



Amsterdam | February 6–10, 2023



DEVLIT-1259

Workbook The World of NSO WEB UI

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)	12
1.3 – Configure Devices using NSO	14
1.4 – Using Rollback's	17
1.5 – Using NSO capabilities to detect Out-of-Band device configurations	18
1.6 – Create Device Groups and Device Templates	21
1.7 – Create a simple NSO Service	27
1.8 – NSO APIs.....	42
1.9 – Create Role Based and Resource Based Access Control rules	47

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 it's operations basics, always using the WEB-UI.

NSO can be installed on a MAC, or Linux. Opening this link you will find the operating system requirements and how to install them.

<https://developer.cisco.com/docs/ns0/#!getting-and-installing-ns0/requirements>

1.1 - Install NSO and Network Element Drivers (NEDs)

Download NSO for your Operating System and the NEDs

<https://developer.cisco.com/docs/ns0/#!getting-and-installing-ns0/download-your-ns0-free-trial-installer-and-cisco-neds>

The screenshot shows the 'Software Download' page on the Cisco developer website. The main heading is 'Network Services Orchestrator Free Trial'. Below it, there is a table listing various software packages:

File Information	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

```
jomiraj@M1Pro2023 # % ls
ncs-6.0-cisco-iosxr-7.43-freetrial.signed.bin
nso-6.0-freetrial.darwin.x86_64.signed.bin
```

In this case you see that it was downloaded the MAC version (darwin) of NSO and IOS XR NED.

```
jomira@M1Pro2023 # % 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-6.0.darwin.x86_64.installer.bin” with the “—local-install” flag and choose the installation directory.

```
jomira@M1Pro2023 # % sh nso-6.0.darwin.x86_64.installer.bin --
local-install ~/NSO-CLEU
INFO Using temporary directory
/var/folders/48/yvryv4r96jq9n8gt3hpm13w40000gn/T//ncs_installer.11084 to stage NCS installation bundle
INFO Unpacked ncs-6.0 in /Users/jomira/NSO-CLEU
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/jomira/NSO-CLEU/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

```
jomira@M1Pro2023 # % cd ~/NSO-CLEU
jomira@M1Pro2023 NSO-CLEU % source ncsrc
```

Execute the NSO Setup script

```
jomira@M1Pro2023 NSO-CLEU % ncs-setup --dest nso-instance
```

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

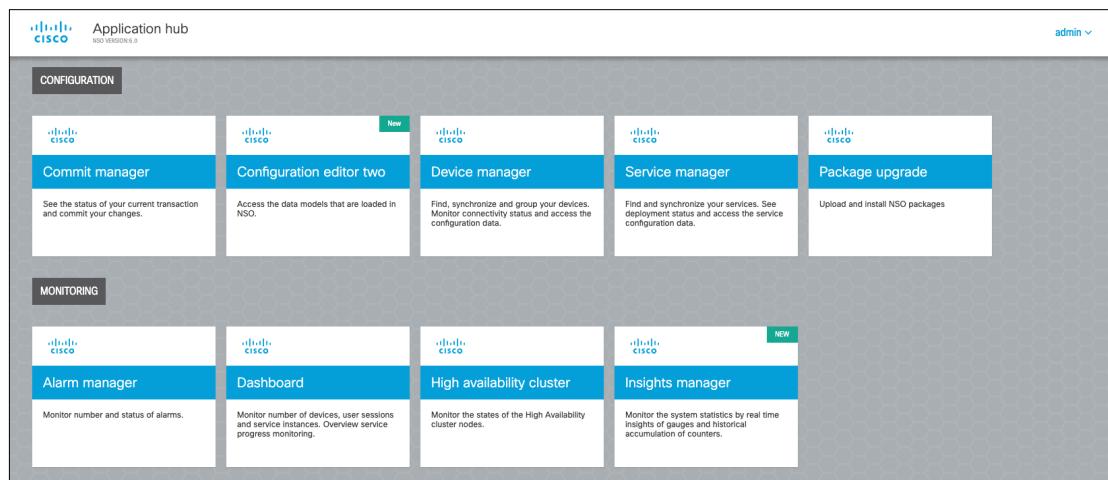
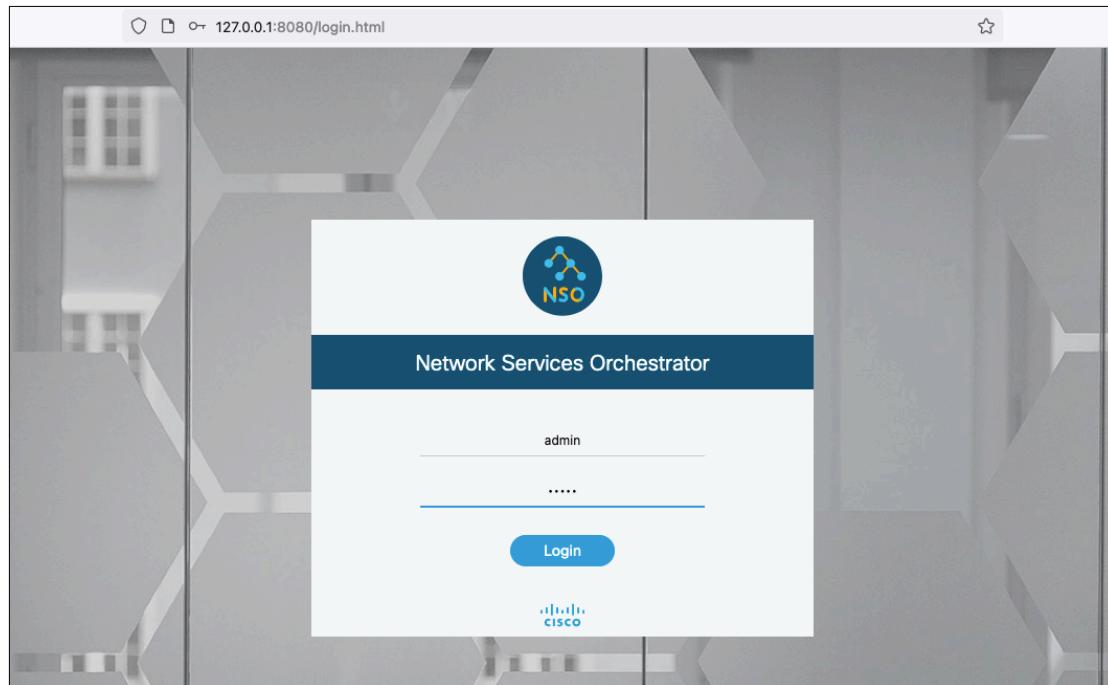
```
jomira@M1Pro2023 NSO-CLEU % cd nso-instance
jomira@M1Pro2023 nso-instance % ncs
```

Check if NSO is running

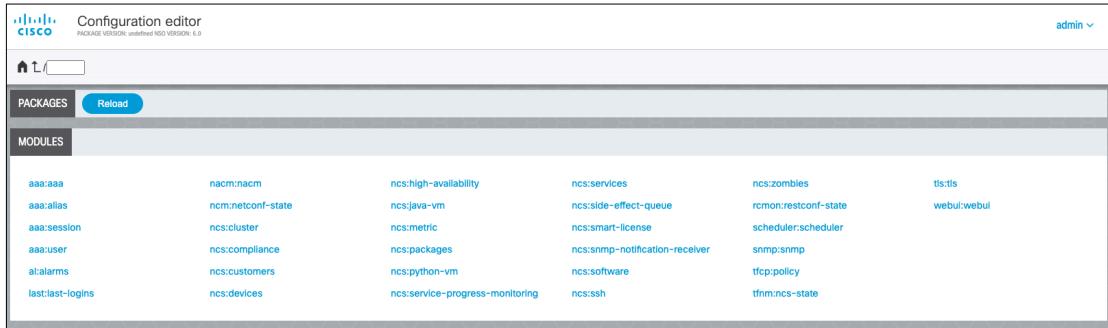
```
jomira@M1Pro2023 nso-instance % ncs --status | grep started
status: started
```

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”.

```
jomira@M1Pro2023 # % 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
```

