

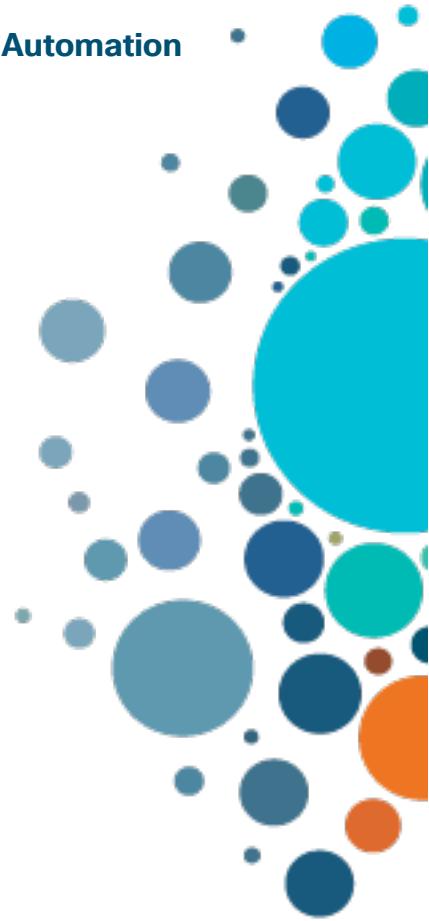
WORKBOOK

The World of NSO WEB UI

DEVLIT-1259

Speakers:

Jorge Mira, Customer Success Specialist - Cross-Domain Automation



Contents

1 - Cisco Network Services Orchestrator	3
1.1 Install NSO and NEDs	3
1.2 Use Netsim's	7
1.3 Configure Devices using NSO	9
1.4 Use Rollback's	12
1.5 Use NSO capabilities to detect Out-of-Band device configurations	13
1.6 Create Device Groups and Device Templates	15
1.7 Create a Simple NSO Service	22
1.8 NSO APIs	35
1.9 Create Role Based and Resource Based Access Control rules	37

1 - Cisco Network Services Orchestrator

1.1 Install NSO and NEDs

Requirements

NSO can be installed on a MAC, or Linux. The additional requirements can be found here:

<https://developer.cisco.com/docs/nso/guides/#!nso-5-7-nso-installation-guide-introduction/prerequisites>

Download NSO for your Operating System and the NEDs

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

```
user@COMPUTER # % ls  
ncs-5.7-cisco-iosxr-7.38.3.signed.bin nso-  
5.7.1.darwin.x86_64.signed.bin
```

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

```
user@COMPUTER # % sh nso-5.7.1.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-  
5.7.1.darwin.x86_64.installer.bin using tailf.cer
```

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

Execute the “nso-5.7.1.darwin.x86_64.installer.bin” with the “–local-install” flag and choose the installation directory

```
user@COMPUTER # % sh nso-5.7.1.darwin.x86_64.installer.bin --local-  
install ~/NSO-5.7.1
```

```
INFO Using temporary directory
/var/folders/gt/460_vz515px80s87tx9gsz3h0000gn/T//ncs_installer.3126
6 to stage NCS installation bundle
INFO Unpacked ncs-5.7.1 in /Users/user/NSO-5.7.1
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 Environment set-up generated in /Users/user/NSO-5.7.1/ncsrc
INFO NSO installation script finished
INFO Found and unpacked corresponding NETSIM_PACKAGE
INFO NCS installation complete
```

Go to the installation folder and source NSO

```
user@COMPUTER # % cd ~/NSO-5.7.1
user@COMPUTER NSO-5.7.1 % source ncsrc
```

Execute the NSO Setup script

```
user@COMPUTER NSO-5.7.1 % ncs-setup --dest nso-instance
```

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

```
user@COMPUTER nso-instance % ls
README.ncs    logs      ncs-cdb      ncs.conf packages scripts
state
user@COMPUTER nso-instance % ncs
```

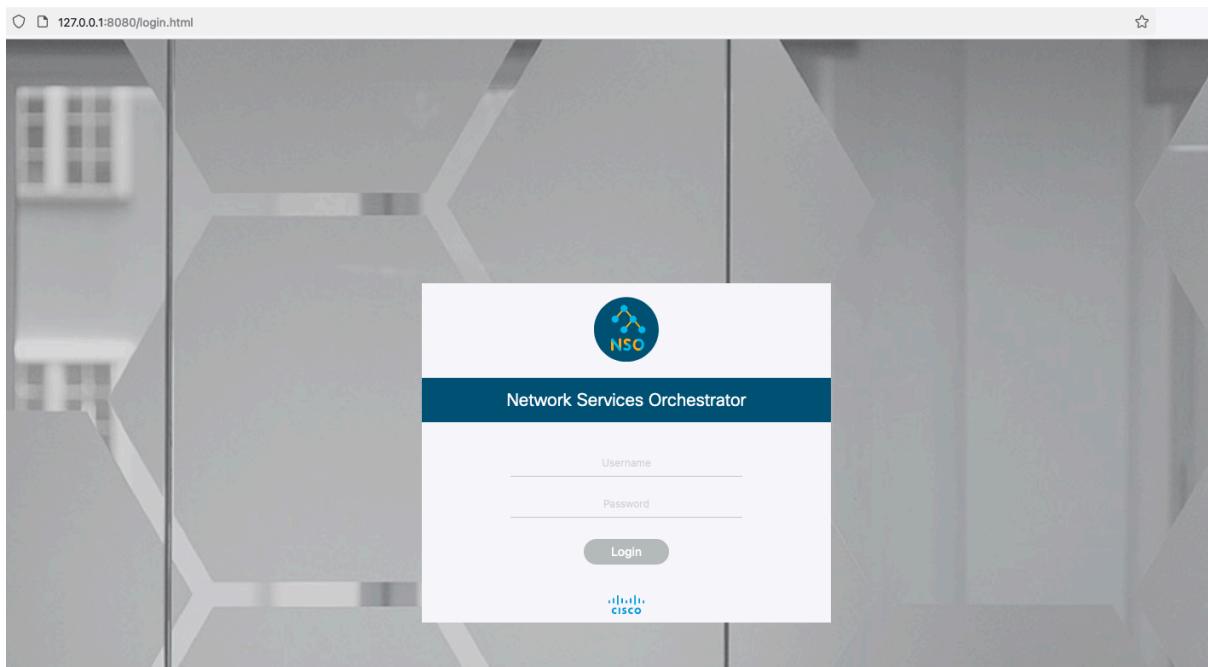
Check if NSO is running

```
user@COMPUTER 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)



CONFIGURATION

- Commit manager**: See the status of your current transaction and commit your changes.
- Configuration editor**: Access the data models that are loaded in NSO.
- Device manager**: Find, synchronize and group your devices. Monitor connectivity status and access the configuration data.
- Service manager**: Find and synchronize your services. See deployment status and access the service configuration data.

MONITORING

- Alarm manager**: Monitor number and status of alarms.
- Dashboard**: Monitor number of devices, user sessions and service instances. Overview service progress monitoring.
- High availability cluster**: Monitor the states of the High Availability cluster nodes. NEW

Go to the “Configuration Editor”

PACKAGES	MODULES
aac:aaa	nacm:nacm
aac:alias	ncm:netconf-state
aac:session	ncs:cluster
aac:user	ncs:compliance
al:alarms	ncs:customers
last:last-logins	ncs:devices
	ncs:high-availability
	ncs:java-vm
	ncs:packages
	ncs:python-vm
	ncs:service-progress-monitoring
	ncs:services
	ncs:side-effect-queue
	ncs:smart-license
	ncs:snmp-notification-receiver
	ncs:software
	ncs:sshd
	ncs:zombies
	rmon:restconf-state
	scheduler:scheduler
	snmp:snmp
	tftp:policy
	tnm:ncs-state
	tls:tls
	webui: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-5.7-cisco-iosxr-7.38.3.signed.bin”

