

The background is a vibrant, abstract graphic. It features a central bright white light source from which numerous colorful rays emanate, creating a sunburst or starburst effect. The rays transition through a spectrum of colors including yellow, orange, red, and various shades of blue and green. Overlaid on this are several large, semi-transparent, wavy shapes in similar color tones, giving the overall image a sense of motion and energy.

cisco *Live!*

Let's go

#CiscoLive



The bridge to possible

Advanced YANG Data Modeling for Cisco NSO

Bartosz Luraniec, Software Consulting Engineer
@lureek
DEVNET-3014

CISCO *Live!*

#CiscoLive

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 June 9, 2023.



<https://ciscolive.ciscoevents.com/ciscolivebot/#DEVNET-3014>

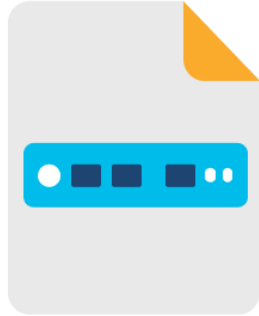
Agenda

- YANG - Introduction
- Path Navigation & Constraints
- Leaf
- Container
- List
- Use Cases
- YANG - Better View
- Summary

YANG Introduction

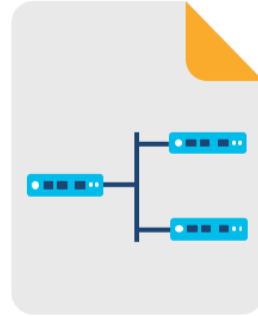


YANG in NSO: Devices & Services



Device Data Models

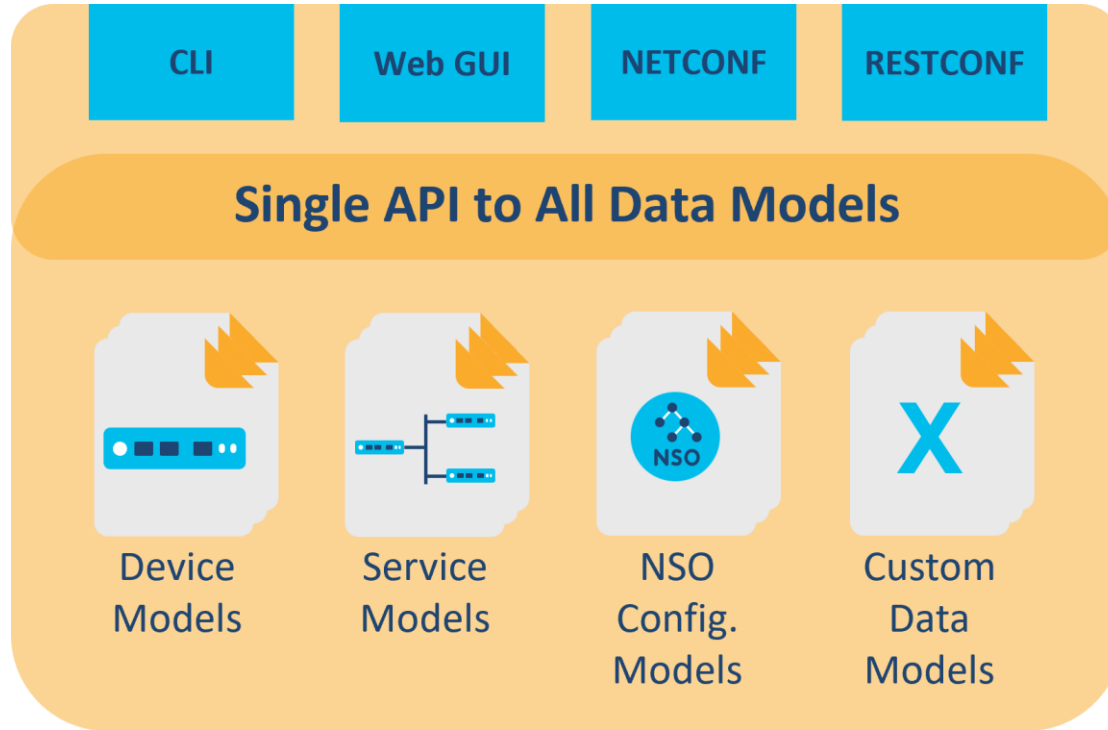
- Interface
- VLAN
- Device ACL
- Tunnel
- OSPF
- Etc.



