The bridge to possible

Network Automation in Theory and Practice

A journey from YANG modelling to NETCONF, RESTCONF, gNMI and CLI management protocols

Jan Lindblad, NSO Engineering Architect

Special thanks to Roque Gagliano and Kristian Larsson

Cisco Webex App

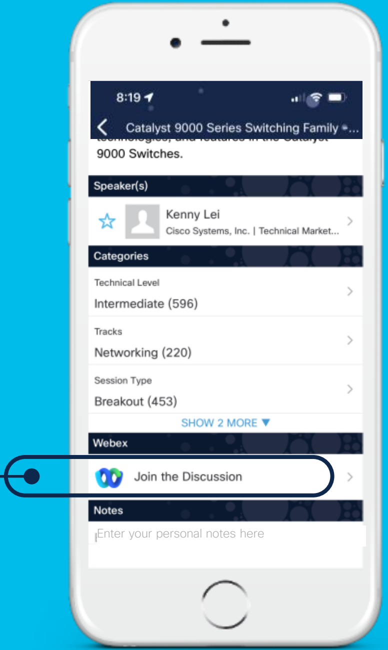
Questions?

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

How

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

Webex spaces will be moderated
by the speaker until February 24, 2023.



8:19

Catalyst 9000 Series Switching Family ...

9000 Switches.

Speaker(s)

Kenny Lei
Cisco Systems, Inc. | Technical Market...

Categories

Technical Level
Intermediate (596)

Tracks
Networking (220)

Session Type
Breakout (453)

SHOW 2 MORE ▼

Webex

Join the Discussion

Notes

Enter your personal notes here

BRKOPS-2431

cisco Live!

Is your Network Management Automated?

Hidden Agenda:
Trick question to make everyone wake up

Automation Levels in Self-Driving Cars

0 Warnings. The automated system issues warnings and may momentarily intervene but has no sustained vehicle control.

1 Hands on. The driver and the automated system share control of the vehicle. E.g. Adaptive Cruise Control. Parking Assist.

2 Hands off. The automated system takes full control of the vehicle: accelerating, braking, and steering. Driver needs to monitor.

3 Eyes off. The driver can safely turn their attention away from the driving tasks.

4 Mind off. As level 3, but no driver attention is ever required for safety.

5 Steering wheel optional. No human intervention is required at all.

Source: https://en.wikipedia.org/wiki/Self-driving_car

Automation Levels in Self-Driving Networks

0 Text Templates. Cutting and pasting from Word.

1 Macro scripts. CLI scripts with little pre- and post-checks. If something goes wrong, up to operator to fix.

2 Adaptive Activation Scripts. CLI scripts with computed values, pre- and post-checks. Cleans up after foreseen errors.

3 Model Driven Services. Works on intent (not CLI/protocol level), and autonomously navigates to desired state.

4 Verified Service Delivery. Also measure and report customer value.

5 Closed Loop. Add planning, prevention, mitigation, optimization.

Look up the full post on the NSO Developer Hub:

https://community.cisco.com/t5/nso-developer-hub-blogs/network-automation-levels/ba-p/4742665

Hidden Agenda:
Establish myself as thought leader early on

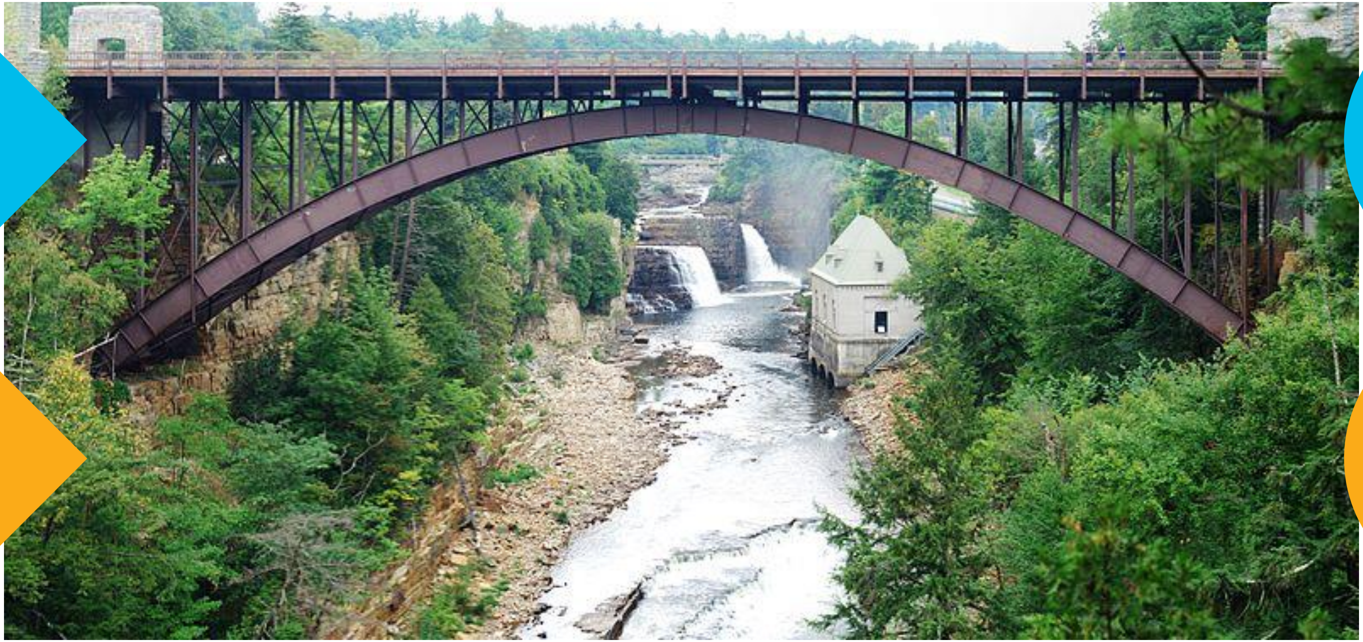
The Network Automation Chasm

Level 0-2

Level 3-5

Wooden house

Fossil car



Concrete building

Electric car

Image Source: https://commons.wikimedia.org/wiki/File:Ausable_Chasm_Bridge_-_1.jpg

Agenda

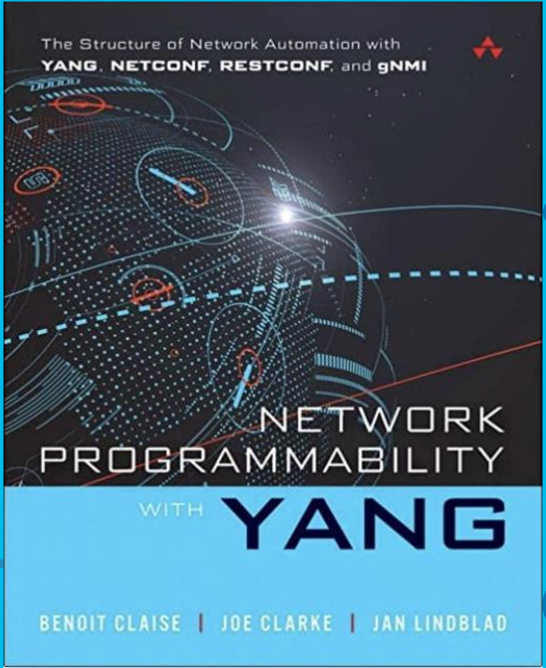
- Knowing your Powers PART 1
- Developing a Service PART 2
- Protocol Deep-Dive PART 3
- What we Learned FINALE

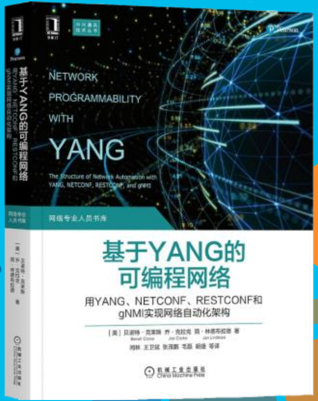
All in 90 minutes

Hidden Agenda:
Make everyone want to take the YANG book-journey

"The YANG Book"

https://www.amazon.com/Network-Programmability-YANG-Modeling-driven-Management/dp/0135180392


The Structure of Network Automation with
YANG, NETCONF, RESTCONF, and gNMI
NETWORK
PROGRAMMABILITY
WITH YANG
BENOIT CLAISE | JOE CLARKE | JAN LINDBLAD

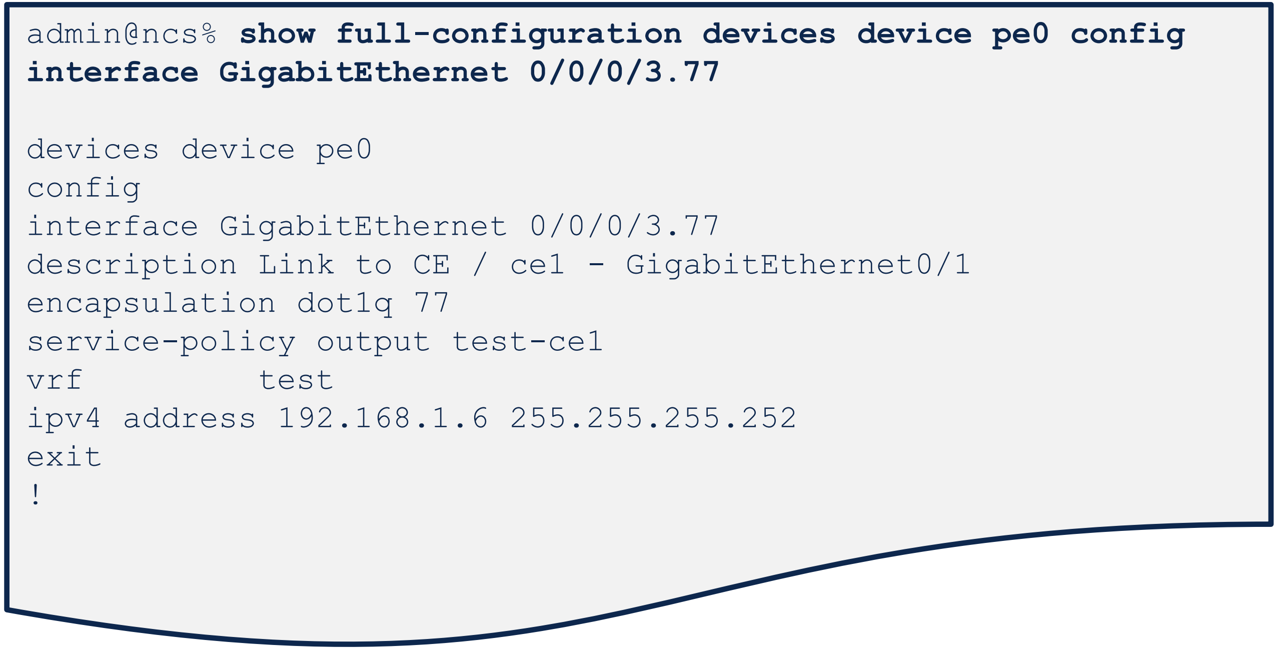

NETWORK
PROGRAMMABILITY
WITH
YANG
The Structure of Network Automation with
YANG, NETCONF, RESTCONF, and gNMI
网络专业人员书库
基于YANG的
可编程网络
用YANG、NETCONF、RESTCONF和
gNMI实现网络自动化架构

Agenda PART 1

Knowing your Powers

- Eight Superpowers to Cross the Chasm
- A Glance at NSO, Network Services Orchestrator
- "FastMap" Algorithm Walk-through

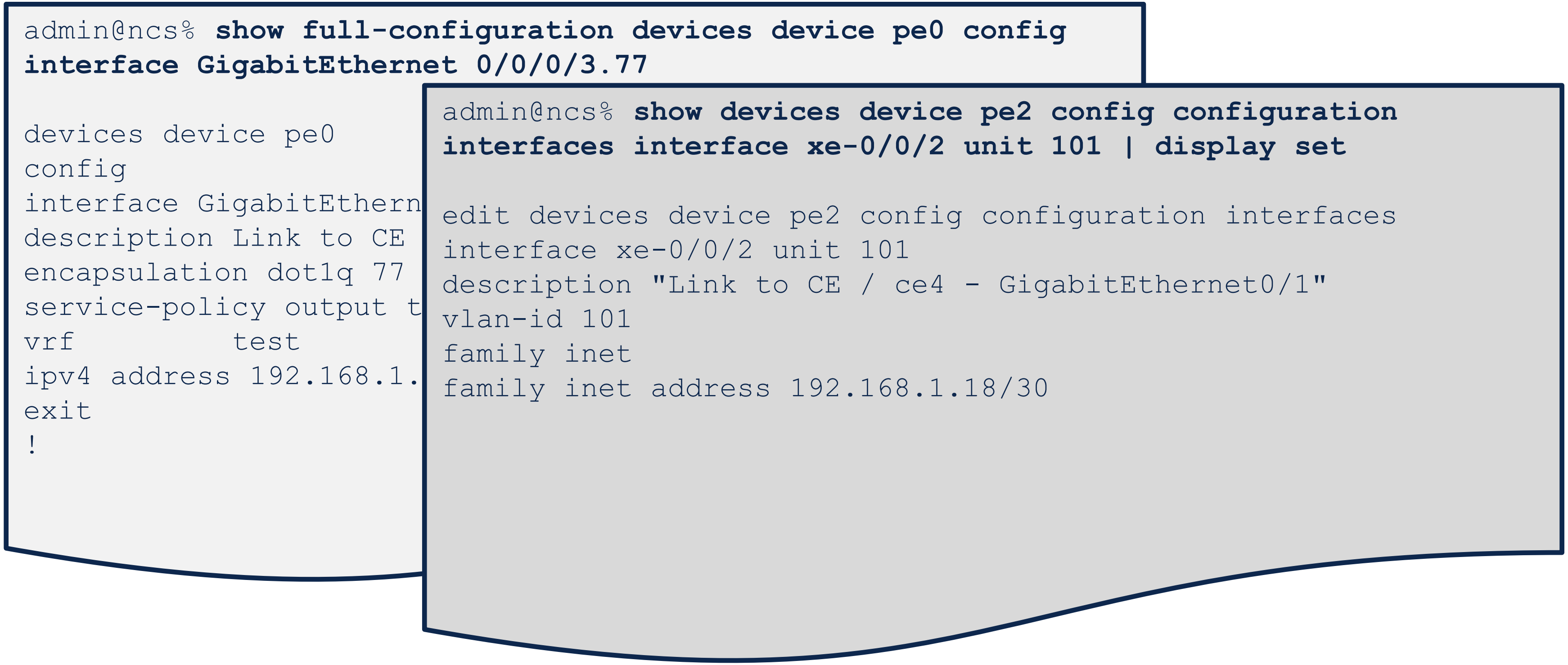
1 Service Centric



```
admin@ncs% show full-configuration devices device pe0 config
interface GigabitEthernet 0/0/0/3.77

devices device pe0
config
interface GigabitEthernet 0/0/0/3.77
description Link to CE / ce1 - GigabitEthernet0/1
encapsulation dot1q 77
service-policy output test-ce1
vrf          test
ipv4 address 192.168.1.6 255.255.255.252
exit
!
```

