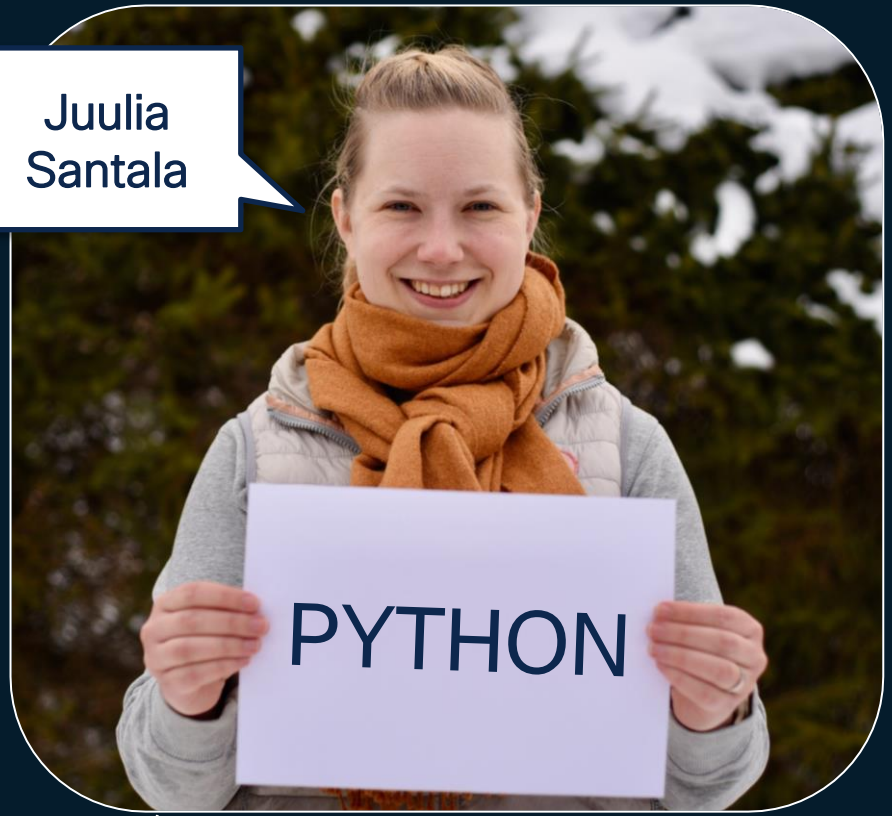
The Network of the Future is Here

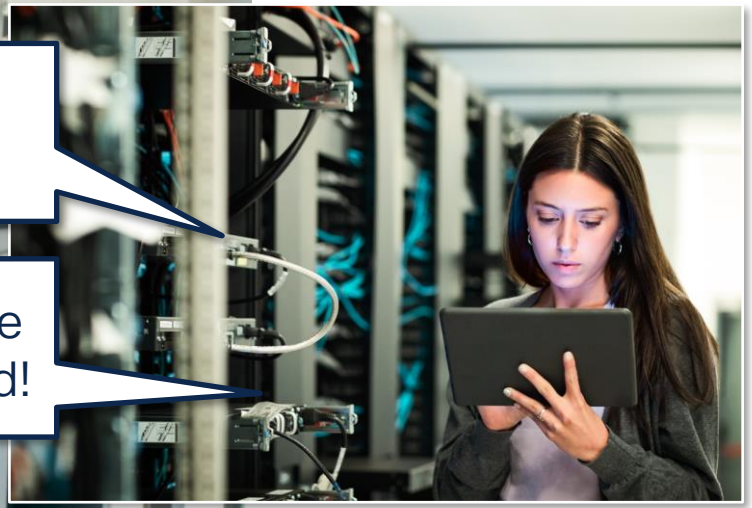
Let's Automate your IPv6 deployment with Python!

Juulia Santala, SE Lead for network programmability
Nicole Wajer, SE Lead for IPv6

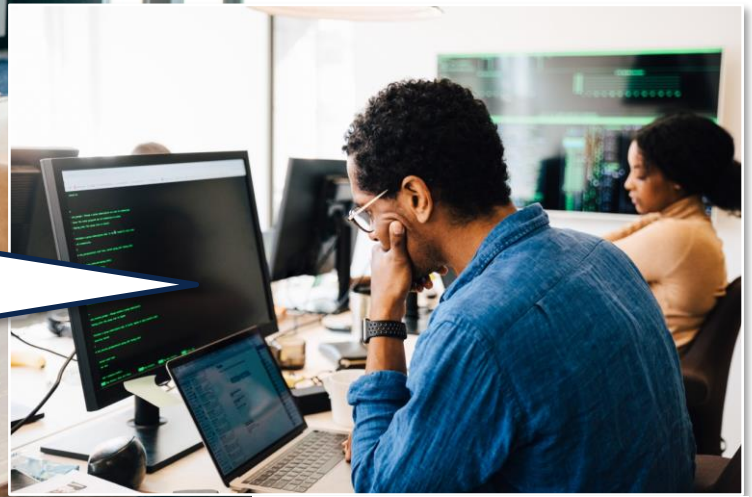
BRKOPS-2223



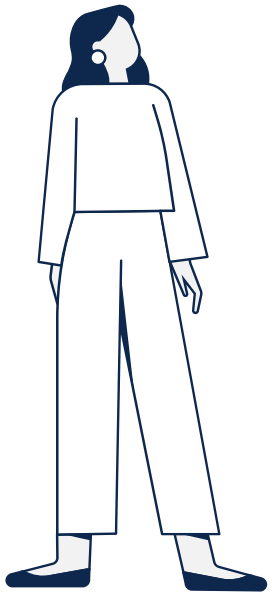
Can we get more IPv4?



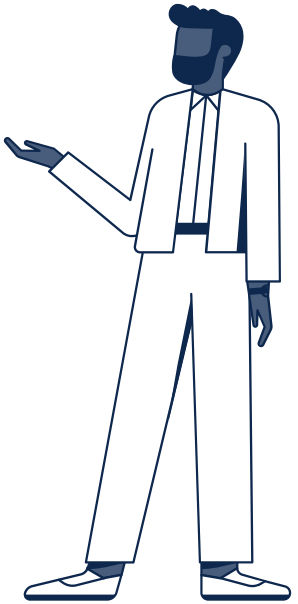
All our endpoints are already IPv6 enabled!



In addition to all these other tickets, now I need to speed up service deployment?



Let's deploy IPv6 with the help of your network's APIs!



Agenda

1. Design your IPv6 network
2. Deploy using your network's APIs
3. Validate and troubleshoot your deployment
4. Scale and confirm with Network as Code

What to expect from this session

IOS XE, Meraki, and Catalyst Center focus

... applicable for other solutions too!

Real Python code examples

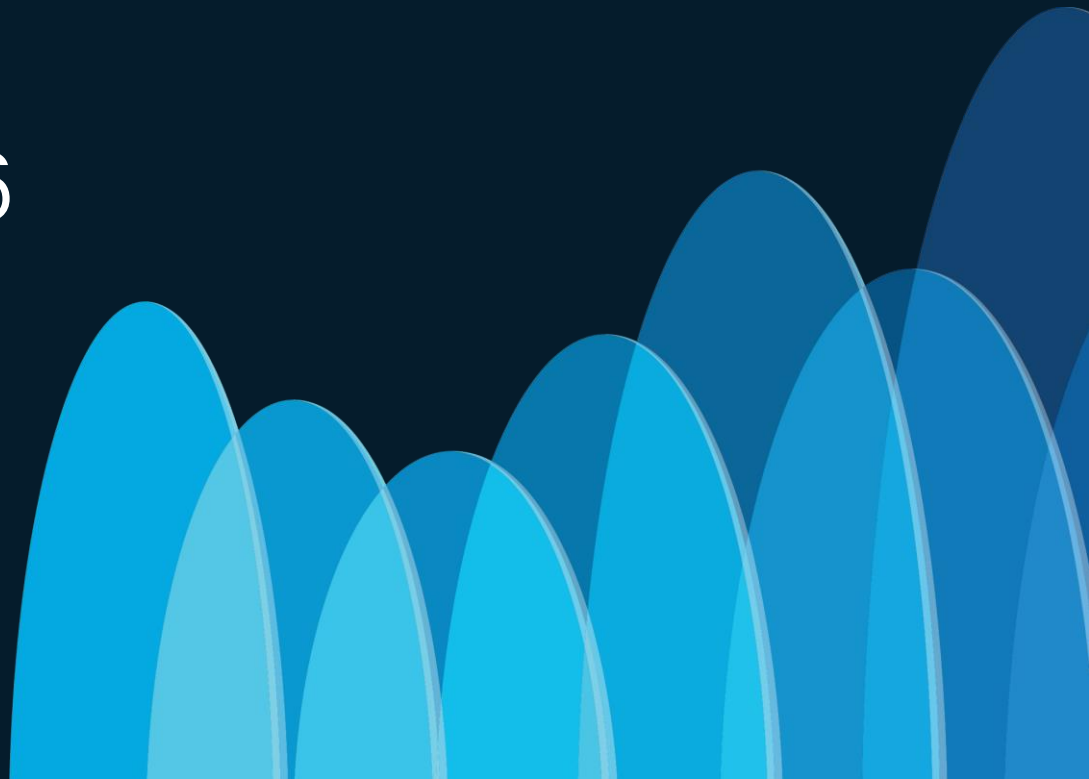
... outcomes can be reached also with other languages

Practical focus on a specific use case

Check resources at the end of the deck for further learning

1. Design | 2. Deploy | 3. Validate | 4. Scale

Design your IPv6 network

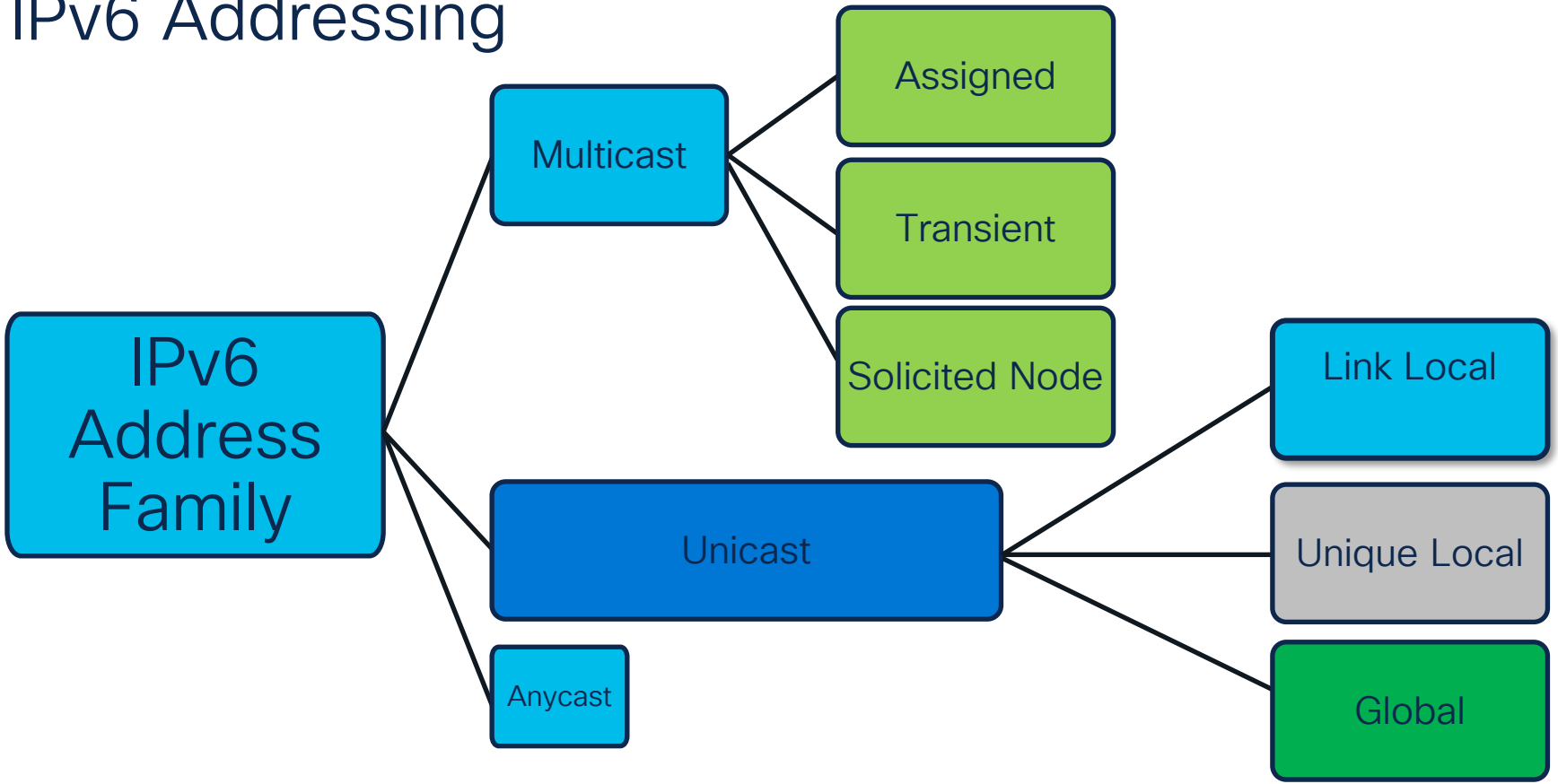


cisco Live!

The Address

2001:0db8:4646:0001:0000:0000:0000:1e2a

IPv6 Addressing



IPv6 Addresses Format

- Full IPv6 address (128 bits represented by 32 Hex characters)
  - Generally, 50% for network ID, 50% for interface ID, current best practice
  - Meaning a Subnet Prefix is generally a /64
- Leading 0's can be omitted
- The double colon (::) can appear only once

Global Prefix (48) Subnet (16) Interface Id (64)

3fff:0db8:4646:0001:0000:0000:0000:1e2a

3fff:db8:4646:1:0:0:0:1e2a

3fff:db8:4646:1::1e2a

Link-Local address

Non routable exists on single layer 2 domain (fe80::/10)

fe80:0000:0000:0000:xxxx:xxxx:xxxx:xxxx

Unique Local Address

Routable within administrative domain (fc00::/7)

fc00:~~~~:~~~~:****:xxxx:xxxx:xxxx:xxxx

fd00:~~~~:~~~~:****:xxxx:xxxx:xxxx:xxxx

Global Address

Routable across the Internet (2000::/3)

Documentation Prefix 2001:0db8::/32 (July 2004) & 3fff::/20 (July 2024)

2000:****:****:****:xxxx:xxxx:xxxx:xxxx

3fff:****:****:****:xxxx:xxxx:xxxx:xxxx

IPv6 Address Space – PI vs PA

IPv6 Address Space – PI vs PA

- Do I get Provider Independent (PI) or Provider “Assigned” (PA)?
  - PI space is great for organizations who want to multihome to different SPs
  - PA if you are single homed or plan to NAT/Proxy everything with IPv6 (not likely)
- Possible Options for PI
  - Get one large global block from local RIR and subnet out per region
  - Get a separate block from each of the RIR you have presence in
- Most organizations are going down the PI path
  - Getting assignments across regional registries provides “insurance” against changing policies
  - Traffic Engineering

The 4 Rules

The 4 Rules

1. Simple
2. Embed information

To help troubleshooting and operation of the network

Examples:
location, country, PIN, VLAN, IPv4 addresses in Link Local and/or Global Addresses

The 4 Rules

1. Simple
2. Embed information
3. Build-in Reserve

Cater for future growth, mergers & acquisitions, new locations

Reserved vs. assigned

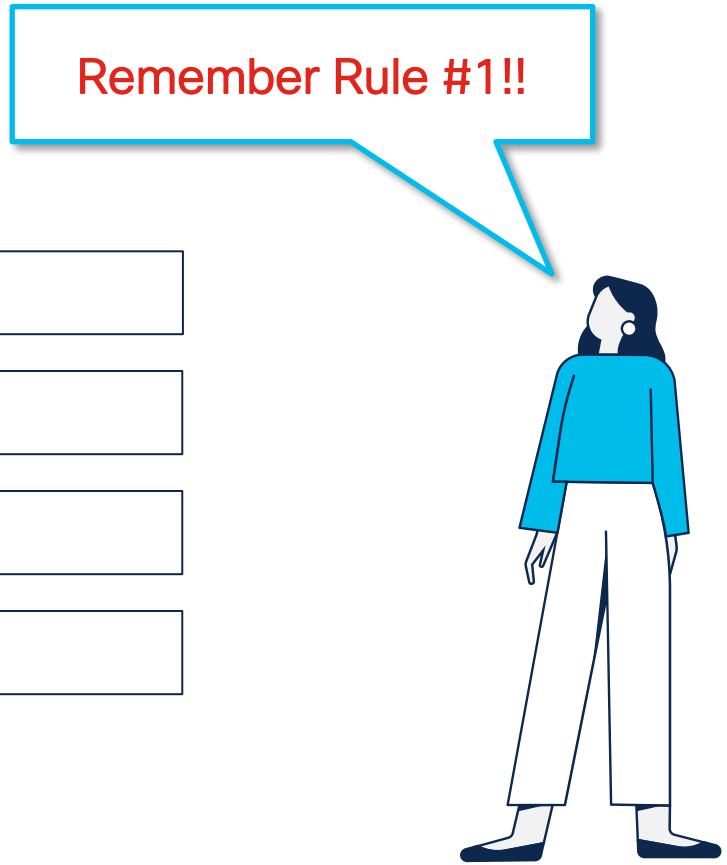
The 4 Rules

1. Simple
2. Embed information
3. Build-in Reserve
4. Aggregatable

Good aggregation is essential, just one address block (per location). Ensures scalability and stability

The 4 Rules

1. Simple
2. Embed information
3. Build-in Reserve
4. Aggregatable



“Do not fear to waste addresses”

Secret Rule #5

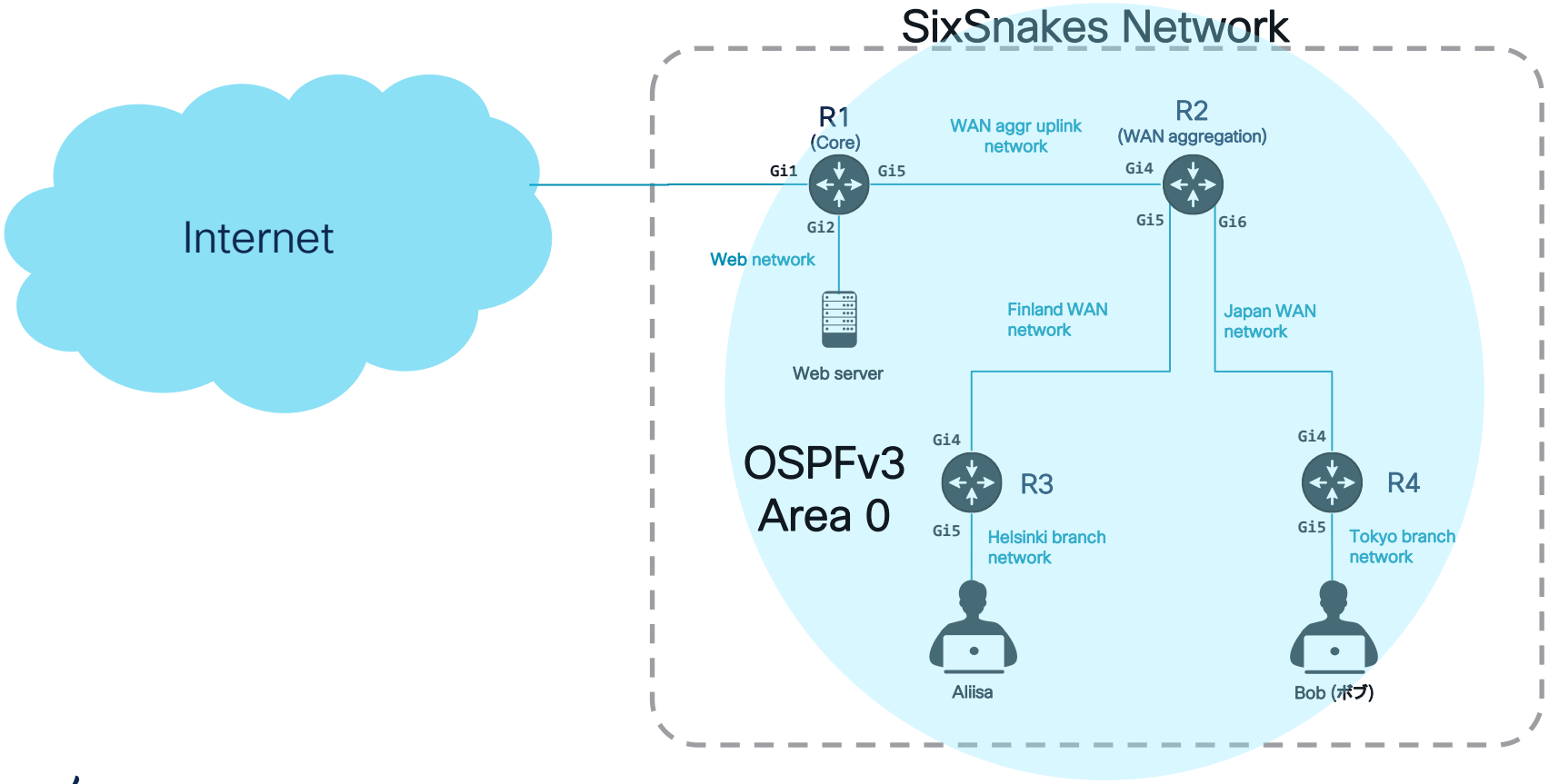
Addressing Plan! (for SixSnakes)

- RIPE gave SixSnakes the requested /32 prefix of 2001:db8::/32
- SixSnakes decides 2001:db8::/48 per branch
  - Fun Fact: A /32 contains 65,536 /48's
- 2001:db8:0::/36 – APJC
- 2001:db8:1::/36 – Reserve APJC
- 2001:db8:2:: /36– Europe 1
- 2001:db8:3::/36 – Europe 2
- 2001:db8:4::/36 – Reserve Europe

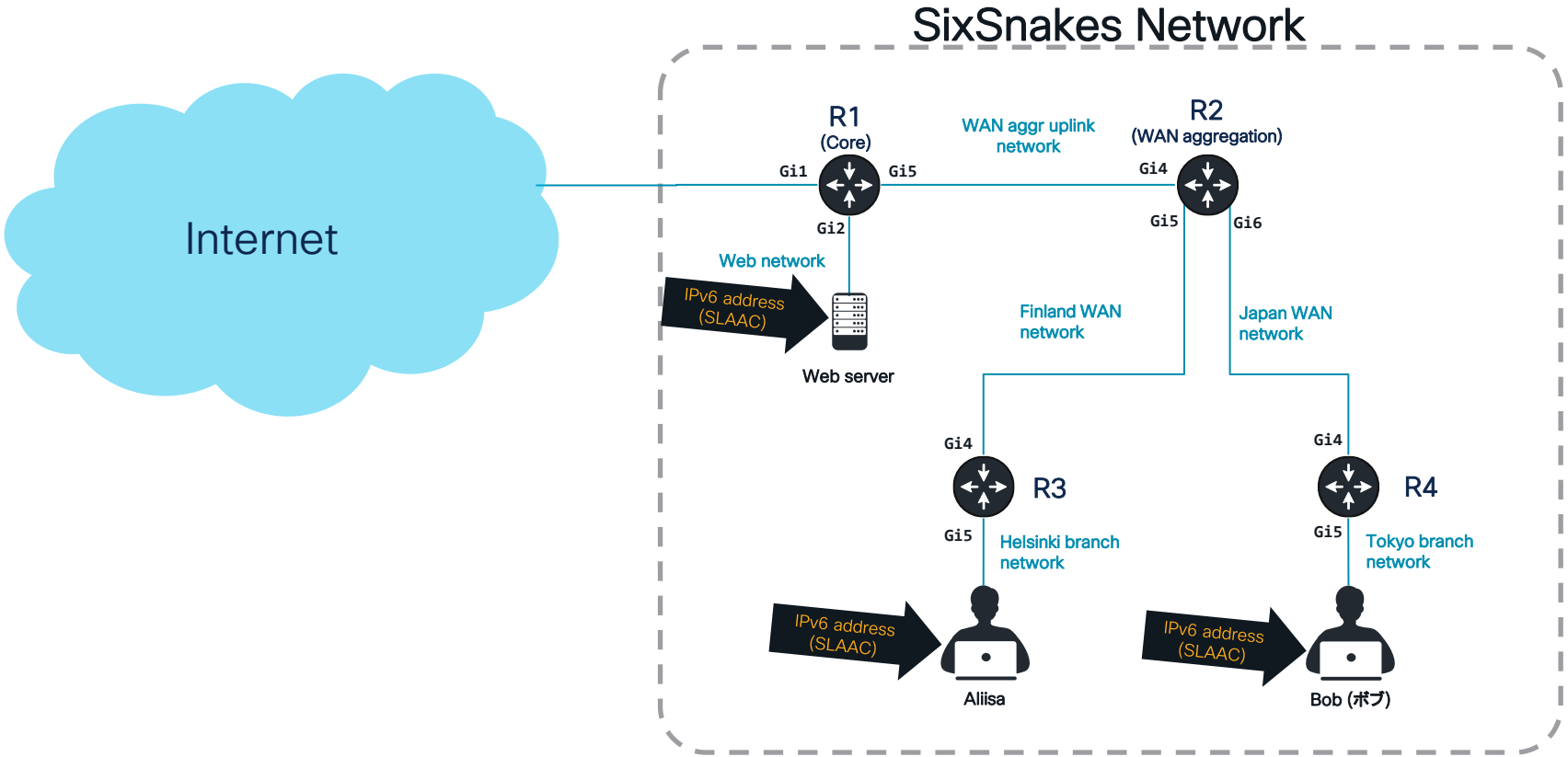
Stateless Address Auto Configuration (SLAAC)