Service Data Models

- Layer 3 MPLS VPN
- MP-BGP
- VRF
- Network ACL
- System Management
- Network Faults
- Etc.

YANG is everywhere



Path Navigation & Constraints



Access the YANG

How to get path for a specific resource in the Configuration Database (CDB) ?

```
admin@ncs# show running-config devices device IOS0 | display xpath
```

```
/devices/device[name='IOS0']/config/ios:interface/GigabitEthernet  
[name='0/0']/ip/address/primary/address 1.1.1.1  
/devices/device[name='IOS0']/config/ios:interface/GigabitEthernet  
[name='0/0']/ip/address/primary/mask 255.255.255.0  
...
```

path()

XPath uses path expressions to select nodes and also this expression can be used to return a collection of nodes, rather than a unique node.

```
list interface {  
    key "name";  
    leaf name {  
        type string;  
    }  
}  
leaf mgmt-interface {  
    type leafref {  
        path "../interface/name";  
    }  
}
```

current()

Starting point for an XPath location path.

```
leaf phase {  
    type enumeration {  
        enum Telnet;  
        enum SSH;  
    }  
}  
leaf mgmtVLAN {  
    when "current() ../../phase = 'SSH'";  
    type string;  
}
```

when()

Allows to expose node under given condition only.

```
leaf device {
  type leafref {
    path "/ncs:devices/ncs:device/ncs:name";
  }
}
container ios {
  when "/ncs:devices/ncs:device[ncs:name=current() / ../device] /
    ncs:platform/ncs:name = 'ios'";
  leaf device-description {
    tailf:info "device description";
    type string;
  }
}
```

must()

Constraint used to restrict and ensure that configuration meets the condition. It is using an XPath expression that must evaluate to true.

```
leaf device {  
    tailf:info "PE Router";  
    type leafref {  
        path "/ncs:devices/ncs:device/ncs:name";  
    }  
    must "starts-with(current(),'PE')" {  
        error-message  
        "Only PE devices can be selected.";  
    }  
}
```

deref()

Follows the reference defined by the first node in document order in the argument node-set, and returns the nodes it refers to.

```
leaf my-ip {  
  type leafref {  
    path "/server/ip";  
  }  
}  
leaf my-port {  
  type leafref {  
    path "/server[ip = current()]/../my-ip]/port";  
  }  
}
```

WITHOUT

```
leaf my-ip {  
  type leafref {  
    path "/server/ip";  
  }  
}  
leaf my-port {  
  type leafref {  
    path "deref(..my-ip)/../port";  
  }  
}
```

WITH

SAME RESULT

contains()

Evaluates if referenced leaf value in the model **contains** given string.

```
leaf port {  
    type string;  
}  
container encapsulation {  
    when "contains(current()../port, '.')" {  
        tailf:dependency "../port";  
    }  
}
```

starts-with()

Evaluates if referenced leaf value in the model **starts with** given string.

```
leaf type {  
  type enumeration {  
    enum test;  
    enum isp;  
    enum mobile_4G;  
    enum mobile_5G;  
  }  
}  
container classifier {  
  when "starts-with(current() / ../type, 'mobile')";  
  presence true;  
  leaf name {  
    type string;  
  }  
}
```


But we have way more...

- * last
- * position
- * count
- * id
- * local-name
- * namespace-uri
- * name
- * string
- * concat
- * starts-with
- * contains
- * substring-before
- * substring-after
- * substring

- * string-length
- * normalize-space
- * translate
- * boolean
- * nodeset-as-boolean
- * false
- * true
- * not
- * number
- * sum
- * floor
- * ceiling
- * round
- * re-match

- * string-compare
- * compare
- * current
- * deref
- * sort-by
- * enum-value
- * bit-is-set
- * min
- * max
- * avg
- * band
- * bor
- * bxor
- * bnot

Evaluate your XPath in NSO

Enable Devtools:

```
admin@ncs# devtools true  
admin@ncs# config
```

Evaluate Xpath:

```
admin@ncs (config)# xpath eval "/devices/device[contains(name, 'a') or  
contains(name, 'i')]"  
  
/devices/device[name='ios0']
```

