

The World of NSO Web UI

DEVLIT-1259

Speakers:

Jorge Mira

Customer Success Specialist
Cross-Domain Automation

Table of Contents

1 - Cisco Network Services Orchestrator (NSO) Requirements.....	3
1.1 - Install NSO and Network Element Drivers (NEDs)	3
1.2 - Using Network Simulators (NETSIMs).....	7
1.3 – Configure Devices using NSO.....	9
1.4 – Using Rollback's	11
1.5 – Using NSO capabilities to detect Out-of-Band device configurations	13
1.6 - Create Device Groups and Device Templates.....	15
1.7 – Create a simple NSO Service	21
1.8 – NSO APIs	35
1.9 – Create Role Based and Resource Based Access Control rules.....	40

1 - Cisco Network Services Orchestrator (NSO) Requirements

In this workbook you will find a step-by-step guide that goes from the NSO installation to its operations basics, always using the WEB-UI.

NSO can be installed on a **MAC**, or **Linux**. With the **NSO 6.1** launch we started offering as well native **docker** support. (guide can be found in nso admin-guide)

Opening this link, you will find the operating system requirements and how to install them.

<https://developer.cisco.com/docs/nsos/#getting-and-installing-nsos/requirements>

1.1 - Install NSO and Network Element Drivers (NEDs)

Download NSO for your Operating System and the NEDs (for trial purposes, only NSO 6.0 is available)

<https://developer.cisco.com/docs/nsos/#getting-and-installing-nsos/download-your-nsos-free-trial-installer-and-cisco-neds>

The screenshot shows the Cisco Software Download page. In the top navigation bar, there is a breadcrumb trail: Downloads Home / Cloud and Systems Management / Service Management and Orchestration / Network Services Orchestrator / Network Services Orchestrator Free Trial. Below the navigation, there is a search bar and buttons for 'Expand All' and 'Collapse All'. A dropdown menu shows 'Latest Release' selected, with '6.0' highlighted. Under 'File Information', there is a table listing several files:

	Release Date	Size	Actions
Cisco Network Services Orchestrator Changes nso-6.0-freetrial.CHANGES Advisories	17-Nov-2022	0.06 MB	Download
Cisco Network Services Orchestrator Readme nso-6.0-freetrial README Advisories	17-Nov-2022	0.01 MB	Download
Cisco Network Services Orchestrator Darwin Installer nso-6.0-freetrial.darwin.x86_64.signed.bin Advisories	17-Nov-2022	194.48 MB	Download
Cisco Network Services Orchestrator Documentation nso-6.0-freetrial.doc.tar.gz Advisories	17-Nov-2022	42.09 MB	Download
Cisco Network Services Orchestrator Linux Installer nso-6.0-freetrial.linux.x86_64.signed.bin Advisories	17-Nov-2022	197.55 MB	Download
Cisco Network Services Orchestrator Installation Guide nso_installation_guide-6.0-freetrial.pdf Advisories	17-Nov-2022	0.28 MB	Download
Cisco NSO Cisco ASA NED ncs-6.0-cisco-asa-6.16-freetrial.signed.bin Advisories	16-Nov-2022	6.81 MB	Download
Cisco NSO Cisco IOS NED ncs-6.0-cisco-ios-6.88-freetrial.signed.bin Advisories	16-Nov-2022	51.47 MB	Download
Cisco NSO Cisco IOS XR NED ncs-6.0-cisco-iosxr-7.43-freetrial.signed.bin Advisories	16-Nov-2022	37.47 MB	Download
Cisco NSO Cisco Nexus NED ncs-6.0-cisco-nx-5.23.6-freetrial.signed.bin Advisories	16-Nov-2022	9.45 MB	Download

Unpack NSO

```
user@Pc # % sh nso-6.0-freetrial.darwin.x86_64.signed.bin
Unpacking...
Verifying signature...
Retrieving CA certificate from
http://www.cisco.com/security/pki/certs/crcam2.cer ...
```

```
Successfully retrieved and verified crcam2.cer.  
Retrieving SubCA certificate from  
http://www.cisco.com/security/pki/certs/innerspace.cer ...  
Successfully retrieved and verified innerspace.cer.  
Successfully verified root, subca and end-entity certificate chain.  
Successfully fetched a public key from tailf.cer.  
Successfully verified the signature of nso-6.0.darwin.x86_64.installer.bin  
using tailf.cer
```

After unpacking and verifying the signature you will get the installer.bin

Execute the “nso-6.1.darwin.x86_64.installer.bin” with the “—local-install” flag and choose the installation directory.

```
user@Pc # % sh nso-6.0.darwin.x86_64.installer.bin --local-install ~/NSO-DD  
INFO Using temporary directory  
/var/folders/48/yvryv4r96jq9n8gt3hpm13w40000gn/T//ncs_installer.11084 to  
stage NCS installation bundle  
INFO Unpacked ncs-6.0 in /Users/user/NSO-DD  
INFO Found and unpacked corresponding DOCUMENTATION_PACKAGE  
INFO Found and unpacked corresponding EXAMPLE_PACKAGE  
INFO Found and unpacked corresponding JAVA_PACKAGE  
INFO Generating default SSH hostkey (this may take some time)  
INFO SSH hostkey generated  
INFO Generating self-signed certificates for HTTPS  
INFO Environment set-up generated in /Users/user/NSO-DD/ncsrc  
INFO NSO installation script finished  
INFO Found and unpacked corresponding NETSIM_PACKAGE  
INFO NCS installation complete
```

Go to the installation folder and source NSO

```
user@Pc # % cd ~/NSO-DD  
user@Pc NSO-DD % source ncsrc
```

Execute the NSO Setup script

```
user@Pc NSO-DD % ncs-setup --dest nso-instance
```

Change directory to the “nso-instance” folder and execute the command “ncs”

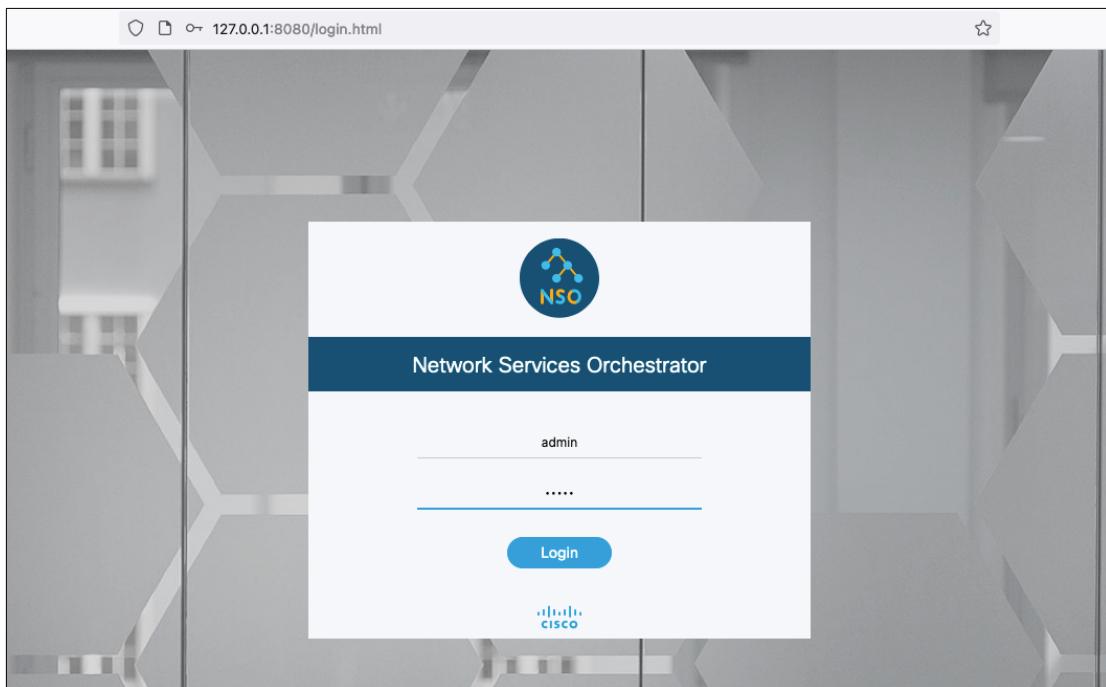
```
user@Pc NSO-DD % cd nso-instance  
user@Pc nso-instance % ncs
```

Check if NSO is running

```
user@Pc nso-instance % ncs --status | grep started  
status: started
```

Login into NSO via WEBUI using the default port 8080 (HTTP) <http://127.0.0.1:8080/login.html>

Use the default username and password (admin : admin)



Go to the “Configuration Editor two” (The older version is deprecated but utilization is still available via webui)

Since it's a fresh installation we've no packages, let's add the NED we've downloaded.

Go back to the folder where you downloaded the NED and repeat the same process we did for the NSO installation. Execute the “ncs-6.0-cisco-iosxr-7.43-freetrial.signed”.

```
user@Pc # % sh ncs-6.0-cisco-iosxr-7.43-freetrial.signed.bin
```

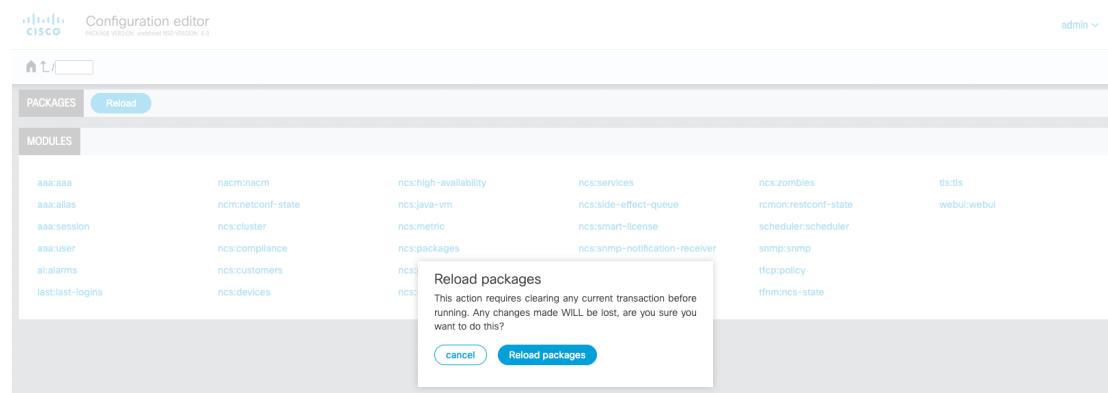
```

Unpacking...
Verifying signature...
Retrieving CA certificate from
http://www.cisco.com/security/pki/certs/crcam2.cer ...
Successfully retrieved and verified crcam2.cer.
Retrieving SubCA certificate from
http://www.cisco.com/security/pki/certs/innerspace.cer ...
Successfully retrieved and verified innerspace.cer.
Successfully verified root, subca and end-entity certificate chain.
Successfully fetched a public key from tailf.cer.
Successfully verified the signature of ncs-6.0-cisco-iosxr-7.43.tar.gz
using tailf.cer

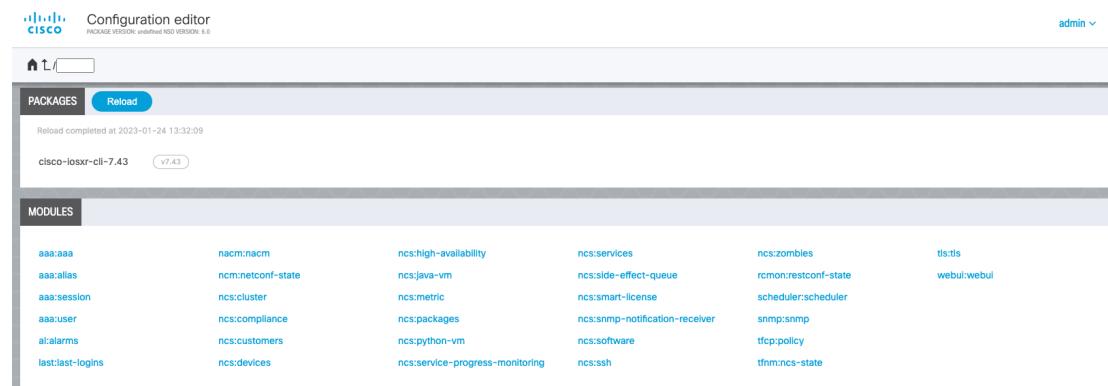
```

Copy the “.tar.gz” file to the /nso-instance/packages folder

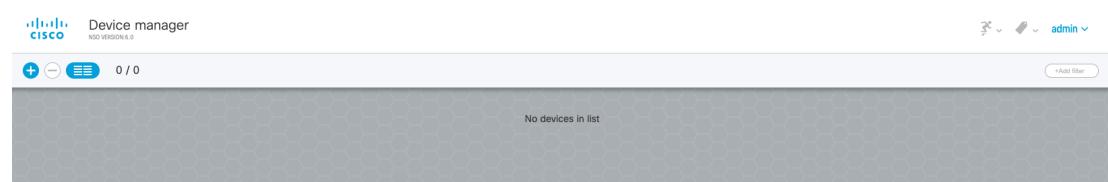
Go back to NSO WEB UI and Click on “Reload” packages button



As the result of the “packages reload” you should now be able to see your “cisco-iosxr” package.



Now that we've added our NED let's go to the “Device Manager”.



There are still no devices on the list, we can add them manually by clicking on + button, or, use scripts / APIs to add them.

1.2 - Using Network Simulators (NETSIMs)

What we will do now is use the NETSIM (Network Simulator) capability to simulate devices and add them to NSO. This is a very useful tool that NSO brings for us to test device configurations.