OSPFv2 vs OSPFv3

Design for SixSnakes



Design for SixSnakes



Design towards Deploy

Design

IPv6 address

Design

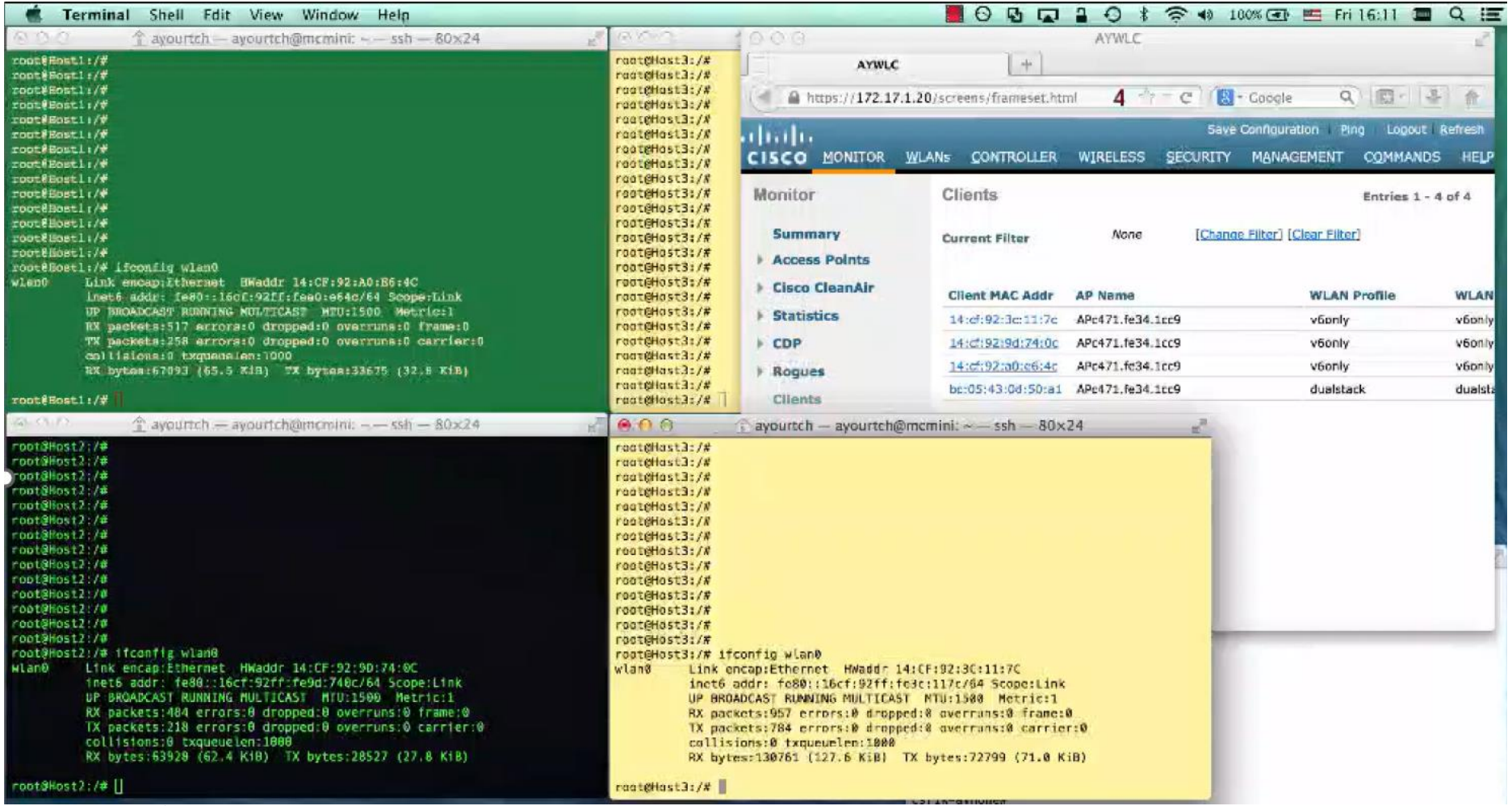
Configuration

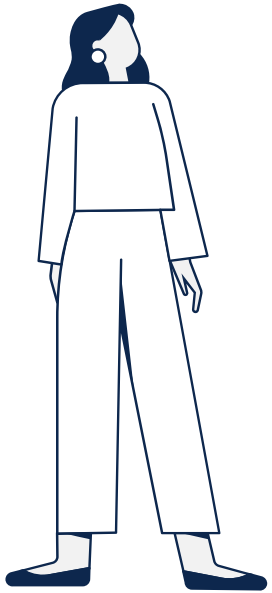
Plan

Address plan

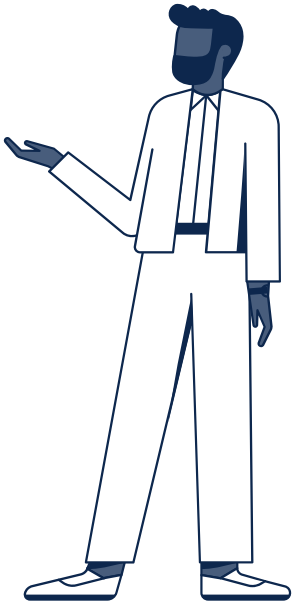
Workflow

Deployment



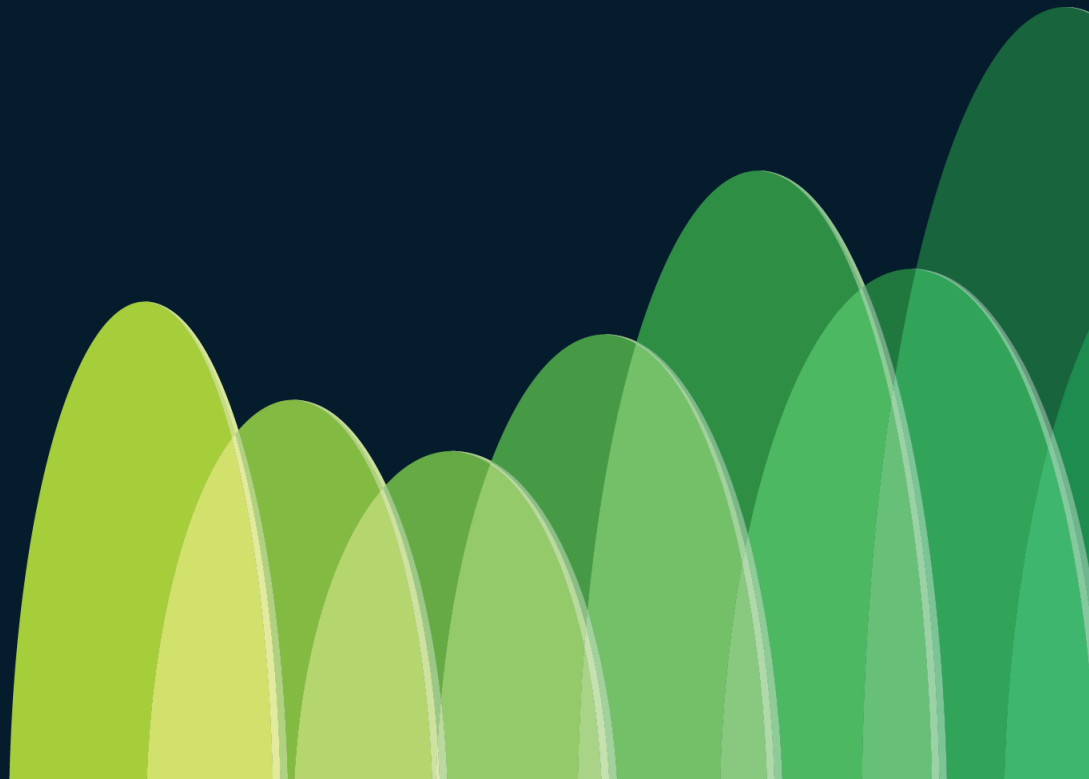


Let's deploy IPv6 with the help of your network's APIs!



1. Design | 2. Deploy | 3. Validate | 4. Scale

Deploy using your network's APIs

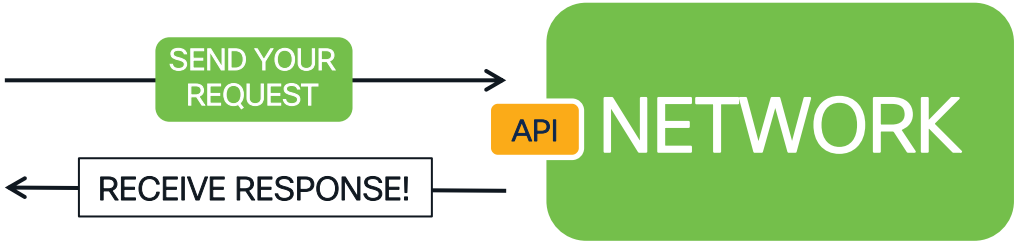


cisco Live!

How can network APIs help?

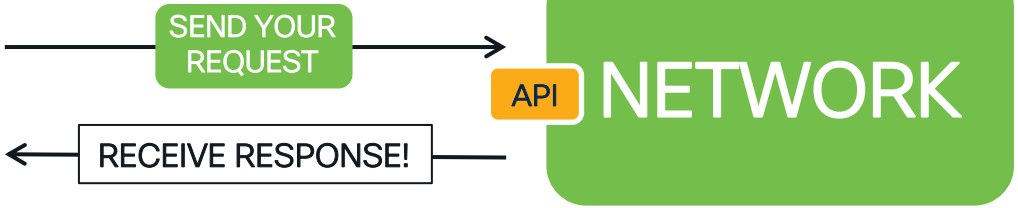
API = Application programming interface

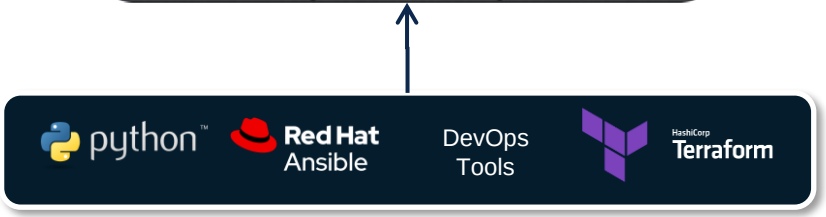
Your manual task



How can network APIs help?

Your manual task





How can network APIs help?

Your manual task

SEND YOUR REQUEST

RECEIVE RESPONSE!

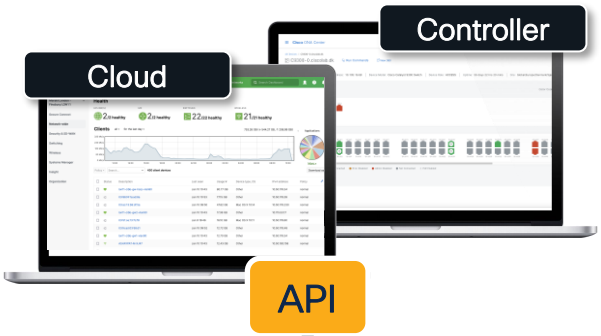
NETWORK

API

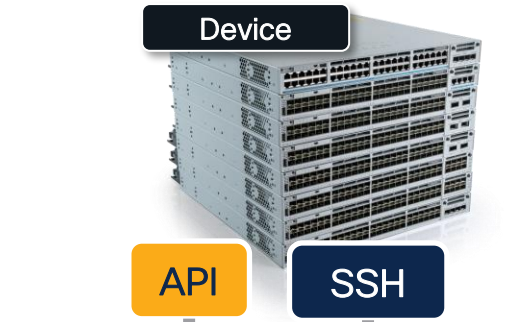
API

API

Your network offers different APIs

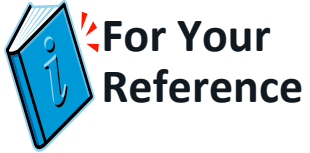


Often **REST** APIs using **URIs** and **HTTP(s)**



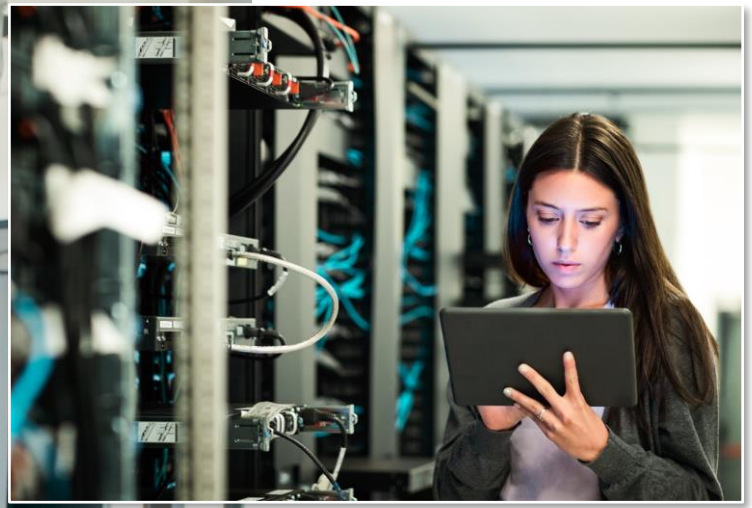
Model Driven Programmability using **YANG** data models

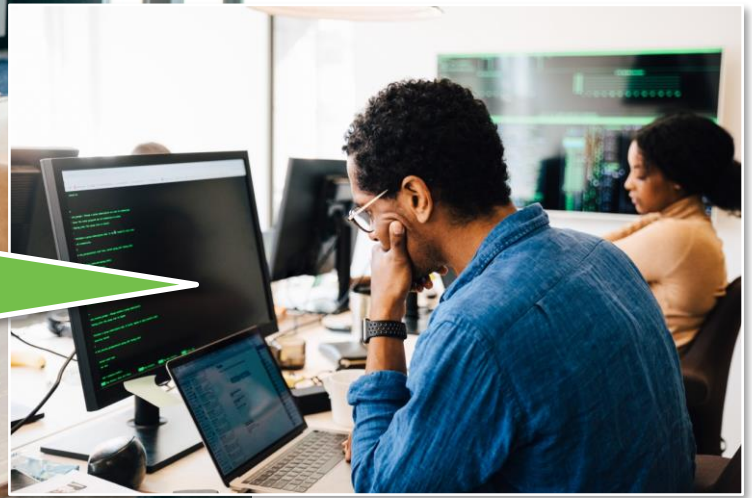
Scripts accessing **normal CLI commands** through SSH



Cisco Networking API tooling

	Cisco Meraki	Cisco Catalyst Center	Catalyst SD-WAN Manager	Cisco IOS XE
Python SDK	meraki	dnacentersdk	catalystwan EARLY BETA RELEASE	*N/A*
Ansible	Cisco.Meraki	Cisco.Dnac	catalystwan PRE-RELEASE	Cisco.Ios
Terraform	meraki	catalystcenter (BU) catalystcenter (CX)	sdwan (CX)	iosxe





Let's enable IPv6 with the help of **IOS XE YANG models**!

Demo

Deploying IPv6 on the network devices

cisco Live!

Scripting against IOS XE devices

Model Driven Programmability using **YANG** data models

Scripts accessing **normal CLI commands** through SSH

Scripting against IOS XE devices

SSH

Scripts accessing **normal CLI commands** through SSH

Example: NETMIKO

- Multi-vendor SSH Python library
- Leverage CLI expertise
- Simplify SSH connection to devices

https://github.com/ktbyers/netmiko

Code:

Scripting against IOS XE devices

SSH

Scripts accessing **normal CLI commands** through SSH

```
1  from netmiko import ConnectHandler
2
3  R2 = {
4      'device_type': 'cisco_xe',
5      'host': '198.18.7.2',
6      'username': 'developer',
7      'password': 'C1sco12345'
8  }
9
10 net_connect = ConnectHandler(**R2)
11
12 config_commands = [
13     'ipv6 unicast-routing',
14     'ipv6 router ospf 1'
15 ]
16
17 output = net_connect.send_config_set(config_commands)
18 print(output)
19
20 net_connect.disconnect()
```

Import Connect Handler from Netmiko

Define connection details to your device

Define your list of CLI configuration commands

Send commands and print the returned output

Code:

Scripting against IOS XE devices

SSH

Scripts accessing **normal CLI commands** through SSH

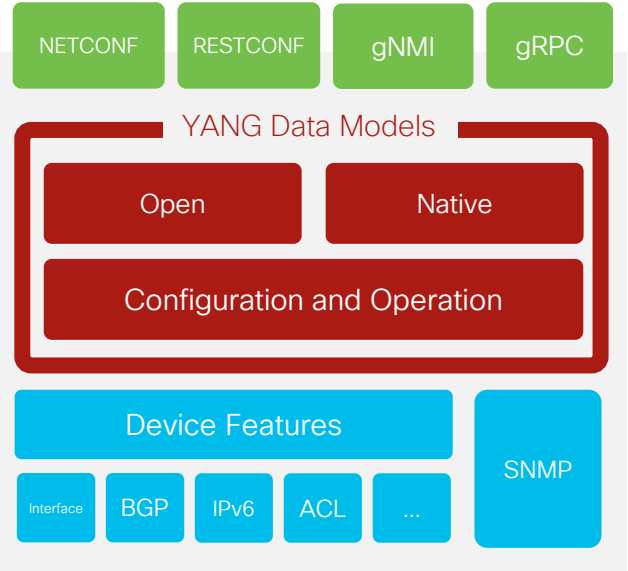
enable_ipv6.py

```python
from netmiko import ConnectHandler

R2 = {
    'device_type': 'cisco_xe',
    'host': '198.18.7.2',
    'username': 'developer',
    'password': 'C1sco12345'
}

net_connect = ConnectHandler(**R2)

config_commands = [
    'ipv6 unicast-routing',
    'ipv6 router ospf 1'
]

output = net_connect.send_config_set(config_commands)
print(output)

net_connect.disconnect()
```

TERMINAL

```
(venv) netmiko (main) $
```

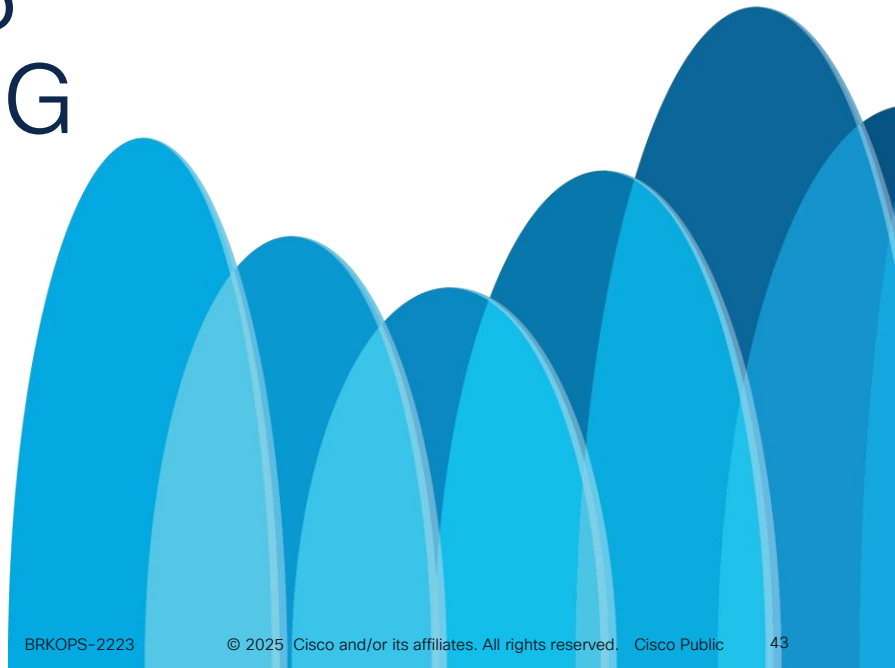
Scripting against IOS XE devices



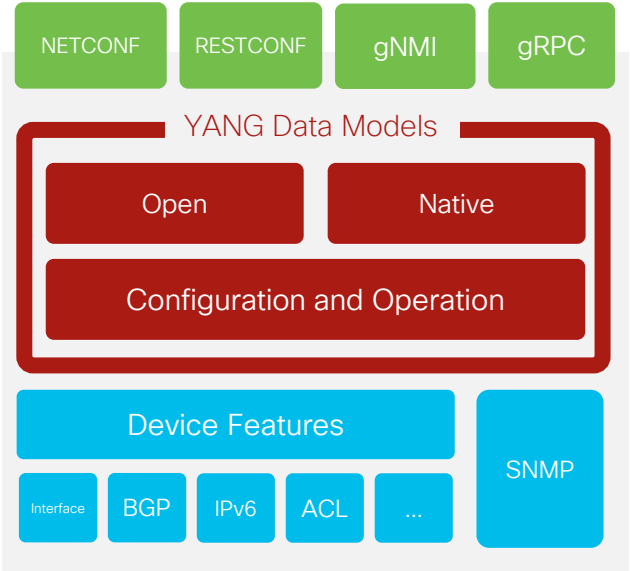
Changing mindset from **human friendly CLI** to **machine friendly models**

Model Driven Programmability using **YANG** data models

Example: Enable IPv6 on IOS XE using YANG models



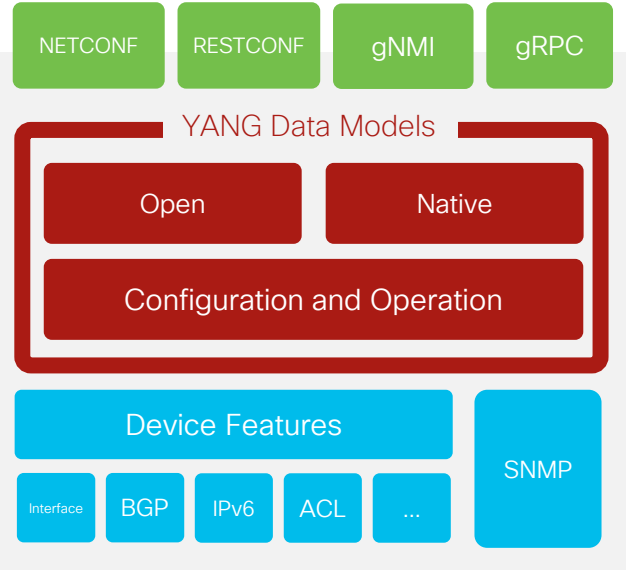
Enable IPv6 on IOS XE using YANG models



Changing mindset from **human friendly CLI** to **machine friendly models**

Model Driven Programmability using **YANG** data models

Enable IPv6 on IOS XE using YANG models

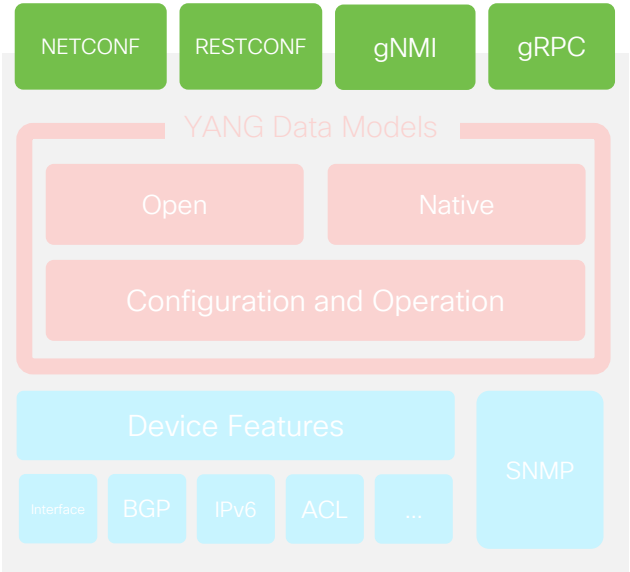


Steps:

1. Select your protocol
2. Define configuration in YANG
3. Send configuration to the device
4. Automate with Python

Enable IPv6 on IOS XE using YANG models

1. Select your protocol



Programmatic Interface

- Configuration data
- State data
- Actions and events

For Your Reference

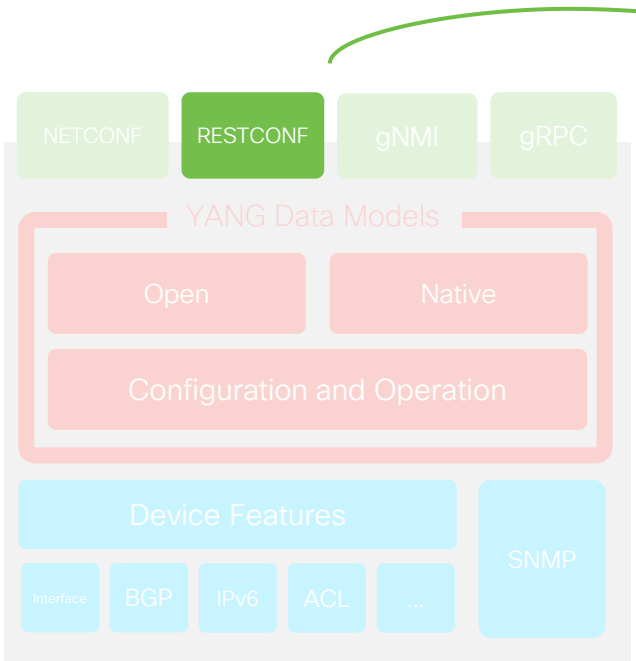
Our focus

	NETCONF (RFC 6241)	RESTCONF (RFC 8040)
Transport	RPC over SSH	HTTP(s)
Encoding	XML	XML/JSON
Commands to enable on IOS XE	`# netconf-yang`	`# ip http secure-server` `# restconf`
Python library	ncclient	requests

