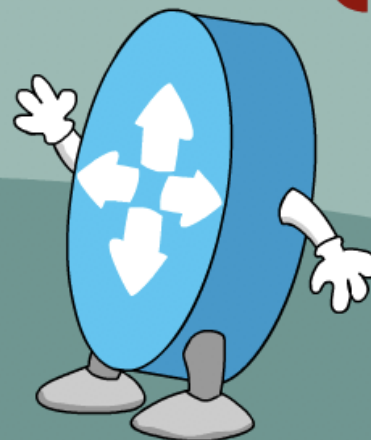




NETDEVOPS {LIVE!}



DEVNET

Designing and Developing Network Services with NSO

Kevin Corbin

.*

@kecorbin

Season 2, Talk 6

<https://developer.cisco.com/netdevops/live>



<http://cs.co/ndl>

Help us track NetDevOps Live Interest!

Agenda

- NSO Overview / Review
- Introduction to NSO Service Packages
- A different spin on YANG
- Service Package Design
- Service Package Development
- Demo Time

NSO The Network Wide API and CLI



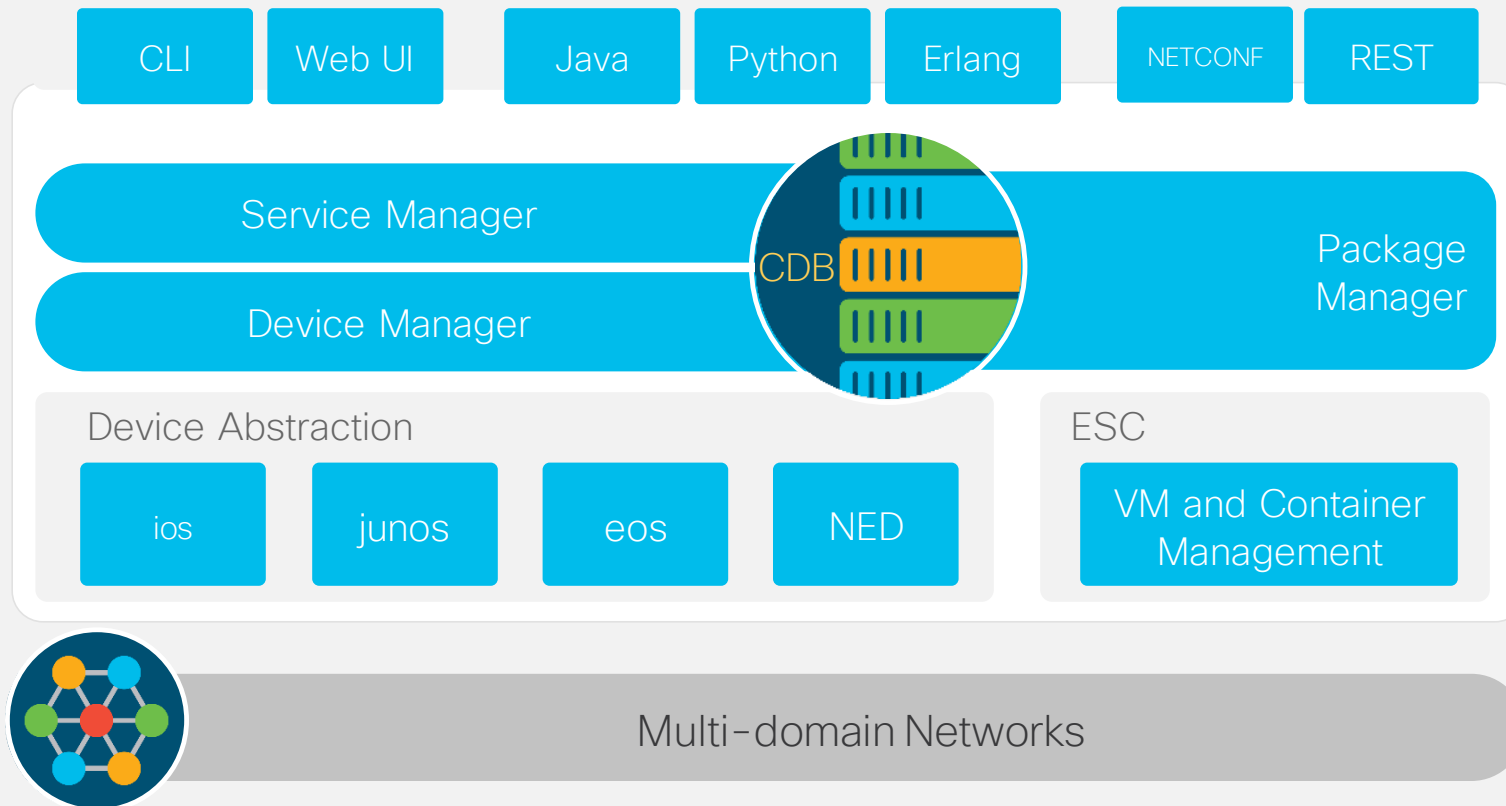
Fixes these chronic issues:

- Lack of automation, Managing device configuration
- Quality issues in delivery
- Inflexibility to change existing configuration (create and delete only)
- CLI Scripting—inflexible and high fallout

BRKDEV-1267

- Exposes a YANG-Based Configuration Database
- device manager do all the tedious stuff
 - sync-from/sync-to/compare Devices
- Rich set of Northbound APIs rendered from the database / devices
- Consistent and Network-wide CLI, UI, REST
- Start with CLI but gradually introduce others e.g. Python, REST, for scripting tasks
- Transaction-safe operations and rollback!
- **Configuration Management AND Accurate network configuration state**
- Choice of interface is up to the **consumer!!**

Cisco NSO - Architecture



- **Network-wide** orchestration across multi-domain, multi-vendor and multi-layer
- Centralized **policy** and **services**
- **Model-driven**, end-to-end service lifecycle and **customer experience** focused
- **Application Server** which runs apps in the form of **packages**
- **Seamless** integration with order managers and ITSM systems
- Loosely-coupled and **modular** architecture leveraging open APIs and **standard** protocols

Packages

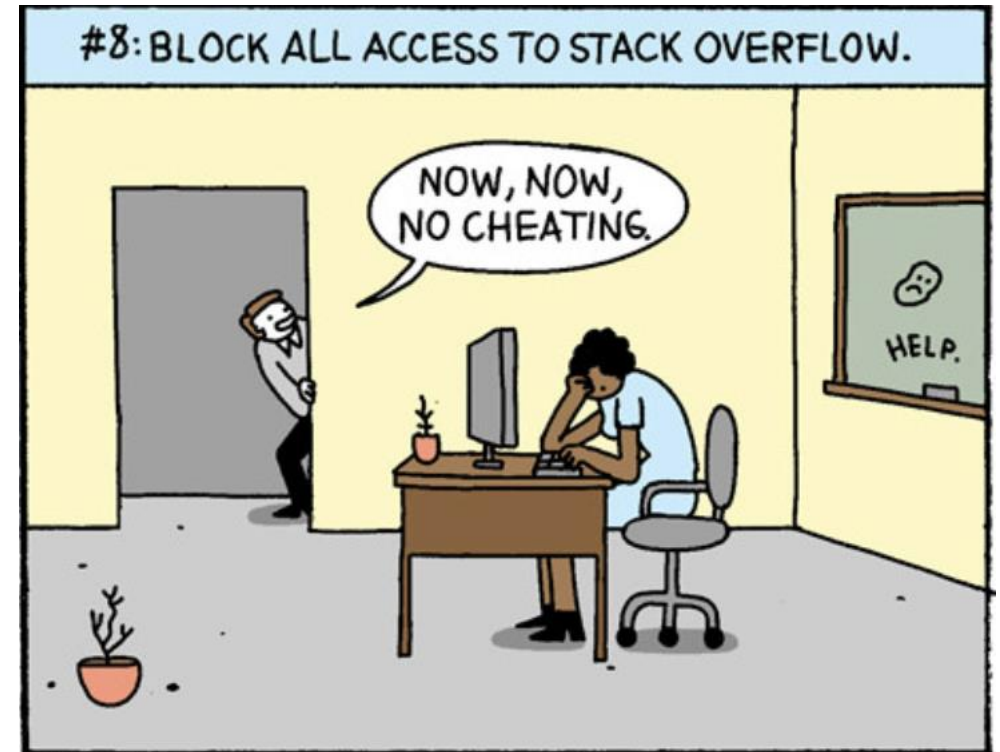
The fundamental building block of NSO



- Three primary package types
 - Network Element Driver (NED) – model of network device + way to manage it
 - **Services** – defines something that you do or provide with a network
 - Tools – other packages that don't fall into the above categories (e.g. device discovery)
- Two consumption models
 - Customer Facing Service (CFS) – stuff business people care about
 - Resource Facing Service (RFS) – stuff infrastructure people care about
- Core Function Packs
 - A curated set of packages for a given use case.