Going back to the CLI (where we did the “source ncsrc” command) and use the command “ncs-netsim” to create simulated devices.

```
user@Pc nso-instance % ncs-netsim create-network packages/cisco-  
iosxr-cli-7.43.tar.gz 4 ios-xr-  
DEVICE ios-xr-0 CREATED  
DEVICE ios-xr-1 CREATED  
DEVICE ios-xr-2 CREATED  
DEVICE ios-xr-3 CREATED
```

Using the command “ncs-netsim list” we’re able to see all the communication ports that we can use to connect to these simulated devices.

```
user@Pc nso-instance % ncs-netsim list  
ncs-netsim list for /Users/user/NSO-DD/nso-instance/netsim  
  
name=ios-xr-0 netconf=12022 snmp=11022 ipc=5010 cli=10022  
dir=/Users/user/NSO-DD/nso-instance/netsim/ios-xr-/ios-xr-0  
name=ios-xr-1 netconf=12023 snmp=11023 ipc=5011 cli=10023  
dir=/Users/user/NSO-DD/nso-instance/netsim/ios-xr-/ios-xr-1  
name=ios-xr-2 netconf=12024 snmp=11024 ipc=5012 cli=10024  
dir=/Users/user/NSO-DD/nso-instance/netsim/ios-xr-/ios-xr-2  
name=ios-xr-3 netconf=12025 snmp=11025 ipc=5013 cli=10025  
dir=/Users/user/NSO-DD/nso-instance/netsim/ios-xr-/ios-xr-3
```

Start the devices.

```
user@Pc nso-instance % ncs-netsim start  
DEVICE ios-xr-0 OK STARTED  
DEVICE ios-xr-1 OK STARTED  
DEVICE ios-xr-2 OK STARTED  
DEVICE ios-xr-3 OK STARTED
```

To test the connectivity to one device you can use the command “ncs-netsim cli-c ios-xr-0”. Let’s now, make a quick Loopback 0 configuration.

```
user@Pc nso-instance % ncs-netsim cli-c ios-xr-0  
  
admin connected from 127.0.0.1 using console on Pc.local  
ios-xr-0# conf  
Entering configuration mode terminal  
ios-xr-0(config)# interface Loopback 0  
ios-xr-0(config-if)# ipv4 address 1.1.1.1 255.255.255.255  
ios-xr-0(config-if)# commit
```

```
Commit complete.
ios-xr-0(config-if)# end
```

Netsims have limited capabilities on show commands. You can try the show-running-config.

```
ios-xr-0# show running-config
admin
  exit-admin-config
!
interface Loopback 0
  no shutdown
  ipv4 address 1.1.1.1 255.255.255.255
exit
ios-xr-0# exit
```

To add the 4 devices to NSO we need to export them to an XML file and use the “ncs_load” command to load them. -l (load) -m (merge)

```
user@Pc nso-instance % ncs-netsim ncs-xml-init > devices.xml
user@Pc nso-instance % ncs_load -l -m devices.xml
```

Go back to NSO and click on Device Manager. Your recently created devices should be there.

Device manager													
	name	address	port	type	services	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration
<input type="checkbox"/>	ios-xr-0	127.0.0.1	10022	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration
<input type="checkbox"/>	ios-xr-1	127.0.0.1	10023	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration
<input type="checkbox"/>	ios-xr-2	127.0.0.1	10024	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration
<input type="checkbox"/>	ios-xr-3	127.0.0.1	10025	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration

The first thing that you need to do after adding a device to NSO is to execute the “sync-from” so you update NSO CDB with the device configuration.

Instead of doing the sync-from device by device we can select all devices (click on the box on the left of the header “name”) and click on sync-from.

Device manager														
	name	address	port	type	services	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration	
<input checked="" type="checkbox"/>	ios-xr-0	127.0.0.1	10022	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration	
<input checked="" type="checkbox"/>	ios-xr-1	127.0.0.1	10023	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration	
<input checked="" type="checkbox"/>	ios-xr-2	127.0.0.1	10024	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration	
<input checked="" type="checkbox"/>	ios-xr-3	127.0.0.1	10025	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration	

If the result is green, means that now NSO CDB is successfully updated with the device configurations.

Device manager														
	name	address	port	type	services	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration	
<input checked="" type="checkbox"/>	ios-xr-0	127.0.0.1	10022	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from ✓	sync-to	compare-config		configuration	
<input checked="" type="checkbox"/>	ios-xr-1	127.0.0.1	10023	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from ✓	sync-to	compare-config		configuration	
<input checked="" type="checkbox"/>	ios-xr-2	127.0.0.1	10024	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from ✓	sync-to	compare-config		configuration	
<input checked="" type="checkbox"/>	ios-xr-3	127.0.0.1	10025	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from ✓	sync-to	compare-config		configuration	

1.3 – Configure Devices using NSO

To configure a device using NSO Web UI we just need to click on the device name. For example, let's enter in ios-xr-0 and verify if the loopback 0 interface that we've configured is already there.

The screenshot shows the NSO Configuration editor interface. At the top, there are tabs for Configuration editor, Edit config, Config, Operdata, Actions, Widgets (None), Containers, Lists, and admin. The main area displays the configuration for the device 'ios-xr-0'. The configuration includes:

name:	ios-xr-0	authgroup:	default
address:	127.0.0.1		
port:	10022		

Below these fields, there are several sections with collapsed buttons:

- ssh
- device-type
- ssh-algorithms
- state
- config

To navigate in the configurations you can use the Top bar (where I typed “confi”) or scroll down and look for what you want to configure

The screenshot shows the NSO Configuration editor interface with the 'config' tab selected. The top bar includes tabs for Configuration editor, Edit config, Config, Operdata, Actions, Widgets (None), and admin. The main area displays the detailed configuration for the device 'ios-xr-0'. The configuration includes:

name	ios-xr-0	authgroup	default
local-user		device-profile	
description		connect-timeout	Valid range: 1 .. 4294967
trace-output	<input checked="" type="checkbox"/> file <input type="checkbox"/> external	write-timeout	Valid range: 1 .. 4294967
address-choice		trace	Select...

Below these fields, there are additional configuration sections:

- out-of-sync-commit-behaviour: Select...
- snmp-notification-address: [Input field]
- device:
 - address: 127.0.0.1
 - port: 10022

You can follow the path and verify the Loopback 0 configuration

Full Path: /ncs:devices/device{ios-xr-0}/config/cisco-ios-xr:interface/Loopback{0}/ipv4/address/

config

cisco-ios-xr-interface

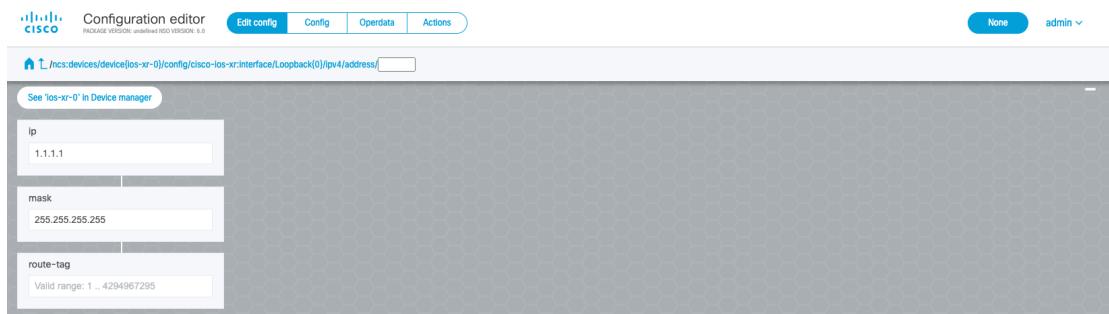
Loopback -> 0

ipv4

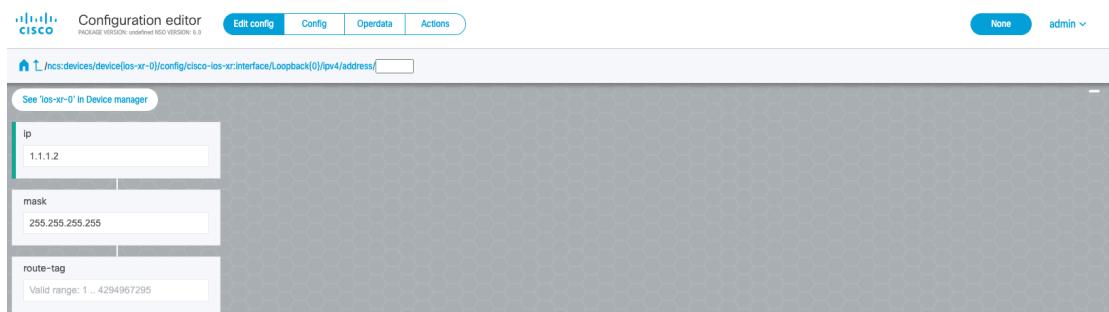
address-choice -> address



We can verify that the IP address we configured previously in the device is shown here.

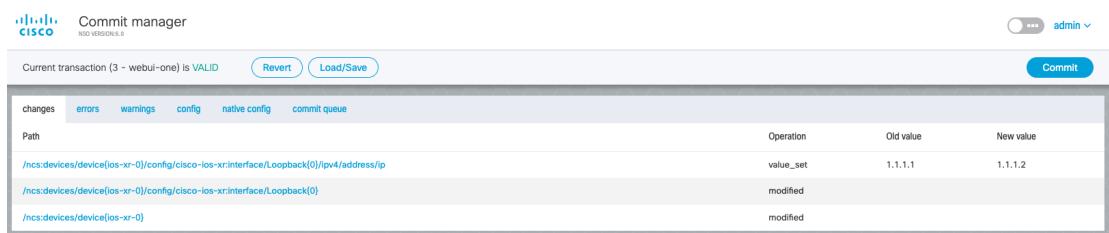


How do we change configuration values? We just need to edit the field that we want, let's change the IP address.



You can notice there is a green bar on the left of the tile that you edited. This means there is a candidate configuration on-going, and this configuration will only be pushed to the device after you commit.

Let's go the Commit Manager.



On the Top we've many options, from "changes" to "commit queue", click on "config" so you can see the changes that will be performed in a diff view like on GitHub.

If we agree with the changes, we click on commit.

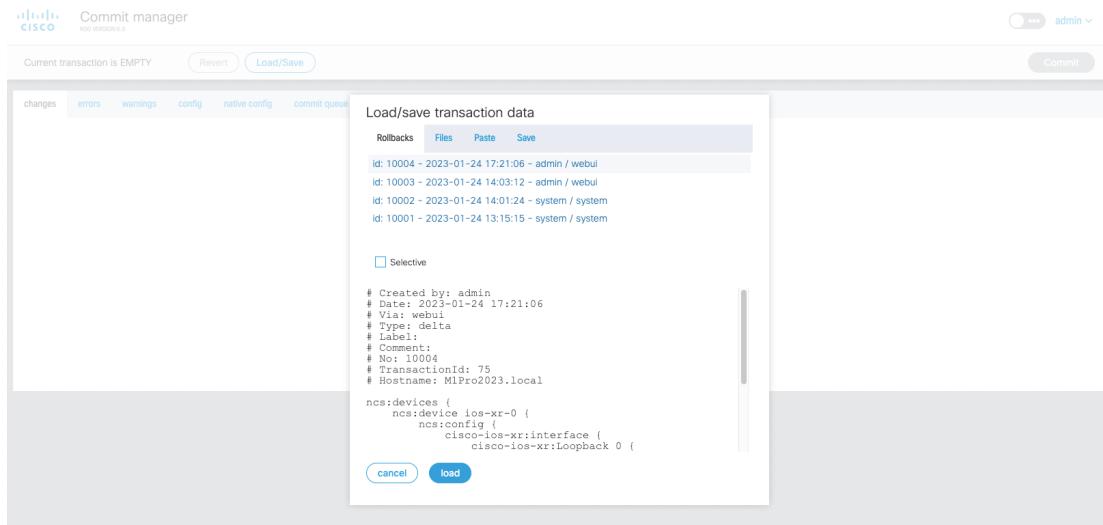
After our commit is done, we can see a rollback is created. This is because, every time we make a commit in NSO we create a rollback file that allow us to revert what we did.

If we go back to CLI of the netsim and execute the “show run”, we can see that our device now have the new IP on the Loopback Interface

```
ios-xr-0# show run
admin
exit-admin-config
!
interface Loopback 0
no shutdown
ipv4 address 1.1.1.2 255.255.255.255
exit
```

1.4 – Using Rollback's

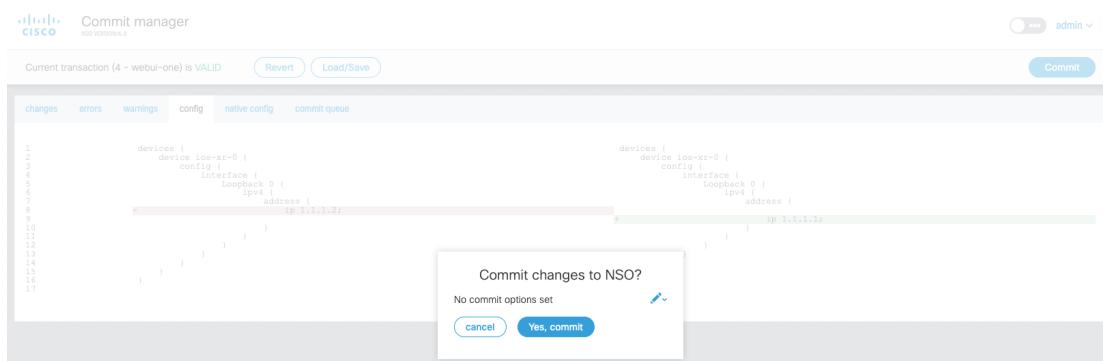
To Rollback this we just need to go back to Commit Manager, click on “Load/Save” Button and choose the rollback file that we want.



