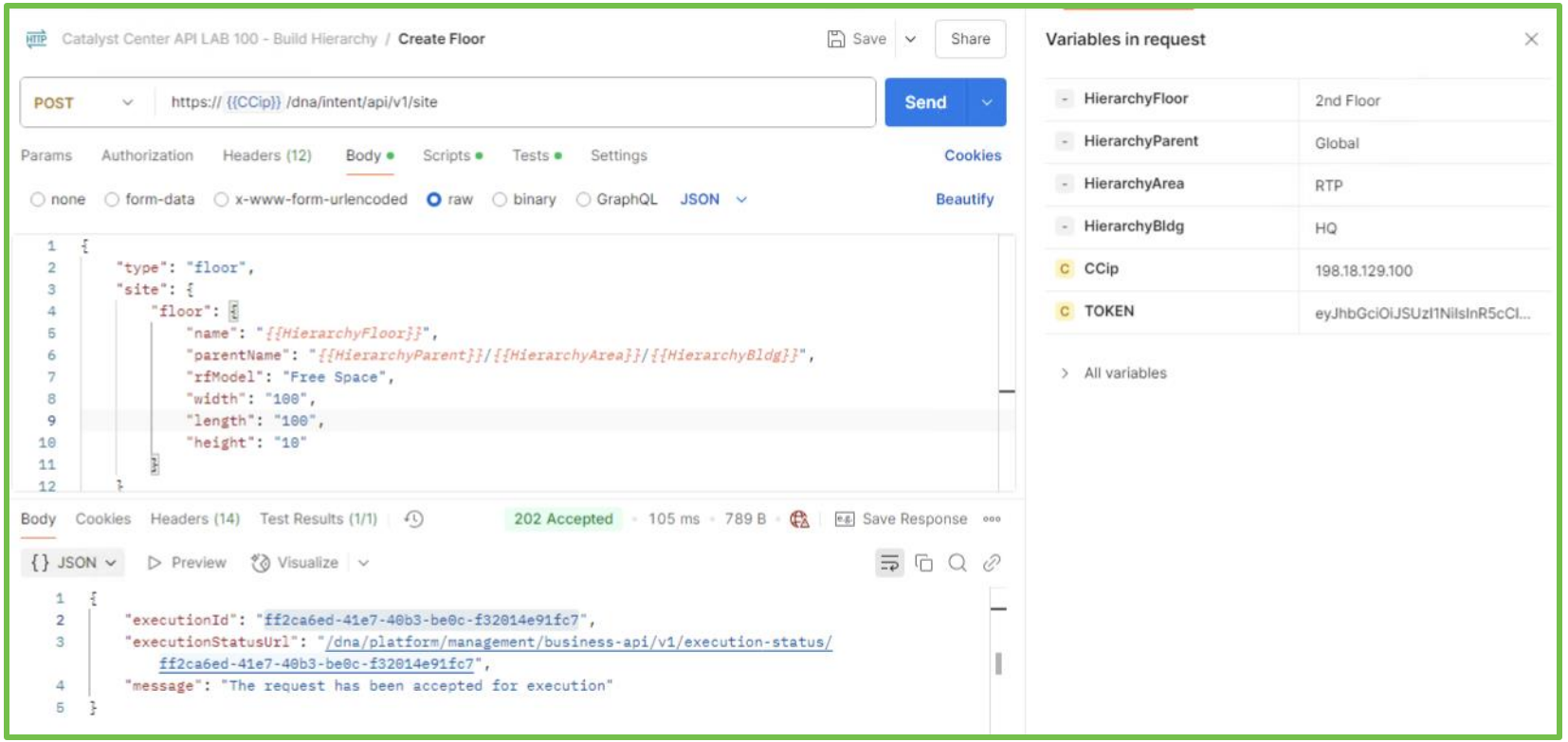
Check asynchronous operations

Platform/Runtime Dashboard



Creating hierarchy using Postman

Add floor



CatC APIs 300-400 level using jupyter notebooks

cisco Live!

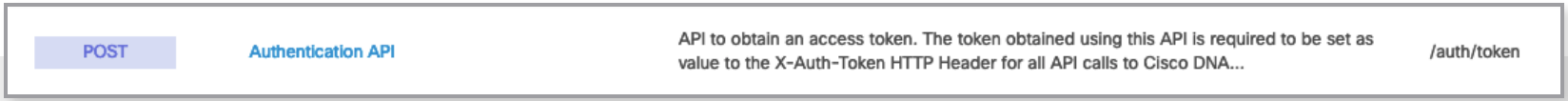
CatC APIs 300 level – Intent APIs

cisco Live!

This lab will use jupyter notebooks

- You can do this in your laptop
- Install jupyter notebooks
  1. Install Python if not installed.
  2. Create and activate a virtual environment.
  3. Run pip install jupyter.
  4. Launch Jupyter notebook with jupyter notebook.
  5. (Optional) For a more advanced UI, install and launch JupyterLab.
- See the file in github for detailed instructions
- Directory for this lab:
  - https://github.com/hemorale/CatC-API-Workshop/blob/main/Intent%20and%20Event%20Management/Managed%20DNAC%20v3.ipynb

Authentication



POST	Authentication API	API to obtain an access token. The token obtained using this API is required to be set as value to the X-Auth-Token HTTP Header for all API calls to Cisco DNA...	/auth/token

```
def get_token(dnac_ip,uname,passwd):
    token = requests.post(
        'https://' + str(dnac_ip)+ '/dna/
        auth=HTTPBasicAuth(
            username=uname,
            password=passwd
        ),
        headers={'content-type': 'applica
        verify=False,
    )
    data = token.json()
    return data['Token']
```

```
# DNAC code
# Define variables
dnac_ip = <Your DNAC DNS/IP>
username = <Your DNAC username>
password = <Your DNAC password>

token = get_token(dnac_ip, username, password)

print(token)
```

cisco Live!

General instructions to call a DNAC API

```
# Headers
headers = {'content-type': 'application/json','x-auth-token': token}

sites = get_sites(BASE_URL, SITE_URL, headers)
```

```
# Authentication
BASE_URL = 'https://dnac-79.infra.ciscomsx.com'
AUTH_URL = '/dna/system/api/v1/auth/token'
# URLs
SITE_URL = '/dna/intent/api/v1/site'
SITE_COUNT_URL = '/dna/intent/api/v1/site/count'
MEMBERSHIP_SITE_URL = '/dna/intent/api/v1/membership/{site_id}'
SITE_HEALTH_URL = '/dna/intent/api/v1/site-health'
ISSUES = url = '/dna/intent/api/v1/issues'
```

```
# Get list of sites
def get_sites(BASE_URL, URL, headers):
    response = requests.get(BASE_URL + URL,
                            headers=headers, verify=False)
    return response.json()['response']
```

Monitoring your network

Site

GET	Get Site Health *Intent*	Returns Overall Health information for all sites	/site-health
GET	Get Site *Intent*	Get site using siteNameHierarchy/siteId/type ,return all sites if these parameters are not given as input.	/site
GET	Get Membership *Intent*	Getting the site children details and device details.	/membership/${siteId}
GET	Issues *Intent*	Intent API to get a list of global issues, issues for a specific device, or issue for a specific client device's MAC address.	/issues

Monitoring your network

```python
# Headers
headers = {'content-type': 'application/json','x-auth-token': token}

sites = get_sites(BASE_URL, SITE_URL, headers)
```

```python
# Authentication
BASE_URL = 'https://dnac-79.infra.ciscomsx.com'
```

```python
# URLs
SITE_URL = '/dna/intent/api/v1/site'
SITE_COUNT_URL = '/dna/intent/api/v1/site/count'
MEMBERSHIP_SITE_URL = '/dna/intent/api/v1/membership/{site_id}'
SITE_HEALTH_URL = '/dna/intent/api/v1/site-health'
ISSUES_URL = url = '/dna/intent/api/v1/issues'
```

```python
# Print nicely only Name, Site Name Hierarchy and Namespace

import json

json_tree = json.dumps(sites,indent=4)

data = json.loads(json_tree)

for item in data:
    name = item.get('name')
    site_name_hierarchy = item.get('siteNameHierarchy')
    namespace = None
    item_id = item.get('id')

    additional_info = item.get('additionalInfo', [])
    for info in additional_info:
        namespace = info.get('nameSpace')
        print("Id:", item_id)
        print("Name:", name)
        print("Site Name Hierarchy:", site_name_hierarchy)
        print("Namespace:", namespace)
        print()
```

```
Id: 816225fd-9d87-406c-aadc-b8973519f02c
Name: Building-Regression
Site Name Hierarchy: Global/Area-Regression/Building-Regression
Namespace: UMBRELLA

Id: 816225fd-9d87-406c-aadc-b8973519f02c
Name: Building-Regression
Site Name Hierarchy: Global/Area-Regression/Building-Regression
Namespace: Location

Id: 816225fd-9d87-406c-aadc-b8973519f02c
Name: Building-Regression
Site Name Hierarchy: Global/Area-Regression/Building-Regression
Namespace: ETA
```

cisco Live!

Monitoring your network

```
ISSUES_URL = url = '/dna/intent/api/v1/issues'
issues_list = get_api(BASE_URL, ISSUES_URL, headers)
print ("\nPretty print response:\n",json.dumps(issues_list,indent=4))
```

```
Pretty print response:
 [
    {
        "issueId": "099bae52-119b-447e-bfc6-1d80120b8f35",
        "name": "Excessive time lag between Cisco DNA Center and device \"MSX-OTT02-CAT3650-01\"",
        "siteId": "",
        "deviceId": "0ed45671-ec6f-4b66-9ab8-a919d31008f5",
        "deviceRole": "",
        "aiDriven": "",
        "clientMac": "",
        "issue_occurence_count": 1,
        "status": "active",
        "priority": "",
        "category": "",
        "last_occurence_time": 1685796841000
    }
]
```

Issue

Monitoring your network

```python
# URLs
SITE_URL = '/dna/intent/api/v1/site'
SITE_COUNT_URL = '/dna/intent/api/v1/site/count'
MEMBERSHIP_SITE_URL = '/dna/intent/api/v1/membership/{site_id}'
SITE_HEALTH_URL = '/dna/intent/api/v1/site-health'
ISSUES_URL = url = '/dna/intent/api/v1/issues'
```

```python
site_health = get_site_health (BASE_URL, SITE_HEALTH_URL, headers)

print ("\nPretty print response:\n",json.dumps(site_health,indent=4))
```

```python
json_tree = json.dumps(site_health,indent=4)
data = json.loads(json_tree)

for site in data:
    site_name = site.get("siteName")
    site_id = site.get("siteId")
    healthy_network_device_percentage = site.get("healthyNetworkDevicePercentage")
    healthy_clients_percentage = site.get("healthyClientsPercentage")
    number_of_clients = site.get("numberOfClients")
    number_of_network_device = site.get("numberOfNetworkDevice")

    # Print the extracted information
    print("Site Name:", site_name)
    print("Site ID:", site_id)
    print("Healthy Network Device Percentage:", healthy_network_device_percentage)
    print("Healthy Clients Percentage:", healthy_clients_percentage)
    print("Number of Clients:", number_of_clients)
    print("Number of Network Devices:", number_of_network_device)
    print()
```

```
Site Name:  All Sites
Site ID:  All Sites
Healthy Network Device Percentage: 50
Healthy Clients Percentage: 100
Number of Clients: 1
Number of Network Devices: 2

Site Name: Amsterdam
Site ID: 9b592581-9388-4665-80d1-c2c30f1c4013
Healthy Network Device Percentage: None
Healthy Clients Percentage: None
Number of Clients: None
Number of Network Devices: None

Site Name: Area-Regression
Site ID: 50f4f3e5-0379-4ef2-9c59-27bd87366f47
Healthy Network Device Percentage: 50
Healthy Clients Percentage: 100
Number of Clients: 1
Number of Network Devices: 2
```