Extract the “.tar.gz” file

```
jomira@M1Pro2023 # % tar -zxvf ncs-6.0-cisco-iosxr-7.43.tar.gz
x cisco-iosxr-cli-7.43/
x cisco-iosxr-cli-7.43/package-meta-data.xml
x cisco-iosxr-cli-7.43/netsim/
x cisco-iosxr-cli-7.43/netsim/confd.i.ccl
x cisco-iosxr-cli-7.43/netsim/confd.conf.netsim.strict
x cisco-iosxr-cli-7.43/netsim/show_debug.sh
x cisco-iosxr-cli-7.43/netsim/confd.c.ccl
x cisco-iosxr-cli-7.43/netsim/Makefile
x cisco-iosxr-cli-7.43/netsim/populate.sh
x cisco-iosxr-cli-7.43/netsim/show_version.sh
x cisco-iosxr-cli-7.43/netsim/exec_command.sh
x cisco-iosxr-cli-7.43/netsim/confd.i.cli
x cisco-iosxr-cli-7.43/netsim/aaa.fxs
x cisco-iosxr-cli-7.43/netsim/show_inventory.sh
x cisco-iosxr-cli-7.43/netsim/show_interfaces.sh
x cisco-iosxr-cli-7.43/netsim/show_diag.sh
x cisco-iosxr-cli-7.43/netsim/tailf-ned-cisco-ios-xr.fxs
x cisco-iosxr-cli-7.43/netsim/start.sh
x cisco-iosxr-cli-7.43/netsim/confd.c.cli
x cisco-iosxr-cli-7.43/netsim/ietf-netconf-acm.fxs
```

```
x cisco-iosxr-cli-7.43/netsim/confd.conf.netsim
x cisco-iosxr-cli-7.43/shared-jar/
x cisco-iosxr-cli-7.43/shared-jar/iosxr-ns.jar
x cisco-iosxr-cli-7.43/doc/
x cisco-iosxr-cli-7.43/src/
x cisco-iosxr-cli-7.43/src/gili/
x cisco-iosxr-cli-7.43/src/gili/README.gili
x cisco-iosxr-cli-7.43/src/gili/cdp_neighbors.gili
x cisco-iosxr-cli-7.43/src/gili/controllers-optics.gili
x cisco-iosxr-cli-7.43/src/gili/if:interfaces-
state_interface_ip:ipv4_address.gili
x cisco-iosxr-cli-7.43/src/gili/if:interfaces-
state_interface_ip:ipv6_address.gili
x cisco-iosxr-cli-7.43/src/gili/lldp_neighbors.gili
x cisco-iosxr-cli-7.43/src/gili/if:interfaces-
state_interface.gili
x cisco-iosxr-cli-7.43/src/gili/inventory.gili
x cisco-iosxr-cli-7.43/src/gili/properties.gili
x cisco-iosxr-cli-7.43/src/gili/controllers-optics-
instance.gili
x cisco-iosxr-cli-7.43/src/nedcom.mk
x cisco-iosxr-cli-7.43/src/metadata/
x cisco-iosxr-cli-7.43/src/metadata/ned-meta-data.properties
x cisco-iosxr-cli-7.43/src/metadata/ned-connector-default.json
x cisco-iosxr-cli-7.43/src/metadata/ned-connector-proxy.json
x cisco-iosxr-cli-7.43/src/metadata/README
x cisco-iosxr-cli-7.43/src/metadata/ned-connector-serial.json
x cisco-iosxr-cli-7.43/src/tools/
x cisco-iosxr-cli-7.43/src/tools/ypp.py
x cisco-iosxr-cli-7.43/src/tools/ypp
x cisco-iosxr-cli-7.43/src/Makefile
x cisco-iosxr-cli-7.43/src/ncsc-out/
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/fxs/
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/fxs/tailf-ned-
cisco-ios-xr-oper.fxs
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/fxs/tailf-ned-
cisco-ios-xr-id.fxs
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/fxs/ietf-
interfaces.fxs
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/fxs/tailf-ned-
cisco-ios-xr-meta.fxs
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/fxs/tailf-ned-
cisco-ios-xr-stats.fxs
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/fxs/ietf-ip.fxs
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/fxs/tailf-ned-
secrets.fxs
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/fxs/cliparser-
extensions-v11.fxs
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/fxs/tailf-ned-
loginscripts.fxs
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/fxs/tailf-ned-
cisco-ios-xr.fxs
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/ietf-
interfaces/
```

```
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/ietf-
interfaces/2014-05-08/
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/ietf-
interfaces/2014-05-08/ietf-interfaces.fxs
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/ietf-
interfaces/2014-05-08/ietf-interfaces.yang.orig
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/ietf-
interfaces/2014-05-08/ietf-interfaces.yang
x cisco-iosxr-cli-7.43/src/ncsc-
out/modules/revisions/cliparser-extensions-v11/
x cisco-iosxr-cli-7.43/src/ncsc-
out/modules/revisions/cliparser-extensions-v11/norev/
x cisco-iosxr-cli-7.43/src/ncsc-
out/modules/revisions/cliparser-extensions-
v11/norev/cliparser-extensions-v11.yang.orig
x cisco-iosxr-cli-7.43/src/ncsc-
out/modules/revisions/cliparser-extensions-
v11/norev/cliparser-extensions-v11.yang
x cisco-iosxr-cli-7.43/src/ncsc-
out/modules/revisions/cliparser-extensions-
v11/norev/cliparser-extensions-v11.fxs
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/ietf-ip/
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/ietf-
ip/2014-06-16/
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/ietf-
ip/2014-06-16/ietf-ip.fxs
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/ietf-
ip/2014-06-16/ietf-ip.yang
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/ietf-
ip/2014-06-16/ietf-ip.yang.orig
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/tailf-
ned-cisco-ios-xr-id/
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/tailf-
ned-cisco-ios-xr-id/norev/
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/tailf-
ned-cisco-ios-xr-id/norev/tailf-ned-cisco-ios-xr-id.fxs
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/tailf-
ned-cisco-ios-xr-id/norev/tailf-ned-cisco-ios-xr-id.yang
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/tailf-
ned-cisco-ios-xr-id/norev/tailf-ned-cisco-ios-xr-id.yang.orig
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/tailf-
ned-cisco-ios-xr-stats/
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/tailf-
ned-cisco-ios-xr-stats/norev/
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/tailf-
ned-cisco-ios-xr-stats/norev/tailf-ned-cisco-ios-xr-stats.fxs
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/tailf-
ned-cisco-ios-xr-stats/norev/tailf-ned-cisco-ios-xr-stats.yang
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/tailf-
ned-cisco-ios-xr-stats/norev/tailf-ned-cisco-ios-xr-
stats.yang.orig
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/tailf-
ned-cisco-ios-xr/
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/tailf-
ned-cisco-ios-xr/2022-10-07/
```

```
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/tailf-
ned-cisco-ios-xr/2022-10-07/tailf-ned-cisco-ios-xr.yang.orig
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/tailf-
ned-cisco-ios-xr/2022-10-07/tailf-ned-cisco-ios-xr.fxs
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/revisions/tailf-
ned-cisco-ios-xr/2022-10-07/tailf-ned-cisco-ios-xr.yang
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/yang/
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/yang/cliparser-
extensions-v11.yang
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/yang/ietf-ip.yang
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/yang/ietf-
interfaces.yang
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/yang/tailf-ned-
cisco-ios-xr-id.yang
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/yang/tailf-ned-
cisco-ios-xr-stats.yang
x cisco-iosxr-cli-7.43/src/ncsc-out/modules/yang/tailf-ned-
cisco-ios-xr.yang
x cisco-iosxr-cli-7.43/src/cisco-iosxr-cli.yang
x cisco-iosxr-cli-7.43/src/yang/
x cisco-iosxr-cli-7.43/src/yang/tailf-ned-cisco-ios-xr-
meta.yang
x cisco-iosxr-cli-7.43/src/yang/cliparser-extensions-v11.yang
x cisco-iosxr-cli-7.43/src/yang/ietf-ip.yang
x cisco-iosxr-cli-7.43/src/yang/tailf-ned-loginscripts.yang
x cisco-iosxr-cli-7.43/src/yang/tailf-ned-cisco-ios-xr-
oper.yang
x cisco-iosxr-cli-7.43/src/yang/ietf-interfaces.yang
x cisco-iosxr-cli-7.43/src/yang/tailf-ned-secrets.yang
x cisco-iosxr-cli-7.43/src/yang/tailf-ned-cisco-ios-xr-
stats.yang
x cisco-iosxr-cli-7.43/src/yang/tailf-ned-cisco-ios-xr.yang
x cisco-iosxr-cli-7.43/src/cisco-iosxr-cli-7.43.yang
x cisco-iosxr-cli-7.43/src/ned-yang-filter.mk
x cisco-iosxr-cli-7.43/src/java/
x cisco-iosxr-cli-7.43/src/java/build.xml
x cisco-iosxr-cli-7.43/src/java/src/
x cisco-iosxr-cli-7.43/src/java/src/com/
x cisco-iosxr-cli-7.43/src/java/src/com/tailf/
x cisco-iosxr-cli-7.43/src/java/src/com/tailf/packages/
x cisco-iosxr-cli-7.43/src/java/src/com/tailf/packages/ned/
x cisco-iosxr-cli-
7.43/src/java/src/com/tailf/packages/ned/iosxr/
x cisco-iosxr-cli-
7.43/src/java/src/com/tailf/packages/ned/iosxr/IosxrCliExtensi-
ons.java
x cisco-iosxr-cli-
7.43/src/java/src/com/tailf/packages/ned/iosxr/IosxrNedCli.jav-
a
x cisco-iosxr-cli-
7.43/src/java/src/com/tailf/packages/ned/iosxr/NedCommand.java
x cisco-iosxr-cli-
7.43/src/java/src/com/tailf/packages/ned/iosxr/UpgradeNedSetti-
ngs.java
x cisco-iosxr-cli-
7.43/src/java/src/com/tailf/packages/ned/iosxr/namespaces/
```

```
x cisco-iosxr-cli-
7.43/src/java/src/com/tailf/packages/ned/iosxr/namespaces/ietf
Interfaces.java
x cisco-iosxr-cli-
7.43/src/java/src/com/tailf/packages/ned/iosxr/namespaces/clip
arserExtensionsV11.java
x cisco-iosxr-cli-
7.43/src/java/src/com/tailf/packages/ned/iosxr/namespaces/tail
fNedCiscoIosXr.java
x cisco-iosxr-cli-
7.43/src/java/src/com/tailf/packages/ned/iosxr/namespaces/tail
fNedCiscoIosXrStats.java
x cisco-iosxr-cli-
7.43/src/java/src/com/tailf/packages/ned/iosxr/namespaces/ietf
Ip.java
x cisco-iosxr-cli-
7.43/src/java/src/com/tailf/packages/ned/iosxr/ModeStack.java
x cisco-iosxr-cli-7.43/src/java/build/
x cisco-iosxr-cli-7.43/src/java/build/classes/
x cisco-iosxr-cli-7.43/src/java/build/classes/com/
x cisco-iosxr-cli-7.43/src/java/build/classes/com/tailf/
x cisco-iosxr-cli-
7.43/src/java/build/classes/com/tailf/packages/
x cisco-iosxr-cli-
7.43/src/java/build/classes/com/tailf/packages/ned/
x cisco-iosxr-cli-
7.43/src/java/build/classes/com/tailf/packages/ned/iosxr/
x cisco-iosxr-cli-
7.43/src/java/build/classes/com/tailf/packages/ned/iosxr/Iosxr
CliExtensions.class
x cisco-iosxr-cli-
7.43/src/java/build/classes/com/tailf/packages/ned/iosxr/Iosxr
NedCli$1PathComp.class
x cisco-iosxr-cli-
7.43/src/java/build/classes/com/tailf/packages/ned/iosxr/ModeS
tack.class
x cisco-iosxr-cli-
7.43/src/java/build/classes/com/tailf/packages/ned/iosxr/Iosxr
NedCli$ShowHandler.class
x cisco-iosxr-cli-
7.43/src/java/build/classes/com/tailf/packages/ned/iosxr/Iosxr
NedCli.class
x cisco-iosxr-cli-
7.43/src/java/build/classes/com/tailf/packages/ned/iosxr/Upgra
deNedSettings.class
x cisco-iosxr-cli-
7.43/src/java/build/classes/com/tailf/packages/ned/iosxr/NedCo
mmand.class
x cisco-iosxr-cli-
7.43/src/java/build/classes/com/tailf/packages/ned/iosxr/Iosxr
NedCli$1.class
x cisco-iosxr-cli-
7.43/src/java/build/classes/com/tailf/packages/ned/iosxr/names
paces/
```

```

x cisco-iosxr-cli-
7.43/src/java/build/classes/com/tailf/packages/ned/iosxr/names
paces/ietfIp.class
x cisco-iosxr-cli-
7.43/src/java/build/classes/com/tailf/packages/ned/iosxr/names
paces/tailfNedCiscoIosXr.class
x cisco-iosxr-cli-
7.43/src/java/build/classes/com/tailf/packages/ned/iosxr/names
paces/ietfInterfaces.class
x cisco-iosxr-cli-
7.43/src/java/build/classes/com/tailf/packages/ned/iosxr/names
paces/tailfNedCiscoIosXrStats.class
x cisco-iosxr-cli-
7.43/src/java/build/classes/com/tailf/packages/ned/iosxr/names
paces/cliparserExtensionsV11.class
x cisco-iosxr-cli-7.43/src/package-meta-data.xml.in
x cisco-iosxr-cli-7.43/src/tailf-ned-cisco-ios-xr-id.yang
x cisco-iosxr-cli-7.43/load-dir/
x cisco-iosxr-cli-7.43/load-dir/tailf-ned-cisco-ios-xr-
oper.fxs
x cisco-iosxr-cli-7.43/load-dir/tailf-ned-cisco-ios-xr-id.fxs
x cisco-iosxr-cli-7.43/load-dir/ietf-interfaces.fxs
x cisco-iosxr-cli-7.43/load-dir/tailf-ned-cisco-ios-xr-
meta.fxs
x cisco-iosxr-cli-7.43/load-dir/tailf-ned-id-cisco-iosxr-clि-
7.43.fxs
x cisco-iosxr-cli-7.43/load-dir/tailf-ned-cisco-ios-xr-
stats.fxs
x cisco-iosxr-cli-7.43/load-dir/ietf-ip.fxs
x cisco-iosxr-cli-7.43/load-dir/tailf-ned-secrets.fxs
x cisco-iosxr-cli-7.43/load-dir/cliparser-extensions-v11.fxs
x cisco-iosxr-cli-7.43/load-dir/tailf-ned-loginscripts.fxs
x cisco-iosxr-cli-7.43/load-dir/tailf-ned-cisco-ios-xr.fxs
x cisco-iosxr-cli-7.43/load-dir/tailf-ned-id-cisco-iosxr-
cli.fxs
x cisco-iosxr-cli-7.43/build-meta-data.xml
x cisco-iosxr-cli-7.43/LICENSE
x cisco-iosxr-cli-7.43/LICENSES.nedcom/
x cisco-iosxr-cli-7.43/LICENSES.nedcom/APACHE_V2_LICENSE
x cisco-iosxr-cli-7.43/LICENSES.nedcom/CUP_LICENSE
x cisco-iosxr-cli-7.43/LICENSES.nedcom/README
x cisco-iosxr-cli-7.43/README
x cisco-iosxr-cli-7.43/CHANGES
x cisco-iosxr-cli-7.43/private-jar/
x cisco-iosxr-cli-7.43/private-jar/nedcom.jar
x cisco-iosxr-cli-7.43/private-jar/iosxr.jar
x cisco-iosxr-cli-7.43/private-jar/cisco-serializer.2.7.2b.jar
x cisco-iosxr-cli-7.43/private-jar/legacy-connector-1.0.jar
x cisco-iosxr-cli-7.43/private-jar/cisco-xalan.2.7.2b.jar
x cisco-iosxr-cli-7.43/private-jar/nanojson-1.2.jar