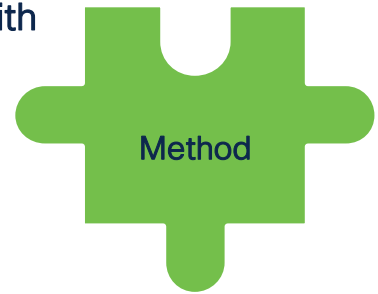
IOS XE Programmability Configuration Guide

Enable IPv6 on IOS XE using YANG models

1. Select your protocol



Important with RESTCONF:



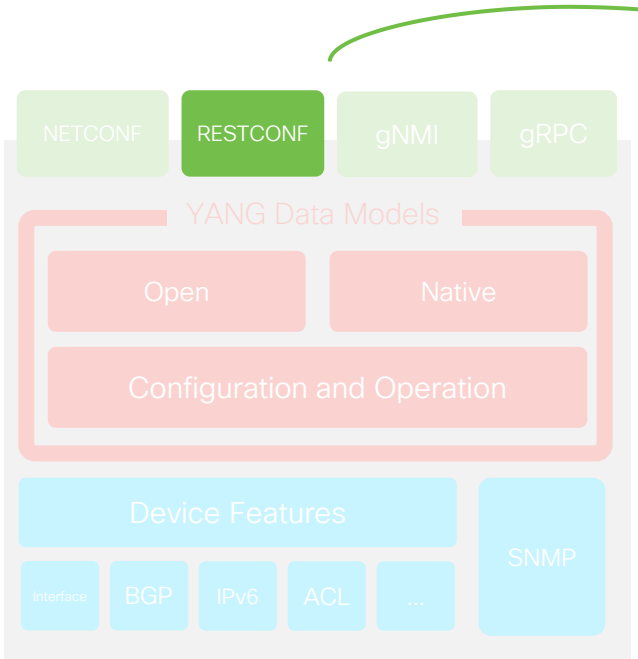

```
R1# show …
R1# conf t
```

GET POST DELETE

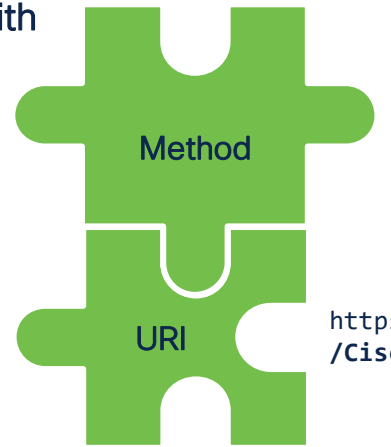
PUT HEAD PATCH

Enable IPv6 on IOS XE using YANG models

1. Select your protocol



Important with RESTCONF:

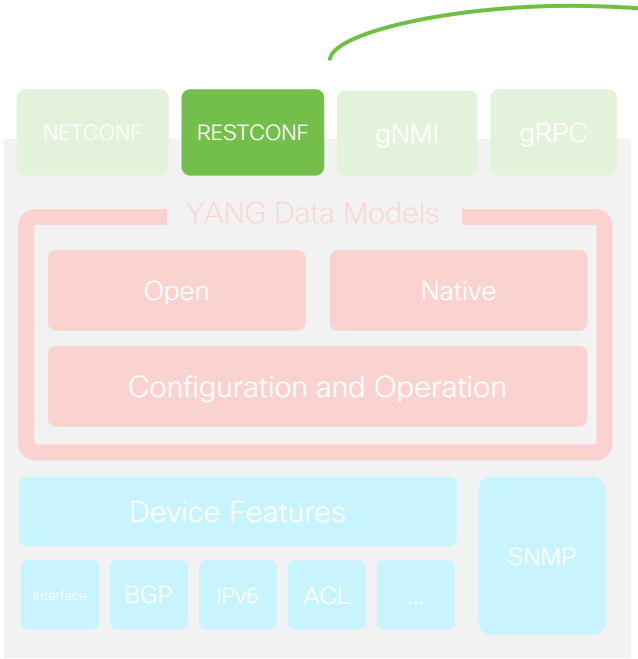



```
R1(config)# int Gi1
```

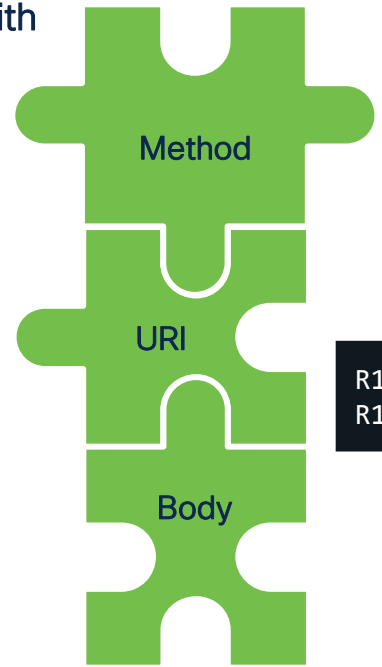
```
https://<device>/restconf/data
/Cisco-IOS-XE-native:native/interface
```

Enable IPv6 on IOS XE using YANG models

1. Select your protocol



Important with RESTCONF:

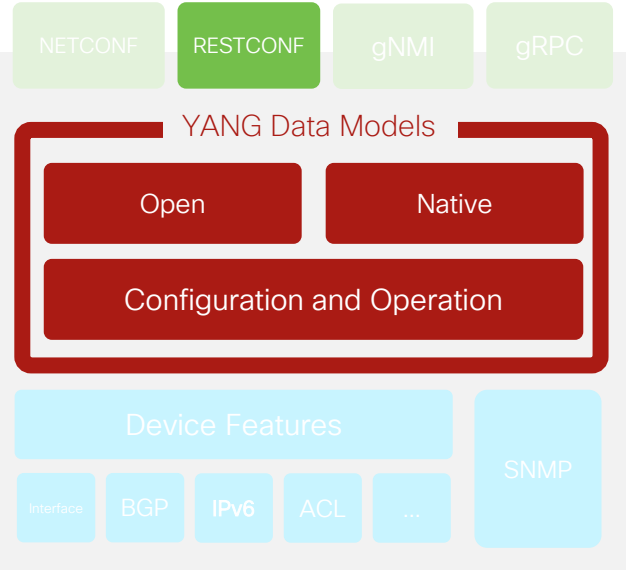



```
R1(config-if)# ipv6 enable
R1(config-if)# ipv6 ospf 1 area 0
```

YANG model

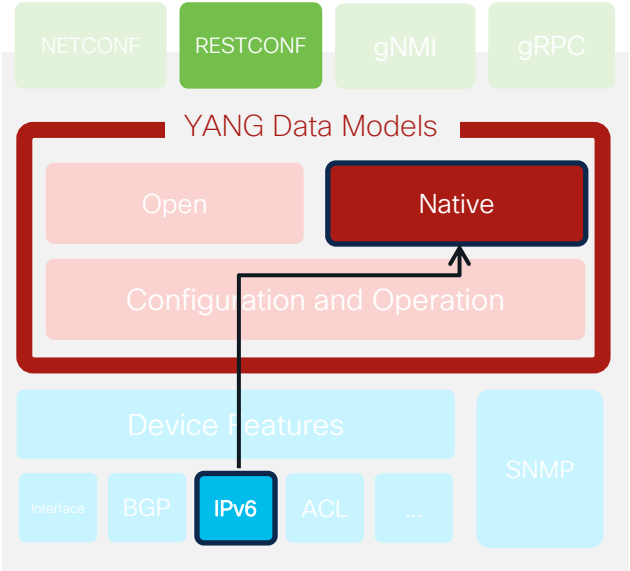
Enable IPv6 on IOS XE using YANG models

2. Define configuration in YANG



Enable IPv6 on IOS XE using YANG models

2. Define configuration in YANG



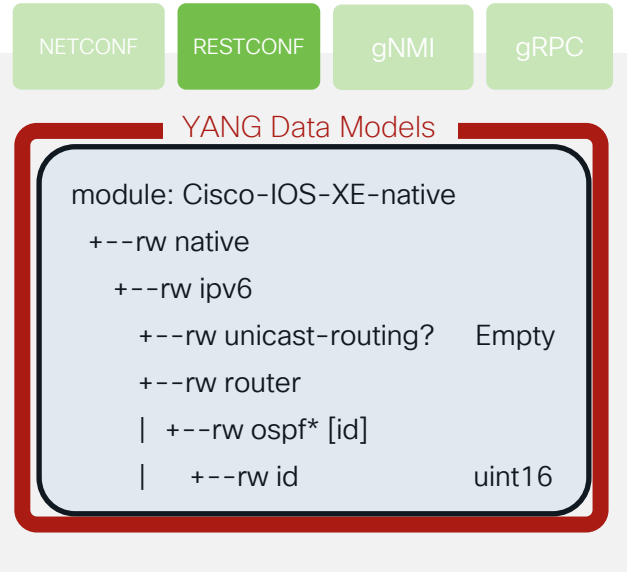
YANG Data Model

```
module: Cisco-IOS-XE-native
  +--rw native
    +--rw ipv6
      +--rw unicast-routing?     Empty
      +--rw router
      |  +--rw ospf* [id]
      |     +--rw id                uint16
```

Note: This snippet is **not exhaustive** and **represents only a portion of the available options.**

Enable IPv6 on IOS XE using YANG models

2. Define configuration in YANG



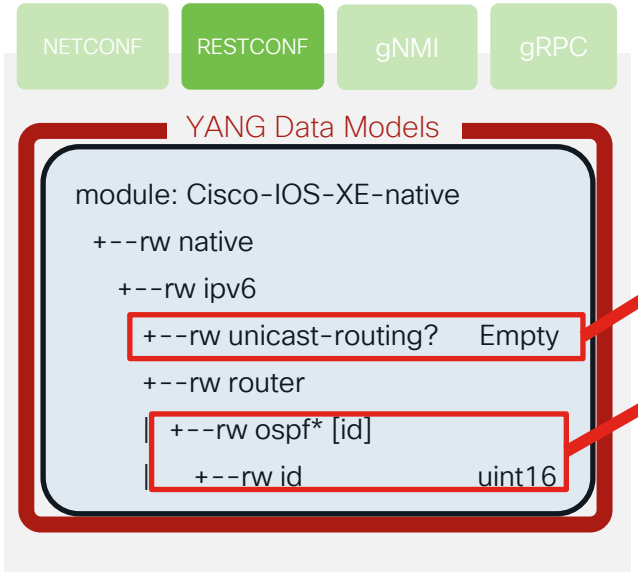
Configuration payload

JSON

```
{
    "Cisco-IOS-XE-native:ipv6": {
        "unicast-routing": [null],
        "router": {
            "ospf": [{"id": 1}]
        }
    }
}
```

Enable IPv6 on IOS XE using YANG models

2. Define configuration in YANG



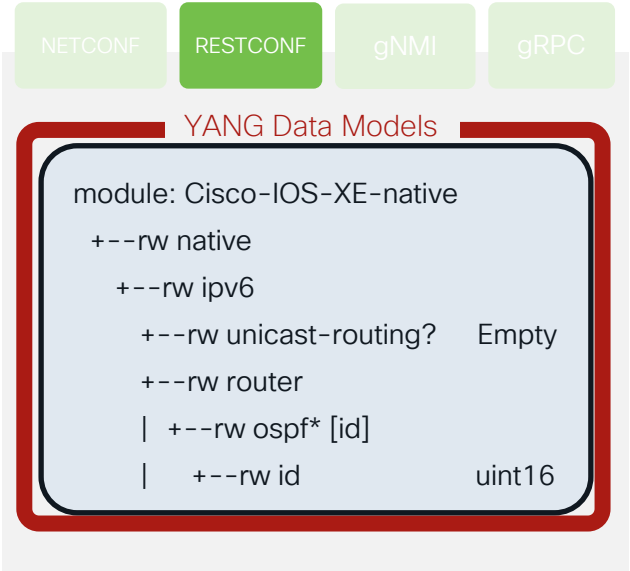
Configuration payload

JSON

```
{
    "Cisco-IOS-XE-native:ipv6": {
        "unicast-routing": [null],
        "router": {
            "ospf": [{"id": 1}]
        }
    }
}
```

Enable IPv6 on IOS XE using YANG models

3. Send configuration to the device

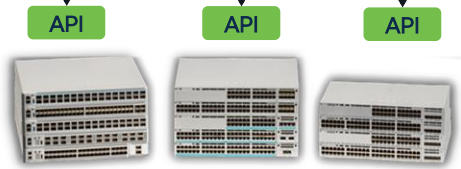


Configuration payload

```
{
    "Cisco-IOS-XE-native:ipv6": {
        "unicast-routing": [null],
        "router": {
            "ospf": [{"id": 1}]
        }
    }
}
```

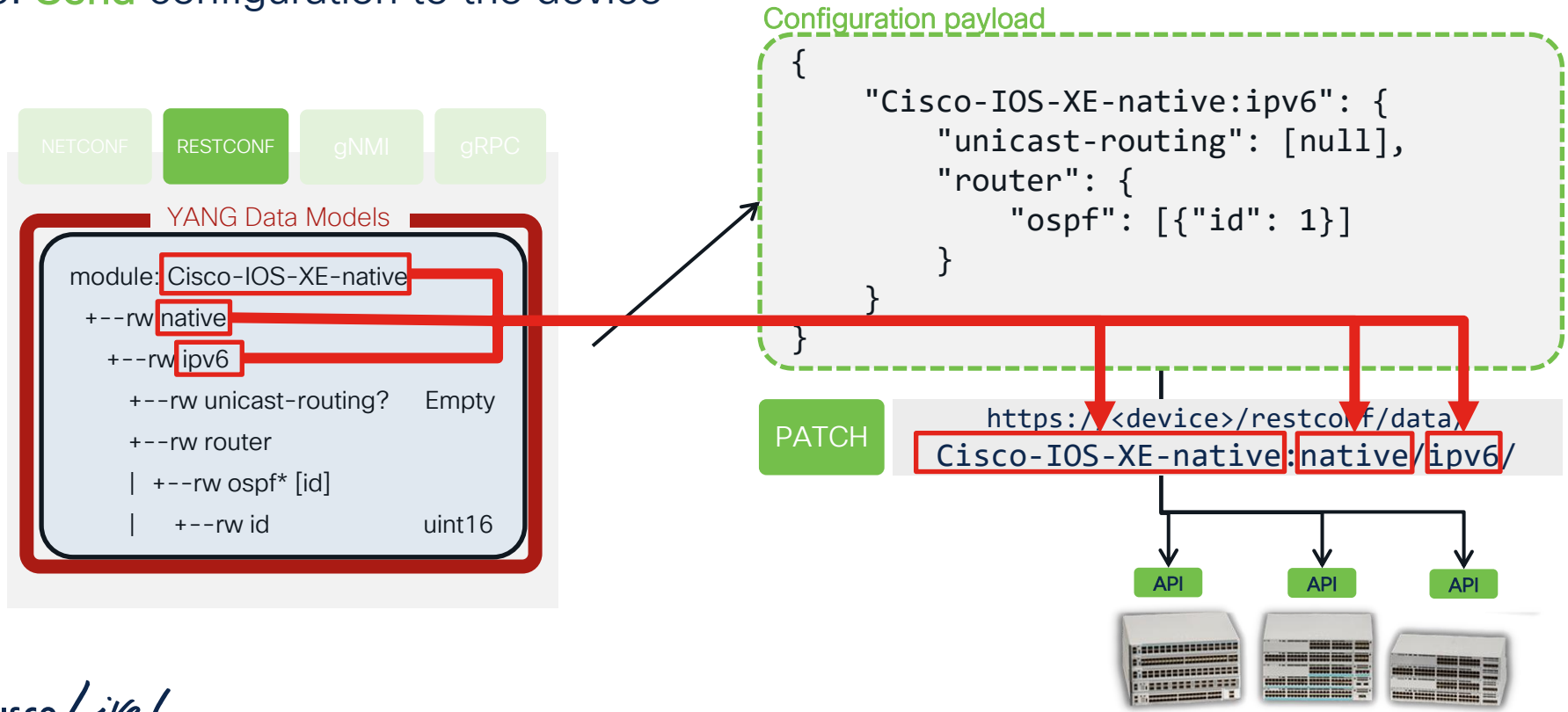
PATCH

`https://<device>/restconf/data/Cisco-IOS-XE-native:native/ipv6/`



Enable IPv6 on IOS XE using YANG models

3. Send configuration to the device



Code:

Enable IPv6 on IOS XE using YANG models

4. Automate with Python

Configuration payload

```
{
    "Cisco-IOS-XE-native:ipv6": {
        "unicast-routing": [null],
        "router": {
            "ospf": [{"id": 1}]
        }
    }
}
```

PATCH `https://<device>/restconf/data/Cisco-IOS-XE-native:native/ipv6/`

```python
import requests

HOST = "198.18.7.2"
PASSWORD = "C1sco12345"
USERNAME = "developer"

payload = {
    "Cisco-IOS-XE-native:ipv6": {
        "unicast-routing": [None],
        "router": {
            "ospf": [{"id": 1}]
        }
    }
}

url = f"https://{HOST}:443/restconf/data/Cisco-IOS-XE-native:native/ipv6/"

header = {"Content-Type": "application/yang-data+json"}

response = requests.patch(url, headers=header,
                          auth=(USERNAME, PASSWORD),
                          json=payload, verify=False)
```

Code:

Enable IPv6 on IOS XE using YANG models

4. Automate with Python

Configuration payload

```
{
    "Cisco-IOS-XE-native:ipv6": {
        "unicast-routing": [null],
        "router": {
            "ospf": [{"id": 1}]
        }
    }
}
```

PATCH `https://<device>/restconf/data/Cisco-IOS-XE-native:native/ipv6/`

```
import requests

HOST = "198.18.7.2"
PASSWORD = "C1sco12345"
USERNAME = "developer"

payload = {
    "Cisco-IOS-XE-native:ipv6": {
        "unicast-routing": [None],
        "router": {
            "ospf": [{"id": 1}]
        }
    }
}

url = f"https://{HOST}:443/restconf/data/Cisco-IOS-XE-native:native/ipv6/"

header = {"Content-Type": "application/yang-data+json"}

response = requests.patch(url, headers=header,
                          auth=(USERNAME, PASSWORD),
                          json=payload, verify=False)
```

Import Requests library

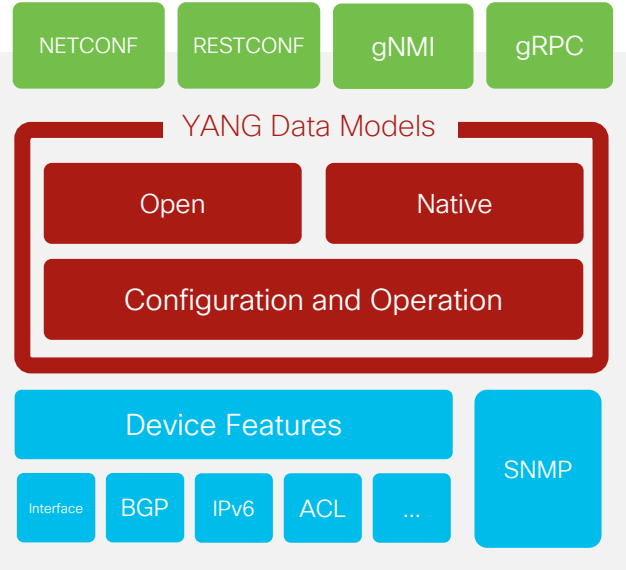
Define connection details to your device

Define your configuration in YANG

URL for the targeted resource

Send to device using the required HTTP(s) method

Enable IPv6 on IOS XE using YANG models

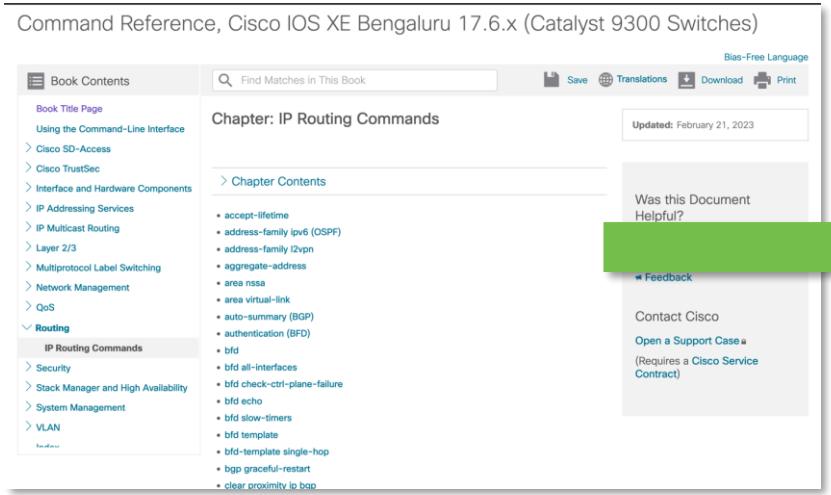


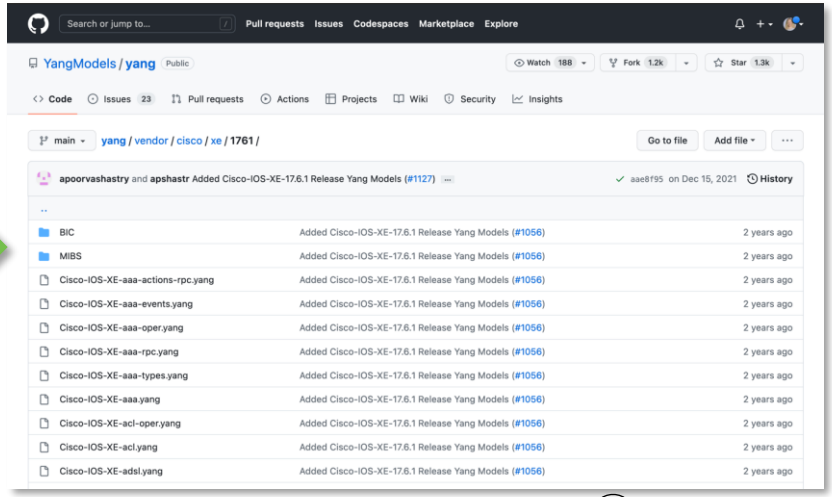
Steps:

1. Select your protocol
2. Define configuration in YANG
3. Send configuration to the device
4. Automate with Python

From command reference to YANG data models

How do I know what YANG data models are available?