Monitoring your network

Topology

GET	Get VLAN details	Returns the list of VLAN names	/topology/vlan/vlan-names
GET	Get Physical Topology	Returns the raw physical topology by specified criteria of nodeType	/topology/physical-topology
GET	Get L3 Topology Details	Returns the Layer 3 network topology by routing protocol	/topology/l3/${topologyType}
GET	Get Site Topology	Returns site topology	/topology/site-topology
GET	Get topology details	Returns Layer 2 network topology by specified VLAN ID	/topology/l2/${vlanID}
GET	Get Overall Network Health *Intent*	Returns Overall Network Health information by Device category (Access, Distribution, Core, Router, Wireless) for any given point of time	/network-health

Monitoring your network

Topology

```
PHYSICAL_TOPOLOGY_URL = '/dna/intent/api/v1/topology/physical-topology'
physical_topology_list = get_api(BASE_URL, PHYSICAL_TOPOLOGY_URL, headers)
print ("\nPretty print response:\n",json.dumps(physical_topology_list,indent=4))
```

```
# Physical topology with device type, name, IP and Node Type
json_tree = json.dumps(physical_topology_list,indent=4)
data = json.loads(json_tree)

# Display deviceType, label, ip, and nodeType
for node in data['nodes']:
    print("Device Type:", node['deviceType'])
    print("Label:", node['label'])
    print("IP:", node['ip'])
    print("Node Type:", node['nodeType'])
    print()
```

Cisco Catalyst

```
Device Type: Cisco Catalyst 3650 Switch Stack
Label: MSX-OTT02-CAT3650-01
IP: 10.85.189.167
Node Type: device
```

```
Device Type: Third Party Device
Label: Street Left Camera
IP: 192.168.128.6
Node Type: device

Device Type: Third Party Device
Label: MR28 - Upper floor
IP: 192.168.1.208
Node Type: device
```

Meraki

```
Device Type: Cisco Catalyst 9800-CL Wireless Controller for Cloud
Label: MSX-OTT02-9800CL-06.cisco.com
IP: 10.85.189.78
Node Type: device
```

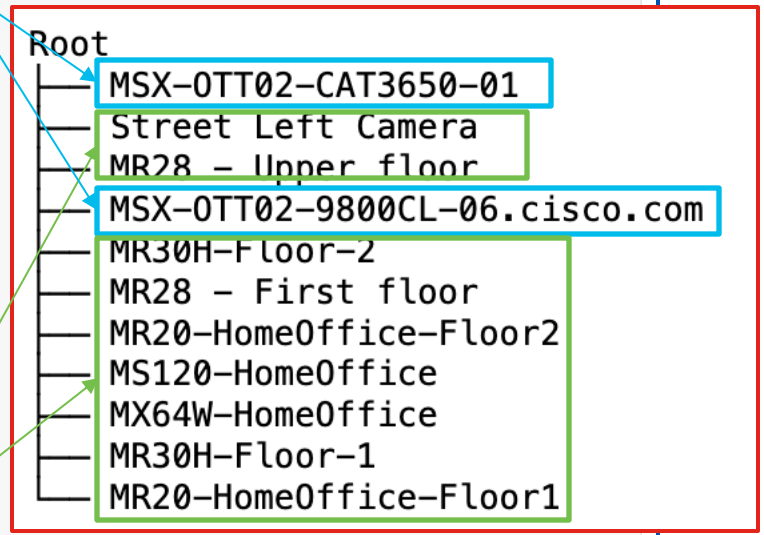
Monitoring your network

Topology beauty example

```python
from anytree import Node, RenderTree

# Parse JSON data
parsed_data = physical_topology_list

# Create a dictionary to store nodes by their IDs
nodes_dict = {}

# Create the root node
root = Node("Root")

# Create nodes and store them in the dictionary
for node_data in parsed_data['nodes']:
    node_id = node_data['id']
    label = node_data['label']
    parent_id = node_data['customParam']['parentNodeId']

    node = Node(label, parent=nodes_dict.get(parent_id, root))
    nodes_dict[node_id] = node

# Print the tree
for pre, fill, node in RenderTree(root):
    print(f"{pre}{node.name}")
```

Cisco Catalyst

Meraki



Monitoring your network

Issues

GET	Issues *Intent*	Intent API to get a list of global issues, issues for a specific device, or issue for a specific client device's MAC address.	/issues


```
{
   "issueId":"04f6c371-5b98-433b-98b2-375d67b6e95c",
   "name":"Excessive time lag between Cisco DNA Center and device \"MSX-OTT02-CAT3650-01\"",
   "siteId":"",
   "deviceId":"0ed45671-ec6f-4b66-9ab8-a919d31008f5",
   "deviceRole":"",
   "aiDriven":"",
   "clientMac":"",
   "issue_occurence_count":1,
   "status":"active",
   "priority":"",
   "category":"",
   "last_occurence_time":1682364608000
```

Monitoring your network

Issues

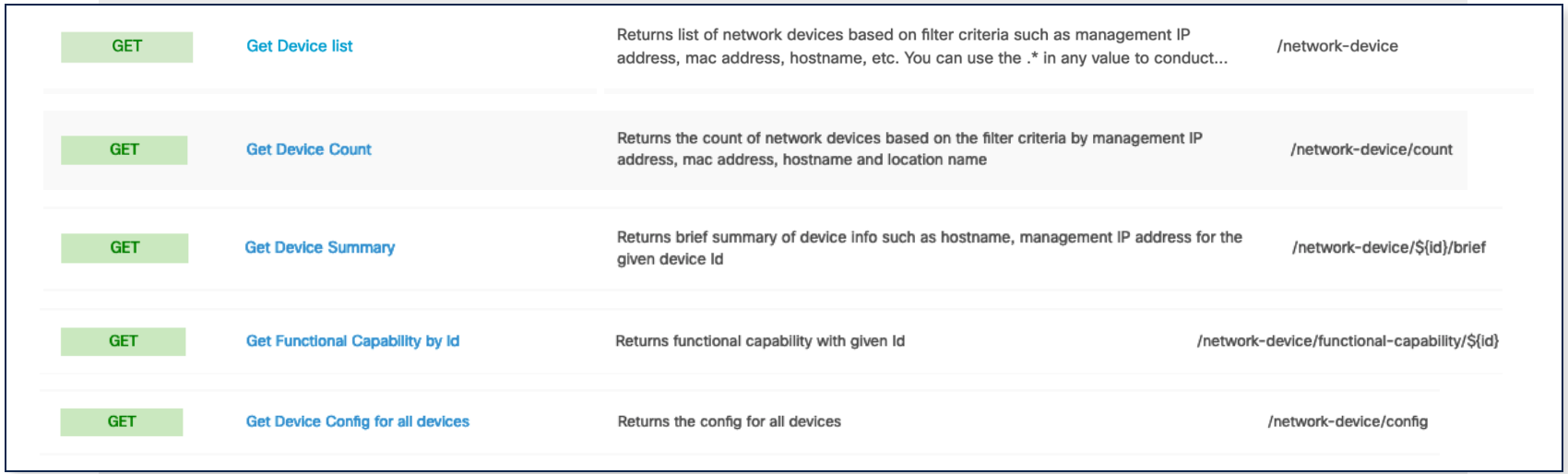
GET	Get Issue Enrichment Details *Intent*	Enriches a given network issue context (an issue id or end user's Mac Address) with details about the issue(s), impacted hosts and suggested actions for...	/issue-enrichment-details


```
"issueDetails":{
  "issue":[
    {
      "issueId":"04f6c371-5b98-433b-98b2-375d67b6e95c",
      "issueSource":"Cisco DNA",
      "issueCategory":"Device",
      "issueName":"device_time_drift",
      "issueDescription":"The time on Cisco DNA Center and Device \"MSX-OTT02-CAT3650-01\"
      "issueEntity":"network_device",
      "issueEntityValue":"0ed45671-ec6f-4b66-9ab8-a919d31008f5",
      "issueSeverity":"HIGH",
      "issuePriority":"P3",
      "issueSummary":"Excessive time lag between Cisco DNA Center and device \"MSX-OTT02-CA
      "issueTimestamp":1682364608000,
      "deviceId":"0ed45671-ec6f-4b66-9ab8-a919d31008f5",
```

```
"suggestedActions":[
  {
    "message":"Check time on the Device",
    "steps":[
      {
        "entityId":"0ed45671-ec6f-4b66-9ab8-a919d31008f5",
        "description":"Check system time",
        "command":"show clock",
        "stepType":"command-Runner",
        "runButton":null
      }
    ]
  },
```

Monitoring your network

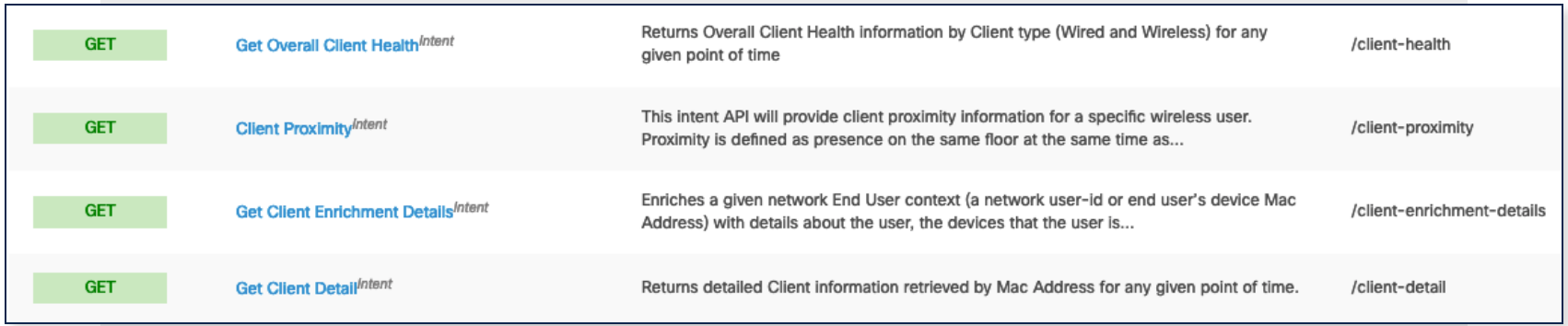
Device



GET	Get Device list	Returns list of network devices based on filter criteria such as management IP address, mac address, hostname, etc. You can use the .* in any value to conduct...	/network-device
GET	Get Device Count	Returns the count of network devices based on the filter criteria by management IP address, mac address, hostname and location name	/network-device/count
GET	Get Device Summary	Returns brief summary of device info such as hostname, management IP address for the given device Id	/network-device/${id}/brief
GET	Get Functional Capability by Id	Returns functional capability with given Id	/network-device/functional-capability/${id}
GET	Get Device Config for all devices	Returns the config for all devices	/network-device/config

Monitoring your network

Client



GET	Get Overall Client Health *Intent*	Returns Overall Client Health information by Client type (Wired and Wireless) for any given point of time	/client-health
GET	Client Proximity *Intent*	This intent API will provide client proximity information for a specific wireless user. Proximity is defined as presence on the same floor at the same time as...	/client-proximity
GET	Get Client Enrichment Details *Intent*	Enriches a given network End User context (a network user-id or end user's device Mac Address) with details about the user, the devices that the user is...	/client-enrichment-details
GET	Get Client Detail *Intent*	Returns detailed Client information retrieved by Mac Address for any given point of time.	/client-detail