Evaluate your XPath in NSO

String Examples:

```
admin@ncs (config) # xpath eval normalize-space(' test')  
test
```

```
admin@ncs (config) # xpath eval translate('test_xyz', '_', '-')  
test-xyz
```

```
admin@ncs (config) # xpath eval compare('abc','abc')  
0
```

```
admin@ncs (config) # xpath eval string-length('test')  
4
```

Evaluate your XPath in NSO

String Examples:

```
admin@ncs (config) # xpath eval substring('test_xyz', 2, 2)
```

```
es
```

```
admin@ncs (config) # xpath eval substring-before('aa-bb', '-')
```

```
aa
```

```
admin@ncs (config) # xpath eval substring-after('aa-bb', '-')
```

```
bb
```

```
admin@ncs (config) #
```

```
xpath eval re-match('1.22.333', '\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}')
```

```
true
```

```
admin@ncs (config) # xpath eval concat('area', '-SW')
```

```
area-SW
```

Evaluate your XPath in NSO

Node-Set Examples:

```
admin@ncs (config) # xpath eval count(devices/device)
```

```
3
```

```
admin@ncs (config) # xpath eval "devices/device[last()]"
```

```
/devices/device[name='JUN0']
```

```
admin@ncs (config) # xpath eval devices/device[position()=2]
```

```
/devices/device[name='IOSXR0']
```

```
admin@ncs (config) # xpath eval name(devices/device)
```

```
ncs:device
```

Evaluate your XPath in NSO

Numeric Examples:

```
admin@ncs (config) # xpath eval floor(4.69)
```

4

```
admin@ncs (config) # xpath eval ceiling(4.69)
```

5

```
admin@ncs (config) # xpath eval "10 div 5"
```

2

```
admin@ncs (config) # xpath eval 2*2
```

4

Evaluate your XPath in NSO

Boolean Examples:

```
admin@ncs (config) # xpath eval boolean(0)
false
```

```
admin@ncs (config) # xpath eval not(0)
true
```

```
admin@ncs (config) # xpath eval true()
true
```

```
admin@ncs (config) # xpath eval false()
false
```

Leaf



leafref

Hint to values for the leaf existing somewhere in the model.
It **enforces** data validation.

```
leaf vrf-name {  
  type leafref {  
    path "/ncs:devices/ncs:device[ncs:name=current()/../device]/  
        ncs:config/iosxr:vrf/iosxr:vrf-list/iosxr:name";  
  }  
}
```

```
admin@ncs(config)# SERVICE vrf-name ?  
Possible completions:  
VRF1  VRF2  
admin@ncs(config)# SERVICE vrf-name VRF3 ?  
Possible completions:  
Error: "VRF3" is an invalid value.
```

tailf:non-strict-leafref

Hint to values for the leaf existing in the model but **without enforcing** any data validation nor evaluation logic.

```
leaf vrf-name {  
    tailf:non-strict-leafref {  
        path "/ncs:devices/ncs:device[ncs:name=current()/../device]/  
            ncs:config/iosxr:vrf/iosxr:vrf-list/iosxr:name";  
    }  
    type string;  
}
```

```
admin@ncs (config) # SERVICE vrf-name VRF3 ?
```

YANG will accept value that is not existing under referenced leaf.

tailf:default-ref

Sets a leaf to the value of another leaf unless it is explicitly set.

```
leaf BVI {  
  type uint16 {  
    range 1..4094;  
  }  
}  
container encapsulation {  
  choice encapsulation-choice {  
    container dot1q {  
      presence true;  
      leaf tag {  
        type uint16 {  
          range 1..4094;  
        }  
        tailf:default-ref "../..../BVI";  
      }  
    }  
  }  
}
```

tailf:cli-show-with-default

Allows to display leaf in configuration even if it is set to the default value.

```
leaf mtu-size {  
    tailf:cli-show-with-default;  
    type uint16;  
    default 1500;  
}
```

```
admin@ncs# show run SERVICE  
SERVICE  
  interface GigabitEthernet 0/0  
    mtu-size      1500  
  !  
  !
```

WITH

```
admin@ncs# show run SERVICE  
SERVICE  
  interface GigabitEthernet 0/0  
  !  
  !
```

WITHOUT

tailf:cli-expose-key-name

By default when inputting a list element – key name doesn't appear in the CLI. We can expose it.

```
list SERVICE {  
    key "device";  
    leaf device {  
        tailf:cli-expose-key-name;  
        type string;  
    }  
}
```

WITHOUT

```
admin@ncs (config) # SERVICE IOS1
```

WITH

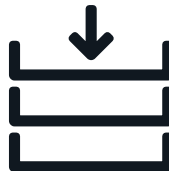
```
admin@ncs (config) # SERVICE device IOS1
```

tailf:hidden

This statement can be used to hide a node from some, or all, northbound interfaces (CLI and Web UI).

```
leaf calculated-lsps {  
    tailf:hidden true;  
    type string;  
}
```

Use it as a storage!