1 Service Centric



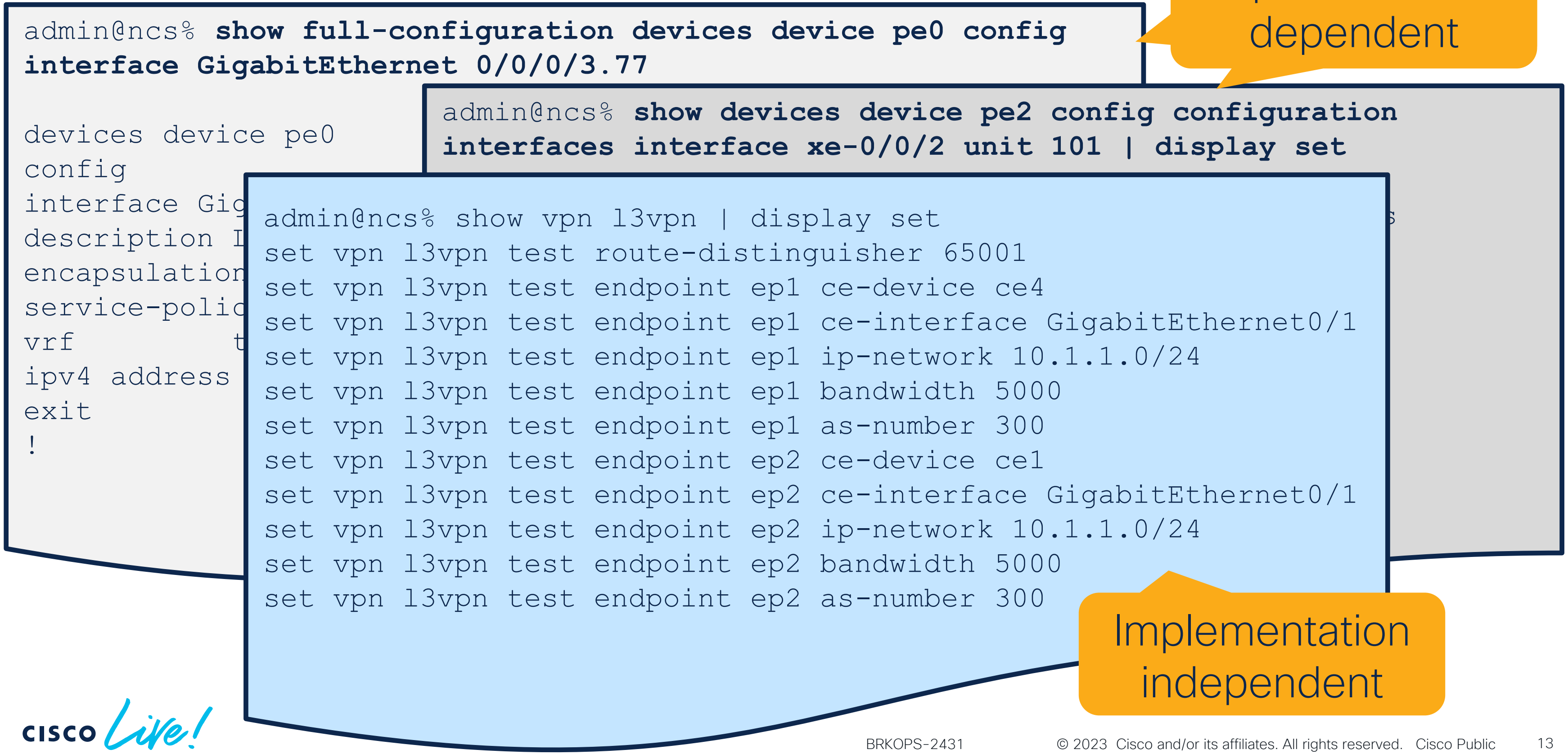
```
admin@ncs% show full-configuration devices device pe0 config
interface GigabitEthernet 0/0/0/3.77

devices device pe0
config
interface GigabitEthern
description Link to CE
encapsulation dot1q 77
service-policy output t
vrf          test
ipv4 address 192.168.1.
exit
!
```

```
admin@ncs% show devices device pe2 config configuration
interfaces interface xe-0/0/2 unit 101 | display set

edit devices device pe2 config configuration interfaces
interface xe-0/0/2 unit 101
description "Link to CE / ce4 - GigabitEthernet0/1"
vlan-id 101
family inet
family inet address 192.168.1.18/30
```

1 Service Centric



```
admin@ncs% show full-configuration devices device pe0 config
interface GigabitEthernet 0/0/0/3.77

devices device pe0
config
interface Gig
description I
encapsulation
service-polic
vrf          t
ipv4 address
exit
!
```

```
admin@ncs% show devices device pe2 config configuration
interfaces interface xe-0/0/2 unit 101 | display set
```

```
admin@ncs% show vpn l3vpn | display set
set vpn l3vpn test route-distinguisher 65001
set vpn l3vpn test endpoint ep1 ce-device ce4
set vpn l3vpn test endpoint ep1 ce-interface GigabitEthernet0/1
set vpn l3vpn test endpoint ep1 ip-network 10.1.1.0/24
set vpn l3vpn test endpoint ep1 bandwidth 5000
set vpn l3vpn test endpoint ep1 as-number 300
set vpn l3vpn test endpoint ep2 ce-device ce1
set vpn l3vpn test endpoint ep2 ce-interface GigabitEthernet0/1
set vpn l3vpn test endpoint ep2 ip-network 10.1.1.0/24
set vpn l3vpn test endpoint ep2 bandwidth 5000
set vpn l3vpn test endpoint ep2 as-number 300
```

Implementation dependent

Implementation independent

2 Template Based

Experience taught me:

If I ask what your service does, how do you describe it?

- Once into details, you will use (pseudo) CLI to express your ideas
- Particularly in the form of templates with variables
- Only the service create-case (never delete or modify)

Always.

Homo Sapiens Networkensis thinks in terms of adding config snippets through CLI.

Can computers do the same?

2 Template Based

Homo Sapiens Networkensis thinks in terms of adding config snippets through CLI.

Can computers do the same?

Yes, but CLIs not really built for automation

- No well-defined behavior
- Plenty of side effects
- Error messages?
- Transactions?

How to leverage Sapiens thinking style while removing automation hurdles?

3 Declarative

Set of configuration additions with variable substitution

Imperative template(s)

Create-case:

```
interface $ifname
 ip-address $ip $mask
 m
```

Delete-case:

```
interface
 no ip-add
 n
```

Sequences of protocol operations with variable substitution

Modify-case 1: ip-changed

```
interface $ifname
 ip-address $ip $mask
```

Modify-case 2: mtu-changed

```
interface $ifname
 mtu $mtu
```

Declarative template

Create-case:

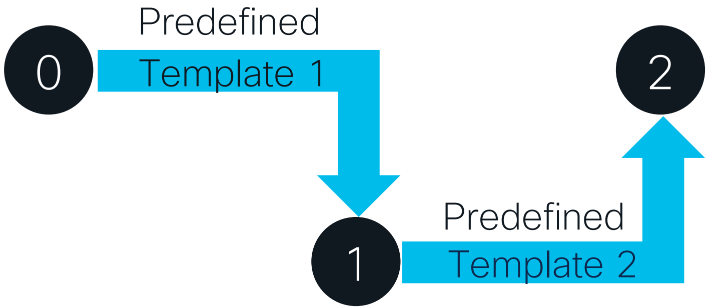
```
interface $ifname
 ip-address $ip $mask
 mtu $mtu
```

Declarative template unchanged

- Regardless of operation
- Regardless of protocol
- **Sequencing** (is transactional)

4 Stateful

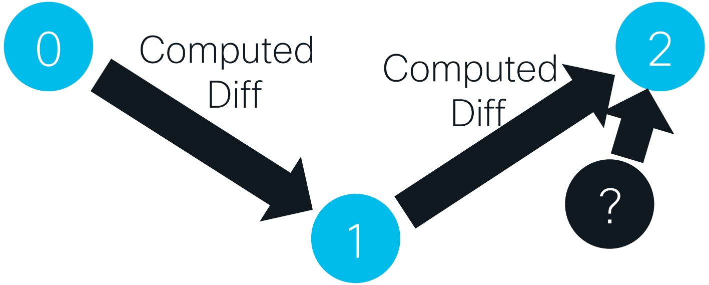
Stateless Management



Stateless approach moves between predefined states

Blindly applies predefined sequences of commands

Stateful Management



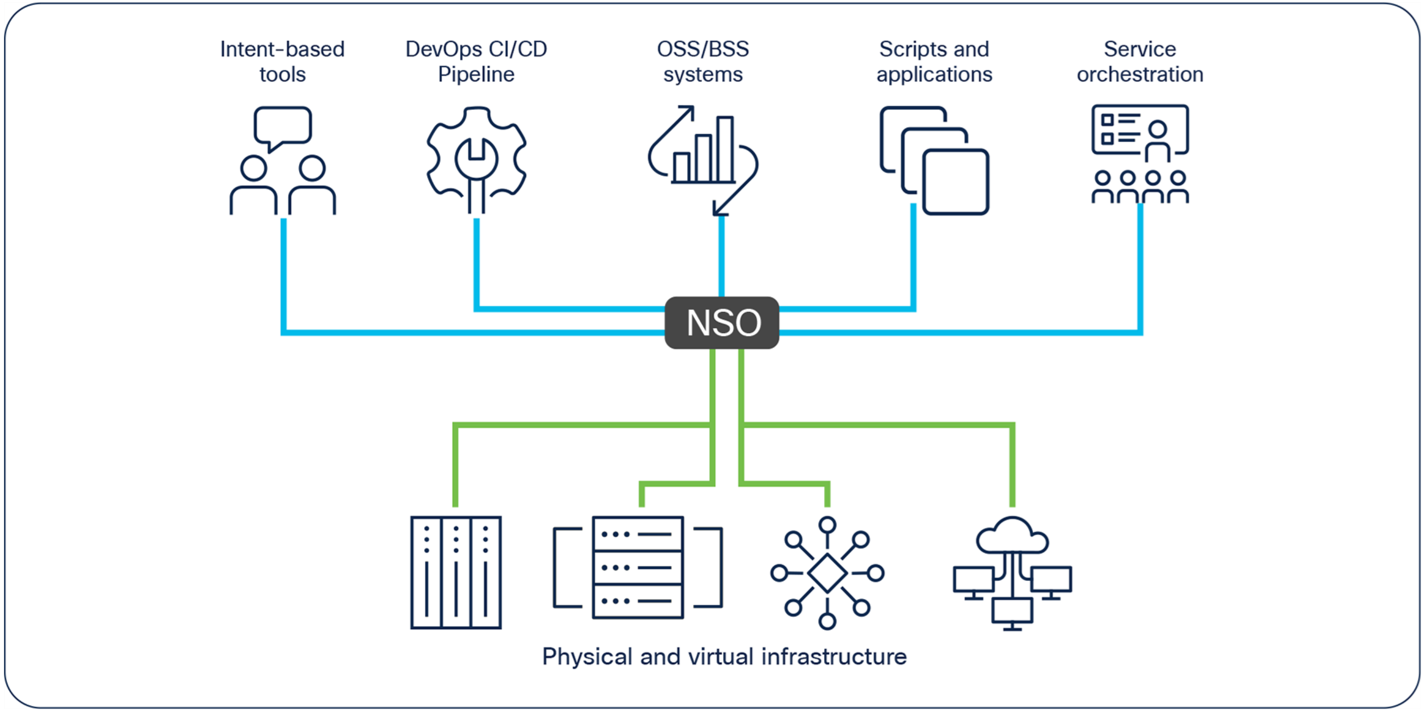
Stateful approach moves from any state to desired state