We can see the user that made the commit and what northbound interface was used. In this case was the user admin via Web UI. We click on load, then config, and we can see that the change that we've made to the Loopback 0 address will be reverted.



If we're ok with the change we hit the commit button and confirm.



After the commit we can check on the device that the change was reverted.

```
ios-xr-0# show run
admin
exit-admin-config
!
interface Loopback 0
no shutdown
ipv4 address 1.1.1.1 255.255.255.255
exit
```

1.5 – Using NSO capabilities to detect Out-of-Band device configurations

So, now let's see one of the main NSO capabilities (check-sync, sync-from, sync-to).

Login into ios-xr-0 device using SSH or “ncs-netsim clic- ios-xr-0” and configure the VTP mode off.

```
user@Pc nso-instance % ncs-netsim cli-c ios-xr-0
ios-xr-0# config
Entering configuration mode terminal
ios-xr-0(config)# vtp mode
  client      Set the device to client mode.
  off         Set the device to off mode.
  server      Set the device to server mode.
  transparent  Set the device to transparent mode.
ios-xr-0(config)# vtp mode off
ios-xr-0(config)# commit
Commit complete.
```

Since this change was done out of NSO we can go to NSO WEB UI / CLI and see if the device is in sync.

name	address	port	type	services	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration
ios-xr-0	127.0.0.1	10022	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	configuration
ios-xr-1	127.0.0.1	10023	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	configuration
ios-xr-2	127.0.0.1	10024	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	configuration
ios-xr-3	127.0.0.1	10025	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	configuration

We got a red alert because this change was done outside of NSO. And we can see that the hash of the configuration present on NSO is not the same that we got from the device.

name	address	port	type	services	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration
ios-xr-0	127.0.0.1	10022	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	configuration

Action: check-sync Performed: 2023-01-24 18:26:52 Status: completed

Result: device was not in sync

out-of-sync
got: b78d090ac5cb8e4d09cec8d8184235b expected: 0e15655ce829290701ae616e2f72eee7

name	address	port	type	services	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration
ios-xr-1	127.0.0.1	10023	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	configuration
ios-xr-2	127.0.0.1	10024	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	configuration
ios-xr-3	127.0.0.1	10025	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	configuration

If we click on “compare-config” button we can see exactly what was done outside of NSO

name	address	port	type	services	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration
ios-xr-0	127.0.0.1	10022	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	configuration
ios-xr-1	127.0.0.1	10023	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	configuration
ios-xr-2	127.0.0.1	10024	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	configuration
ios-xr-3	127.0.0.1	10025	cisco-iosxr-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	configuration

Device manager
NSO VERSION 6.9 admin

name	address	port	type	services	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration
ios-xr-0	127.0.0.1	10022	cisco-iosxr-cli-7.43...cisco-iosxr-cli-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration
ios-xr-1	127.0.0.1	10023	cisco-iosxr-cli-7.43...cisco-iosxr-cli-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration
ios-xr-2	127.0.0.1	10024	cisco-iosxr-cli-7.43...cisco-iosxr-cli-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration
ios-xr-3	127.0.0.1	10025	cisco-iosxr-cli-7.43...cisco-iosxr-cli-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration

Action: compare-config Performed: 2023-01-24 18:27:33 Status: completed

Result: CDB and device config does not match

```

1   devices {
2     device ios-xr-0 {
3       config {
4         vtp {
5           mode off;
6         }
7       }
8     }
9   }
10 }
11

```

And now we've 2 options

Option 1 : We "SYNC-FROM" the device

Option 2 : We "SYNC-TO" device

If we go with Option 1 we will grab the configuration that was done in the device without using NSO and push to NSO. If we use the Option 2 we will push to the device the configuration that NSO had previously about the device (it will act like a rollback)

In this case we want to delete the configuration that was done outside of NSO so we will click on the "sync-to" button

Device manager
NSO VERSION 6.9 admin

name	address	port	type	services	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration
ios-xr-0	127.0.0.1	10022	cisco-iosxr-cli-7.43...cisco-iosxr-cli-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration
ios-xr-1	127.0.0.1	10023	cisco-iosxr-cli-7.43...cisco-iosxr-cli-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration
ios-xr-2	127.0.0.1	10024	cisco-iosxr-cli-7.43...cisco-iosxr-cli-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration
ios-xr-3	127.0.0.1	10025	cisco-iosxr-cli-7.43...cisco-iosxr-cli-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration

Device manager
NSO VERSION 6.9 admin

name	address	port	type	services	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration
ios-xr-0	127.0.0.1	10022	cisco-iosxr-cli-7.43...cisco-iosxr-cli-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration
ios-xr-1	127.0.0.1	10023	cisco-iosxr-cli-7.43...cisco-iosxr-cli-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration
ios-xr-2	127.0.0.1	10024	cisco-iosxr-cli-7.43...cisco-iosxr-cli-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration
ios-xr-3	127.0.0.1	10025	cisco-iosxr-cli-7.43...cisco-iosxr-cli-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration

Action: sync-to Performed: 2023-01-24 18:28:25 Status: completed

Result: config was synced from device true

We can now click on Check-Sync and confirm that both device and NSO device configuration are the same.

name	address	port	type	services	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration
ios-xr-0	127.0.0.1	10022	cisco-iosxr-cli-7.43...cisco-iosxr-cli-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration
ios-xr-1	127.0.0.1	10023	cisco-iosxr-cli-7.43...cisco-iosxr-cli-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration
ios-xr-2	127.0.0.1	10024	cisco-iosxr-cli-7.43...cisco-iosxr-cli-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration
ios-xr-3	127.0.0.1	10025	cisco-iosxr-cli-7.43...cisco-iosxr-cli-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration

If we login to the device, we will confirm that the VTP configuration is not there anymore.

```
ios-xr-0(config) # end
ios-xr-0# show run
admin
  exit-admin-config
!
interface Loopback 0
  no shutdown
  ipv4 address 1.1.1.1 255.255.255.255
exit
```

1.6 - Create Device Groups and Device Templates

Ok, so what if we want to apply configurations to several devices at the same time? To achieve this we need to create a device Group.

Go into configuration Editor and select the “ncs:devices” module.

aaa:aaa	nacm:nacm	ncs:high-availability	ncs:services	ncs:zombies	tts:tts
aaa:alias	ncm:netconf-state	ncs:java-vm	ncs:side-effect-queue	rcmon:restconf-state	webui:webui
aaa:session	ncs:cluster	ncs:metric	ncs:smart-license	scheduler:scheduler	
aaa:user	ncs:compliance	ncs:packages	ncs:snmp-notification-receiver	snmp:snmp	
al:alarms	ncs:customers	ncs:python-vm	ncs:software	tftp:policy	
last:last-logins	ncs:devices	ncs:service-progress-monitoring	ncs:ssh	tftm:ncs-state	

In the Device-Group click on the Plus button

template	This list is empty	Add list item → +
device-group	This list is empty	Add list item → +
mib-group	Filter by keys	
<input type="checkbox"/> name		
<input type="checkbox"/> snmp		

Create the group “ios-xr-devices”

The screenshot shows the Cisco Configuration editor interface. In the center, a modal dialog box is open with the title "Add new list item". Inside the dialog, there is a single input field labeled "name" which contains the text "ios-xr-devices". At the bottom of the dialog are two buttons: "cancel" on the left and "confirm" on the right. The background of the editor shows a list of device-groups under the "device-group" section. One device-group entry, "ios-xr-devices", is highlighted with a blue selection bar.

Click on “confirm”

The screenshot shows the Cisco Configuration editor after the "confirm" button was clicked. The "ios-xr-devices" device-group has been successfully added to the list. It is now listed under the "device-group" section along with other entries like "name". The "ios-xr-devices" entry is highlighted with a blue selection bar.

Now let's click on the device-group that you've created.

To Add devices to the device Group we click on the + button next to device-name tile.

The screenshot shows the Cisco Configuration editor on the "ios-xr-devices" device-group details page. The top navigation bar shows the URL as "/ncs/devices/device-group/ios-xr-devices". The main content area displays the "name" field containing "ios-xr-devices". Below it are sections for "device-name", "device-group", and "location". The "device-name" section contains a "+ button" which is highlighted with a blue selection bar. A small modal dialog box is also visible over the "device-name" section, titled "Add new list item", with a dropdown menu showing options like "ios-xr-0", "ios-xr-1", "ios-xr-2", and "ios-xr-3".

It will show up all the devices we've available

The screenshot shows the Cisco Configuration editor on the "ios-xr-devices" device-group details page. The "device-name" section now contains the text "ios-xr-0" after the "+ button" was clicked. A modal dialog box is open, titled "Add new list item", with a dropdown menu showing "ios-xr-0", "ios-xr-1", "ios-xr-2", and "ios-xr-3". The "Source" option is also visible at the bottom of the dropdown.

If we select all the devices this should be the result.

The screenshot shows the 'Configuration editor' interface with the URL `/ncs/devices/device-group/ios-xr-devices`. The top navigation bar includes tabs for 'Edit config', 'Config', 'Operdata', and 'Actions'. On the right, there are buttons for 'None', 'Containers', and 'admin'. The main area displays a 'device-group' entry with the name 'ios-xr-devices'. Below it is a list of five devices: 'ios-xr-0', 'ios-xr-1', 'ios-xr-2', 'ios-xr-3', and 'ios-xr-4'. A tooltip message 'This list is empty. Add list item → +' is displayed next to the list. At the bottom, there are icons for edit, delete, and refresh.

After creating a Device group, let's create a Device Template so we can create a config that we will spread across all the devices in the group with just few clicks.

Go to the “Configuration Editor” and choose the “ncs:devices” module

In the template tile, click on the + button.

The screenshot shows the 'Configuration editor' interface with the URL `/ncs/devices`. The top navigation bar includes tabs for 'Edit config', 'Config', 'Operdata', and 'Actions'. On the right, there are buttons for 'None', 'Containers', and 'admin'. The main area displays a 'template' entry. Below it is a list titled 'device-group' which is currently empty. A tooltip message 'Add list item → +' is displayed next to the list. At the bottom, there are icons for edit, delete, and refresh.

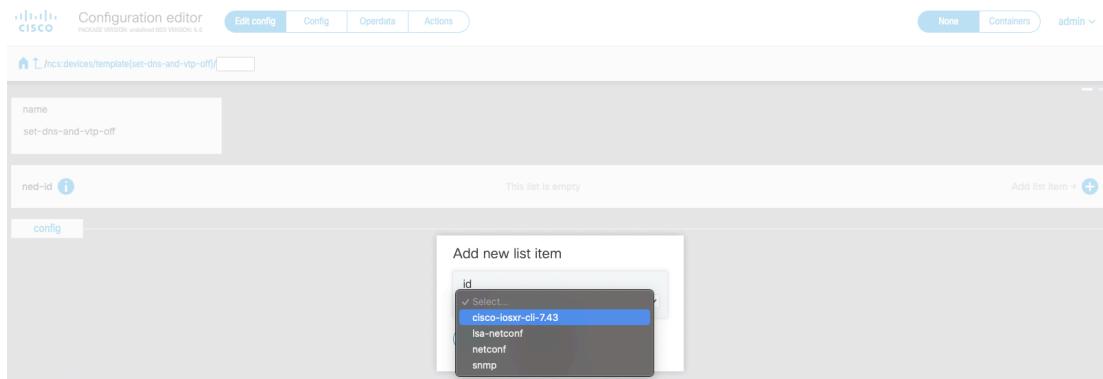
Create the template “set-dns-and-vtp-off”.

The screenshot shows the 'Configuration editor' interface with the URL `/ncs/devices`. The top navigation bar includes tabs for 'Edit config', 'Config', 'Operdata', and 'Actions'. On the right, there are buttons for 'None', 'Containers', and 'admin'. The main area displays a 'template' entry. Below it is a list titled 'device-group' which is currently empty. A modal dialog box is open, prompting to 'Add new list item'. The 'name' field contains 'set-dns-and-vtp-off'. There are 'cancel' and 'confirm' buttons at the bottom of the dialog. The background shows other sections like 'mib-group'.

Click on the “confirm” button

The screenshot shows the 'Configuration editor' interface with the URL `/ncs/devices`. The top navigation bar includes tabs for 'Edit config', 'Config', 'Operdata', and 'Actions'. On the right, there are buttons for 'None', 'Containers', and 'admin'. The main area displays a 'template' entry. Below it is a list titled 'device-group' which now contains the entry 'set-dns-and-vtp-off'. The background shows other sections like 'mib-group'.

Enter in the template and select the NED that we're using (ios-xr) and click Confirm.