Monitoring your network

Health and Performance

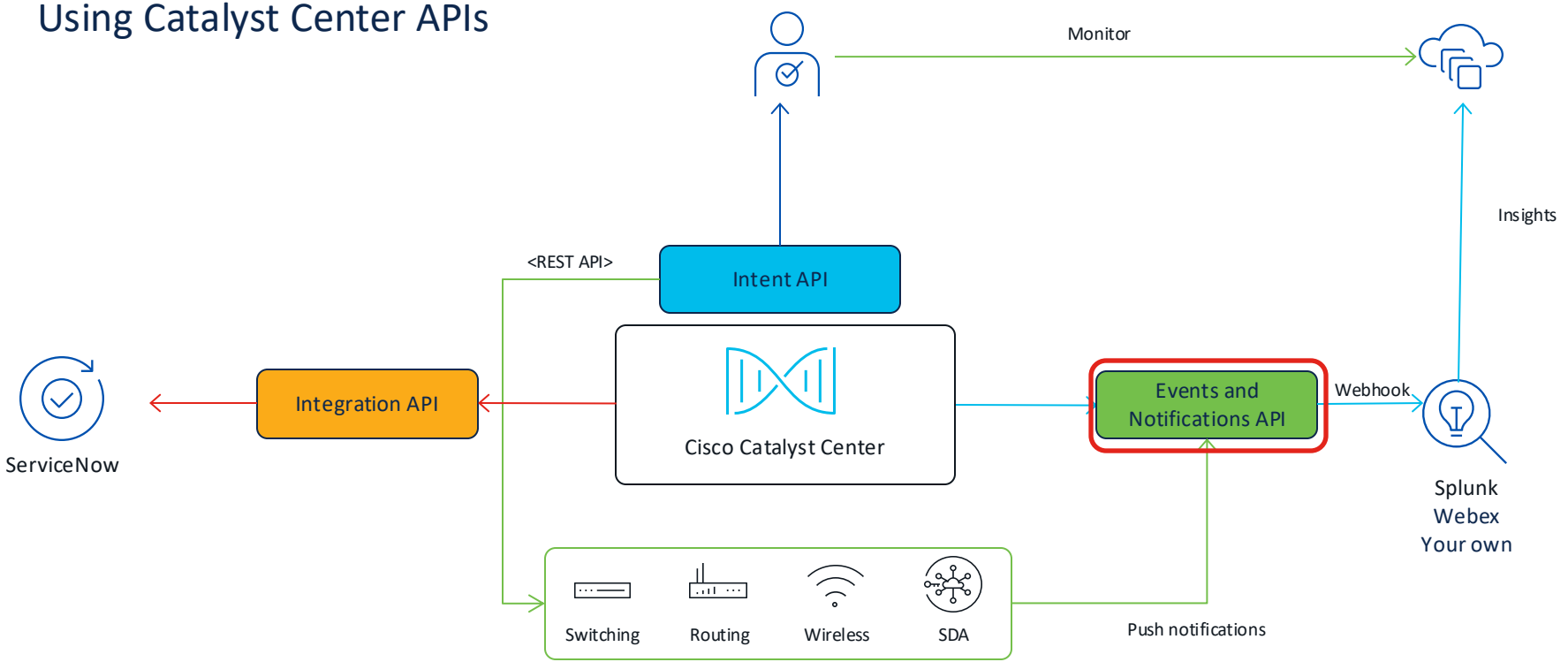
```python
# Get System Health
def get_system_health (BASE_URL, URL, headers):
    response = requests.get(BASE_URL + URL,
                            headers=headers, verify=False)
    print(response)
    return response.json()
```

Note: this response does not have the 'response' key in the API response

GET	System Health Count API *Intent*	This API gives the count of the latest system events	/diagnostics/system/health/count
GET	System Health API *Intent*	This API retrieves the latest system events	/diagnostics/system/health
GET	System Performance Historical API *Intent*	This API retrieves the historical performance indicators . The data can be retrieved for the last 3 months.	/diagnostics/system/performance/history
GET	System Performance API *Intent*	This API gives the aggregated performance indicators. The data can be retrieved for the last 3 months.	/diagnostics/system/performance

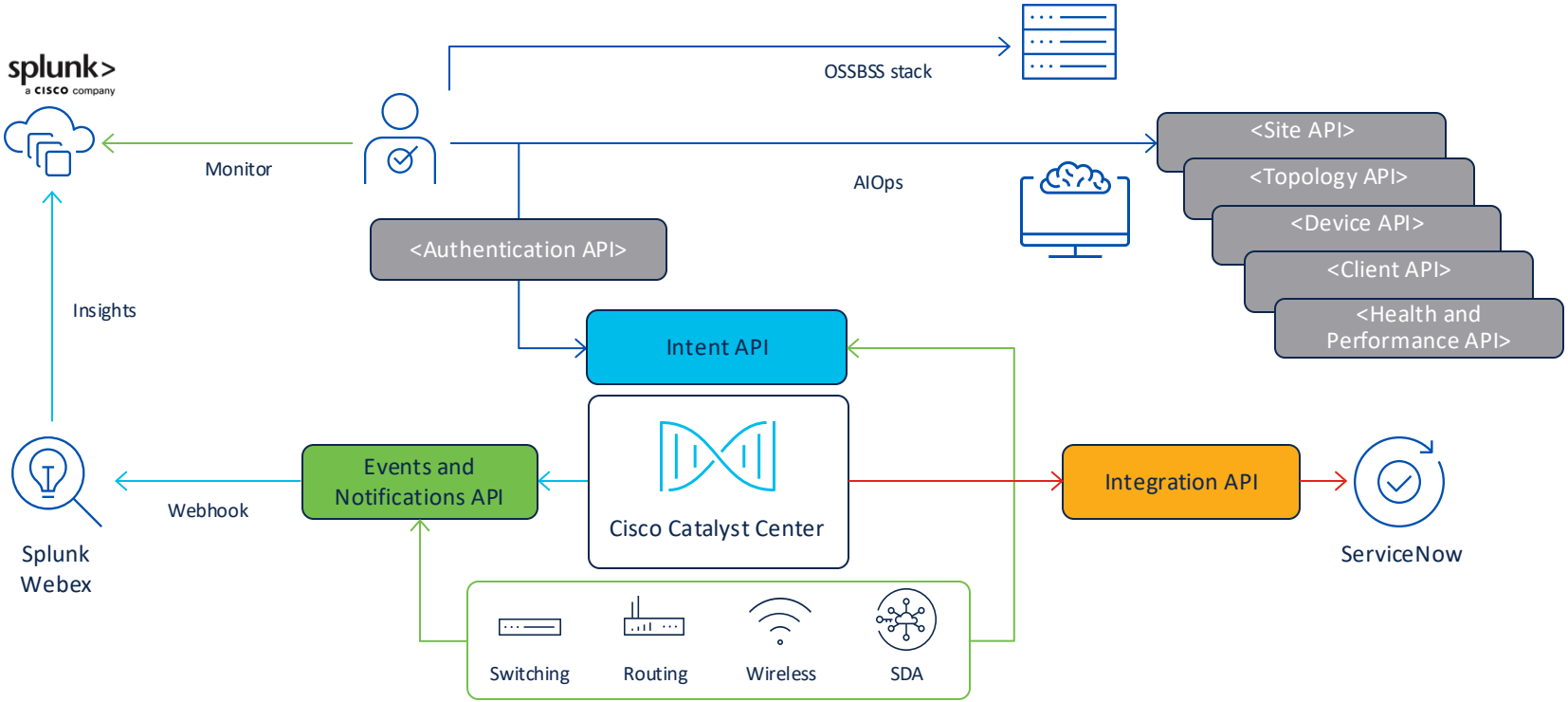
Headless operations

Using Catalyst Center APIs



Catalyst Center headless operations

Recap



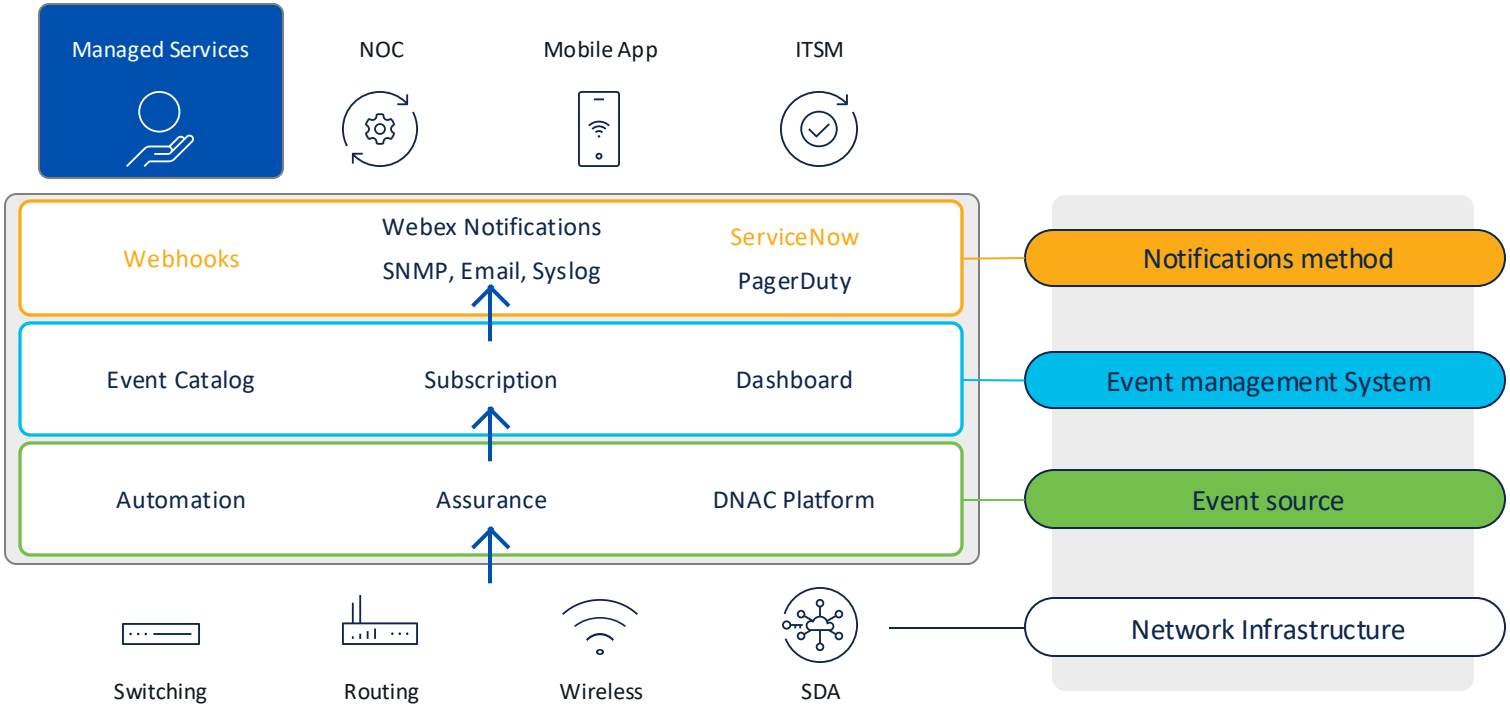
CatC APIs 400 level – Event Management API

cisco Live!