Computes and applies minimal diff from current to desired state

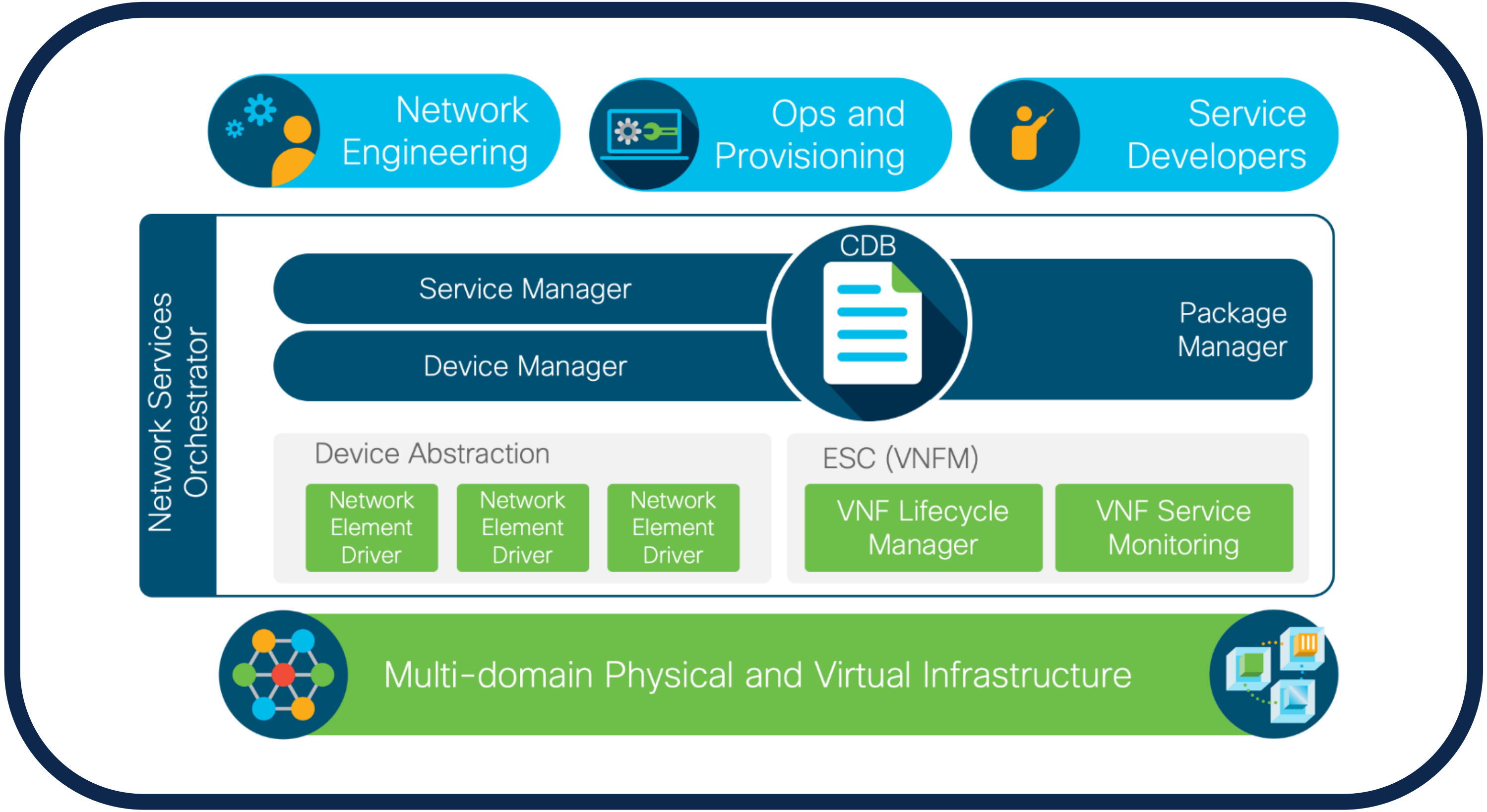
“You have seen them. You may have experienced them.

Managers that are stateless.”

Network Services Orchestrator (NSO)



Network Services Orchestrator (NSO)



5 Model Driven

YANG (RFC 7950)

- Schema language

Data type, range, meaning, units, any constraints, place in tree

CLI is for Instance Data

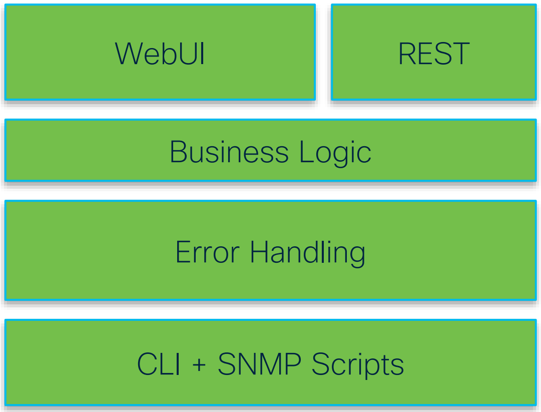
- Enables Model Driven design
- YANG modules covers 4 areas:
  - Configuration
  - Operational State
  - Actions
  - Notifications

Other Schema languages

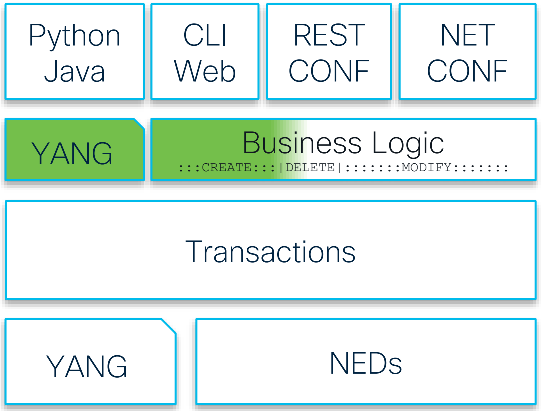
- SMI (Structure of Management Information)
- XSD (XML Schema Description)
- UML (Unified Modeling Language) + text
- OpenAPI / Swagger
- JSON Schema + YAML

6 Transactional

Traditional NMS Design



NSO



"Someone" is going to have to clean up when things go south

Traditionally

50% of code is

- Error detection
- Error handling

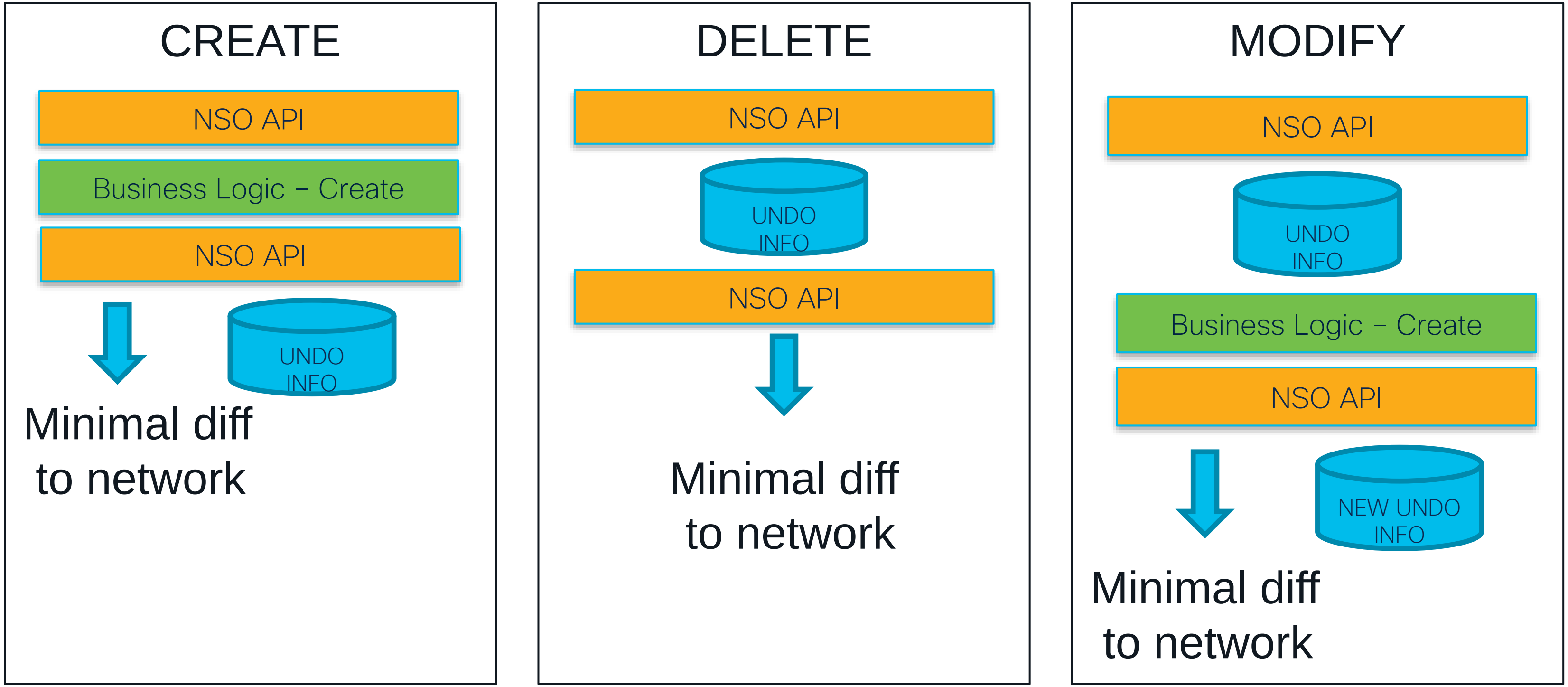
Error handling code is way more complicated than average code, so cost of error handling code development >>50%

7 Create-only

"FastMap"

Business Logic

:::CREATE:::|DELETE|::::::MODIFY::::::



CREATE

NSO API

Business Logic – Create

NSO API

UNDO INFO

Minimal diff to network

DELETE

NSO API

UNDO INFO

NSO API

Minimal diff to network

MODIFY

NSO API

UNDO INFO

Business Logic – Create

NSO API

NEW UNDO INFO

Minimal diff to network

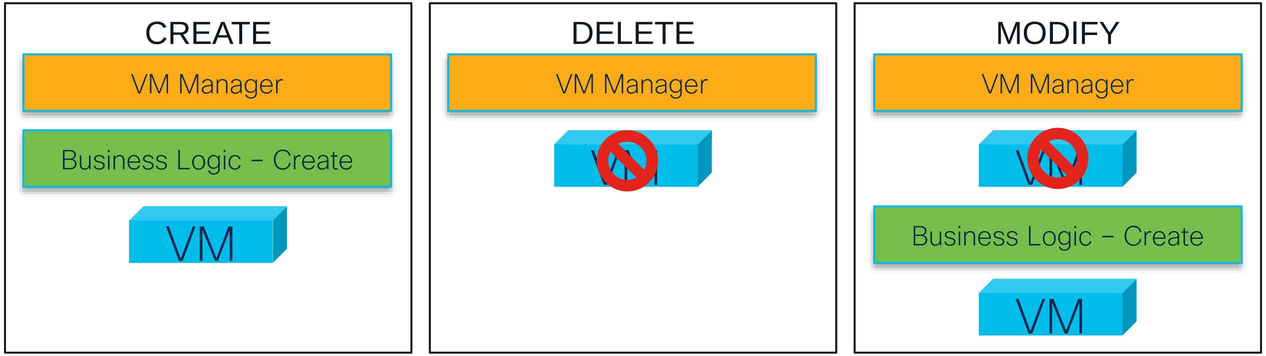
Are you a Destroyer?

There are other systems that also focus on the create-case for simplicity

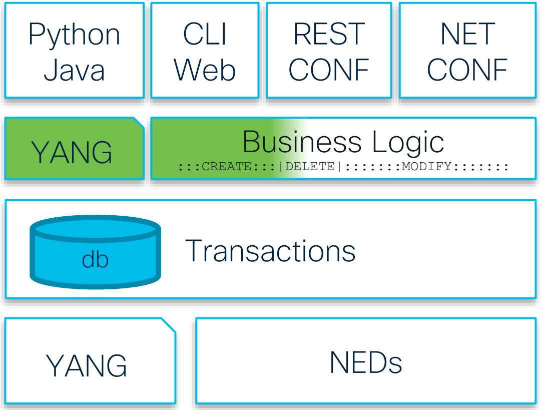
- E.g. Terraform, used for spinning up VMs
  - Code the Create-case
  - Delete by removing entire VM
  - Modify by running Delete + Create

This is great, if

- You are working with VMs
- It's ok to spin up a new VM every time there is a change



8 Composability

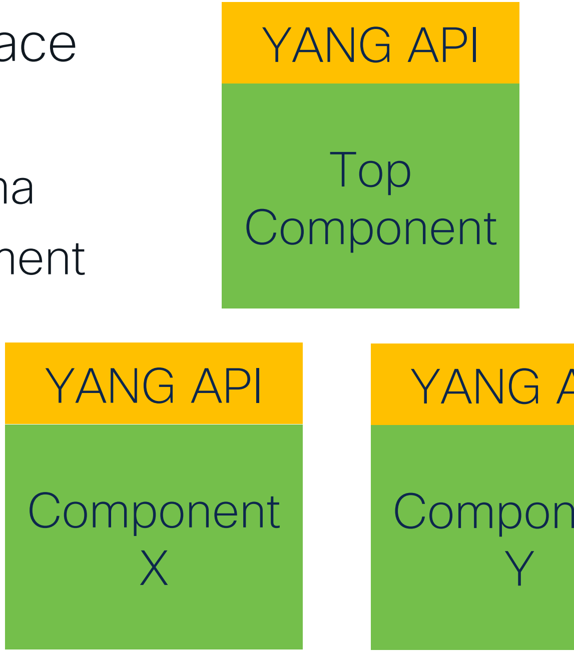


YANG Model your interface

- Render User Interfaces
- Render Database Schema
- Render Device Management Protocol Messages

YANG Components

YANG Transactionality



PART 2

Developing a Service

Hidden Agenda:
Now that concepts are understood, make everyone feel at home with development process

Agenda

Developing a Service PART 2

- Vision of what to Develop
- Modeling a Basic Service
- Giving it Legs (… well, a Template)
- Version 2: Add some QoS
- Version 3: Make it Multi-Vendor

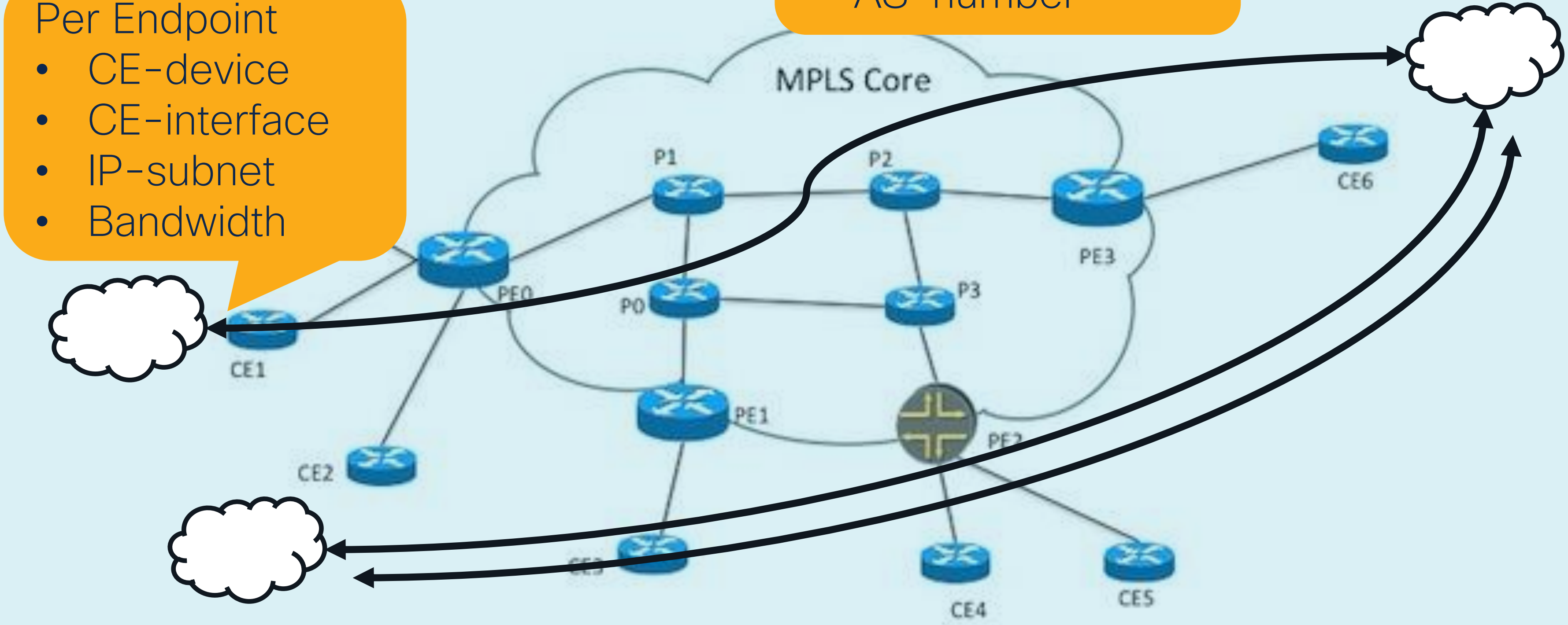
My VPN Service

Per Network

- Network Name
- AS-number

Per Endpoint

- CE-device
- CE-interface
- IP-subnet
- Bandwidth



Try it yourself

My VPN Service

~/nso/<nso-version>/examples.ncs/getting-started/developing-with-ncs/17-mpls-vpn-python/