Container



presence

If you want to make container optional (where you have mandatory leaf elements). If anything is configured under container – the container will be true, if nothing is configured under the container – then it is absent (false).

```
container routing-protocol {  
  container bgp {  
    presence true;  
    // MANDATORY LEAF elements  
  }  
  container ospf {  
    presence true;  
    // MANDATORY LEAF elements  
  }  
}
```


tailf:cli-flatten-container

```
container customer {  
    tailf:cli-flatten-container;  
    leaf name {  
        type string;  
    }  
    leaf address {  
        type string;  
    }  
}  
container interface {  
    leaf int-type {  
        type string;  
    }  
    leaf int-id {  
        type string;  
    }  
}
```

Allows the CLI to exit the container and continue to input from the parent container when all leaf elements in the current container has been set.

```
admin@ncs (config) # SERVICE customer name  
CLIENT address KRK interface int-type GE  
int-id 0/1
```

WITH

```
admin@ncs (config) # SERVICE customer name  
CLIENT address KRK  
admin@ncs (config) # exit  
admin@ncs (config) # interface int-type GE  
int-id 0/1
```

WITHOUT

tailf:cli-add-mode

Creates a mode of the container. It can be used in config nodes only.

```
container vrf-config {  
  tailf:cli-add-mode;  
  leaf vrf-name {  
    type string;  
  }  
  leaf local-as-number {  
    tailf:info "BGP AS";  
    type enumeration {  
      enum 12345;  
      enum 23456;  
    }  
  }  
}
```

```
admin@ncs (config) # SERVICE IOS0 vrf-config  
<ENTER>  
admin@ncs (config-vrf-config) # ?  
Possible completions:  
  local-as-number  
  vrf-name
```

WITH

```
admin@ncs (config) # SERVICE IOS0 vrf-config  
-----^  
syntax error: incomplete path  
admin@ncs (config) # SERVICE IOS0 vrf-config ?  
Possible completions:  
  local-as-number  
  vrf-name
```

WITHOUT

List



min-elements/max-elements

Specify minimum/maximum number of elements that have to occur in this list or leaf-list.

```
list FastEthernet {  
    tailf:info "FastEthernet port";  
    key port;  
    max-elements 1;  
    min-elements 1;  
    leaf port {  
        type string;  
    }  
}
```

tailf:cli-flat-list-syntax

Allow operators to enter the leaf-list values without the brackets.

```
leaf-list vrf-name {  
    tailf:cli-flat-list-syntax;  
    type string;  
}
```

```
admin@ncs (config) # SERVICE vrf-name [ VRF1 VRF2 ]
```

WITHOUT

```
admin@ncs (config) # SERVICE vrf-name VRF1 VRF2
```

WITH

tailf:cli-compact-syntax

```
container vrf-config {  
  tailf:cli-compact-syntax;  
  leaf vrf {  
    type string;  
  }  
  
  leaf local-as-number {  
    tailf:info "AS Number";  
    type enumeration {  
      enum 12345;  
      enum 23456;  
    }  
  }  
}
```

Use the compact representation for this node in the 'show running-configuration' command (all leaf elements are shown in a single line).

```
admin@ncs# show running-config SERVICE  
SERVICE service  
  vrf-config vrf vrf1 local-as-number 12345  
!
```

WITH

```
admin@ncs# show running-config SERVICE  
SERVICE service  
  vrf-config vrf vrf1  
  vrf-config local-as-number 12345  
!
```

WITHOUT

tailf:cli-sequence-commands

```
list link {  
    tailf:cli-sequence-commands;  
    key "id";  
    leaf id {  
        mandatory true;  
        type string;  
    }  
    leaf source {  
        type inet:ipv4-address;  
    }  
    leaf destination {  
        type inet:ipv4-address;  
    }  
}
```