This lab will use python

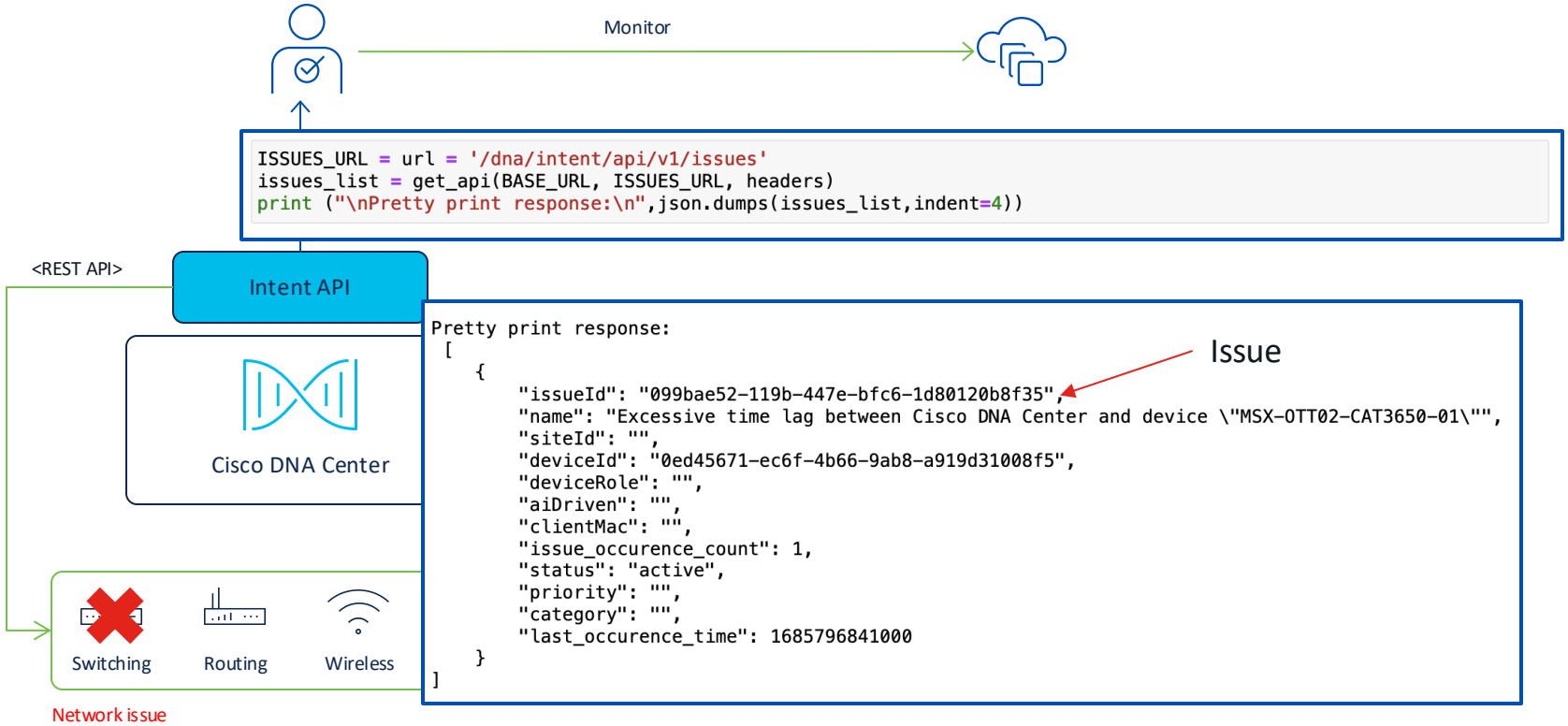
- You can do this in your laptop
- You already have installed Python, you might need additional packages.
- Directory for this lab:
  - https://github.com/hemorale/CatC-API-Workshop/blob/main/Intent%20and%20Event%20Management/

Catalyst Center Notifications Framework



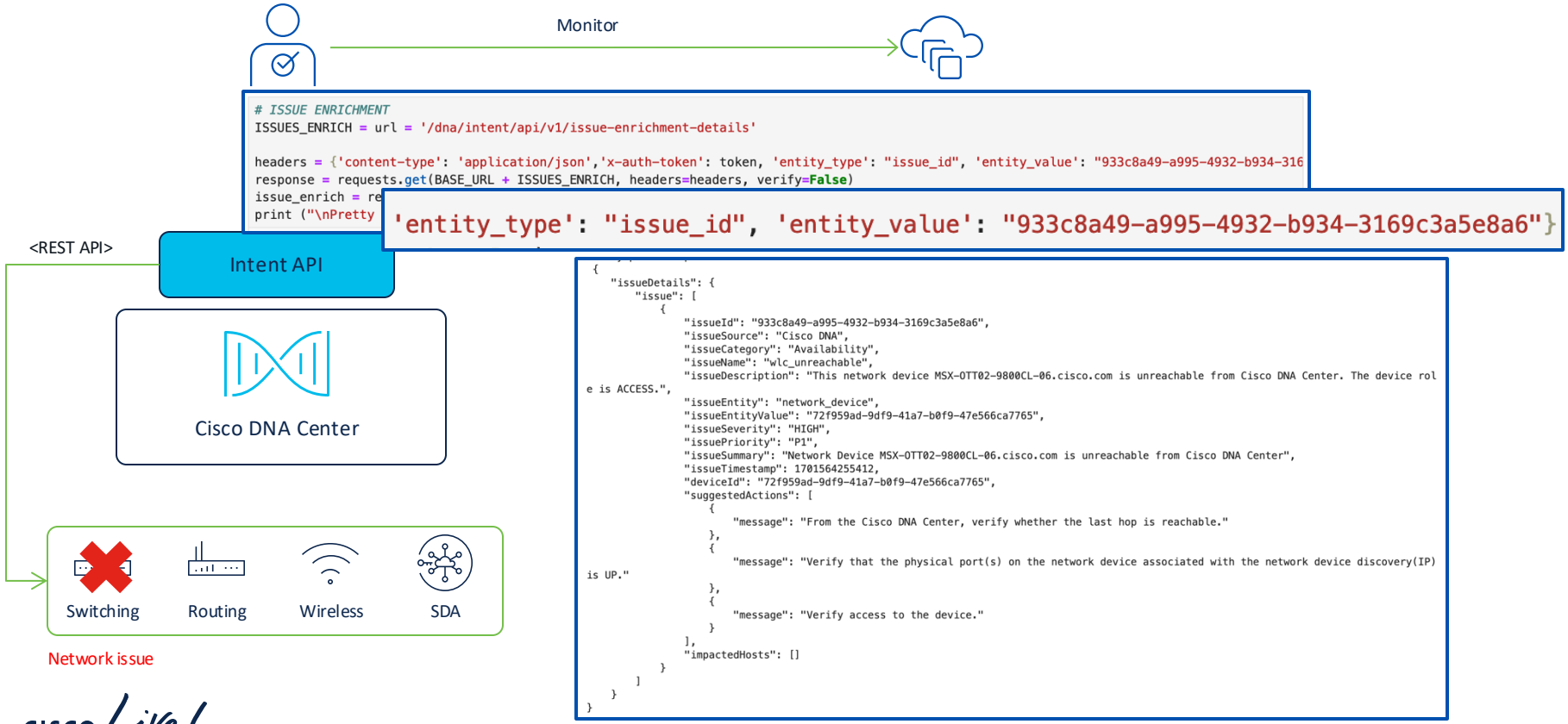
Managing your campus

Get issue using Intent API



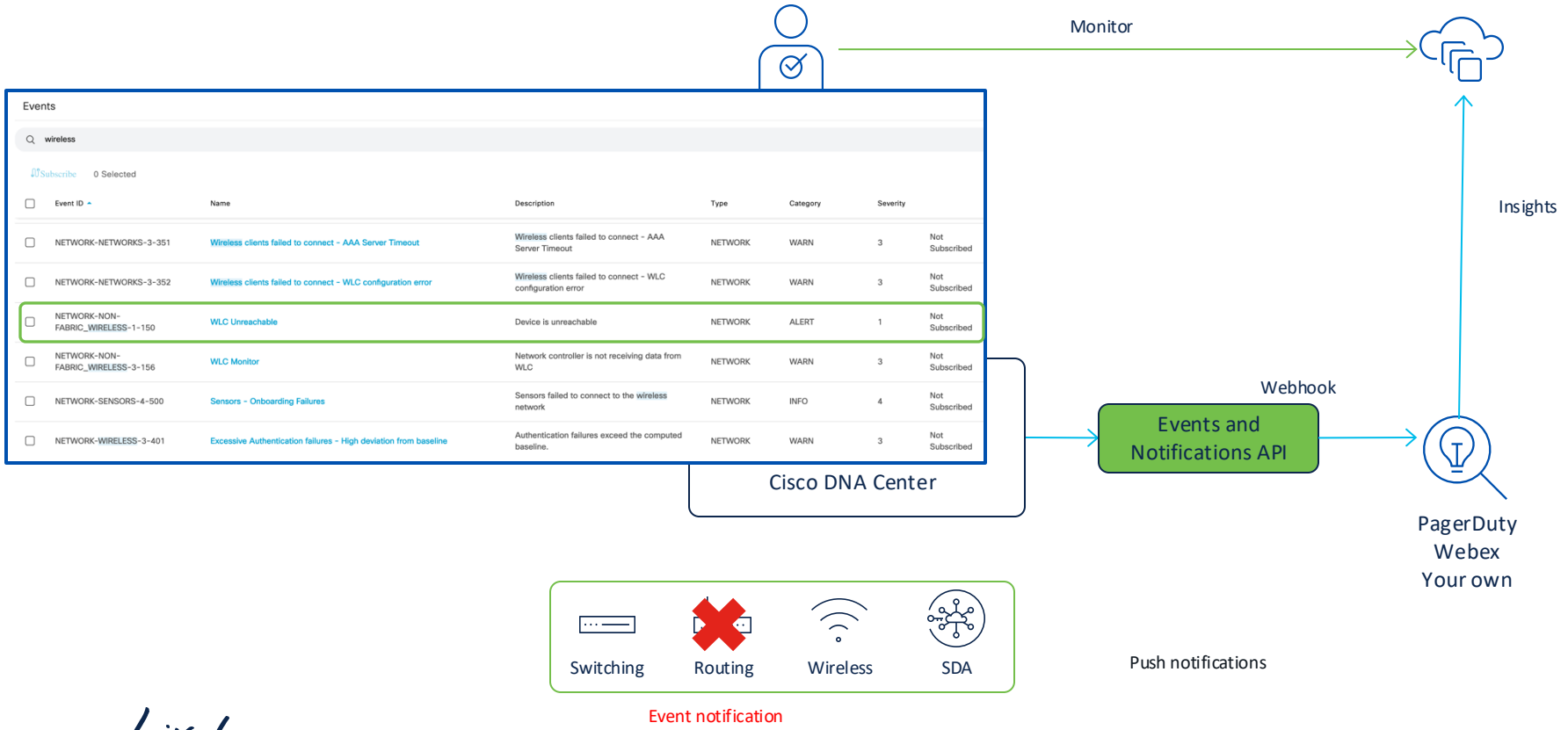
Managing your campus

Issue enrichment using Intent API



Managing your campus

Event notification

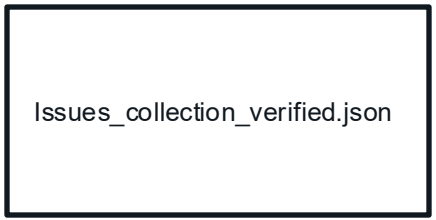


Lab prerequisites

1. Git clone the collection
   https://github.com/hemorale/CatC-API-Workshop/tree/main/Intent%20and%20Event%20Management
2. Pip install all requirements
3. Open the jupyter notebook

Lab 1

Find the common issue between these collections



‘issueId’

‘issueId’ in issueDetails{}, issue[]