```

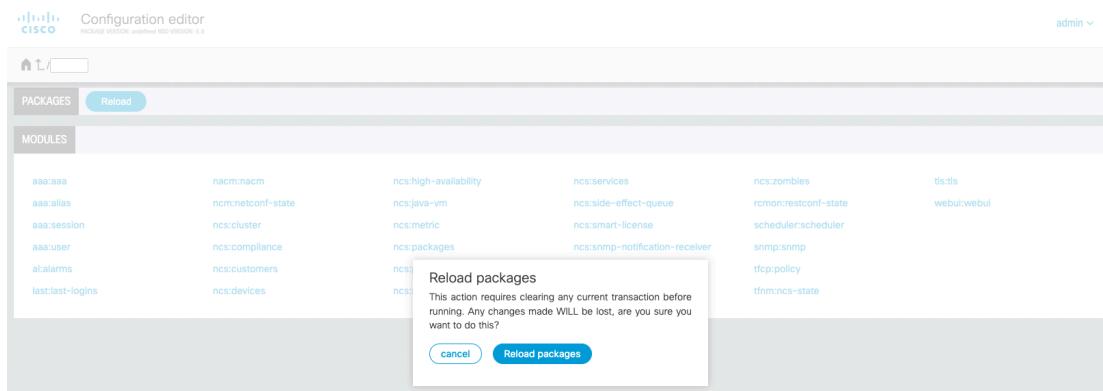
Copy the extracted directory to “nso-instance/packages” folder

```

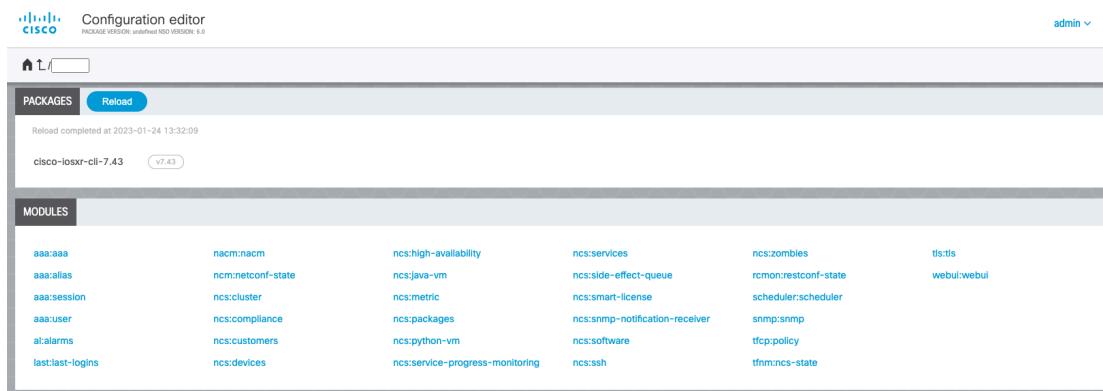
jomira@M1Pro2023 # % cp -R cisco-iosxr-cli-7.43 ~/NSO-
CLEU/nso-instance/packages

```

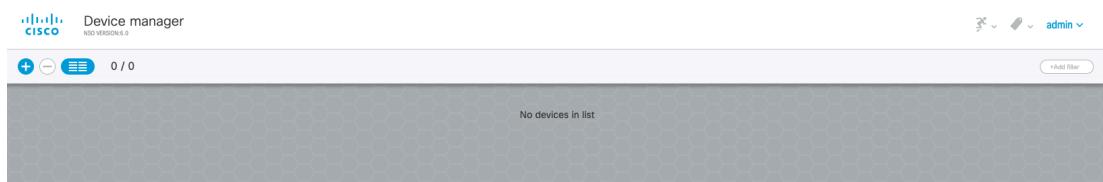
Go back to NSO WEB UI and Click on “Reload” packages button



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

```
jomira@M1Pro2023 nso-instance % ncs-netsim create-network  
packages/cisco-iosxr-cli-7.43 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.

```
jomira@M1Pro2023 nso-instance % ncs-netsim list
ncs-netsim list for /Users/jomira/NSO-CLEU/nso-
instance/netsim

name=ios-xr-0 netconf=12022 snmp=11022 ipc=5010 cli=10022
dir=/Users/jomira/NSO-CLEU/nso-instance/netsim/ios-xr-/ios-xr-
0
name=ios-xr-1 netconf=12023 snmp=11023 ipc=5011 cli=10023
dir=/Users/jomira/NSO-CLEU/nso-instance/netsim/ios-xr-/ios-xr-
1
name=ios-xr-2 netconf=12024 snmp=11024 ipc=5012 cli=10024
dir=/Users/jomira/NSO-CLEU/nso-instance/netsim/ios-xr-/ios-xr-
2
name=ios-xr-3 netconf=12025 snmp=11025 ipc=5013 cli=10025
dir=/Users/jomira/NSO-CLEU/nso-instance/netsim/ios-xr-/ios-xr-
3
```

Start the devices.

```
jomira@M1Pro2023 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”. Make a quick Loopback 0 configuration.

```
jomira@M1Pro2023 nso-instance % ncs-netsim cli-c ios-xr-0

admin connected from 127.0.0.1 using console on
M1Pro2023.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)

```

jomira@M1Pro2023 nso-instance % ncs-netsim ncs-xml-init >
devices.xml
jomira@M1Pro2023 nso-instance % ncs_load -l -m devices.xml

```

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

	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-cl-7.43...cisco-iosxr-cl-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	configuration
<input type="checkbox"/>	ios-xr-1	127.0.0.1	10023	cisco-iosxr-cl-7.43...cisco-iosxr-cl-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	configuration
<input type="checkbox"/>	ios-xr-2	127.0.0.1	10024	cisco-iosxr-cl-7.43...cisco-iosxr-cl-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	configuration
<input type="checkbox"/>	ios-xr-3	127.0.0.1	10025	cisco-iosxr-cl-7.43...cisco-iosxr-cl-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	configuration

The first thing you need to do after a device to NSO is execute the “sync-from” so you update NSO CDB with the device configuration. Select all devices and make a sync-from.

	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-cl-7.43...cisco-iosxr-cl-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	configuration
<input checked="" type="checkbox"/>	ios-xr-1	127.0.0.1	10023	cisco-iosxr-cl-7.43...cisco-iosxr-cl-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	configuration
<input checked="" type="checkbox"/>	ios-xr-2	127.0.0.1	10024	cisco-iosxr-cl-7.43...cisco-iosxr-cl-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	configuration
<input checked="" type="checkbox"/>	ios-xr-3	127.0.0.1	10025	cisco-iosxr-cl-7.43...cisco-iosxr-cl-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	configuration

	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-cl-7.43...cisco-iosxr-cl-7.43	0	ping	connect	check-sync	sync-from ▾	sync-to	compare-config	configuration	configuration
<input checked="" type="checkbox"/>	ios-xr-1	127.0.0.1	10023	cisco-iosxr-cl-7.43...cisco-iosxr-cl-7.43	0	ping	connect	check-sync	sync-from ▾	sync-to	compare-config	configuration	configuration
<input checked="" type="checkbox"/>	ios-xr-2	127.0.0.1	10024	cisco-iosxr-cl-7.43...cisco-iosxr-cl-7.43	0	ping	connect	check-sync	sync-from ▾	sync-to	compare-config	configuration	configuration
<input checked="" type="checkbox"/>	ios-xr-3	127.0.0.1	10025	cisco-iosxr-cl-7.43...cisco-iosxr-cl-7.43	0	ping	connect	check-sync	sync-from ▾	sync-to	compare-config	configuration	configuration

1.3 – Configure Devices using NSO

You can enter in ios-xr-0 to verify if the loopback 0 interface that we've configured is already there.

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

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

Let's make a change in the IP address. To do it, you just need to edit the field IP.

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.

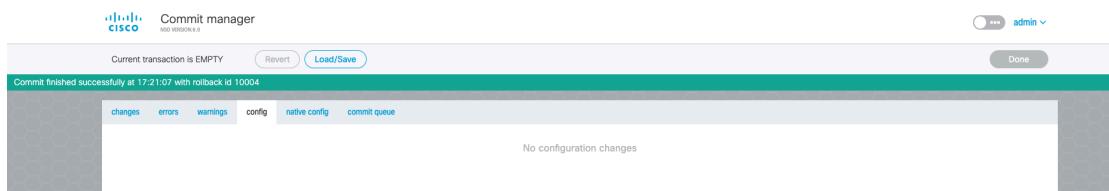
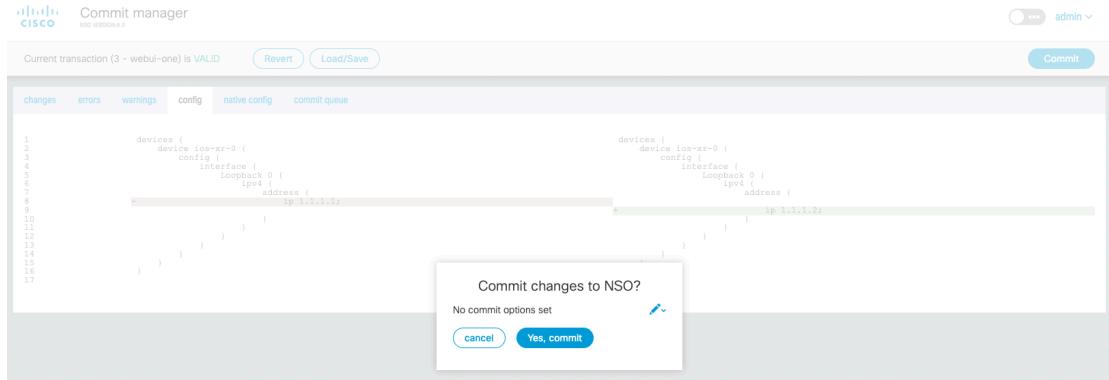
Path	Operation	Old value	New value
/ncs/devices/device[ios-xr-0]/config/cisco-ios-xr:interface/Loopback(0)/ipv4/address/ip	value_set	1.1.1.1	1.1.1.2
/ncs/devices/device[ios-xr-0]/config/cisco-ios-xr:interface/Loopback(0)	modified		
/ncs/devices/device[ios-xr-0]	modified		

```

1 devices {
2   device ios-xr-0 {
3     config {
4       interface {
5         Loopback 0 {
6           ipv4 {
7             address {
8               ip 1.1.1.1;
9             }
10            )
11          )
12        )
13      )
14    )
15  )
16}
17

```

In the commit manager we have 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.

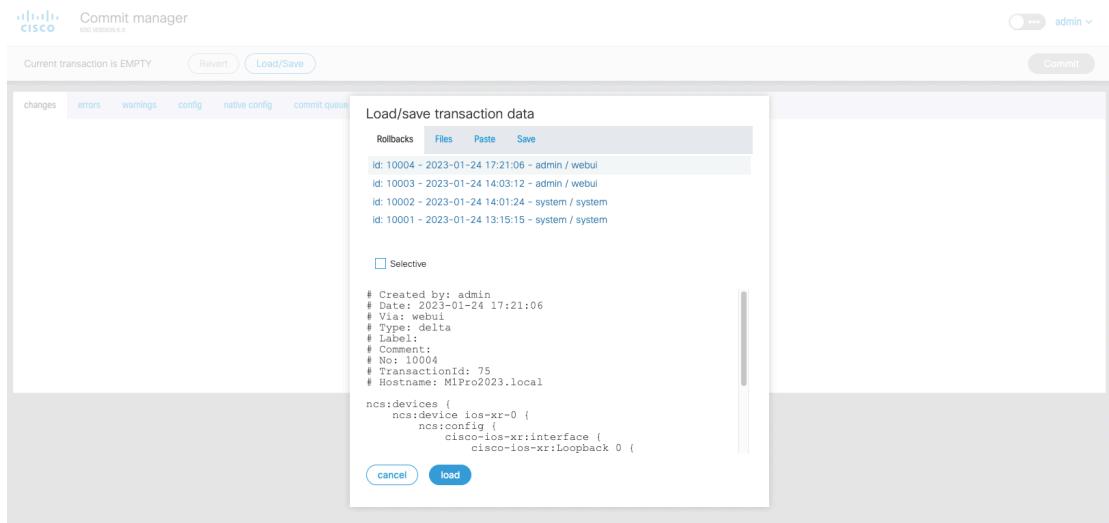


If we go back to CLI, we can see that our device already has the new IP on the Loopback Interface