 - SD-WAN
 - NFVO
 - Secure Agile Exchange

Why Develop Services?

- Provide an abstraction between devices/vendors and **YOUR** business context
- Integration with existing tools/systems
 - DNS/IPAM/CMDB/ITSM/etc
- Enforce Policy / Constraints / Standards – easily remediate
- Visibility into inter-dependent services / configuration
- Accurate deployment across entire configuration lifecycle



<https://blog.toggl.com/troll-your-developer/>

Service Package Components

- Separation of Concerns
 - YANG – defines the services data model, API, CLI, and constraints in high level terms (architecture)
 - Logic – (**optional**) python/java code which collects additional information (variables), performs verification on instances (policy/governance)
 - Templates – how the service is rendered on one or more device/types (engineering)
 - package-meta-data.xml – minimum required NSO version, package dependencies, components provided (requirements.txt)
 - Tests
 - unit tests for your service (dev)
 - Custom validation scripts/playbooks for your deployed services (ops)

Config, Deploy, Validate w/ organizational, standards, process, best Practices

DAMN! THERE YOU GO AGAIN...

FORCING ME TO THINK!!!

memecrunch.com

What's YANG?

- YANG is a language
- YANG is a standard
- YANG is “open-sourced”

What is a service?

Customer Facing Service
"Who's wants to use it"



"Top Down"
a.k.a The Overlay

- Amazon VPC
- MPLS L3 VPN
- Data Center VXLAN/VLAN
- Branch Deployment / End-to-End / Multi-Domain

Resource Facing Service
"Who has to Manage it"



"Bottom Up"
a.k.a The Underlay

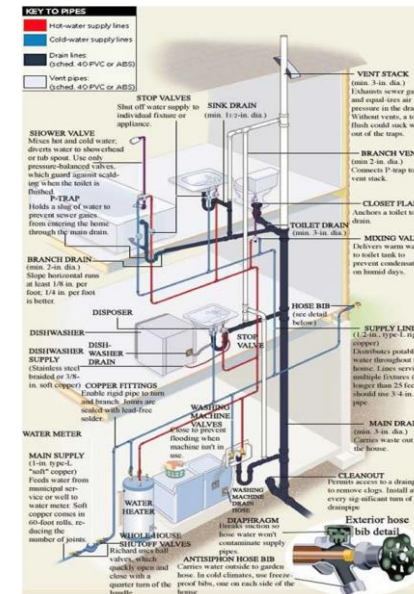
- PE Router
- Distribution Switch
- Data Center VXLAN Fabric
- DMZ/PCI/STIG Device
- SNMP
- openconfig-vlan.yang

What is a Model?