```
user@COMPUTER # % sh ncs-5.7-cisco-iosxr-7.38.3.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-5.7-cisco-iosxr-
7.38.3.tar.gz using tailf.cer
```

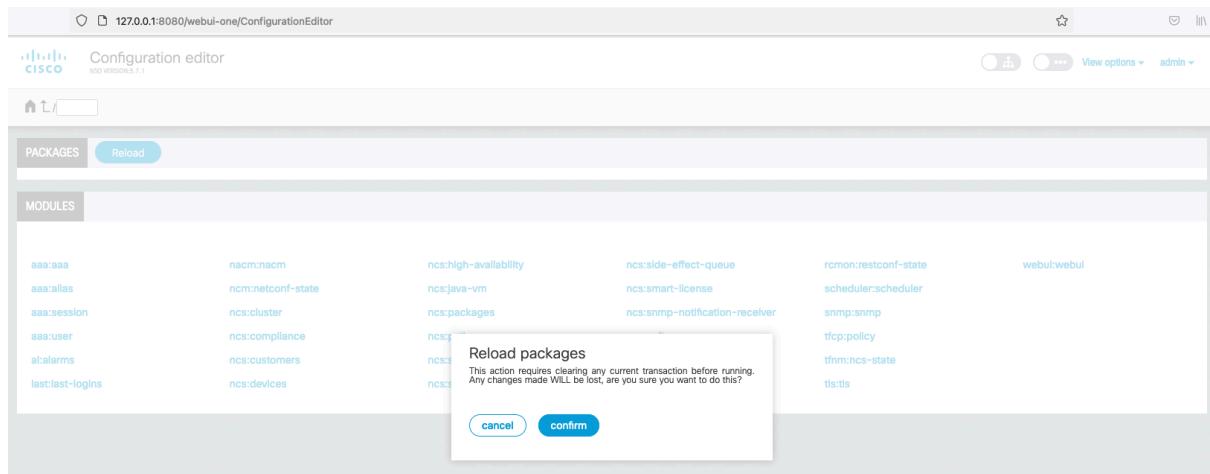
Extract the “.tar.gz” file

```
user@COMPUTER # % tar -zxvf ncs-5.7-cisco-iosxr-7.38.3.tar.gz
x cisco-iosxr-cli-7.38/
x cisco-iosxr-cli-7.38/package-meta-data.xml
x cisco-iosxr-cli-7.38/netsim/
x cisco-iosxr-cli-7.38/netsim/confd.i.ccl
x cisco-iosxr-cli-7.38/netsim/confd.conf.netsim.strict
x cisco-iosxr-cli-7.38/netsim/show_debug.sh
x cisco-iosxr-cli-7.38/netsim/confd.c.ccl
x cisco-iosxr-cli-7.38/netsim/Makefile
x cisco-iosxr-cli-7.38/netsim/populate.sh
x cisco-iosxr-cli-7.38/netsim/show_version.sh
x cisco-iosxr-cli-7.38/netsim/exec_command.sh
x cisco-iosxr-cli-7.38/netsim/confd.i.cli
x cisco-iosxr-cli-7.38/netsim/aaa.fxs
x cisco-iosxr-cli-7.38/netsim/show_interfaces.sh
x cisco-iosxr-cli-7.38/netsim/tailf-ned-cisco-ios-xr.fxs
x cisco-iosxr-cli-7.38/netsim/start.sh
...
x cisco-iosxr-cli-7.38/README
x cisco-iosxr-cli-7.38/CHANGES
x cisco-iosxr-cli-7.38/private-jar/
x cisco-iosxr-cli-7.38/private-jar/nedcom.jar
x cisco-iosxr-cli-7.38/private-jar/iosxr.jar
x cisco-iosxr-cli-7.38/private-jar/cisco-serializer.2.7.2b.jar
x cisco-iosxr-cli-7.38/private-jar/cisco-xalan.2.7.2b.jar
x cisco-iosxr-cli-7.38/private-jar/nanojson-1.2.jar
```

Copy the extracted directory to “NSO-INSTANCE/packages” folder

```
user@COMPUTER # % cp -r cisco-iosxr-cli-7.38 ~/NSO-5.7.1/nso-
instance/packages
```

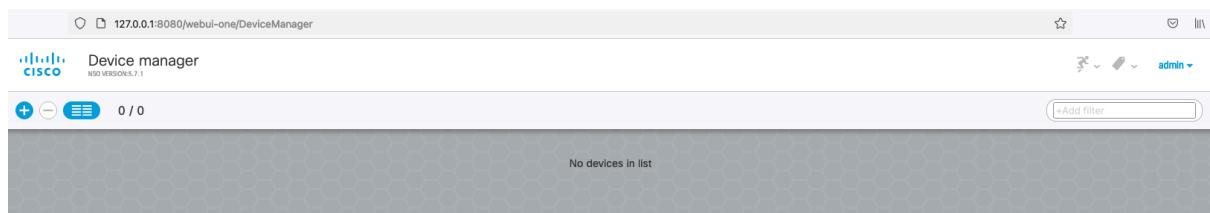
Go back to NSO WEB UI and Click on “Reload” packages button



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 Use Netsim's

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.

```
user@COMPUTER nso-instance % ncs-netsim create-network
packages/cisco-iosxr-cli-7.38/ 4 ios-xr-
DEVICE ios-xr-0 CREATED
DEVICE ios-xr-1 CREATED
DEVICE ios-xr-2 CREATED
DEVICE ios-xr-3 CREATED
```

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

```
user@COMPUTER nso-instance % ncs-netsim list
ncs-netsim list for /Users/user/NSO-5.7.1/nso-instance/netsim

name=ios-xr-0 netconf=12022 snmp=11022 ipc=5010 cli=10022
dir=/Users/user/NSO-5.7.1/nso-instance/netsim/ios-xr-/ios-xr-0
name=ios-xr-1 netconf=12023 snmp=11023 ipc=5011 cli=10023
dir=/Users/user/NSO-5.7.1/nso-instance/netsim/ios-xr-/ios-xr-1
name=ios-xr-2 netconf=12024 snmp=11024 ipc=5012 cli=10024
dir=/Users/user/NSO-5.7.1/nso-instance/netsim/ios-xr-/ios-xr-2
name=ios-xr-3 netconf=12025 snmp=11025 ipc=5013 cli=10025
dir=/Users/user/NSO-5.7.1/nso-instance/netsim/ios-xr-/ios-xr-3
```

Start the devices.

```
user@COMPUTER 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.

```
user@COMPUTER nso-instance % ncs-netsim cli-c ios-xr-0

admin connected from 127.0.0.1 using console on COMPUTER
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.
```

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
```

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

```
user@COMPUTER nso-instance % ncs-netsim ncs-xml-init > devices.xml
user@COMPUTER 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-cll-7.38...cisco-iosxr-cll-7.38	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration
<input type="checkbox"/>	ios-xr-1	127.0.0.1	10023	cisco-iosxr-cll-7.38...cisco-iosxr-cll-7.38	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration
<input type="checkbox"/>	ios-xr-2	127.0.0.1	10024	cisco-iosxr-cll-7.38...cisco-iosxr-cll-7.38	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration
<input type="checkbox"/>	ios-xr-3	127.0.0.1	10025	cisco-iosxr-cll-7.38...cisco-iosxr-cll-7.38	0	ping	connect	check-sync	sync-from	sync-to	compare-config		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-cll-7.38...cisco-iosxr-cll-7.38	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration
<input checked="" type="checkbox"/>	ios-xr-1	127.0.0.1	10023	cisco-iosxr-cll-7.38...cisco-iosxr-cll-7.38	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration
<input checked="" type="checkbox"/>	ios-xr-2	127.0.0.1	10024	cisco-iosxr-cll-7.38...cisco-iosxr-cll-7.38	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration
<input checked="" type="checkbox"/>	ios-xr-3	127.0.0.1	10025	cisco-iosxr-cll-7.38...cisco-iosxr-cll-7.38	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration

1.3 Configure Devices using NSO

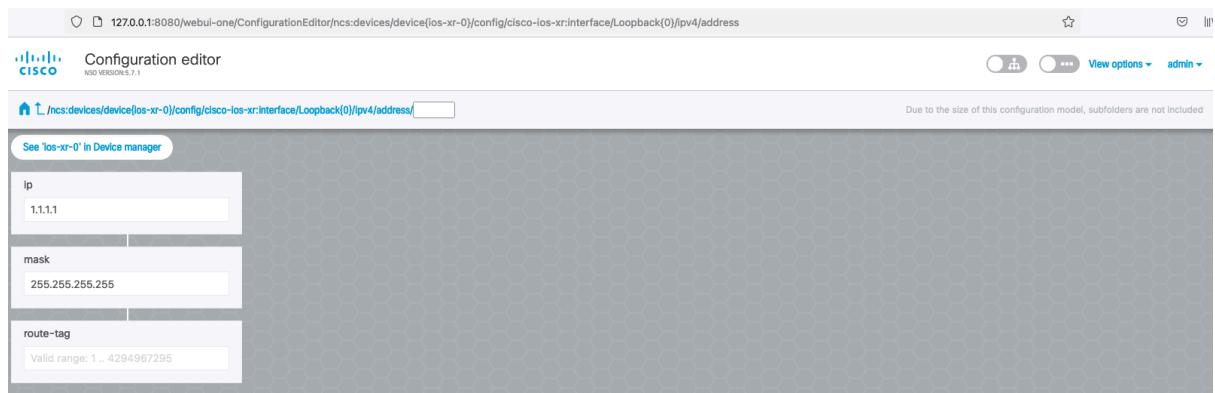
You can enter in ios-xr-0 to verify if the loopback 0 interface that we've configured is already there.

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.