```
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 webui. We click on load, then config, and we can see that the change will be reverted.

```

1 devices {
2   device ios-xr-0 {
3     config {
4       interface {
5         Loopback 0 {
6           ipv4 {
7             address {
8               ip 1.1.1.2;
9             }
10            )
11          )
12        )
13      )
14    )
15  )
16}
17

```

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

```

1 devices {
2   device ios-xr-0 {
3     config {
4       interface {
5         Loopback 0 {
6           ipv4 {
7             address {
8               ip 1.1.1.2;
9             )
10            )
11          )
12        )
13      )
14    )
15  )
16}
17

```

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.

```

jomira@M1Pro2023 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
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
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
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

We got a red alert because this change was done out 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
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
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
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 we click on Compare-config we can see exactly what was done out 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
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
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
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

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
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
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
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

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.

In this case we want to delete that configuration that was done without NSO so we will “sync-to”

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
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
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
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

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
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
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
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

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-cll-7.43...cisco-iosxr-cll-7.43	0	ping	connect	check-sync	sync-from	sync-to	compare-config		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
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
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 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? Let's do it. Let's create a device Group.

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

The screenshot shows the Cisco Configuration editor interface. At the top, there is a header with the Cisco logo and the title "Configuration editor". Below the header, there is a navigation bar with tabs: "PACKAGES" (selected), "Reload", and "admin". Under the "PACKAGES" tab, a package named "cisco-iosxr-cll-7.43" is listed with version "v7.43". In the main content area, there is a table titled "MODULES" containing various Cisco modules. The columns include: aaa:aaa, nacm:nacm, ncs:high-availability, ncs:services, ncs:zombies, t1:t1s; aaa:alias, ncm:netconf-state, ncs:java-vm, ncs:side-effect-queue, rcmn: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. The "ncs:devices" module is highlighted in the table.

In the Device-Group click on the Plus button

The screenshot shows the "device-group" section of the Configuration editor. There is a table with two rows: "template" and "device-group". The "device-group" row has a sub-table with two columns: "name" and "snmp". A modal dialog box titled "Add new list item" is open, showing fields for "name" (with "ios-xr-devices" typed in) and "confirm". The "device-group" row is selected, indicated by a blue border around its cells.

The screenshot shows the "device-group" section again. The "ios-xr-devices" entry has been added to the list. A modal dialog box titled "Add new list item" is open, showing the "name" field filled with "ios-xr-devices" and a "confirm" button. The "device-group" row is selected.

The screenshot shows the "device-group" section after the addition. The "ios-xr-devices" entry is now listed under the "device-group" row. The "device-group" row is selected.

Click confirm and then click on the device-group that you've created.

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

The screenshot shows two screenshots of the Cisco Configuration Editor interface. The top screenshot shows a device group named 'ios-xr-devices' with an empty 'device-name' list. The bottom screenshot shows the same device group with a modal dialog for adding a new list item. The 'device-name' field contains 'ios-xr-0' and has a dropdown menu showing 'ios-xr-0', 'ios-xr-1', 'ios-xr-2', 'ios-xr-3', and 'Source'. The 'location' field is also visible at the bottom of the screen.

This should be the result.

The screenshot shows the final state of the device group 'ios-xr-devices'. The 'device-name' list now contains four entries: 'ios-xr-0', 'ios-xr-1', 'ios-xr-2', and 'ios-xr-3'. The 'location' field is also visible at the bottom of the screen.

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

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

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 make the configurations that you want to apply. Let's create a DNS configuration and set the VTP to off.

Start with VTP, choose the off.

The screenshot shows the Cisco Configuration editor interface. The URL in the address bar is `/ncs/devices/template[set-dns-and-vtp-off]/ned-id[cisco-iosxr-cl-7.43:cisco-iosxr-cl-7.43]/config/cisco-ios-xr:vtp/`. The main content area displays a configuration block with a single line: `mode off`.

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

The screenshot shows the Cisco Configuration editor interface. The URL in the address bar is `/ncs/devices/template[set-dns-and-vtp-off]/ned-id[cisco-iosxr-cl-7.43:cisco-iosxr-cl-7.43]/config/cisco-ios-xr:domain/`. The main content area displays a configuration block with a 'name' field containing 'cisco.com' and a 'name-server' list containing '2.2.2.2'.

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

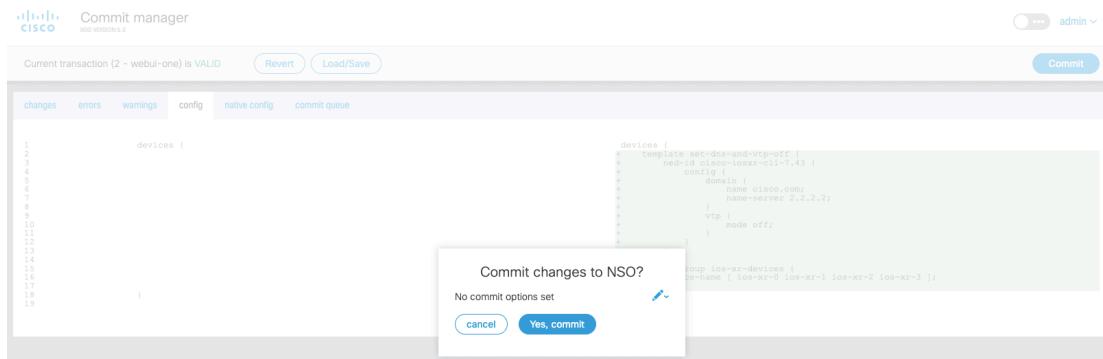
From the config diff we can see that we've created a device-group and the configuration template that we've created.

The screenshot shows the Cisco Commit manager interface. The URL in the address bar is `/commit`. The main content area displays a configuration diff between two transactions. The diff highlights the creation of a 'device-group' named 'ios-xr-devices' with a single member 'ios-xr-0'.

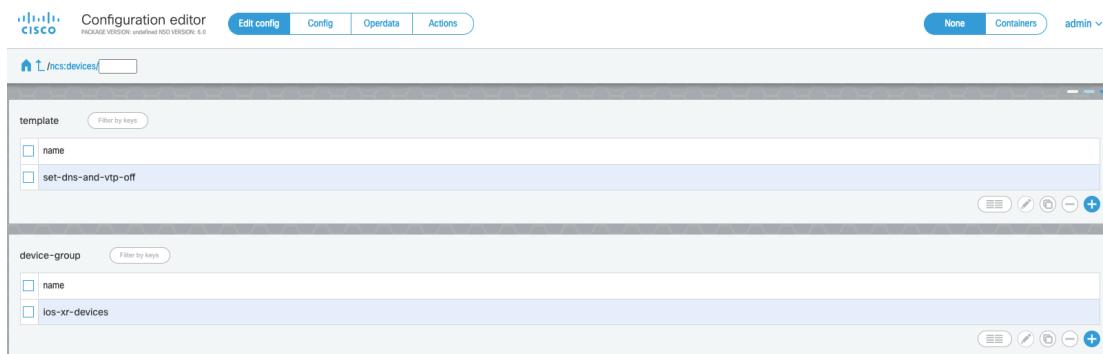
```
1 devices {
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19 }
```

```
1+     devices [
2+       template set-dns-and-vtp-off
3+       ned-id cisco-iosxr-cl-7.43 {
4+         config (
5+           domain (
6+             name cisco.com;
7+             name-server 2.2.2.2;
8+           )
9+           vtp (
10+             mode off;
11+           )
12+         )
13+       }
14+       device-group ios-xr-devices [
15+         device-name [ ios-xr-0 ios-xr-1 ios-xr-2 ios-xr-3 ];
16+       ]
17+     }
18+   }
```

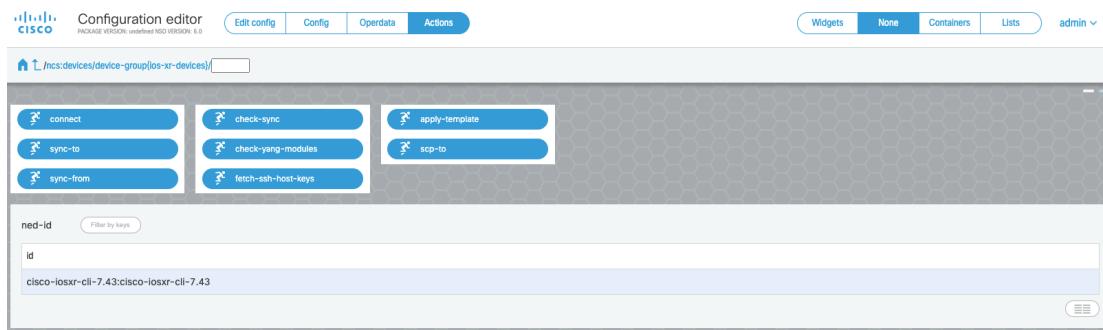
After commit, we can now go to the Device Group and Apply the Template.



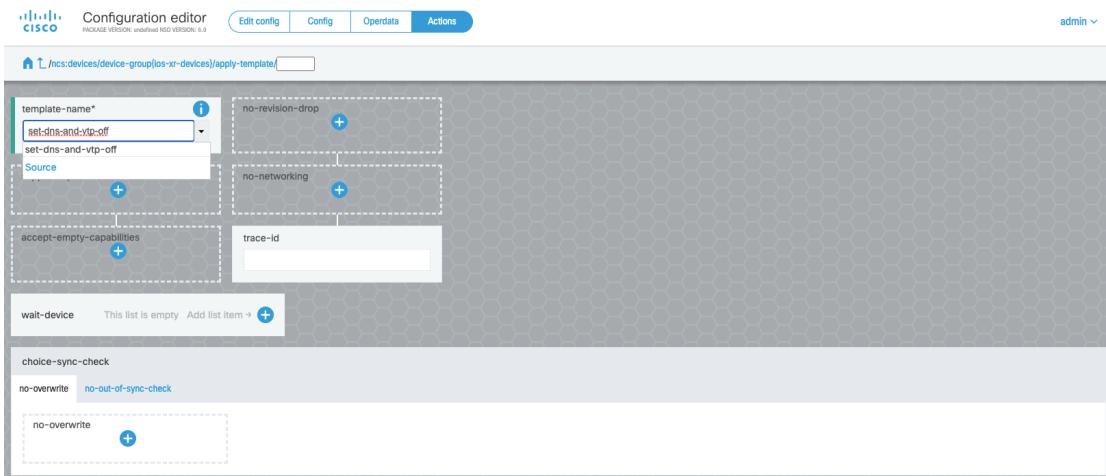
To go to the Device Group we go to the Configuration Editor, “ncs:devices” module and click on the recently created device-group “ios-xr-devices”.



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

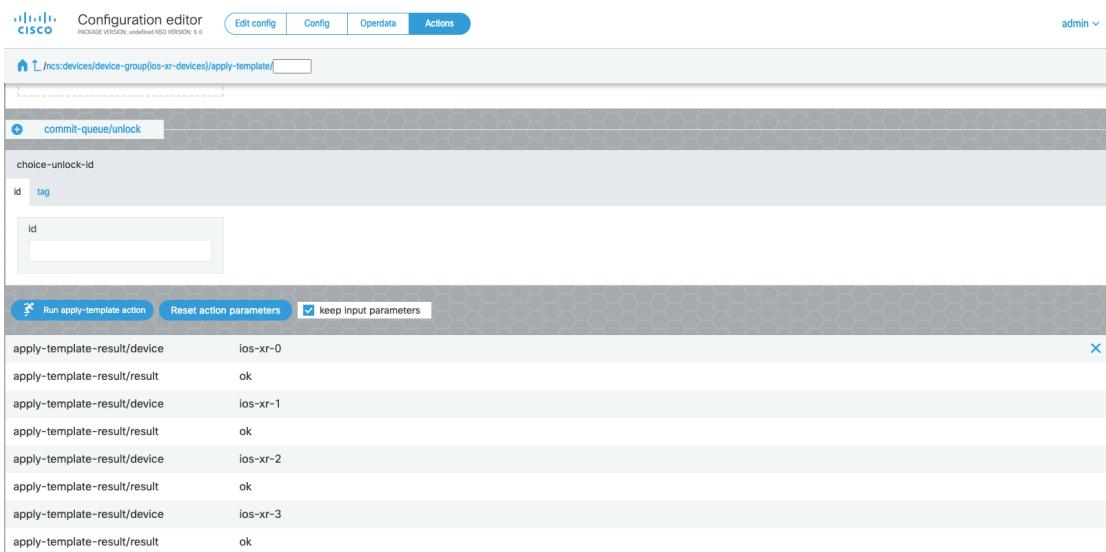


And now, select the “template-name” that we’ve created.



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

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



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

We click on Commit 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 consistency. 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, you 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 can compare-configs and re-deploy if necessary.

First step is to create a service.

NSO already brings a script that creates a service skeleton.

Use the command “ncs-make-package -h”