- Customer Facing

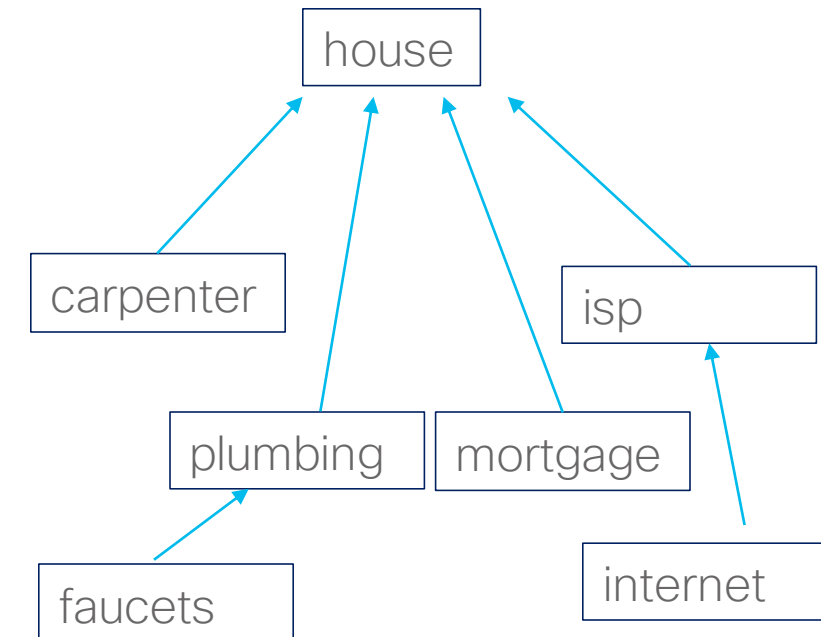


- Resource Facing



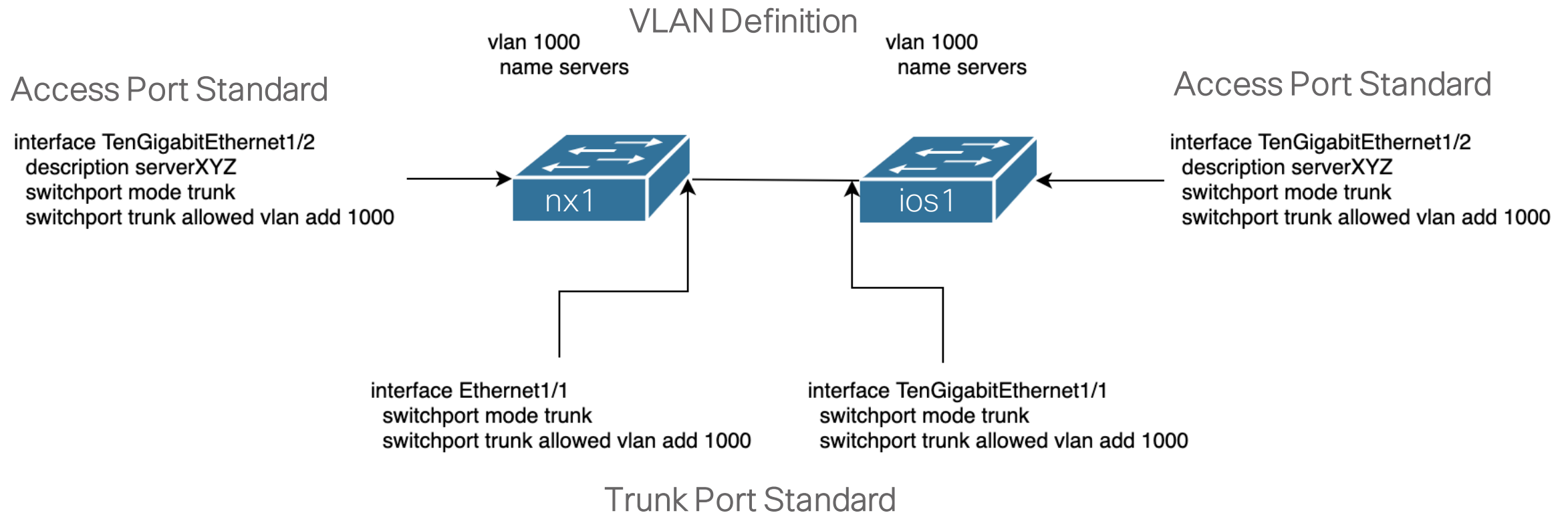
Model Granularity + Composition

- Orchestration composes various models to deliver final “product” to “customer”
- Granularity is chosen to
 - reduce the complexity of each model
 - allow different teams to own different models
 - allow independent life-cycles
- One top model describes the customer facing service
- Lower models describes intermediate resource facing services
- An Ecosystem of models



VLAN Provisioning – Current State

$\text{config}(\text{nx1}) + \text{config}(\text{ios1}) = \text{config}(\text{Vlan10})$



Where is the source of truth for how VLAN 1000 is configured?