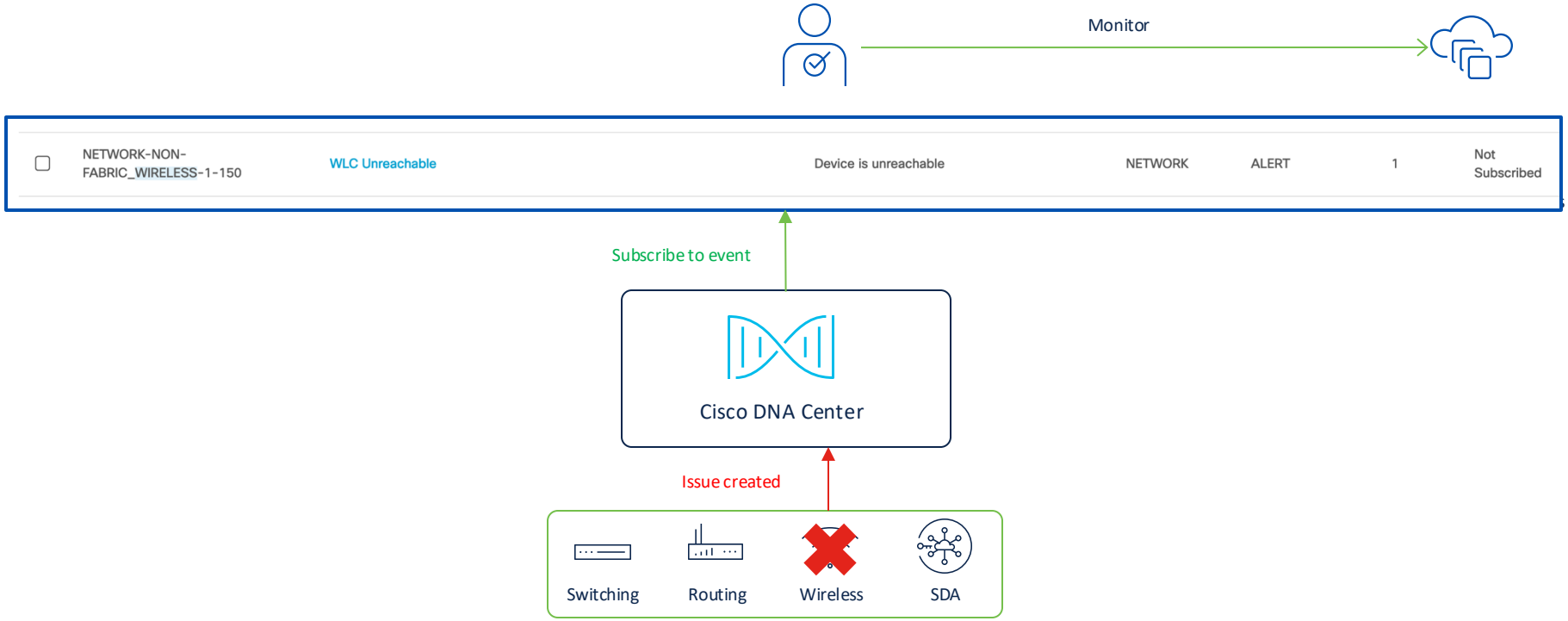
‘instanceId’

1. Open jupyter notebook
2. Reusable python script: findissues.py

Webhook notifications workflow

Better using event notifications

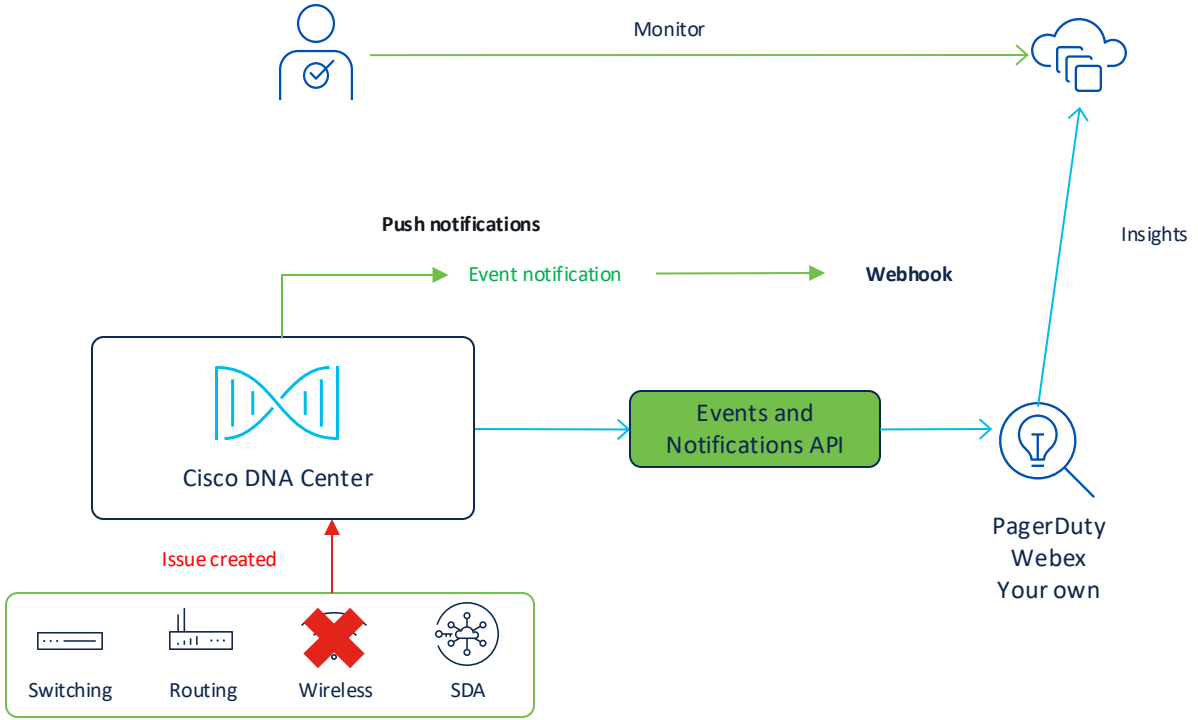
Event subscription



Better using event notifications

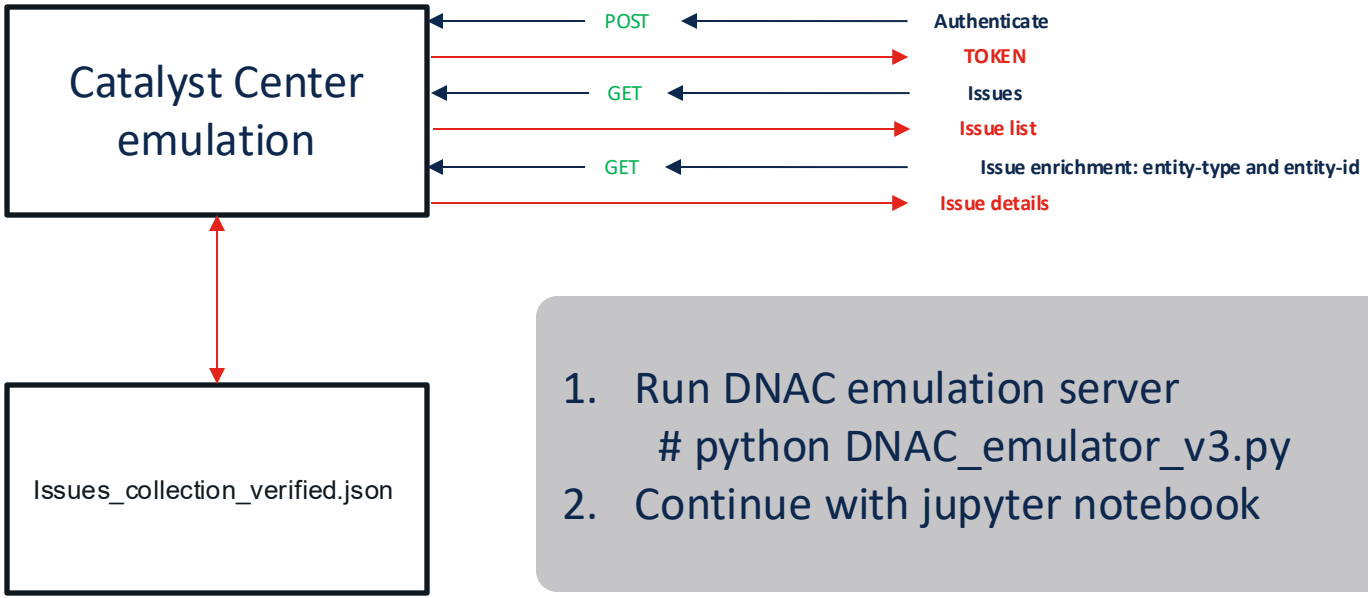
Event subscription

Once an issue is triggered by the assurance engine and if the event is subscribed, an event is sent out to the remote webhook location



Lab 2

Emulating Catalyst Center to get issues and issue enrichment

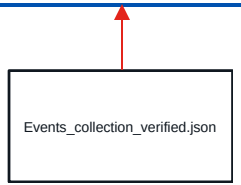


1. Run DNAC emulation server
   # python DNAC_emulator_v3.py
2. Continue with jupyter notebook

The webhook server

Emulating events and post them

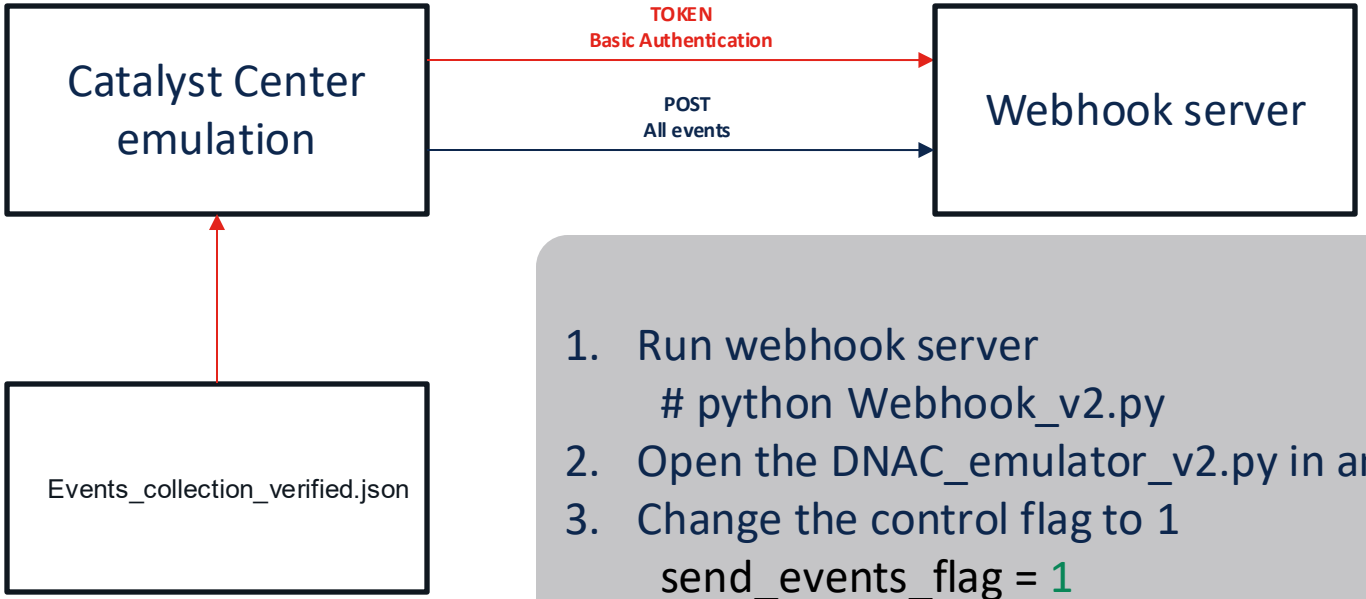
```python
def send_periodic_events():
    global send_events_flag, post_interval
    while True:
        if send_events_flag:
            try:
                with open('events_collection_verified.json', 'r') as file:
                    events_data = json.load(file)

                # Encode the credentials
                credentials = f"{VALID_USERNAME}:{VALID_PASSWORD}"
                encoded_credentials = base64.b64encode(credentials.encode()).decode()

                # Set up headers with Basic Authentication
                headers = {
                    'Authorization': f'Basic {encoded_credentials}',
                    'content-type': 'application/json'
                }

                # Sending POST request
                post_url = 'http://127.0.0.1:5001/dna/events/'  # Update to your target server's URL
                response = requests.post(post_url, json=events_data, headers=headers, verify=True)
                print("POST request sent. Status Code:", response.status_code)
            except Exception as e:
                print("Error sending POST request:", str(e))
        time.sleep(post_interval)
```




```python
# Define valid credentials for Basic Authentication
users = {
    "admin": "CiscoLive100%"
}

@auth.verify_password
def verify_password(username, password):
    if username in users and users[username] == password:
        return username

@app.route('/dna/events/', methods=['POST'])
@auth.login_required
def receive_events():
    data = request.json
    print("Received POST request with data:", json.dumps(data, indent=4))
    return jsonify({"message": "Data received successfully"}), 200
```