The screenshot shows the Cisco Configuration Editor interface. The top navigation bar includes a search bar with "confi", a user icon, and a "View options" dropdown. The main content area displays the configuration for the device "ios-xr-0". The "config" section is currently selected, showing fields for "name" (ios-xr-0), "local-user", "description", "uthgroup" (set to "default"), and various timeout and behavior settings. A "service-list" section is present but empty. Below the configuration form are several blue rectangular buttons representing different configuration actions. The "address-choice" section at the bottom shows the device is set to "127.0.0.1".

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/



The screenshot shows the configuration path for the IP address of the Loopback 0 interface. The "ip" field is currently set to "1.1.1.1". The "mask" field is set to "255.255.255.255". The "route-tag" field has a validation message: "Valid range: 1..4294967295".

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

The screenshot shows the Configuration editor interface for NSO version 3.7.1. The URL is 127.0.0.1:8080/webui-one/ConfigurationEditor/ncs:devices/device{ios-xr-0}/config/cisco-ios-xr:interface/Loopback(0)/ipv4/address. The configuration being edited is for 'ios-xr-0'. The 'ip' field contains '1.1.1.2' with a green bar above it, indicating a candidate configuration. Other fields shown are 'mask' (255.255.255.255) and 'route-tag' (Valid range: 1 .. 4294967295).

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.

The screenshot shows the Commit manager interface for NSO version 3.7.1. The URL is 127.0.0.1:8080/webui-one/CommitManager/config. It displays a diff view between two configuration snapshots. The left snapshot shows a configuration for 'ios-xr-0' with an IP address of '1.1.1.1'. The right snapshot shows the same device with an IP address of '1.1.1.2'. The 'native config' tab is selected, and the 'Commit' button is visible at the top right.

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.

The screenshot shows the Commit manager interface after a successful commit. The URL is 127.0.0.1:8080/webui-one/CommitManager/config. A green banner at the top indicates 'Commit finished successfully at 14:21:30 with rollback id 10004'. The 'Done' button is visible at the top right. The configuration tabs show 'changes', 'errors', 'warnings', 'config', 'native config', and 'commit queue'. The message 'No configuration changes' is displayed below the tabs.

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 runn
admin
exit-admin-config
!
interface Loopback 0
no shutdown
```

```
 ipv4 address 1.1.1.2 255.255.255.255
exit
```

1.4 Use 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.

The screenshot shows the Cisco Commit Manager interface. In the center, a modal window titled "Load/save transaction data" is open. It displays a list of transactions:

- Id: 10004 - 2022-06-06 14:21:29 - admin / webui
- Id: 10003 - 2022-06-06 14:14:34 - admin / webui
- Id: 10002 - 2022-06-06 13:58:57 - system / system
- Id: 10001 - 2022-06-06 13:33:34 - system / system

Below the list, there is a checkbox labeled "Selective". The configuration code shown is:

```
# Created by: admin
# Date: 2022-06-06 14:21:29
# Via: webui
# Type: delta
# Label:
# Comment:
# No: 10004
# TransactionId: 69
# Hostname: JOMIRA-M-M3LA
ncs:devices {
    ncs:device ios-xr-0 {
        ncs:config {
            cisco-ios-xr:interface {
                cisco-ios-xr:Loopback 0 {
```

At the bottom of the dialog are two buttons: "cancel" and "load".

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, and we can see that the change will be reverted.

The screenshot shows the Cisco Commit Manager interface with the "native config" tab selected. A configuration diff is displayed, comparing two configurations. The changes are highlighted in red and green. The red highlights show the removal of the IP address assignment, and the green highlights show the addition of a new IP address assignment. The configuration code is as follows:

```
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 }
```

On the right side of the diff, the configuration after the revert is shown:

```
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 }
```

At the top right of the configuration area, there is a "Commit" button.

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

```
ios-xr-0# show run
```

```

admin
  exit-admin-config
!
interface Loopback 0
  no shutdown
  ipv4 address 1.1.1.1 255.255.255.255
exit

```

1.5 Use 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

```

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

```

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-7.38...cisco-iosxr-cll-7.38	0	ping	connect	check-sync	sync-from	sync-to	compare-config	!	configuration
ios-xr-1	127.0.0.1	10023	cisco-iosxr-cll-7.38...cisco-iosxr-cll-7.38	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration
ios-xr-2	127.0.0.1	10024	cisco-iosxr-cll-7.38...cisco-iosxr-cll-7.38	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration
ios-xr-3	127.0.0.1	10025	cisco-iosxr-cll-7.38...cisco-iosxr-cll-7.38	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
<input type="checkbox"/>	ios-xr-0	127.0.0.1	10022	cisco-iosxr-cli-7.38...cisco-iosxr-cli-7.38	0	ping	connect	check-sync	sync-from	sync-to

Action: check-sync Performed: 2022-06-06 14:46:03 Status: completed

Result: device was not in sync

out-of-sync
got: b78d090ac5cbf8e4d09cec8d8184235b expected: 0e15655ce829290701ae616e2f72eee7

If we click on Compare-config we can see exactly what was done out of NSO

Device manager										
NSO VERSION 5.7.1										
+ - Add filter										
	name	address	port	type	services	ping	connect	check-sync	sync-from	sync-to
<input type="checkbox"/>	ios-xr-0	127.0.0.1	10022	cisco-iosxr-cli-7.38...cisco-iosxr-cli-7.38	0	ping	connect	check-sync	sync-from	sync-to
<input type="checkbox"/>	ios-xr-1	127.0.0.1	10023	cisco-iosxr-cli-7.38...cisco-iosxr-cli-7.38	0	ping	connect	check-sync	sync-from	sync-to
<input type="checkbox"/>	ios-xr-2	127.0.0.1	10024	cisco-iosxr-cli-7.38...cisco-iosxr-cli-7.38	0	ping	connect	check-sync	sync-from	sync-to
<input type="checkbox"/>	ios-xr-3	127.0.0.1	10025	cisco-iosxr-cli-7.38...cisco-iosxr-cli-7.38	0	ping	connect	check-sync	sync-from	sync-to

Device manager										
NSO VERSION 5.7.1										
+ - Add filter										
	name	address	port	type	services	ping	connect	check-sync	sync-from	sync-to
<input type="checkbox"/>	ios-xr-0	127.0.0.1	10022	cisco-iosxr-cli-7.38...cisco-iosxr-cli-7.38	0	ping	connect	check-sync	sync-from	sync-to

Action: compare-config Performed: 2022-06-06 14:49:12 Status: completed

Result: CDB and device config does not match

```

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

	name	address	port	type	services	ping	connect	check-sync	sync-from	sync-to
<input type="checkbox"/>	ios-xr-1	127.0.0.1	10023	cisco-iosxr-cli-7.38...cisco-iosxr-cli-7.38	0	ping	connect	check-sync	sync-from	sync-to
<input type="checkbox"/>	ios-xr-2	127.0.0.1	10024	cisco-iosxr-cli-7.38...cisco-iosxr-cli-7.38	0	ping	connect	check-sync	sync-from	sync-to
<input type="checkbox"/>	ios-xr-3	127.0.0.1	10025	cisco-iosxr-cli-7.38...cisco-iosxr-cli-7.38	0	ping	connect	check-sync	sync-from	sync-to

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-cli-7.38...cisco-iosxr-cli-7.38	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	
Action: sync-to Performed: 2022-06-06 14:54:03 Status: completed													
Result: config was synced from device													
true													
	ios-xr-1	127.0.0.1	10023	cisco-iosxr-cli-7.38...cisco-iosxr-cli-7.38	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	
	ios-xr-2	127.0.0.1	10024	cisco-iosxr-cli-7.38...cisco-iosxr-cli-7.38	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	
	ios-xr-3	127.0.0.1	10025	cisco-iosxr-cli-7.38...cisco-iosxr-cli-7.38	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-cli-7.38...cisco-iosxr-cli-7.38	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	
	ios-xr-1	127.0.0.1	10023	cisco-iosxr-cli-7.38...cisco-iosxr-cli-7.38	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	
	ios-xr-2	127.0.0.1	10024	cisco-iosxr-cli-7.38...cisco-iosxr-cli-7.38	0	ping	connect	check-sync	sync-from	sync-to	compare-config	configuration	
	ios-xr-3	127.0.0.1	10025	cisco-iosxr-cli-7.38...cisco-iosxr-cli-7.38	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# 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 text "Configuration editor". Below the header, there are tabs for "PACKAGES" and "MODULES". The "MODULES" tab is currently selected. In the main content area, a list of modules is displayed in a table format:

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

In the Device-Group click on the Plus button

The screenshot shows the Cisco Configuration editor interface. At the top, there are several blue buttons for actions like connect, disconnect, fetch-ssh-host-keys, scp-to, partial-sync-from, sync-to, check-sync, migrate, apply-template, and clear-trace. Below these are sections for template, device-group, and mib-group. A modal window titled "Add new list item" is open, prompting for a "name" which is currently empty. There are "cancel" and "confirm" buttons at the bottom of the modal.