Now click on the config (inside the NED)

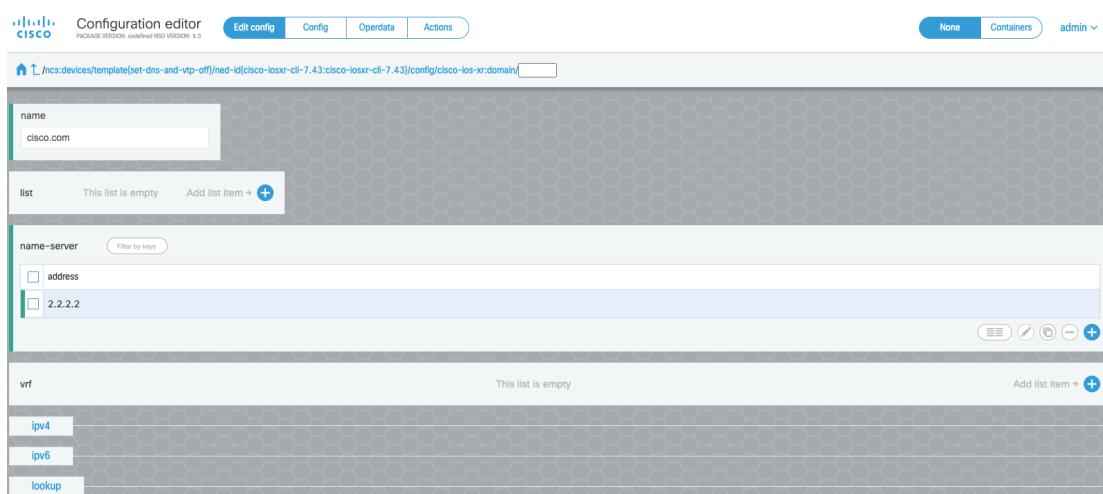


And now you just need to setup the configurations that you want to apply. Let's create a DNS configuration and set the VTP to off.

Start with VTP, choose the mode "off".



Now, go back to the config and look for Domain. Add the Domain name and name-server.



Now, we go to the commit manager and let's see the differences.

From the config diff we can see the device-group and the configuration template that we've created.

```

1      devices {
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
}

```

```

device [
  template set-dns-and-vtp-off {
    ned-id cisco-iosxr-cl7-7.43 (
      config [
        domain [
          name cisco.com;
          name-server 2.2.2.2;
        ];
        vtp [
          mode off;
        ];
      ]
    );
  }
]
device-group ios-xr-devices [
  device-name [ ios-xr-0 ios-xr-1 ios-xr-3 ];
]

```

After commit, we can now go to the Device Group and Apply the Template.

```

1      devices {
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
}

```

```

device [
  template set-dns-and-vtp-off {
    ned-id cisco-iosxr-cl7-7.43 (
      config [
        domain [
          name cisco.com;
          name-server 2.2.2.2;
        ];
        vtp [
          mode off;
        ];
      ]
    );
  }
]
device-group ios-xr-devices [
  device-name [ ios-xr-0 ios-xr-1 ios-xr-3 ];
]

```

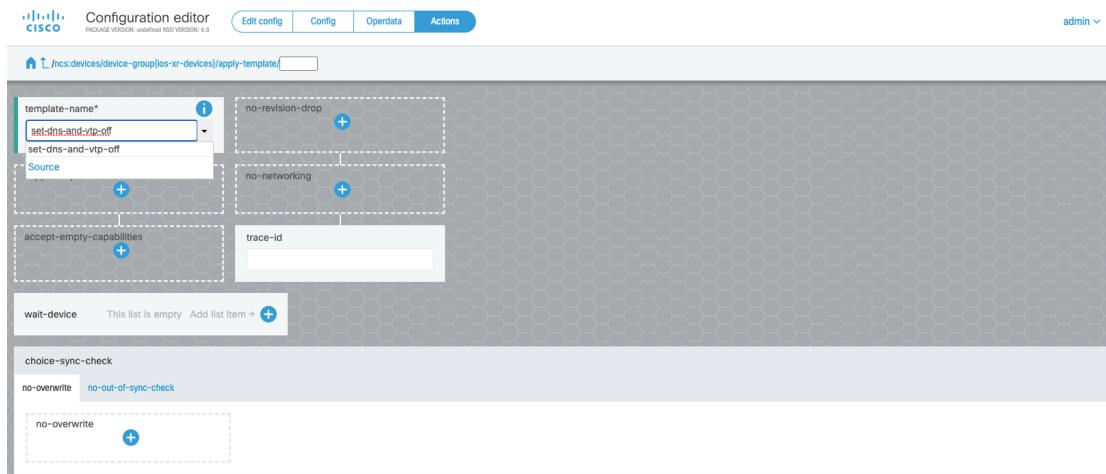
To manage the Device Group we go to the Configuration Editor, “ncs:devices” module and click on the recently created device-group “ios-xr-devices”.

name	ios-xr-devices	Actions
		apply-template

We click on “Actions” and then on the blue button “Apply-Template”

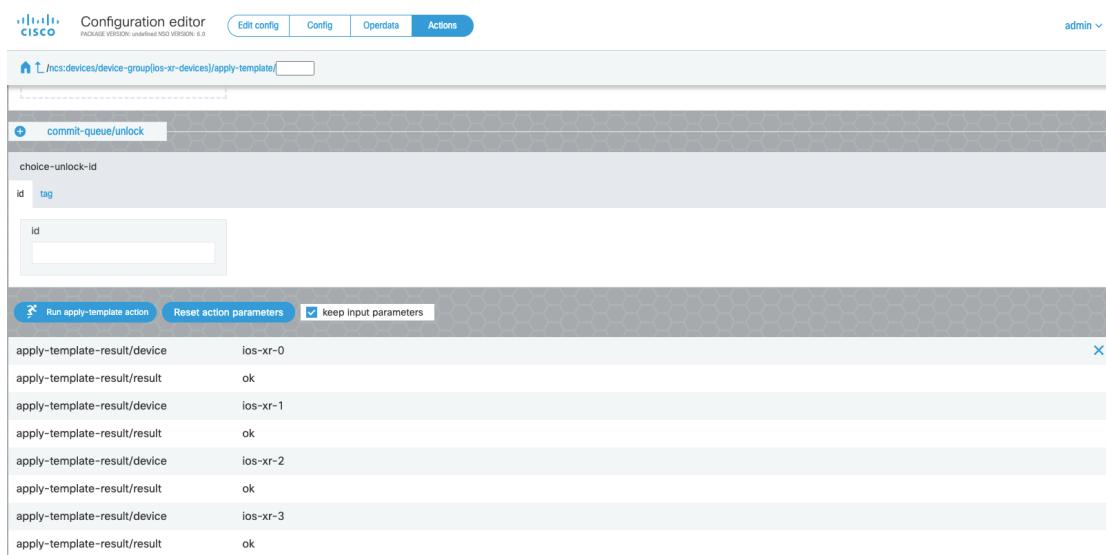
connect	check-sync	apply-template
		apply-template

And now, select the “template-name” that we’ve created.



Scroll down and click on “Run apply-template action”.

We can see the apply-template result “ok” for each device present in the device group.



If we go back to the “Commit Manager” we can see that with the application of the template, we will push the same configuration to all the devices at the same time.

We click on the “commit” button and if we go inside the Device CLI we can see that the configurations were applied.

```
ios-xr-0# show run
admin
  exit-admin-config
!
domain name cisco.com
domain name-server 2.2.2.2
vtp mode off
interface Loopback 0
  no shutdown
  ipv4 address 1.1.1.1 255.255.255.255
exit
```

1.7 – Create a simple NSO Service

Templates are very flexible and easy to create. But they lack on configuration consistency after being applied. If someone goes thought NSO and change one configuration that you've done via template, you don't have an easy way to see it and rollback. For that purposes and much more, we have the NSO Services.

An NSO service will allow you to see if the configuration that you applied is still present on the devices. And if someone changed, we could compare-configs and re-deploy if necessary.

The first step is to create a service.

NSO already brings a script that creates a service skeleton.

Use the command “ncs-make-package -h”

```
user@Pc nso-instance % ncs-make-package -h
Usage: ncs-make-package [options] package-name

ncs-make-package --netconf-ned DIR package-name
ncs-make-package --lsa-netconf-ned DIR package-name
ncs-make-package --generic-ned-skeleton package-name
ncs-make-package --snmp-ned DIR package-name
ncs-make-package --service-skeleton TYPE package-name
ncs-make-package --data-provider-skeleton package-name
ncs-make-package --erlang-skeleton package-name

where TYPE is one of:
    java                  Java based service
    java-and-template     Java service with template
    python                Python based service
    python-and-template   Python service with template
    template              Template service (no code)

ADDITIONAL OPTIONS
--dest DIR
--build
--verbose
--no-test
--no-fail-on-warnings
-h | --help

SERVICE specific options:
--augment PATH
--root-container NAME

JAVA specific options:
--java-package NAME

NED specific options:
--no-java
--no-netsim
--no-python
--no-template
--vendor STRING
```

```

--package-version STRING

NETCONF NED specific options:
  --ncs-depend-package DIR
  --pyang-sanitize
  --confd-netsim-db-mode candidate | startup | running-only

LSA NETCONF NED specific options:
  --lsa-lower-nso PKG | DIR
    where PKG can be one of: cisco-nso-nc-4.7 cisco-nso-nc-5.6
cisco-nso-nc-5.7 cisco-nso-nc-5.8 cisco-nso-nc-6.0

PYTHON specific options:
  --component-class NAME (default main.Main)
  --action-example
  --subscriber-example

ERLANG specific options:
  --erlang-application-name NAME (uses package name as default)

See manpage for ncs-make-package(1) for more info.

```

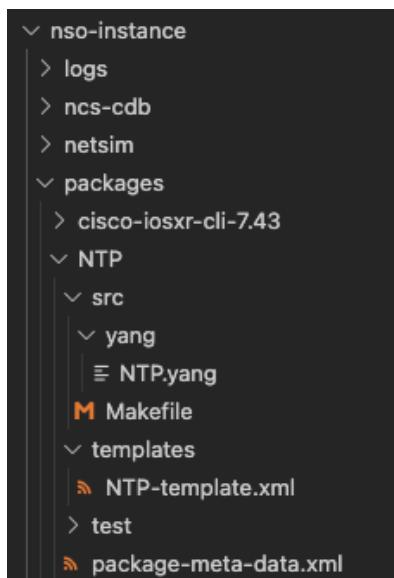
You can see that we've many options here.

Python and Java options are when we want to apply templates/services, based on logic. Imagine you want to apply QOS data to a device interface but only based on the interface high utilization. You can check the interface utilization via external APIs and only send the device configs according to what you receive.

Let's choose the Template option. (create the service inside the packages folder, otherwise you will have to copy the service folder to the packages folder after you're done)

```
user@Pc packages % ncs-make-package --service-skeleton template NTP
```

After you run the script, you will find this structure.



`package-meta-data.xml` is where you will find the service version and other useful information.

```

» package-meta-data.xml ×

nso-instance > packages > NTP > » package-meta-data.xml > ncs-package
1  <ncs-package xmlns="http://tail-f.com/ns/ncs-packages">
2    <name>NTP</name>
3    <package-version>1.0</package-version>
4    <description>Template-based NTP resource facing service</description>
5    <ncs-min-version>6.0</ncs-min-version>
6
7    <!-- It's possible to add more components to the -->
8    <!-- same package, multiple services, data providers etc -->
9
10   </ncs-package>
11

```

Then you have the XML file and the YANG file.

This is what we will find in the XML file.

```

» NTP-template.xml ×

nso-instance > packages > NTP > templates > » NTP-template.xml > config-template
1  <config-template xmlns="http://tail-f.com/ns/config/1.0"
2    ...
3      | servicepoint="NTP">
4      <devices xmlns="http://tail-f.com/ns/ncs">
5          <device>
6              <!--
7                  | Select the devices from some data structure in the service
8                  | model. In this skeleton the devices are specified in a leaf-list.
9                  | Select all devices in that leaf-list:
10                 -->
11                 <name>{/device}</name>
12                 <config>
13                     <!--
14                         | Add device-specific parameters here.
15                         | In this skeleton the service has a leaf "dummy"; use that
16                         | to set something on the device e.g.:
17                         <ip-address-on-device>{/dummy}</ip-address-on-device>
18                     -->
19                     </config>
20                 </device>
21             </devices>
22         </config-template>

```

And this is what we will find on the YANG file:

```

≡ NTP.yang ×

nso-instance > packages > NTP > src > yang > ≡ NTP.yang
1 module NTP {
2   namespace "http://com/example/NTP";
3   prefix NTP;
4
5   import ietf-inet-types {
6     prefix inet;
7   }
8   import tailf-ncs {
9     prefix ncs;
10  }
11
12  list NTP {
13    key name;
14
15    uses ncs:service-data;
16    ncs:servicepoint "NTP";
17
18    leaf name {
19      type string;
20    }
21
22    // may replace this with other ways of referring to the devices.
23    leaf-list device {
24      type leafref {
25        path "/ncs:devices/ncs:device/ncs:name";
26      }
27    }
28
29    // replace with your own stuff here
30    leaf dummy {
31      type inet:ipv4-address;
32    }
33  }
34}
35

```

Our next step will be:

Go to NSO and make the configurations that we want to automate via the service.

After all the configs are done instead of doing the commit, we will ask for the output of the configurations to be shown in XML.

In this case we want to automate the NTP server and peer configuration.

```

user@Pc packages % ncs_cli -Cu admin
admin@ncs# config
Entering configuration mode terminal
admin@ncs(config)# devices device ios-xr-1
admin@ncs(config-device-ios-xr-1)# config
admin@ncs(config-config)# ntp

```

```

admin@ncs(config-ntp)# server 172.16.22.44 minpoll 8 maxpoll 12
admin@ncs(config-ntp)# peer 192.168.22.33
admin@ncs(config-ntp)# commit dry-run outformat xml
result-xml {
    local-node {
        data <devices xmlns="http://tail-f.com/ns/ncs">
            <device>
                <name>ios-xr-1</name>
                <config>
                    <ntp xmlns="http://tail-f.com/ned/cisco-ios-xr">
                        <peer>
                            <address>192.168.22.33</address>
                        </peer>
                        <server>
                            <server-list>
                                <name>172.16.22.44</name>
                                <minpoll>8</minpoll>
                                <maxpoll>12</maxpoll>
                            </server-list>
                        </server>
                    </ntp>
                </config>
            </device>
        </devices>
    }
}

```

Now, we grab the XML that we got from the NSO cli config and we paste into the skeleton XML file that was generated.