YANG model documentation can be found from the Github repository, for example: [models supported on IOS XE 17.6]

Tools to help read and use the the models

1. Python library pyang

```
$ pip install pyang
```

```
$ pyang -f tree --tree-path="/native/ipv6" Cisco-IOS-XE-native.yang
```

YANG model cloned from Github

TERMINAL

```
module: Cisco-IOS-XE-native
  +--rw native
     +--rw ipv6
        +--rw source-guard
        |  +--rw policy* [source-guard-policy-name]
        |     +--rw source-guard-policy-name    string
        |     +--rw validate
        |     |  +--rw address-config?   boolean
        |     |  x--rw address?          empty
        |     |  +--rw prefix?           empty
        |     +--rw deny
        |     |  +--rw global-autoconf?   empty
        |     +--rw permit
        |     |  +--rw link-local?   empty
        |     +--rw trusted?                    empty
        +--rw destination-guard
        |  +--rw policy* [destination-guard-policy-name]
        |     +--rw destination-guard-policy-name    string
```

pyang documentation: https://github.com/mbj4668/pyang

Tools to help read and use the the models

2. Directly on your IOS XE CLI

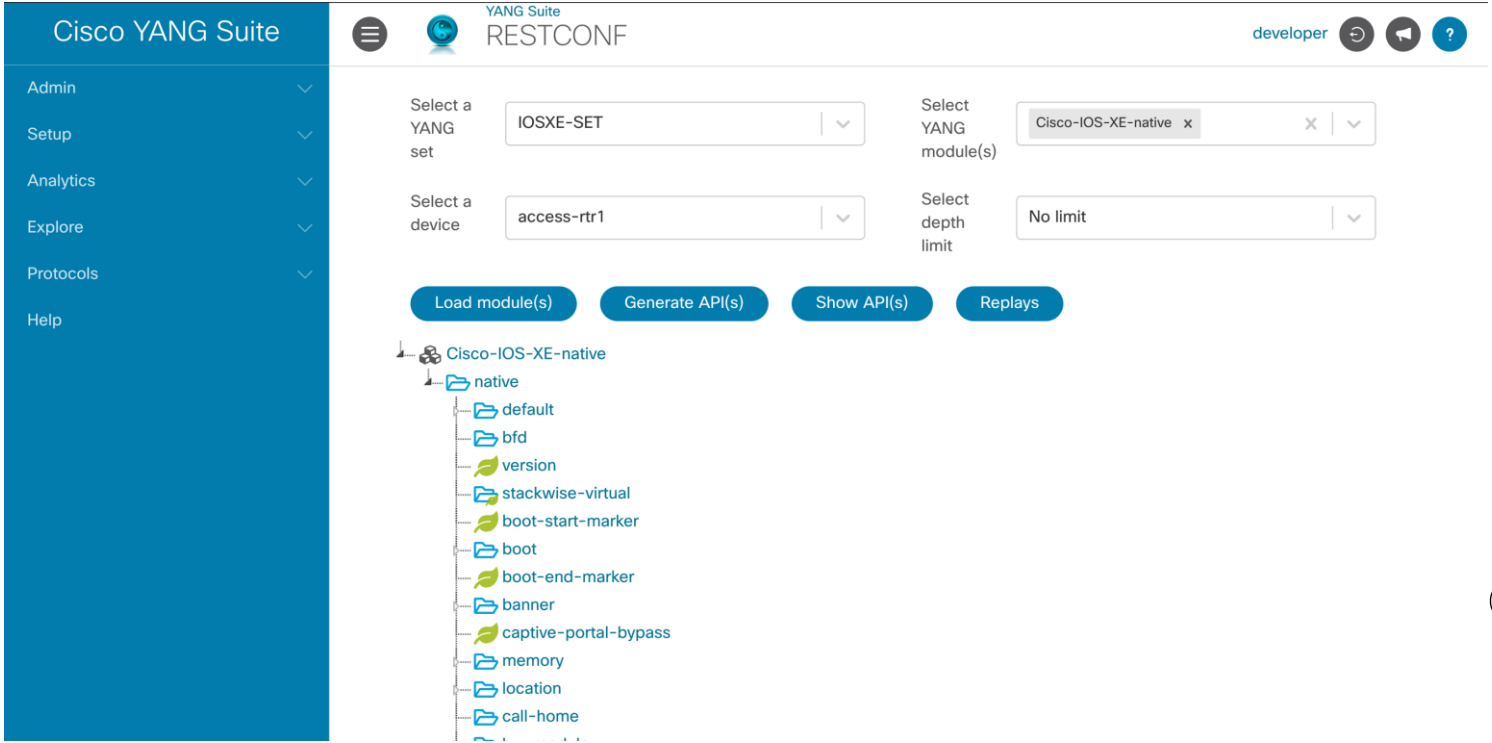
```
# show running-config | format restconf-json
```

```
# show running-config | format netconf-xml
```

Note: Supported on IOS XE 17.7.1 and later releases

Tools to help read and use the the models

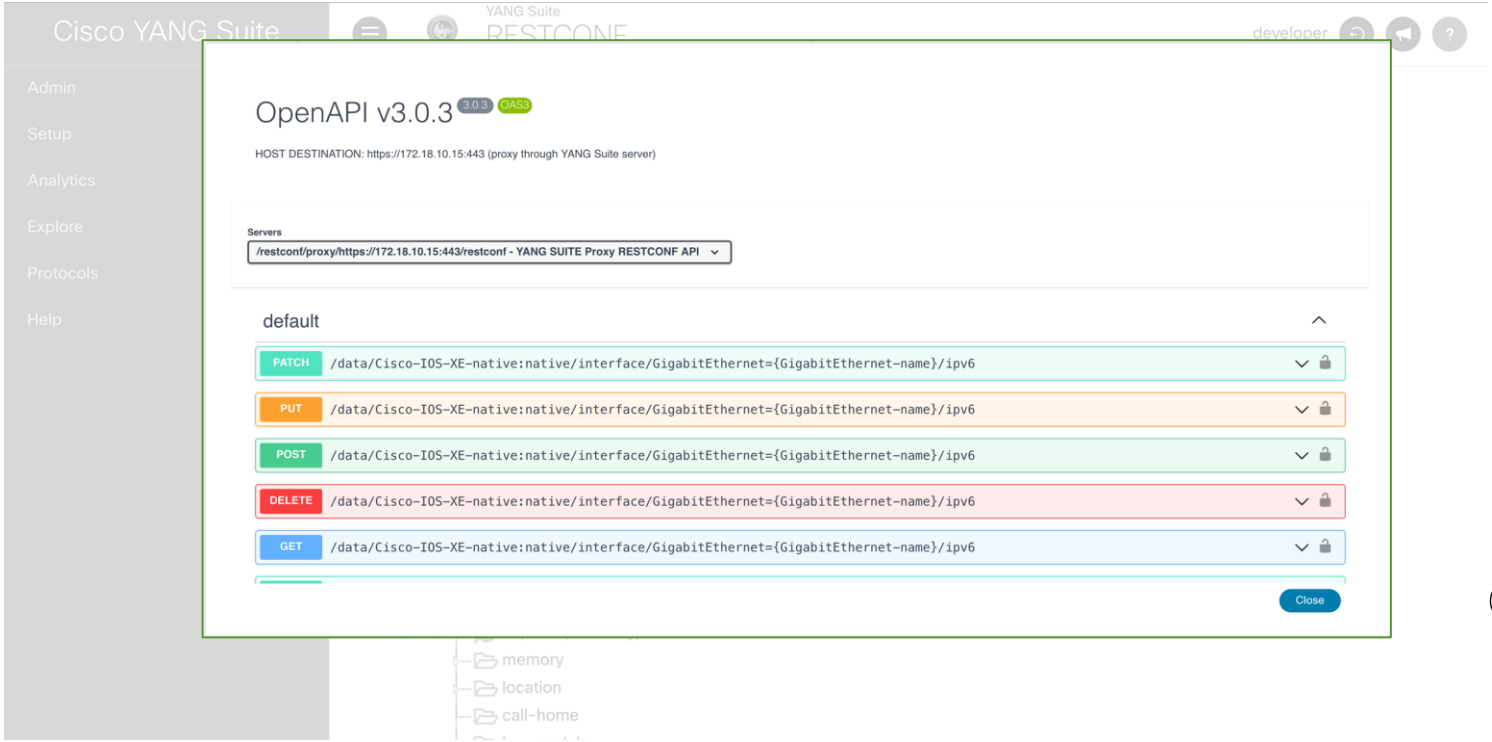
3. YANG Suite



YANG suite can be downloaded from here: https://developer.cisco.com/yangsuite/

Tools to help read and use the the models

3. YANG Suite



YANG suite can be downloaded from here: https://developer.cisco.com/yangsuite/

Time to deploy the full configuration

Define the full configuration

These examples are **not** scalable or robust for production – Ch4 will cover more on this!

Example with **Netmiko**

```python
from netmiko import ConnectHandler

R2 = {
    'device_type': 'cisco_xe',
    'host': '198.18.7.2',
    'username': 'developer',
    'password': 'C1sco12345'
}

net_connect = ConnectHandler(**R2)

config_commands = [
    'ipv6 unicast-routing',
    'ipv6 router ospf 1',
    'int gig 4',
    'ipv6 enable',
    'ipv6 ospf 1 area 0',
    'int gig 5',
    'ipv6 enable',
    'ipv6 ospf 1 area 0',
    'int gig 6',
    'ipv6 enable',
    'ipv6 ospf 1 area 0'
]

output = net_connect.send_config_set(config_commands)
print(output)

net_connect.disconnect()
```

Example with **RESTCONF**

```python
import requests

HOST = "198.18.7.2"
PASSWORD = "C1sco12345"
USERNAME = "developer"

payload = {…

url = f"https://{HOST}:443/restconf/data/Cisco-IOS-XE-native:native/"

header = {"Content-Type": "application/yang-data+json"}

response = requests.patch(url, headers=header,
                          auth=(USERNAME, PASSWORD),
                          json=payload, verify=False)
```

Code:

Define the full configuration

Example with **Netmiko**

```
11
12  config_commands = [
13      'ipv6 unicast-routing',
14      'ipv6 router ospf 1',
15      'int gig 4',
16      'ipv6 enable',
17      'ipv6 ospf 1 area 0',
18      'int gig 5',
19      'ipv6 enable',
20      'ipv6 ospf 1 area 0',
21      'int gig 6',
22      'ipv6 enable',
23      'ipv6 ospf 1 area 0'
24  ]
25
26  output = net_connect.send_config_set(config_command
    print(output)
```

Commands after each other as you would type them on CLI

Code:

Define the full configuration

```
6
7  payload = {…
57
58  url = f"https://{HOST}:443/restconf/data/Cisco-IOS-XE-native:native/"
59
60  header = {"Content-Type": "application/yang-data+json"}
61
62  response = requests.patch(url, headers=header,
63                            auth=(USERNAME, PASSWORD),
64                            json=payload, verify=False)
65
```

Extend **RESTCONF** payload

Code:

Define the full configuration

```
6
7  payload = {…
57
58 url = f"https://{HOST}:443/restconf/data/Cisco-IOS-XE-native:native/"
59
60 header = {"Content-Type": "application/yang-data+json"}
61
62 response = requests.patch(url, headers=header,
63                           auth=(USERNAME, PASSWORD),
64                           json=payload, verify=False)
65
```

Example with **RESTCONF**

All the device's configuration defined in one YANG payload

Scripting against IOS XE devices

API

SSH

Example covered: RESTCONF

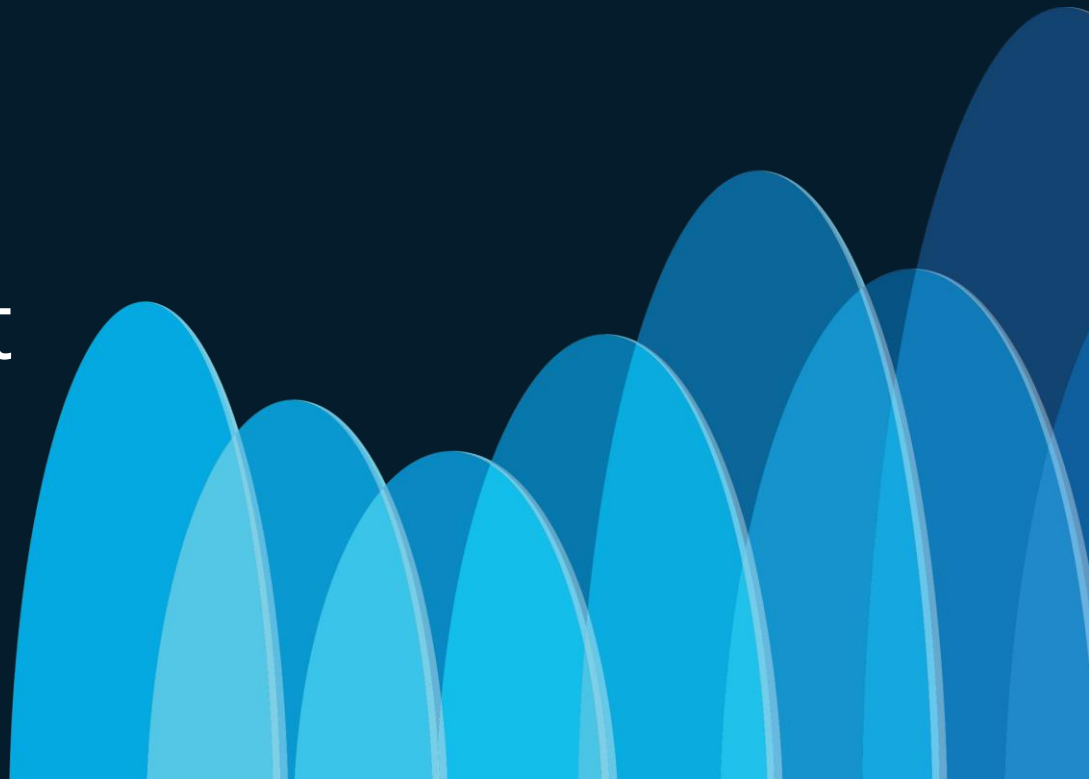
Model Driven Programmability using **YANG** data models

Scripts accessing **normal CLI commands** through SSH

Example covered: Netmiko

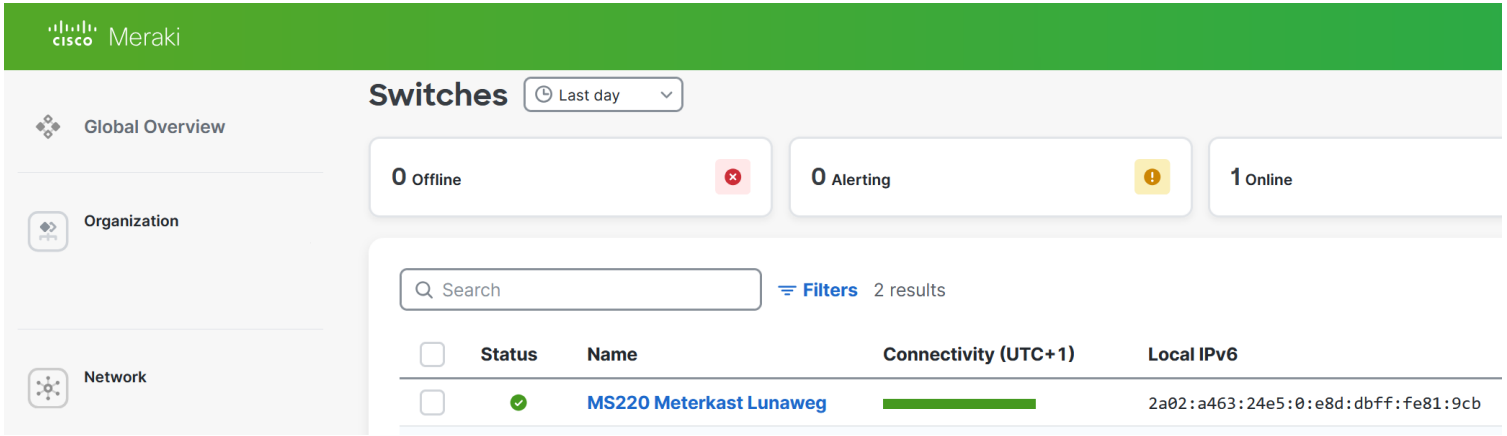
1. Design | 2. Deploy | 3. Validate | 4. Scale

Validate and troubleshoot your deployment



cisco Live!

Meraki Dashboard



Catalyst Center

Catalyst Center | Provision / Inventory

SJ1

All | Routers | Switches | Wireless Controllers | Access Points | Sensors

DEVICE WORK ITEMS

- Unreachable
- Unassigned
- Untagged
- Failed Provision
- Non Compliant
- Outdated Software Image
- No Golden Image
- Failed Image Prechecks
- Under Maintenance
- Security Advisories

Devices (6) Focus: Inventory

Take a tour | Export

Click here to apply basic or advanced filters or view recently applied filters

0 Selected | Tag | Add Device | Edit Device | Delete Device | Actions

As of: Jan 21, 2025 3:12 PM

Tags	Device Name	IP Address	Vendor	Reachability	EoX Status	Manageability	Compliance
	AP5CE1.7628.EE30	2001:db8:1:1:1833:c2d0:9f2e:a3ff	NA	Reachable	Not Scanned	Managed	NA
	AP5CE1.7628.F698	2001:db8:1:1:9037:9c:9027:431c	NA	Reachable	Not Scanned	Managed	NA
	C9300-FE	2001:db8:1:1::3	Cisco	Reachable	1 alert	Managed	Non-Compliant
	C9300Access-1-Pod2.cisco.com	2001:db8:1:1::2	Cisco	Reachable	1 alert	Managed	Non-Compliant
	Core-IPV6	2001:db8:1:1::1	Cisco	Reachable	5 alerts	Managed	Non-Compliant
	Pod2-WLC.tmelab.com	2001:db8:1:1::30	Cisco	Reachable	1 alert	Managed	Non-Compliant

6 Record(s)

Show Records: 25 | 1 - 6

Catalyst Center

Catalyst Center | Provision / Inventory

SJ1

All | Routers | Switches | Wireless Controllers | Access Points | Sensors

DEVICE WORK ITEMS

- Unreachable
- Unassigned
- Untagged
- Failed Provision
- Non Compliant
- Outdated Software Image
- No Golden Image
- Failed Image Prechecks
- Under Maintenance
- Security Advisories

Devices (6) Focus: Inventory

Take a tour | Export

Click here to apply basic or advanced filters or view recently applied filters

0 Selected | Tag | Add Device | Edit Device | Delete Device | Actions

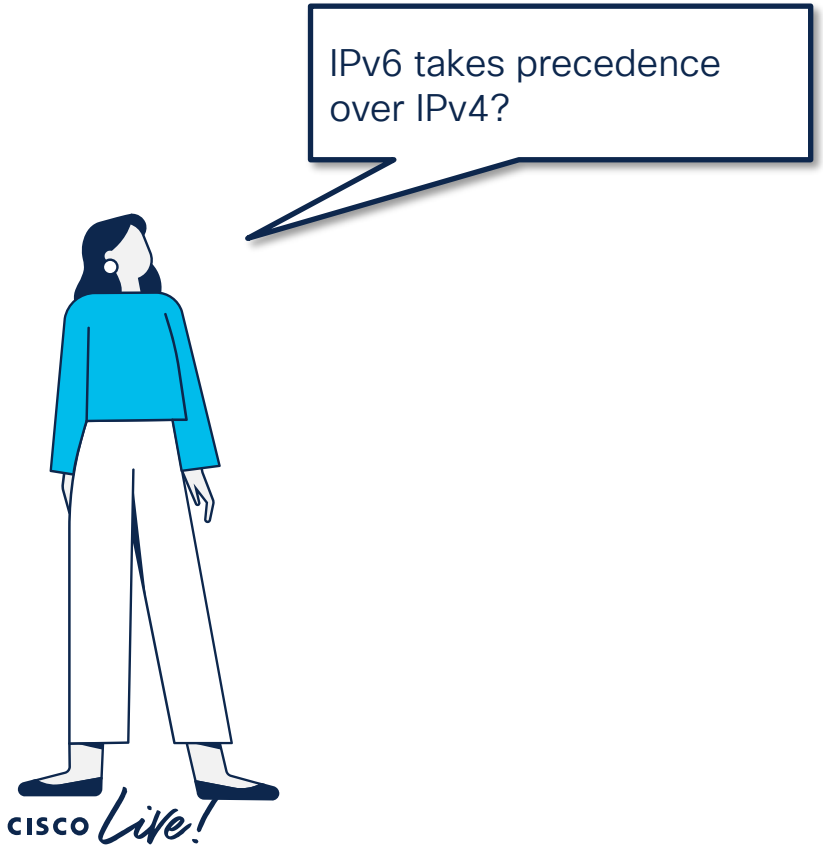
As of: Jan 21, 2025 3:12 PM

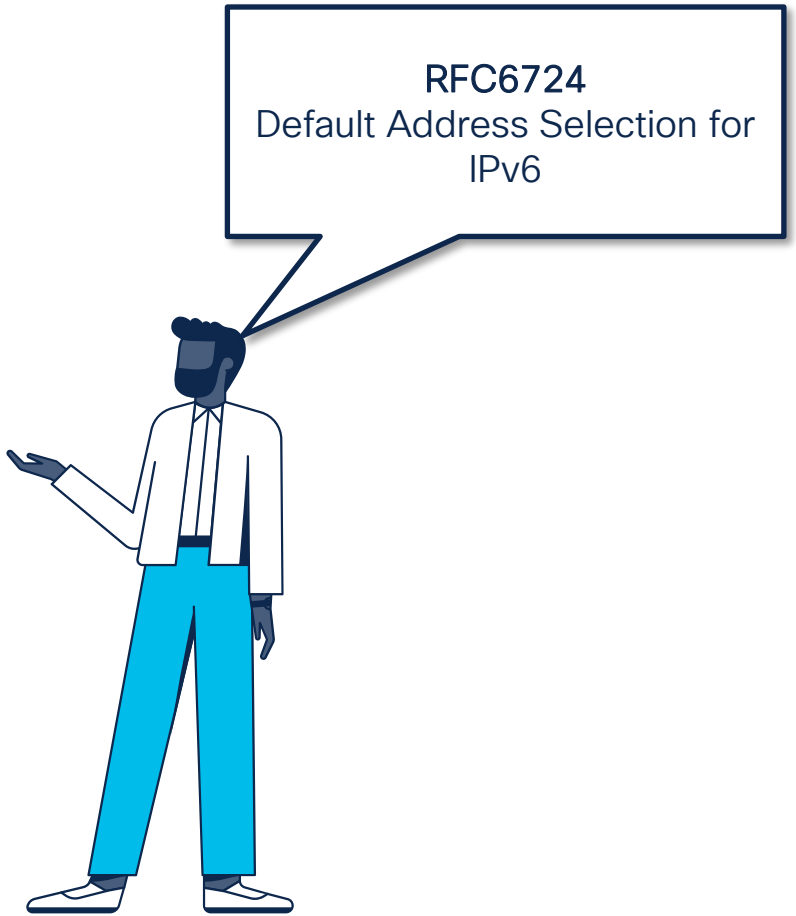
Tags	Device Name	IP Address	Vendor	Reachability	EoX Status	Manageability	Compliance
	AP5CE1.7628.EE30	2001:db8:1:1:1833:c2d0:9f2e:a3ff				Managed	NA
	AP5CE1.7628.F698	2001:db8:1:1:9037:9c:9027:431c				Managed	NA
	C9300-FE	2001:db8:1:1::3				Managed	Non-Compliant
	C9300Access-1-Pod2.cisco.com	2001:db8:1:1::2				Managed	Non-Compliant
	Core-IPV6	2001:db8:1:1::1				Managed	Non-Compliant
	Pod2-WLC.tmelab.com	2001:db8:1:1::30				Managed	Non-Compliant

6 Record(s)

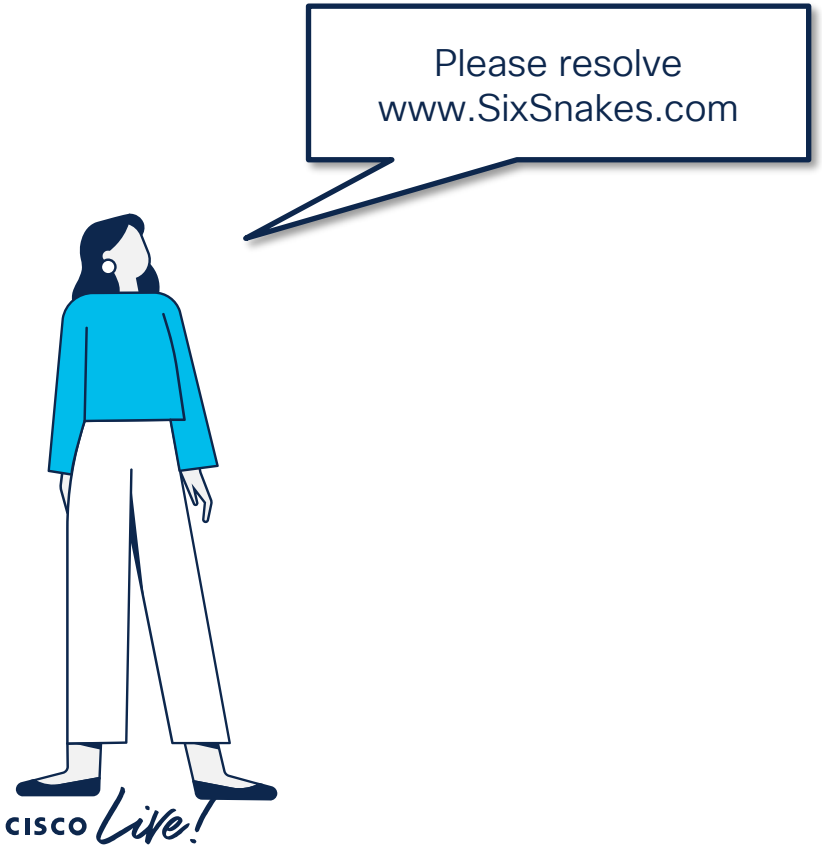
Show Records: 25 | 1 - 6

Which address-family first?