Modeling a Basic Service

```
container vpn {
 list l3vpn {
  key name;
  leaf name { … }
  leaf as-number { … }
  list endpoint {
   key id;
   leaf id { … }
   leaf ce-device { … }
   leaf ce-interface { … }
   leaf ip-network { … }
   leaf bandwidth { … }
  }
 }
}
```

Per Network

- Network Name
- AS-number

Per Endpoint

- CE-device
- CE-interface
- IP-subnet
- Bandwidth

Modeling a Basic Service

```
container vpn {
 list l3vpn {
  key name;
  leaf name {
   type string;
  }
  leaf as-number {
   mandatory true;
   type uint32;
  }
  list endpoint {
   key id;
   leaf id {
    type string;
   }
```

Name will be any (UTF-8) string

AS-number will be any 32-bit integer

endpoint is a list, keyed by a string id

Modeling a Basic Service

```
 leaf ce-device {
  mandatory true;
  type leafref {
   path /ncs:devices/ncs:device/ncs:name;
  }
 }
 leaf ce-interface {
  mandatory true;
  type string;
 }
 leaf ip-network {
  mandatory true;
  type inet:ip-prefix;
 }
 leaf bandwidth {
  mandatory true;
  type uint32;
 }
}
```

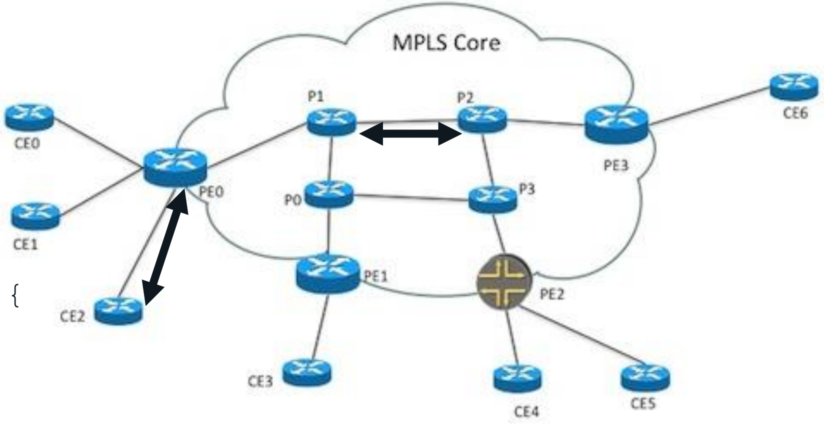
CE-device references the device at this endpoint from the NSO device list

endpoint is the interface name of the ce-device in this endpoint

IP-network is the IP-prefix for the area at this endpoint

Bandwidth is the max kbits/s delivered+received from this endpoint

Modeling a Basic Service



```
container topology {
 list role {
  key role;
  leaf role {
   type enumeration {
    enum ce;
    enum pe;
    enum p;
   }
  }
  leaf-list device {
   type leafref {
    path /ncs:devices
         /ncs:device/ncs:name;
   }
  }
 }
```

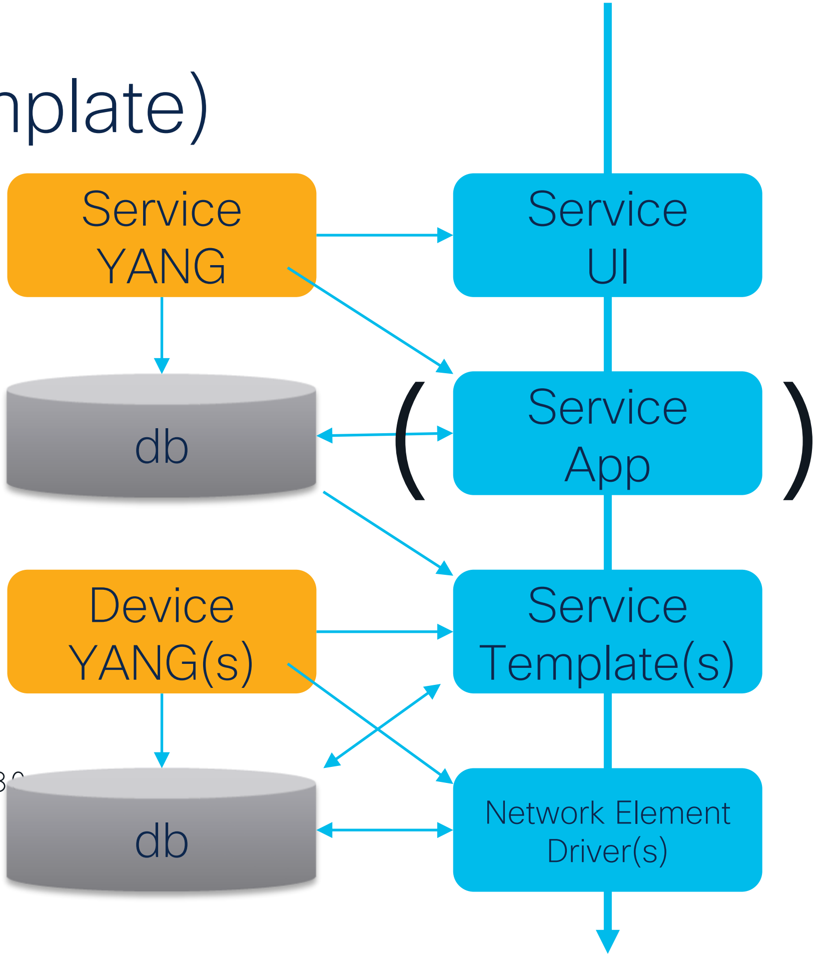
```
 list connection {
  key name;
  leaf name {
   type string;
  }
  container endpoint-1 {
   uses connection-grouping;
  }
  container endpoint-2 {
   uses connection-grouping;
  }
  leaf link-vlan {
   type uint32;
  }
 }
}
```

```
grouping connection-grouping {
 leaf device { … }
 leaf interface { … }
 leaf ip-address { … }
}
```

Giving it Legs (… well, a Template)

```
admin@ncs# packages reload
…
admin@ncs# show running-config topology
topology role ce
 device [ ce0 ce1 ce2 ce3 ce4 ce5 ce6 ce7 ce8 ]
topology role pe
 device [ pe0 pe1 pe2 pe3 ]
topology role p
 device [ p0 p1 p2 p3 ]

topology connection c0
 endpoint-1 device ce0 interface
  GigabitEthernet0/8 ip-address 192.168.1.1/30
 endpoint-2 device pe0 interface
  GigabitEthernet0/0/0/3 ip-address 192.168.1.2/30
 link-vlan 88
!
…
```



Giving it Legs (… well, a Template)

Configure device as needed using CLI, sync, then …

```
# show full-configuration devices device
cel config interface GigabitEthernet
0/1.77
devices device ce1
 config
  interface GigabitEthernet0/1.77
   encapsulation dot1Q 77
   ip address 192.168.1.5 255.255.255.252
   service-policy output abba
  exit
 !
!
# show full-configuration devices device
ce1 config interface GigabitEthernet
0/1.77 | display xml
```

… you get this XML

```
<interface xmlns="urn:ios">
 <GigabitEthernet>
  <name>0/1.77</name>
  <encapsulation>
   <dot1Q><vlan-id>77</vlan-id></dot1Q>
  </encapsulation>
  <ip>
   <address>
    <primary>
     <address>192.168.1.5</address>
     <mask>255.255.255.252</mask>
    </primary>
   </address>
  </ip>
  <service-policy>
   <output>abba</output>
  </service-policy>
 </GigabitEthernet>
</interface>
```

Giving it Legs (… well, a Template)

```
<interface xmlns="urn:ios" tags="merge">
 <GigabitEthernet>
  <name>{$CE_INT_NAME}.{$VLAN_ID}</name>
  <encapsulation>
    <dot1Q><vlan-id>{$VLAN_ID}</vlan-id>
  </encapsulation>
  <ip>
   <address>
    <primary>
     <address>{$LINK_CE_ADR}</address>
     <mask>{$LINK_MASK}</mask>
    </primary>
   </address>
  </ip>
  <service-policy>
   <output>{/name}</outpu
  </service-policy>
 </GigabitEthernet>
</interface>
```

```
<interface xmlns="urn:ios">
 <GigabitEthernet>
  <name>0/1.77</name>
  <encapsulation>
   <dot1Q><vlan-id>77</vlan-id></dot1Q>
  </encapsulation>
  <ip>
   <address>
    <primary>
     <address>192.168.1.5</address>
     <mask>255.255.255.252</mask>
    </primary>



              cy>
               </output>
               icy>
 </GigabitEthernet>
</interface>
```

Paste the XML into a template, replacing non-constant values with variables

Giving it Legs (… well, a Template)

```
<interface xmlns="urn:ios" tags="merge">
 <GigabitEthernet>
  <name>{$CE_INT_NAME}.{$VLAN_ID}</name>
  <encapsulation>
    <dot1Q><vlan-id>{$VLAN_ID}</vlan-id>
  </encapsulation>
  <ip>
   <address>
    <primary>
     <address>{$LINK_CE_ADR}</address>
     <mask>{$LINK_MASK}</mask>
    </primary>
   </address>
  </ip>
  <service-policy>
   <output>{/name}</output>
  </service-policy>
 </GigabitEthernet>
</interface>
```

Templates are in XML

- Nothing to do with NETCONF
  But everything to do with the device YANG
- Just unambiguous, vendor neutral language

Format unrelated to how NSO communicates with device

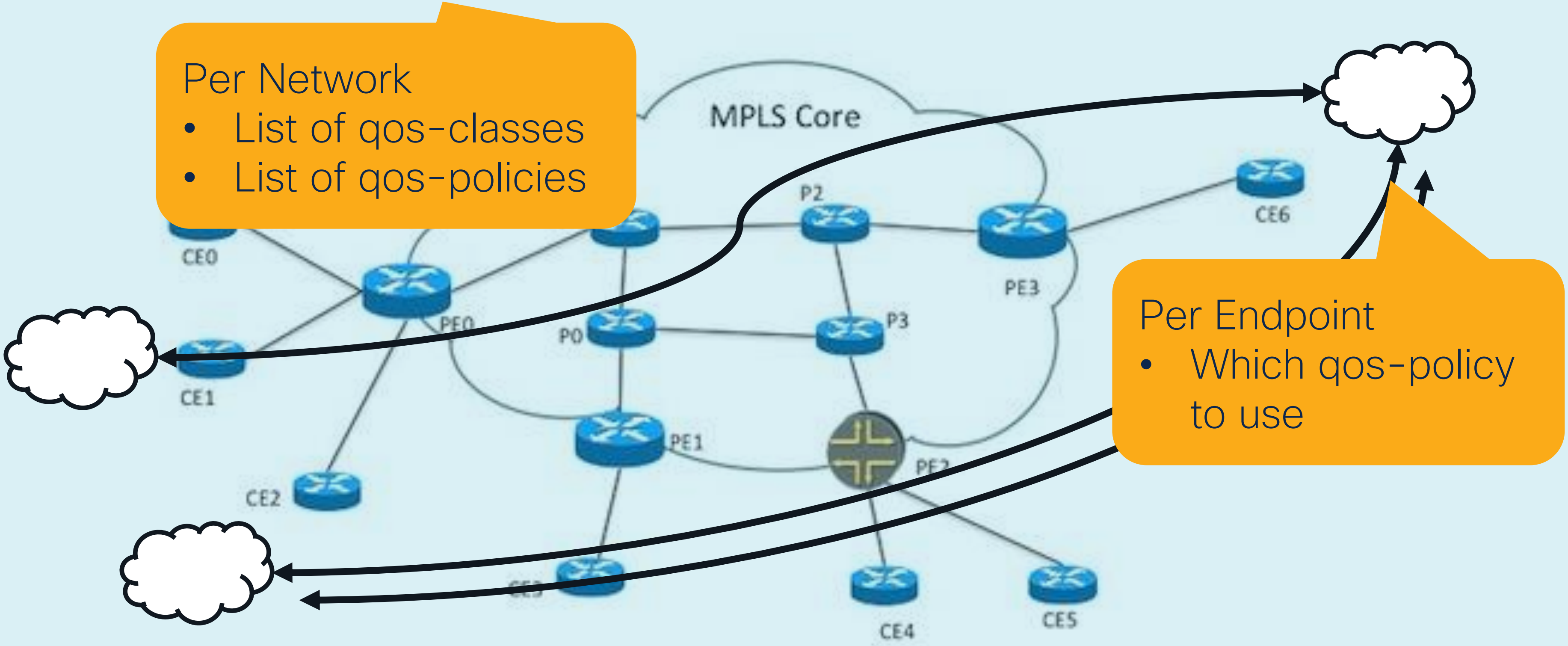
- Might be sending CLI to device.
  Or SNMP. Or NETCONF. Depends on NED.