```
jomira@M1Pro2023 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.
```

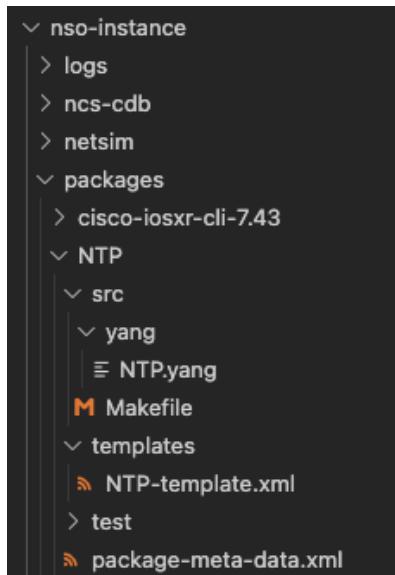
You can see that we've many options here.

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 push that data via external APIs and only send the device configs according to what you receive.

Let's choose the Template option. (Use it inside the packages folder, otherwise you will have to copy the folder there after)

```
jomira@M1Pro2023 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:

```

Ξ NTP.yang ×
nso-instance > packages > NTP > src > yang > Ξ 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 sent in XML.

```

jomira@M1Pro2023 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.

```

↳ NTP-template.xml ×
nso-instance > packages > NTP > templates > ↳ NTP-template.xml > ...
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      <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      <ntp xmlns="http://tail-f.com/ned/cisco-ios-xr">
18        <peer>
19          <address>192.168.22.33</address>
20        </peer>
21        <server>
22          <server-list>
23            <name>172.16.22.44</name>
24            <minpoll>8</minpoll>
25            <maxpoll>12</maxpoll>
26          </server-list>
27        </server>
28      </ntp>
29    </config>
30  </device>
31 </devices>
32 </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 variables.

In this list I see 4 inputs :

Peer IP-address

Server IP-address

Minpool

Maxpool

After defining the variables our XML should look like this

```

» NTP-template.xml ×
nso-instance > packages > NTP > templates > » NTP-template.xml > ⏺ config-template > ⏺ devices > ⏺ device > ⏺ config > ⏺ ntp > ⏺ server
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            <ntp xmlns="http://tail-f.com/ned/cisco-ios-xr">
20              <peer>
21                <address>{/peer-address}</address>
22              </peer>
23              <server>
24                <server-list>
25                  <name>{/server-address}</name>
26                  <minpoll>{/minpool}</minpoll>
27                  <maxpoll>{/maxpool}</maxpoll>
28                </server-list>
29              </server>
30            </ntp>
31          </config>
32        </device>
33      </devices>
34    </config-template>

```

Going to the YANG after setting up the definition for each variable they should look like this.

So we will setup as mandatory the peer-address and server-address and then the minpoll and maxpoll we setup as non-mandatory and with default values.

```

NTP-template.xml      NTPyang
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

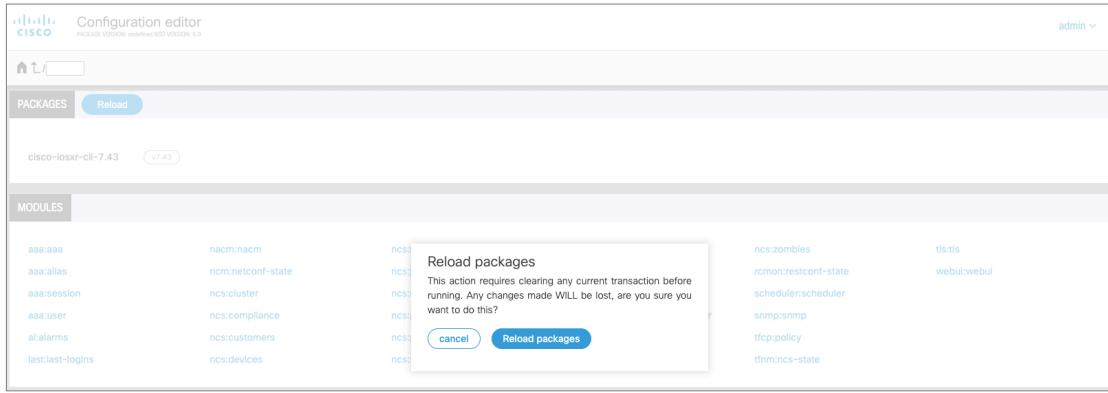
And apply the “make” command.

```

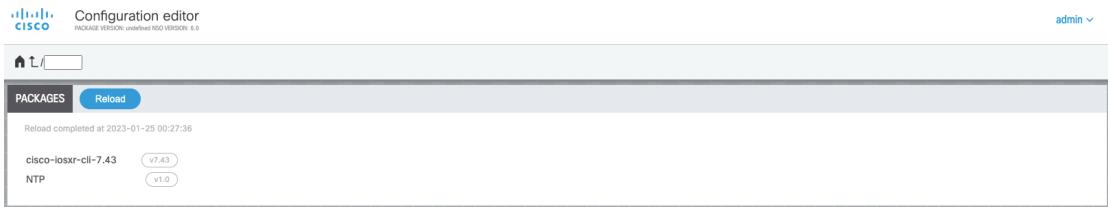
jomira@M1Pro2023 packages % cd NTP
jomira@M1Pro2023 NTP % cd src
jomira@M1Pro2023 src % make
mkdir -p ../load-dir
/Users/jomira/NSO-CLEU/bin/ncsc `ls NTP-ann.yang > /dev/null
2>&1 && echo "-a NTP-ann.yang` \
--fail-on-warnings \
\
-c -o ../load-dir/NTP.fxs yang/NTP.yang

```

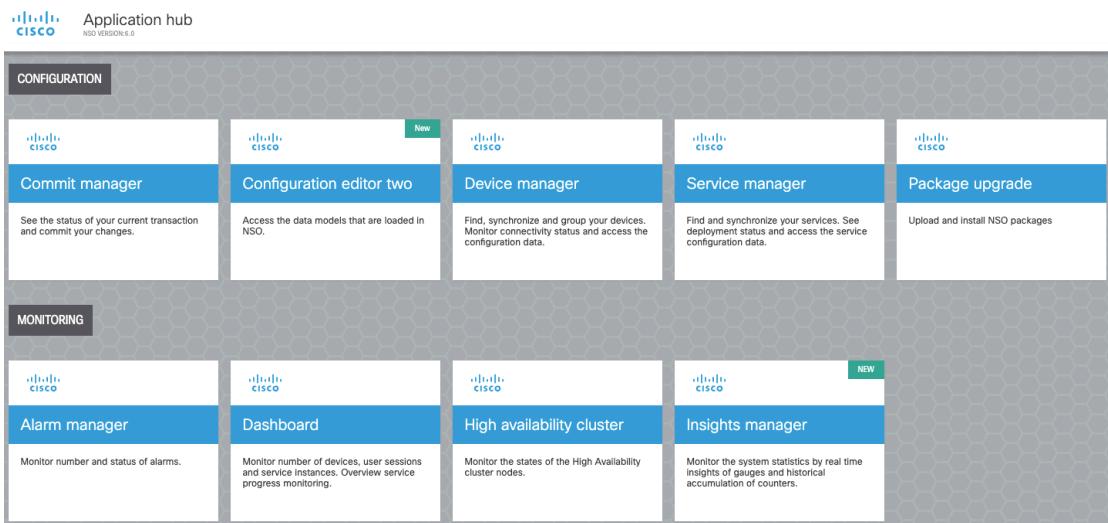
Then, we should go to NSO, and ask for a package reload.



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

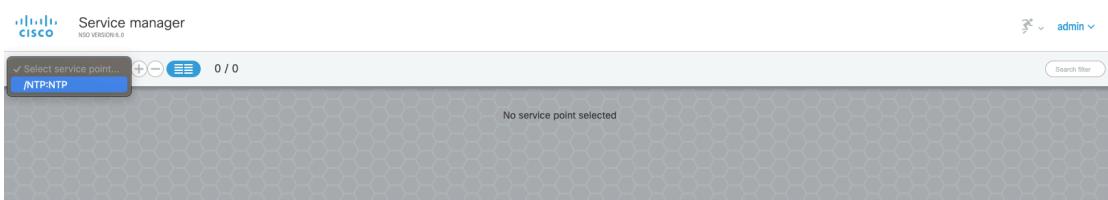


Going back to the Homepage we click on the “Service Manager”

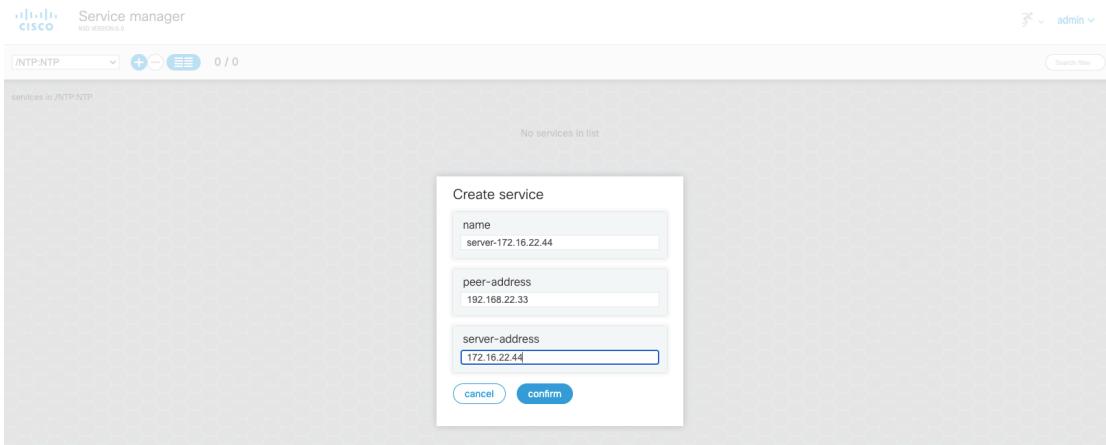


We select the NTP service that we've created

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 shows multiple NTP server definitions across four devices:

- ios-xr-0:** peer 192.168.22.33 server 172.16.22.44 minpoll 8 maxpoll 12
- ios-xr-1:** peer 192.168.22.33 server 172.16.22.44 minpoll 8 maxpoll 12
- ios-xr-2:** peer 192.168.22.33 server 172.16.22.44 minpoll 8 maxpoll 12
- ios-xr-3:** peer 192.168.22.33 server 172.16.22.44 minpoll 8 maxpoll 12

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

```

After commit, we can now see if the service is in Sync

The configuration transaction has been committed, and the service is now in sync.

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

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      devices {
2        device ios-xr-2 {
3          config {
4            = peer 192.168.22.33 {
5              server {
6                server-list 172.16.22.44 {
7                  minpoll 8;
8                  maxpoll 12;
9                }
10              }
11            }
12          }
13        }
14      }
15    }
16  }
17  device ios-xr-3 {
18    config {
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
31

```

```

1 devices {
2   device los-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 los-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}
31

```

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

services in /NTP:NTP				
name	devices	check-sync	re-deploy	re-deploy dry-run
server-172.16.22.44	4	check-sync	re-deploy	re-deploy dry-run

services in /NTP:NTP				
name	devices	check-sync	re-deploy	re-deploy dry-run
server-172.16.22.44	4		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

services in /NTP:NTP				
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 }
31

```

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

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.

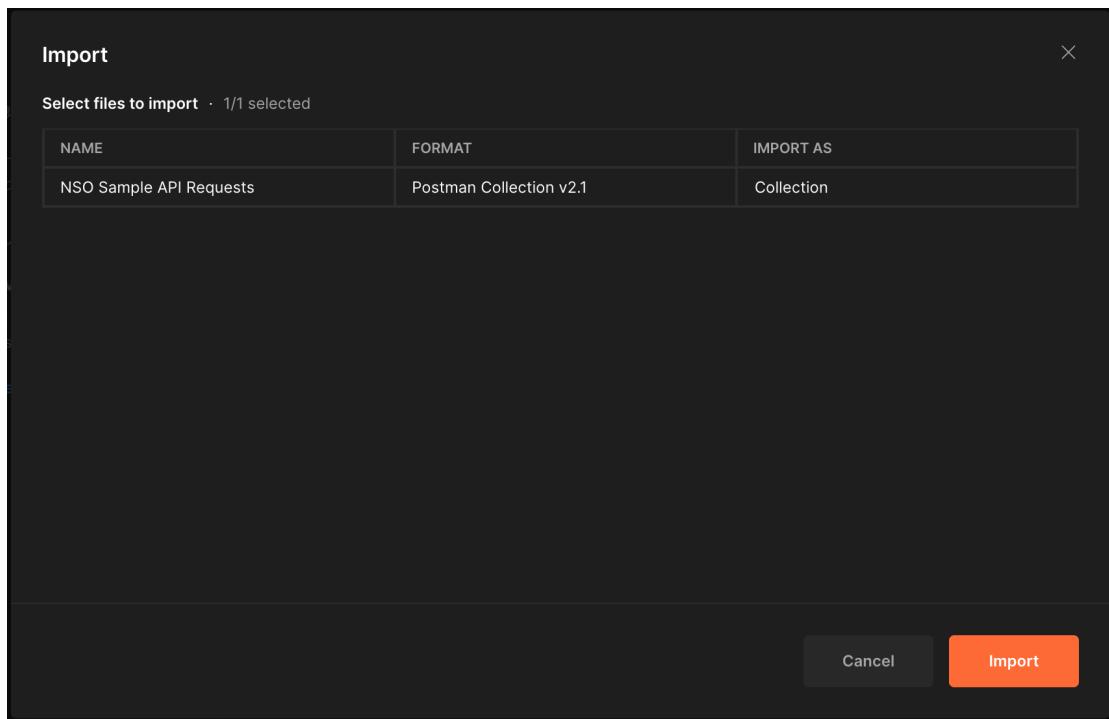
You can download it here.

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

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

The screenshot shows the Cisco DevNet Public API interface. On the left, there's a sidebar with various categories like Collections, APIs, Environments, Mock Servers, Monitors, Flows, and History. Under 'Collections', 'NSO Sample API Requests' is listed. A context menu is open over this collection, with 'Export' highlighted. The main area shows the 'NSO Sample API Requests' collection details, including sections for Authorization (set to Basic Auth), Tests, and Variables. A warning message about sensitive data is visible. To the right, there's a 'Documentation' panel with some sample code and a 'NEXT IN THIS COLLECTION' link.

The screenshot shows the Postman interface with the 'Import' dialog box open. The dialog has tabs for File, Folder, Link, Raw text, Code repository, and a 'New' button. It also includes tabs for OpenAPI, GraphQL, cURL, WSDL, and HAR. Below the tabs is an 'Upload Files' button and a note about supported import formats. The background shows the 'Scratch Pad' section of Postman.



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

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 NSO Scratch Pad interface. On the left, there's a sidebar with 'Collections' containing 'NSO Sample API Requests' (with 'Always-On Read Only Requests' expanded), 'APIs', 'Environments', 'Mock Servers', 'Monitors', and 'History'. The main area shows a 'List Device Groups' request under 'NSO Root of the App Hierarc...'. The 'Headers' tab is selected, showing three checked headers: Content-Type (application/yang-data+json), Authorization (Basic YWRtaW46YWRtaW4=), and Accept (application/yang-data+json). Below the headers, the 'Body' tab is selected, displaying a JSON response structure. The response body is as follows:

```

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-cli-7.43:cisco-iosxr-cli-7.43"
19       }
20     ]
21   }
22 ]

```

At the bottom right, it says 'Status: 200 OK Time: 45 ms Size: 1.04 KB Save Response'.

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/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
"AAAAB3NzaC1yc2EAAAQABAAQDAW+oFswFE439ByAMZ++iMLcnhVsI+L
ui5mn/Wfiie\npC3czutSf1fdIBOIC4sCpqYrxoqowXHhU2T9wnx5sAwzPJuM
jbbgUVnUs1CV8MeF/jWOUL4\n2Rr6IbECuV7qvMFfZmcrTR/BqXUdbqBtb5f0T
zHdvT2gBP7AgLPptzVa5/BDkNV70UG60cxc\nIsqWN15471U0Dol7rSyufKz8y
1sa4E0Dayn+9+O8uK6iq7GzPqyf9fkrQjhmlNc5sozuAPQ4\nFMmjIAAMbQOzz
5LvjktUw2QFgFm53Dzp83LxG6nVWBNS4JlgniLEE4rG06xPDh6IAy5MKiBS\nt
N4yBYF++0WL2tKszrCo//jZsDbv2ieTFROTpmp2ldwgt7YUyep8j6JWVqlRPDV
IjgXeGI6C\nG8xa81VW5uElmwGDSSjVFZVYXJQVKogDNw19vREJHtSRAqjGknN
648zMtdR+BI1c9fLzdxS\nb1o2jJtU/P5WLLFmJDbw0+A+d39iNuNr9H5PRj8
="
/restconf/data/tailf-ncs:devices/device=ios-xr-0/ssh/host-
key=ssh-ed25519/key-data
AAAAC3NzaC1lZDI1NTE5AAAAIGvE1TpobfiHd3r/Z+1Sxr8bARNNyZIGQt0DMg
vpNjmj
/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
"AAAAB3NzaC1yc2EAAAQABAAQDAW+oFswFE439ByAMZ++iMLcnhVsI+L
ui5mn/Wfiie\npC3czutSflfdIBOIC4sCpqYrXoqowXHhU2T9wnx5sAWwzPJuM
jbbgUVnUs1CV8MeF/jWOUL4\n2Rr6IbECuV7qvMFZmcrTR/BqXUdbqBtb5f0T
zHdvT2gBP7AgLPptzVa5/BDkNV70UG60cxc\nIsqWN15471U0Dol7rSyufKz8y
1sa4E0Dayn+9+08uK6iq7GzPqyf9fkrQjhmlNc5sozuAPQ4\nFMmjIAAMbQOzz
5LvjktUw2QFgFm53Dzp83LxG6nVWBNS4JlgnileE4rG06xPDh6IAy5MKiBS\nt
N4yBYF++0WL2tKszrCo//jzsDbv2ieTFROTPmp2ldwgt7YUyep8j6JWVqLRPDV
IjgXeGI6C\nG8xa81VW5uElmwGDSSjVFZVYXJQVKogDNw19vREJHtSRAqjGknN
648zMtdR+BI1c9fLzdxS\Nb1o2jJtU/P5WLLFmJDbw0+A+d39iNuNr9H5PRj8
="
/restconf/data/tailf-ncs:devices/device=ios-xr-0/ssh/host-
key=ssh-ed25519/key-data
AAAAC3NzaC1lZDI1NTE5AAAAIGvE1TpobfiHd3r/Z+1Sxr8bARNNyzIGQt0DMg
vpNjmj
/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
"AAAAB3NzaC1yc2EAAAQABAAQDAW+oFswFE439ByAMZ++iMLcnhVsI+L
ui5mn/Wfiie\npC3czutSflfdIBOIC4sCpqYrXoqowXHhU2T9wnx5sAWwzPJuM
jbbgUVnUs1CV8MeF/jWOUL4\n2Rr6IbECuV7qvMFZmcrTR/BqXUdbqBtb5f0T
zHdvT2gBP7AgLPptzVa5/BDkNV70UG60cxc\nIsqWN15471U0Dol7rSyufKz8y
1sa4E0Dayn+9+08uK6iq7GzPqyf9fkrQjhmlNc5sozuAPQ4\nFMmjIAAMbQOzz
5LvjktUw2QFgFm53Dzp83LxG6nVWBNS4JlgnileE4rG06xPDh6IAy5MKiBS\nt
N4yBYF++0WL2tKszrCo//jzsDbv2ieTFROTPmp2ldwgt7YUyep8j6JWVqLRPDV
IjgXeGI6C\nG8xa81VW5uElmwGDSSjVFZVYXJQVKogDNw19vREJHtSRAqjGknN
648zMtdR+BI1c9fLzdxS\Nb1o2jJtU/P5WLLFmJDbw0+A+d39iNuNr9H5PRj8
="
/restconf/data/tailf-ncs:devices/device=ios-xr-0/ssh/host-
key=ssh-ed25519/key-data
AAAAC3NzaC1lZDI1NTE5AAAAIGvE1TpobfiHd3r/Z+1Sxr8bARNNyzIGQt0DMg
vpNjmj

```

```

/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 Device Manager Configuration editor interface. At the top, there's a navigation bar with tabs: Configuration editor, Edit config, Config, Operdata, Actions, and Widgets. The Widgets tab is currently active, showing a dropdown menu with options: None, Containers, Lists, and admin. The 'admin' option is selected. Below the navigation bar, the URL is /ncs/devices/device[ios-xr-0]/. In the main content area, there's a table with columns: name, address, port, authgroup, and a show default values checkbox. The table shows the following data:

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

Below the table, there are several buttons labeled: ssh, device-type, ssh-algorithms, state, and config. The 'ssh' button is currently highlighted. On the far right, there's a sidebar titled 'Widgets' with a list of options: JSON, KeyPath, Maagic, NETCONF, RESTCONF, Set, XML, and XPath. The 'RESTCONF' option is also highlighted.

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 Cisco Configuration Editor interface with the RESTCONF tab selected. The URL in the address bar is `/ncs/devices/device[ios-xr-0]`. The main content area displays a large block of configuration XML code for a Cisco IOS-XR device, specifically version 7.43. The XML includes details like host keys, device types, and network interfaces.