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 the Cisco Configuration editor interface. It displays a "device-group" section with a "name" field containing "ios-xr-devices". Below this are sections for device-name, member, and ned-id. The device-name section has a "connect" button and a "device-group" button. The member section has a "device-group" button. The ned-id section is empty. On the left, there are fields for location and alarm-summary.

The screenshot shows the Cisco Configuration editor interface. It displays a "device-group" section with a "name" field containing "ios-xr-devices". Below this are sections for device-name, member, and ned-id. The device-name section has a "connect" button and a "device-group" button. The member section has a "device-group" button. The ned-id section is empty. On the left, there are fields for location and alarm-summary. A modal window titled "Add new list item" is open, prompting for a "device-name" which is currently empty. A dropdown menu shows options: "ios-xr-0", "ios-xr-1", "ios-xr-2", "ios-xr-3", and "Source".

This should be the result.

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.

Click on the NED to setup the configuration.

The screenshot shows the Cisco Configuration editor interface. At the top, there are navigation icons and a status bar indicating 'NSO VERSION:5.7.1'. Below the header, the URL is shown as [/ncs/devices/template{set-dns-and-vtp-off}/](#). A message at the top right states 'Due to the size of this configuration model, subfolders are not included'. The main content area displays a configuration template. A sidebar on the left lists sections: 'name', 'set-dns-and-vtp-off', 'ned-id', and 'config/'. The 'ned-id' section is currently active, showing a list of items: 'id' and 'cisco-iosxr-cli-7.38:cisco-iosxr-cli-7.38'. There are also standard UI controls like copy, paste, and delete buttons.

Now click on the config (inside the NED)

This screenshot shows the 'config/' section of the configuration template. It contains a single entry for 'id': 'cisco-iosxr-cli-7.38:cisco-iosxr-cli-7.38'. Below this entry is a blue 'copy' button. The interface is consistent with the previous screenshot, featuring the same header, sidebar, and overall design.

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.

This screenshot shows the 'cisco-ios-xr:vtp' configuration section. Under the 'mode' field, the value 'off' is entered. The interface includes the standard Cisco Configuration editor header, sidebar, and URL path.

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

The screenshot shows the Configuration Editor interface for a Cisco IOS-XR device. The URL is [/ncs/devices/template{set-dns-and-vtp-off}/ned-id\[cisco-iosxr-cl-7.38:cisco-iosxr-cl-7.38\]/config/cisco-ios-xr:domain/](#). The configuration template is defined as follows:

```

devices {
    template set-dns-and-vtp-off {
        ned-id cisco-iosxr-cl-7.38 {
            config {
                domain {
                    name cisco.com;
                }
                vtp {
                    mode off;
                }
            }
        }
    }
    device-group ios-xr-devices {
        device-name [ ios-xr-0 ios-xr-1 ios-xr-2 ios-xr-3 ];
    }
}

```

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 Commit manager interface. The configuration diff section highlights the newly created device-group:

```

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

```

```

devices {
    template set-dns-and-vtp-off {
        ned-id cisco-iosxr-cl-7.38 {
            config {
                domain {
                    name cisco.com;
                }
                vtp {
                    mode off;
                }
            }
        }
    }
    device-group ios-xr-devices {
        device-name [ ios-xr-0 ios-xr-1 ios-xr-2 ios-xr-3 ];
    }
}

```

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”.

The screenshot shows the Configuration editor interface for the “ncs:devices” module. The “device-group” section shows the “ios-xr-devices” group selected:

```

device-group {
    name ios-xr-devices;
}

```

We click on the blue button “Apply-Template”

The screenshot shows the Cisco Configuration editor interface for a device group named "ios-xr-devices". The "device-name" section lists five devices: "ios-xr-0", "ios-xr-1", "ios-xr-2", "ios-xr-3", and "ios-xr-4". The "device-group" section is currently empty. The "member" section lists four members: "ios-xr-0", "ios-xr-1", "ios-xr-2", and "ios-xr-3". Below these sections are several action buttons: "connect", "sync-to", "sync-from", "check-sync", "check-yang-modules", "apply-template", and "fetch-ssh-host-keys". There are also fields for "ned-id" and "location/".

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

The screenshot shows the Cisco Configuration editor interface for applying a template. The "template-name*" field is set to "set-dns-and-vtp-off". The "Source" section contains "no-revision-drop", "no-networking", and "accept-empty-capabilities". The "wait-device" section is empty. The "choice-sync-check" section contains "no-overwrite" and "no-out-of-sync-check". The "choice-lsa" section contains "no-overwrite".

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.

The screenshot shows the 'Commit manager' interface with the URL [/ncs/devices/device-group/ios-xr-devices/apply-template/](#). At the top, there's a 'choice-unlock-id' section with an 'id' field containing 'tag'. Below it is a table of results for applying a template to three devices:

apply-template-result/device	apply-template-result/result
ios-xr-0	ok
ios-xr-1	ok
ios-xr-2	ok

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.

The screenshot shows the 'Commit manager' interface with the URL [/ncs/devices/device-group/ios-xr-devices/commit/](#). It displays the configuration for three devices (ios-xr-0, ios-xr-1, ios-xr-2) with identical settings for 'domain' and 'vtp'.

```

1 devices {
2     device ios-xr-0 {
3         config {
4             domain {
5                 name cisco.com;
6                 # first
7                 name-server 2.2.2.2;
8             }
9             vtp {
10                 mode off;
11             }
12         }
13     }
14     device ios-xr-1 {
15         config {
16             domain {
17                 name cisco.com;
18                 # first
19                 name-server 2.2.2.2;
20             }
21             vtp {
22                 mode off;
23             }
24         }
25     }
26     device ios-xr-2 {
27         config {
28             domain {
29                 name cisco.com;
30                 # first
31                 name-server 2.2.2.2;
32             }
33             vtp {
34                 mode off;
35             }
36         }
37     }
38     device ios-xr-3 {
39         config {
40             domain {
41                 name cisco.com;
42                 # first
43                 name-server 2.2.2.2;
44             }
45             vtp {
46                 mode off;
47             }
48         }
49     }
50 }

```

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”

```
user@COMPUTER 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-5.4
cisco-nso-nc-5.5 cisco-nso-nc-5.6 cisco-nso-nc-5.7

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)

```

user@COMPUTER nso-instance % cd packages
user@COMPUTER packages % ncs-make-package --service-skeleton
template NTP

```

After you run the script, you will find this structure

```
✓ nso-instance
  > logs
  > ncs-cdb
  > netsim
  ✓ packages
    > cisco-iosxr-cli-7.38
  ✓ NTP
    ✓ src
      ✓ yang
        ≡ NTP.yang
      M Makefile
    ✓ templates
      NTP-template.xml
    > test
    N package-meta-data.xml
```

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 > ncspackage
1   ...
2     <name>NTP</name>
3     <package-version>1.0</package-version>
4     <description>Template-based NTP resource facing service</description>
5     <ncs-min-version>5.7</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 > ...
1   <config-template xmlns="http://tail-f.com/ns/config/1.0"
2     ...
3       | servicepoint="NTP">
4       <devices xmlns="http://tail-f.com/ns/ncs">
5         <device>
6           <!--
7               | Select the devices from some data structure in the service
8               | model. In this skeleton the devices are specified in a leaf-list.
9               | Select all devices in that leaf-list:
10              -->
11             <name>{/device}</name>
12             <config>
13               <!--
14                   | Add device-specific parameters here.
15                   | In this skeleton the service has a leaf "dummy"; use that
16                   | to set something on the device e.g.:
17                   <ip-address-on-device>{/dummy}</ip-address-on-device>
18               -->
19             </config>
20           </device>
21         </devices>
22     </config-template>
```

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

```

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

```

Our next step will be:

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

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

```

user@COMPUTER NTP % ncs_cli -Cu admin

User admin last logged in 2022-06-06T15:04:16.629449+00:00, to
COMPUTER, from 127.0.0.1 using webui-http
admin connected from 127.0.0.1 using console on COMPUTER
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 > config-template > devices > device > config > ntp
1   <config-template xmlns="http://tail-f.com/ns/config/1.0"
2     servicepoint="NTP">
3   <devices xmlns="http://tail-f.com/ns/ncs">
4     <device>
5       <!--
6         Select the devices from some data structure in the service
7         model. In this skeleton the devices are specified in a leaf-list.
8         Select all devices in that leaf-list:
9       -->
10      <name>{/device}</name>
11      <config>
12        <!--
13          Add device-specific parameters here.
14          In this skeleton the service has a leaf "dummy"; use that
15          to set something on the device e.g.:
16          <ip-address-on-device>{/dummy}</ip-address-on-device>
17        -->
18        <ntp xmlns="http://tail-f.com/ned/cisco-ios-xr">
19          <peer>
20            <address>192.168.22.33</address>
21          </peer>
22          <server>
23            <server-list>
24              <name>172.16.22.44</name>
25              <minpoll>8</minpoll>
26              <maxpoll>12</maxpoll>
27            </server-list>
28          </server>
29        </ntp>
30      </config>
31    </device>
32  </devices>
33 </config-template>

```

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

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

```

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.