Giving it Legs (… well, a Template)

```
admin@ncs(config)# vpn l3vpn abba
 as-number 65500
 endpoint helsinki
  ce-device     ce4
  ce-interface GigabitEthernet0/3
  ip-network   10.0.3.0/26
  bandwidth     20000
 !
 endpoint stockholm
  ce-device     ce1
  ce-interface GigabitEthernet0/2
  ip-network   10.0.2.0/24
  bandwidth     10000
 !
!
```

```
admin@ncs(config)# commit dry-run
device ce1 {
  interface {
+   GigabitEthernet 0/1.77 {
+     encapsulation {
+       dot1Q {
+         vlan-id 77;
+       }
+     }
+     ip {
+       address {
+         primary {
+           address 192.168.1.5;
+           mask 255.255.255.252;
…
device ce4 { … }
device pe0 { … }
device pe2 { … }
```

My VPN Service V2: Add some QoS



Per Network

- List of qos-classes
- List of qos-policies

Per Endpoint

- Which qos-policy to use

My VPN Service V2: Add some QoS

```
list qos-class {
  key  name;
  leaf name {
    type string;
  }
  leaf dscp-value {
    type dscp-type;
  }
  list match-traffic {
    key  name;
    leaf name {
      type string;
    }
    uses qos-match-grouping;
  }
}
```

```
grouping qos-match-grouping {
  leaf source-ip { … }
  leaf destination-ip { … }
  leaf port-start { … }
  leaf port-end { … }
  leaf protocol { … }
}
```

```
list qos-policy {
  key  name;
  leaf name {
    type string;
  }
  list class {
    key  qos-class;
    leaf qos-class {
      type leafref {
        path /qos/qos-class/name;
      }
    }
    leaf bandwidth-percentage {
      type uint32;
    }
    leaf priority {
      type empty;
    }
}
```

My VPN Service V2: Add some QoS

```
# show running-config qos qos-class
qos qos-class BUSINESS-CRITICAL
 dscp-value af21
 match-traffic ssh
  source-ip      any
  destination-ip any
  port-start     22
  port-end       22
  protocol       tcp
 !
!
qos qos-class MISSION-CRITICAL
 dscp-value af31
 match-traffic call-signaling
  source-ip      any
  destination-ip any
  port-start     5060
…
```

```
# show running-config qos qos-policy
qos qos-policy BRONZE
 class BUSINESS-CRITICAL
  bandwidth-percentage 20
 !
 class MISSION-CRITICAL
  bandwidth-percentage 10
 !
 class REALTIME
  bandwidth-percentage 10
 !
!
qos qos-policy GOLD
 class BUSINESS-CRITICAL
  bandwidth-percentage 20
 !
 class MISSION-CRITICAL
  bandwidth-percentage 25
…
```

My VPN Service V2: Add some QoS

```
list l3vpn {
  …
  list endpoint {
    leaf id { … }
    leaf ce-device { … }
    leaf ce-interface { … }
    leaf ip-network { … }
    leaf bandwidth { … }
    container qos {
      leaf qos-policy {
        type leafref {
          path /qos/qos-policy/name;
        }
      }
    }
```

Service Model

- Add qos-class, qos-policy, reference to policy for each VPN

Service Code

- Compute variable values, e.g. CLASS_NAME, MATCH_ENTRY
- Apply templates

Template

- Add acl, qos-class templates

My VPN Service V2: Add some QoS

```
def setup_qos_class(…):
  …
  for m in e.match_traffic:
    av = ncs.template.Variables()
    set_acl_vars(av, m, 'GLOBAL')
    av.add('CE', ce_endpoint.device)
    tmpl = ncs.template.Template(service)
    tmpl.apply('l3vpn-acl', av)
    av.add('CLASS_NAME', e.name)
    av.add('MATCH_ENTRY', 'GLOBAL-' + m.name)
    tmpl.apply('l3vpn-qos-class', av)
```

Service Model

- Add qos-class, qos-policy, reference to policy for each VPN

Service Code

- Compute variable values, e.g. CLASS_NAME, MATCH_ENTRY
- Apply templates

Template

- Add acl, qos-class templates

My VPN Service V2: Add some QoS

```
<config-template>
 <devices>
  <device tags="nocreate">
   <name>{$CE}</name>
   <config>
    <class-map xmlns="urn:ios" tags="merge">
     <name>{$CLASS_NAME}</name>
     <prematch>match-any</prematch>
     <match>
      <access-group>
       <name>{$MATCH_ENTRY}</name>
      </access-group>
     </match>
    </class-map>
   </config>
  </device>
 </devices>
</config-template>
```

Service Model

- Add qos-class, qos-policy, reference to policy for each VPN

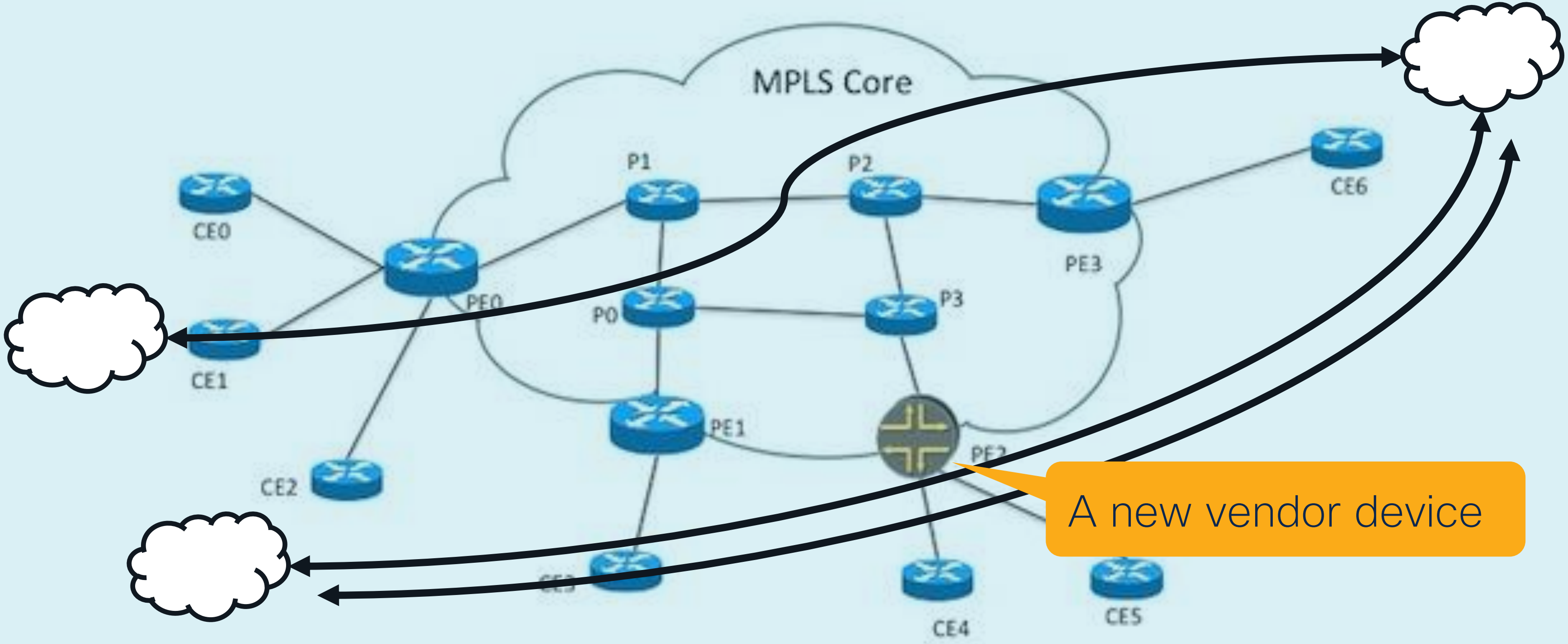
Service Code

- Compute variable values, e.g. CLASS_NAME, MATCH_ENTRY
- Apply templates

Template

- Add acl, qos-class templates

My VPN Service V3: Add a new Vendor



A new vendor device

Service Version 3: Let's Add a Vendor

```
<config-template>
 <devices>
  <device tags="nocreate">
   <name>{$PE}</name>
    <config>

     <policy-map tags="merge"
       xmlns="http://tail-f.com/ned/cisco-ios-xr">
      <name>{/name}-{$CE}</name>
      <class> …

     <configuration tags="merge"
       xmlns="http://xml.juniper.net/xnm/1.1/xnm">
      <scheduler-maps>
       <name>{$POLICY_NAME}</name>
       <forwarding-class>
        <name>{$CLASS_NAME}</name>
        <scheduler>{$POLICY_NAME}-…
```

No Service Model changes

No Service Code changes

Template

- Add new device type to template

PART 3

Protocol Deep-Dive

Hidden Agenda:
Now that the development process is clear, make everyone feel at home with running the system

Agenda

Protocol Deep-Dive PART 3

- Let's Create a few Clients
- The Power of Redeploy
- Brief History of Management Protocols
- Deep-dive: CLI and Automation
- Deep-dive: NETCONF and Network-wide Transactions
- Deep-dive: RESTCONF and User Interfaces
- Deep-dive: gNMI and Telemetry
- When to use CLI, NETCONF, RESTCONF, gNMI

Let's Create a few Clients

```
# vpn l3vpn enya as-number 65502

# endpoint cork bandwidth 25000
ce-device ce2 ce-interface
GigabitEthernet0/24 ip-network
10.17.3.192/26

# endpoint dublin bandwidth 40000
ce-device ce3 ce-interface
GigabitEthernet0/24 ip-network
10.17.4.192/26

# endpoint limerick bandwidth
12000 ce-device ce3 ce-interface
GigabitEthernet0/24 ip-network
10.17.7.128/26

# commit dry-run
```

```
device ce2 {
   config {
    interface {
+    GigabitEthernet 0/1.99 {
+     description "Link to PE / pe0 - GigabitEthernet0/0/0/3";
+     encapsulation {
+      dot1Q {
+       vlan-id 99;
+      }
+     }
+     ip {
+      address {
+       primary {
+        address 192.168.1.9;
+        mask 255.255.255.252;
+       }
+      }
+     }
+     service-policy {
+      output enya;
+     }
+    }
     GigabitEthernet 0/24 {
+     description "enya local network";
      ip {
       address {
        primary {
+        address 10.17.3.193;
+        mask 255.255.255.192;
        }
       }
      }
     }
    }
+    policy-map enya {
+     class class-default {
+      shape {
+       average {
+        bit-rate 25000;
+       }
+      }
+     }
+    }
     router {
+     bgp 65502 {
+      neighbor 192.168.1.10 {
+       remote-as 100;
+       activate;
+      }
+      network 10.17.3.192;
+     }
     }
    }
   }
```

```
device ce3 {
   config {
    interface {
+     GigabitEthernet 0/2.100 {
+      description "Link to PE / pe1 - GigabitEthernet0/0/0/5";
+      encapsulation {
+       dot1Q {
+        vlan-id 100;
+       }
+      }
+      ip {
+       address {
+        primary {
+         address 192.168.1.13;
+         mask 255.255.255.252;
+        }
+       }
+      }
+      service-policy {
+       output enya;
+      }
+     }
     GigabitEthernet 0/24 {
+      description "enya local network";
      ip {
       address {
        primary {
+         address 10.17.7.129;
+         mask 255.255.255.192;
        }
       }
      }
     }
    }
+    policy-map enya {
+     class class-default {
+      shape {
+       average {
+        bit-rate 12000;
+       }
+      }
+     }
+    }
    router {
+     bgp 65502 {
+      neighbor 192.168.1.14 {
+       remote-as 100;
+       activate;
+      }
+      network 10.17.4.192;
+      network 10.17.7.128;
+     }
    }
   }
  }
```

```
device pe0 {
   config {
    vrf {
+     vrf-list enya {
+      address-family {
+       ipv4 {
+        unicast {
+         import {
+          route-target {
+           address-list 65502:1;
+          }
+         }
+         export {
+          route-target {
+           address-list 65502:1;
+          }
+         }
+        }
+       }
+      }
+     }
    }
+    policy-map enya-ce2 {
+     class {
+      class-default class-default {
+       shape {
+        average {
+         25000;
+         bps;
+        }
+       }
+      }
+     }
+    }
    interface {
+     GigabitEthernet 0/0/0/3.99 {
+      description "Link to CE / ce2 - GigabitEthernet0/1";
+      ipv4 {
+       address {
+        ip 192.168.1.10;
+        mask 255.255.255.252;
+       }
+      }
+      service-policy {
+       output enya-ce2;
+      }
+      vrf enya;
+      encapsulation {
+       dot1q {
+        99;
+       }
+      }
+     }
    }
    router {
     bgp {
+     bgp-no-instance 100 {
+      vrf enya {
+       rd 65502:1;
+       address-family {
+        ipv4 {
+         unicast {
+         }
+        }
+       }
+       neighbor 192.168.1.9 {
+        remote-as 65502;
+        address-family {
+         ipv4 {
+          unicast {
+           as-override {
+           }
+          }
+         }
+        }
+       }
+      }
+     }
     }
    }
   }
  }
```

```
device pe1 {
   config {
    vrf {
+     vrf-list enya {
+      address-family {
+       ipv4 {
+        unicast {
+         import {
+          route-target {
+           address-list 65502:1;
+          }
+         }
+         export {
+          route-target {
+           address-list 65502:1;
+          }
+         }
+        }
+       }
+      }
+     }
    }
+    policy-map enya-ce3 {
+     class {
+      class-default class-default {
+       shape {
+        average {
+         12000;
+         bps;
+        }
+       }
+      }
+     }
+    }
    interface {
+     GigabitEthernet 0/0/0/5.100 {
+      description "Link to CE / ce3 - GigabitEthernet0/2";
+      ipv4 {
+       address {
+        ip 192.168.1.14;
+        mask 255.255.255.252;
+       }
+      }
+      service-policy {
+       output enya-ce3;
+      }
+      vrf enya;
+      encapsulation {
+       dot1q {
+        100;
+       }
+      }
+     }
    }
    router {
     bgp {
+     bgp-no-instance 100 {
+      vrf enya {
+       rd 65502:1;
+       address-family {
+        ipv4 {
+         unicast {
+         }
+        }
+       }
+       neighbor 192.168.1.13 {
+        remote-as 65502;
+        address-family {
+         ipv4 {
+          unicast {
+           as-override {
+           }
+          }
+         }
+        }
+       }
+      }
+     }
     }
    }
   }
  }
```