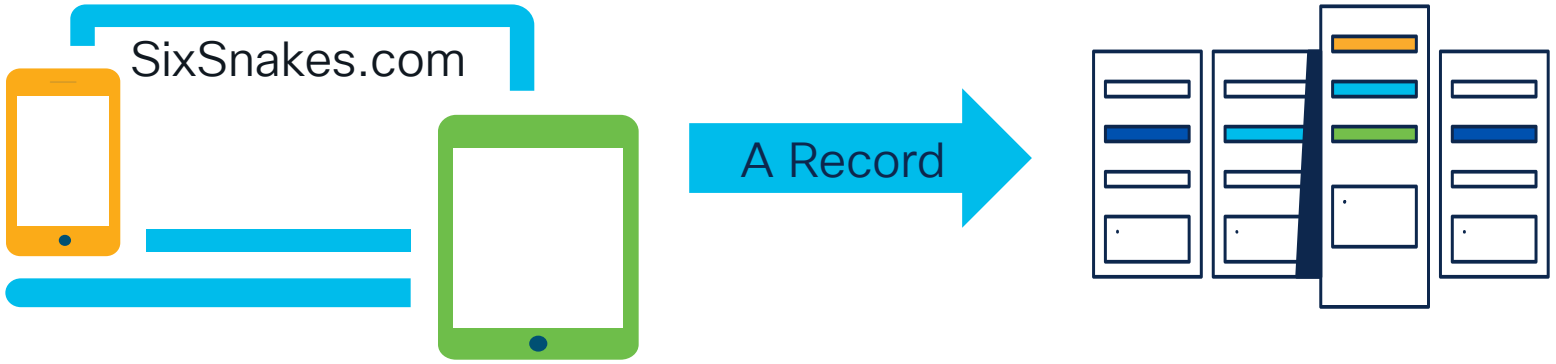


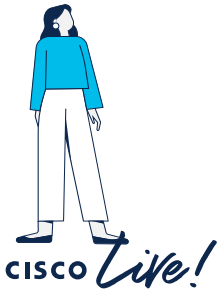


Resolving hosts – the IPv4 way

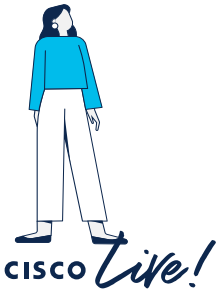


Resolving hosts – the IPv4 way

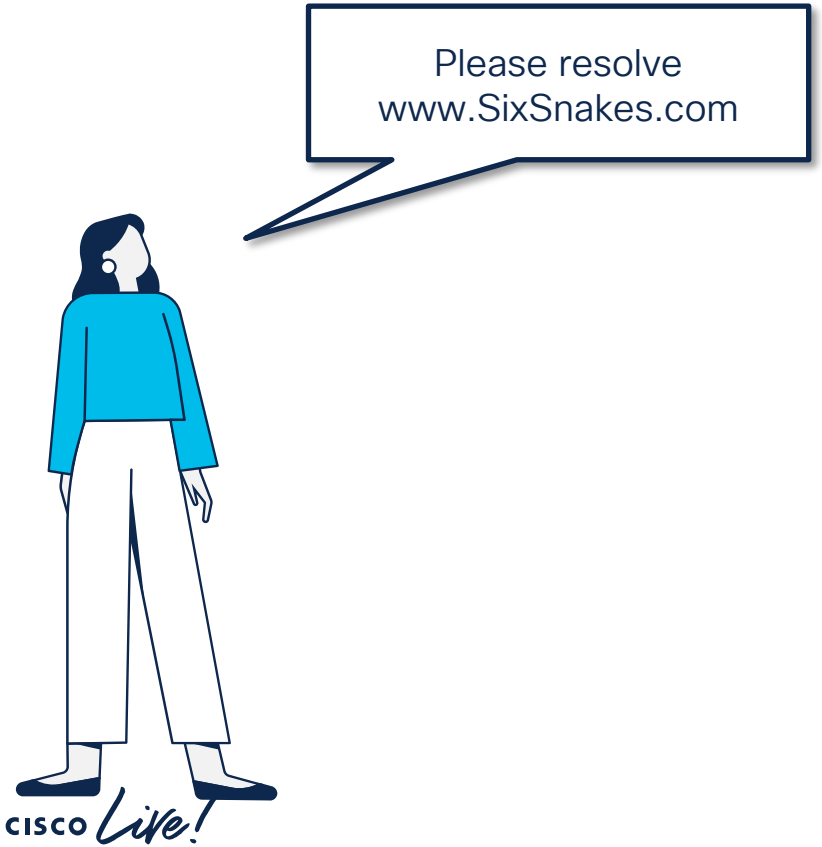




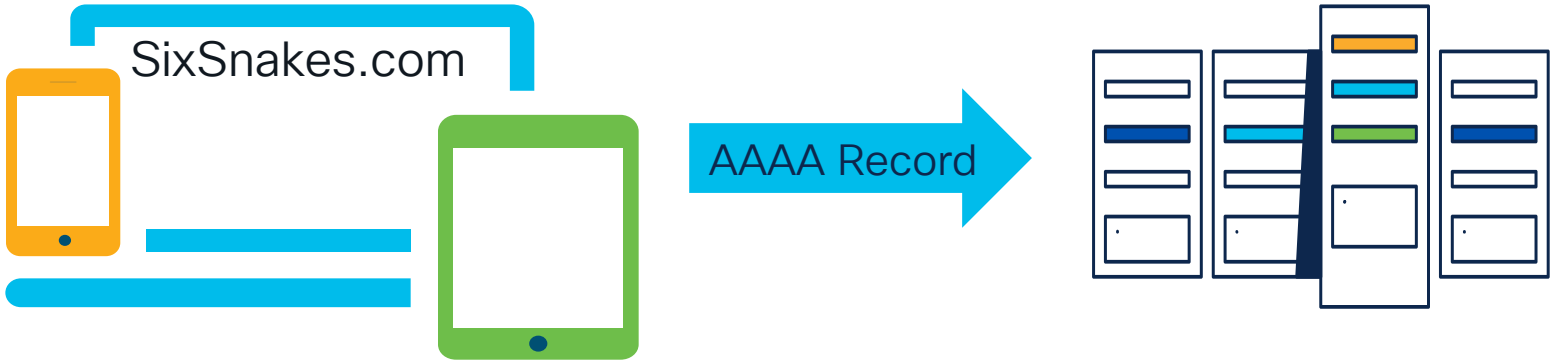
Resolving hosts – the IPv4 way

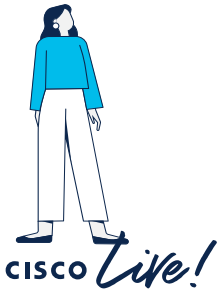


Resolving hosts – the IPv6 way

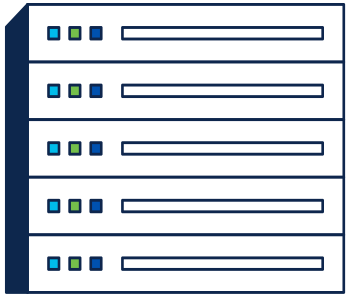


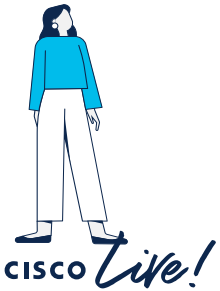
Resolving hosts – the IPv6 way



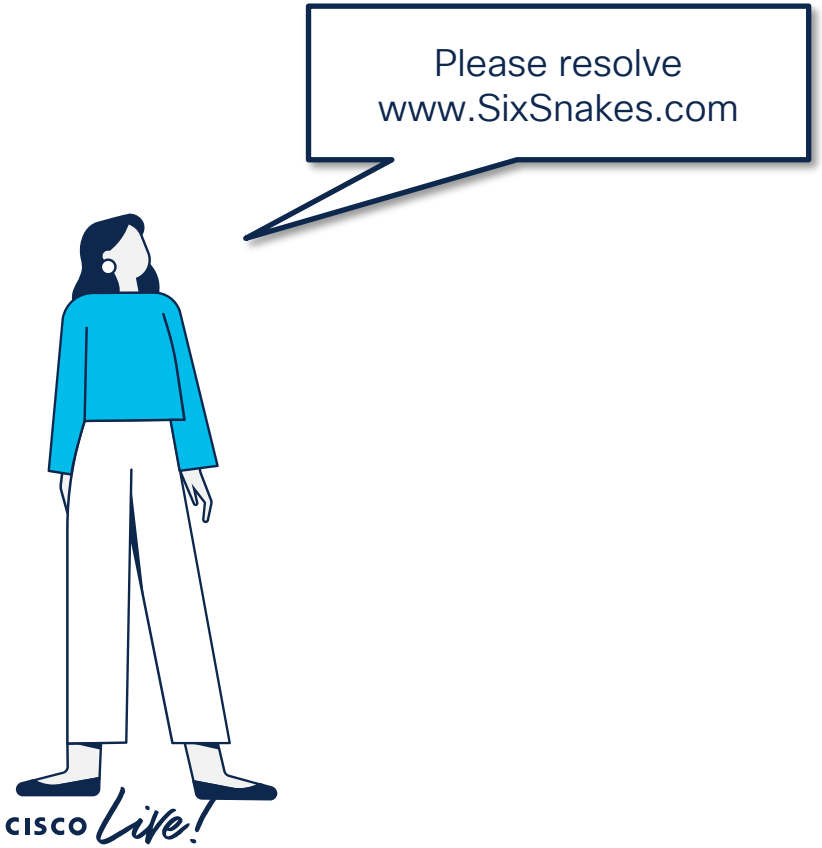


Resolving hosts – the IPv6 way

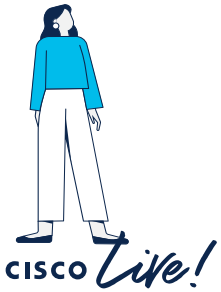




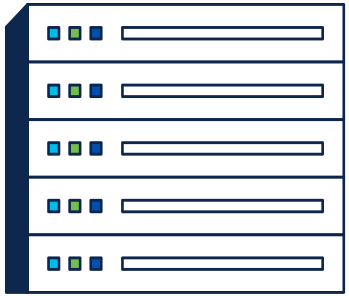
Resolving hosts – Dual stack (IPv4 & IPv6)

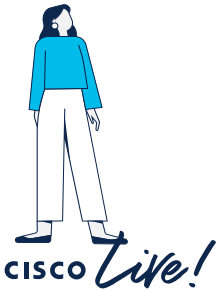


Resolving hosts – Dual stack (IPv4 & IPv6)

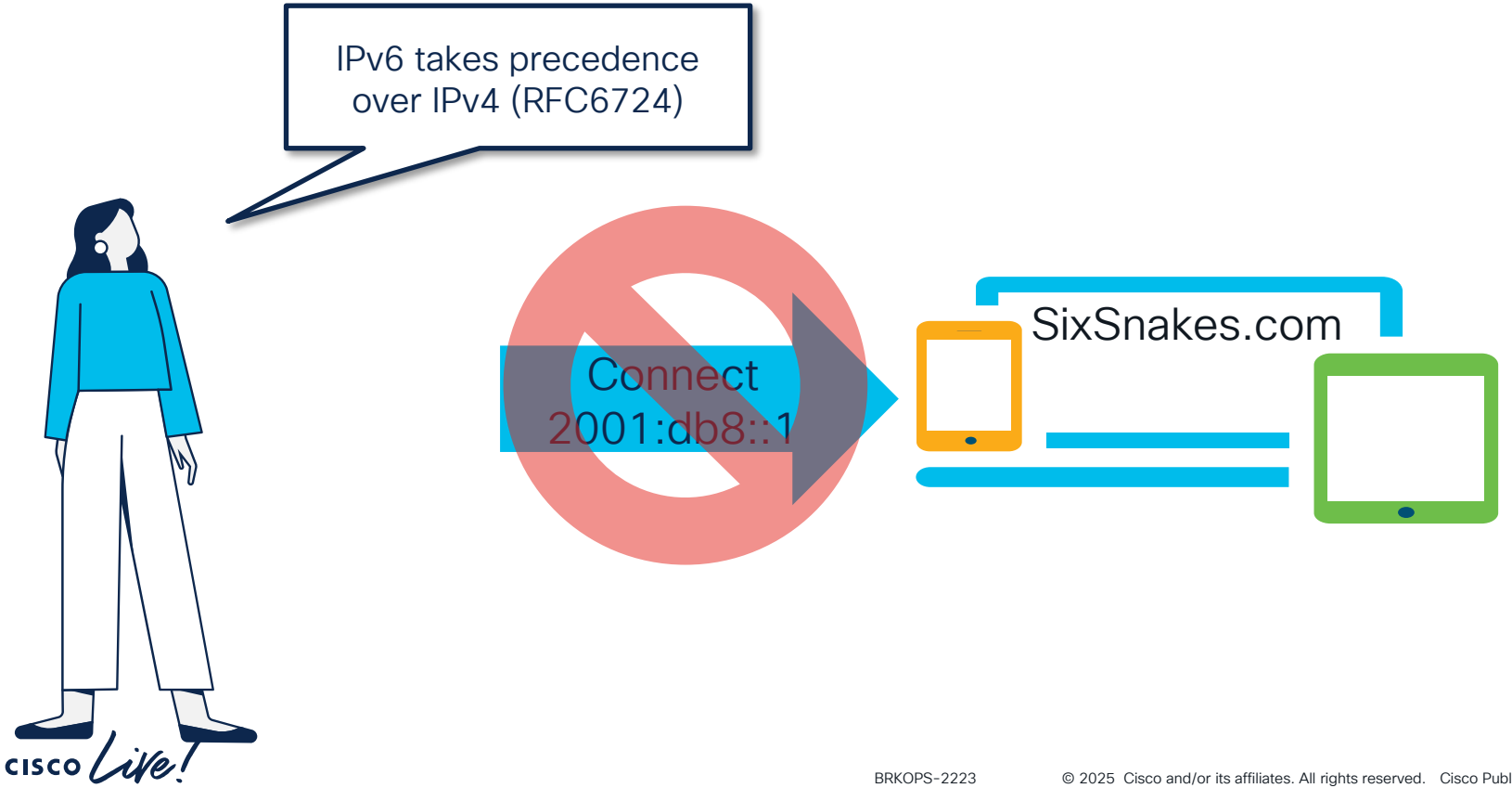


Resolving hosts – Dual stack (IPv4 & IPv6)

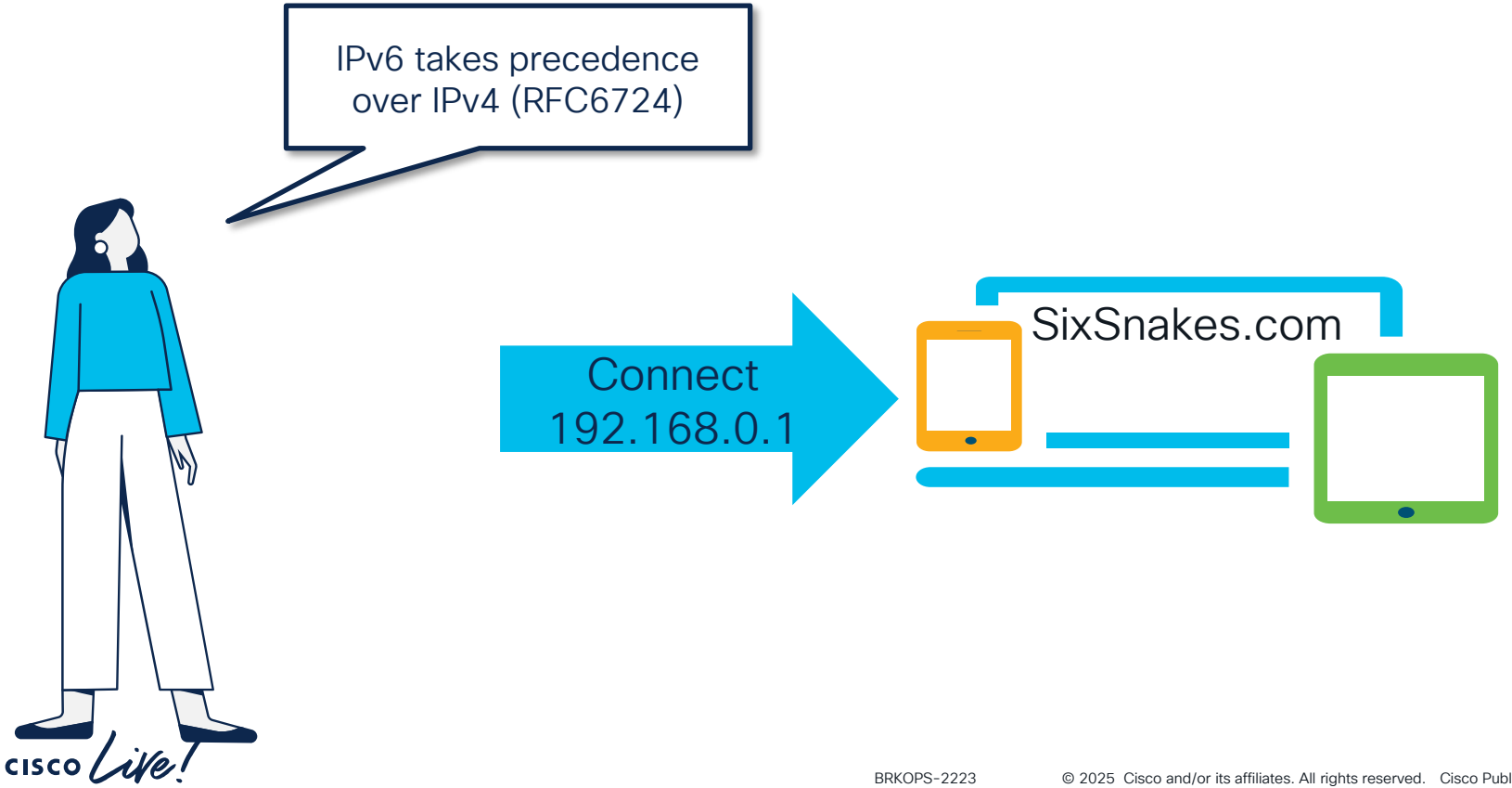




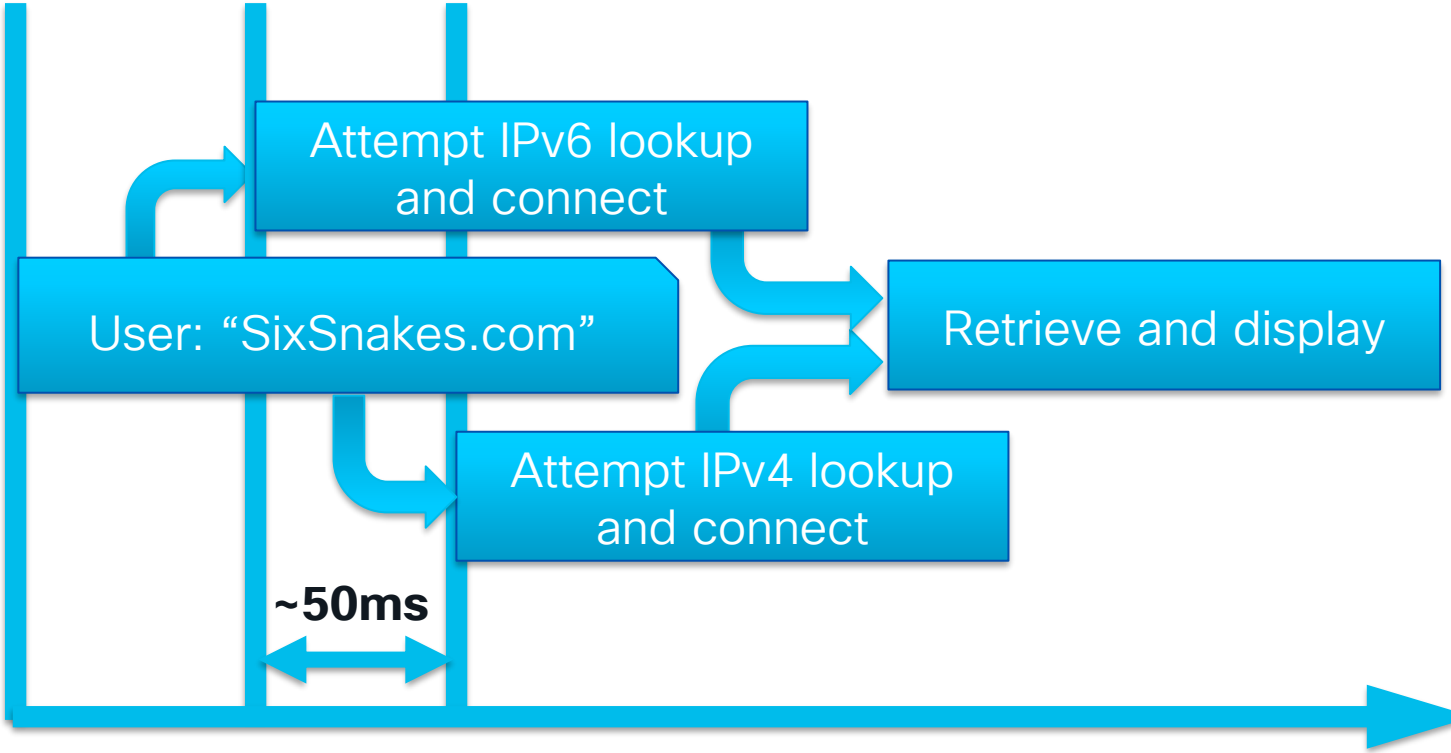
Resolving hosts

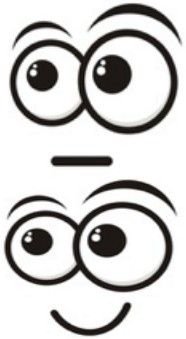


Resolving hosts



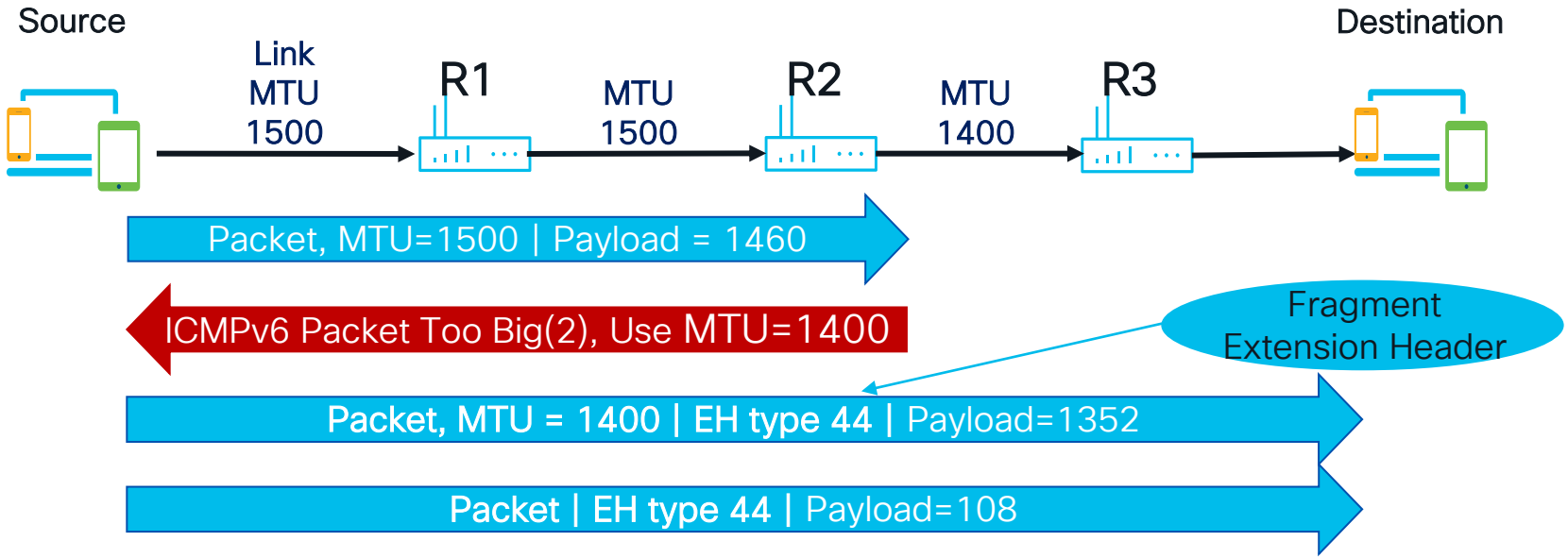
RFC8305 in a nutshell (RFC6555 Obsolete)





Source: ©adynue

Path MTU Discovery

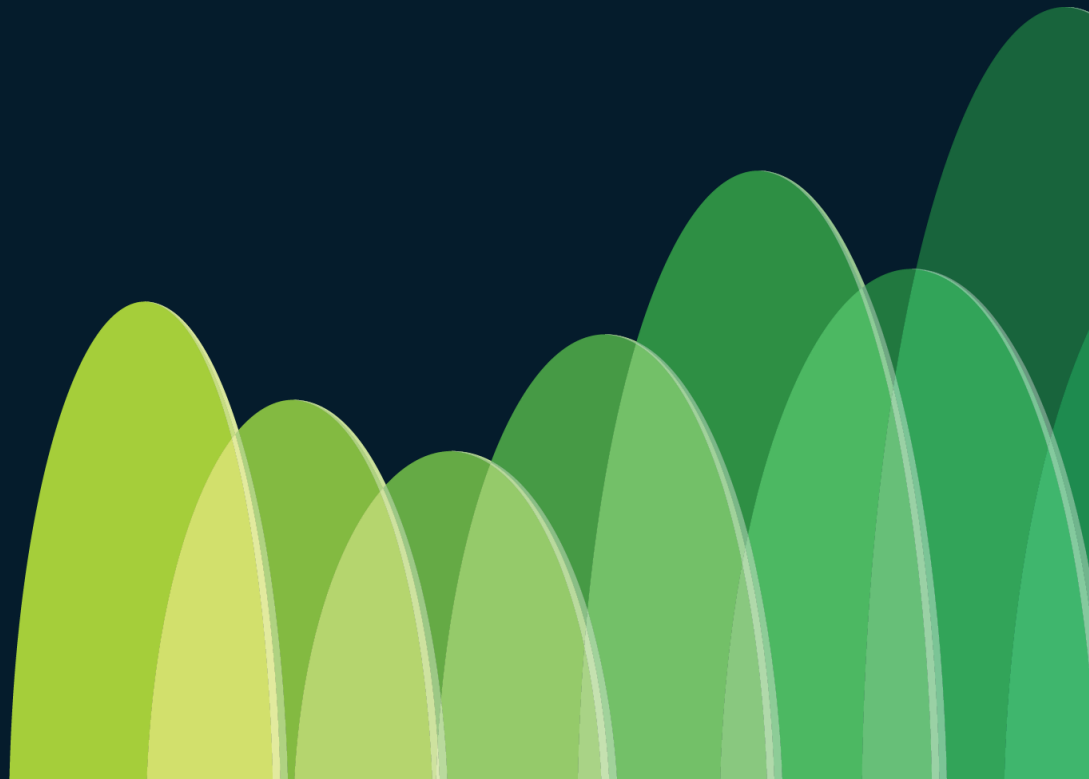


Towards scalable and robust automation

	Source	Interpretation	Configuration	Verification
Minimum viable product (MVP)	 Hardcoded values	 Manual triggering	 May or may not be correct & complete	 None or manual
Network as Code (NaC)	 Source of Truth	 Automated Workflow	 Versioned, complete configuration	 Automated testing

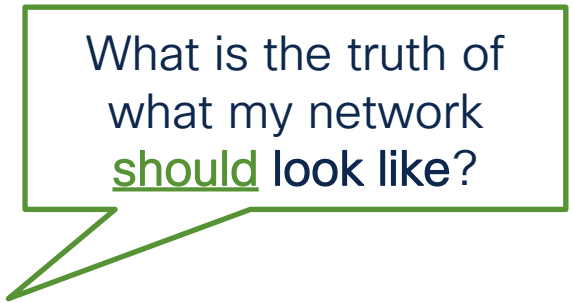
1. Design | 2. Deploy | 3. Validate | 4. Scale

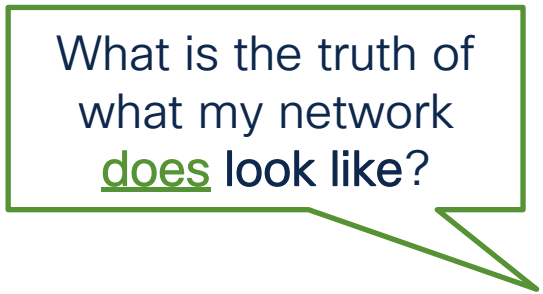
Scale and confirm with Network as Code



cisco Live!