```

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 { prefix inet; }
5   import tailf-ncs { prefix ncs; }
6   list NTP {
7     key name;
8     uses ncs:service-data;
9     ncs:servicepoint "NTP";
10    leaf name {
11      type string;
12    }
13    leaf-list device {
14      type leafref {
15        path "/ncs:devices/ncs:device/ncs:name";
16      }
17    }
18
19    leaf peer-address {
20      type inet:ipv4-address;
21      mandatory true;
22    }
23    leaf server-address {
24      type inet:ipv4-address;
25      mandatory true;
26    }
27    leaf minpoll {
28      type uint8;
29      default "8";
30    }
31    leaf maxpoll {
32      type uint8;
33      default "12";
34    }
35  }
36}

```

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

And apply the “make” command.

```

user@COMPUTER NTP % cd src
user@COMPUTER src % make
mkdir -p ../load-dir
/Users/user/NSO-5.7.1/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.

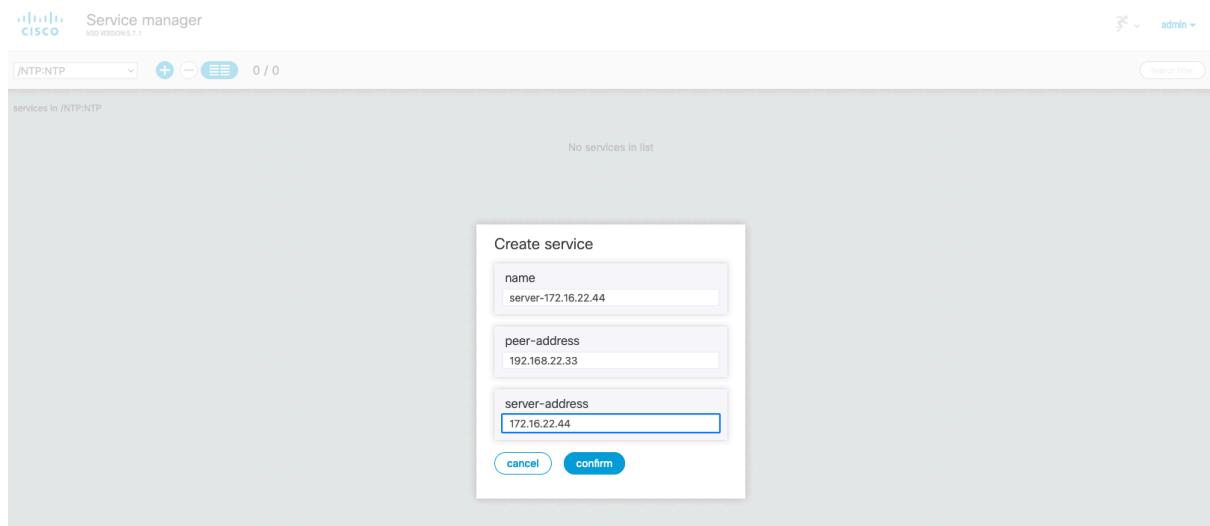
After the reload you will see the NTP service package that we've just created.

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

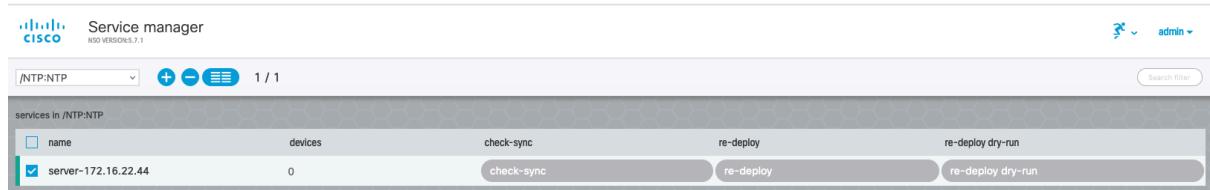
We select the NTP service that we've created

Then, we click on + button

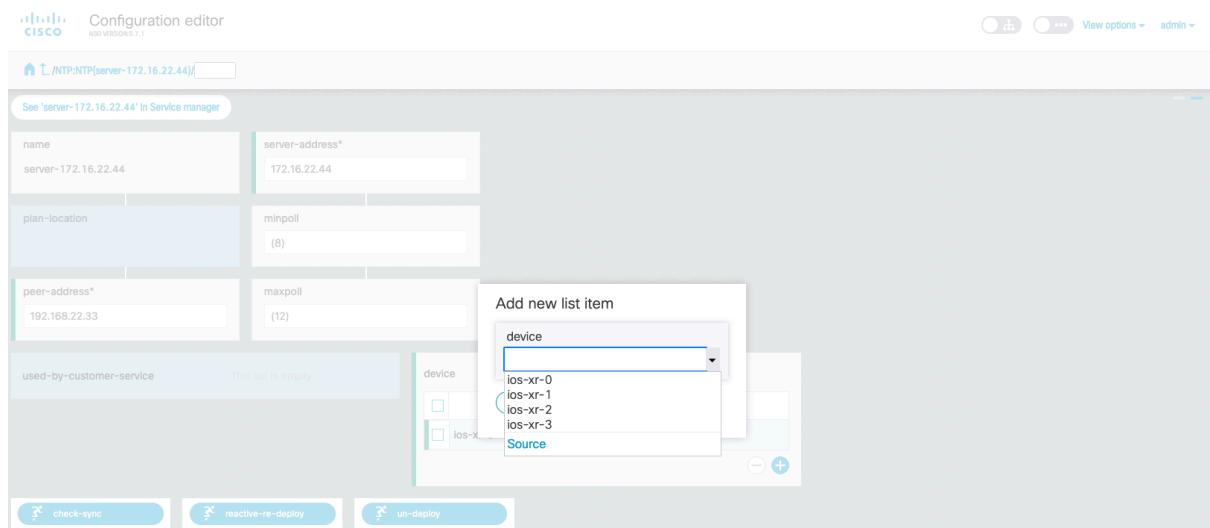
We should give a service name (help us identify the configs applied) and setup the mandatory arguments, in our case, the peer-address and server-address



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     devices {
7         device ios-xr-0 {
8             config {
9                 ntp {
10                     server {
11                         }
12                     }
13                 }
14             }
15         }
16         device ios-xr-1 {
17             config {
18                 ntp {
19                     server {
20                         }
21                     }
22                 }
23             }
24         }
25         device ios-xr-2 {
26             config {
27                 ntp {
28                     server {
29                         }
30                     }
31                 }
32             }
33         }
34         device ios-xr-3 {
35             config {
36                 ntp {
37                     server {
38                         }
39                     }
40                 }
41             }
42         }
43         device ios-xr-0 {
44             config {
45                 ntp {
46                     server {
47                         }
48                     }
49                 }
50             }
51         }
52         device ios-xr-1 {
53             config {
54                 ntp {
55                     server {
56                         }
57                     }
58                 }
59             }
60         }
61         device ios-xr-2 {
62             config {
63                 ntp {
64                     server {
65                         }
66                     }
67                 }
68             }
69         }
70         device ios-xr-3 {
71             config {
72                 ntp {
73                     server {
74                         }
75                     }
76                 }
77             }
78         }
79     }
80
81     +NTP server=172.16.22.44 {
82         device [ ios-xr-0 ios-xr-1 ios-xr-2 ios-xr-3 ];
83         peer-address 192.168.22.33;
84         server-address 172.16.22.44;
85     }
86
87     devices [
88         device ios-xr-0 {
89             config {
90                 ntp {
91                     peer 192.168.22.33 {
92                         server-list 172.16.22.44 {
93                             minpoll 8;
94                             maxpoll 12;
95                         }
96                     }
97                 }
98             }
99         }
100        device ios-xr-1 {
101            config {
102                ntp {
103                    peer 192.168.22.33 {
104                        server-list 172.16.22.44 {
105                            minpoll 8;
106                            maxpoll 12;
107                        }
108                    }
109                }
110            }
111        }
112        device ios-xr-2 {
113            config {
114                ntp {
115                    peer 192.168.22.33 {
116                        server-list 172.16.22.44 {
117                            minpoll 8;
118                            maxpoll 12;
119                        }
120                    }
121                }
122            }
123        }
124        device ios-xr-3 {
125            config {
126                ntp {
127                    peer 192.168.22.33 {
128                        server-list 172.16.22.44 {
129                            minpoll 8;
130                            maxpoll 12;
131                        }
132                    }
133                }
134            }
135        }
136    }
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589

```

C* Commit manager E Configuration A Alarm manager B Dashboard D Device manager S Service manager

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

/NTP:NTP				
	name	devices	check-sync	re-deploy
<input checked="" type="checkbox"/>	server=172.16.22.44	4 ▾	check-sync	re-deploy

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.



Configuration editor
NSO VERSION 3.7.1

See 'ios-xr-2' in Device manager

max-associations

master

update-calendar

trusted-key

This list is empty

Add list item → +

peer

<input checked="" type="checkbox"/> address	af
<input checked="" type="checkbox"/> 192.168.22.33	ipv4

+ - +

See 'ios-xr-2' in Device manager

server-list

<input checked="" type="checkbox"/> name	minpoll	maxpoll
<input checked="" type="checkbox"/> 172.16.22.44	8	12

vrf

This list is empty

Add list item → +

Go to the commit manager and press “Commit”

Commit manager
NSO VERSION 3.7.1

Current transaction (7 - webui-one) is VALID

Revert Load/Save Commit

changes errors warnings config native config commit queue