Create a few Clients

```
# vpn l3vpn yello as-number 65510

# endpoint bern bandwidth 10000
ce-device ce7 ce-interface
GigabitEthernet0/12 ip-network
10.107.60.0/23

# endpoint zurich bandwidth 10000
ce-device ce2 ce-interface
GigabitEthernet0/9 ip-network
10.24.24.0/24

# commit dry-run { outformat
native }
```

CLI Device

```
device {
 name ce5
  encapsulation dot1Q 102
  ip address 192.168.1.21 255.255.255.252
  exit
  policy-map yello
   class class-default
    shape average 10000
  interface GigabitEthernet0/1.102
…
```

NETCONF Device

```
device {
 name pe2
  <rpc message-id="1">
   <edit-config>
    <config>
     <configuration xmlns="http://xml.juniper.net/
      <interfaces>
       <interface>
        <name>xe-0/0/2</name>
```

The Power of Redeploy

Change Endpoint IP

```
# vpn l3vpn abba endpoint helsinki
ip-network 10.0.23.0/26
# commit dry-run outformat native
native {
 device {
  name ce4
  data
   interface GigabitEthernet0/3
   ip address 10.0.23.1 255.255.255.192
   exit
   router bgp 65500
   no network 10.0.3.0
   network 10.0.23.0
 }
}
```

Change Endpoint CE Device

```
# vpn l3vpn yello endpoint bern ce-device ce6
# commit dry-run outformat native
```

```
device ce5 {

device {
      name ce5
      data interface GigabitEthernet0/1.102
            description Link to PE / pe2 - xe-0/0/2
            encapsulation dot1Q 102
            ip address 192.168.1.21 255.255.255.252
           exit
           policy-map yello
            class class-default
             shape average 10000
            !
           !
           interface GigabitEthernet0/1.102
            service-policy output yello
           exit
           interface GigabitEthernet0/9
            description yello local network
            ip address 10.24.24.1 255.255.255.0
           exit
           router bgp 65510
            neighbor 192.168.1.22 remote-as 100
            neighbor 192.168.1.22 activate
            network 10.24.24.0
```

```
device pe3 {

device {
      name pe3
      data vrf yello
            address-family ipv4 unicast
             import route-target
              65510:1
             exit
             export route-target
              65510:1
             exit
            exit
           exit
           policy-map yello-ce6
            class class-default
             shape average 10000 bps
            !
            end-policy-map
           !
           interface GigabitEthernet 0/0/0/2.105
            description Link to CE / ce6 - GigabitEthernet0/1
            ipv4 address 192.168.1.34 255.255.255.252
            service-policy output yello-ce6
            vrf yello
            encapsulation dot1q 105
           exit
           router bgp 100
            vrf yello
             rd 65510:1
             address-family ipv4 unicast
             exit
             neighbor 192.168.1.33
              remote-as 65510
              address-family ipv4 unicast
               as-override
              exit
             exit
            exit
           exit
   }
```

```
device pe2 {

device {
      name pe2
      data <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
                message-id="1">
             <edit-config xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
               <target>
                 <candidate/>
               </target>
               <test-option>test-then-set</test-option>
               <error-option>rollback-on-error</error-option>
               <with-inactive xmlns="http://tail-f.com/ns/netconf/inactive/1.0"/>
               <config>
                 <configuration xmlns="http://xml.juniper.net/xnm/1.1/xnm">
                   <interfaces>
                     <interface>
                       <name>xe-0/0/2</name>
                       <unit>
                         <name>102</name>
                         <description>Link to CE / ce5 - GigabitEthernet0/1</description>
                         <family>
                           <inet>
                             <address>
                               <name>192.168.1.22/30</name>
                             </address>
                           </inet>
                         </family>
                         <vlan-id>102</vlan-id>
                       </unit>
                     </interface>
                   </interfaces>
                   <class-of-service>
                     <interfaces>
                       <interface>
                         <name>xe-0/0/2</name>
                         <unit>
                           <name>102</name>
                           <shaping-rate>
                             <rate>10000</rate>
                           </shaping-rate>
                         </unit>
                       </interface>
                     </interfaces>
                   </class-of-service>
                   <policy-options>
                     <community>
                       <name>yello-comm-exp</name>
                       <members>target:65510:1</members>
                     </community>
                     <community>
                       <name>yello-comm-imp</name>
                       <members>target:65510:1</members>
                     </community>
                     <policy-statement>
                       <name>yello-EXP</name>
                       <then>
                         <community>
                           <key1>add</key1>
                           <community-name>yello-comm-exp</community-name>
                         </community>
                         <accept/>
                       </then>
                       <from>
                         <protocol>bgp</protocol>
                       </from>
                     </policy-statement>
                     <policy-statement>
                       <name>yello-IMP</name>
                       <then>
                         <accept/>
                       </then>
                       <from>
                         <community>yello-comm-imp</community>
                         <protocol>bgp</protocol>
                       </from>
                     </policy-statement>
                   </policy-options>
                   <routing-instances>
                     <instance>
                       <name>yello</name>
                       <vrf-export>yello-EXP</vrf-export>
                       <interface>
                         <name>xe-0/0/2.102</name>
                       </interface>
                       <vrf-table-label/>
                       <vrf-import>yello-IMP</vrf-import>
                       <protocols>
                         <bgp>
                           <group>
                             <name>yello</name>
                             <neighbor>
                               <name>192.168.1.21</name>
                             </neighbor>
                             <peer-as>65510</peer-as>
                             <local-address>192.168.1.22</local-address>
                             <local-as>
                               <as-number>100</as-number>
                             </local-as>
                           </group>
                         </bgp>
                       </protocols>
                       <instance-type>vrf</instance-type>
                       <route-distinguisher>
                         <rd-type>65510:1</rd-type>
                       </route-distinguisher>
                     </instance>
                   </routing-instances>
                 </configuration>
               </config>
             </edit-config>
           </rpc>
   }
```

```
device ce6 {

device {
      name ce6
      data interface GigabitEthernet0/1.105
            description Link to PE / pe3 - GigabitEthernet0/0/0/2
            encapsulation dot1Q 105
            ip address 192.168.1.33 255.255.255.252
           exit
           policy-map yello
            class class-default
             shape average 10000
            !
           !
           interface GigabitEthernet0/1.105
            service-policy output yello
           exit
           interface GigabitEthernet0/12
            description yello local network
            ip address 10.107.60.1 255.255.254.0
           exit
           router bgp 65510
            neighbor 192.168.1.34 remote-as 100
            neighbor 192.168.1.34 activate
            network 10.107.60.0
```

A Brief History of Management Protocols

- CLI
- SNMP
- RFC 3535: Requirements
- NETCONF
- YANG
- RESTCONF
- gNMI, CORECONF, …

CLI

Command Line Interface

- Very common
- No standards
- Complex to get right
  - Sequencing dependencies
  - Side effects
  - Sub-modes
  - Consistent error messages?
  - Transactional?
- Partial coverage
  - Due to complexity, cost
  - Based on customer demand

Deep-dive: CLI and Automation

```
# vpn l3vpn abba endpoint helsinki
ip-network 10.0.23.0/26
# commit dry-run outformat native
native {
 device {
  name ce4
  data
!!  interface GigabitEthernet0/3
     ip address 10.0.23.1 255.255.255.192
!!  exit
     router bgp 65500
!!   no network 10.0.3.0
     network 10.0.23.0
 }
}
```

YANG model

- Much manual labor
- Lots of annotations to describe device specific behavior

```
// interface GigabitEthernet *
list GigabitEthernet {
 tailf:info "GigabitEthernet IEEE 802.3z";
 tailf:cli-allow-join-with-key {
  tailf:cli-display-joined;
 }
 tailf:cli-mode-name "config-if";
 tailf:cli-suppress-key-abbreviation;
 key name;
 leaf name {
  type string {
   pattern "[0-9]+.*";
   tailf:info "<0-66>/<0-128>;;Gigabit
     Ethernet interface number";
```

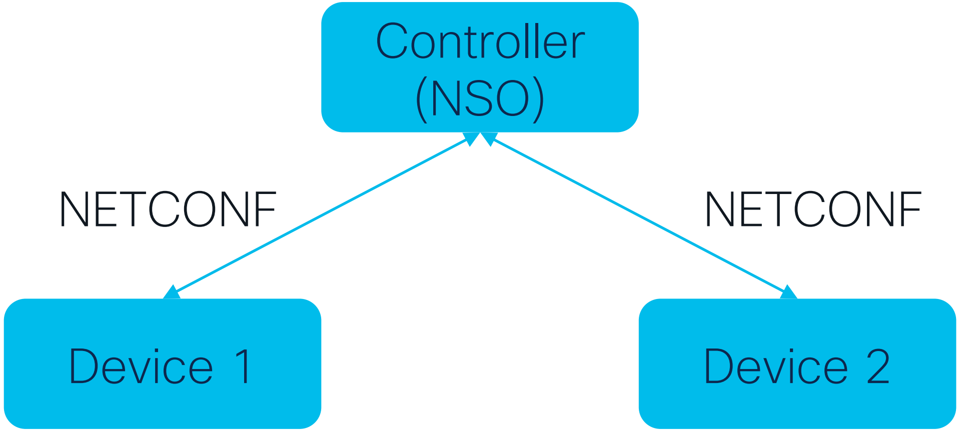
NETCONF

Management Protocol, rendered from YANG model,

based on RFC 3535 requirements:

- Cover all management functionality incl. config, state, actions, notifications
- Separate config/state data
- Save and load textual configs, UTF-8
- High security, SSH
- Transactions, network wide

Deep-dive: NETCONF Network-wide Transactions



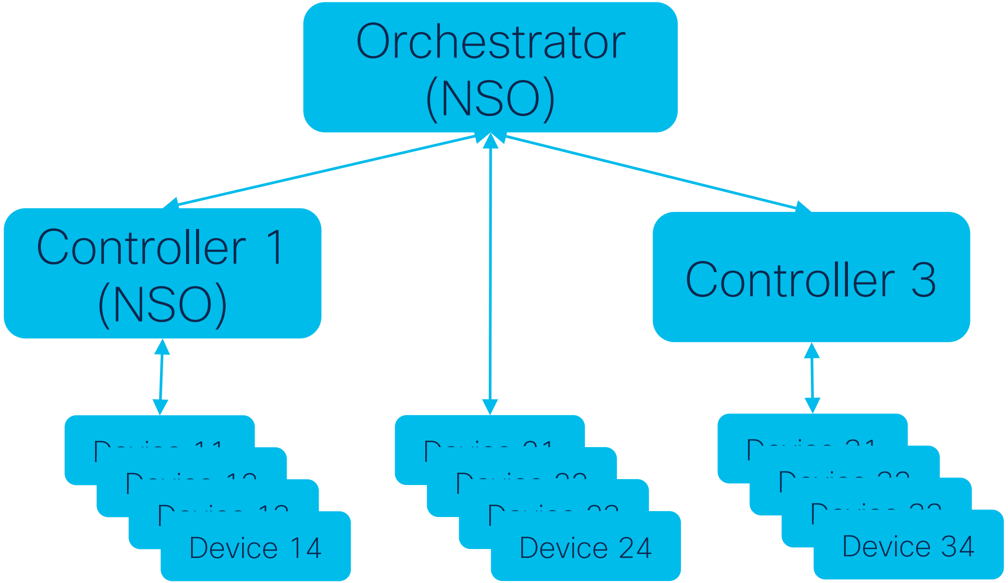
2-phase commit

- Prepare
- Commit

3-phase commit

- Prepare
- Commit
- Confirm

Deep-dive: NETCONF Network-wide Transactions



2-phase commit

- Prepare
- Commit

3-phase commit

- Prepare
- Commit
- Confirm

Deep-dive: NETCONF Network-wide Transactions

Exchange between Controller (NSO) and Device (XR)

Client (NSO) sends

```
<hello>
  <capabilities>
    <capability>
      urn:ietf:params:netconf:base:1.0
    </capability>
    <capability>
      urn:ietf:params:netconf:base:1.1
    </capability>
  </capabilities>
</hello>
```

Server (XR) responds

```
<hello>
  <capabilities>
    <capability>
      urn:ietf:params:netconf:base:1.0
    </capability>
    <capability>
      urn:ietf:params:netconf:base:1.1
    </capability>

    … 375 more capabilities…

    <capability>urn:ietf:params:xml:ns:
      yang:ietf-yang-types?
      module=ietf-yang-types&amp;
      revision=2013-07-15
    </capability>
  </capabilities>
  <session-id>16</session-id>
</hello>
```

Deep-dive: NETCONF Network-wide Transactions

Exchange between Controller (NSO) and Device (XR)

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netco
nf:base:1.0" message-id="1">
 <get-config>
  <source><running/></source>
  <filter>
   <tty xmlns="http://cisco.com/ns/yang/
    Cisco-IOS-XR-tty-server-cfg">
   </tty>
   <tacacs-server xmlns="http://www.cisc
    o.com/ns/yang/Cisco-IOS-XR-sysadmin-
    tacacs-tacacs-server">
   </tacacs-server>
   <hostname xmlns="http://www.cisco.com
    /ns/yang/Cisco-IOS-XR-sysadmin-nto-
    misc-set-hostname">
   </hostname>
…
```

```
<?xml version="1.0"?>
<rpc-reply message-id="1" xmlns="urn:iet
f:params:xml:ns:netconf:base:1.0">
 <data>
  <tty xmlns="http://cisco.com/ns/yang/
   Cisco-IOS-XR-tty-server-cfg">
   <tty-lines>
    <tty-line>
     <name>console</name>
     <exec>
      <timeout>
       <minutes>0</minutes>
       <seconds>0</seconds>
      </timeout>
     </exec>
    </tty-line>
    <tty-line>
     <name>default</name>
…
```

Deep-dive: NETCONF Network-wide Transactions

Exchange between Controller (NSO) and Device (XR)

```
<rpc message-id="1">
  <discard-changes/>
</rpc>

<rpc message-id="2">
  <lock>
    <target><candidate/></target>
  </lock>
</rpc>

<rpc message-id="3">
  <get> …
    <transaction-id/> …
  </get>
</rpc>
```

```
<rpc-reply message-id="1">
  <ok/>
</rpc-reply>

<rpc-reply message-id="2">
  <ok/>
</rpc-reply>

<rpc-reply message-id="3">
  <data> …
    <transaction-id>
      1671-35757-396554
    </transaction-id> …
  </data>
</rpc>
```

Deep-dive: NETCONF Network-wide Transactions

Exchange between Controller (NSO) and Device (XR)

```
<rpc message-id="4">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <test-option>test-then-set</test-option>
    <error-option>rollback-on-error</error-
    <config>
      <vrfs>
        <vrf>
          <vrf-name>abba</vrf-name>
          <address-family>
…
<rpc message-id="5">
  <validate>
    <source><candidate/></source>
  </validate>
</rpc>
```

```
<rpc-reply message-id="4">
  <ok/>
</rpc-reply>

<rpc-reply message-id="5">
  <ok/>
</rpc-reply>
```

Deep-dive: NETCONF Network-wide Transactions

Exchange between Controller (NSO) and Device (XR)

```
<rpc message-id="6">
  <commit><confirmed/></commit>
</rpc>
```

```
<rpc-reply message-id="6">
  <ok/>
</rpc-reply>
```

Wait; are we happy with the situation? Proceed or abort?