Enforces the exact order while inputting parameters. The order is the same as in the YANG model.

```
admin@ncs(config)# SERVICE link primary ?  
Possible completions:  
    source <cr>
```

WITH

```
admin@ncs(config)# SERVICE link primary ?  
Possible completions:  
    destination source <cr>
```

WITHOUT

tailf:cli-incomplete-command

```
container dhcp {  
  leaf port {  
    tailf:cli-incomplete-command;  
    type uint16;  
  }  
  leaf subnet {  
    type inet:ipv4-prefix;  
  }  
}
```

Specifies that an auto-rendered command should be considered incomplete. Can be used to prevent "<cr>" from appearing in the completion list for optional internal nodes.

```
admin@ncs(config)# SERVICE dhcp port 12 ?  
Possible completions:  
  subnet
```

WITH

```
admin@ncs(config)# SERVICE dhcp port 12 ?  
Possible completions:  
  subnet <cr>
```

WITHOUT

Use Cases



Vendor agnostic interfaces modelling



Vendor agnostic interfaces modelling

```
container ios-xr {  
    tailf:cli-drop-node-name;  
    when "/ncs:devices/ncs:device[ncs:name=current()/../hostname]/  
        ncs:platform/ncs:name = 'ios-xr'";  
    list interfaces {  
        key "interface-type port";  
        leaf interface-type {  
            type enumeration {  
                enum GigabitEthernet;  
                enum TenGigabitEthernet;  
                enum BundleEthernet;  
            }  
        }  
        leaf port {  
            type string;  
        }  
    }  
}
```

Vendor agnostic interfaces modelling

```
container junos {  
  tailf:cli-drop-node-name;  
  when "/ncs:devices/ncs:device[ncs:name=current()/../hostname]/  
    ncs:platform/ncs:name = 'junos'";  
  list interfaces {  
    key "interface-type port";  
    leaf interface-type {  
      mandatory true;  
      type enumeration {  
        enum xe-;  
        enum ge-;  
        enum ae-;  
      }  
    }  
    leaf port {  
      type string;  
    }  
  }  
}
```

CLI Autocomplete Action



```
list l3vpn-service {
  key "device";
  leaf device {
    type string;
  }
  list interface {
    key "interface-type port";
    uses ncs:service-data;
    ncs:servicepoint l3vpn-servicepoint;
    leaf interface-type {
      type enumeration {
        enum GigabitEthernet;
        enum TenGigE;
      }
    }
    leaf port {
      type string;
      tailf:cli-completion-actionpoint "l3vpn-cli-completion-action";
    }
  }
}
```

```
from _ncs.dp import action_reply_completion

## ACTION

def cb_completion(self, uinfo, cli_style, token, completion_char, kp,
                   cmdpath, cmdparam_id, simpleType, extra):

    ## OPENED MAAPI TRANSACTION
    result_list = []
    strkp = str(kp)

    device_name = strkp.split("l3vpn{")[1].split(" ")[0]
    interface_type = strkp.split("interface")[1].split("{")[1].split("}")[0]
    interface_list = getattr(root.devices.device[device_name].config.cisco_ios
                             _xr__interface, interface_type)
    for interface in interface_list:
        result_list.append((0, interface.id, None))

action_reply_completion(uinfo, result_list)
```