```

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 }
```

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

Service manager
NSO VERSION 3.7.1

/NTP:NTP

services in /NTP:NTP

name	devices	check-sync	re-deploy	re-deploy dry-run
<input checked="" type="checkbox"/> server-172.16.22.44	4 ▾	check-sync	re-deploy	re-deploy dry-run

Action: check-sync Performed: 2022-06-06 16:54:39 Status: completed

Result: service was not in sync
false

By clicking on Re-deploy Dry-Run

Action: re-deploy dry-run Performed: 2022-06-06 16:55:07 Status: completed

Result: service has changes

```

1 devices {
2   device ios-xr-2 {
3     config {
4       ntp {
5         server {
6           peer 192.168.22.33 {
7             minpoll 8;
8             maxpoll 12;
9           }
10          }
11        }
12      }
13    }
14  }
15 }
16 device ios-xr-3 {
17   config {
18     ntp {
19       server {
20         peer 192.168.22.33 {
21           minpoll 8;
22           maxpoll 12;
23         }
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: 2022-06-06 16:56:25 Status: completed

Result: service was re-deployed

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

```

ios-xr-2# show run
admin
exit-admin-config
!
domain name cisco.com
domain name-server 2.2.2.2
```

```
ntp
peer 192.168.22.33
server 172.16.22.44 minpoll 8 maxpoll 12
exit
vtp mode off
```

1.8 NSO APIs

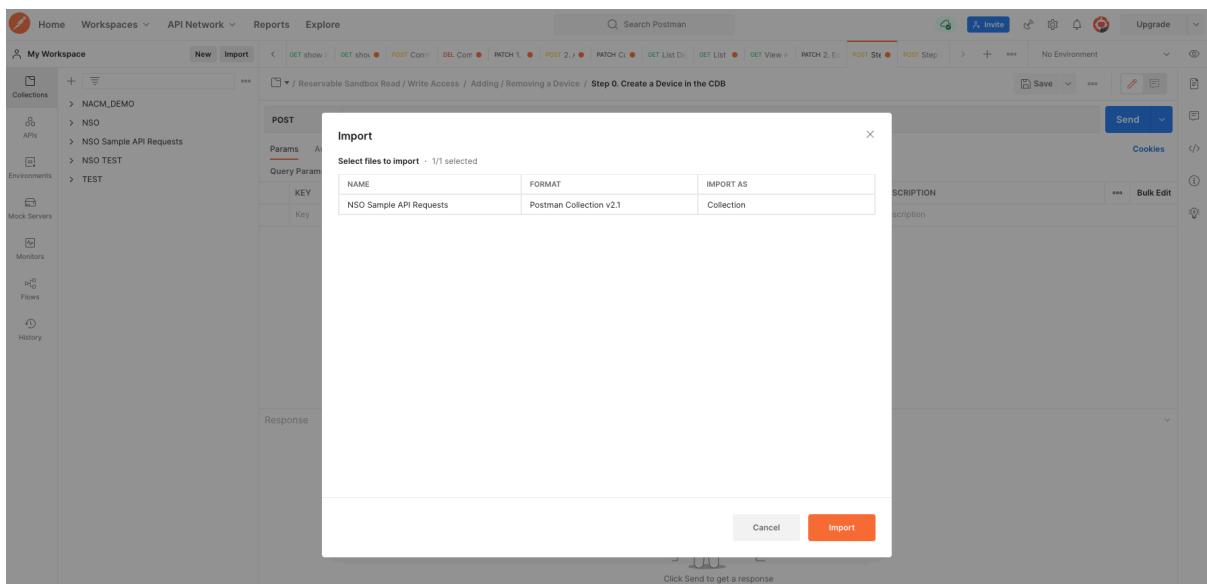
IN NSO you can interact via REST APIs as well.

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

You can download it here.

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

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



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

The screenshot shows the Postman application interface. The left sidebar is titled "My Workspace" and contains sections for Collections, APIs, Environments, Mock Servers, Monitors, Flows, and History. Under "Collections", there is a expanded section for "NSO Sample API Requests" which includes "Always-On Read Only Requests", "NSO Root of the App Hierarchy", "View Device Config", "NSO Device Groups", "View Existing NSO Service ...", "Operational Data", "Reservable Sandbox Read / ...", and "NSO Root of the App Hierarchy". The main workspace is titled "NSO Sample API Requests" and shows tabs for Authorization, Pre-request Script, Tests, and Variables. The "Variables" tab is selected, displaying a table of variables with their initial and current values. The table includes variables like NSO_IP, NSO_HTTP_PORT, username, password, sample_ios_device, sample_xr_device, sample_junos_device, PROTOCOL, and SAMPLE_IOS_DEVICE, along with a note to add a new variable.

VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ
NSO_IP	127.0.0.1	127.0.0.1
NSO_HTTP_PORT	8080	8080
username	admin	admin
password	admin	admin
sample_ios_device	ios-0	ios-0
sample_xr_device	ios-xr-0	ios-xr-0
sample_junos_device	junos0	junos0
PROTOCOL	http	http
SAMPLE_IOS_DEVICE	ios-xr-0	ios-xr-0

Here is an example of the Device Group request.

The screenshot shows the Postman application interface. The left sidebar contains navigation links: Home, Workspaces, API Network, Reports, Explore, Collections, APIs, Environments, Mock Servers, Monitors, Flows, and History. The main area displays the 'My Workspace' section. A collection named 'NSO Sample API Requests' is selected, containing several items like 'Always-On Read Only Requests', 'View Device Config', and 'NSO Device Groups'. Under 'NSO Device Groups', there is a sub-item 'List Device Groups'. The right side shows the details for this API call:

NSO Sample API Request | **GET show interfaces** | **GET List Device Groups**

Always-On Read Only Requests / NSO Device Groups / List Device Groups

GET `((PROTOCOL))://((NSO_IP)):((NSO_HTTP_PORT))/restconf/data/tailf-ncs:devices/device-group`

Headers (12)

Header	Value
Authorization	Basic e3t1c2VybmbFtZX19n7cGFzc3dvcmR9fQ==
Cache-Control	no-cache
Postman-Token	<calculated when request is sent>
Host	<calculated when request is sent>
User-Agent	PostmanRuntime/7.29.0
Accept	/*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
Content-Type	application/yang-data+json
Authorization	Basic YWRtaW46YWRtaW4=
Accept	application/yang-data+json

Body | **Cookies** | **Headers (12)** | **Test Results (0/1)**

Pretty | **Raw** | **Preview** | **Visualize** | **JSON** | **CSV**

```
1 "tailf-ncs:device-group": [
2   {
3     "name": "ios-xr-devices",
4     "device-name": [
5       "ios-xi-0",
6       "ios-xr-1",
7       "ios-xr-2",
8       "ios-xi-3"
9     ],
10    "member": [
11      "ios-xr-0",
12      "ios-xr-1",
13      "ios-xr-2",
14      "ios-xr-3"
15    ]
16  }
17]
```