We only grab what is inside the <config></config> flags.

```

<ntp xmlns="http://tail-f.com/ned/cisco-ios-xr">
    <peer>
        <address>192.168.22.33</address>
    </peer>
    <server>
        <server-list>
            <name>172.16.22.44</name>
            <minpoll>8</minpoll>
            <maxpoll>12</maxpoll>
        </server-list>
    </server>
</ntp>

```

This should be the result of the final XML file.

```

NTP-template.xml ×
nso-instance > packages > NTP > templates > NTP-template.xml > ...
1   <config-template xmlns="http://tail-f.com/ns/config/1.0"
2     servicepoint="NTP">
3   <devices xmlns="http://tail-f.com/ns/ncs">
4     <device>
5       <!--
6         Select the devices from some data structure in the service
7         model. In this skeleton the devices are specified in a leaf-list.
8         Select all devices in that leaf-list:
9       -->
10      <name>{/device}</name>
11      <config>
12        <!--
13          Add device-specific parameters here.
14          In this skeleton the service has a leaf "dummy"; use that
15          to set something on the device e.g.:
16          <ip-address-on-device>{/dummy}</ip-address-on-device>
17        -->
18        <ntp xmlns="http://tail-f.com/ned/cisco-ios-xr">
19          <peer>
20            <address>192.168.22.33</address>
21          </peer>
22          <server>
23            <server-list>
24              <name>172.16.22.44</name>
25              <minpoll>8</minpoll>
26              <maxpoll>12</maxpoll>
27            </server-list>
28            </server>
29          </ntp>
30        </config>
31      </device>
32    </devices>
33  </config-template>

```

If we want to apply this static config. We just need to compile the service and we're good to go.

But let's see how we define input variables because if server or peer addresses change we just need to go to our service and change the values to apply the change to all the devices.

Let's create 4 inputs variables:

Peer IP-address

Server IP-address

Minpool

Maxpool

After defining the variables our XML should look like this

```

❷ NTP-template.xml ×
nso-instance > packages > NTP > templates > ❸ NTP-template.xml > config-template > devices > device > config > ntp > server
1   <config-template xmlns="http://tail-f.com/ns/config/1.0"
2     servicepoint="NTP">
3     <devices xmlns="http://tail-f.com/ns/ncs">
4       <device>
5         <!--
6           Select the devices from some data structure in the service
7           model. In this skeleton the devices are specified in a leaf-list.
8           Select all devices in that leaf-list:
9         -->
10        <name>{/device}</name>
11        <config>
12          <!--
13            Add device-specific parameters here.
14            In this skeleton the service has a leaf "dummy"; use that
15            to set something on the device e.g.:
16            <ip-address-on-device>{/dummy}</ip-address-on-device>
17          -->
18        <ntp xmlns="http://tail-f.com/ned/cisco-ios-xr">
19          <peer>
20            <address>{/peer-address}</address>
21          </peer>
22          <server>
23            <server-list>
24              <name>{/server-address}</name>
25              <minpoll>{/minpoll}</minpoll>
26              <maxpoll>{/maxpoll}</maxpoll>
27            </server-list>
28          </server>
29        </ntp>
30      </config>
31    </device>
32  </devices>
33 </config-template>

```

Now we need to go to the YANG file and setup the definition for each variable.

So, we will setup as mandatory inputs the peer-address and server-address (accepting only IP address strings as input) and then the minpoll and maxpoll (they must be the type uint8) will be non-mandatory and with default values.

```

NTP-template.xml   NTP.yang
nso-instance > packages > NTP > src > yang > NTP.yang

1  module NTP {
2    namespace "http://com/example/NTP";
3    prefix NTP;
4    import ietf-inet-types {
5      prefix inet;
6    }
7    import tailf-ncs {
8      prefix ncs;
9    }
10   list NTP {
11     key name;
12     uses ncs:service-data;
13     ncs:servicepoint "NTP";
14     leaf name {
15       type string;
16     }
17     leaf-list device {
18       type leafref {
19         path "/ncs:devices/ncs:device/ncs:name";
20       }
21     }
22
23     // replace with your own stuff here
24     leaf peer-address {
25       type inet:ipv4-address;
26       mandatory true;
27     }
28     leaf server-address {
29       type inet:ipv4-address;
30       mandatory true;
31     }
32     leaf minpool {
33       type uint8;
34       default "8";
35     }
36     leaf maxpool {
37       type uint8;
38       default "12";
39     }
40   }
41 }

```

After all of this is done, we go to the packages folder. Change Directory to NTP/src

And apply the “make” command.

```

user@Pc packages % cd NTP
user@Pc NTP % cd src
user@Pc src % make
mkdir -p ../load-dir
/Users/user/NSO-DD/bin/ncsc `ls NTP-ann.yang > /dev/null 2>&1 &&
echo "-a NTP-ann.yang` \
      --fail-on-warnings \
      \
      -c -o ../load-dir/NTP.fxs yang/NTP.yang

```

Then, we should go to NSO, and do a packages reload.

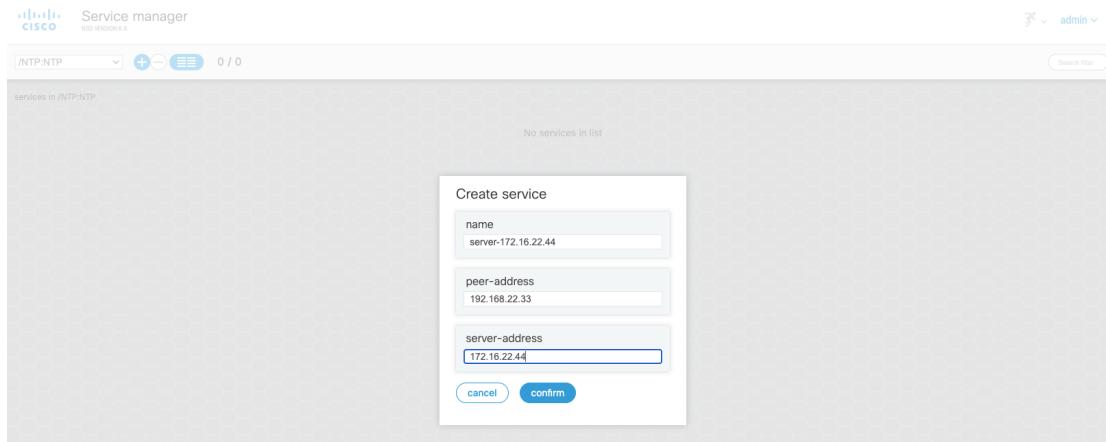
After the reload you will see the NTP service package that we've just created.

Going back to the Homepage we click on the “Service Manager”

We select the NTP service that we've created

Then, we click on + button

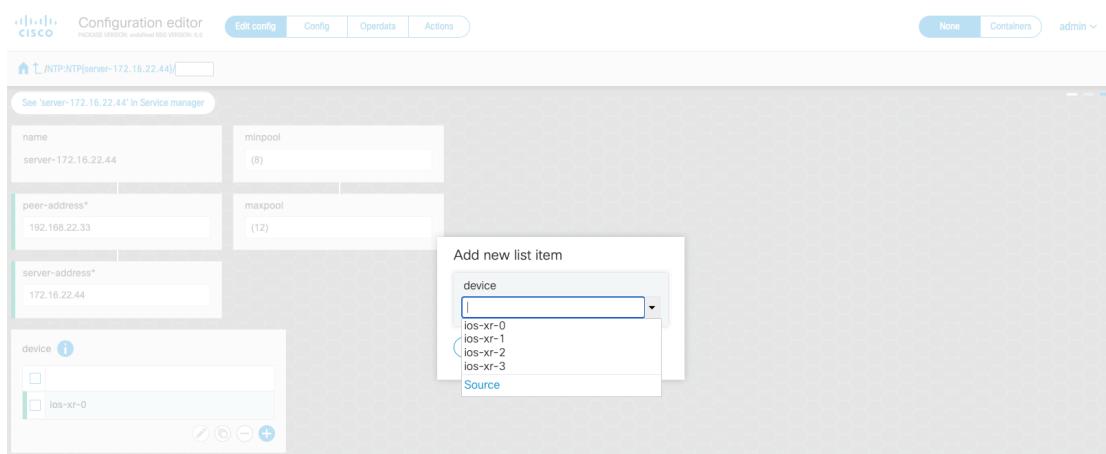
We should give a service name (help us identify the configs applied) and setup the mandatory arguments, in our case, the peer-address and server-address



Service Created! Let's add the devices that we want to affect with the service.



Add the devices you want.



We can keep the minpoll and maxpoll with the default values. And now, if we go to the commit manager we will see the configs applied to all the devices that we've selected.

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64

```

The configuration code shows multiple device definitions (ios-xr-0 to ios-xr-3) with their respective NTP configurations.

If we click on native config, we can even see the native commands that NSO will send to the devices.

```

ios-xr-0
ntp
peer 192.168.22.33
server 172.16.22.44 minpoll 8 maxpoll 12
exit

ios-xr-1
ntp
peer 192.168.22.33
server 172.16.22.44 minpoll 8 maxpoll 12
exit

ios-xr-2
ntp
peer 192.168.22.33
server 172.16.22.44 minpoll 8 maxpoll 12
exit

ios-xr-3
ntp
peer 192.168.22.33
server 172.16.22.44 minpoll 8 maxpoll 12
exit

```

We click on “commit” button and confirm.

The configuration code for the devices is shown again, followed by a confirmation dialog box asking if the user wants to commit changes to NSO.

Now we can see if the service is in Sync (meaning that the configs that we applied are still there)

Now, to see the how powerful the services are let's go inside 2 devices, ios-xr-2 and ios-xr-3 and delete the NTP config.

Go to the Device Manager, click on IOS-XR-2 and delete the peer and server configs by selecting them and clicking on the “-“ button.

Go to the commit manager and press “Commit”

```

1 device ios-xr-2 {
2   config {
3     ntp {
4       peer 192.168.22.33 {
5         server {
6           server-list 172.16.22.44 {
7             maxpoll 12;
8           }
9         }
10      }
11    }
12  }
13 }
14 }
15 }
16 }
17 }
18 }
19 }
20 }
21 }
22 }
23 }
24 }
25 }
26 }
27 }
28 }
29 }
30 }
31 }

```

Now, if we go back to the “Service Manager” and we click on “Check-Sync” we will get a RED signal.

name	devices	check-sync	re-deploy	re-deploy dry-run
server-172.16.22.44	4	check-sync	re-deploy	re-deploy dry-run

name	devices	check-sync	re-deploy	re-deploy dry-run
server-172.16.22.44	4	check-sync	re-deploy	re-deploy dry-run

Action: check-sync Performed: 2023-01-25 00:41:28 Status: completed
Result: **service was not in sync**
false

By clicking on Re-deploy Dry-Run

name	devices	check-sync	re-deploy	re-deploy dry-run
server-172.16.22.44	4	check-sync	re-deploy	re-deploy dry-run

Action: re-deploy dry-run Performed: 2023-01-25 00:44:17 Status: completed

Result: service has changes

```

1 devices {
2   device ios-xr-2 {
3     config {
4       ntp {
5         peer 192.168.22.33 {
6           server {
7             server-list 172.16.22.44 {
8               minpoll 8;
9               maxpoll 12;
10            }
11          }
12        }
13      }
14    }
15  }
16  device ios-xr-3 {
17    config {
18      ntp {
19        peer 192.168.22.33 {
20          server {
21            server-list 172.16.22.44 {
22              minpoll 8;
23              maxpoll 12;
24            }
25          }
26        }
27      }
28    }
29  }
30 }
```

We will see exactly what needs to be added to the devices for them to be compliant with the service. And to configure the devices again we just need to click on re-deploy.

Action: re-deploy Performed: 2023-01-25 00:45:11 Status: completed

Result: service was re-deployed

Action: re-deploy Performed: 2023-01-25 00:45:11 Status: completed

Result: service was re-deployed

And the service was re-deployed. If we connect to ios-xr-2 and make a "show run" we can see that the configs are there again.