```

<restconf/data/tailf-ncs:devices/device=ios-xr-0/address 127.0.0.1
</restconf/data/tailf-ncs:devices/device=ios-xr-0/port 1022
</restconf/data/tailf-ncs:devices/device=ios-xr-0/host-key=ssh-rsa/key-data "AAAAAB3NzaC1yc2EAAADQABAAA8qODAw+oFwFB439ByANz++iMLcnhVsI+lu5mn/Wfile\nc3czutS1fdIBOIC4s
</restconf/data/tailf-ncs:devices/device=ios-xr-0/host-key=ssh-ed25519/key-data AAAAC3nzaC1ZD1NTESAAAGvElTpobfId3z/Z+1Sxr0bARNy+z1GQt0DMyvpNjmj
</restconf/data/tailf-ncs:devices/device=ios-xr-0/ssh/authorhost/default
</restconf/data/tailf-ncs:devices/device=ios-xr-0/device-type/cli<1>ios-xr<1>cisco-iosxr-cl<1>7.43
</restconf/data/tailf-ncs:devices/device=ios-xr-0/ssh/algorithm/pubkey-type ssh-ed25519 ssh-rsa ]
</restconf/data/tailf-ncs:devices/device=ios-xr-0/ssh/ssh-state unlocked
</restconf/data/tailf-ncs:devices/device=ios-xr-0/config/tailf-ned:cisco-ios-xr:domain/name cisco.com
<1>restconf/data/tailf-ncs:devices/device=ios-xr-0/config/tailf-ned:cisco-ios-xr:domain/name-server=2.2.2.2
<1>restconf/data/tailf-ncs:devices/device=ios-xr-0/config/tailf-ned:cisco-ios-xr:ntp/server-list=172.16.22.44:minpoll 12
<1>restconf/data/tailf-ncs:devices/device=ios-xr-0/config/tailf-ned:cisco-ios-xr:vtp/mode off
<1>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
<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

```

Another option to check the paths, is a package that was built for NSO (don't use in PROD env), named "rest-api-explorer"

<https://gitlab.com/nso-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 with the URL `/restconf/data/devices`. The left sidebar shows a tree view of the RESTCONF data model, including sections like /api, /ncs:devices, and /ncs:global-settings. The right panel displays the `/restconf/data/devices` endpoint with its various HTTP methods: GET, HEAD, OPTIONS, PATCH, POST, and PUT. Each method is described with its purpose: GET retrieves data and meta-data, HEAD retrieves headers, OPTIONS discovers methods, PATCH updates an existing resource, POST creates a new resource, and PUT creates or replaces the target resource.

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 aaa:aaa module

The screenshot shows the Cisco Configuration Editor interface with the RESTCONF tab selected. In the top navigation bar, it says "Configuration editor" and "admin". The main content area has tabs for "PACKAGES" and "Reload". Under "PACKAGES", there is a list of packages: "cisco-iosxr-cl-7.43 v7.43" and "NTP v1.0". Under "MODULES", there is a list of modules, each with a brief description:

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

Then we go to the authentication/users

name	uid	gid	password	ssh_keydir	homedir
admin	65534	65534	\$6\$HFJLfaShdkSx8QQ\$...pxYVNjPY6QOZKY.D8.30	/var/ncs/homes/admin/.ssh	/var/ncs/homes/admin
oper	65534	65534	\$6\$23VdVhL6zXVJjkUS...avJlOPHsf4nfIZSH7w/	/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\$HFJLfaShdkSx8QQ\$...pxYVNjPY6QOZKY.D8.30	/var/ncs/homes/admin/.ssh	/var/ncs/homes/admin
oper	65534	65534	\$6\$23VdVhL6zXVJjkUS...avJlOPHsf4nfIZSH7w/	/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
<input type="checkbox"/> read_user					

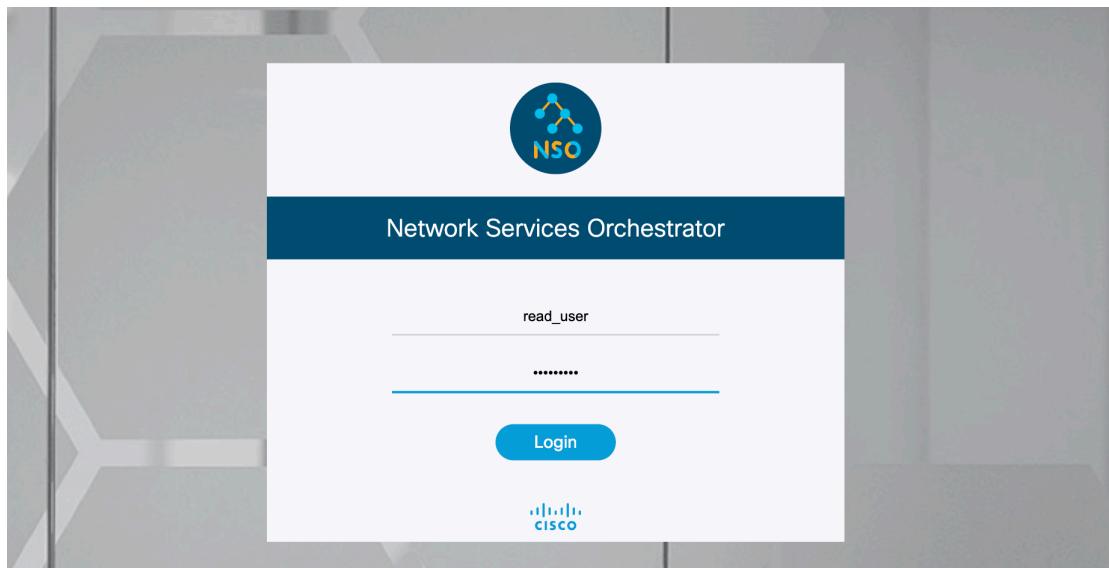
Let's create the user “read_user”.

Add new list item

name	read_user
uid	1
gid	1
password	*****
ssh_keydir	/var/ncs/homes/public/.ssh
homedir	/var/ncs/homes/public

After this we can “Commit”.

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.

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)

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 Cisco Configuration Editor interface. At the top, there's a header with the Cisco logo and the text "Configuration editor PACKAGE VERSION: unverified NSO VERSION: 6.0". On the right, there's a "read_user" dropdown. Below the header, there's a search bar and a "Reload" button. The main area has two tabs: "PACKAGES" and "MODULES". Under "MODULES", the "ncs:devices" module is selected, highlighted with a blue border. Other modules listed include aaa:aaa, aaa:alias, aaa:session, aaa:user, al:alarms, last:last-logins, ncm:netconf-state, ncs:cluster, ncs:compliance, ncs:customers, ncs:high-availability, ncs:java-vm, ncs:metric, ncs:packages, ncs:python-vm, ncs:service-progress-monitoring, ncs:services, ncs:software, ncs:ssh, ncs:zombies, NTP:NTP, ncmon:restconf-state, scheduler:scheduler, ncs:side-effect-queue, ncs:smart-license, ncs:snmp-notification-receiver, ncp:snmp, and tcp:policy, tftp:ncs-state, webui:webui.

Then we select authgroup “default”

The screenshot shows the Cisco Configuration Editor interface. At the top, there's a header with the Cisco logo and the text "Configuration editor PACKAGE VERSION: unverified NSO VERSION: 6.0". On the right, there's a "read_user" dropdown. Below the header, there's a search bar and a "Reload" button. The main area has four tabs: "Edit config", "Config", "Operdata", and "Actions". Under "Edit config", there's a "None" button and a "Containers" button. The URL in the address bar is "/ncs:devices/authgroups/". The page displays a list of authgroups. The first group, "group", has two entries: "name" and "default". The second group, "snmp-group", also has "name" and "default". Both groups have standard edit, delete, and add icons at the bottom.

At this moment we can make these operations on the “read_user” since we still don't have not made any change to permissions.

The screenshot shows the Cisco Configuration Editor interface. At the top, there's a header with the Cisco logo and the text "Configuration editor PACKAGE VERSION: unverified NSO VERSION: 6.0". On the right, there's a "read_user" dropdown. Below the header, there's a search bar and a "Reload" button. The main area has four tabs: "Edit config", "Config", "Operdata", and "Actions". Under "Edit config", there's a "None" button and a "Containers" button. The URL in the address bar is "/ncs:devices/authgroups/group/default/". The page displays the details for the "default" authgroup. It shows a "name" field containing "default" and a "umap" section. The "umap" section contains three entries: "local-user", "admin", and "oper". At the bottom, there's a "+ default-map" button and standard edit, delete, and add icons.

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` here

The screenshot shows the Cisco Configuration editor interface. The URL in the address bar is `/ncs/devices/authgroups/group[default]/umap[read_user]`. The top navigation bar includes tabs for `Edit config`, `Config`, `Operdata`, and `Actions`. On the right, there are buttons for `None`, `Containers`, and a dropdown set to `read_user`. A modal window titled "Add new list item" is open, showing a list of items: `local-user` and `read_user`. The `read_user` entry is selected. At the bottom of the modal are "cancel" and "confirm" buttons.

Now, you can put the remote authentication for the devices that will belong have this authgroup.

The screenshot shows the Cisco Configuration editor interface. The URL in the address bar is `/ncs/devices/authgroups/group[default]/umap[read_user]`. The top navigation bar includes tabs for `Edit config`, `Config`, `Operdata`, and `Actions`. On the right, there are buttons for `None`, `Containers`, and a dropdown set to `read_user`. The main pane displays a configuration for a local-user map named `read_user`. It includes sections for `login-credentials` (set to `callback`) and `remote-user` (set to `same-user`). Under `remote-user`, there is a sub-section for `remote-name` with the value `admin`. Below this, under `remote-auth`, there are sections for `same-pass` (disabled), `remote-password` (selected, with a password field containing "*****"), and `public-key` (disabled). The bottom navigation bar includes icons for Commit manager (C*), Configuration editor two (E2), Alarm manager (A), Dashboard (B), Device manager (D), Service manager (S), Package upgrade (P), and Insights manager (I).

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).

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 Cisco Device manager interface. The URL in the address bar is `/ncs/devices`. The top navigation bar includes tabs for `Device manager`, `Alarm manager`, `Dashboard`, `Device manager`, `Service manager`, `Package upgrade`, and `Insights manager`. On the right, there are buttons for `None`, `Containers`, and a dropdown set to `read_user`. The main pane displays a table of devices. The columns include: name, address, port, type, services, ping, connect, check-sync, sync-from, sync-to, compare-config, alarm, and configuration. There are four entries in the table:

	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	1 ▾	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	1 ▾	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	1 ▾	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	1 ▾	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration

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

The screenshot shows the Cisco Configuration editor interface. At the top, it displays 'Configuration editor' and 'admin'. Below the header, there are sections for 'PACKAGES' and 'MODULES'. In the 'MODULES' section, the 'nacm:nacm' module is selected, highlighted with a blue background. Other modules listed include aaa:aaa, aaa:alias, aaa:session, aaa:user, ai:alarms, last:last-logins, and several nc* modules like 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:ssh, ncs:snmp-notification-receiver, ncs:software, ncs:ssh, ncs:zombies, NTP:NTP, ncs:snmp, tfcp:policy, tfnm:ncs-state, tis:tis, webui:webui, and scheduler:scheduler.

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.

The screenshot shows the 'rule-list' section of the nacm:nacm configuration. It displays several policy entries:

enable-nacm	exec-default	cmd-exec-default
(true)	(permit)	(permit)
read-default	enable-external-groups	log-if-default-permit
(permit)	(true)	+ (button)
write-default	cmd-read-default	enforce-nacm-on-services
permit	(permit)	(false)

Below the rule-list, there is a 'groups' section containing three entries: 'name', 'admin', and 'any-group'. At the bottom of the screen, a navigation bar includes icons for Commit manager, Configuration editor two, Alarm manager, Dashboard, Device manager, Service manager, Package upgrade, and Insights manager.

Let's create the “read_group”

The screenshot shows the Cisco Configuration editor interface. In the top navigation bar, there are tabs for 'Edit config', 'Config', 'Operdata', and 'Actions'. On the right, there are buttons for 'None', 'Containers', and 'admin'. Below the navigation, the URL is /nacm:nacm/groups/. A modal window titled 'Add new list item' is open, showing a single input field labeled 'name' with the value 'read_group'. There are 'cancel' and 'confirm' buttons at the bottom of the modal.

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

The screenshot shows the Configuration editor with the URL /nacm:nacm/groups/group{read_group}/. The 'read_group' entry is selected. A modal window titled 'Add new list item' is open, showing an input field labeled 'user-name' with the value 'read_user'. There are 'cancel' and 'confirm' buttons at the bottom of the modal.

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

The screenshot shows the Configuration editor with the URL /nacm:nacm/. A modal window titled 'Add new list item' is open, showing an input field labeled 'name' with the value 'read_rule'. There are 'cancel' and 'confirm' buttons at the bottom of the modal. The background shows various configuration options like 'enable-nacm', 'exec-default', and 'cmd-exec-default'.

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.

The screenshot shows the Cisco Configuration editor interface for the NACM rule list. The top navigation bar includes tabs for 'Edit config', 'Config', 'Operdata', and 'Actions'. The main area displays a rule named 'read_rule' with a group assignment to 'read_group'. The 'rule' and 'tacm:cmdrule' sections are currently empty.

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

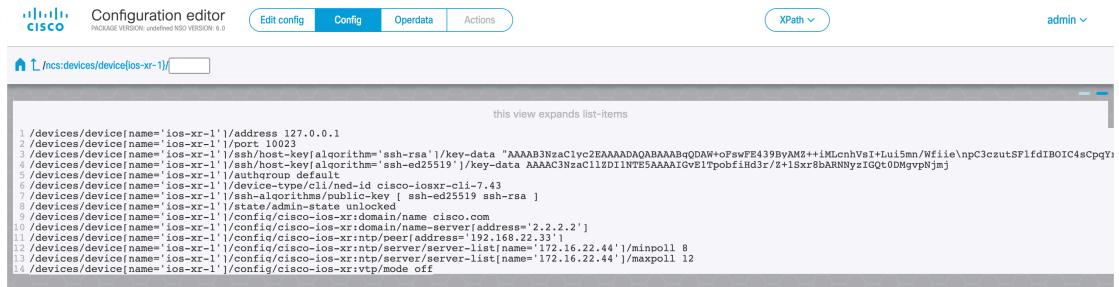
Let's add a rule to deny the sync-from

The screenshot shows the addition of a new list item for the 'read_rule' group. A modal dialog box titled 'Add new list item' is open, showing a 'name' field containing 'deny-sync-from' and an 'action' dropdown set to 'deny'. The main configuration area shows the 'read_group' assigned to the 'read_rule'.

In the rule we've to specify a path

The screenshot shows the detailed configuration of the 'deny-sync-from' rule. The 'path' field is set to '/devices/device-sync-from'. Other fields include 'action' (denied), 'module-name' (empty), 'comment' (empty), 'access-operations' (empty), 'context' (empty), and 'rule-type' (data-node). A 'log-if-permit' option is also present.

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.



The screenshot shows the Configuration editor interface with the XPath view selected. The URL is `/ncs:devices/device[device[name='ios-xr-1']]`. The content area displays a large block of XML configuration code for the device, including details like host keys, user authentication, and network interfaces.

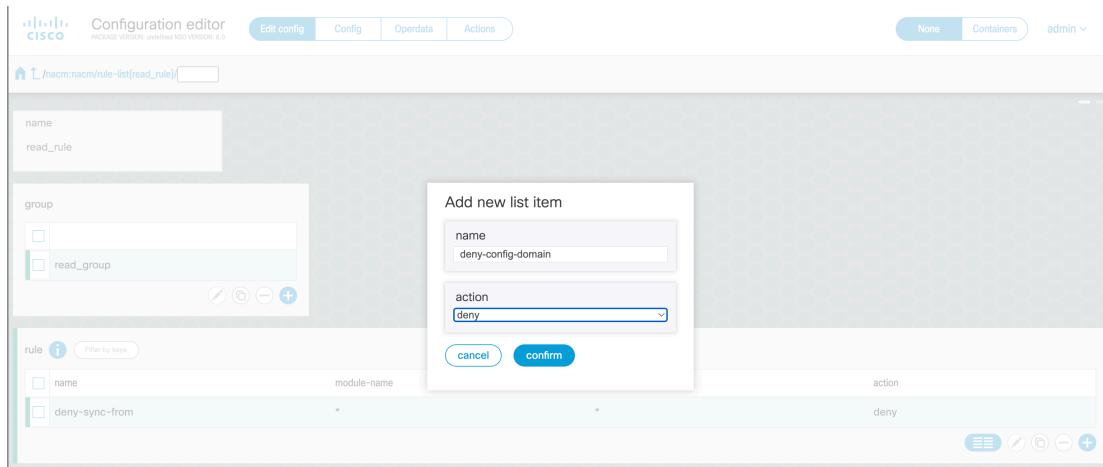
```