Your network's Source of Truth ≠ Source of data





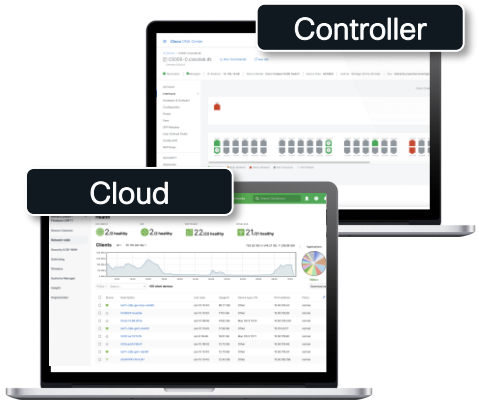
Your network's Source of Truth ≠ Source of data

What is the truth of what my network **should** look like?

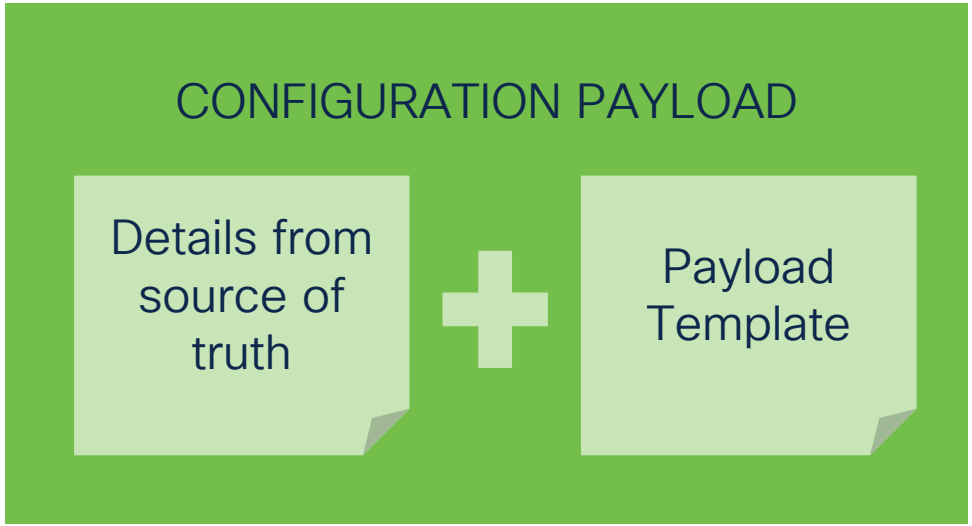
What is the truth of what my network **does** look like?

An up-to-date, version-controlled external file or solution that can be accessed programmatically.

Examples:
- YAML file
- Excel
- NetBox
- Nautobot



Scale your IPv6 configuration



This is a simplified example – see more extensive example in the session's repository

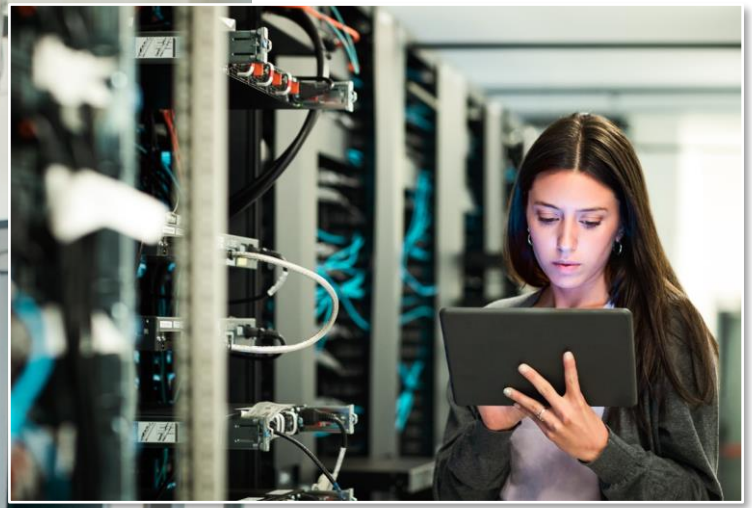
Scale your IPv6 configuration

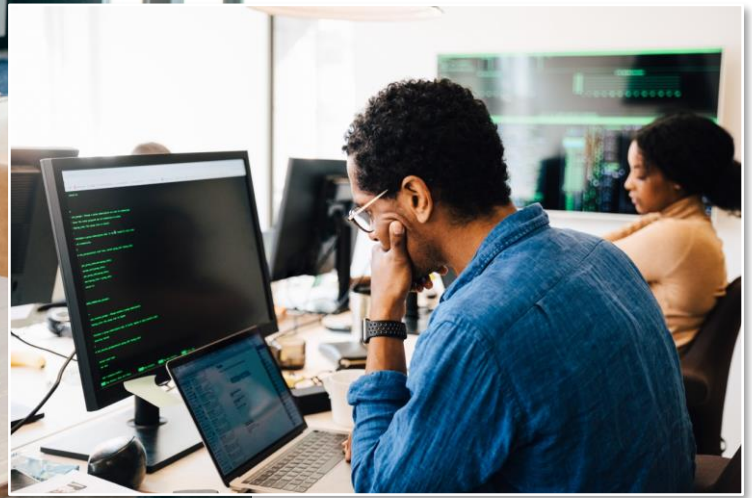
```yaml
R2:
  - type: "GigabitEthernet"
    number: "5"
    description: "TO Finland"
    ospfv3:
      process_id: "1"
      process_area: "0"
      priority: "0"
  - type: "GigabitEthernet"
    number: "6"
    description: "TO Japan"
    ospfv3:
      process_id: "1"
      process_area: "0"
      priority: "0"
```

+

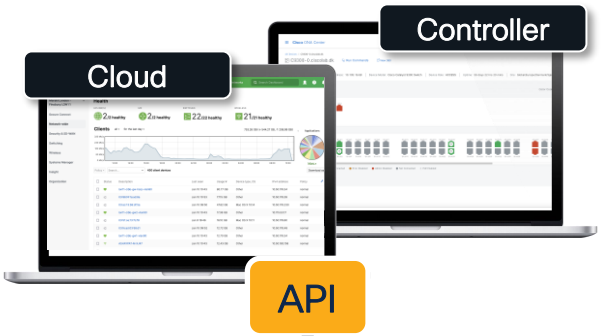
```
{
    "Cisco-IOS-XE-ospfv3:ospfv3": {
        "process": {
            "id": {{ ospfv3.process_id }},
            "area": {{ ospfv3.process_area }}
        },
        "priority": {{ ospfv3.priority }}
    }
}
```

But did it work for real?





Confirm with your network's APIs

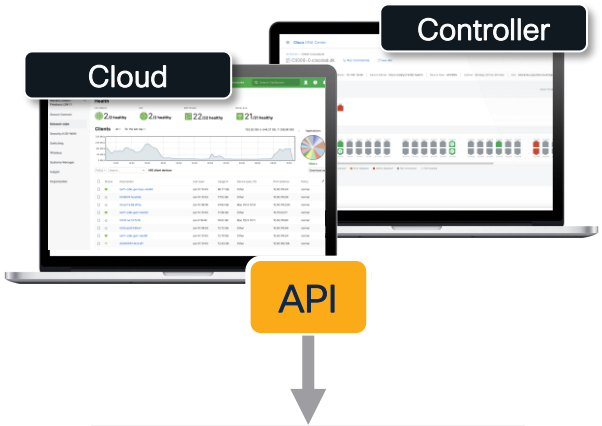


Often **REST** APIs using **URIs** and **HTTP(s)**

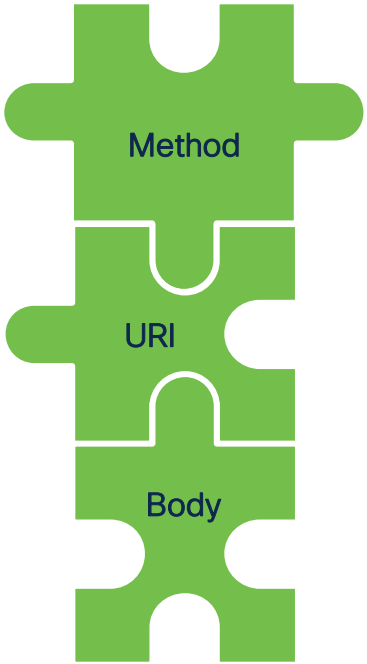
Model Driven Programmability using **YANG** data models

Scripts accessing **normal CLI commands** through SSH

Management platform APIs



Often **REST** APIs using **URIs** and **HTTP(s)**



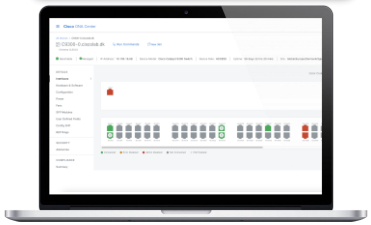
GET POST DELETE
PUT HEAD PATCH

```
https://<host>/dna/
data/api/v1/clients
```

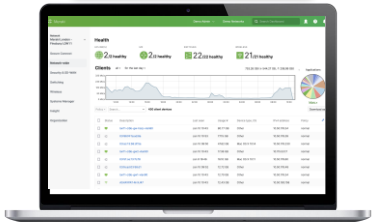
JSON payload based on documentation

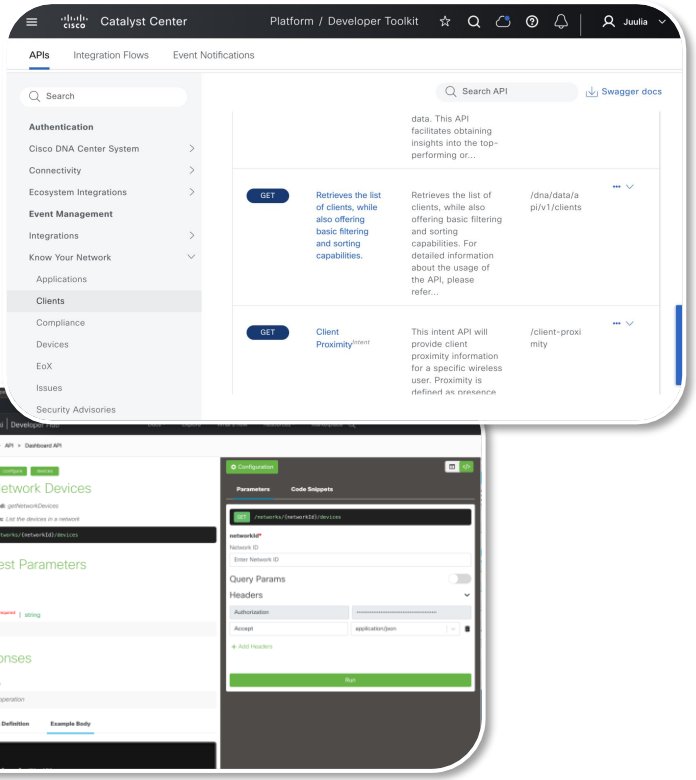
API documentation

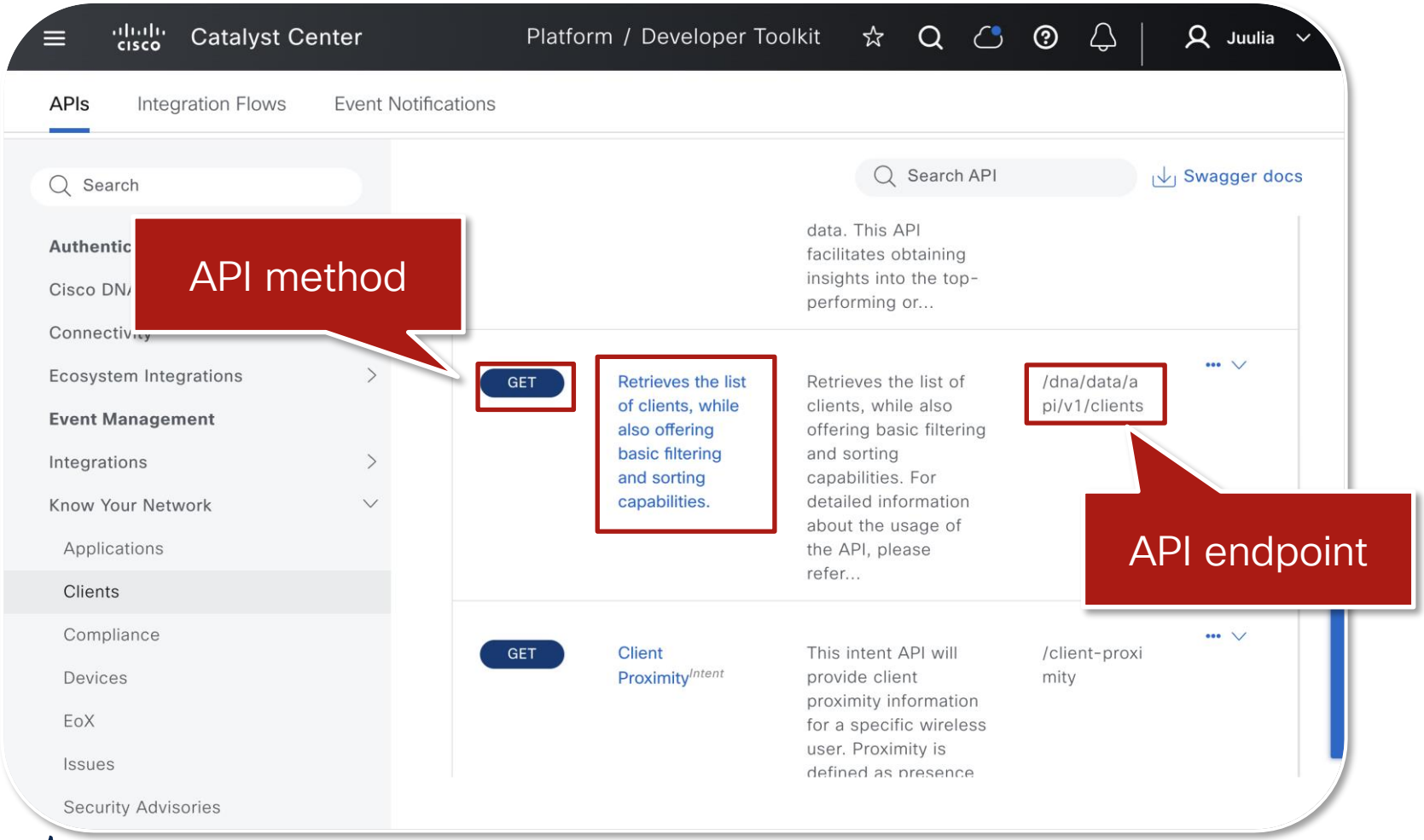
Catalyst Center

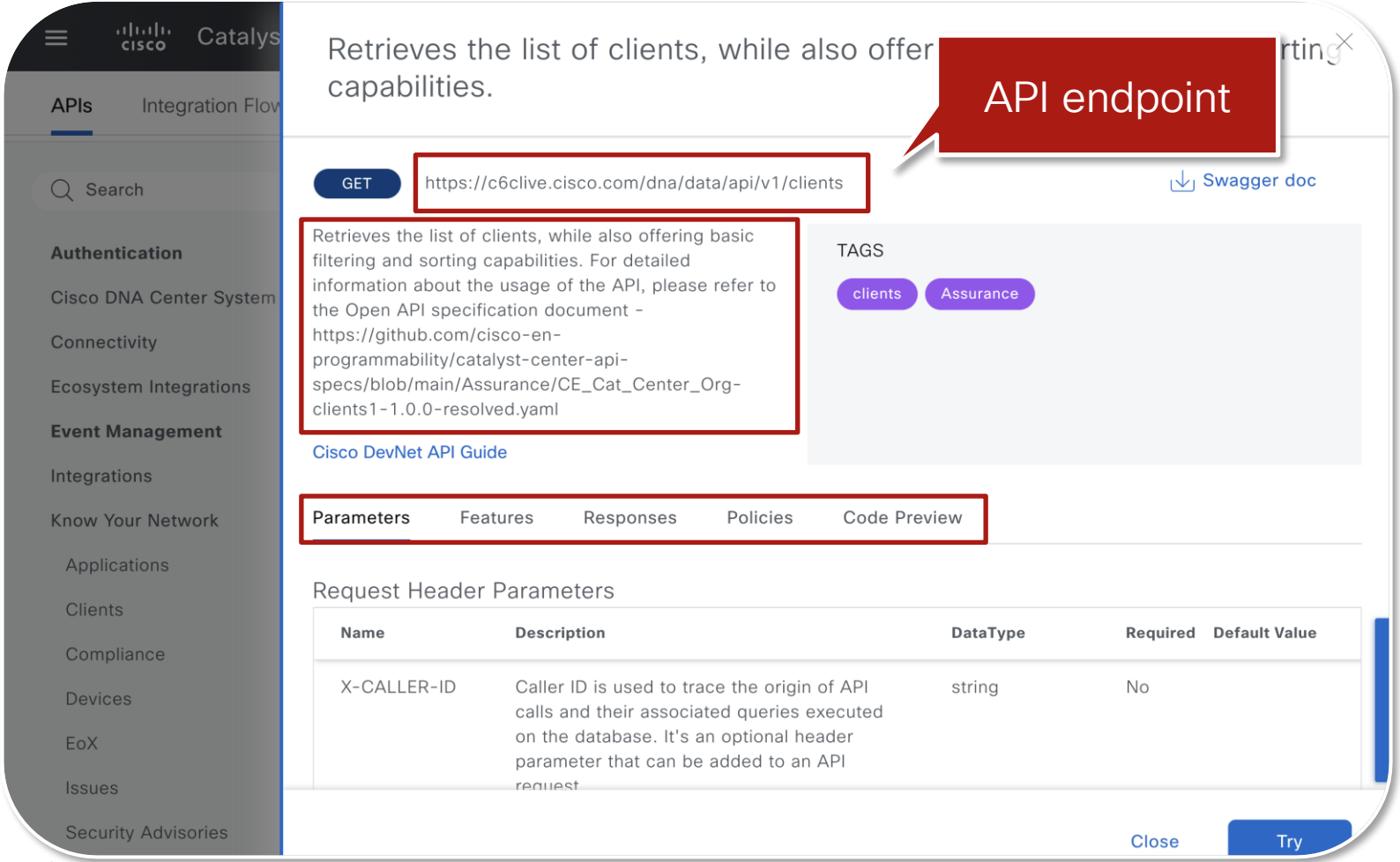


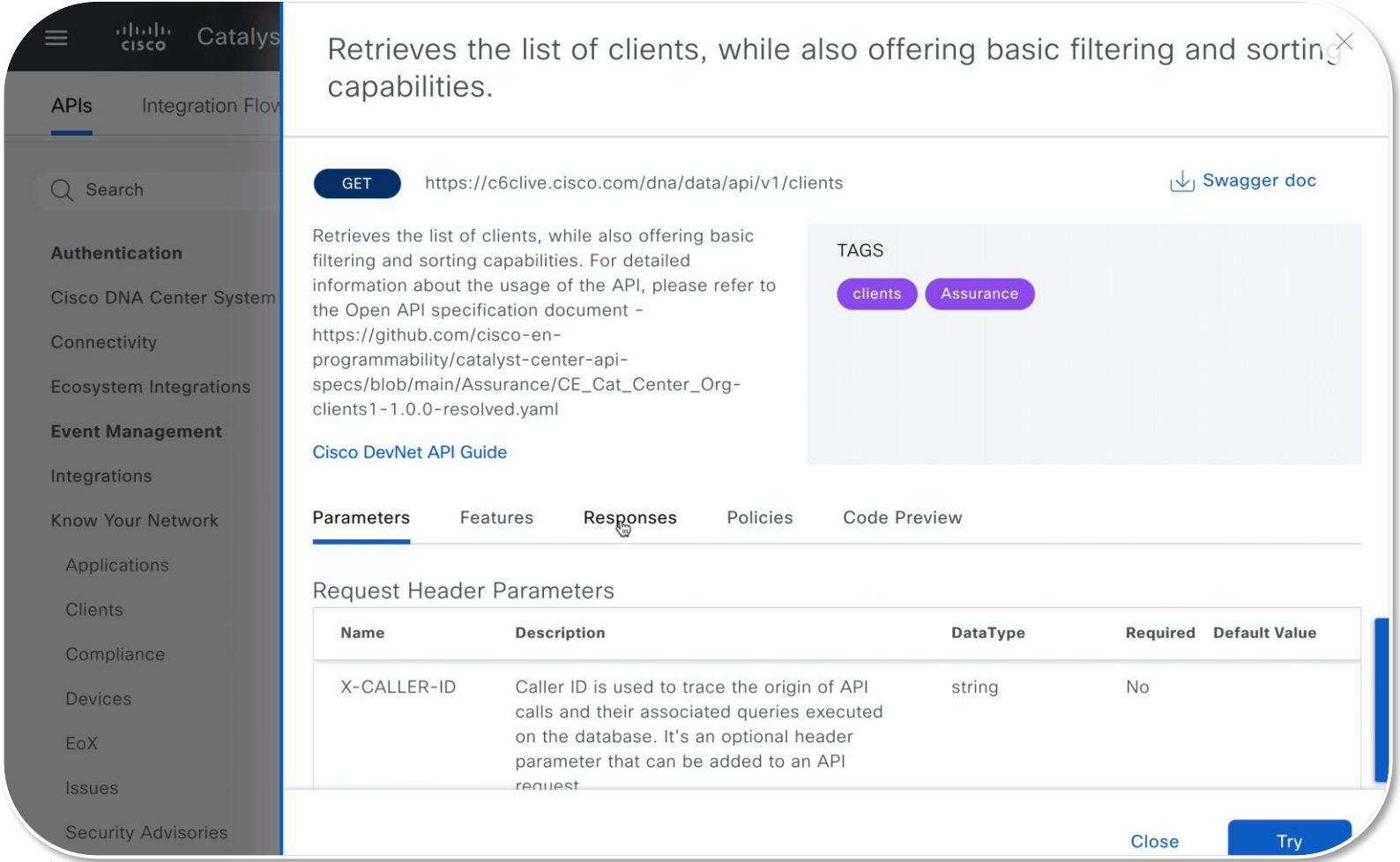
Meraki

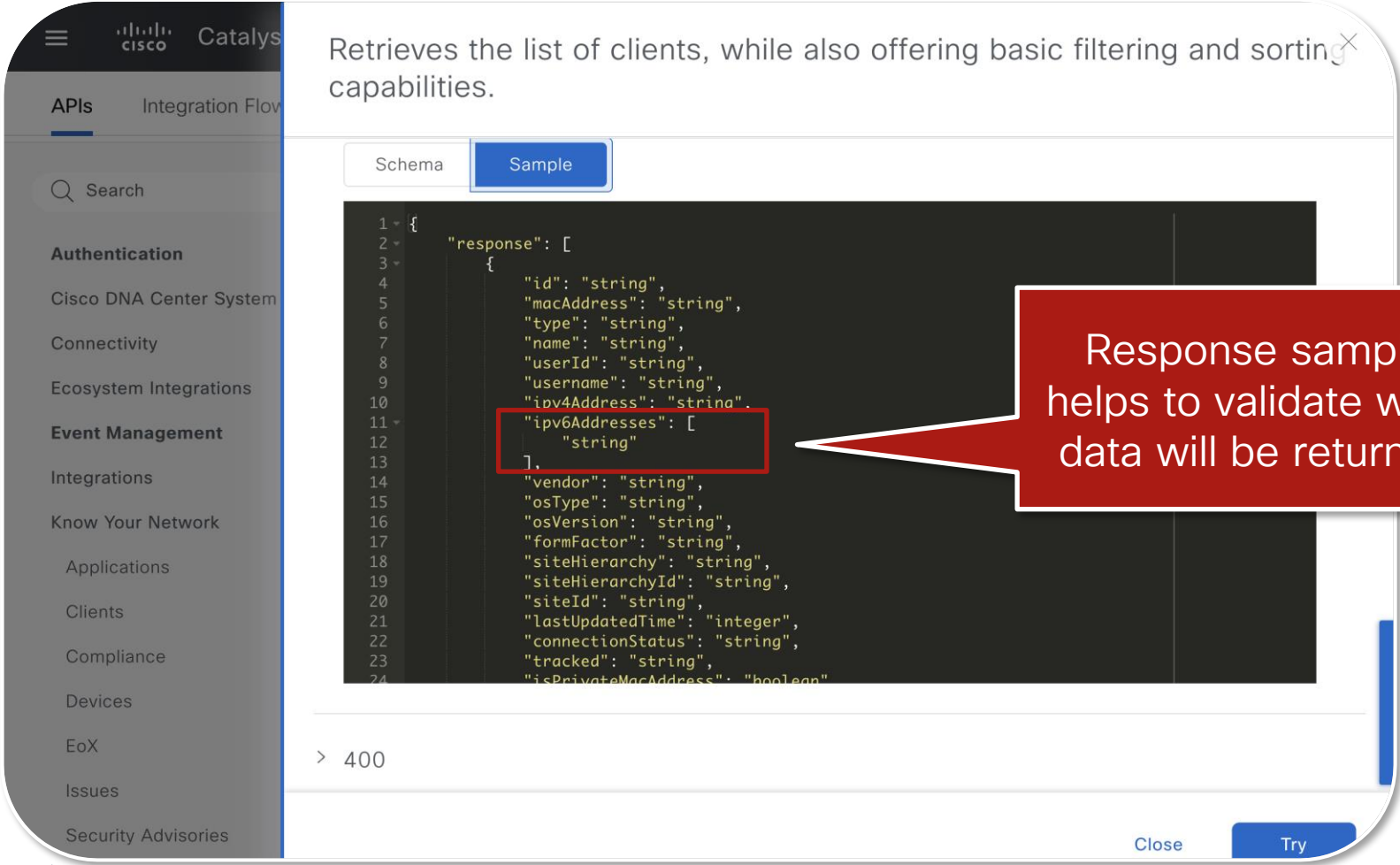




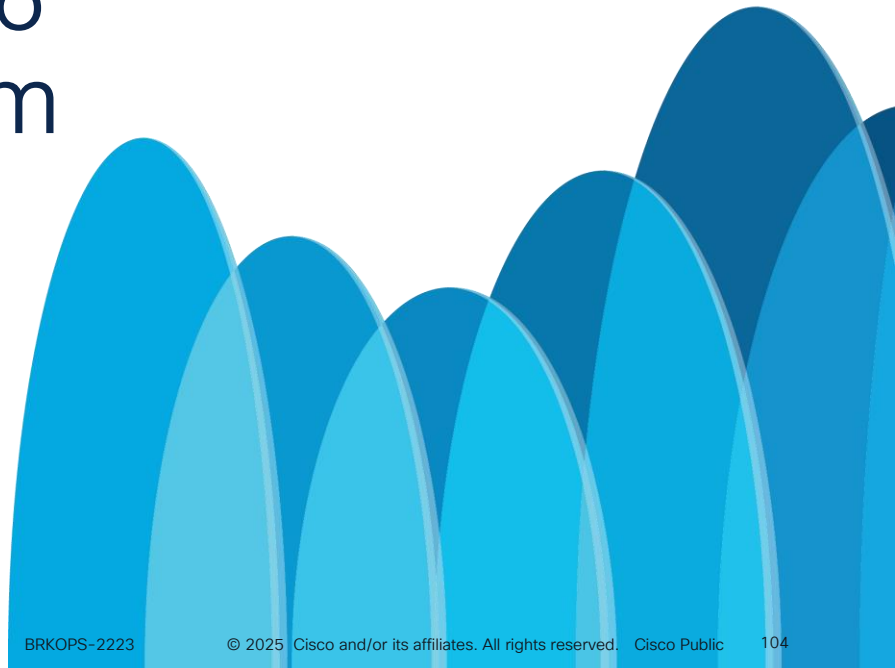








Example: Retrieve IPv6 client information from Catalyst Center



Retrieve IPv6 client info from Catalyst Center

Example with Python library `requests`

1. Authentication

POST `https://<host>/dna/system/api/v1/auth/token`

```
{"Token": "abcd1234…"}
```