How to get the RESTCONF paths ? The best way to get 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/authgroup default
/restconf/data/tailf-ncs:devices/device=ios-xr-0/device-
type/cli/ned-id cisco-iosxr-cli-7.38
/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

```

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/ns0-developer/rest-api-explorer>

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

Method	Endpoint	Description
GET	/restconf/data/devices	Retrieve data and meta-data.
HEAD	/restconf/data/devices	Retrieve just the headers without the response body.
OPTIONS	/restconf/data/devices	Discover methods supported by the server for a specific resource.
PATCH	/restconf/data/devices	Updates an existing resource.
POST	/restconf/data/devices	Create a data resource.
PUT	/restconf/data/devices	Create or replace 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. At the top, it says "Configuration editor" and "NSO VERSION 5.7.1". There are tabs for "PACKAGES" and "MODULES". Under "MODULES", the "aaa:aaa" module is selected. The configuration table includes columns for package names and their versions:

PACKAGE	VERSION
NTP	v1.0
cisco-ios-cl-6.79	v6.79
cisco-iosxr-cl-7.38	v7.38.3

Then we go to the authentication/users and we click on “+” button

The screenshot shows the "aaa:aaa/authentication/users/" page. It lists existing users: admin, oper, private, and public. A new user entry is being added:

name	uid	gid	password	ssh_keydir	homedir
admin	65534	65534	\$6\$kvW68pptz6Zne68k\$...JOyUlszETvxb3mUpCs9.	/var/ncs/homes/admin/.ssh	/var/ncs/homes/admin
oper	65534	65534	\$6\$SpM17Q5X9ORpmgE35\$..NLNobKXhv.tSDwSHyu1	/var/ncs/homes/oper/.ssh	/var/ncs/homes/oper
private	65534	65534	\$6\$	/var/ncs/homes/private/.ssh	/var/ncs/homes/private
public	65534	65534	\$6\$	/var/ncs/homes/public/.ssh	/var/ncs/homes/public

Let's create the user “read_user”

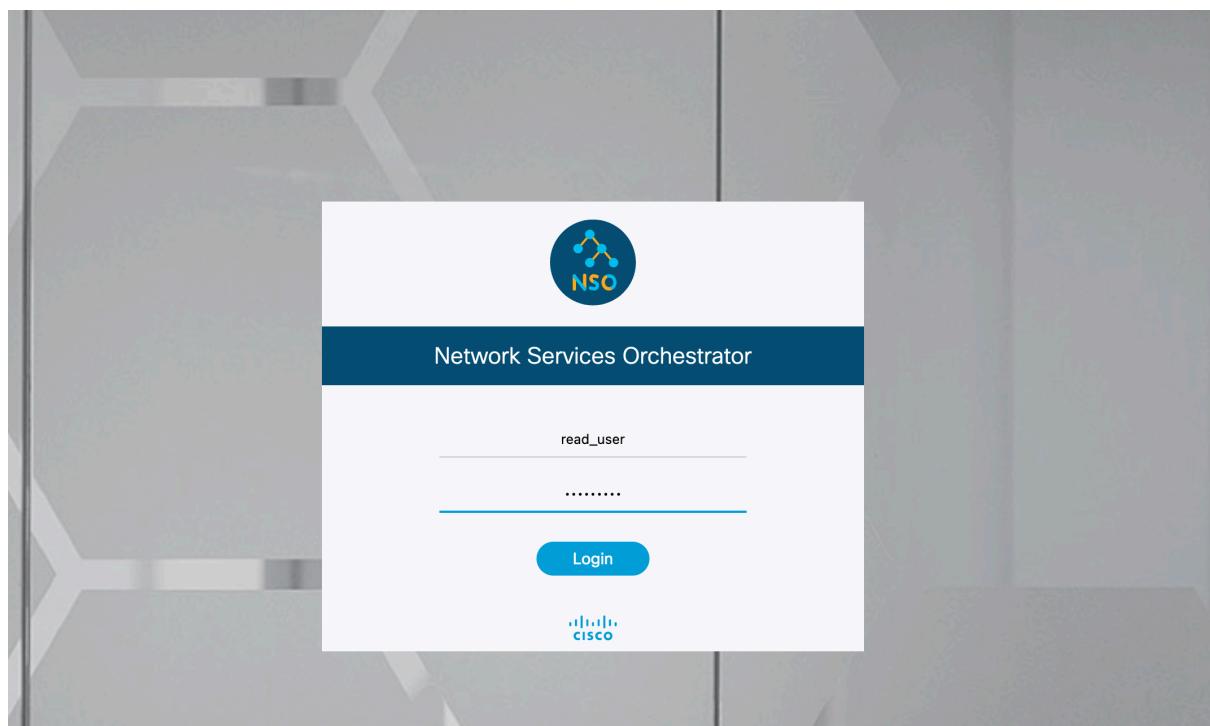
The screenshot shows the "aaa:aaa/authentication/users/" page with a modal dialog for creating a new user "read_user". The dialog fields are:

- name:** read_user
- uid:** 1
- gid:** 1
- password:** (redacted)
- ssh_keydir:** /var/ncs/homes/read_user/.ssh
- homedir:** /var/ncs/homes/read_user

After this we can “Commit”

The screenshot shows the Cisco Commit manager interface. At the top, it says "Commit manager" and "NSO VERSION 5.7.1". Below that, there are tabs for "changes", "errors", "warnings", "config", "native config", and "commit queue". The "changes" tab is selected, showing a list of configuration paths and their values. A modal dialog box is overlaid on the screen, asking "Commit changes to NSO?" with "cancel" and "Yes, commit" buttons.

And login to the new recently created user



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

The screenshot shows the Cisco Device manager interface. It lists several devices: "ios-clr-0", "ios-clr-1", "ios-xr-0", "ios-xr-1", "ios-xr-2", and "ios-xr-3". The "ios-clr-0" device is selected. In the table, under the "ping" column, the first row for "ios-clr-0" shows an action of "ping", performed on "2022-06-10 15:21:05", with a status of "error". A note below says "Method failed reason: Resource authgroup for read_user doesn't exist".

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

Component	Module
NTP:NTP	a:alarms
aaa:aaa	last:last-logins
aaa:alias	ncm:netconf-state
aaa:session	ncs:cluster
aaa:user	ncs:compliance
	ncs:customers
	ncs:devices
	ncs:high-availability
	ncs:java-vm
	ncs:packages
	ncs:python-vm
	ncs:services
	ncs:side-effect-queue
	ncs:smart-license
	ncs:snmp-notification-receiver
	ncs:software
	ncs:sshd
	ncs:zombies
	rmon:restconf-state
	scheduler:scheduler
	snmp:snmp
	tfcpc:policy
	tfnmc:ncs-state
	webui:webui

Then we select authgroup “default”

group
<input type="checkbox"/> name
<input checked="" type="checkbox"/> default

snmp-group
<input type="checkbox"/> name
<input checked="" type="checkbox"/> default

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. The URL in the address bar is `/ncs/devices/authgroups/group/default/`. The main pane displays the configuration for the 'default' authgroup, which has no name assigned. Below this, the 'umap' section is visible, containing three entries: 'local-user', 'admin', and 'oper'. A modal dialog box at the bottom is titled 'Add new list item' and contains a single input field with the value 'read_user'. There are 'cancel' and 'confirm' buttons at the bottom of the dialog.

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

This screenshot shows the same Cisco Configuration editor interface as the previous one, but with a modal dialog box open. The dialog is titled 'Add new list item' and contains a single input field with the value 'read_user'. At the bottom of the dialog are 'cancel' and 'confirm' buttons. The background shows the 'umap' section of the 'default' authgroup configuration, where the 'local-user' entry is currently listed.

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

The screenshot shows the Cisco NSO Configuration editor interface. A new user named 'read_user' is being created under the 'local-user' group. In the 'remote-user' section, the 'remote-name' field is set to 'admin'. Under 'remote-auth', the 'remote-password' field contains '*****'. The 'configuration' tab is selected.

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

name	address	port	type	services	ping	connect	check-sync	sync-from	sync-to	compare-config	alarm	configuration
ios-cli-0	127.0.0.1	10026	cisco-ios-cll-6.79:cisco-ios-cll-6.79	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration
ios-cli-1	127.0.0.1	10027	cisco-ios-cll-6.79:cisco-ios-cll-6.79	0	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration
ios-xr-0	127.0.0.1	10022	cisco-iosxr-cll-7.38:cisco-iosxr-cll-7.38	1 ▾	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration
ios-xr-1	127.0.0.1	10023	cisco-iosxr-cll-7.38:cisco-iosxr-cll-7.38	1 ▾	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration
ios-xr-2	127.0.0.1	10024	cisco-iosxr-cll-7.38:cisco-iosxr-cll-7.38	1 ▾	ping	connect	check-sync	sync-from	sync-to	compare-config		configuration
ios-xr-3	127.0.0.1	10025	cisco-iosxr-cll-7.38:cisco-iosxr-cll-7.38	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

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

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

Let's create the “read_group”

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

The screenshot shows the Cisco Configuration editor interface. In the main pane, there is a configuration for a group named "read_group" with a "gid" of 1. Below this, there is a list item section labeled "user-name" which is currently empty. A modal window titled "Add new list item" is open, showing a single input field "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 Cisco Configuration editor interface. In the main pane, there is a configuration for a rule-list named "read_rule". A modal window titled "Add new list item" is open, showing a single input field "name" with the value "read_rule". There are "cancel" and "confirm" buttons at the bottom of the modal.

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. In the main pane, there is a configuration for a rule-list named "read_rule". Under the "group" section, the "read_group" is listed. Below this, there is a "rule" section which is currently empty. There is also a "tacm:cmdrule" section which is also 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 Cisco Configuration editor interface. A modal window titled "Add new list item" is open, prompting for a "name" (set to "deny-sync-from") and an "action" (set to "deny"). The background shows a list of items under a "group" named "read_group".

In the rule we've to specify a path

The screenshot shows the configuration of a specific rule. The "path" field is set to "/devices/device/sync-from". Other fields include "action" (denied), "module-name" (*), and "rule-type" (notification).

This path is nothing more than an XPATH. To see how we can get the xpaths, we go back to CLI and we execute the command "show devices device | display xpath"