What If???

```
vlan servers
id 1000
device ios1
  trunk-port 1/1
  !
  access-port 1/2
  description serverXYZ
  !
!
device nx1
  trunk-port 1/1
  !
  access-port 1/2
  description serverXYZ
  !
!
```

```
→ nso-vlan curl -u admin:admin -H "Accept: application/vnd.yang.data+json" http://localhost:8080/api/running/vlan/servers?deep
{
  "vlan:vlan": {
    "name": "servers",
    "id": 1000,
    "device": [
      {
        "device": "ios1",
        "trunk-port": [
          {
            "port": "1/1"
          }
        ],
        "access-port": [
          {
            "port": "1/2",
            "description": "serverXYZ"
          }
        ]
      },
      {
        "device": "nx1",
        "trunk-port": [
          {
            "port": "1/1"
          }
        ],
        "access-port": [
          {
            "port": "1/2",
            "description": "serverXYZ"
          }
        ]
      }
    ]
  },
  "operations": {
    "check-sync": "/api/running/vlan/servers/_operations/check-sync",
    "deep-check-sync": "/api/running/vlan/servers/_operations/deep-check-sync",
    "re-deploy": "/api/running/vlan/servers/_operations/re-deploy",
  }
}
```

“I want to turn the whole thing upside down”

– Jack Johnson



<https://github.com/kecorbin/nso-service-development>

NSO Development Environment

- Local NSO Instance
- netsim – Multi-Vendor Network Simulator
- ncs-make-package – generate code skeletons
- devtools – xpath testing
- logs

Getting Started

- `ncs-make-package` is provided to generate boilerplate service package code.

```
ncs-make-package --dest packages/vlan --service-skeleton python-and-template vlan
```

- Packages must be compiled **and** reloaded before they are available to NSO

```
make -C packages/vlan/src clean all
```

```
rm -rf ../load-dir java/src//  
mkdir -p ../load-dir  
mkdir -p java/src  
/Users/kecorbin/nso47/bin/ncsc `ls vlan-ann.yang` > /dev/null 2>&1 && echo "-a vlan-ann.yang" ` \  
-c -o ../load-dir/vlan.fxs yang/vlan.yang
```

- Must be re-compiled whenever the model changes, reloaded whenever template/logic changes occur

Reload Packages

```
admin@ncs# packages reload

>>> System upgrade is starting.
>>> Sessions in configure mode must exit to operational mode.
>>> No configuration changes can be performed until upgrade has completed.
>>> System upgrade has completed successfully.
reload-result {
    package cisco-ios
    result true
}
reload-result {
    package cisco-nx
    result true
}
reload-result {
    package vlan
    result true
}
admin@ncs#
```

Templates

- Map service configuration to actual device configuration
- XML files stored in *templates* subdirectory of package
- Easily generated from “golden” devices
- Supports processing instructions



Variables

- Variables from two sources can be used in templates
 - Service Model
 - Referenced as Xpath using the service as the context.
 - Logic
 - Computed / Retrieved using python/java/etc, and passed to the template during rendering phase