```

ios-xr-2# show run
admin
exit-admin-config
!
domain name cisco.com
domain name-server 2.2.2.2
ntp
  peer 192.168.22.33
  server 172.16.22.44 minpoll 8 maxpoll 12

```

```
exit  
vtp mode off
```

1.8 – NSO APIs

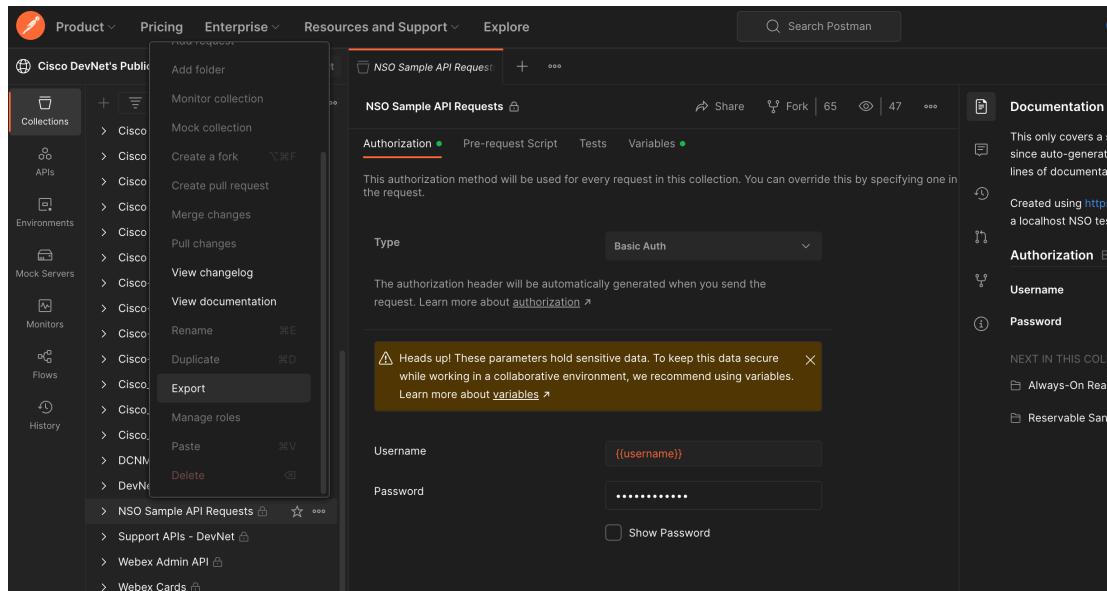
IN NSO you can interact via REST APIs as well.

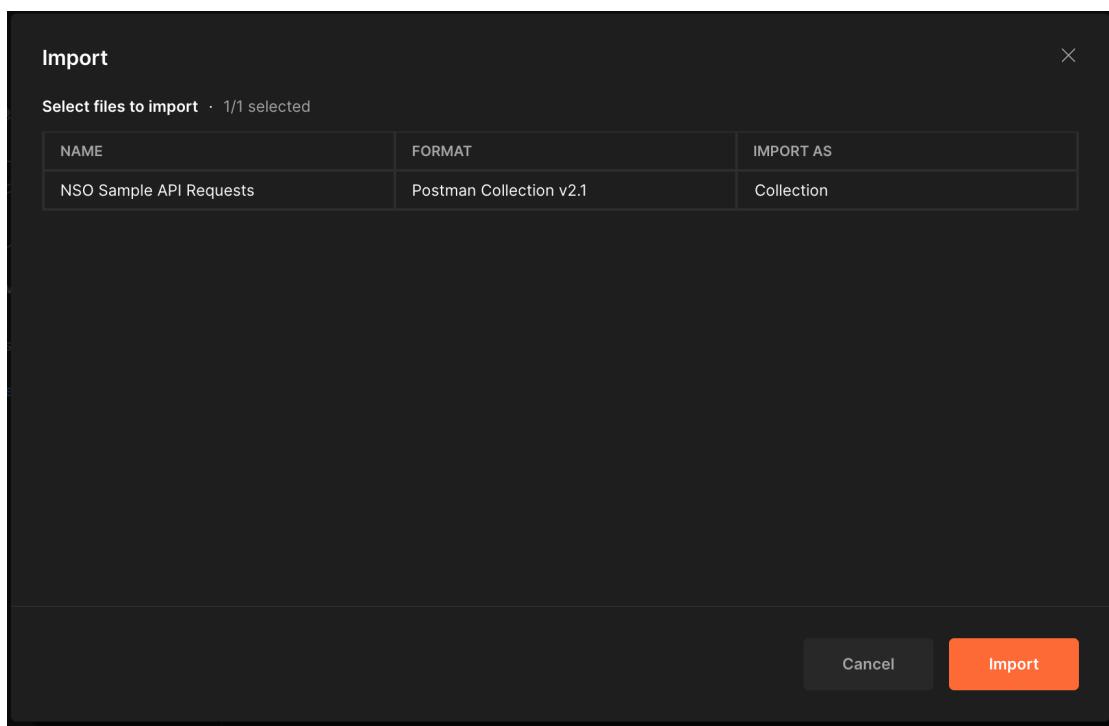
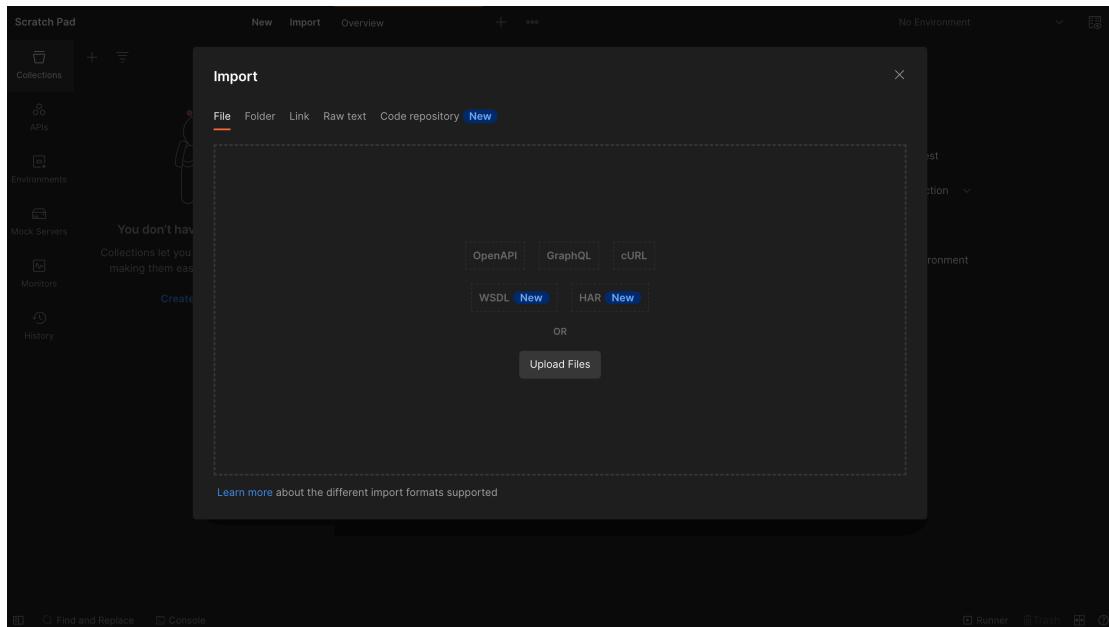
Cisco Provides a Postman collection already with a lot of options for you to interact with NSO.

You can download it here.

Download and Import the collection (NSO Sample API Requests).

<https://developer.cisco.com/docs/ns0/#!ns0-postman-collections>





Change the Collection variables to the ones that fit your environment.

NSO Sample API Requests

Authorization • Pre-request Script Tests Variables •

These variables are specific to this collection and its requests. Learn more about [collection variables](#).

VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...	Persist All	Reset All
NSO_IP	127.0.0.1	127.0.0.1	...		
NSO_HTTP_PORT	8080	8080	...		
username	admin	admin	...		
password	admin	admin	...		
sample_ios_device	ios0	ios0	...		
sample_xr_device	ios-xr-0	ios-xr-0	...		
sample_junos_device	junos0	junos0	...		

If you're using the default installation, change the NSO_HTTP_PORT from 8009 to 8080

And, if you gave the same name to the devices, you can change the sample_xr_device to "ios-xr-0"

Define the variable PROTOCOL with values http

<input checked="" type="checkbox"/>	sample_xr_device	ios-xr-0	ios-xr-0
<input checked="" type="checkbox"/>	sample_junos_device	junos0	junos0
<input checked="" type="checkbox"/>	PROTOCOL	http	http
Add a new variable			

Here is an example of the Device Group request.

The screenshot shows the Postman application interface. The left sidebar contains collections like 'NSO Sample API Requests' and 'NSO Device Groups'. The main area shows a successful 'GET' request to 'List Device Groups' with a status of 200 OK. The response body is a JSON object representing a device group with members and a ned-id.

```
1 "tailf-ncs:device-group": [
2   {
3     "name": "ios-xr-devices",
4     "device-name": [
5       "ios-xr-0",
6       "ios-xr-1",
7       "ios-xr-2",
8       "ios-xr-3"
9     ],
10    "member": [
11      "ios-xr-0",
12      "ios-xr-1",
13      "ios-xr-2",
14      "ios-xr-3"
15    ],
16    "ned-id": [
17      {
18        "id": "cisco-iosxr-cl-7.43:cisco-iosxr-cl-7.43"
19      }
20    ]
21  }
22]
```

How to get the RESTCONF paths ? There are many ways to get them, one of them is going thought NSO CLI and use the command “show running-config devices device ios-xr-0 | display restconf”

And it will return the paths available for each device.

```
admin@ncs# show running-config devices device ios-xr-0 | display restconf  
/restconf/data/tai1f-ncs:devices/device=ios-xr-0/address 127.0.0.1
```

```

/restconf/data/tailf-ncs:devices/device=ios-xr-0/port 10022
/restconf/data/tailf-ncs:devices/device=ios-xr-0/ssh/host-key=ssh-
rsa/key-data
"AAAAB3NzaC1yc2EAAAQABAAABgQDAW+oFswFE439ByAMZ++iMLcnhVsI+Lui5mn/
Wfiie\ncpC3czutSFlfdIBOIC4sCpqYrxoqowXHhU2T9wnx5sAWwzPJJuMjbbgUVnUs1CV
8MeF/jWOUL4\n2Rr6IbECuV7qvMHfZmcrTR/BqXUdbqBtb5f0TzHdvT2gBP7AgLPptzV
a5/BDkNV70UG6Ocx\ncsqWN15471U0Dol7rSyufKz8y1sa4E0Dayn+9+O8uK6iq7GzP
qyf9fkrQjhmlNc5sozuAPQ4\nFMmjIAAMbQOzz5LvjktUw2QFgFm53Dzp83LxG6nVWBN
S4Jlgnile4rG06xPDh6IAy5MKiBS\ntN4yBYF++0WL2tKszrCo//jZsDbv2ieTFROTp
mP2ldwgt7YUyep8j6JWVqLRPDVIjgXeGI6C\ng8xa81VW5uElmwGDSSjVFZVyXJQVKog
DNw19vREJHtSRAqjGknN648zMtdR+BI1c9fLzdxS\nb1o2jJtU/P5WLLFmJDbw0+A+d
39iNuNr9H5PRj8="
/restconf/data/tailf-ncs:devices/device=ios-xr-0/ssh/host-key=ssh-
ed25519/key-data
AAAC3NzaC1lZDI1NTE5AAAAIGvE1TpobfiHd3r/Z+1Sxr8bARNNyzIGQt0DMgvpNjmj
/restconf/data/tailf-ncs:devices/device=ios-xr-0/authgroup default
/restconf/data/tailf-ncs:devices/device=ios-xr-0/device-
type/cli/ned-id cisco-iosxr-cli-7.43
/restconf/data/tailf-ncs:devices/device=ios-xr-0/ssh-
algorithms/public-key [ ssh-ed25519 ssh-rsa ]
/restconf/data/tailf-ncs:devices/device=ios-xr-0/state/admin-state
unlocked
/restconf/data/tailf-ncs:devices/device=ios-xr-0/config/tailf-ned-
cisco-ios-xr:domain/name cisco.com
/restconf/data/tailf-ncs:devices/device=ios-xr-0/config/tailf-ned-
cisco-ios-xr:domain/name-server=2.2.2.2
/restconf/data/tailf-ncs:devices/device=ios-xr-0/config/tailf-ned-
cisco-ios-xr:n/restconf/data/tailf-ncs:devices/device=ios-xr-
0/address 127.0.0.1
/restconf/data/tailf-ncs:devices/device=ios-xr-0/port 10022
/restconf/data/tailf-ncs:devices/device=ios-xr-0/ssh/host-key=ssh-
rsa/key-data
"AAAAB3NzaC1yc2EAAAQABAAABgQDAW+oFswFE439ByAMZ++iMLcnhVsI+Lui5mn/
Wfiie\ncpC3czutSFlfdIBOIC4sCpqYrxoqowXHhU2T9wnx5sAWwzPJJuMjbbgUVnUs1CV
8MeF/jWOUL4\n2Rr6IbECuV7qvMHfZmcrTR/BqXUdbqBtb5f0TzHdvT2gBP7AgLPptzV
a5/BDkNV70UG6Ocx\ncsqWN15471U0Dol7rSyufKz8y1sa4E0Dayn+9+O8uK6iq7GzP
qyf9fkrQjhmlNc5sozuAPQ4\nFMmjIAAMbQOzz5LvjktUw2QFgFm53Dzp83LxG6nVWBN
S4Jlgnile4rG06xPDh6IAy5MKiBS\ntN4yBYF++0WL2tKszrCo//jZsDbv2ieTFROTp
mP2ldwgt7YUyep8j6JWVqLRPDVIjgXeGI6C\ng8xa81VW5uElmwGDSSjVFZVyXJQVKog
DNw19vREJHtSRAqjGknN648zMtdR+BI1c9fLzdxS\nb1o2jJtU/P5WLLFmJDbw0+A+d
39iNuNr9H5PRj8="
/restconf/data/tailf-ncs:devices/device=ios-xr-0/ssh/host-key=ssh-
ed25519/key-data
AAAC3NzaC1lZDI1NTE5AAAAIGvE1TpobfiHd3r/Z+1Sxr8bARNNyzIGQt0DMgvpNjmj
/restconf/data/tailf-ncs:devices/device=ios-xr-0/authgroup default
/restconf/data/tailf-ncs:devices/device=ios-xr-0/device-
type/cli/ned-id cisco-iosxr-cli-7.43
/restconf/data/tailf-ncs:devices/device=ios-xr-0/ssh-
algorithms/public-key [ ssh-ed25519 ssh-rsa ]
/restconf/data/tailf-ncs:devices/device=ios-xr-0/state/admin-state
unlocked
/restconf/data/tailf-ncs:devices/device=ios-xr-0/config/tailf-ned-
cisco-ios-xr:domain/name cisco.com
/restconf/data/tailf-ncs:devices/device=ios-xr-0/config/tailf-ned-
cisco-ios-xr:domain/name-server=2.2.2.2
/restconf/data/tailf-ncs:devices/device=ios-xr-0/config/tailf-ned-
cisco-ios-xr:n/restconf/data/tailf-ncs:devices/device=ios-xr-
0/address 127.0.0.1

```