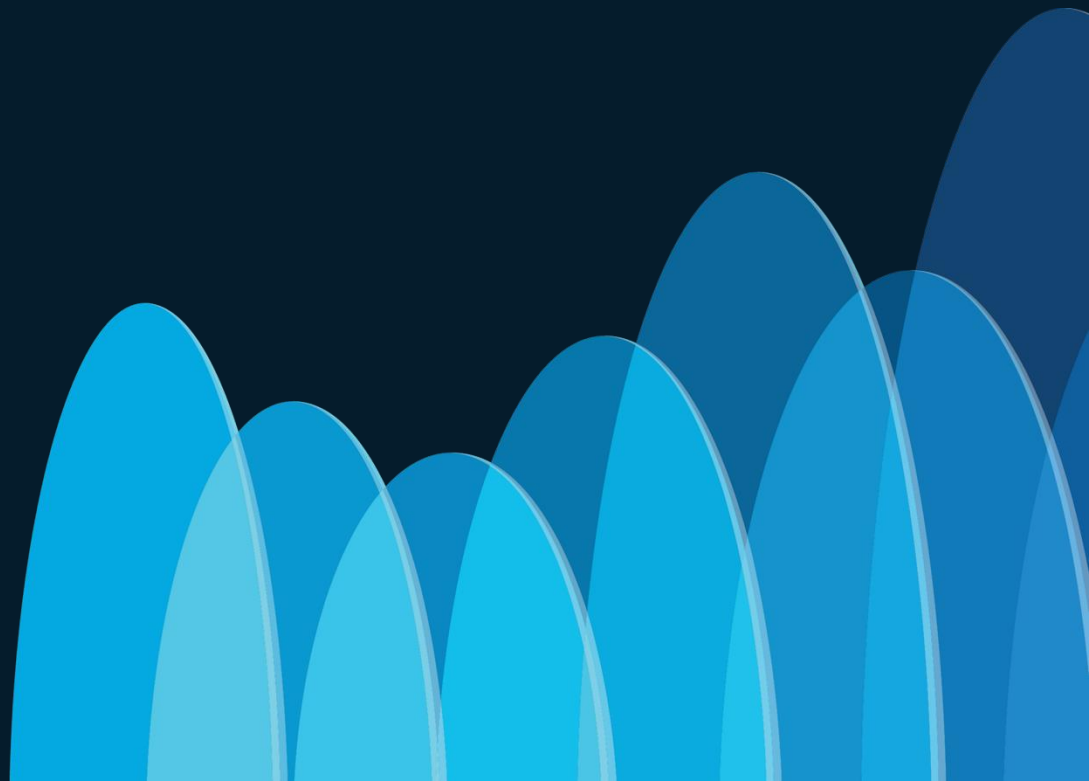
Lab 3

Creating a webhook server



1. Run webhook server
   # python Webhook_v2.py
2. Open the DNAC_emulator_v2.py in an editor
3. Change the control flag to 1
   send_events_flag = 1

Setting up a webhook destination on Catalyst Center



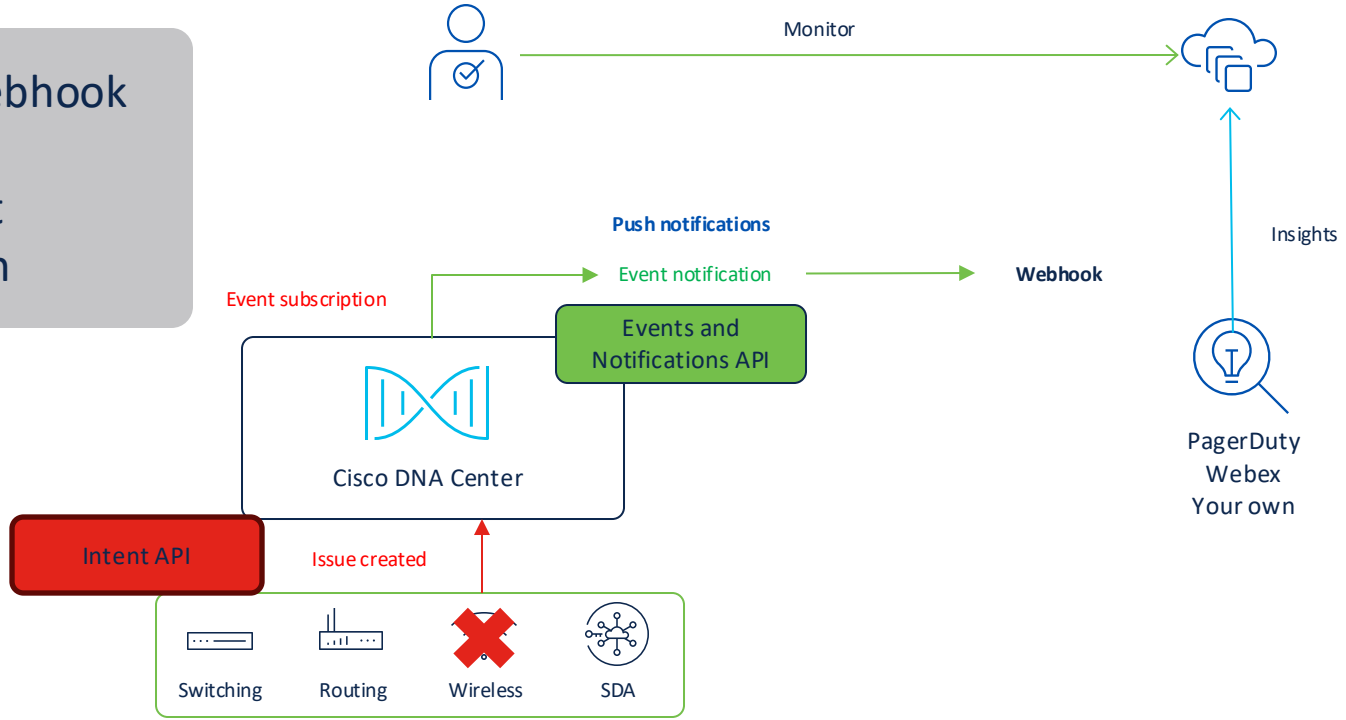
cisco Live!

This lab will use python

- You can do this in your laptop
- You will need Ngrok and the Webhook_server.py script
- Detailed Ngrok installation instructions are in the collection
- Directory for this lab:
  - https://github.com/hemorale/CatC-API-Workshop/blob/main/Intent%20and%20Event%20Management/

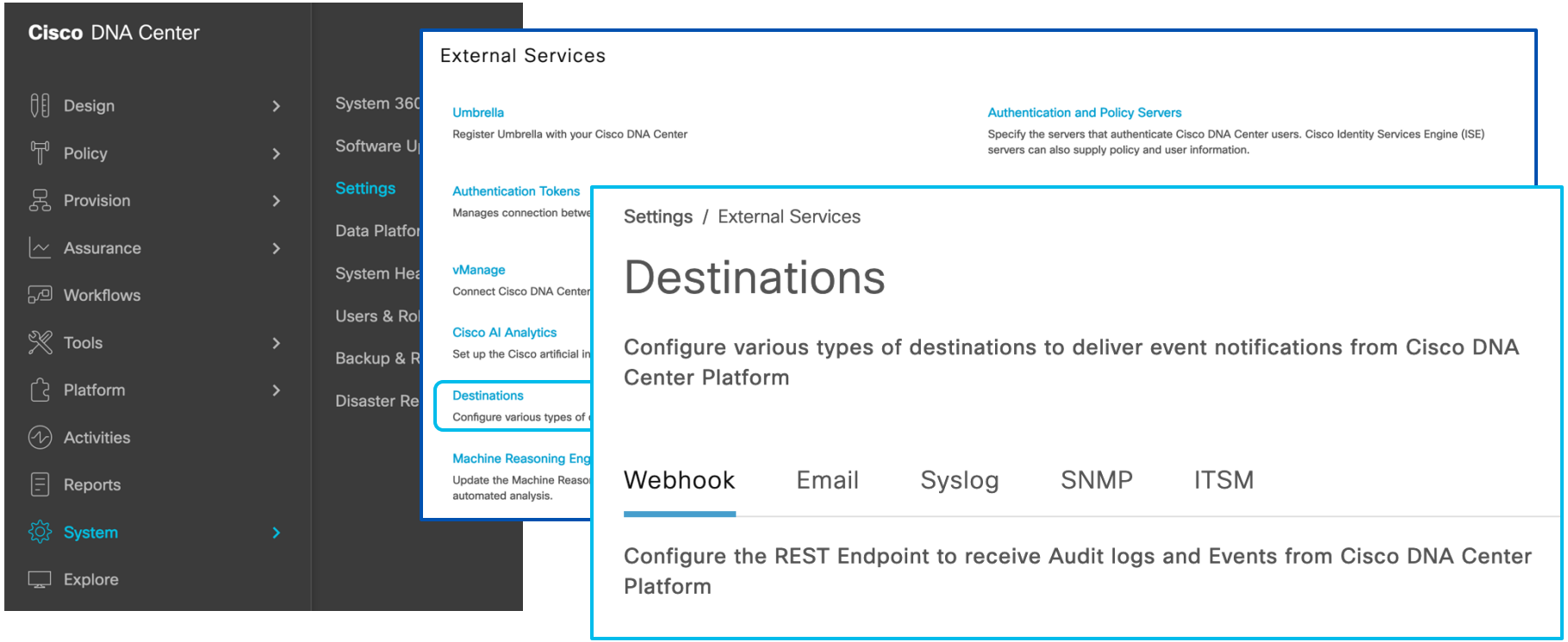
Event notifications flow

1. Setup a new webhook destination
2. Subscribe event
3. Test notification

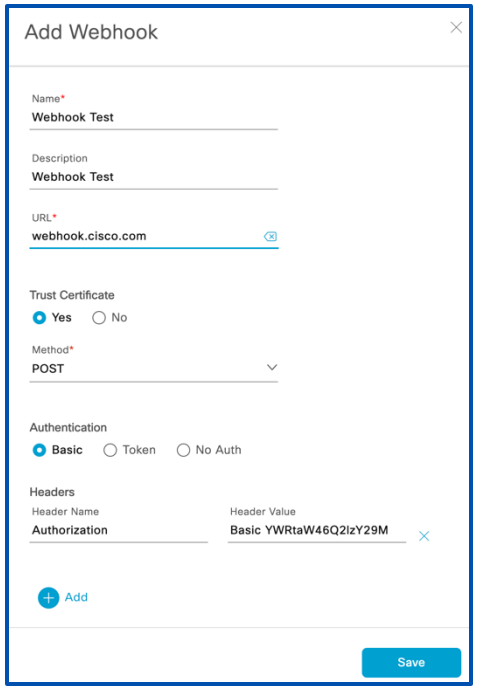


Setting up a destination

Webhook



Webhook provisioning



Trust Certificate: **YES**
Method: **POST**
Authentication: **Basic**
Headers:

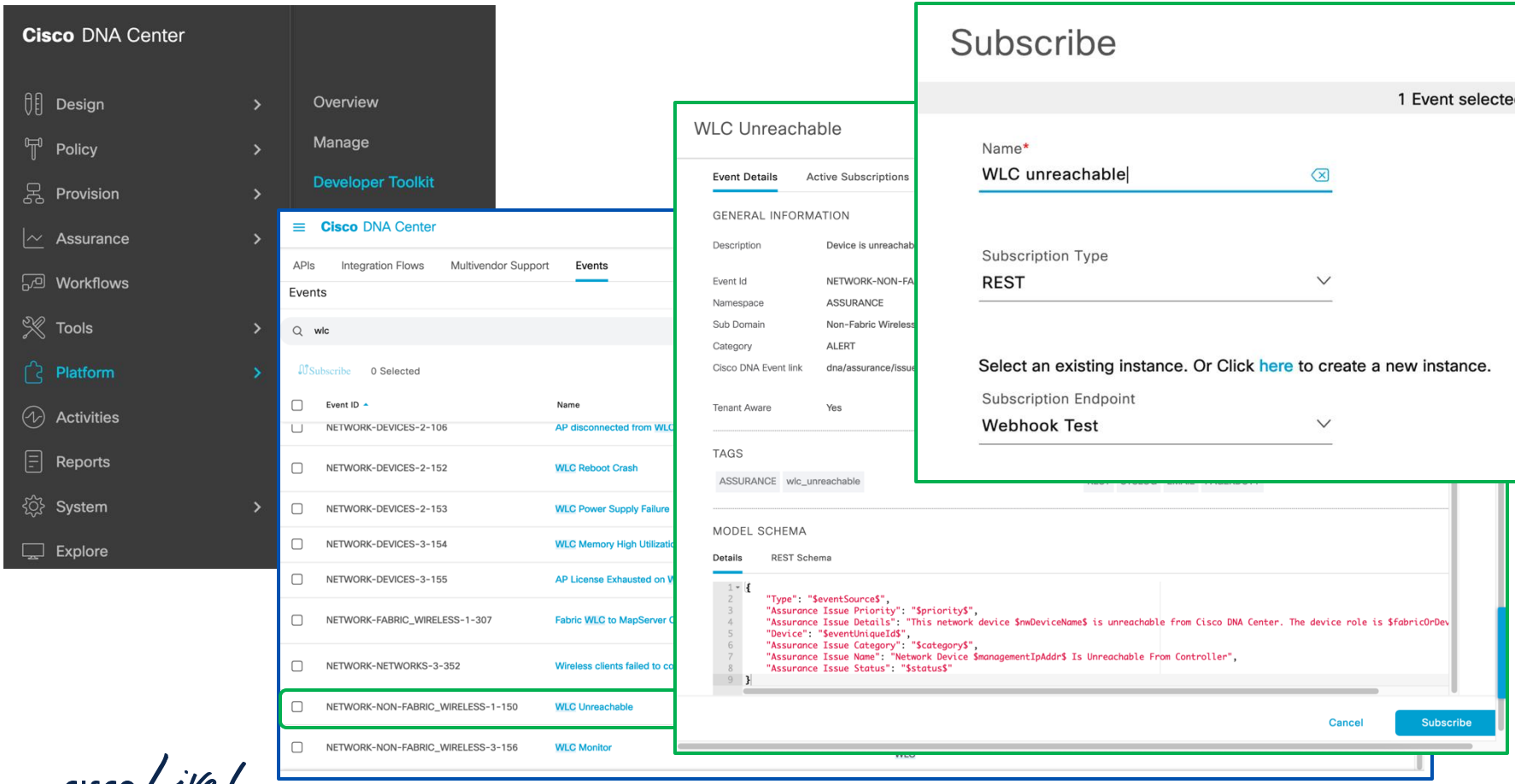
'Authorization': 'Basic Encoded64(Username+Password)'

```python
import base64

# Define valid credentials
VALID_USERNAME = 'admin'
VALID_PASSWORD = 'CiscoLive100%'

# Encode the credentials
credentials = f"{VALID_USERNAME}:{VALID_PASSWORD}"
encoded_credentials = base64.b64encode(credentials.encode()).decode()

print(encoded_credentials)
print(f'Basic {encoded_credentials}')
```

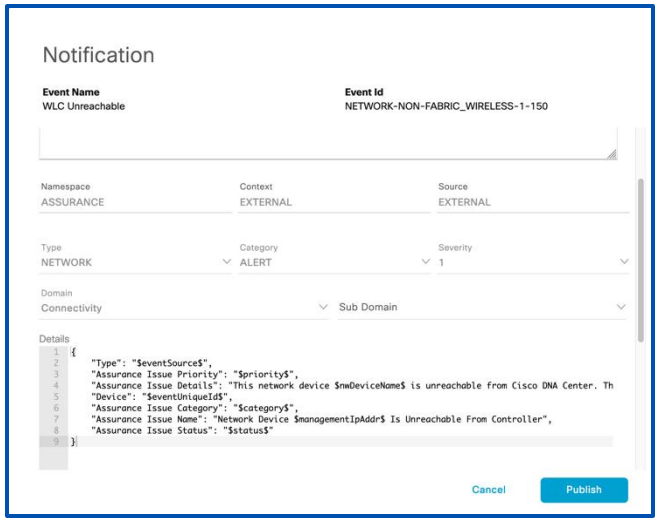
```
YWRtaW46Q2lzY29MaXZlMTAwJQ==
Basic YWRtaW46Q2lzY29MaXZlMTAwJQ==
```

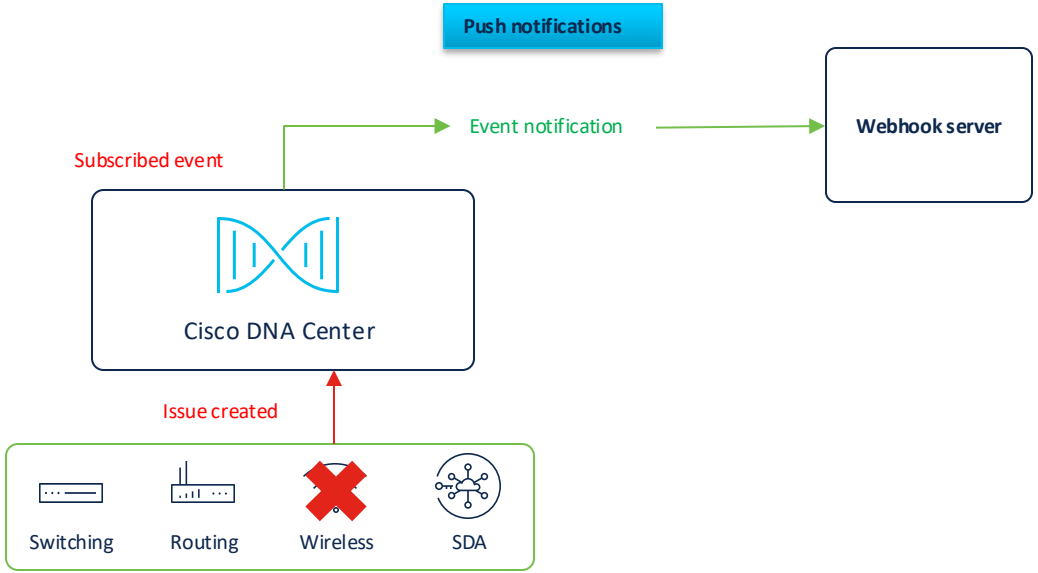
Name	Description	URL	Method
Webhook Test	Webhook Test	https://webhook.cisco.com	POST