YANG Better view



Yanger

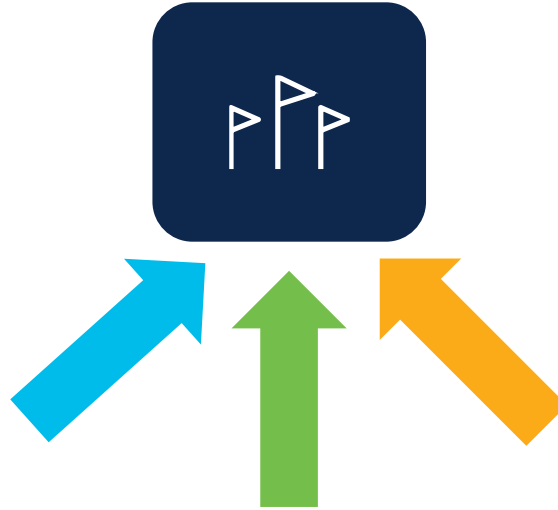
```
[developer@nso ~]$ yanger /var/opt/ncs/packages/cisco-ios-cli-6.67/src/yang/ietf-interfaces.yang -f tree
module: ietf-interfaces
  +--rw interfaces
  |   +--rw interface* [name]
  |   |   +--rw name                string
  |   |   +--rw description?       string
  |   |   +--rw type                identityref
  |   |   +--rw enabled?           boolean
  |   |   +--rw link-up-down-trap-enable? enumeration {if-mib}?
  |   +--ro interfaces-state
  |   |   +--ro interface* [name]
  |   |   |   +--ro name            string
  |   |   |   +--ro type            identityref
  |   |   |   +--ro admin-status    enumeration {if-mib}?
  |   |   |   +--ro oper-status     enumeration
  |   |   |   +--ro last-change?    yang:date-and-time
  |   |   |   +--ro if-index       int32 {if-mib}?
  |   |   |   +--ro phys-address?   yang:phys-address
  |   |   |   +--ro higher-layer-if* interface-state-ref
  |   |   |   +--ro lower-layer-if* interface-state-ref
  |   |   |   +--ro speed?         yang:gauge64
  |   |   |   +--ro statistics
  |   |   |   |   +--ro discontinuity-time yang:date-and-time
  |   |   |   |   +--ro in-octets?      yang:counter64
  |   |   |   |   +--ro in-unicast-pkts? yang:counter64
  |   |   |   |   +--ro in-broadcast-pkts? yang:counter64
```

Summary



Summary

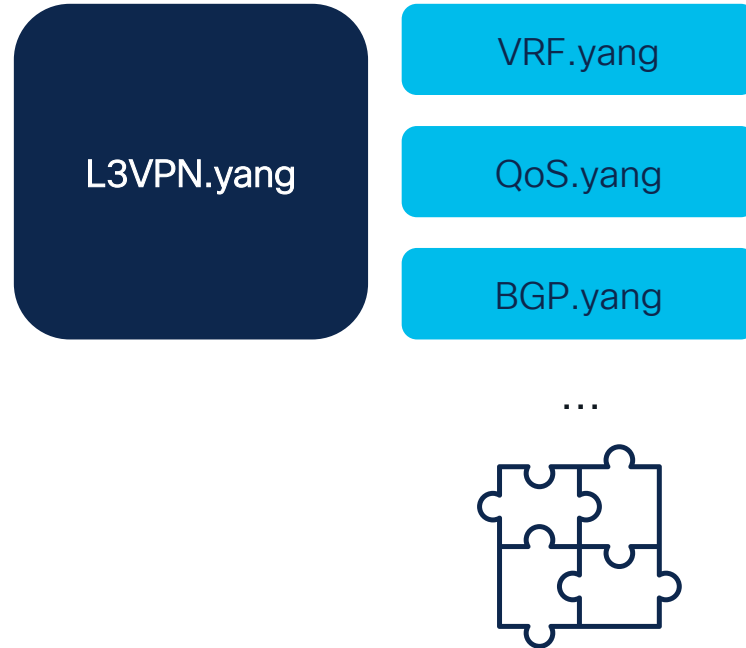
In YANG there are many different ways to accomplish the same goal.



But always follow the **3xM** rule ...


1. Be Modular

- ✓ Use groupings (e.g. interfaces)
- ✓ Decouple modules (e.g. separate VRF, QoS, Routing Protocol from L3VPN)
- ✓ Re-use modules across packages



2. Be Meticulous

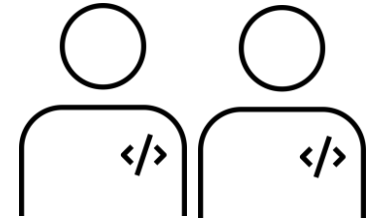
- ✓ Use range, length, pattern etc. to:
 - avoid inconsistency between Service model and NEDs
 - support future tooling around testing (payload generation)
- ✓ Get inspired from NEDs ...
but do not copy-paste
- ✓ Declare your modules version