<1 /devices/device[name='ios-xr-1']/address 127.0.0.1
<2 /devices/device[name='ios-xr-1']/port 1000
<3 /devices/device[name='ios-xr-1']/ssh/host-key/algorithm='ssh-rsa']/key-data "AAAAAB3NzaC1yc2EAAAQABAAQbQDAW+oFswFE439ByAmZ2+iMlcnhVsI+Lui5mn/Wfie\npC3czutSP1fdIBOIC4sCpqY
<4 /devices/device[name='ios-xr-1']/ssh/host-key/algorithm='ssh-ed25519']/key-data AAAAC3NzaC1lZDI1NTE5AAAAIGv81Tpobf1Hd3r/Z+1Sxr8bARNNyZIGQt0DMgvpNjmj
<5 /devices/device[name='ios-xr-1']/authzgroups/default
<6 /devices/device[name='ios-xr-1']/aaa/authentication/med-id cisco-iosxr-cl1-7.43
<7 /devices/device[name='ios-xr-1']/ssh/algorithms/public-key [ ssh-ed25519 ssh-rsa ]
<8 /devices/device[name='ios-xr-1']/state/admin-state unlocked
<9 /devices/device[name='ios-xr-1']/config/cisco-logs-peer-list[peer[address='2.2.2.2']]
<10 /devices/device[name='ios-xr-1']/config/cisco-iosxr:domain[name-server-address='2.2.2.2']
<11 /devices/device[name='ios-xr-1']/config/cisco-ios-xr:ntp/peer[address='192.168.22.33']
<12 /devices/device[name='ios-xr-1']/config/cisco-ios-xr:ntp/server/server-list[name='172.16.22.44']/minpoll 8
<13 /devices/device[name='ios-xr-1']/config/cisco-ios-xr:ntp/server/server-list[name='172.16.22.44']/maxpoll 12
<14 /devices/device[name='ios-xr-1']/config/cisco-ios-xr:vtp mode off

```

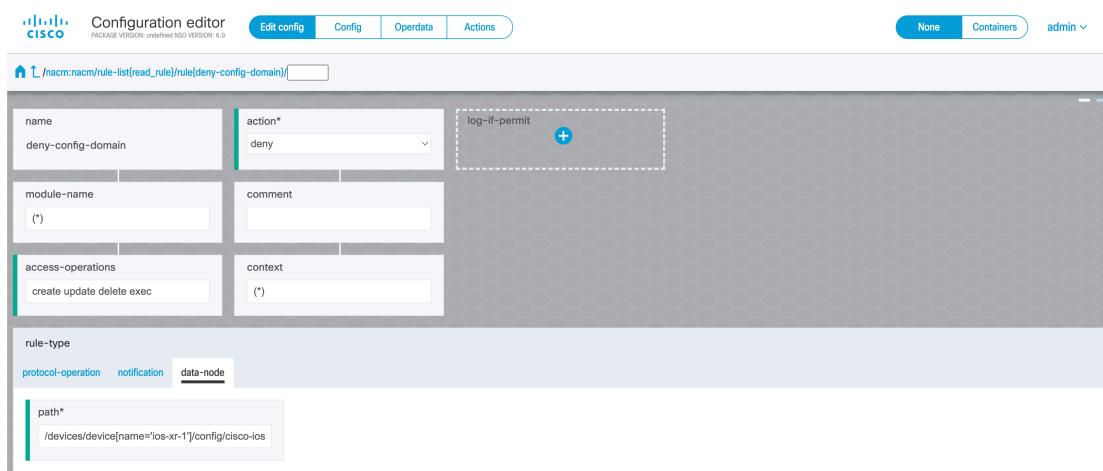
For example, we can grab the path “/devices/device[name='ios-xr-1']/config/cisco-ios-xr:domain” and restrict all the operations with the exception of reading.

We create the rule “deny-config-domain”



The screenshot shows the Configuration editor interface with the NACM rule list view selected. A new rule named `read_rule` is being edited. A modal window titled “Add new list item” is open, showing a single entry: `name: deny-config-domain, action: deny`. The main interface shows other rules like `deny-sync-from`.

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”



The screenshot shows the Configuration editor interface with the NACM rule list view selected. The rule `deny-config-domain` is being edited. The configuration includes the following fields:

- name:** deny-config-domain
- action:** deny
- module-name:** (*)
- comment:** (*)
- access-operations:** create update delete exec
- context:** (*)
- rule-type:** data-node
- path:** /devices/device[name='ios-xr-1']/config/cisco-ios

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.

Name: access-operations
Path: /nacm:nacm/rule-list(read_rule)/rule(deny-config-domain)/access-operations
Access: create, read, update, delete
Kind: leaf
Default Value: *
Union: Yes
Union (1) Type: string, a string
Union (1) Pattern: ^*
Union (2) Bits: create, read, update, delete, exec
Union (2) Size: 32

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” 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-cli-7.43...cisco-iosxr-cli-7.43	1	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	1	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	1	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	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

name: cisco.com

list: This list is empty Add list item → +

name-server: Filter by keys

- address: 2.2.2.2

vrf: This list is empty Add list item → +

ipv4

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

The screenshot shows the Cisco Configuration editor interface. At the top, there's a navigation bar with tabs for 'Edit config', 'Config', 'Operdata', and 'Actions'. Below the navigation bar, the URL is /ncs/devices/device[ios-xr-1]/config/cisco-ios-xr:domain/. The main content area contains several configuration sections:

- name-server**: A list with two items: 'address' and '2.2.2.2'. The 'address' item has a checked checkbox. There are edit and delete icons next to each item.
- vrf**: A section labeled 'This list is empty' with an 'Add list item' button.
- list**: A section labeled 'This list is empty' with an 'Add list item' button.
- ipv4**: A section labeled 'This list is empty' with an 'Add list item' button.

A yellow warning message at the bottom states: "Error: Method failed. data: reason: access denied. type: rpc.method.failed. internal: jsonrpc_data375. code: -32000".

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 Cisco Configuration editor interface for device 'ios-xr-0'. The URL is /ncs/devices/device[ios-xr-0]/config/cisco-ios-xr:domain/. The main content area contains several configuration sections:

- name-server**: A list with two items: 'address' and '2.2.2.2'. Both items have unchecked checkboxes. There are edit and delete icons next to each item.
- vrf**: A section labeled 'This list is empty' with an 'Add list item' button.
- list**: A section labeled 'This list is empty' with an 'Add list item' button.
- ipv4**: A section labeled 'This list is empty' with an 'Add list item' button.

To correct this, 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. The rule is named "deny-config-domain". The "action*" dropdown is set to "deny". A "log-if-permit" checkbox is checked. The "rule-type" section shows "data-node" is selected. The "path*" field contains the value "/devices/device/config/cisco-ios-xr:domain".

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 of all the managed devices with this user.

This screenshot shows a list of name-servers. Two items are selected: "address" and "2.2.2.2". Below this is a "vrf" section which is currently empty.

This screenshot shows a configuration page for the device "ios-xr-2". A warning message at the top states: "Error: Method failed. data: reason: access denied. type: rpc.method.failed. internal: jsonrpc_data375. code: -32000". Below this is an "ipv4" section which is currently empty.