```
<rpc message-id="7">
  <commit/>
</rpc>
```

Proceed.

```
<rpc-reply message-id="7">
  <ok/>
</rpc-reply>
```

```
<rpc message-id="8">
  <get> …
    <transaction-id/> …
  </get>
<rpc/>
```

```
<rpc-reply message-id="8">
  <data> …
    <transaction-id>
      1671-35948-567882
    </transaction-id> …
  </data>
</rpc-reply>
```

```
<rpc message-id="9">
  <unlock>
    <target><candidate/></target>
  </unlock>
</rpc>
```

```
<rpc-reply message-id="9">
  <ok/>
</rpc-reply>
```

cisco Live!

RESTCONF

Management Protocol rendered from YANG model,

cross of REST + NETCONF:

- Because REST is stateless, some functionality removed
- XML and/or JSON payload
- Transactional, but no network-wide transactions
- New YANG-PATCH operation added

Deep-dive: RESTCONF and User Interfaces

GET
- Get configuration/operational state

POST
- Create/merge one configuration subtree
- Execute one action

PUT
- Replace one configuration subtree

DELETE
- Delete one configuration subtree

PATCH
- Update one configuration subtree

YANG-PATCH (optional)
- Transactionally update a collection of configuration subtrees

RESTCONF is (one kind of) REST
- REST is (generally) not RESTCONF
- HTTPS often passes through firewall
- HATEOAS principles
  Hypermedia as the Engine of Application State

Deep-dive: RESTCONF and User Interfaces

Exchange between Postman (client) and Controller (NSO)

```
GET http://localhost:8080/restconf/data/
l3vpn:vpn?content=config&depth=3

 Accept: application/yang-data+xml
```

```
<vpn>
    <l3vpn>
        <name>abba</name>
        <as-number>65500</as-number>
        <endpoint/>
        <endpoint/>
    </l3vpn>
    <l3vpn>
        <name>enya</name>
        <as-number>65502</as-number>
        <endpoint/>
        <endpoint/>
        <endpoint/>
    </l3vpn>
    <l3vpn>
        <name>yello</name>
        <as-number>65510</as-number>
        <endpoint/>
        <endpoint/>
    </l3vpn>
```

Deep-dive: RESTCONF and User Interfaces

Exchange between Postman (client) and Controller (NSO)

```
GET http://localhost:8080/restconf/data/
l3vpn:vpn?content=config&depth=3

 Accept: application/yang-data+json
```

```
{
  "l3vpn:vpn": {
    "l3vpn": [
      {
        "name": "abba",
        "as-number": 65500,
        "endpoint": []
      },
      {
        "name": "enya",
        "as-number": 65502,
        "endpoint": []
      },
      {
        "name": "yello",
        "as-number": 65510,
        "endpoint": []
      }
    ]
  }
}
```

Deep-dive: RESTCONF YANG-PATCH

Exchange between Postman (client) and Controller (NSO)

Request to create client david

```
PATCH http://localhost:8080/restconf/data

 Content-Type: application/yang-patch+json
 Accept: application/yang-data+json

{
  "ietf-yang-patch:yang-patch" : {
    "patch-id" : "Order 4711",
    "edit" : [
      {
        "edit-id" : "edit1",
        "operation" : "merge",
        "target" : "/l3vpn:vpn/l3vpn=abba
                    /endpoint=helsinki",
        "value" : {
          "l3vpn:endpoint" : [
            {
              "ip-network": "10.0.42.0/23",
              "bandwidth": 30000
            }
          ]
        }
      },
      {
        "edit-id" : "edit2",
        "operation" : "delete",
        "target" : "/l3vpn:vpn/l3vpn=yello"
      }
    ]
  }
}
```

200 OK

gNMI

gRPC Network Management Interface

gRPC

gRPC Remote Procedure Call

gNMI, Management Protocol rendered from YANG

- Base operations defined using gRPC
  - gRPC can be encoded as JSON or Google protobufs
- Fairly similar to RESTCONF
- Popular for telemetry
- Almost transactional

Deep-dive: gNMI and Telemetry

```
$ cisco-gnmi get
-encoding JSON
-data_type ALL
-os NX-OS
-root_certificates
 ./gnmi.pem
-ssl_target_override
 divya
-xpath "/interfaces
 /interface
 [name='mgmt0']"
 172.25.75.81:50051

Username: admin
Password: xxxxxx
```

```
INFO:root:notification {
  timestamp: 1596066432721
  update {
    path {
      origin: "openconfig"
      elem {
        name: "interfaces"
      }
      elem {
        name: "interface"
        key {
          key: "name"
          value: "mgmt0"
        }
      }
    }
```