```

/restconf/data/tailf-ncs:devices/device=ios-xr-0/port 10022
/restconf/data/tailf-ncs:devices/device=ios-xr-0/ssh/host-key=ssh-
rsa/key-data
"AAAAB3NzaC1yc2EAAAQABAAQgQDAW+oFswFE439ByAMZ++iMLcnhVsI+Lui5mn/
Wfiie\ncpC3czutSF1fdIBOIC4sCpqYrxoqowXHhU2T9wnx5sAWwzPJUbjbbgUVnUs1CV
8MeF/jWOUL4\n2Rr6IbECuV7qvMHfZmcrTR/BqXUdbqBtb5f0TzHdvT2gBP7AgLPptzV
a5/BDkNV70UG60cxc\nIsqWN15471U0Do17rSyufKz8ylsa4E0Dayn+9+O8uK6iq7GzP
qyf9fkrQjhLNc5sozuAPQ4\nFMmjIAAMbQOzz5LvjktUw2QFgFm53Dzp83LxG6nVWBNS4Jlgn
iLEE4rG06xPDh6IAy5MKiBS\ntN4yBYF++0WL2tKszrCo//jZsDbv2ieTFROTp
mP2ldwgt7YUyep8j6JWVqLRPDVIjgXeGI6C\nG8xa81VW5uElmwGDSSjVFZVyXJQVKog
DNw19vREJHtSRAqjGknN648zMtdR+BI1c9fLzdxS\nb1o2jJtU/P5WLLFmJDbw0+A+d
39iNuNr9H5PRj8="
/restconf/data/tailf-ncs:devices/device=ios-xr-0/ssh/host-key=ssh-
ed25519/key-data
AAAC3NzaC1lZDI1NTE5AAAAIGVe1TpobfiHd3r/Z+1Sxr8bARNNyzIGQt0DMgvPnjmj
/restconf/data/tailf-ncs:devices/device=ios-xr-0/authgroup default
/restconf/data/tailf-ncs:devices/device=ios-xr-0/device-
type/cli/ned-id cisco-iosxr-cli-7.43
/restconf/data/tailf-ncs:devices/device=ios-xr-0/ssh-
algorithms/public-key [ ssh-ed25519 ssh-rsa ]
/restconf/data/tailf-ncs:devices/device=ios-xr-0/state/admin-state
unlocked
/restconf/data/tailf-ncs:devices/device=ios-xr-0/config/tailf-ned-
cisco-ios-xr:domain/name cisco.com
/restconf/data/tailf-ncs:devices/device=ios-xr-0/config/tailf-ned-
cisco-ios-xr:domain/name-server=2.2.2.2
/restconf/data/tailf-ncs:devices/device=ios-xr-0/config/tailf-ned-
cisco-ios-xr:ntp/peer=192.168.22.33
/restconf/data/tailf-ncs:devices/device=ios-xr-0/config/tailf-ned-
cisco-ios-xr:ntp/server/server-list=172.16.22.44/minpoll 8
/restconf/data/tailf-ncs:devices/device=ios-xr-0/config/tailf-ned-
cisco-ios-xr:ntp/server/server-list=172.16.22.44/maxpoll 12
/restconf/data/tailf-ncs:devices/device=ios-xr-0/config/tailf-ned-
cisco-ios-xr:vtp	mode off
/restconf/data/tailf-ncs:devices/device=ios-xr-0/config/tailf-ned-
cisco-ios-xr:interface/Loopback=0/ipv4/address/ip 1.1.1.1
/restconf/data/tailf-ncs:devices/device=ios-xr-0/config/tailf-ned-
cisco-ios-xr:interface/Loopback=0/ipv4/address/mask 255.255.255.255

```

The same output can be reached from the WEBUI

If we navigate to the Device Manager and we select the device “ios-xr-0” we will find on the top right a dropdown named “Widgets” and one of the options is “RESTCONF”

The screenshot shows the Cisco Configuration editor interface. At the top, there are tabs for 'Edit config', 'Config', 'Operdata', and 'Actions'. Below these, a navigation bar shows the path '/ncs/devices/device[ios-xr-0]/'. A 'Widgets' dropdown menu is open, with 'RESTCONF' selected. Other options in the dropdown include 'JSON', 'KeyPath', 'Magic', 'NETCONF', 'Set', 'XML', and 'XPath'. The main content area displays configuration parameters for the device 'ios-xr-0', including 'name', 'address', and 'port' fields.

Clicking on RESTCONF option this will be the output. Remember, to use the Widgets function we must be in the Config mode of the device.

The screenshot shows the Configuration editor with 'RESTCONF' selected in the Widgets dropdown. The main content area displays a large block of RESTCONF output, which is a JSON representation of the device configuration. The output includes various keys such as 'address', 'port', and 'ssh' along with their corresponding values.

Another option to check the paths, is a package that was built for NSO (not made for PROD env), named “rest-api-explorer”

<https://gitlab.com/ns0-developer/rest-api-explorer>

Using this package, you can explore all the available APIs for devices and services inside NSO.

The screenshot shows the 'rest-api-explorer' interface. On the left, there is a tree view of available APIs under '/restconf/data'. One node, '/restconf/data/devices', is expanded, showing its methods: GET, HEAD, OPTIONS, PATCH, and POST. Each method has a brief description and a corresponding colored button. The right side of the screen shows a search bar and some descriptive text about the managed devices and device communication settings.

1.9 – Create Role Based and Resource Based Access Control rules

In this section the idea is to show you how we can create Role Based and Resource Based Access Control rules in NSO.

The first step is to add a new user.

In the Configuration Editor we navigate to the “aaa:aaa” module.

Then we go to the authentication/users

name	uid	gid	password	ssh_keydir	homedir
admin	65534	65534	\$6\$JHFjLfaShdkSx8QQ\$.pxYVNPY6QOZKY.D8.30	/var/ncs/homes/admin/.ssh	/var/ncs/homes/admin
oper	65534	65534	\$6\$23vdVhL6zXVJkbU\$.avJlOPHsff4nfIZSH7w/	/var/ncs/homes/oper/.ssh	/var/ncs/homes/oper
private	65534	65534	\$6\$	/var/ncs/homes/private/.ssh	/var/ncs/homes/private
public	65534	65534	\$6\$	/var/ncs/homes/public/.ssh	/var/ncs/homes/public

Change from Config to Edit config and we click on “+” button.

name	uid	gid	password	ssh_keydir	homedir
admin	65534	65534	\$6\$JHFjLfaShdkSx8QQ\$.pxYVNPY6QOZKY.D8.30	/var/ncs/homes/admin/.ssh	/var/ncs/homes/admin
oper	65534	65534	\$6\$23vdVhL6zXVJkbU\$.avJlOPHsff4nfIZSH7w/	/var/ncs/homes/oper/.ssh	/var/ncs/homes/oper
private	65534	65534	\$6\$	/var/ncs/homes/private/.ssh	/var/ncs/homes/private
public	65534	65534	\$6\$	/var/ncs/homes/public/.ssh	/var/ncs/homes/public

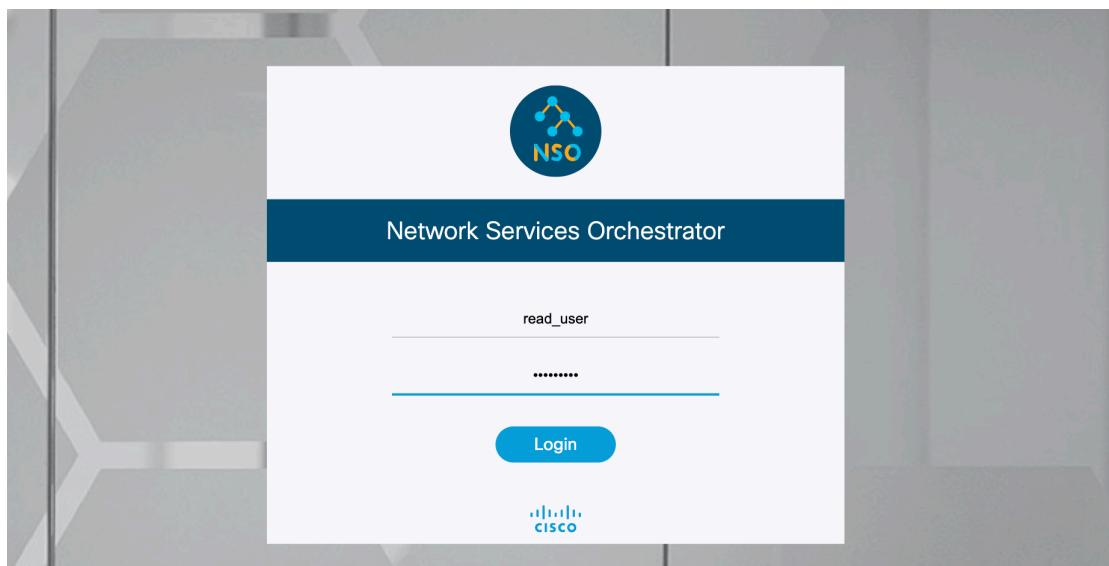
Let's create the user “read_user”.

The screenshot shows the Cisco Configuration editor interface. In the center, a modal window titled "Add new list item" is open, prompting for configuration details for a new user named "read_user". The fields include "uid" (set to 1), "gid" (set to 1), "password" (set to a masked value), "ssh_keydir" (set to "/var/ncs/homes/public/.ssh"), and "homedir" (set to "/var/ncs/homes/public"). At the bottom of the modal are "cancel" and "confirm" buttons. The background shows a list of existing users: admin, oper, private, and public. To the right, a sidebar displays file paths for SSH key directories: /var/ncs/homes/admin/.ssh, /var/ncs/homes/oper/.ssh, /var/ncs/homes/private/.ssh, and /var/ncs/homes/public/.ssh. The bottom navigation bar includes icons for Commit manager, Configuration editor, Alarm manager, Dashboard, Device manager, Service manager, Package upgrade, and Insights manager.

After this we can “Commit”.

The screenshot shows the Commit manager interface. It lists several configuration changes made in the transaction, including setting the homedir for the "read_user" to "/var/ncs/homes/public", setting the uid to 1, and creating the user. A modal window titled "Commit changes to NSO?" is displayed, asking if the user wants to commit these changes. The modal has "cancel" and "Yes, commit" buttons. The top right corner of the screen shows the "admin" status.

And login to the new recently created user



We can now go to the Device Manager and check if this user is able to do all the normal operations.

The screenshot shows the Cisco Device manager interface. At the top, there are tabs for 'Device manager' and 'read_user'. Below the tabs, there's a toolbar with icons for add, delete, and search, followed by the page number '4 / 4'. The main area displays a table of devices. One row for 'ios-xr-0' has its 'ping' button highlighted in red, indicating a failure. A message box shows the error details: 'Action: ping', 'Performed: 2023-01-25 23:06:05', 'Status: error', 'Result: error', 'Method failed', and 'Reason: Resource authgroup for read_user doesn't exist'. Other rows for 'ios-xr-1', 'ios-xr-2', and 'ios-xr-3' show successful ping operations.

And, as we can see Ping is not possible because the new user does not belong to any authgroup.

The Authgroup is the feature that allows mapping the NSO local user to the device local user. Each Device must belong to an authgroup, and each user must be present in one authgroup as well (or, have a default-mapping per authgroup)

To create a new Authgroup or associate the user with an existing authgroup we go to the Configuration Editor, to the module ncs:devices

The screenshot shows the Configuration editor interface. At the top, there are tabs for 'Configuration editor' and 'read_user'. Below the tabs, there's a toolbar with icons for add, delete, and search, followed by the page number '1 / 1'. The main area displays the 'MODULES' section. Under the 'ncs:devices' module, several sub-modules are listed: aaa:aaa, aaa:alias, aaa:session, aaa:user, al:alarms, last:last-logins, ncm:netconf-state, ncs:cluster, ncs:compliance, ncs:customers, ncs:devices, ncs:high-availability, ncs:java-vm, ncs:metric, ncs:packages, ncs:python-vm, ncs:services, ncs:side-effect-queue, ncs:smart-license, ncs:software, ncs:sshd, ncs:zombies, NTP:NTP, rcmon:restconf-state, scheduler:scheduler, snmp:snmp, tftp:policy, tfm:ncs-state, webui:webui.

Then we select authgroup “default”

The screenshot shows the Configuration editor interface. At the top, there are tabs for 'Configuration editor' and 'read_user'. Below the tabs, there's a toolbar with icons for edit config, config, operdata, and actions, followed by the page number '1 / 1'. The main area displays the 'ncs:devices/authgroups' module. It shows two groups: 'group' and 'snmp-group'. Under 'group', there are entries for 'name' and 'default'. Under 'snmp-group', there are also entries for 'name' and 'default'.

At this moment we can make these operations on the “read_user” since we didn’t made any change to user permissions.