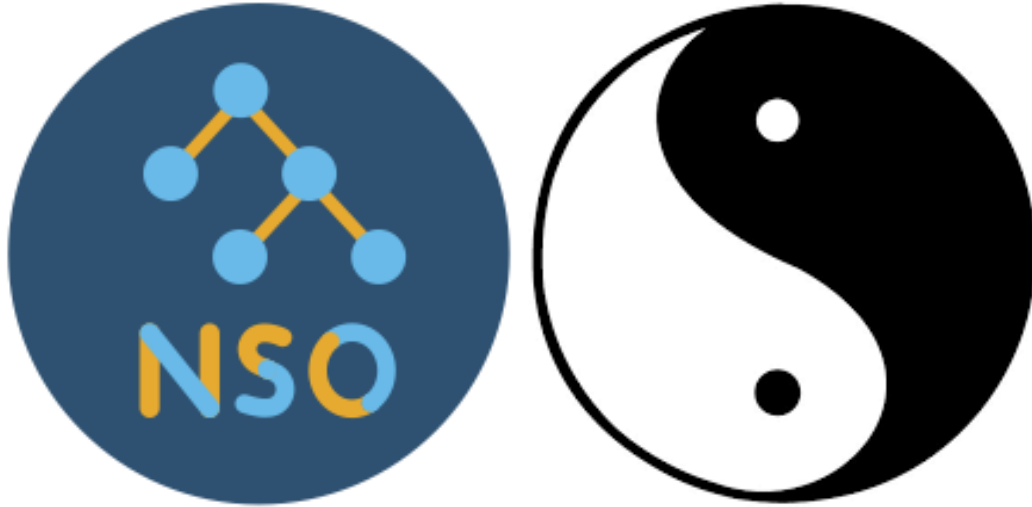
```
container acl { ...  
  leaf acl-description {  
    type string {  
      length "0..64";  
      pattern "[0-9a-zA-Z]*";  
    }  
    description "Purpose of ACL";  
  }  
}
```

3. Be Mindful!

- ✓ **Keep it simple** for the Operator
 - test flow of the CLI
 - use learnt tricks:
 - containers with **tailf:cli-drop-node-name**, vendor-specific config
 - **tailf:cli-sequence-commands**, speed up operator input flow
 - **tailf:cli-completion-actionpoint**, complex leafref at runtime
 - And many more!
- ✓ Prevent Operator mistakes in Service inputs
 - use range, length, pattern etc.



YIN-YANG



“Automation is as good as the Data Models”

Fill out your session surveys!



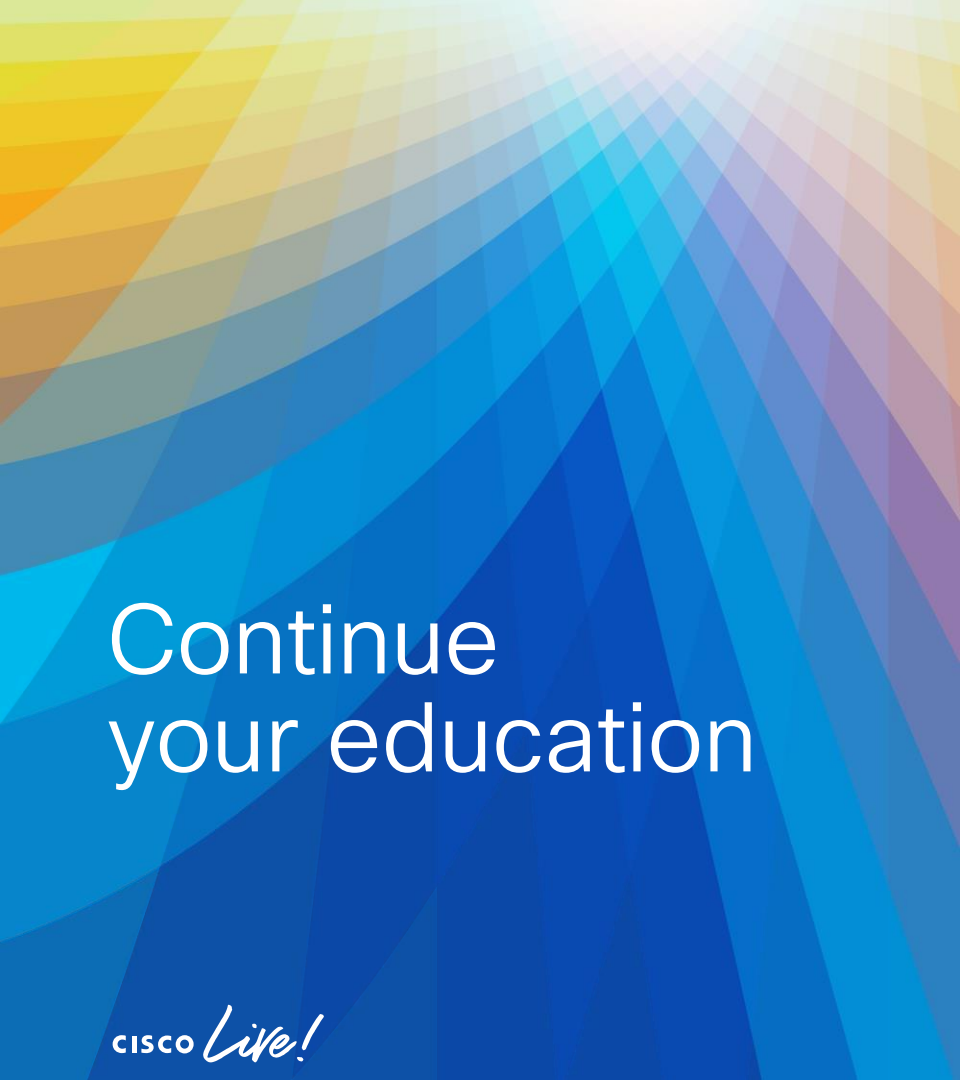
Attendees who fill out a minimum of four session surveys and the overall event survey will get **Cisco Live-branded socks** (while supplies last)!