Subscribe to events

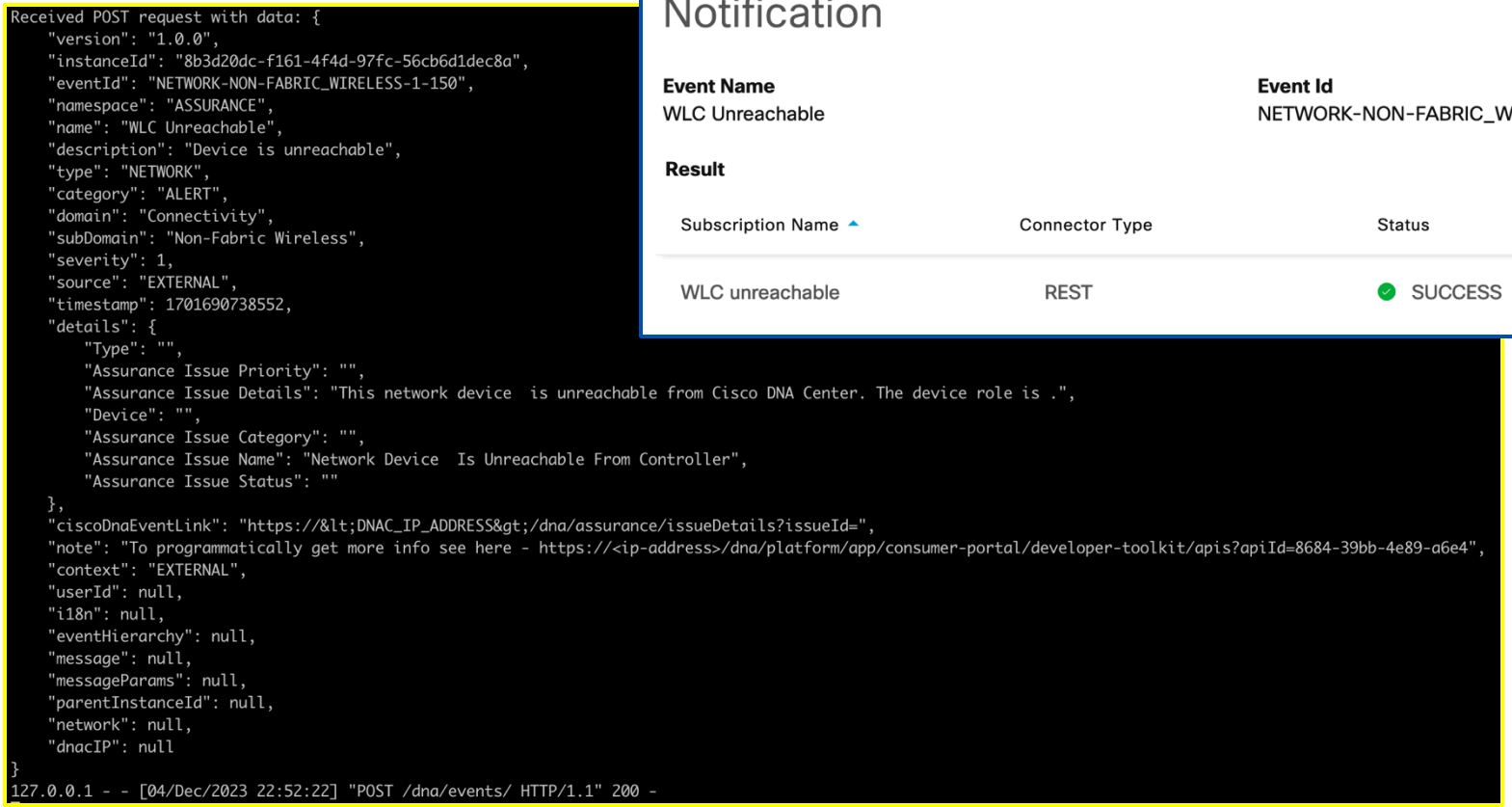


CC Webhooks notifications workflow



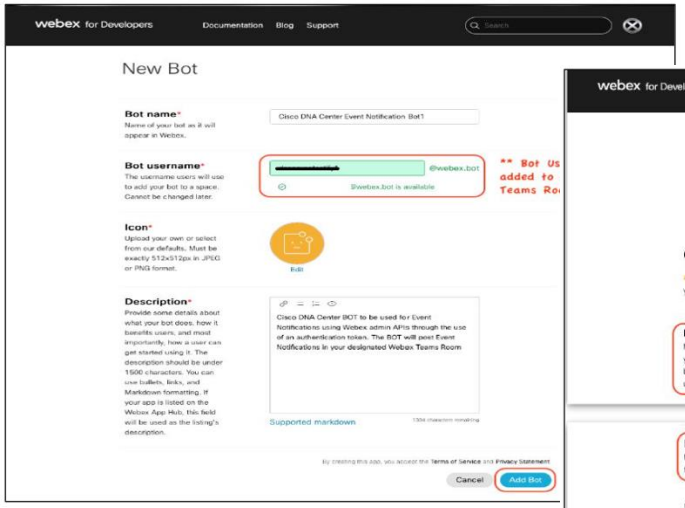


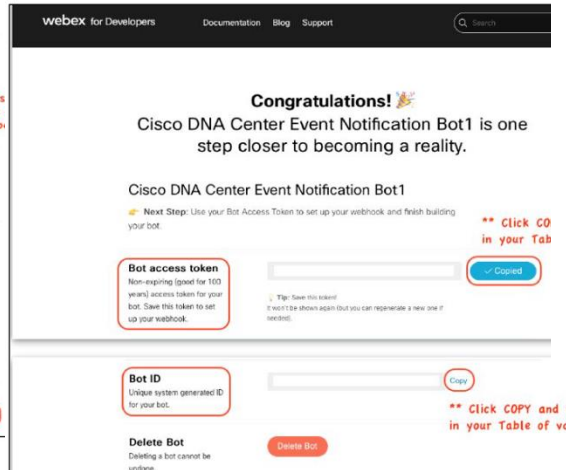
Try on your own

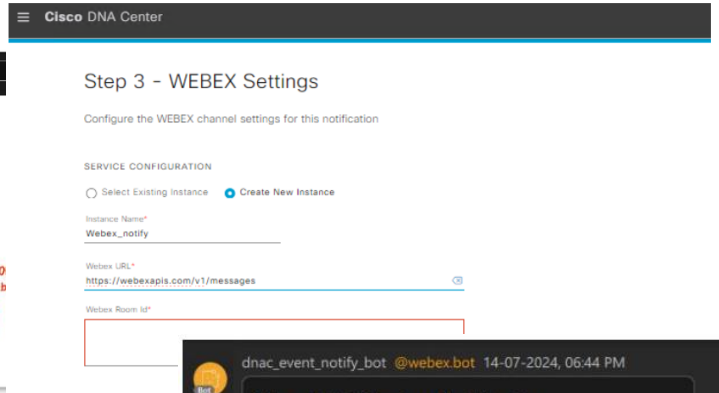


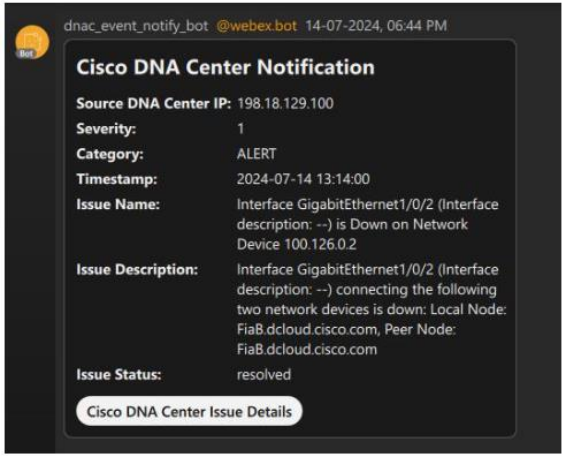
Best Practices

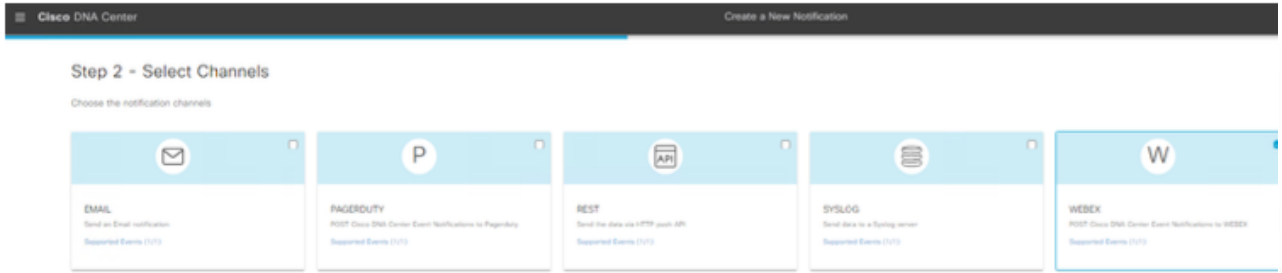
Webex Bot as webhook destination











More information on salesconnect (access for partners)

Automated bulk upgrade of network devices

Vulnerability assessment for devices managed by multiple Catalyst-Center

The objective of this use case is to streamline and provision the process of upgrading network devices managed by multiple Catalyst Centers driven by vulnerability assessments to ensure that all network devices are up-to-date with the recommended security patches and firmware versions as suggested by Catalyst Center.

1 Perform Security Advisory Retrieval

```
post_url = "https://"+ip+"/api/system/v1/auth/token"
url_sites = f"https://{ip_address}/dna/intent/api/v1/site"
url_membership = f"https://{ip_address}/dna/intent/api/v1/membership/{site_id}"
url_advisory = f'https://{ip_address}/dna/intent/api/v1/security-advisory/device/{key}/advisory'
```

2 Perform Software Image Management on the devices on network

```
url_images = f'https://{ip}/dna/intent/api/v1/image/importation'
url_distribute = f'https://{ip}/dna/intent/api/v1/image/distribution'
url_device = f'https://{ip}/dna/intent/api/v1/network-device'
url = f'https://{host}/api/v1/task/{taskid}'
url_activate = f'https://{ip}/dna/intent/api/v1/image/activation/device'
```

More information on salesconnect (access for partners)

Pushing customized template

Devices spread across multiple sites (topology, inventory, template)

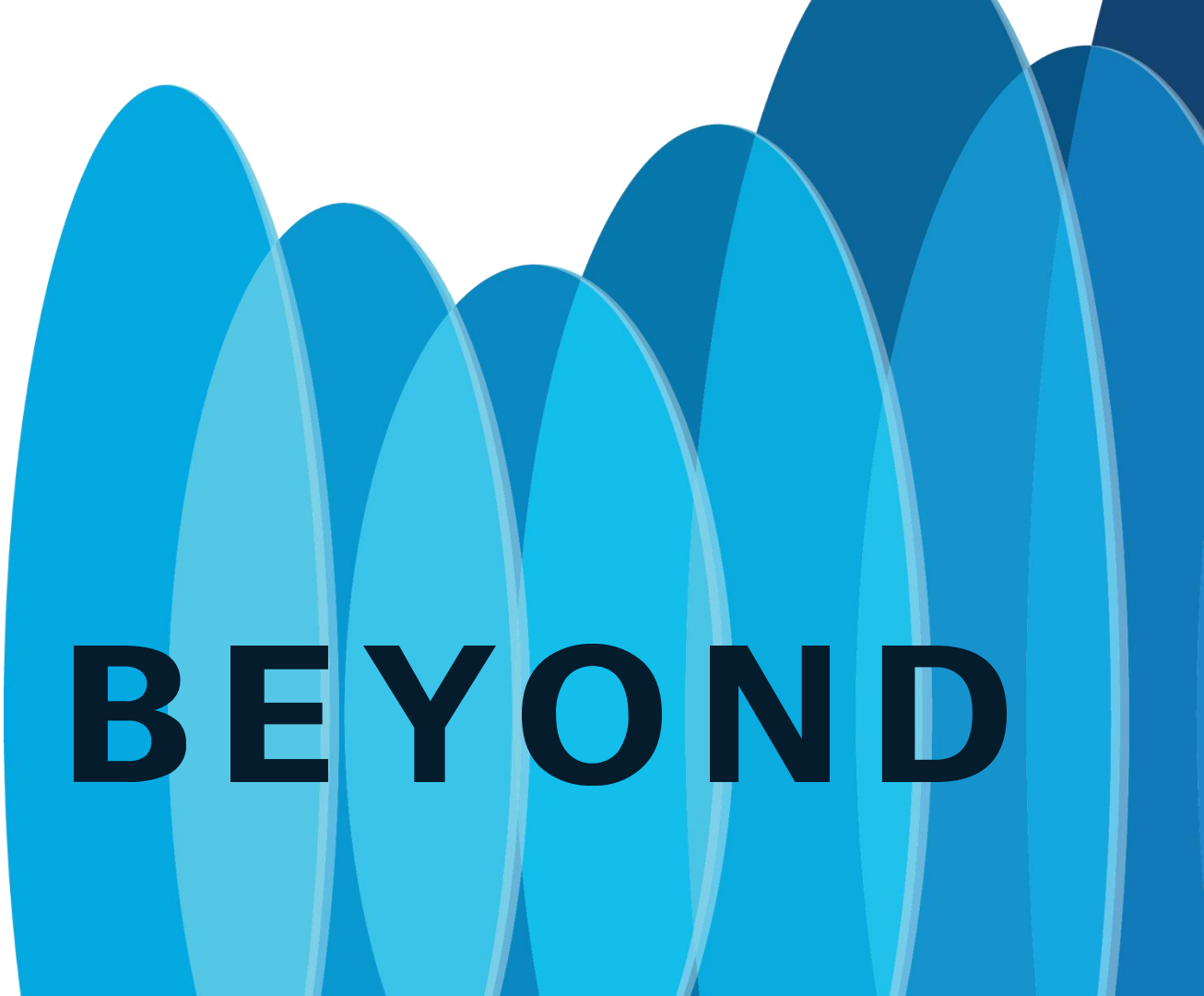
To automate the deployment of customised configuration templates to network devices distributed across multiple sites by leveraging APIs to manage topology, inventory, and template application, thereby ensuring consistent and efficient configuration management, and reducing manual intervention.


```
post_url = "https://"+ip+"/api/system/v1/auth/token"
url_device = f'https://{ip}/dna/intent/api/v1/network-device'
url_show_template = f'https://{ip}/dna/intent/api/v1/template-programmer/template'
url_search_template = f'https://{ip}/dna/intent/api/v1/template-programmer/template'
url_template = f'https://{ip}/dna/intent/api/v1/template-programmer/template/{templateId}'
url_template_apply = f'https://{ip}/dna/intent/api/v2/template-programmer/template/deploy'
url = f'https://{host}/api/v1/task/{taskid}'
url = f'https://{host}/dna/intent/api/v1/template-programmer/template/deploy/status/{deploymentId}'
```

More information on salesconnect (access for partners)


cisco
Thank you
cisco Live!

cisco Live!



GO BEYOND