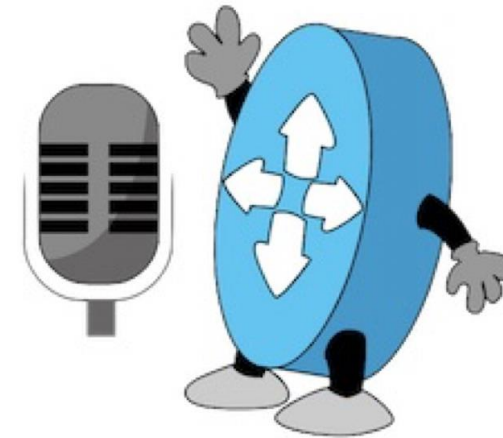
Template Processing Instructions

Syntax	Description
<code><?set variable = value?></code>	Allows to assign new variables or manipulate existing variable value. If used to create a new variable, then the scope of visibility of this variable is limited to the parent tag of the processing instruction or the current processing instruction block. Specifically, if a new variable is defined inside a loop, then it is discarded at the end of each iteration.
<code><?if {expression}?>...<?else?>...<?end?></code>	Processing instruction block that allows conditional execution based on the boolean result of the expression. For the detailed description see the section called "Conditional Statements"
<code><?foreach {expression}?>...<?end?></code>	The expression must evaluate to a (possibly empty) XPath node-set. The template engine will then iterate over each node in the node-set by changing the XPath current context node to this node and evaluating all children tags within this context. For the detailed description see the section called "Loop Statements"
<code><?for [variable = initial value]; {progress condition}; [variable = next value]?>...<?end?></code>	This processing instruction allows to iterate over the same set of template tags by changing a variable value. The variable visibility scope obeys the same rules as in the case of <code>set</code> processing instruction, except the variable value is carried over to the next iteration instead of being discarded at the end of each iteration. The square brackets indicate optional clauses, so only the condition expression is mandatory. For the detailed description see the section called "Loop Statements"
<code><?copy-tree {source}?></code>	This instruction is analogous to <code>copy_tree</code> function available in the MAAPI API. The parameter is an XPath expression which must evaluate to exactly one node in the data tree and indicates the source path to copy from. The target path is defined by the position of the <code>copy-tree</code> instruction in the template within the current context.
<code><?set-context-node {expression}?></code>	Allows to manipulate the current context node used to evaluate XPath expressions in the template. The expression is evaluated within the current XPath context and must evaluate to exactly one node in the data tree.
<code><?set-root-node {expression}?></code>	Allows to manipulate the root node of the XPath accessible tree. This expression is evaluated in an XPath context where the accessible tree is the entire datastore, which means that it is possible to select a root node outside the current accessible tree. The current context node remains unchanged. Just like with the <code>set-context-node</code> instruction the expression must evaluate to exactly one node in the data tree.
<code><?save-context name?></code>	Store both the current context node and the root node of the XPath accessible tree with <code>name</code> being the key to access it later. It is possible to switch to this context later using <code>switch-context</code> with the name. Multiple contexts can be stored simultaneously under different names. Using <code>save-context</code> with the same name multiple times will result in the stored context being overwritten.
<code><?switch-context name?></code>	Used to switch to a context stored using <code>save-context</code> with the specified name. This means that both the current context node and the root node of the XPath accessible tree will be changed to the stored values. <code>switch-context</code> does not remove the context from the storage and can be used as many times as needed, however using it with a name that does not exist in the storage would cause an error.

Documentation \$NCS_DIR/doc/html/nso_development/ch11s03s02.html

Summary, what did talk about?

- NSO Overview / Review
- Introduction to NSO Service Packages
- Service Package Design
- Service Package Development



Webinar Resource List

- Sample code from today
 - <https://github.com/kecorbin/nso-service-development>
- Getting Started/Download
 - <https://developer.cisco.com/docs/nso/#!getting-nso>
- Docs and Links
 - <https://developer.cisco.com/docs/nso/#!nso-fundamentals/nso-fundamentals>
 - <https://developer.cisco.com/site/nso/>
- Learning Labs
 - NSO usage for different users: <https://developer.cisco.com/learning/modules/nso>
 - NSO Basics by Cisco IT: <https://developer.cisco.com/learning/modules/nso-basics>
- DevNet Sandboxes
 - Mutli-IOs VIRL Sandbox <http://cs.co/sbx-multi>



Looking for more about NetDevOps?

- NetDevOps on DevNet
developer.cisco.com/netdevops
- NetDevOps Live!
developer.cisco.com/netdevops/live
- NetDevOps Blogs
blogs.cisco.com/tag/netdevops
- Network Programmability Basics Video Course
developer.cisco.com/video/net-prog-basics/



Got more questions? Stay in touch!



@CiscoDevNet



facebook.com/ciscocodevnet/

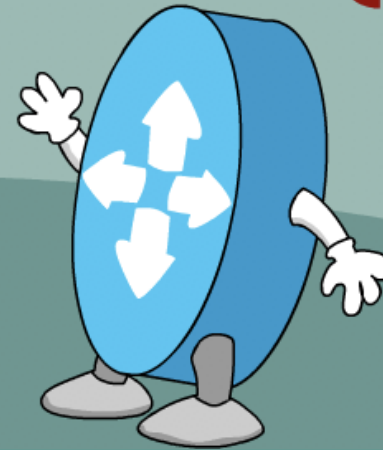


<http://github.com/CiscoDevNet>



NETDEVOPS {LIVE!}

 DEVNET



<https://developer.cisco.com/netdevops/live>
@netdevopslive 