The screenshot shows the Cisco Configuration editor interface. The top navigation bar includes tabs for 'Edit config' (which is selected), 'Config', 'Operdata', and 'Actions'. On the right, there are buttons for 'None', 'Containers', and a user dropdown set to 'read_user'. The main content area displays the configuration for an 'authgroups/group/default' entry. The 'name' field is set to 'default'. Below it, the 'umap' section is expanded, showing a list of users: 'local-user', 'admin', and 'oper'. A modal dialog box is open, titled 'Add new list item', with the 'local-user' entry already populated in the input field. At the bottom of the dialog are 'cancel' and 'confirm' buttons.

Now, inside the “default” authgroup we’ve the umap and default-map.

The umap, is a mapping for a local user that belongs to this Authgroup.

And we’ve a default-map, this one is a default mapping for each user that belongs to this group.

Let’s add our read_user to the “default” authgroup.

This screenshot shows the same configuration interface as the previous one, but with a modal dialog open over the 'umap' list. The dialog is titled 'Add new list item' and contains a single input field with the value 'read_user'. At the bottom of the dialog are 'cancel' and 'confirm' buttons. The background shows the 'umap' list with 'local-user', 'admin', and 'oper' entries.

Now, you can choose what will be the credentials that this NSO user will use when connecting to the devices. (we can use admin-admin for the demo proposes).

The screenshot shows the Cisco Configuration editor interface. At the top, there are tabs for 'Edit config' (which is selected), 'Config', 'Operdata', and 'Actions'. On the right, there are buttons for 'None', 'Containers', and a dropdown for 'read_user'. Below the tabs, the URL is displayed as /ncs/devices/authgroups/group/default1/umap/read_user/. The main area contains configuration sections for 'local-user' and 'read_user'. Under 'login-credentials', there are tabs for 'stored' (selected) and 'callback'. The 'remote-user' section shows 'same-user' and 'remote-name' defined. The 'remote-auth' section shows 'remote-password' defined. The 'remote-secondary-auth' section shows 'remote-secondary-password' defined. At the bottom, there are navigation icons for Commit manager (C*), Configuration editor (E2), Alarm manager (A), Dashboard (B), Device manager (D), Service manager (S), Package upgrade (P), and Insights manager (I).

Go to the commit manager and commit.

Going now back to the “Device Manager” we can now check the Ping again.

The screenshot shows the Device manager interface. At the top, there are icons for adding a device, deleting, and filtering, followed by the text 'Device manager' and 'IOS VERSION 8.0'. On the right, there are buttons for 'read_user' and '+Add filter'. Below the header, there is a table with 4 rows. The first three rows have checkboxes checked. The columns are: name, address, port, type, services, ping, connect, check-sync, sync-from, sync-to, compare-config, alarm, and configuration. The last row has no checked checkboxes. The table is labeled '4 / 4' at the top right.

For now, this user can make all the normal operations.

Let's start making some restrictions.

Go back to the admin user.

Then, go to the configuration Editor.

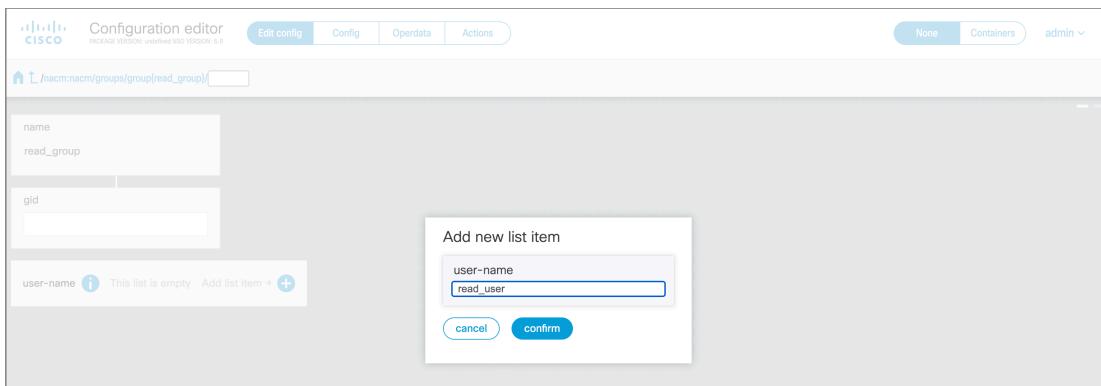
And then to the module `nacm:nacm`

In the NACM module you will see the “rule-list” and the “groups”

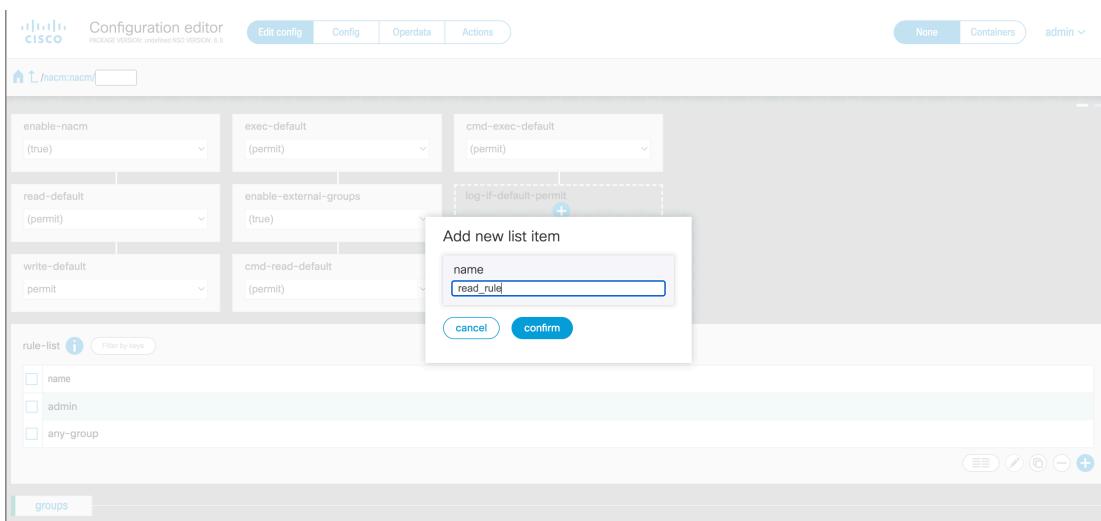
So, to set permissions we need to associate a user to a group, and then associate that group to a rule-list.

Let's create the “read_group”

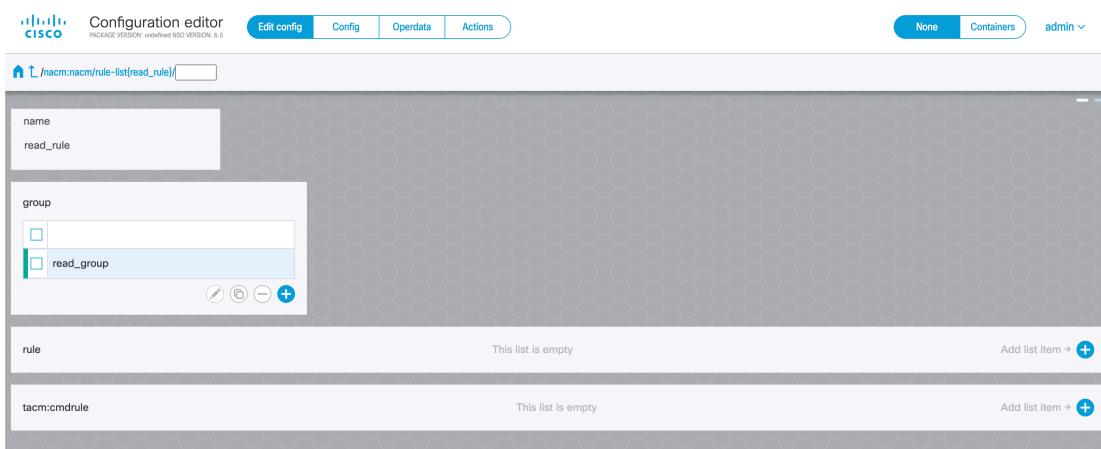
Then we setup one group-id (1 for example) and we add the user “read_user” to that group



After this, we can go back and create the rule-list "read_rule"



On this rule, we will add the read_group in groups. This means that all the rules that we will setup here will affect only the users that belong to this group.



So, now we've 2 options, the cmdrule where the entries list are examined for command authorization, and the rule where entries are examined for rpc, notification, and data authorization.

Let's add a rule to deny the sync-from

The screenshot shows the Cisco Configuration editor interface. In the center, a modal window titled "Add new list item" is open. It contains two fields: "name" with the value "deny-sync-from" and "action" with the value "deny". Below these fields are "cancel" and "confirm" buttons. To the left of the modal, there's a sidebar with a tree view under the "group" section, showing a node named "read_group". At the bottom of the screen, there are tabs for "Edit config", "Config", "Operdata", and "Actions". A top navigation bar includes the Cisco logo, the title "Configuration editor", and user information like "None", "Containers", and "admin".

In the rule we've to specify a path

This screenshot shows the configuration of a specific NACM rule. The main panel displays the rule details: name "deny-sync-from", action "deny", module-name "(*")", access-operations "(*)", and context "(*)". Below this, the "rule-type" section is expanded, showing the "data-node" tab selected. Under "path*", the value "/devices/device sync-from" is entered. On the right side of the screen, there's a large, semi-transparent hexagonal overlay that obscures much of the interface. The top navigation bar and tabs are visible at the top.

This path is nothing more than an XPATH. Like we saw before XPATHs can be found in many ways. In the next screenshot we can see the paths for the device IOS-XR-1 with the configurations in place.

This screenshot shows the list of paths for the device "ios-xr-1". The list is quite long and includes various configuration nodes such as "/devices/device[name='ios-xr-1']/address", "/devices/device[name='ios-xr-1']/port", and numerous SSH and NTP-related configurations. The top navigation bar and tabs are visible at the top.

For example, we can grab the path "/devices/device[name='ios-xr-1']/config/cisco-ios-xr:domain" and restrict all the operations with the exception of reading.

We create the rule "deny-config-domain"

And inside the rule on the data-node we use the path that we just got, we setup the action to deny and access-operations we will deny “create,update,delete,exec”

To see all the access-operations you can toggle the button near “view options” on the top right and then the info button will appear in each tile.

After this, we commit.

We logout from the admin user and we login into the read_user.

Now we go to the Device Manager and we test our rules. We can check the “ping”, “connect”, “check-sync” and “sync-from” and we will see that the “sync-from” operation got denied.

name	address	port	type	services	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration
ios-xr-0	127.0.0.1	10022	cisco-iosxr-cl-7.43...cisco-iosxr-cl-7.43	1	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration
ios-xr-1	127.0.0.1	10023	cisco-iosxr-cl-7.43...cisco-iosxr-cl-7.43	1	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration
ios-xr-2	127.0.0.1	10024	cisco-iosxr-cl-7.43...cisco-iosxr-cl-7.43	1	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration
ios-xr-3	127.0.0.1	10025	cisco-iosxr-cl-7.43...cisco-iosxr-cl-7.43	1	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration

Now, if we go to the device ios-xr-1 and we go into config/domain we will see that we’re only able to read the info

And, if we try to remove anything by clicking on the - button we will get the “access denied” notification.

Since we've made the rule just for the ios-xr-1 device, if we go to the other devices, for example ios-xr-0 we see that we can still make changes in the domain configuration.

The screenshot shows the Configuration editor interface for device 'ios-xr-0'. The top navigation bar includes tabs for 'Edit config', 'Config', 'Operdata', and 'Actions'. The user is logged in as 'read_user'. The main content area displays the configuration for the 'cisco-ios-xr:domain' package. The 'name' field is set to 'cisco.com'. In the 'name-server' section, the 'address' field is populated with '2.2.2.2'. The 'vrf' section is currently empty. A '+' button is available to add more list items.

To apply this to all devices we need to login into the admin user, and change the PATH of this rule.

So, instead of "/devices/device[name='ios-xr-1']/config/cisco-ios-xr:domain/"

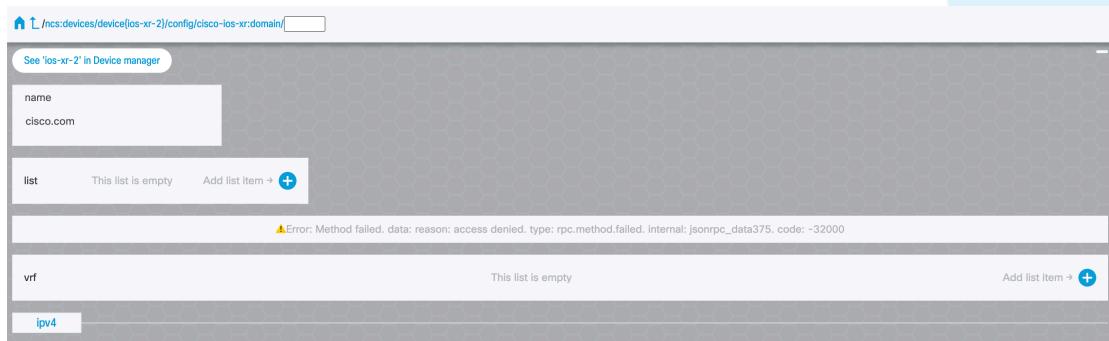
We will change to "/devices/device/config/cisco-ios-xr:domain/"

The screenshot shows the Configuration editor interface for a NACM rule named 'deny-config-domain'. The 'action' is set to 'deny'. The 'rule-type' is 'data-node'. The 'path' field is being modified from '/devices/device/config/cisco-ios-xr:domain' to '/devices/device/config/cisco-ios-xr:domain'. The 'comment' field is '(*)'. The 'access-operations' field includes 'create update delete exec'. The 'context' field is '(*)'. A 'log-if-permit' action is also present in the rule definition.

Commit and go back to the read_user.

We can see that after this change, we're not able to make any change in domain configuration on all managed devices with this user.

The screenshot shows the Configuration editor interface for device 'ios-xr-0' after the path change. The 'name-server' section now has both 'address' and '2.2.2.2' checked, with a blue info icon next to the 'address' entry. The 'vrf' section remains empty. The '+' button is available to add more list items.



Thank you !