2. Retrieve the client data

GET `https://<host>/dna/data/api/v1/clients`

```
{"response": [
    {
        "id": ”abc",
        "macAddress": ”abc",
        "type": "Wired",
     …
}
```

Code:

Retrieve IPv6 client info from Catalyst Center

Example with Python library `requests`

1. Authentication

POST `https://<host>/dna/system/api/v1/auth/token`

2. Retrieve the client data

GET `https://<host>/dna/data/api/v1/clients`

```python
import requests

requests.urllib3.disable_warnings()

HOST = "198.18.129.100"
PASSWORD = "C1sco12345"
USERNAME = "admin"

# Get authentication token

url = f"{HOST}/dna/system/api/v1/auth/token"
response = requests.post(url, auth=(USERNAME, PASSWORD), verify=False)
token = response.json()["Token"]

# Retrieve client data
url = f"{HOST}/dna/data/api/v1/clients"
headers = {"x-auth-token":token}
response = requests.get(url, headers=headers, verify=False)

data = response.json()
for client in data["response"]:
    print(f"Client {client['name']}: IPv6 addresses: {client['ipv6Addresses']}")
```

Code:

Retrieve IPv6 client info from Catalyst Center

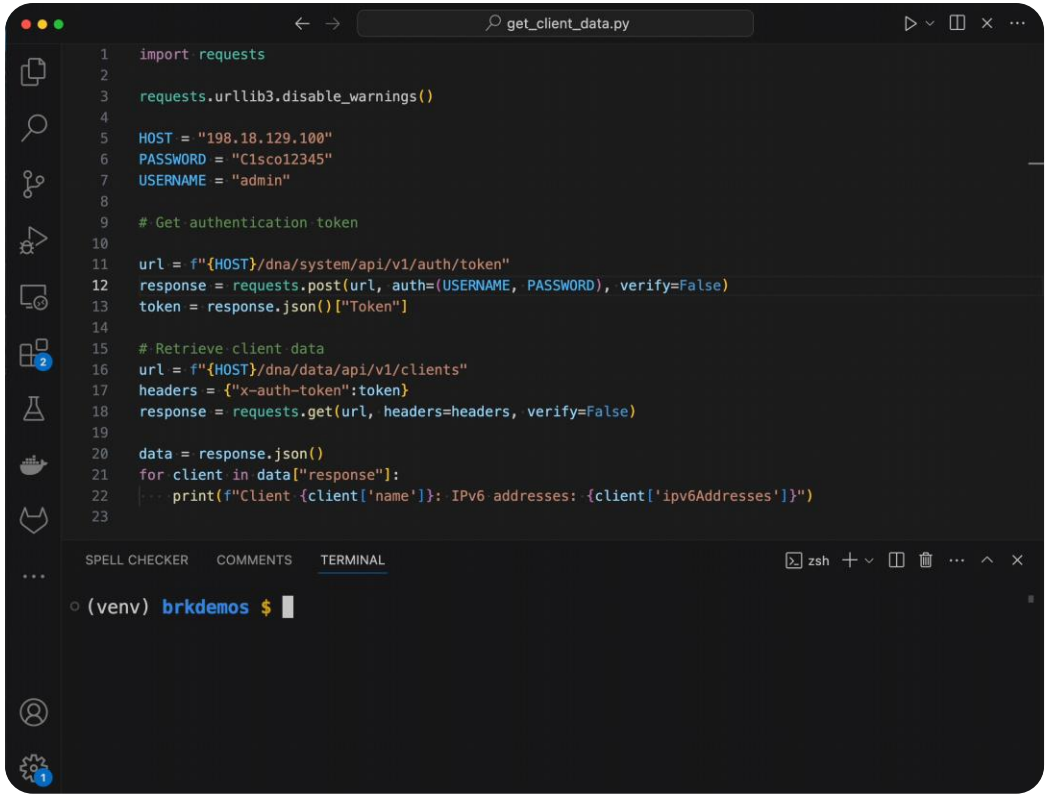
Example with Python library `requests`

1. Authentication

POST `https://<host>/dna/system/api/v1/auth/token`

2. Retrieve the client data

GET `https://<host>/dna/data/api/v1/clients`

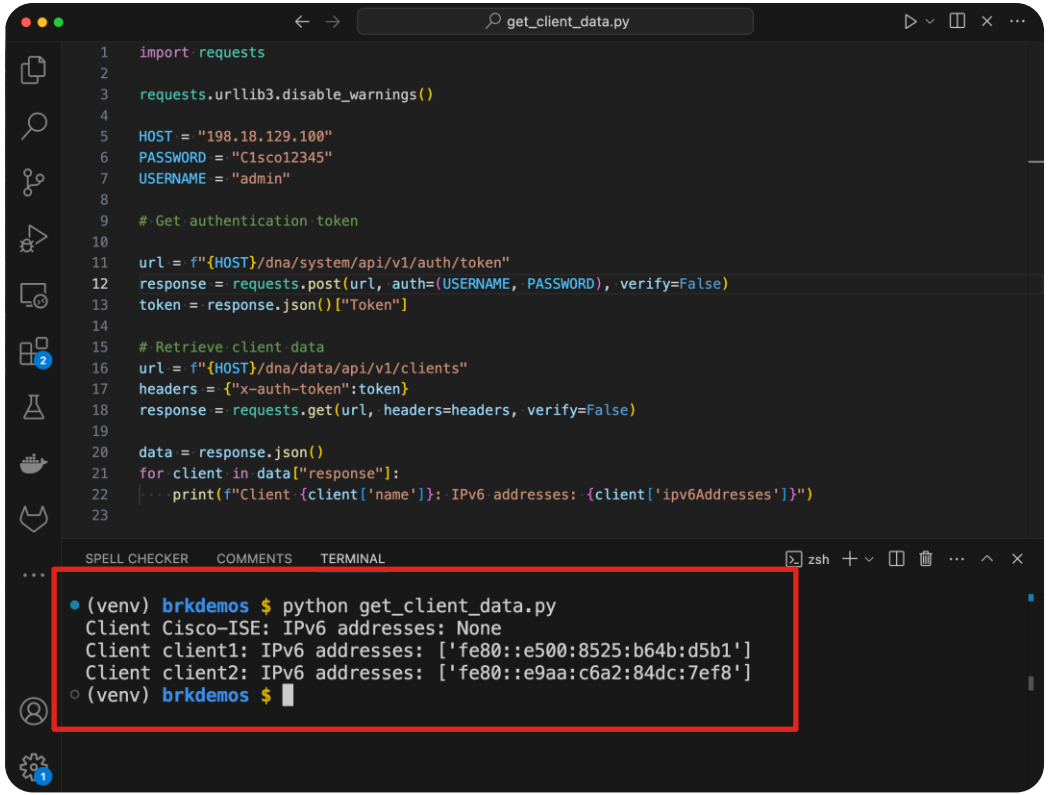


Code:

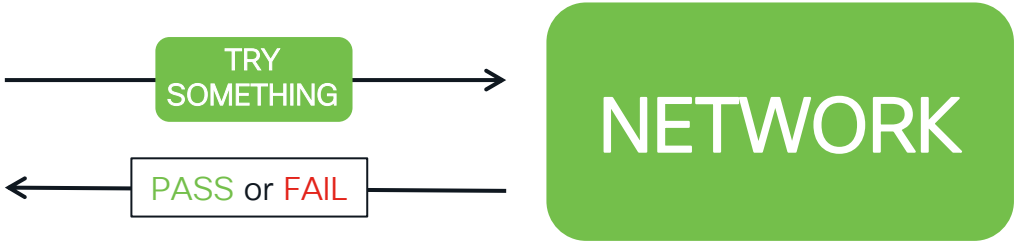
Retrieve IPv6 client info from Catalyst Center

Example with Python library `requests`

You have just retrieved information and decided whether there is an issue or not...

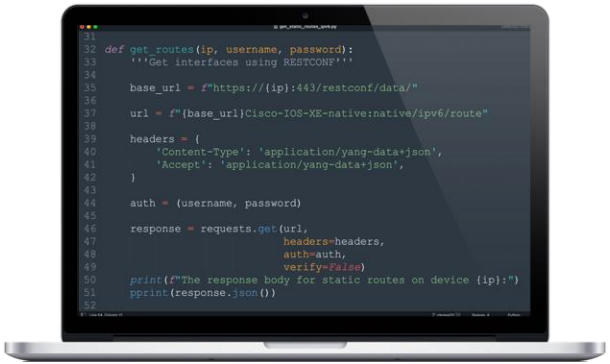


… you have done a test with Catalyst Center!



Tests can be automated

Testcase




```
response = requests.get(url)
```

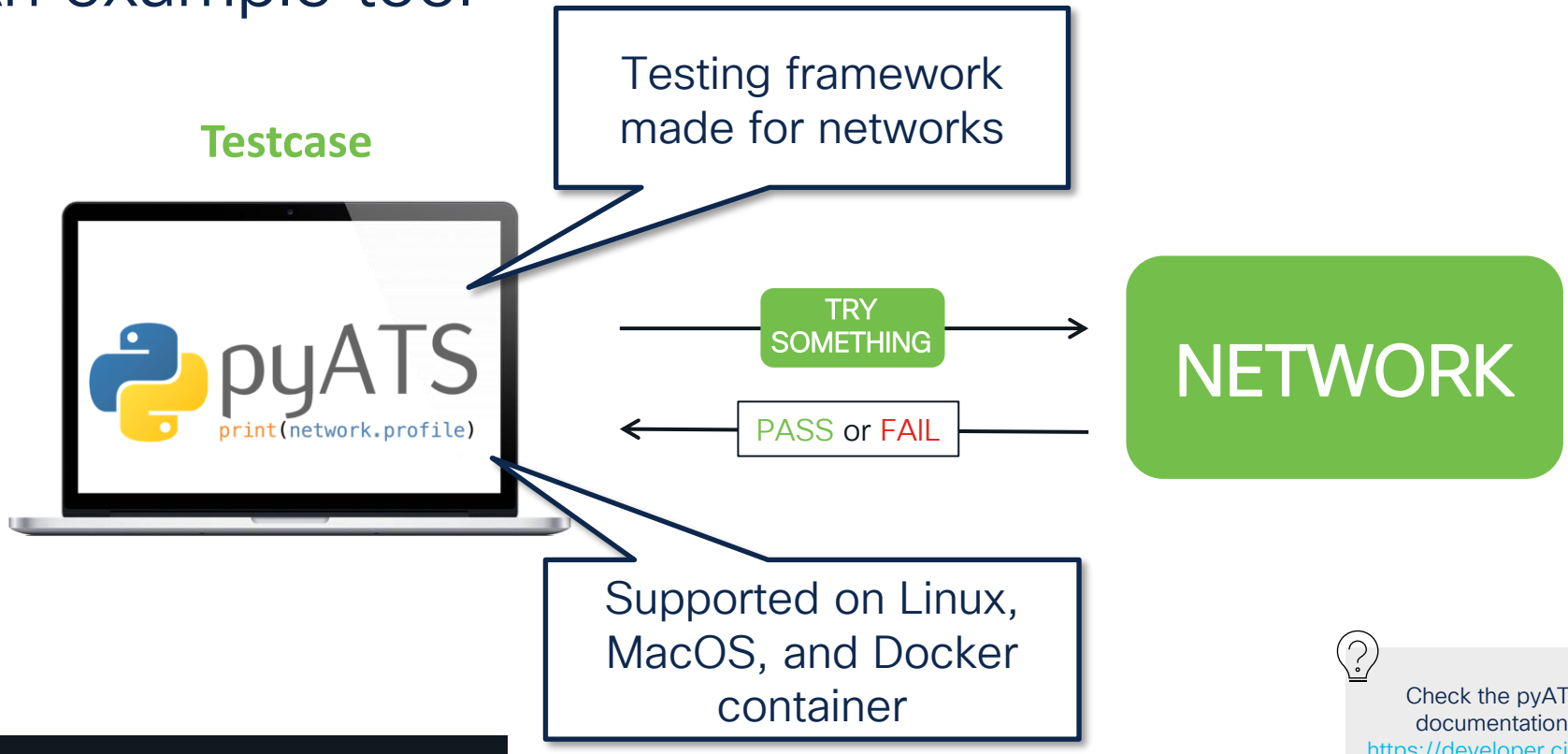
TRY SOMETHING →

NETWORK

← PASS or FAIL

```
step.failed(f'Client {name}: No IPv6')
```

An example tool




```
Freely downloadable!
$ pip install pyats[full]
```

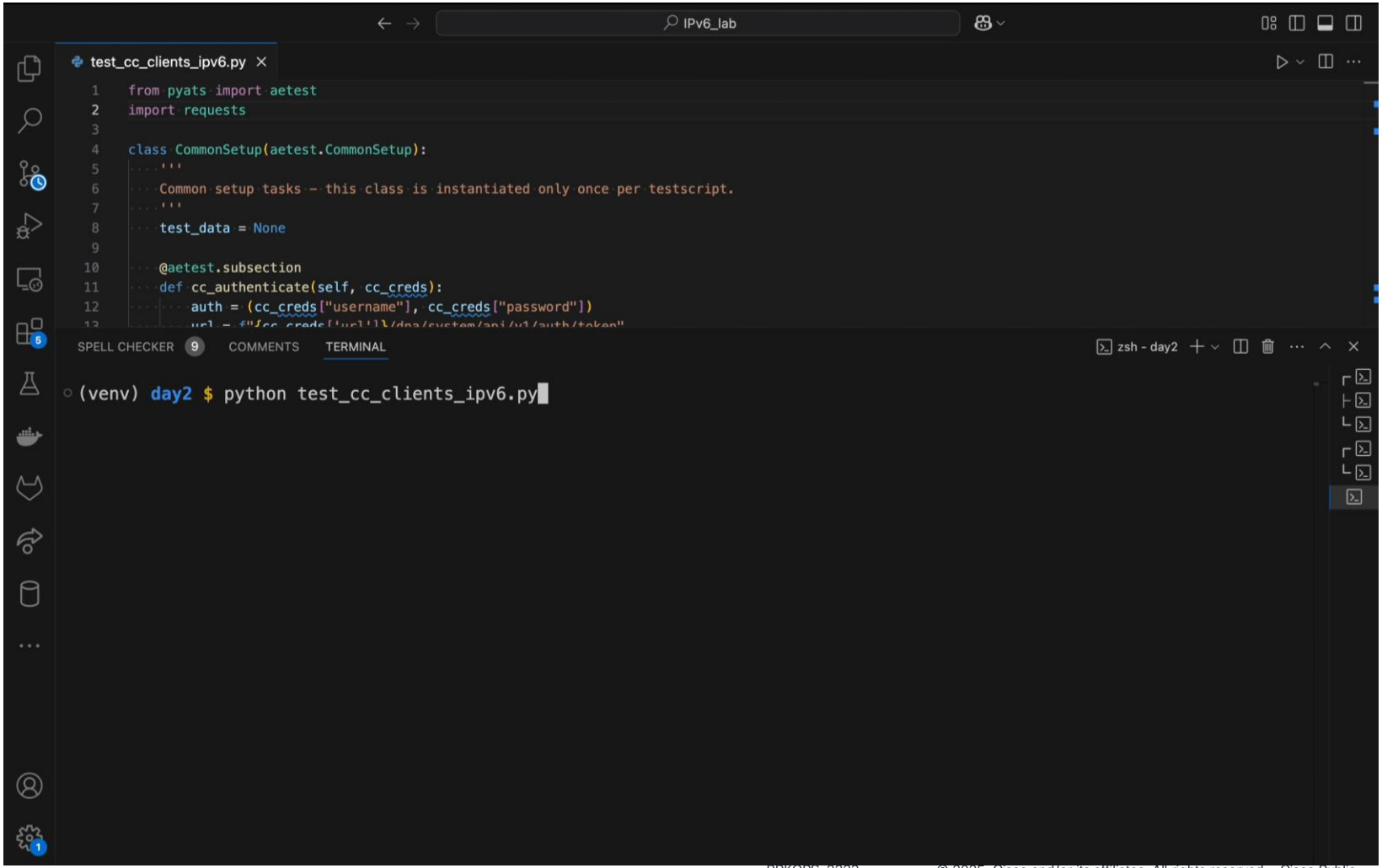
Check the pyATS documentation: https://developer.cisco.com/pyats/

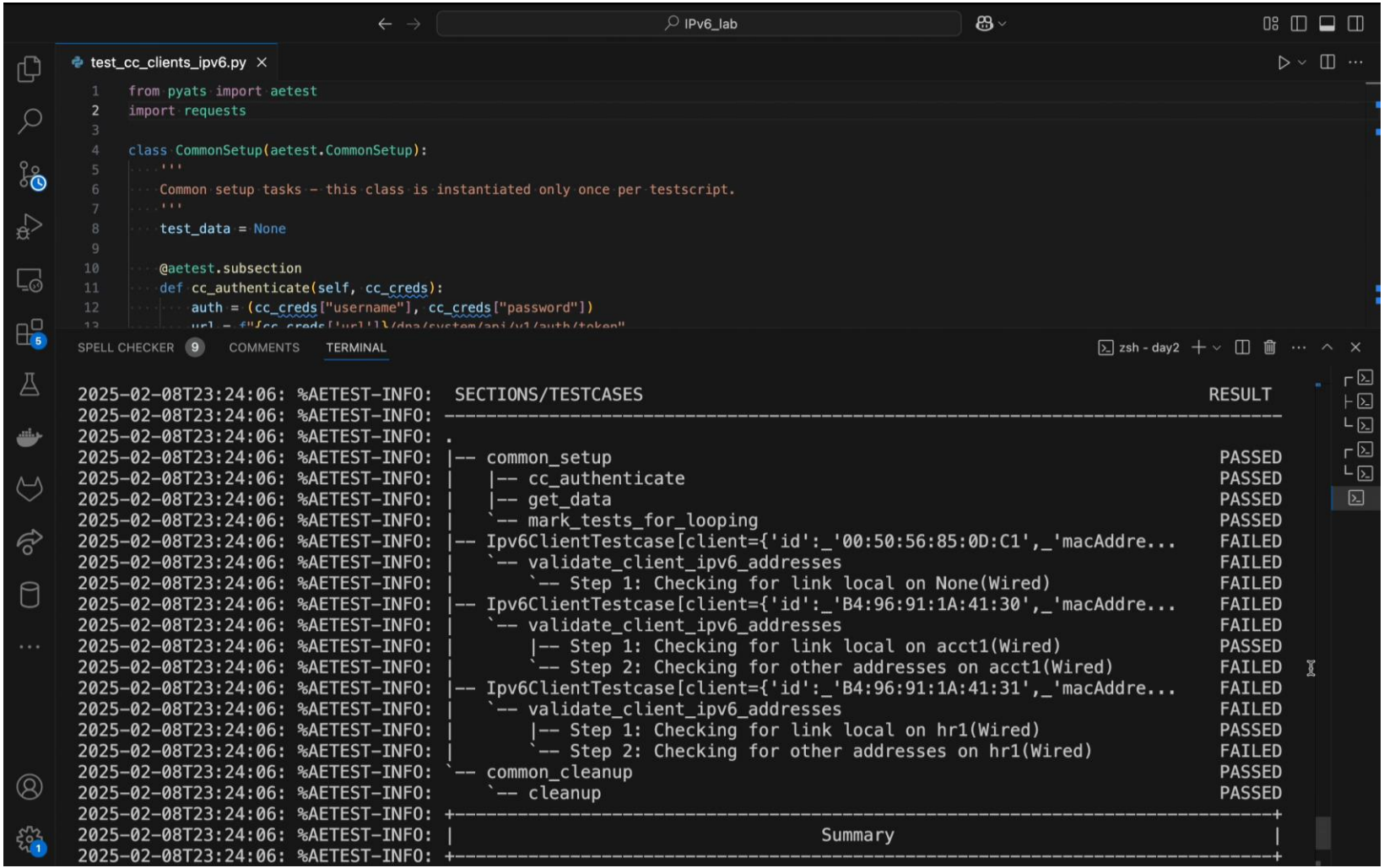
Recorded Demo

Running a test with pyATS and Catalyst Center APIs

cisco Live!

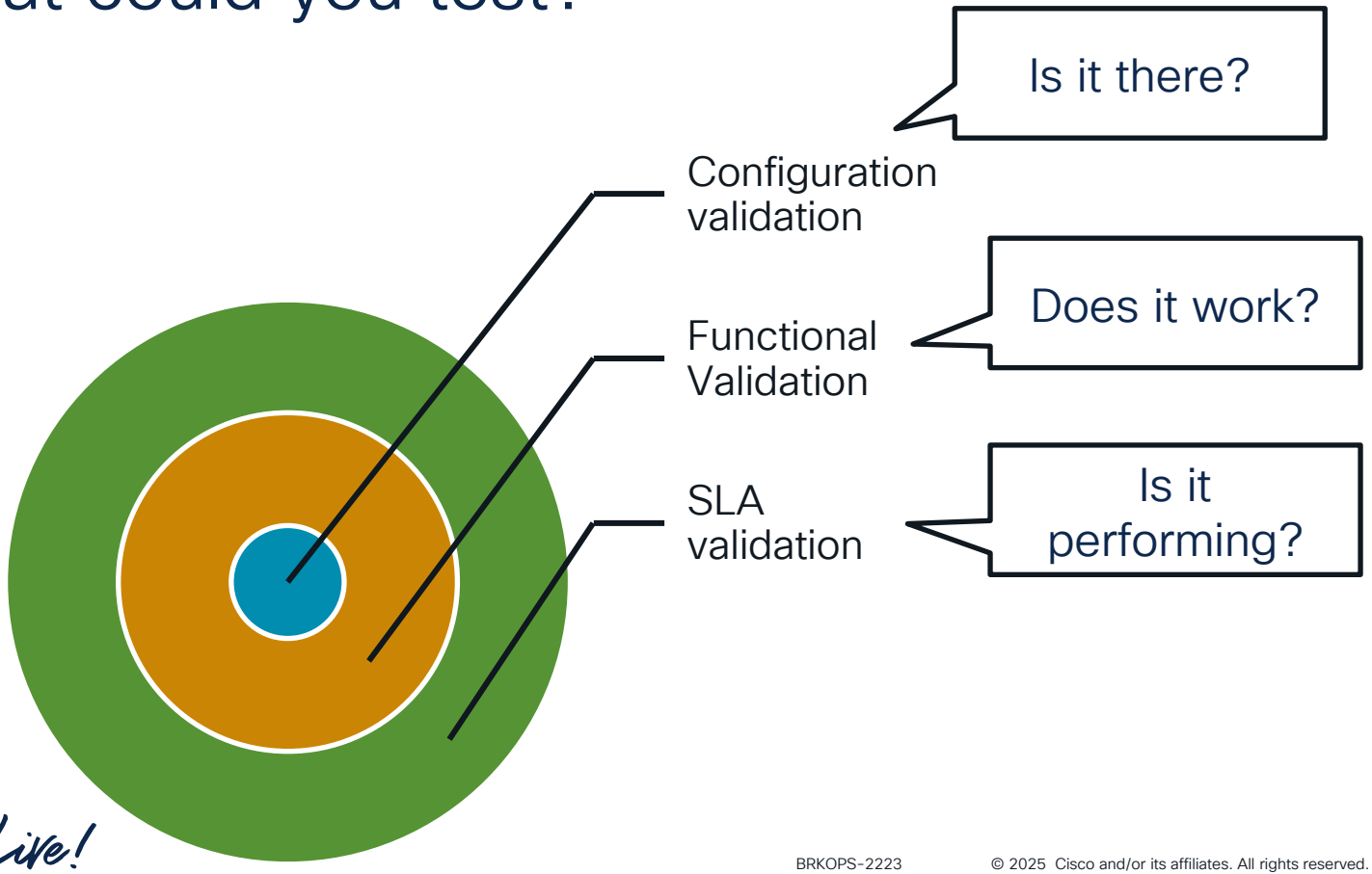
Code:





Code:

What could you test?