```
admin@ncs# show devices device | display xpath
/devices/device[name='ios-cli-0']/commit-queue/queue-length 0
/devices/device[name='ios-cli-0']/active-settings/connect-timeout 20
/devices/device[name='ios-cli-0']/active-settings/read-timeout 20
/devices/device[name='ios-cli-0']/active-settings/write-timeout 20
/devices/device[name='ios-cli-0']/active-settings/connect-timeout 20
/devices/device[name='ios-cli-0']/active-settings/read-timeout 20
/devices/device[name='ios-cli-0']/active-settings/write-timeout 20
...
...
```

This will show all devices path, if we want to see the paths for each device

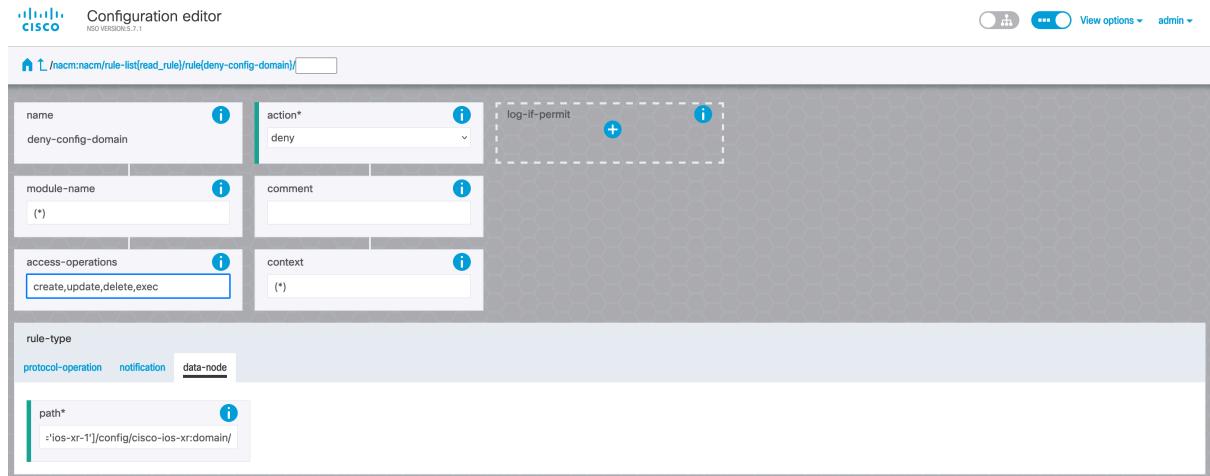
```
admin@ncs(config)# show full-configuration devices device ios-xr-1 | display xpath
/devices/device[name='ios-xr-1']/address 127.0.0.1
/devices/device[name='ios-xr-1']/port 10023
/devices/device[name='ios-xr-1']/ssh/host-key[algorithm='ssh-rsa']/key-data
"AAAAB3NzaC1yc2EAAAQABAAQgQC3e3MWufWvWFny7iZpmqmRnErM4BZq/gYl4QP
YjS+2\nF/TuFMA1XNxQjE8ewyy4VGVCwBnoBCF40kjCq2BxVR+h4qrJ/VjRONkHahFa7
axLh5pjvSCJ\ntg4ut5Vw1c02qyou6ItRvRbzsj+1vPBryClCOVQUoOs+YKev9qpoLvK
def3JTqo2Z5k46ifi\nuzk1X0Agwi0xVDUcs41KksdN0c7gZnz/rYV/Mt2gKU7oqA30k
ZQPDpW8dvwl3Ig8ninVK3h9\nuSvKPrD8iO7wMOidtLldw6xqwJkcACtx5/0io7wZ+Bk
WycBmhoS9QRwuhEG0rbqT+iwgJ7S\nnX0wSYF9k2/YVnM4Wa73zlkXnsnqeXdmbYfow
EFxjJPP7YG50BLnJErXfzBmsSmGgzR1aCzs\nnW8qCET8JdkleAL+R74ZsFbY45H9+1xa
b0djFJkN4kH5cBL9FjOpCzMJzNlox+4WY2UJb9TZw\nn0u31BCWQ+8F+ODPkPeUg31098
IkeYFvDqODfEgM="
/devices/device[name='ios-xr-1']/ssh/host-key[algorithm='ssh-ed25519']/key-data
AAAC3NzaC1zDI1NTE5AAAAIN+PlB/juJTHnDrFBAMjeT1EEdiC42EEd7RBLGHOfDNI
/devices/device[name='ios-xr-1']/authgroup default
/devices/device[name='ios-xr-1']/device-type/cli/ned-id cisco-iosxr-cl-7.38
/devices/device[name='ios-xr-1']/ssh-algorithms/public-key [ ssh-ed25519 ssh-rsa ]
/devices/device[name='ios-xr-1']/state/admin-state unlocked
/devices/device[name='ios-xr-1']/config/cisco-ios-xr:domain/name cisco.com
/devices/device[name='ios-xr-1']/config/cisco-ios-xr:domain/name-server[address='2.2.2.2']
/devices/device[name='ios-xr-1']/config/cisco-ios-xr:ntp/peer[address='192.168.22.33']
/devices/device[name='ios-xr-1']/config/cisco-ios-xr:ntp/server/server-list[name='172.16.22.44']/minpoll 8
/devices/device[name='ios-xr-1']/config/cisco-ios-xr:ntp/server/server-list[name='172.16.22.44']/maxpoll 12
/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 block all the operations with the exception of reading.

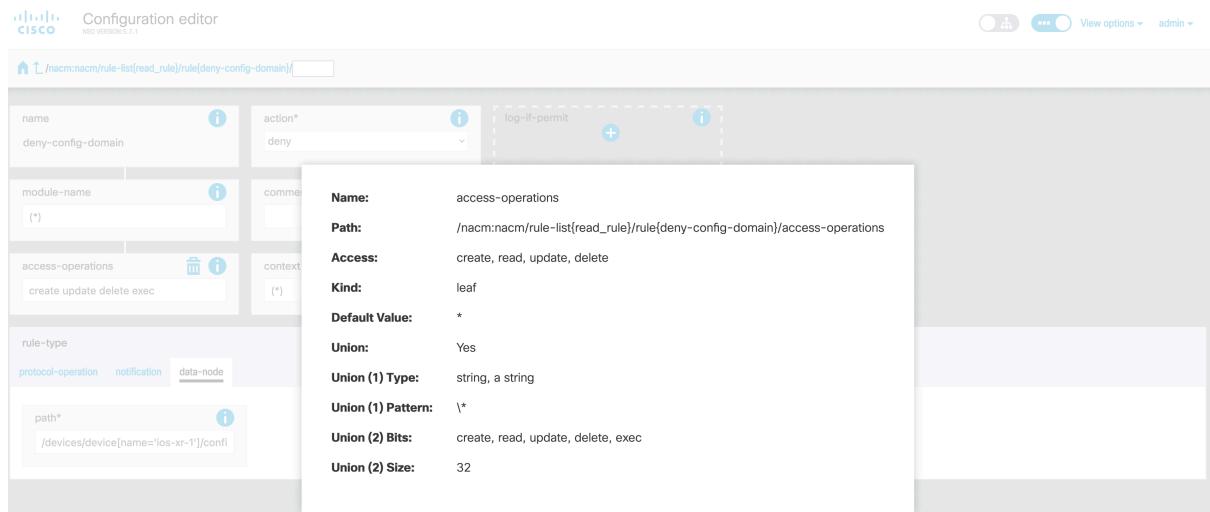
We create the rule “deny-config-domain”

rule	name	module-name	access-operations	action	context
name	*	*	*	deny	*
deny-sync-from	*	*	*	deny	*
deny-config-domain	*	*	*	deny	*

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



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



After this, we commit.

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

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

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

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

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

The screenshot shows the Cisco Configuration editor interface for device 'ios-xr-0'. The top bar includes the Cisco logo, 'Configuration editor', 'NSO VERSION 3.1', and 'View options' with 'read_user' selected. The main area displays the configuration path: /ncs:devices/device[ios-xr-0]/config/cisco-ios-xr:domain/. The configuration sections shown are:

- name**: A single entry 'cisco.com'.
- list**: A section labeled 'This list is empty' with an 'Add list item → +' button.
- name-server**: A list containing 'address' and '2.2.2.2'.
- vrf**: A section labeled 'This list is empty' with an 'Add list item → +' button.
- ipv4/**, **ipv6/**, **lookup/**: These sections are currently empty.

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/"

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.

The screenshot shows the Cisco Configuration editor interface for device 'ios-xr-0' after changing the configuration path to /devices/device/config/cisco-ios-xr:domain/. The interface is identical to the previous screenshot, but the configuration sections are now empty, indicating that changes cannot be made through this user path.