```
  val {
   json_val:
"[{\"name\":\"mgmt0\",\"config\":{\"enab
led\":true,\"mtu\":1500,\"name\":\"mgmt0
\",\"type\":\"ethernetCsmacd\"},\"state\
":{\"admin-status\":\"UP\",\"last-
change\":\"1595937502596000000\",\"oper-
status\":\"UP\",\"enabled\":true,\"mtu\"
:1500,\"name\":\"mgmt0\",\"type\":\"ethe
rnetCsmacd\"},\"subinterfaces\":{\"subin
terface\":[{\"index\":0,\"config\":{\"in
dex\":0},\"ipv4\":{\"addresses\":{\"addr
ess\":[{\"ip\":\"172.25.75.81\",\"config
\":{\"ip\":\"172.25.75.81\",\"prefix-
length\":23}}]},\"proxy-
arp\":{\"config\":{\"mode\":\"DISABLE\"}
}}}]}}]"
  }
 }
}
```

Source: Divya Rao's blog post, link on next slide

Telemetry in Action: NETCONF and gNMI with a Custom-Built Collector!

by Divya Rao

https://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/white-paper-c11-744191.html

When to use CLI, NETCONF, RESTCONF, gNMI

	CLI	NETCONF	RESTCONF	gNMI
Sweet spot	When other protocol options are not supported	Multi-device configuration use cases	Portals and Front-End applications	Telemetry data collection applications
Avoid when	When other options are available. Expensive to integrate & maintain. Due to cost, integrated functionality is often limited.	Avoid when device's support for NETCONF is poor/untested.	Not well suited to configure multiple devices. Lacks advanced management features.	Not well suited for configuration. Thin/weak specifications often lead to interoperability issues.

FINALE

What we Learned

Hidden Agenda:
Now drive home the main points again, and ensure they stick in everybody's memory.

Summary

Service Development

+ Start with Your Vision
+ Model your Service in YANG
  - Gives you a User Interface
  - Gives you a Database Schema
+ Load up Network Element Driver(s) for your network
  - Gives you Device YANG Models
+ Add declarative Template(s) for the device(s)/service(s) you use
+ Add any Service Code
  - If you are doing advanced stuff

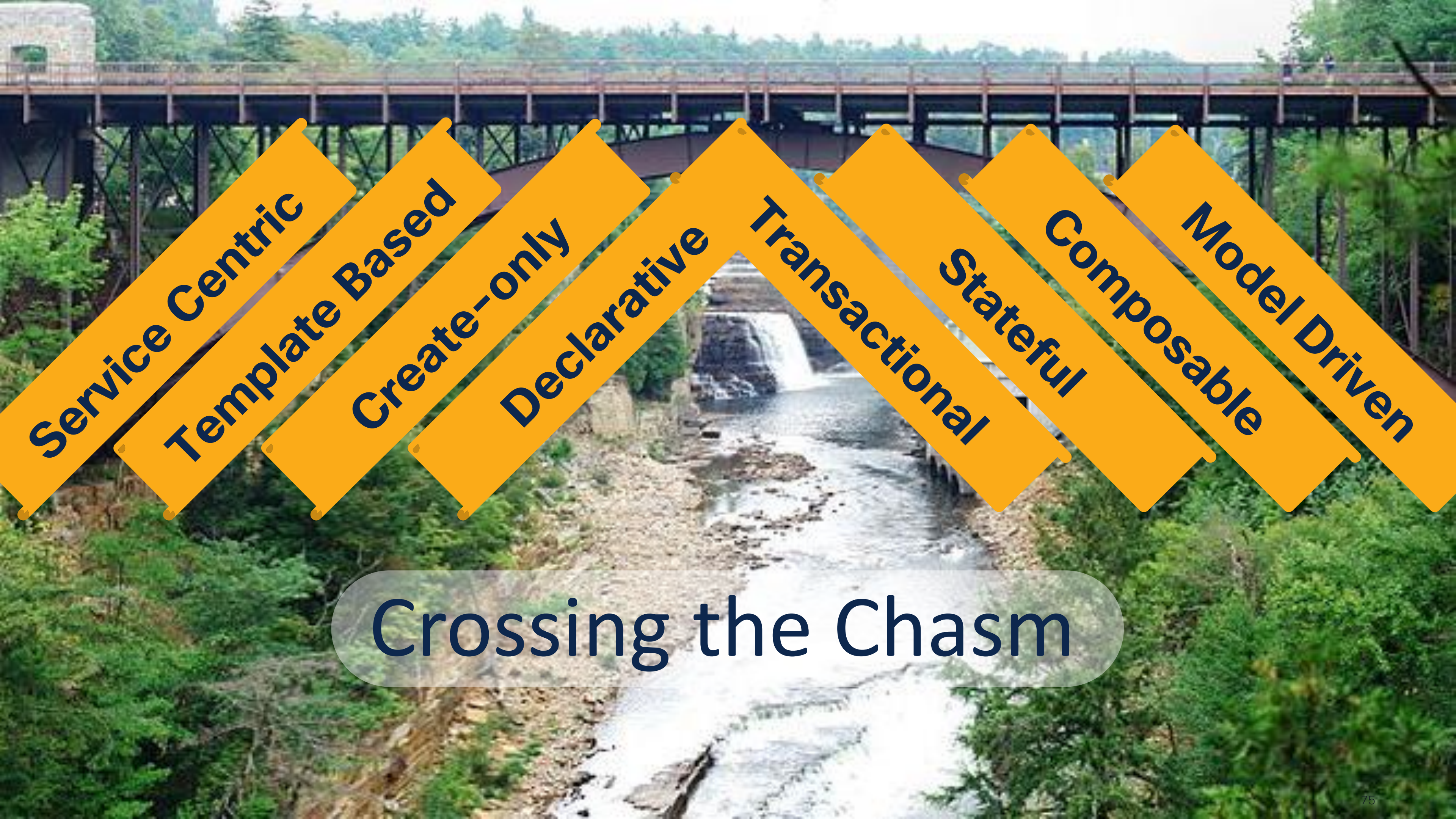
= Getting things done

Summary

Protocol Deep-dive

- CLI is still used a lot, but is expensive to automate against
- NETCONF is currently the most powerful management protocol out there
- RESTCONF is the REST-based cousin of NETCONF
- gNMI is great for Telemetry
- The Network Element Drivers (NEDs) isolates operators and applications from the protocol shuffling (quirks)
- Templates hide vendor differences

Crossing the Chasm



Service Centric

Template Based

Create-only

Declarative

Transactional

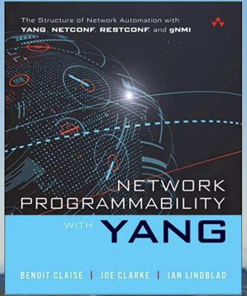
Stateful

Composable

Model Driven

cisco

The bridge to possible



The Structure of Network Automation with YANG, NETCONF, RESTCONF, and gNMI

NETWORK PROGRAMMABILITY WITH YANG

BENOIT CLAISE | JOE CLARKE | JAN LINDBLAD

“There are ways to cross the Network Management Automation chasm.”

NSO

The bridge to possible

Thank you

Complete your Session Survey

- Please complete your session survey after each session. Your feedback is important.
- Complete a minimum of 4 session surveys and the Overall Conference survey (open from Thursday) to receive your Cisco Live t-shirt.
- All surveys can be taken in the Cisco Events Mobile App or by logging in to the Session Catalog and clicking the "Attendee Dashboard” at https://www.ciscolive.com/emea/learn/sessions/session-catalog.html

Continue Your Education

Visit the Cisco Showcase for related demos.

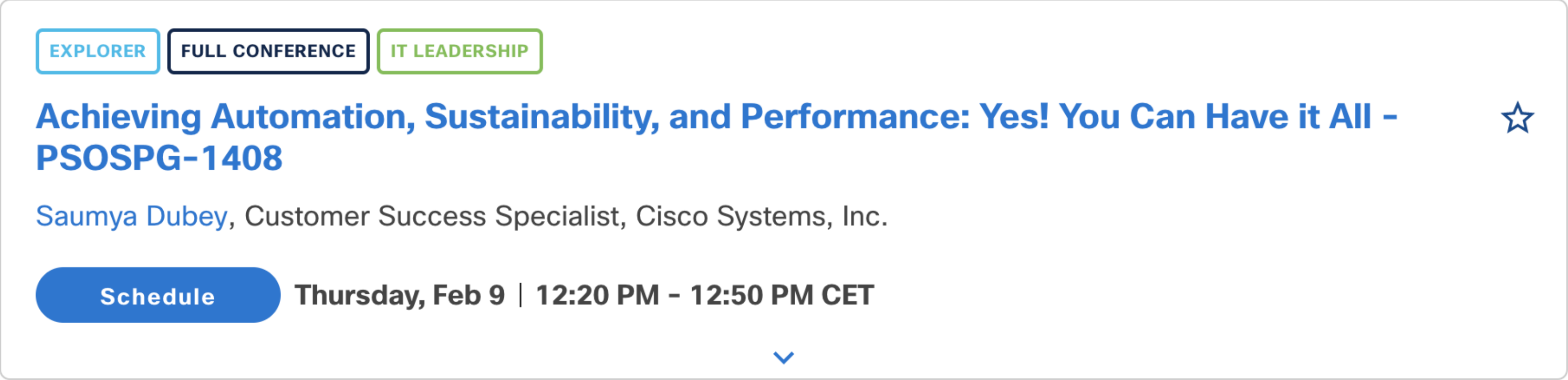
Book your one-on-one Meet the Engineer meeting.

Attend any of the related sessions at the DevNet, Capture the Flag, and Walk-in Labs zones.

Visit the On-Demand Library for more sessions at ciscolive.com/on-demand.

CISCO Live!

Afternoon NSO Sessions, NOW and 1PM

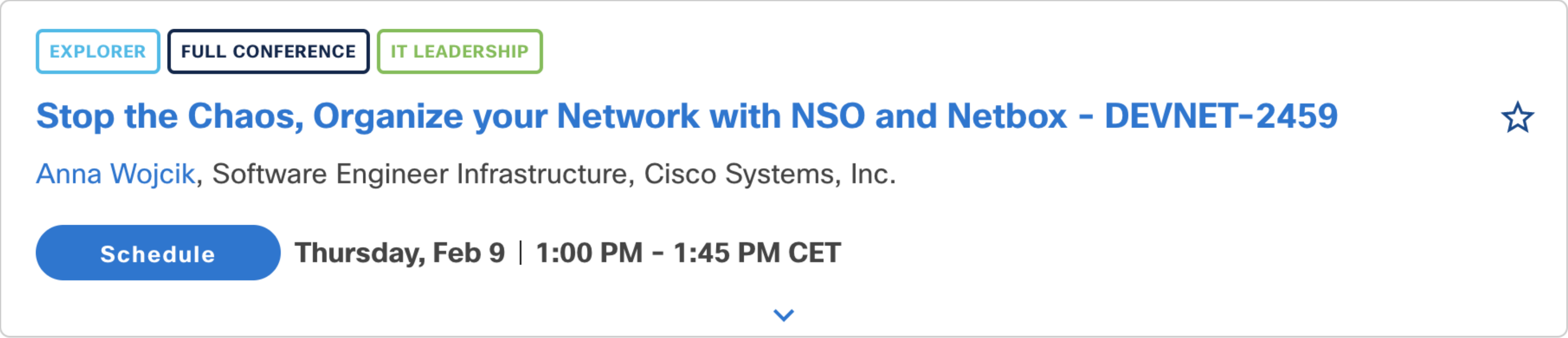


EXPLORER FULL CONFERENCE IT LEADERSHIP

Achieving Automation, Sustainability, and Performance: Yes! You Can Have it All - PSOSPG-1408

Saumya Dubey, Customer Success Specialist, Cisco Systems, Inc.

Schedule **Thursday, Feb 9 | 12:20 PM - 12:50 PM CET**



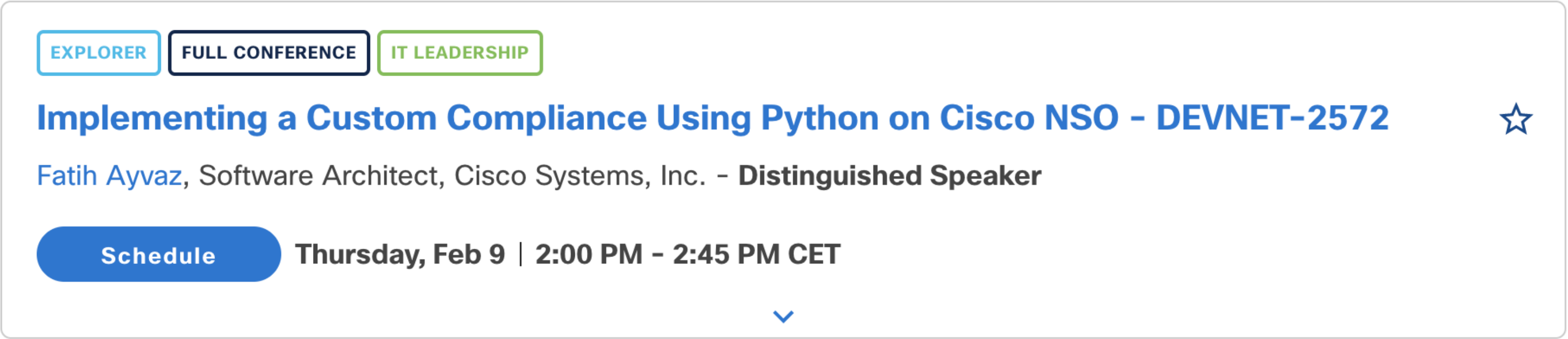
EXPLORER FULL CONFERENCE IT LEADERSHIP

Stop the Chaos, Organize your Network with NSO and Netbox - DEVNET-2459

Anna Wojcik, Software Engineer Infrastructure, Cisco Systems, Inc.

Schedule **Thursday, Feb 9 | 1:00 PM - 1:45 PM CET**

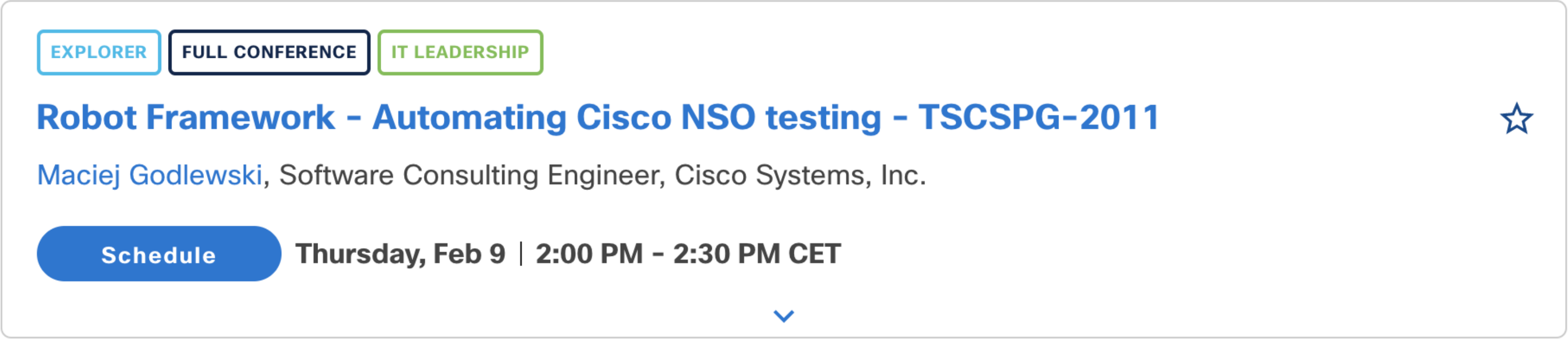
Afternoon NSO Sessions, 2PM



Implementing a Custom Compliance Using Python on Cisco NSO - DEVNET-2572

Fatih Ayvaz, Software Architect, Cisco Systems, Inc. - **Distinguished Speaker**

Thursday, Feb 9 | 2:00 PM - 2:45 PM CET

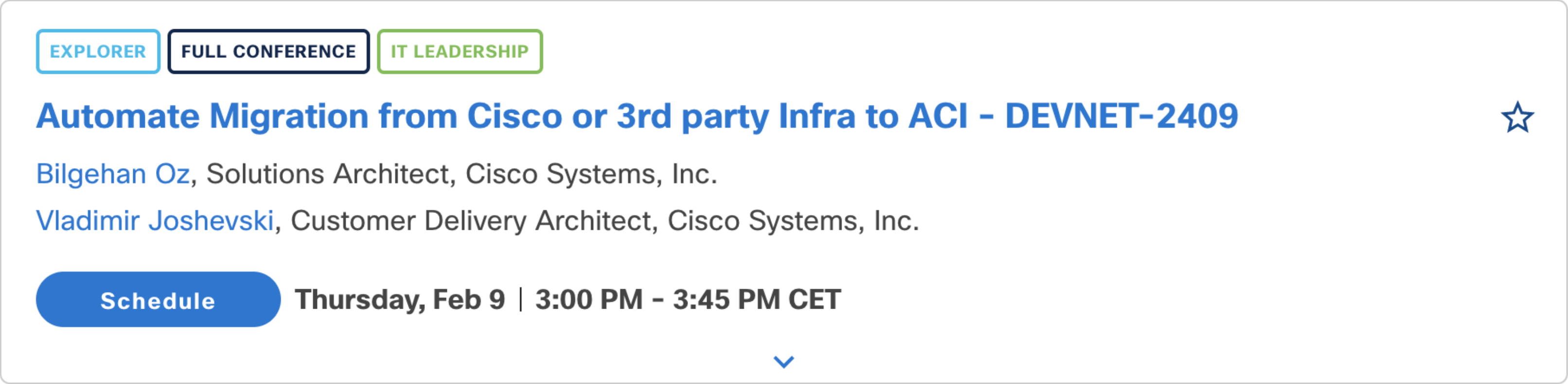


Robot Framework - Automating Cisco NSO testing - TSCSPG-2011

Maciej Godlewski, Software Consulting Engineer, Cisco Systems, Inc.

Thursday, Feb 9 | 2:00 PM - 2:30 PM CET

Afternoon NSO Sessions, 3PM



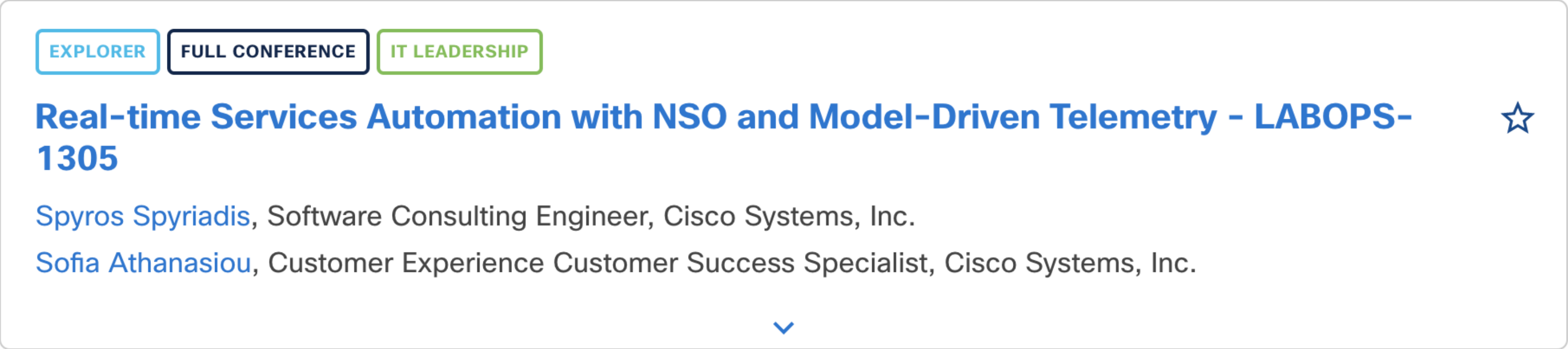
Automate Migration from Cisco or 3rd party Infra to ACI - DEVNET-2409

Bilgehan Oz, Solutions Architect, Cisco Systems, Inc.

Vladimir Joshevski, Customer Delivery Architect, Cisco Systems, Inc.

Thursday, Feb 9 | 3:00 PM - 3:45 PM CET

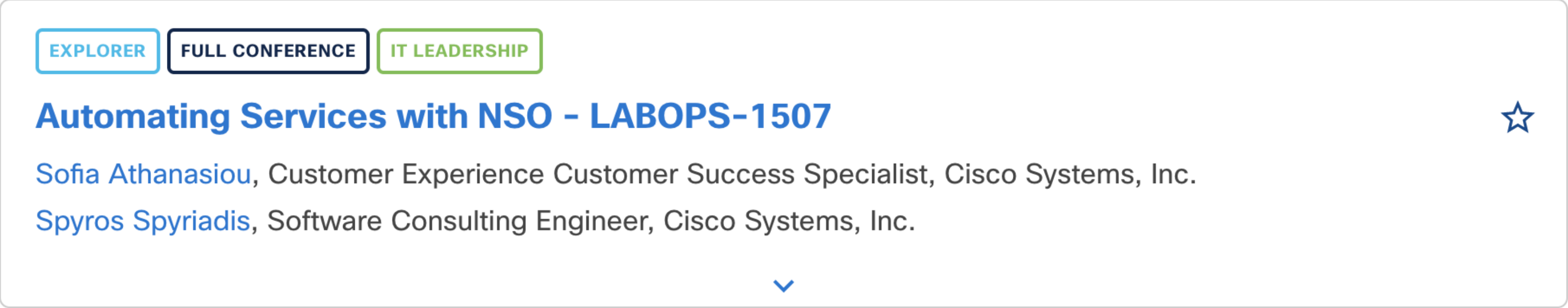
NSO Walk-in Labs



Real-time Services Automation with NSO and Model-Driven Telemetry - LABOPS-1305

Spyros Spyriadis, Software Consulting Engineer, Cisco Systems, Inc.

Sofia Athanasiou, Customer Experience Customer Success Specialist, Cisco Systems, Inc.



Automating Services with NSO - LABOPS-1507

Sofia Athanasiou, Customer Experience Customer Success Specialist, Cisco Systems, Inc.

Spyros Spyriadis, Software Consulting Engineer, Cisco Systems, Inc.

Developer Days
Automation

cisco
The bridge to possible

Can Automation enable the future Internet and help you transition to a greener network?

Service Developers, Operations and DevOps Engineers

May 9-11, Stockholm

Find information on how to submit a talk or register on The Developer Hub!
www.cisco.com/go/nsohub

cisco Live!