Attendees will also earn 100 points in the **Cisco Live Challenge** for every survey completed.



These points help you get on the leaderboard and increase your chances of winning daily and grand prizes



Continue your education

CISCO *Live!*

- Visit the Cisco Showcase for related demos
- Book your one-on-one Meet the Engineer meeting
- Attend the interactive education with DevNet, Capture the Flag, and Walk-in Labs
- Visit the On-Demand Library for more sessions at www.CiscoLive.com/on-demand



The bridge to possible

Thank you

CISCO *Live!*

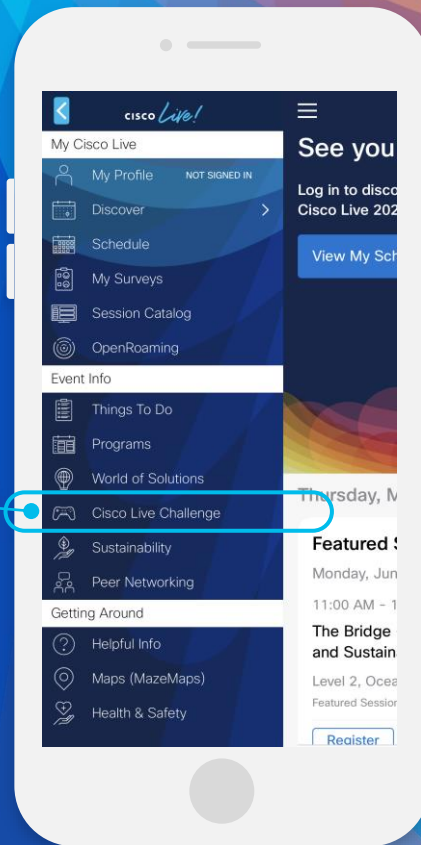
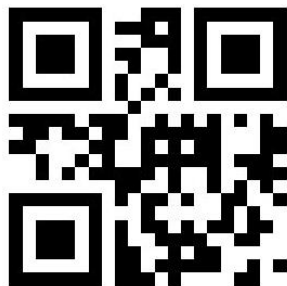
#CiscoLive

Cisco Live Challenge

Gamify your Cisco Live experience!
Get points for attending this session!

How:

- 1 Open the Cisco Events App.
- 2 Click on 'Cisco Live Challenge' in the side menu.
- 3 Click on View Your Badges at the top.
- 4 Click the + at the bottom of the screen and scan the QR code:



The background is a vibrant, abstract graphic. It features a central bright white light source from which numerous colorful rays emanate, creating a sunburst or starburst effect. The rays transition through a spectrum of colors: yellow, orange, red, and then various shades of blue and green. Overlaid on this are large, flowing, wavy shapes in similar colors, giving the impression of liquid or smoke being illuminated by the light. The overall effect is dynamic and energetic.

cisco *Live!*

Let's go

#CiscoLive