Towards scalable and robust automation

	Source	Interpretation	Configuration	Verification
Minimum viable product (MVP)	 Hardcoded values	 Manual triggering	 May or may not be correct & complete	 None or manual
Network as Code (NaC)	 Source of Truth	 Automated Workflow	 Versioned, complete configuration	 Automated testing

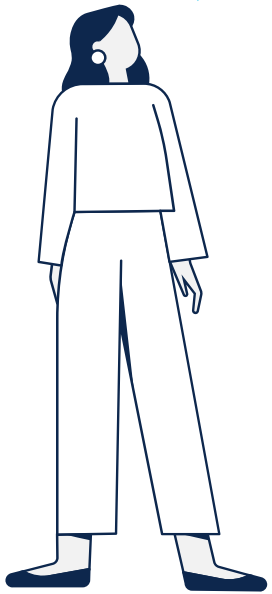
All the code is available for you!

Clone the session's repo:
https://github.com/juuliasantala/BRKOPS-2223

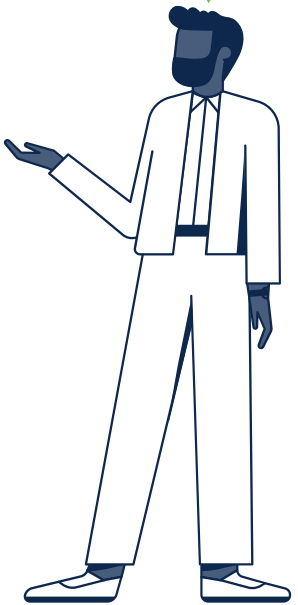
cisco Live!

When planning for IPv6, plan it in an automated way

Python gives a lot of opportunities deploy, monitor and test!



Let's deploy IPv6 with the help of your network's APIs!



Webex App

Questions?

Use the Webex app to chat with the speaker after the session

How

1. Find this session in the Cisco Events 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 February 28, 2025.

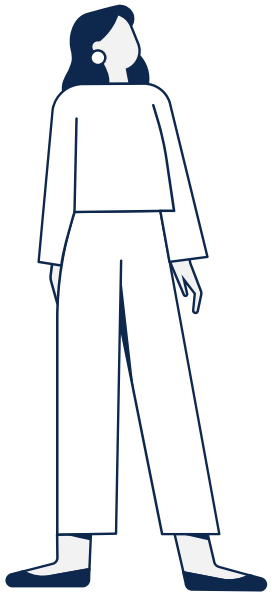
Fill Out Your Session Surveys

Participants who fill out a minimum of 4 session surveys and the overall event survey will get a unique Cisco Live t-shirt.

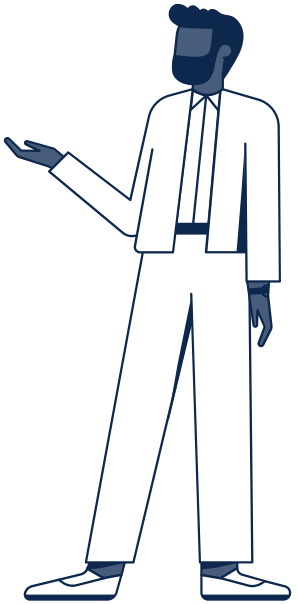
(from 11:30 on Thursday, while supplies last)

All surveys can be taken in the Cisco Events mobile app or by logging in to the Session Catalog and clicking the ‘Participant Dashboard’

Content Catalog

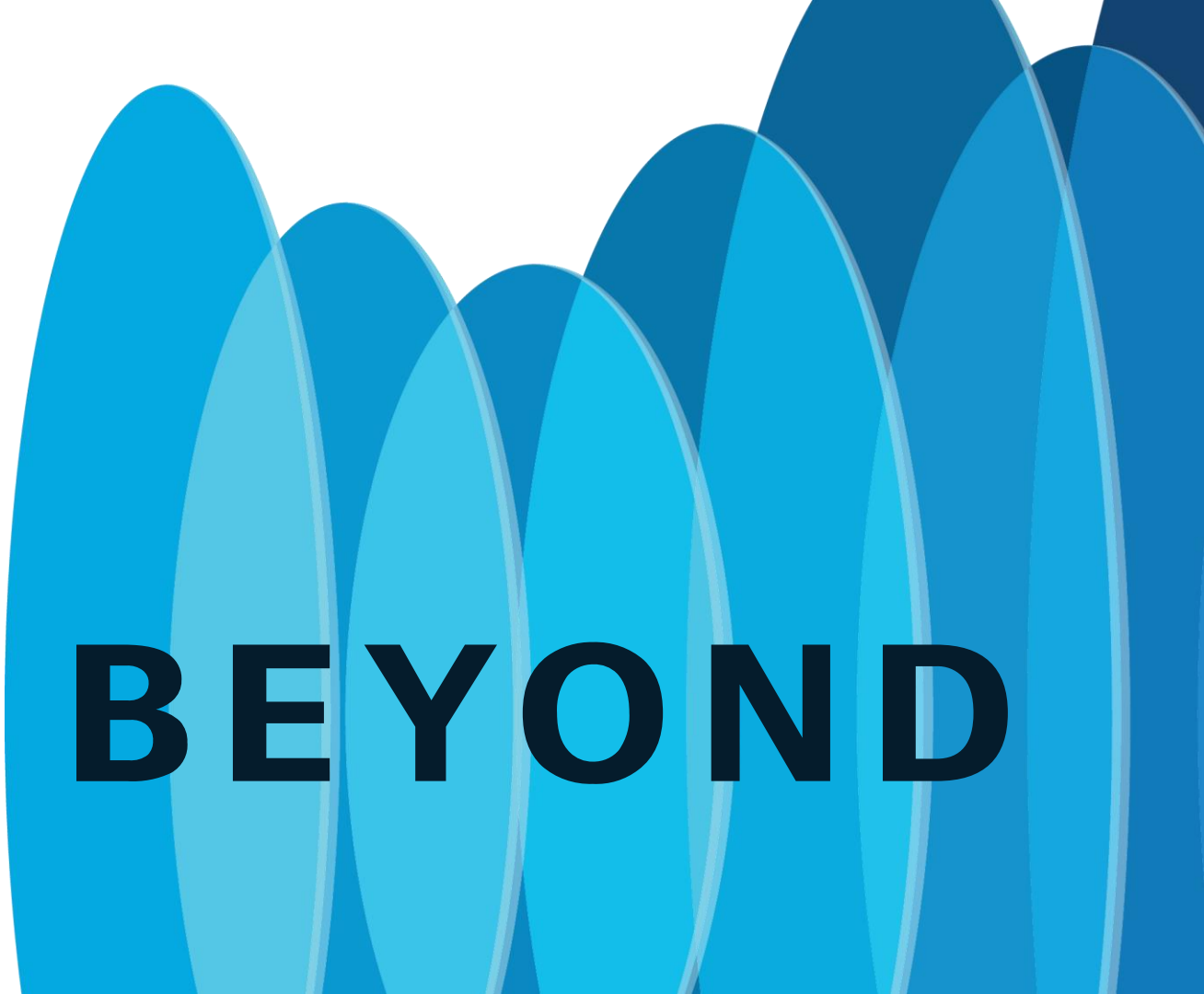


Let's deploy IPv6 with the help of your network's APIs!



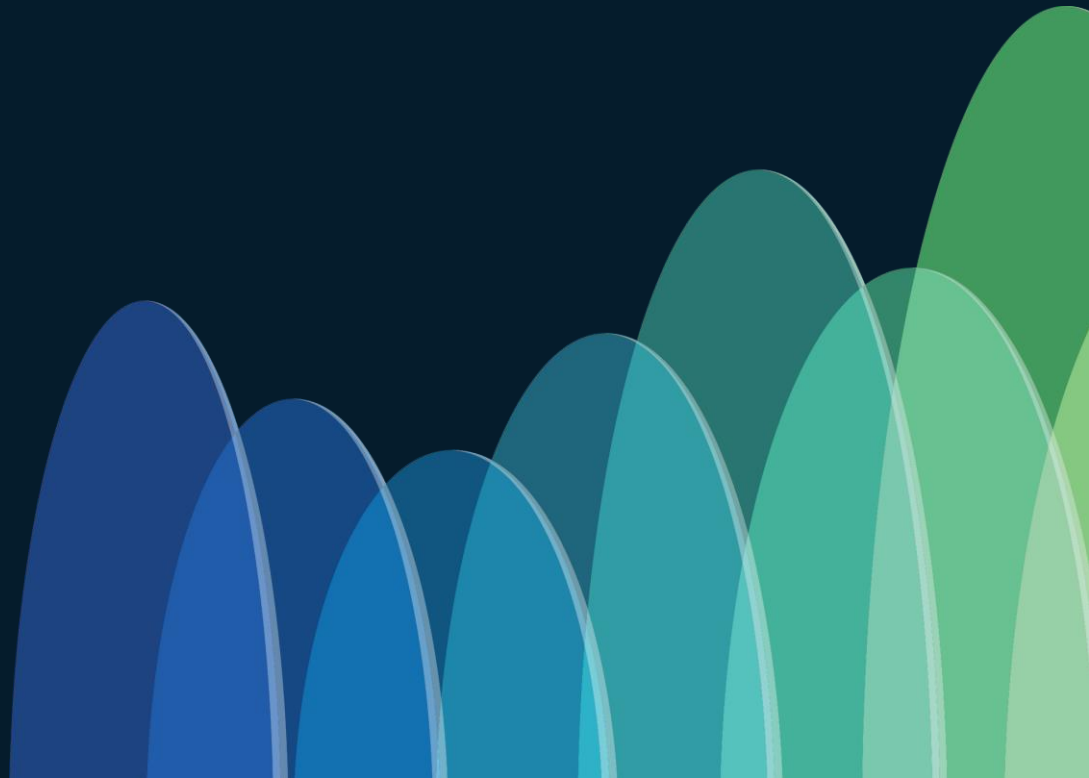

cisco
Thank you
cisco Live!

cisco Live!



GO BEYOND

Resources



Cisco Live Amsterdam 2025 IPv6 Learning Map

Sunday–9th

TECIPV-2000 13:45
IPv6 in the Host and in the Local Network

Monday–10th

TECENT-2150 8:30
6+3=100! Use IPv6 and Python 3 to Transform how you do Networking

TECIPV-2001 8:45
IPv6 Beyond the Local Network

Tuesday–11th

BRKIPV-1007 8:00
Deploying Catalyst Center for IPv6 Networks

BRKSEC-2044 10:30
Secure Operations for an IPv6 Network

IBOIPV-2000 13:30
Sharing Experience on IPv6 Deployments

BRKSPG-2203 14:30
Introduction to SRv6 uSID Technology

BRKIPV-2191 16:30
IPv6:: It's Happening!

Wednesday–12th

BRKEWN-2834 8:00
IPv6-Enabled Wireless (Wi-Fi) Access: Design and Deployment Strategies

CTF-1001 10:15
IPv6: The Internet's best kept secret!

BRKIPV-2186 13:15
IPv6 Networking in a Cloud Native World

CISCOU-1038 14:45
IPv6 Groove: Get By with a Little Help from My Friends!

BRKENT-2008 13:00
Goodbye Legacy, the Move to an IPv6-Only Enterprise

BRKIPV-1616 16:00
IPv6 – What Do You Mean There Isn't a Broadcast?

Thursday–13th

IBOENT-2811 11:30
Everything You Wanted to Know about IPv6 but Were Afraid to Ask

IBOIPV-2000 13:30
Sharing Experience on IPv6 Deployments

BRKSPG-3198 14:15
Advanced Innovations in SRv6 uSID and IP Measurements

BRKOPS-2223 15:00
The Network of the Future is Here - Let's Automate your IPv6 deployment with Python!

BRKIPV-2228 17:00
The Automation Travel Guide for Your IPv6 Journey!

Friday –14th

BRKIPV-2418 9:00
Deploying IPv6 Routing Protocols: Specifics and Considerations

BRKENT-3340 11:00
The Hitchhiker's Guide to Troubleshooting IPv6

BRKENT-3002 11:15
IPv6 Security in the Local Area with First Hop Security

Walk in Labs

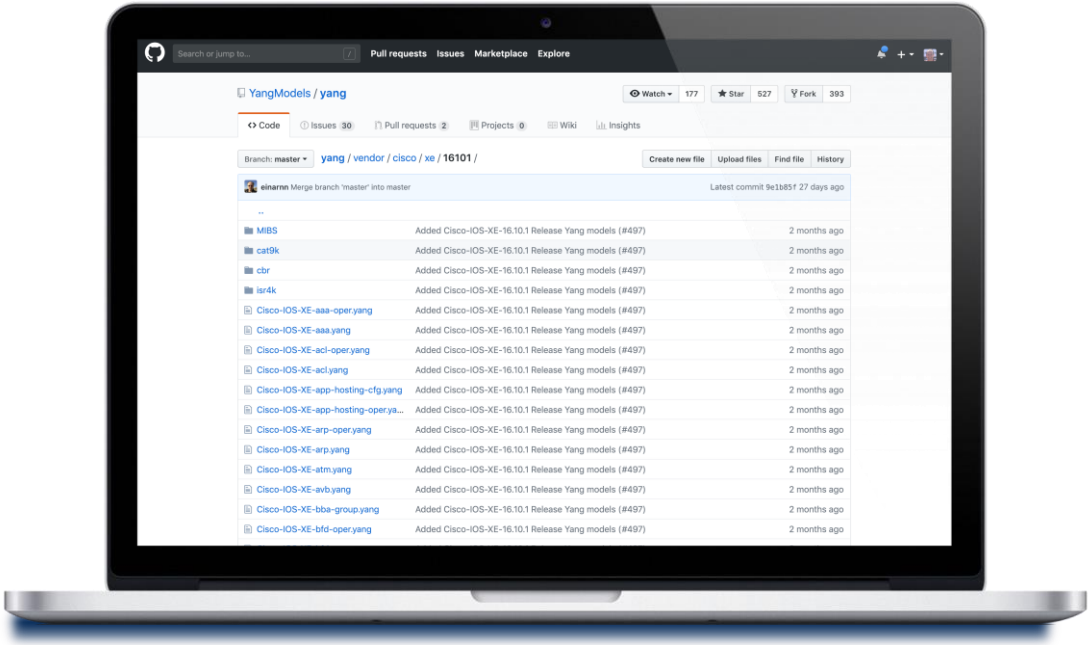
LABIPV-1639 IPv6 Foundations: A Dive into Basic Networking Concepts
LABIPV-2640 IPv6 Deep Dive: Beyond Basics to Brilliance
LABMPL-1201 SRv6 Basics
LABSP-2129 SRv6 Micro-Segment Basics
LABSP-3393 Implementing Segment Routing v6 (SRv6) Transport on NCS 55xx/5xx and Cisco 8000: Advanced

Instructor-led Labs

LTRIPV-2222 Implementing Future-Ready Networks - Deploy IOS XE IPv6 Configuration with Python!
LTRSPG-2212 SRv6 and Cloud-Native: A Platform for Network Service Innovation

cisco Live!

YANG – A data modeling language



IOS XE YANG MODELS:
https://github.com/YangModels/yang/tree/master/vendor/cisco/xe

IETF YANG CATALOG:
https://yangcatalog.org/

CISCO YANG SUITE:
https://developer.cisco.com/yangsuite/

DevNet learning material

- Developer sandboxes: https://developer.cisco.com/site/sandbox/
- Learning labs: https://developer.cisco.com/learning/
  - DevNet learning lab: Introduction to YANG Suite
  - DevNet learning lab: Developer Workstation and Environment Setup
  - DevNet learning lab: Coding & APIs
  - DevNet learning lab: Introduction to Model Driven Programmability
- pyATS documentation

Training Options

- HexaBuild
  - https://hexabuild.arlo.co/w/
- Pluralsight
  - https://www.pluralsight.com/courses/ipv6-introduction-to-protocol
  - https://www.pluralsight.com/courses/ipv6-microsoft-windows
- NterOne
  - https://www.nterone.com/training/cisco/courses/ip6fd
- O'Reilly LiveLessons
  - https://www.oreilly.com/videos/ipv6-design-and/9780134655529
- Rick Graziani – YouTube Playlist
  - https://www.youtube.com/playlist?list=PLMLm7-g0V0kfGg8g8KutNFK7rS3laA9QQ

Cisco Live! (On-Demand)

IPv6:: It's Happening! – BRKIPV-2191 – Nathan Sherrard

https://www.ciscolive.com/c/dam/r/ciscolive/global-event/docs/2023/pdf/BRKIPV-2191.pdf

What Do you Mean there isn't a Broadcast? – BRKIPV-1616 – Fish Fishburne

https://www.ciscolive.com/c/dam/r/ciscolive/global-event/docs/2023/pdf/BRKIPV-1616.pdf

Deploying IPv6 in the Cloud – BRKIPV-3927 – Shannon McFarland

https://www.ciscolive.com/c/dam/r/ciscolive/global-event/docs/2023/pdf/BRKIPV-3927.pdf

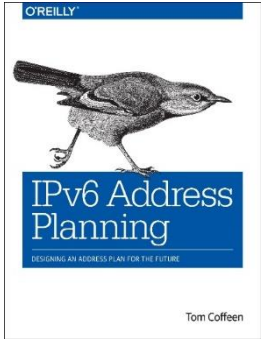
The Hitchhiker's Guide to Troubleshooting IPv6 - BRKENT-3340 – Nicole Wajer

https://www.ciscolive.com/c/dam/r/ciscolive/emea/docs/2024/pdf/BRKENT-3340.pdf

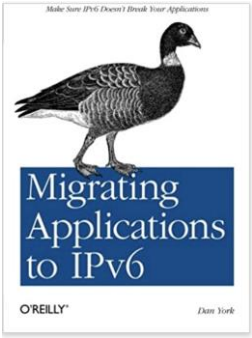
Resources - Books

https://www.amazon.com/IPv6-Fundamentals-Straightforward-Approach-Understanding/dp/1587144778

https://www.oreilly.com/library/view/dns-and-bind/9781449308025/



https://www.amazon.com/IPv6-Address-Planning-Designing-Future/dp/1491902760



https://www.oreilly.com/library/view/migrating-applications-to/9781449309688/

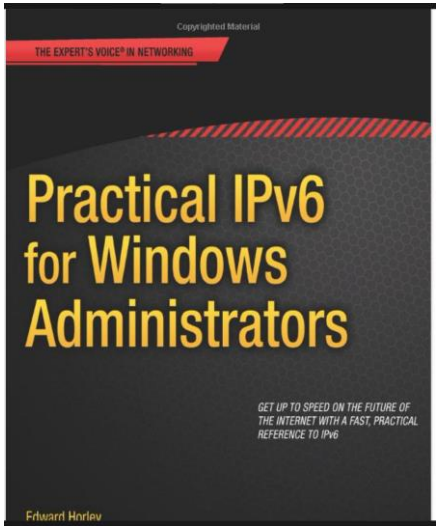
Additional Books:

Cisco Press - https://www.ciscopress.com/store/ipv6-design-and-deployment-livelessons-9780134655512

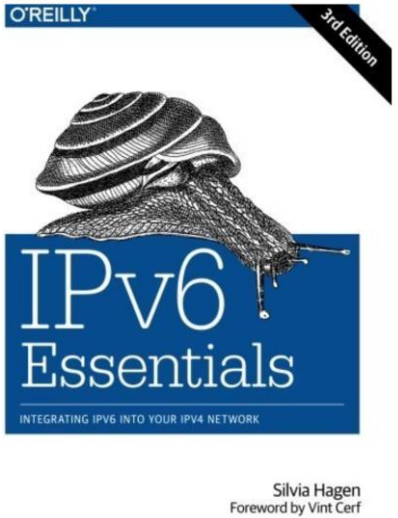
Cisco Press - https://www.ciscopress.com/store/ipv6-fundamentals-livelessons-a-straightforward-approach-9781587204579

O'Reilly Media – https://www.oreilly.com/library/view/introduction-to-ipv6/9781771375269/

More Books



https://amzn.eu/d/1GhV2Gn



https://amzn.eu/d/i5PVjAs

Online References

IPv6 Buzz Podcast - https://packetpushers.net/series/ipv6-buzz/

Infoblox IPv6 Center of Excellence - https://blogs.infoblox.com/category/ipv6-coe/

ARIN IPv6 Information - https://www.arin.net/resources/guide/ipv6/

APNIC IPv6 Information - https://www.apnic.net/community/ipv6/

RIPE IPv6 Info Centre - https://www.ripe.net/publications/ipv6-info-centre

Akamai IPv6 Adoption Visualization - https://www.akamai.com/internet-station/cyber-attacks/state-of-the-internet-report/ipv6-adoption-visualization

Cisco 6lab - https://6lab.cisco.com/

Google IPv6 Statistics - https://www.google.com/intl/en/ipv6/statistics.html

Tunnelbroker Hurricane Electric - https://tunnelbroker.net/

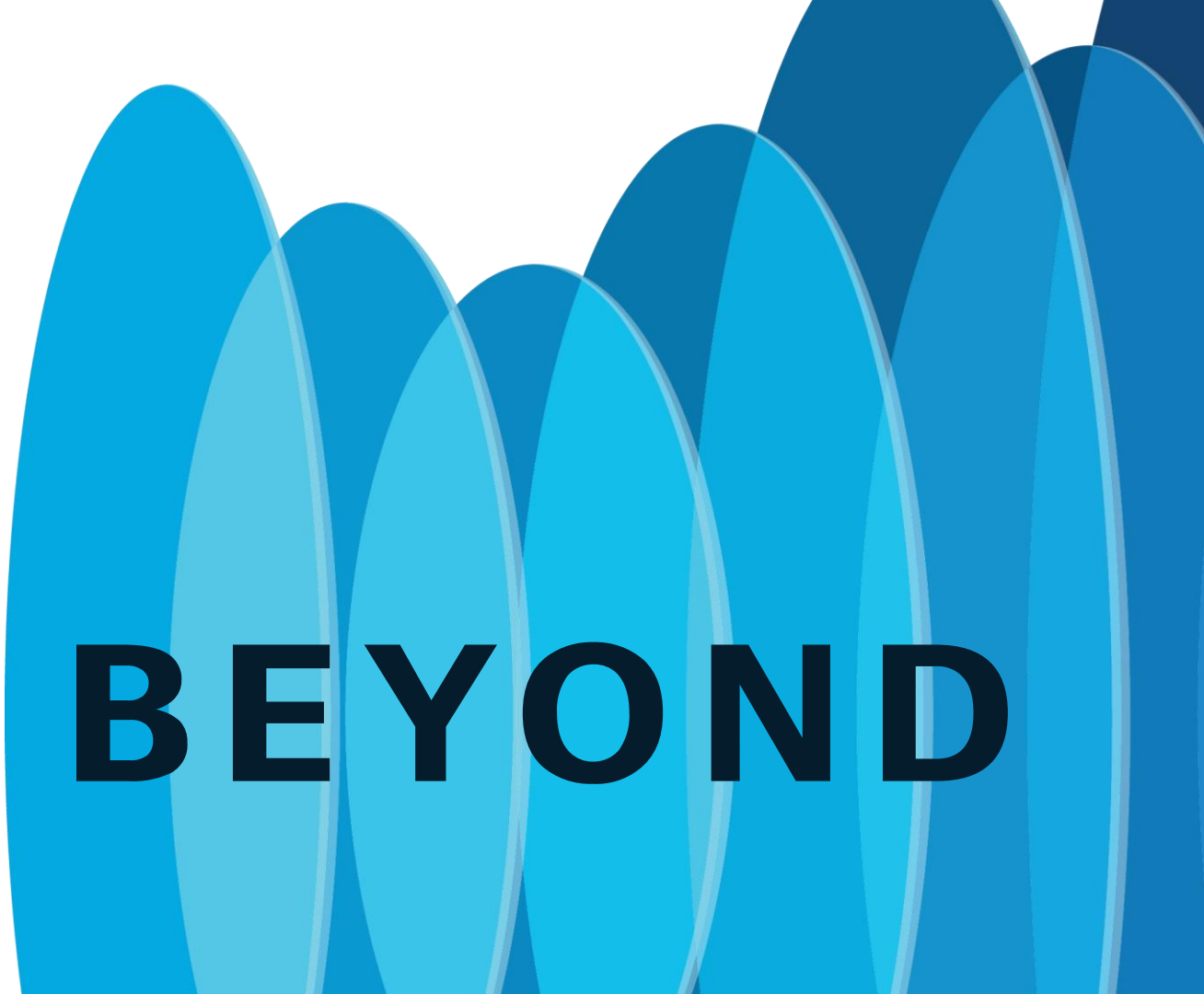
World IPv6 Launch - https://www.worldipv6launch.org/

IPv6 troubleshooting for Helpdesks

- http://isp.testipv6.com ➔

https://www.ripe.net/ripe/groups/tf/bcop/ipv6-troubleshooting-for-residential-isp-helpdesks

cisco Live!



GO BEYOND